ECE 232E: Large-Scale Social and Complex Networks: Models and Algorithms

Project 1: Random Graphs and Random Walks

Akshay Sharma (504946035)

Anoosha Sagar (605028604)

Nikhil Thakur(804946345)

Rahul Dhavalikar (205024839)

I. 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.

Ans: The ER model is a method for generating random graphs. The G(n, p) formulation in this model has 2 parameters, 'n' and 'p'. Here 'n' is the number of vertices of the graph and 'p' is the edge probability. For each pair of distinct vertices, 'v' and 'w', 'p' is the probability that the edge (v,w) is present. In many cases, p will be a function of n such as p = d/n. ***The distribution of the degree of any particular vertex is binomial.***

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

For large n, this can be approximated with Poisson distribution:

$$P(deg(v) = k) = \frac{z^k e^{-z}}{k!}$$

, where z is the average degree.

Below, we have given the calculated and theoretical mean and variance of degree distribution for different values of 'p'. We observe that the mean and variance increase as we increase the value of 'p'.

For a random variable X, the expectation is

$$E[X] = \sum_{x \in dom(X)} x Pr[X = x]$$

In ER model, X is an edge and for each edge we multiply the probability that it will occur by the number of the edge, and take 1 the sum for all edges. Thus the expected mean degree is:

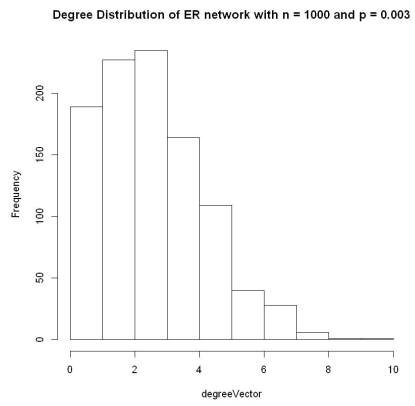
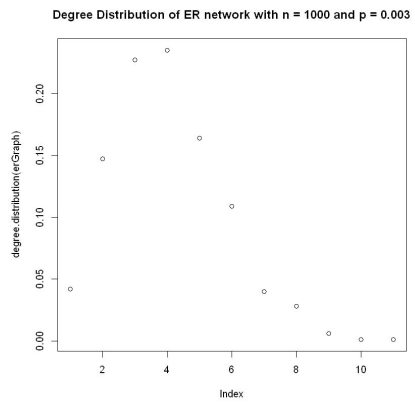
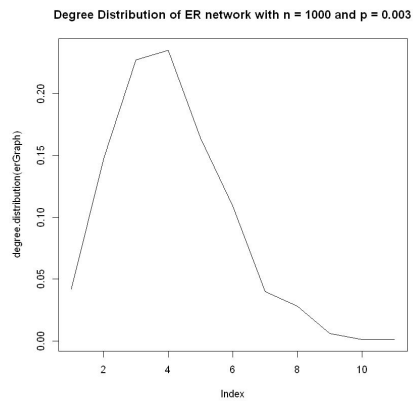
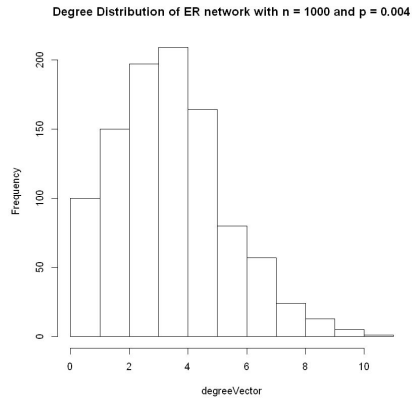
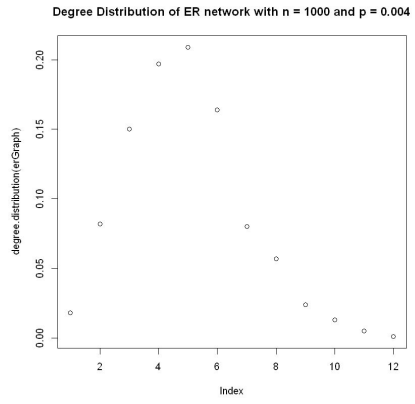
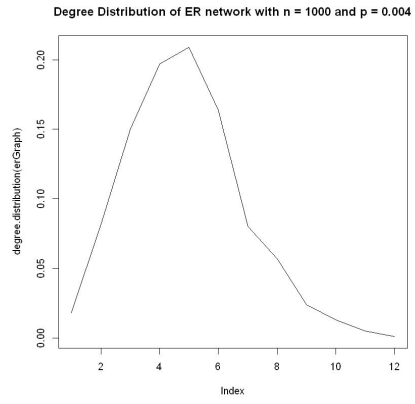
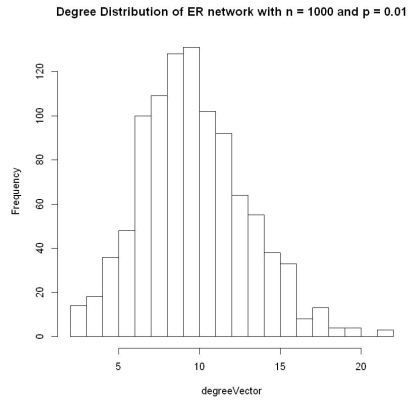
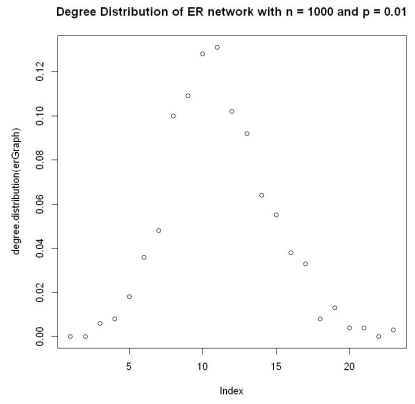
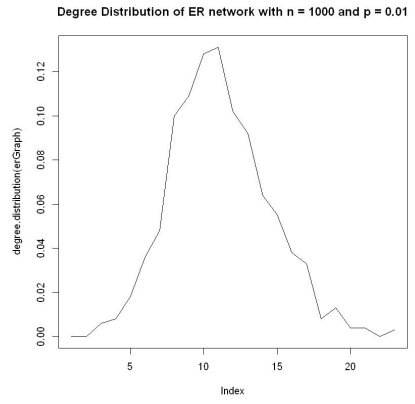
$$\sum_{d=0}^{n} d\binom{n}{d} p^d (1-p)^{n-d} = np$$

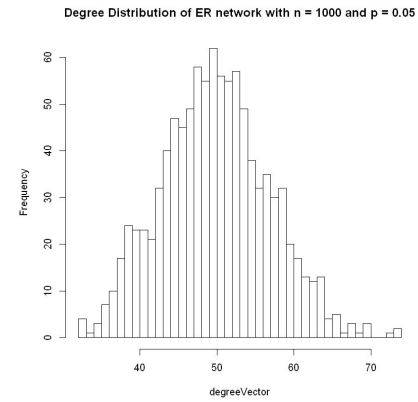
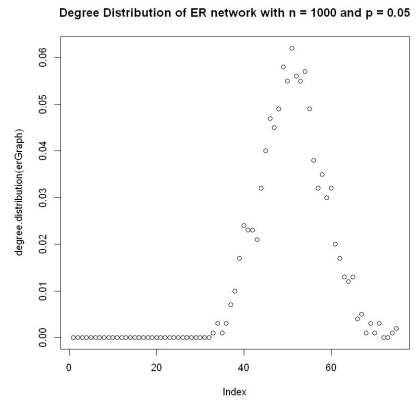
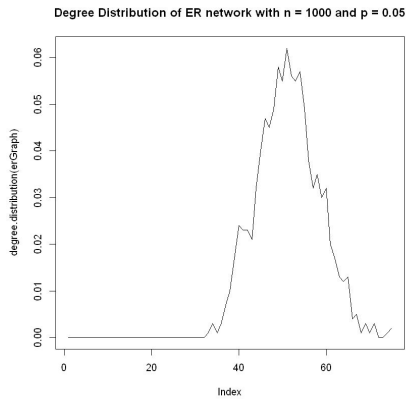
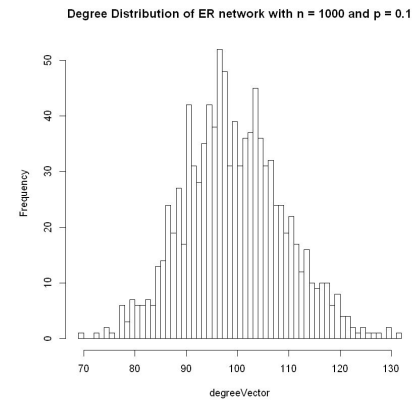
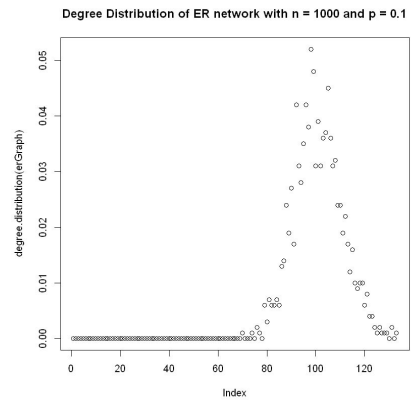
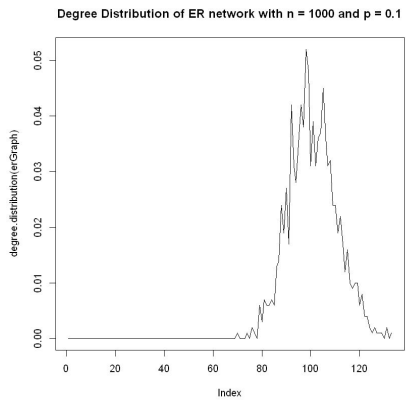
The variance of the degree distribution is:

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

p	Number of Edges	Mean of Degree Distribution		Variance of Degree Distribution	
		Observed	Theoretical *(np)*	Observed	Theoretical (*np(1-p)*)
0.003	1505	3.01	3	2.839	2.991
0.004	1939	3.878	4	3.747	3.984
0.01	5064	10.128	10	10.995	9.9
0.05	25151	50.302	50	48.721	47.5
0.1	49906	99.812	100	94.179	90

The table below contains the plots for the degree distribution. We have given a histogram, scatter plot and a line plot. As the value of 'p' increases the graph evolves and we observe that the range of the values in degree Vector of the histogram also increases. The same observation can also be seen in the scatter plot. As 'p' increases, the model becomes more more likely to include graphs with more edges and less likely to include graphs with fewer edges. ***This shows that the properties of the graph depend sensitively on the choice of the parameter p.***

p	Histogram of Degree Distribution	Scatter Plot of Degree Distribution	Line Plot of Degree Distribution
0.003			
0.004			
0.01			

0.05			
0.1			

The ER model results in Poisson degree distributions that have exponential decay whereas, most real networks exhibit power-law degree distributions that decay much slower than exponential. Hence, we can say that ER model is a poor predictor of degree distribution compared to real networks.

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?

Ans: Not all realizations of the ER network are connected. For the range of values from 0.01 to 0.1 for 'p' the graph is connected whereas for 0.003 and 0.004 it is not. The probability formula for connectivity is as below,

$$P(n) = \lambda * \frac{log(n)}{n}$$

'λ' is given by the product of 'n' and 'p'. The diameter of a network is defined as the longest shortest path between pair of nodes.

$$d = \max_{v \in V} \epsilon(v).$$

In a connected network with n nodes, the diameter is in the range 1 (completely connected) to n - 1 (linear chain)

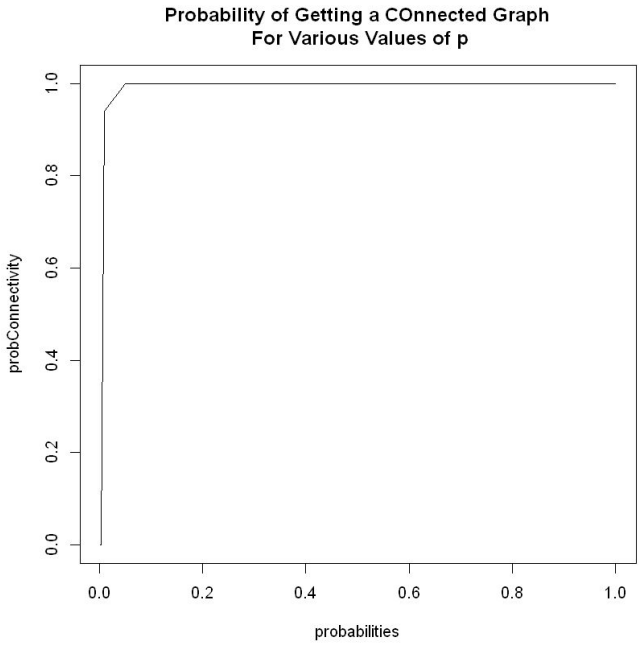
p	Is Graph Connected?	Size of GCC	Normalized SIze of GCC	Edges in GCC	Diameter of GCC/ Diameter of Graph
0.003	NO	950	0.95	1500	14
0.004	NO	980	0.98	1938	12
0.01	YES	NA			6
0.05	YES	NA			3
0.1	YES	NA			3

In the following table, we estimate the probability for getting a connected graph for a particular value of p. For each value of p, we have created 100 graphs and counted the number of times we are getting a connected graph to estimate the probability. We observe that initially for lower values of p, the ER model is disconnected containing a majority of isolated nodes. As p increases, the probability also increases with at p > 0.01, the probability being equal to 1.

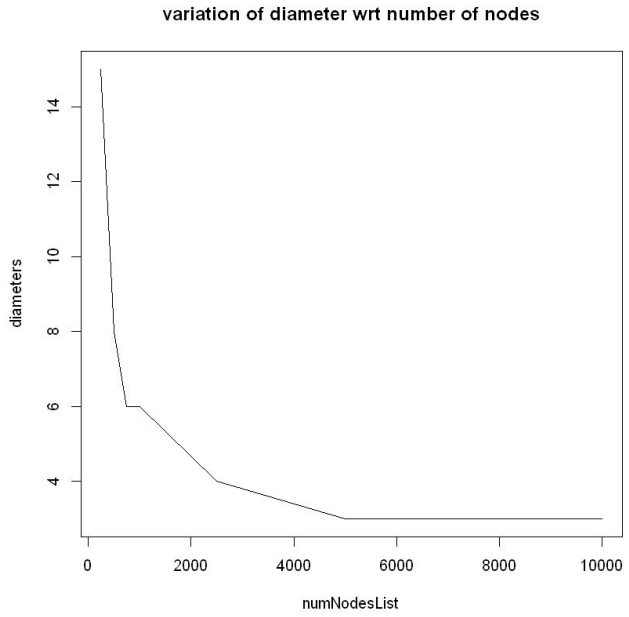
We have also visualized the results in a graph showing the the sharp increase in the probability for values of p.

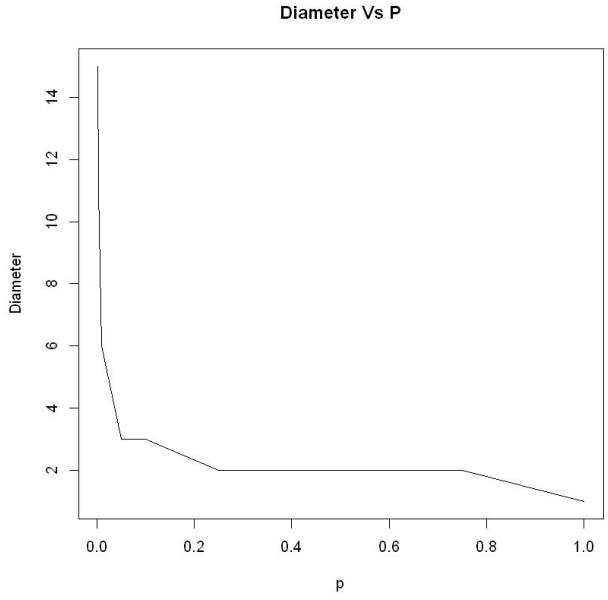
p	Probability of Getting a Connected Graph
0.003	0

0.004	0
0.01	0.94
0.05	1
0.1	1
0.25	1
0.5	1
0.75	1
1	1



Here we show the change in diameter of the graph for n = 1000 as we vary p from 0 to 1. Initially when the graph is not connected, we measure the diameter of the GCC and as the graph becomes connected, the diameter of the graph is measured. We observe, that for a given n, as we vary the model parameter p from 0 to 1, at some critical value of p, the graph becomes finite (network becomes connected) and continuous to decrease, becoming 1 when p = 1.

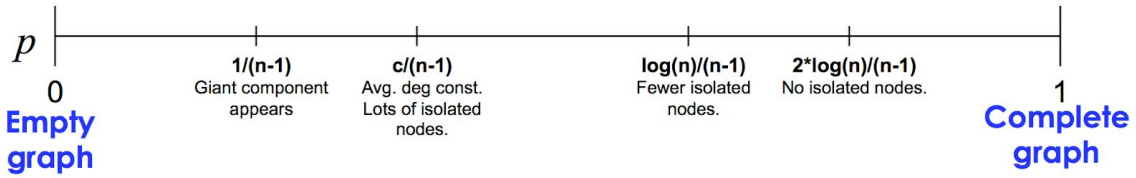




On observing the above information, we can state that ER model results in networks with small diameters, capturing the “small-world” property observed in many real networks. Hence, we can say that the ER model is a good predictor of diameter and average path length compared to real networks.

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?

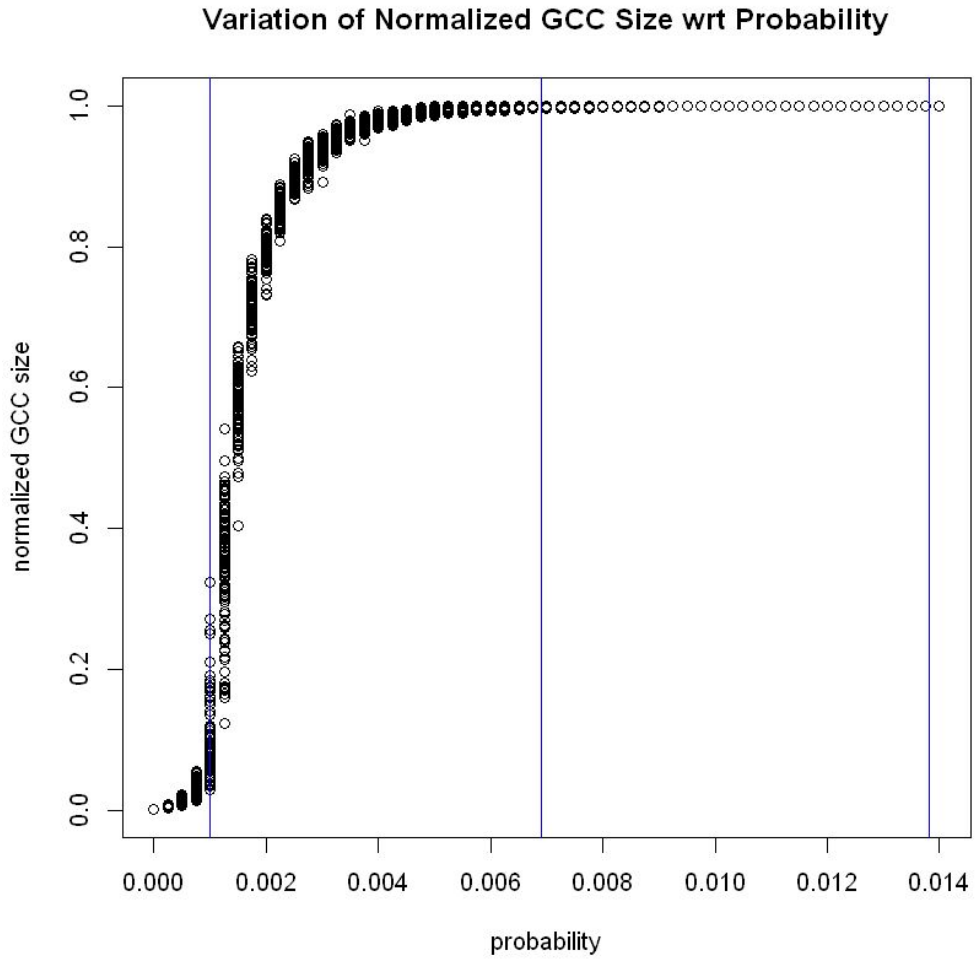
Ans: Below image represents the evolution of G_{np} as 'p' changes.



The plot of normalized GCC size vs 'p' is shown below. There are 3 vertical lines in the graph at some specific 'p' values. The first line represents the value of 'p' at which the Giant Component appears, the second line represents the point with fewer isolated nodes and the last line represents the point where there are no isolated nodes left in the graph. We define "emergence" as the point where the giant connected component starts to appear. ***Hence from the above image, a giant connected component starts to appear when the value of p is 1/(n-1) or when the average degree is 1. In our case the value of 'p' is 0.001.***

Before p = 1/(n-1), we observe that mostly isolated nodes are present in the graph indicating the lack of a GCC. Hence we can say that the ER graph is a collection of disjoint trees in this scenario. As we increase p beyond 1/(n-1) towards ln(n) / n which is roughly 0.0069, we observe that the size of the GCC increases and the number of isolated nodes decreases. At this point, normalized size of the GCC is around 0.99 indicating that very few isolated nodes remain in the graph. As we increase p furthermore till 2 * ln(n) / n which is approximately 0.014, the graph becomes connected with no isolated nodes in the graph.

Another interesting insight obtained from the graph is that at lower values of p, greater variance in the GCC sizes is observed and as p reaches the value of ln(n)/n the variance in GCC sizes drastically reduces with all the random instances of the graphs having similar GCC size.



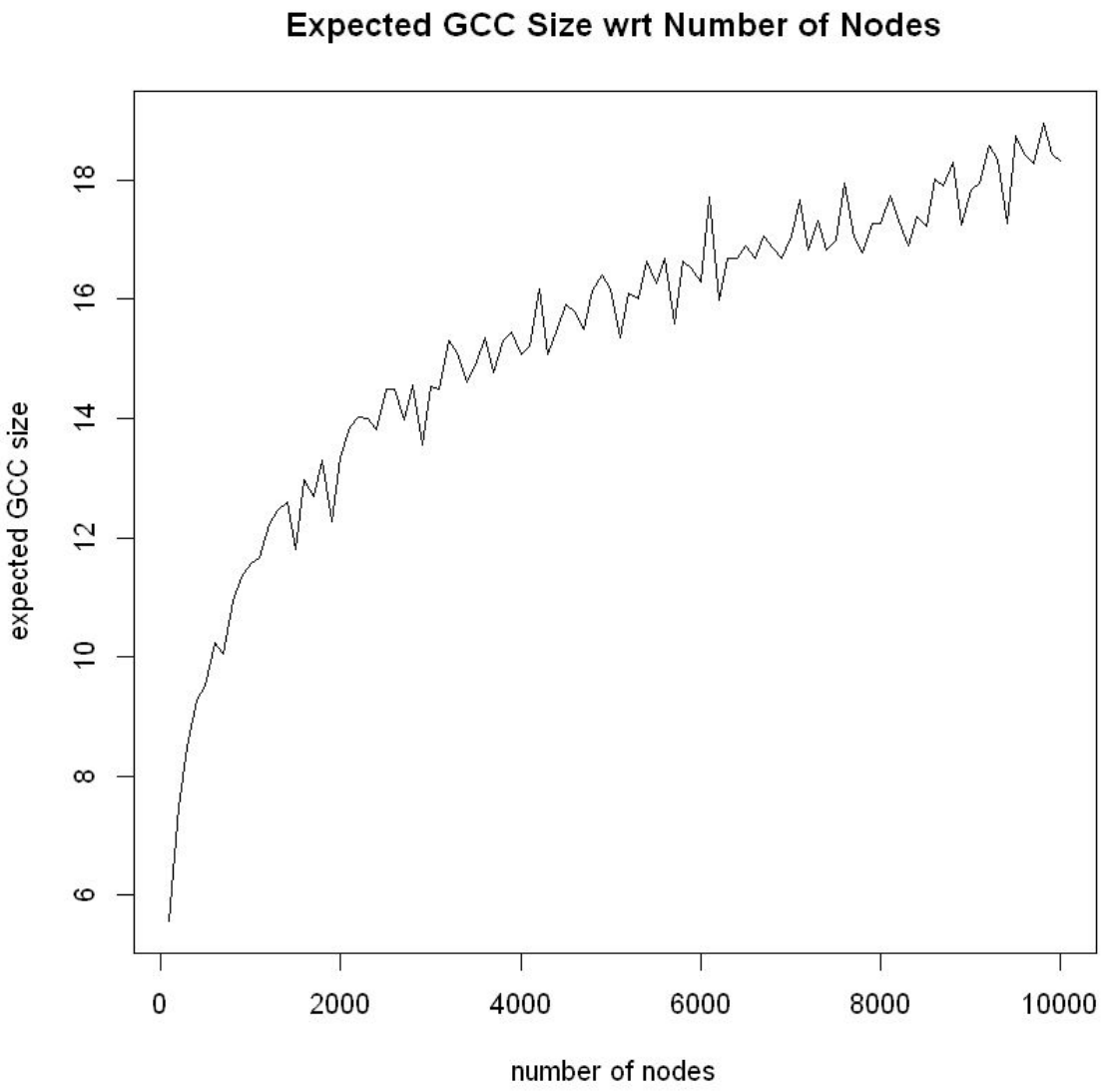
d)

i) Define the average degree of nodes c = n x 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?

Ans: We plotted the graph of expected GCC size vs the number of nodes. We observe that overall the trend is increasing but with ups and downs almost throughout the entire range.

With the values obtained for the size of GCC for various values of n in this scenario, we can conclude that a large GCC has not started to emerge and that there is a presence of a large number of isolated nodes in the graph. This is in alignment with the property of ER graph which states that for np <

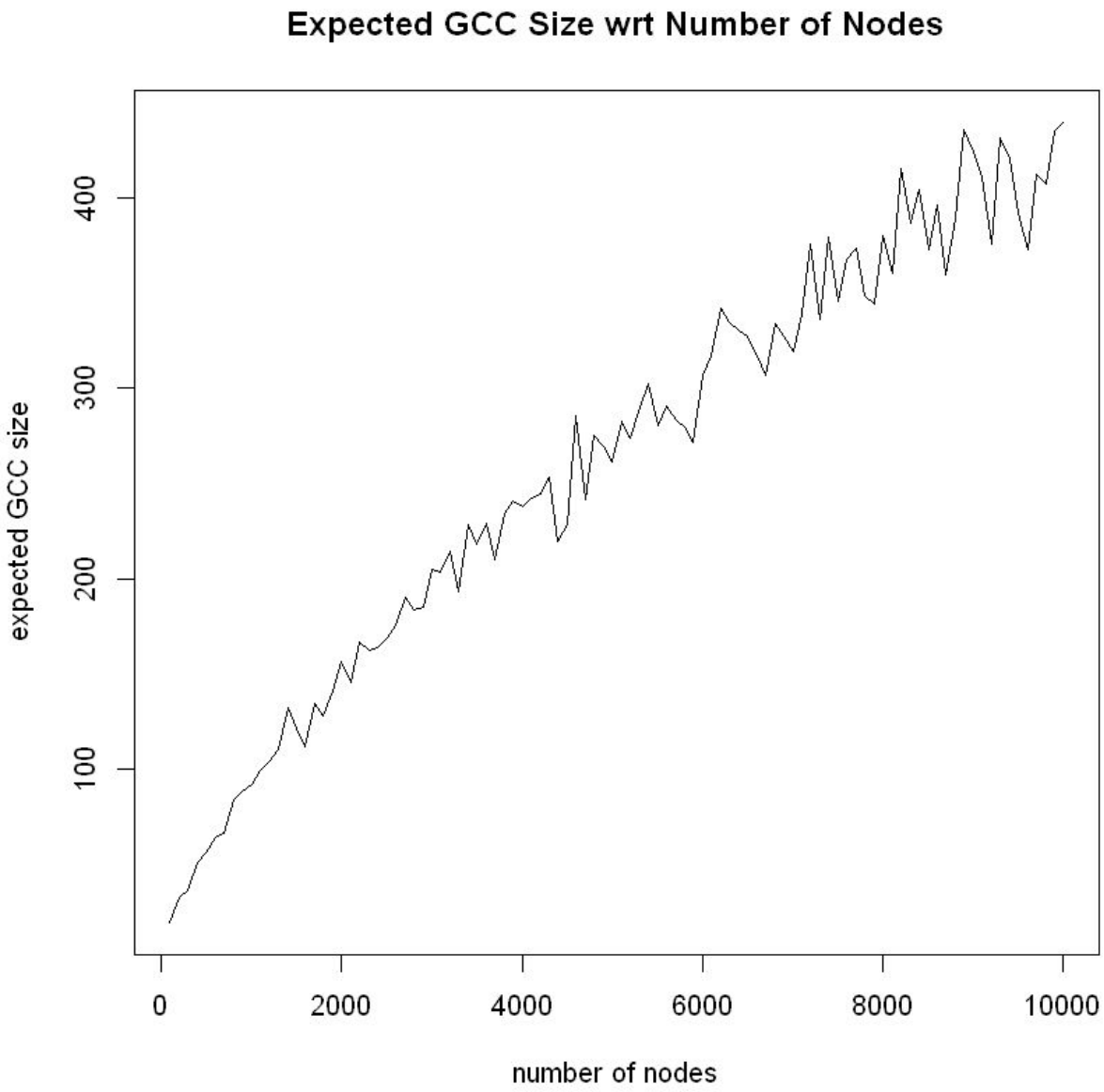
1, G(n, p) will almost surely have no have no connected components of size larger than O(log(n)).



ii) Repeat the same for c = 1.

Ans. In this scenario, the average degree is now 1. As we observed in the previous question, a giant connected component for an ER model starts to emerge as the average degree reaches 1. We observe the same thing here - a giant connected component has started to emerge for every value of n.

As per the behavior of the ER graph, when np = 1, the size of the GCC will be of the order $n^{\frac{2}{3}}$ which we have been able to verify with the values obtained for the graph.



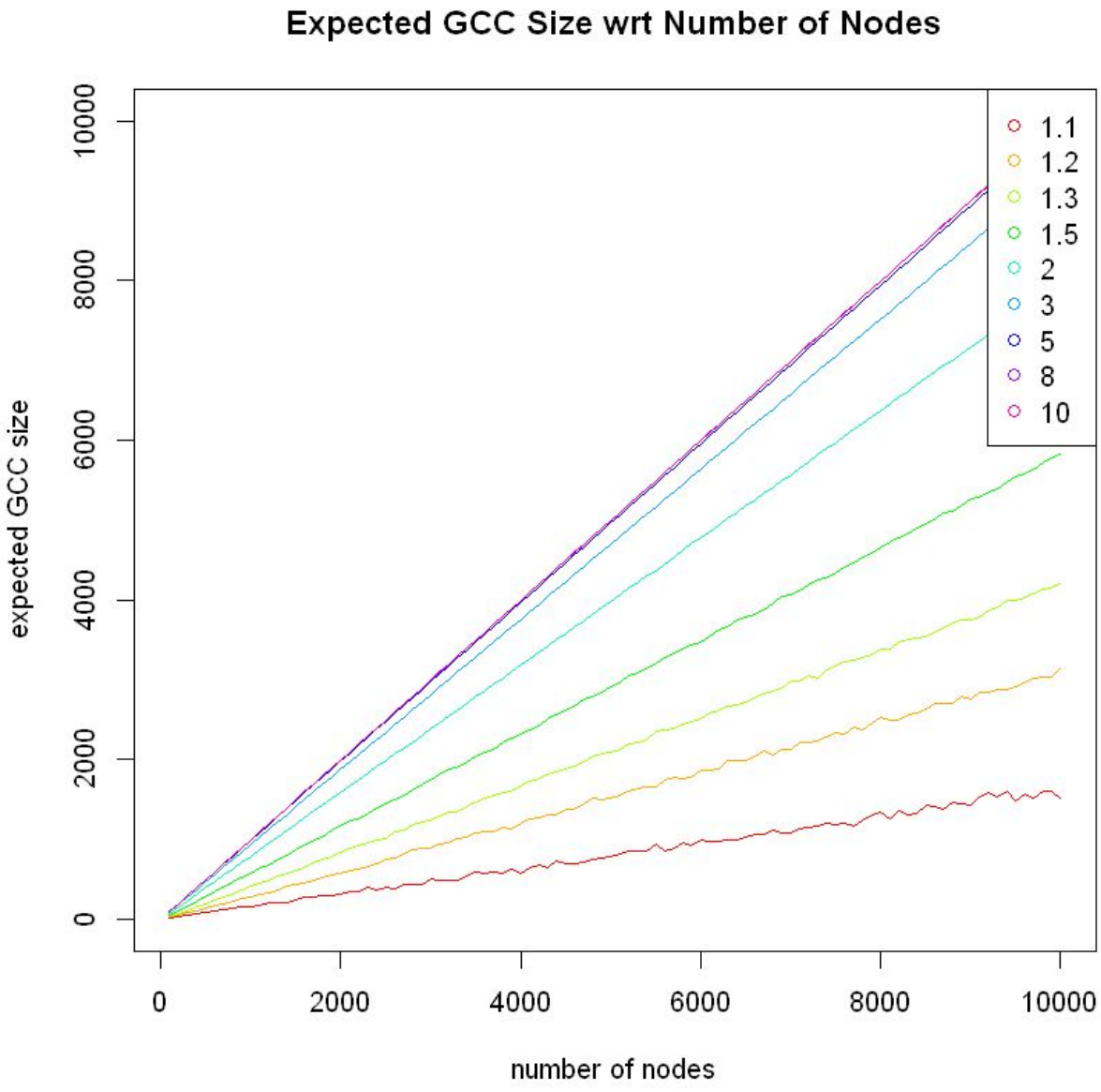
iii) Repeat the same for values of c = 1.1; 1.2; 1.3, and show the results for these three values in a single plot.

Ans: For the scenario when c is greater than 1, a graph in G(n,p) will almost surely have a unique giant component containing a positive fraction of the vertices.

As we have seen from the previous parts, the threshold for generating a connected graph (with negligible isolated nodes) is c = ln(n)/n, which is a reducing expression with max value at (n = 100) = 0.046 and minimum at (n = 10000) = 9.21*e-4

For any c > 4.6, p > 4.6/100 and the size of GCC will be equal to to the number of nodes, which is clearly visible from the graph below.

On increasing c, the edge formation probability increase, which will lead to a larger giant connected component for fixed n.



2. Create networks using preferential attachment model

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?

Ans: There are two major differences between ER models (random networks) and real networks:

a. Growth: Real networks are the result of a growth process that continuously increases N. In contrast the random network model assumes that the number of nodes, N, is fixed.
b. Preferential Attachment: In real networks new nodes tend to link to the more connected nodes. In contrast nodes in random networks randomly choose their interaction partners.

Real networks can be modeled by preferential attachment models. Here we have used the Barabasi-Albert model which generates scale free model and is defined as follows:

We start with m_o nodes, the links between which are chosen arbitrarily, as long as each node has at least one link. The network develops following two steps:

a. Growth: At each timestep we add a new node with m (<= m_0) links that connect the new node to m nodes already in the network
b. Preferential attachment: The probability $\prod(k)$ that a link of the new node connects to node i depends on the degree k_i as

$$\prod(k) = \frac{k_i}{\sum_j k_j}$$

In the preferential attachment model, the new network members prefer to make a connection to the more popular existing members. More specifically, a node's chance of being selected is directly proportional to the number of connections it already has, or its degree. This is the mechanism which is called preferential attachment. We have used the 'barabasi.game' method for creating preferential attachment graphs.

At each timestep, the newly added node creates m edges to any of the existing nodes. Hence, there is no possibility of occurrence of any isolated nodes and hence the graph is always connected.

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

Ans: In this part, we have reported the number of communities and sizes of respective communities for n = 1000 and n = 10000. Additionally, we have also plotted the networks with the communities highlighted in them.

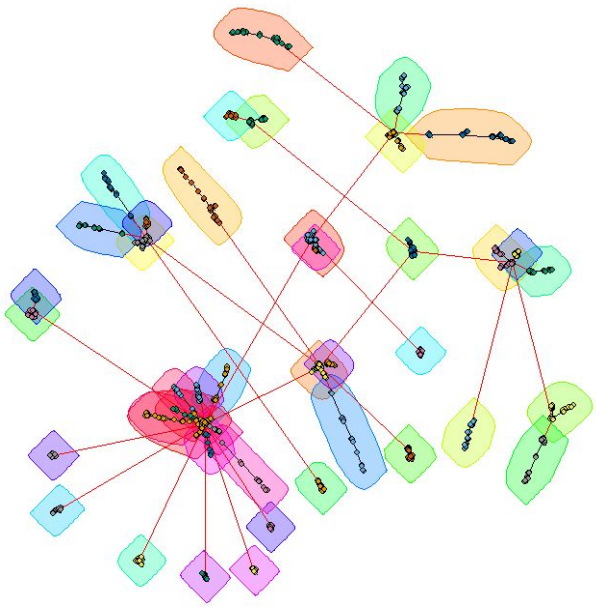
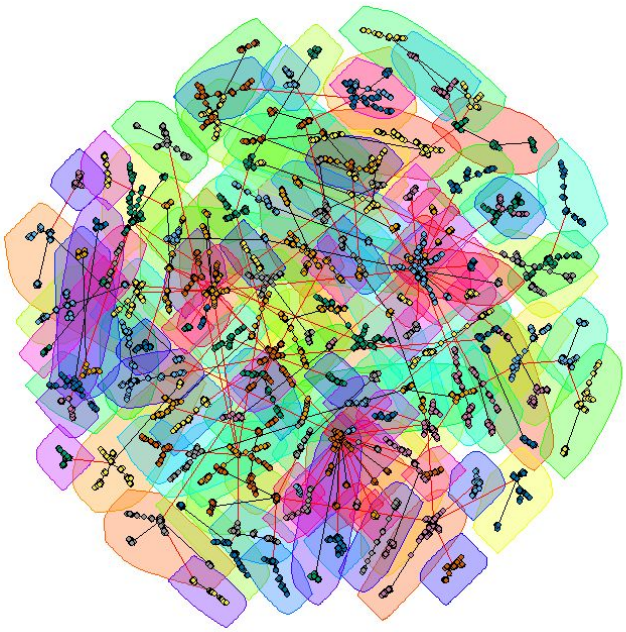
The fast greedy method for community detection is a bottom-up hierarchical approach which tries to optimize modularity in a greedy manner. Initially, every vertex belongs to a separate community, and communities are merged iteratively such that each merge is locally optimal (i.e. yields the largest increase in the current value of modularity). The algorithm stops when it is not possible to increase the modularity any more, so it gives you a grouping as well as a dendrogram. The method is fast and it is the method that is usually tried as a first approximation because it has no parameters to tune. However, it is known to suffer from a resolution limit, i.e. communities below a given size threshold will always be merged with neighboring communities.

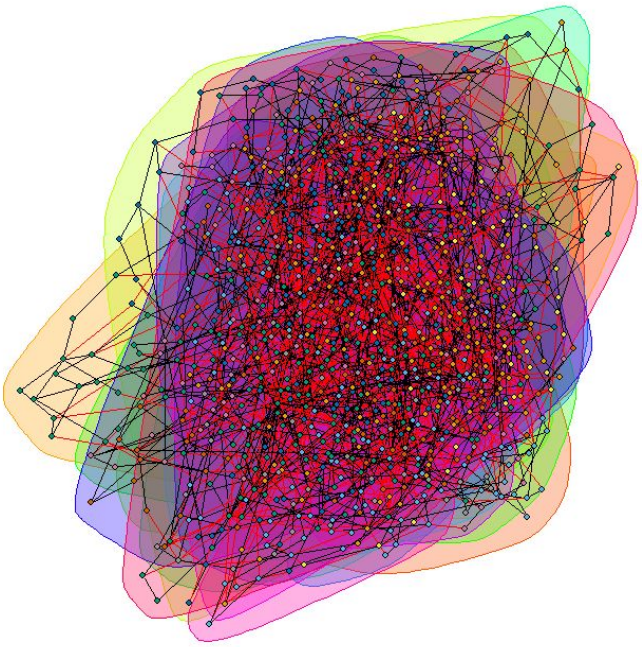
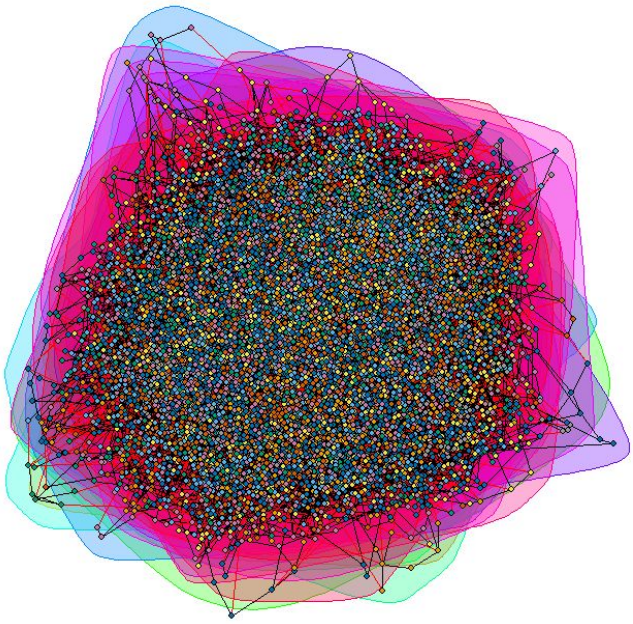
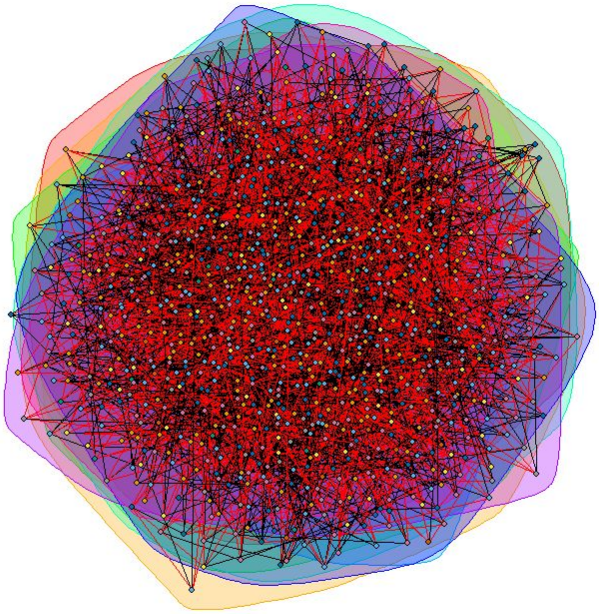
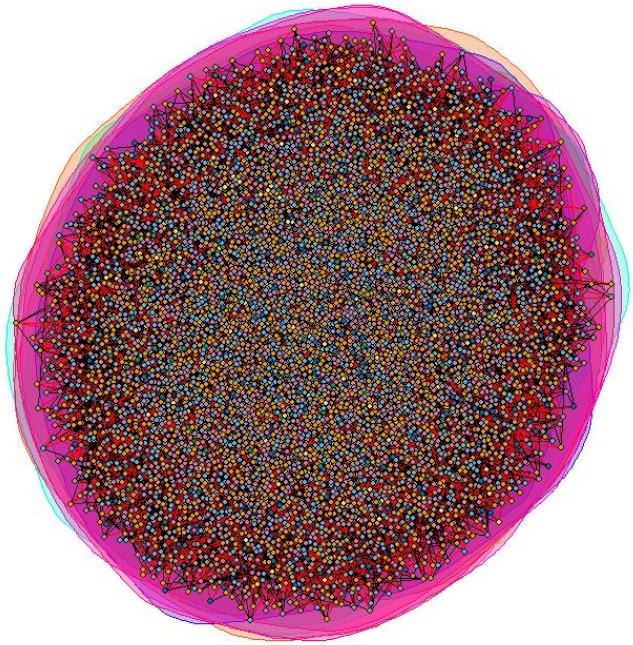
We observe that as m increases, the number of communities decrease. Also, the number of communities for n = 10000 is greater than what was observed for n = 1000. Also, as we increase m, we observe that the size of the largest community increases, i.e., for m = 1, the largest community has size 69, whereas for the same n (1000), the largest community size is 106 for m = 2 and 226 for m = 5.

On observing the community structure of the networks generated, we observe that while most nodes in the network only have a few links, a few of these nodes gradually turn into hubs. These hubs can be attributed to the *rich-gets-richer* phenomenon: due to preferential attachment new nodes are more likely to connect to the more connected nodes than to smaller nodes. Hence, larger nodes acquire links at the expense of smaller nodes, eventually turning into hubs.

	N = 1000		N = 10000	
m	Number of Communities	Community Sizes	Number of Communities	Community Sizes
1	43	69 56 37 39 33 34 31 37 37 30 29 29 34 28 35 26 24 25 23 26 19 21 19 19	114	165 259 137 165 134 246 149 153 204 138 173 150 149 130 129 147 121 120 162 127 117 147 125 122 122 134 109 107 107 109 105 103 110

		16 16 16 16 15 15 14 14 14 14 12 12 11 10 10 9 9 9 8		109 104 108 98 112 96 113 92 96 113 92 108 91 91 90 92 92 88 94 93 84 86 83 79 83 82 97 97 77 100 79 73 71 70 70 68 66 66 62 68 72 57 58 64 66 56 59 53 53 55 50 56 48 47 47 46 46 48 46 46 45 44 47 41 45 40 52 37 34 33 33 35 29 32 30 26 29 23 22 20 22
2	19	90 61 83 33 71 89 41 26 55 42 22 20 106 44 38 25 34 22 98	38	107 167 155 159 94 140 151 78 109 271 76 75 544 126 509 58 130 103 53 78 791 154 260 77 138 245 192 282 218 207 384 469 383 679 485 287 708 858
5	9	226 192 22 69 191 30 140 93 37	17	268 1335 68 64 90 56 54 547 2028 49 33 64 242 29 2060 1889 1124

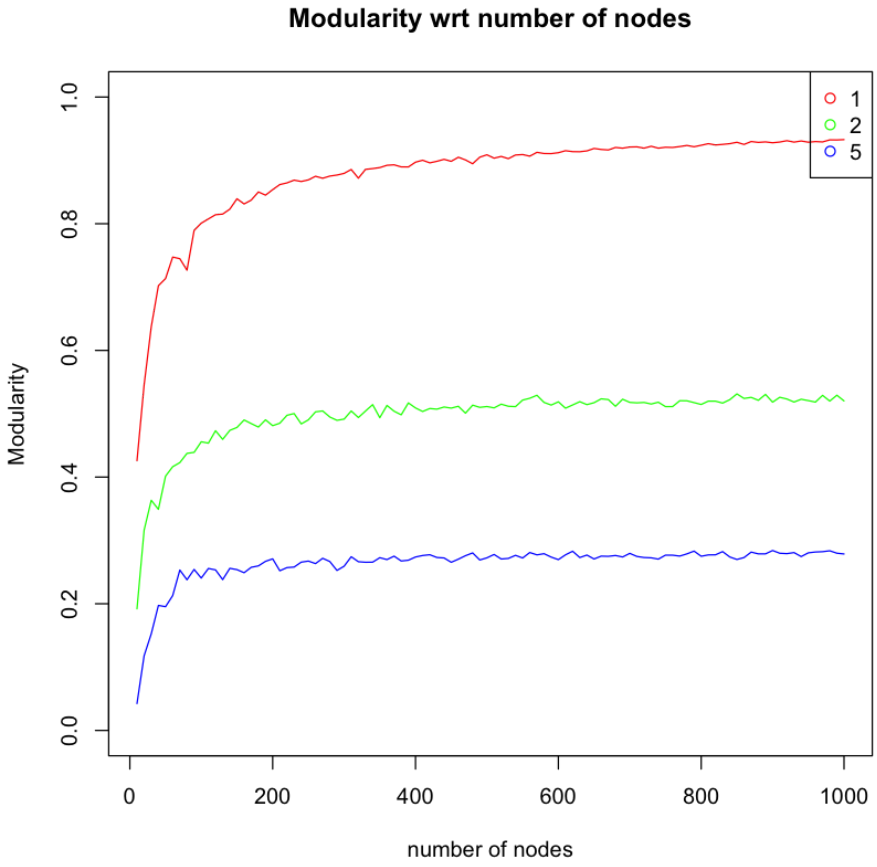
m	N = 1000	N = 10000
1		

2		
5		

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?

Ans: Modularity is the fraction of the edges that fall within the given groups minus the expected fraction if edges were distributed at random. measure the strength of division of a network into modules (also called groups, clusters or communities). Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.

We can observe the difference in the values of modularity for n=1000 and n=10000 for different values of m. As we increase the value of 'm', the modularity is found to be decreasing. For a fixed value of 'm', as we increase the size of the network, the modularity increases by a small value.



Following is the equation the modularity of scale-free networks as a function of the network size 'N' for different values of 'm'.

$$M(N, m = 1) = 1 - \frac{2}{\sqrt{N}}$$

m	n	Is This Graph Connected?	Number of Edges	Modularity
1	**1000**	YES	999	0.931970008046087
	10000	YES	9999	0.977643008825351
2	**1000**	YES	1997	0.528712698491902
	10000	YES	19997	0.531060845054657
5	**1000**	YES	4985	0.282460319775776
	10000	YES	49985	0.277326738683798

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

Preferential attachment network is a type of a scale free network. A ***scale-free*** network is a network whose degree distribution follows the ***power law***.

The degree distribution of a power law network can be approximated with

$p_k \sim k^{-\gamma}$ where the exponent $-\gamma$ is its degree exponent

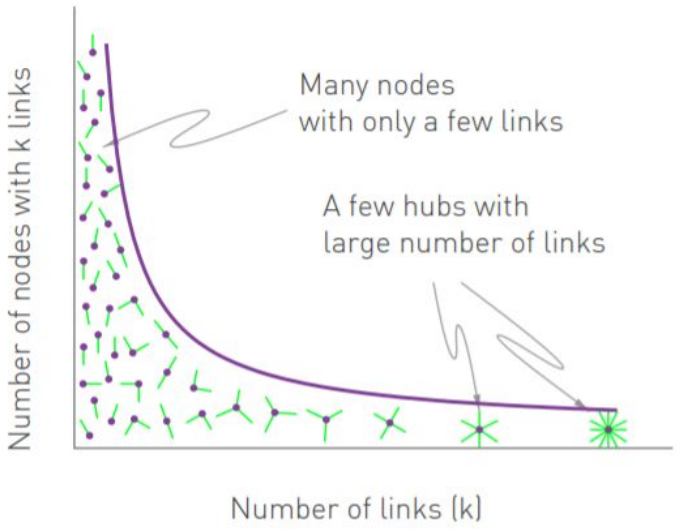
If we take logarithm of the above equation, we get

$$log\, p_k \sim -\gamma log\, k$$

Hence, $log\, p_k$ depends linearly on $log\, k$, the slope of this line being the degree exponent γ.

In this problem, we plot both the log log plots of the degree distribution and also a histogram representing the degree distribution of the network.

From the histograms we can observe that a large number of nodes have lower degrees and fewer nodes have large degrees. This indicates the presence of hubs which connect nodes with lower degrees. We also observe that with in increase in the number of nodes in the network, the size of hubs increases which is indicated by the range of degrees in the network. The following representative image highlights the expected degree distribution in preferential attachment models which is in line with our observations.



We also observe that the range of degrees also increase with an increase in m, with the degrees going up to 40 for m = 2 and 80 for m = 5.

The preferential attachment model is called a fat tailed network as its degree distribution has a power law tail in the high-k region. As a consequence $< k^2 >$ is much larger than <k>, resulting in considerable degree variations.

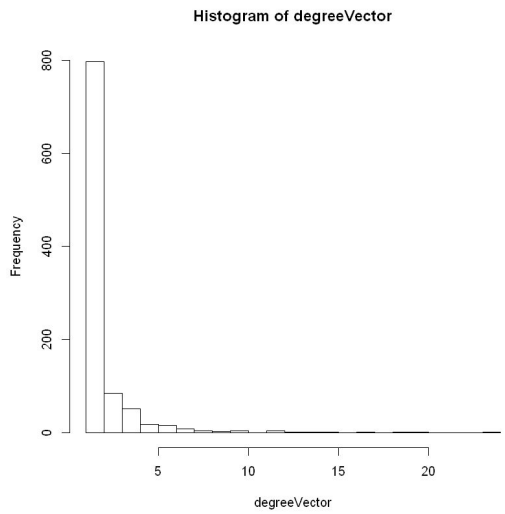
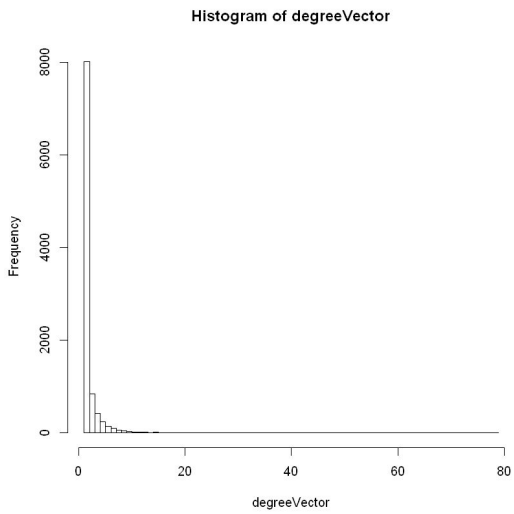
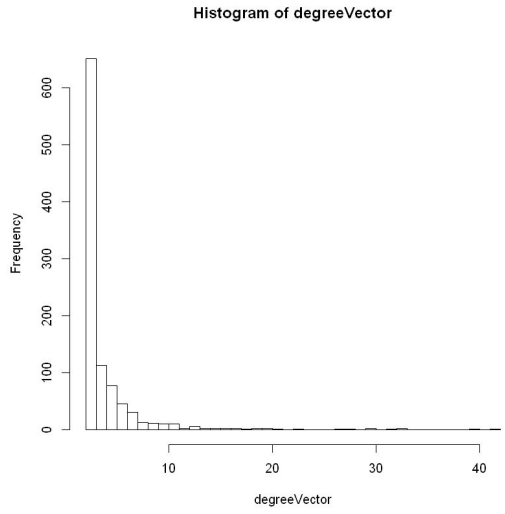
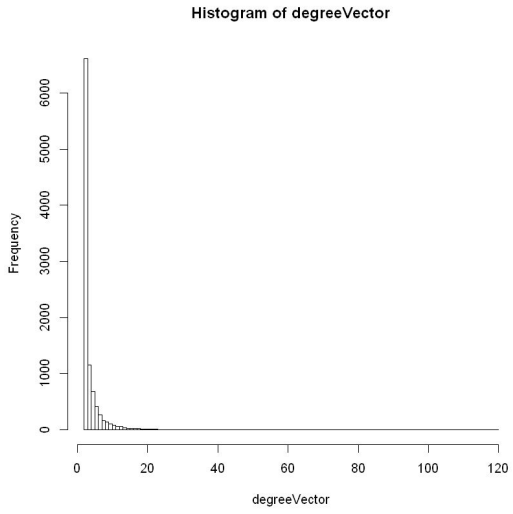
In scale free models such as the preferential attachment model, γ lies between 2 and 3. The Barabasi model predicts the degree distribution as

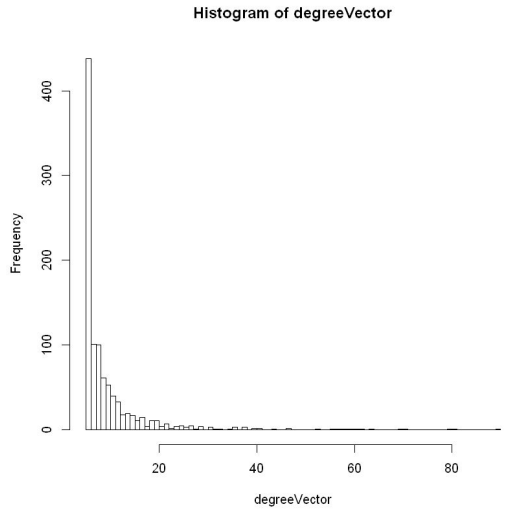
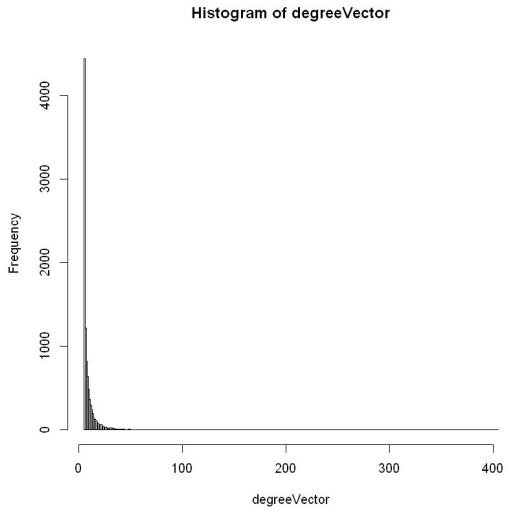
$$p(k) \approx 2m^{\frac{1}{\beta}} k^{-\gamma}$$

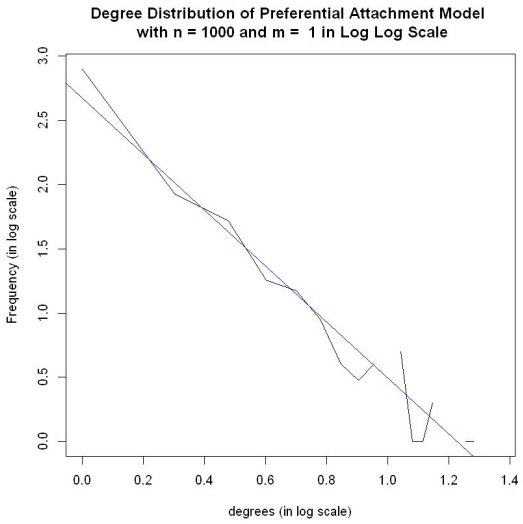
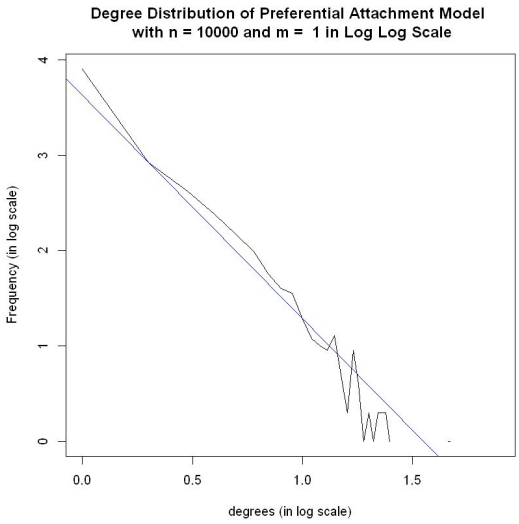
With

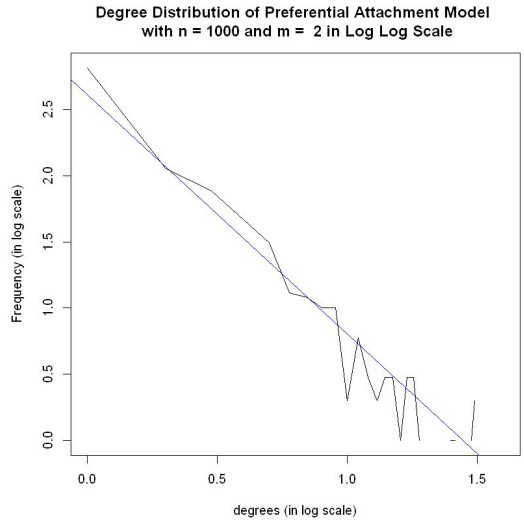
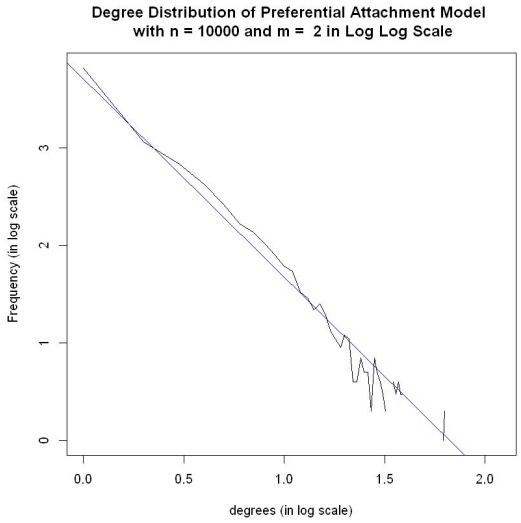
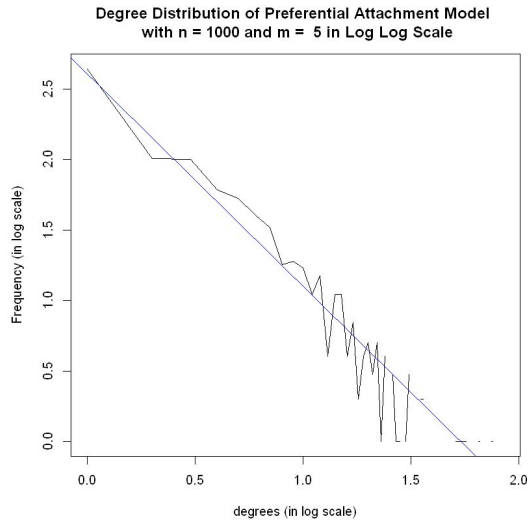
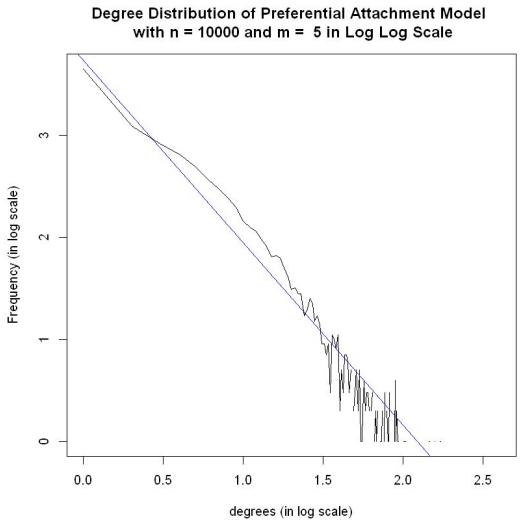
$$\gamma = \frac{1}{\beta} + 1 = 3$$

The degree distribution from the Barabasi Albert model follows a power law with degree exponent $\gamma = 3$. We have validated this value in our experiments by measuring the slope of the log log representation of the degree distribution by fitting a line among the data points and further verified by using the fit_power_law degree function. In all the Barabasi models, the power law exponent obtained is close to 3.

m	n = 1000	n = 10000
1		
2		

5		

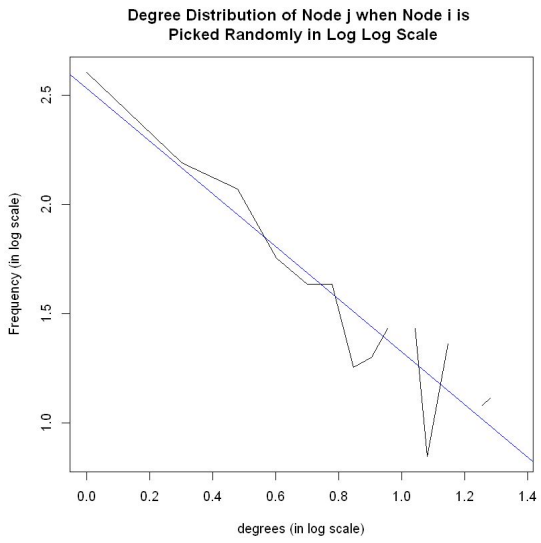
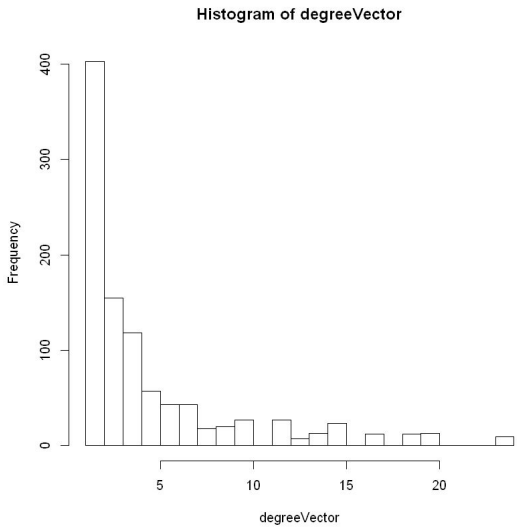
m	**n = 1000**	**n = 10000**
1		
	Slope = -3.008	Slope = -3.1077

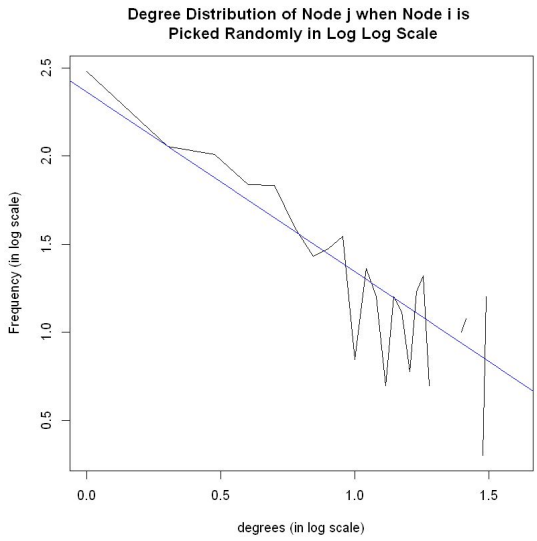
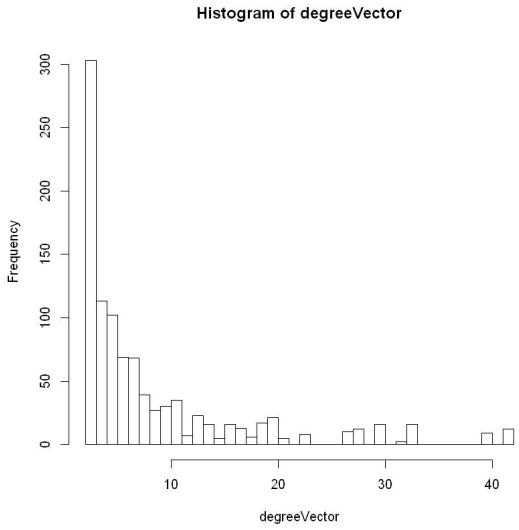
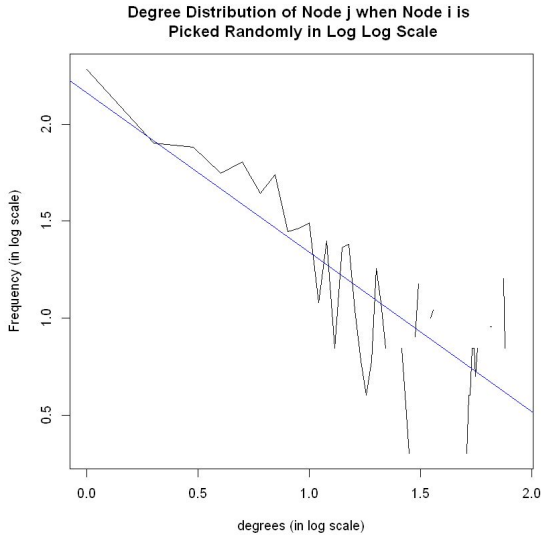
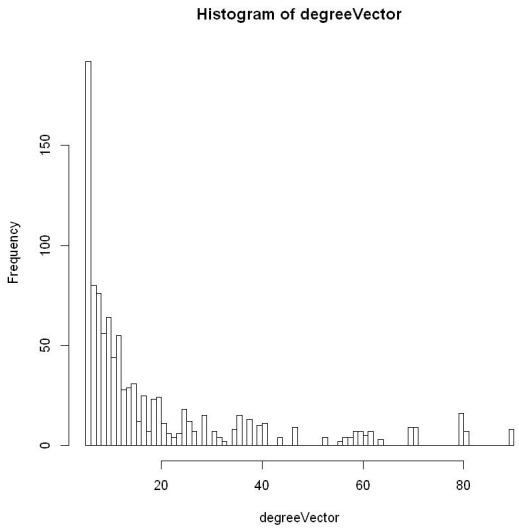
2		
	Slope = -2.9756	Slope = -3.3367
5		
	Slope = -2.8978	Slope = -2.9313

(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?

On plotting the log log representation of the degree distribution, we observe that this distribution also follows the power law equation with γ ranging between 2 and 3. Hence this distribution also represents scale free networks where the first moment of the degree distribution is finite but the second and higher moments diverge as N -> ∞.

On further comparison of this distribution with the degree distribution of the preferential attachment models obtained earlier, we observe that though both follow power law dynamics, the γ differs for both of them with it being closer to 3 for the barabasi model and it being closer to 2 in this part.

<table>
<tr><th>m</th><th colspan="2">Degree Distribution of Node j when Node i is Picked Randomly and its Neighbor j is also Picked Randomly</th></tr>
<tr><td rowspan="2">1</td><td> 
</td><td rowspan="2"> 
</td></tr>
<tr><td>Slope = -2.2304</td></tr>
</table>

2		
	Slope = -2.056	
5		
	Slope = -2.1797	

(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.

The age of the node is the time since it is born at timestep t0 to the current timestep 't'. In preferential attachment graphs, nodes appear one by one, each selecting m other nodes at random to connect to. One node is born at each time tick and so at time t, there will be t nodes. So change in degree ki of node i born at time i is given by,

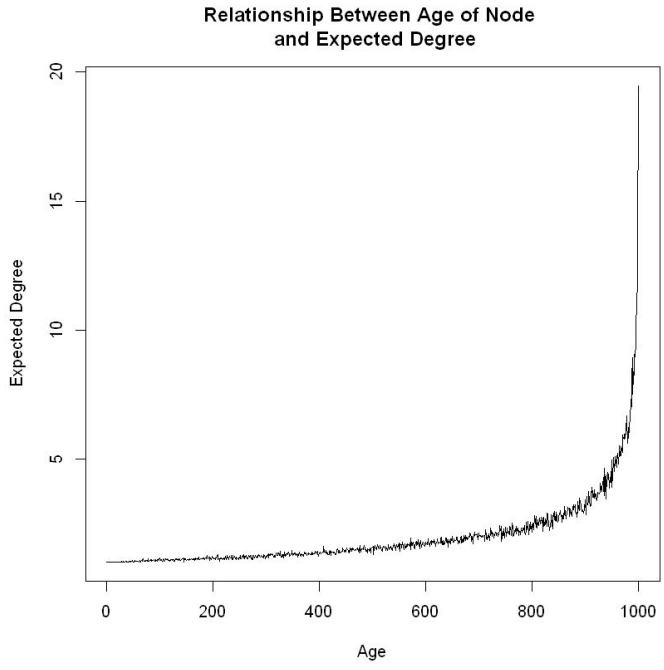
$$\partial k_i(t)/\partial t = m/t$$

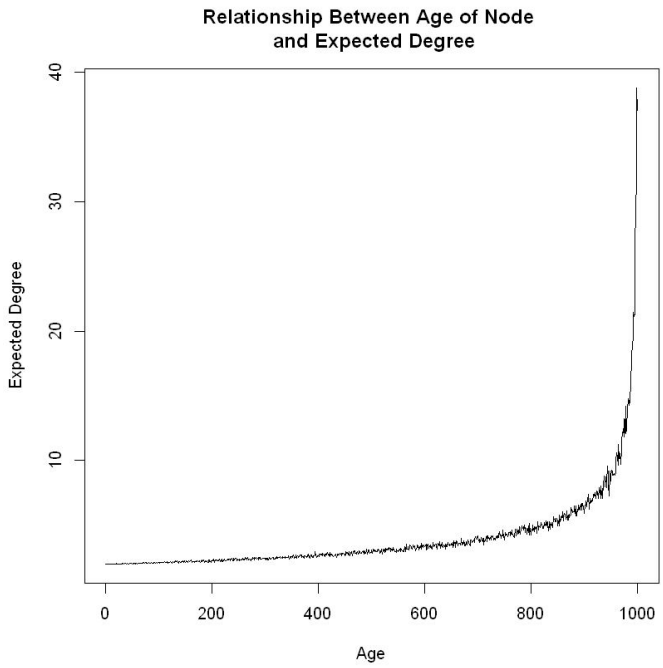
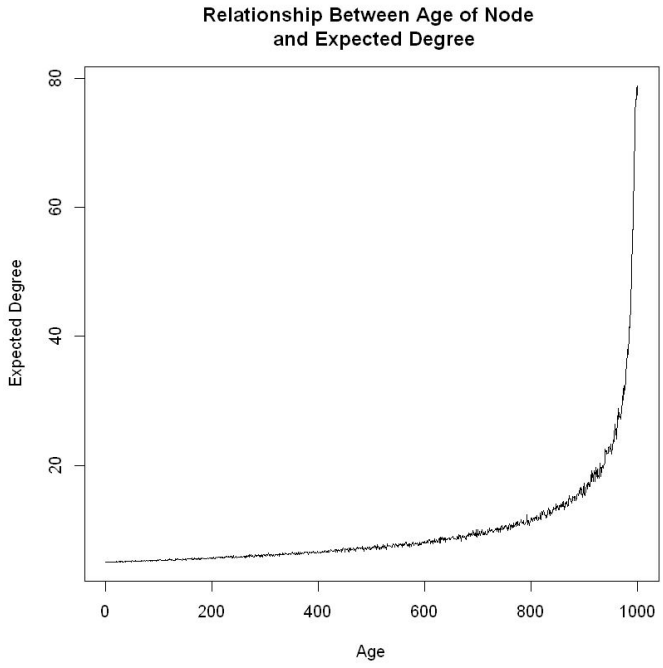
To get how many new edges does a node accumulate since its birth at time i until time t, we integrate the above equation from i to t, to get,

$$k_i(t) = m + m\log(t/i)$$

So if $i<j$, $k_i(t) > k_j(t)$ implying that older nodes on average have more edges(higher degree)

This formula has been experimentally verified and the expected degree of a node and its relation to its age is shown through the graphs below for different values of m.

m	Relationship Between Age of Node and Expected Degree
1	

2	 Relationship Between Age of Node and Expected Degree Expected Degree 10 20 30 40 0 200 400 600 800 1000 Age
5	 Relationship Between Age of Node and Expected Degree Expected Degree 20 40 60 80 0 200 400 600 800 1000 Age

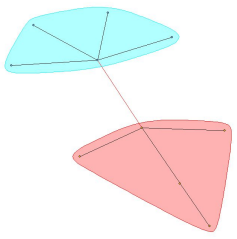
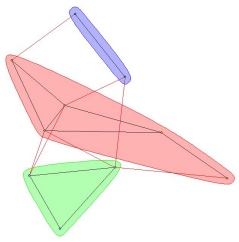
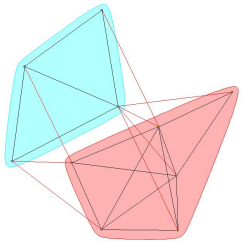
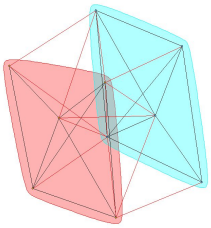
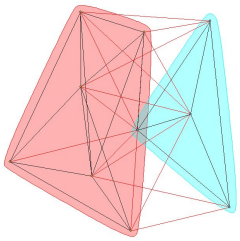
(g) Repeat the previous parts for m = 2, and m = 5. Why was modularity for m = 1 high?

Ans: The results for m = 2 and m = 5 have been incorporated in the tables in the previous parts for easier comparison. The relation between modularity and 'm' is given by the following equation.

$$M(N, m) = (a + \frac{1-a}{m}) * (1 - \frac{2}{\sqrt{N}})$$

'a' represents some constant value, and 'N' is the size of the network.

From the above equation, we can see that modularity for a fixed 'N' inversely depends on 'm' and hence we observe a decrease in modularity for large values of 'm'. To further understand this better, we have marked communities in a small preferential model with N = 10 for various values of m and have reported the results in the table below. As we can see in the table below, for m=1 we can clearly see that the network has a tree like structure with more connections within communities and less connections among communities. As we increase 'm', though the intra-community connections remain dense, the inter-community connections are no longer sparse thereby reducing the modularity.

M = 1	M = 2	M = 3	M = 4	M = 5
				

(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.

Ans. For this problem, we have constructed a preferential attachment network using the 'barabasi.game' function and have taken its degree sequence to create two graphs using the 'degree.sequence.game' function. We have explored two algorithms for creating these graphs - simple.no.multiple, and the vl algorithm.

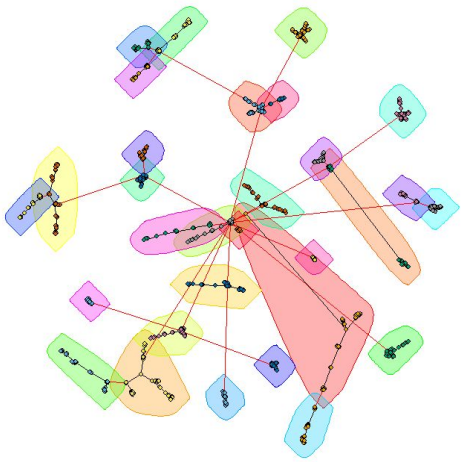
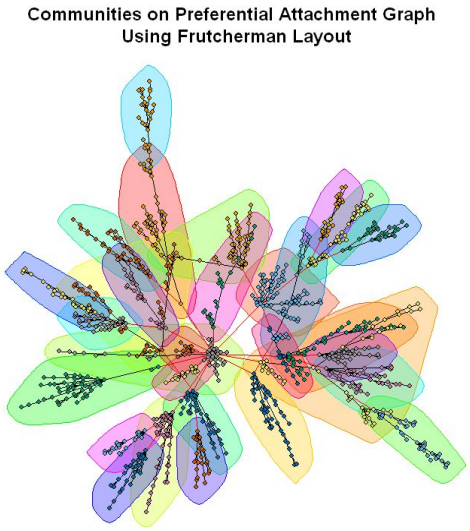
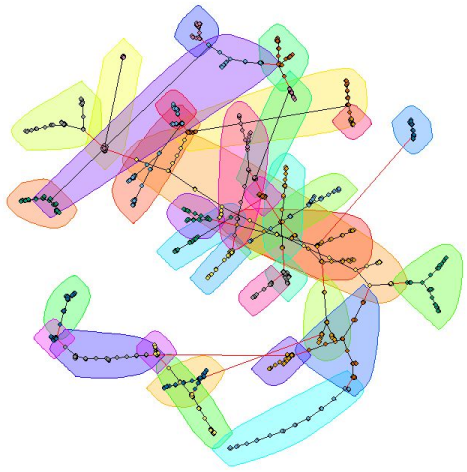
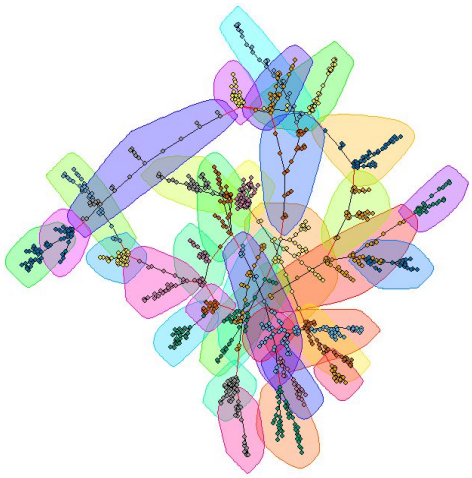
The simple.no.multiple method avoids the formation of multiple edges and loop edges whereas the vl algorithm is a more sophisticated algorithm which always generates undirected, connected simple graphs.

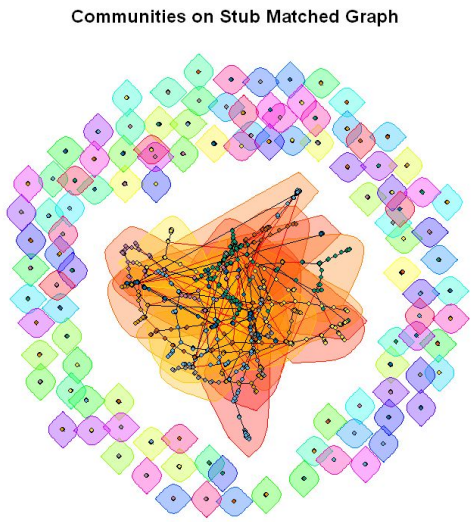
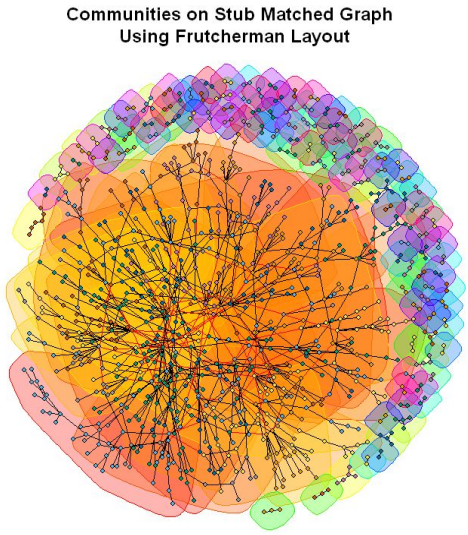
From our observation of the connectivity and modularity of these graphs, we observe that both the preferential attachment and the graph generated by the vl algorithm are connected, whereas the simple.no.multiple algorithm produces an unconnected graph with isolated nodes.

On observing the community structure and modularity of the preferential attachment model and the graph generated by the vl algorithm, we see that both look similar and have high modularity. In both the graphs we observe the presence of hubs indicating the possibility that both create scale free networks following the power law equation. Additionally, the presence of these distinct clusters with sparse inter community links point to these models having high modularity as can be verified by the values obtained by the inbuilt modularity function.

On the other hand, the graph generated by the simple.no.multiple method shows the presence of many small isolated clusters containing one or two nodes and one giant cluster containing multiple nodes. This property of the graph indicates that the model will have a lower modularity as compared to the vl method for generating graphs which has also been verified by value obtained by the inbuilt modularity function.

Type of Graph	**Number of Edges**	**Is This Graph Connected?**	**Modularity**
Preferential Attachment	999	YES	0.933307682056434
Degree Sequence with vl algorithm	999	YES	0.935184433682934
Degree Sequence with simple.no.multiple algorithm	999	NO	0.852107362617876

Graph	Community Structure	
Preferential Attachment		
Degree Sequence with vl algorithm		

Degree Sequence with simple.no. multiple algorithm		

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:** $P[i] \sim (ck_i^{\alpha} + a)(dl_i^{\beta} + b),$ **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,** α **= 1;** β **=1, and a = c = d = 1; b = 0. Plot the degree distribution. What is the power law exponent?**

The combined results for both the parts have been reported at the end of part b. For the simulation of this network, we have used the sample_pa_age function. This creates a discrete time step model of a growing graph. We start with a network containing a single vertex (and no edges) in the first time step. Then in each time step (starting with the second) a new vertex is added and it initiates a number of edges to the old vertices in the network. The probability that an old vertex is connected to is proportional to

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

Here k_i is the in-degree of vertex i in the current time step and l_i is the age of vertex i. The age is simply defined as the number of time steps passed since the vertex is added, with the extension that vertex age is divided to be in aging.bin bins.

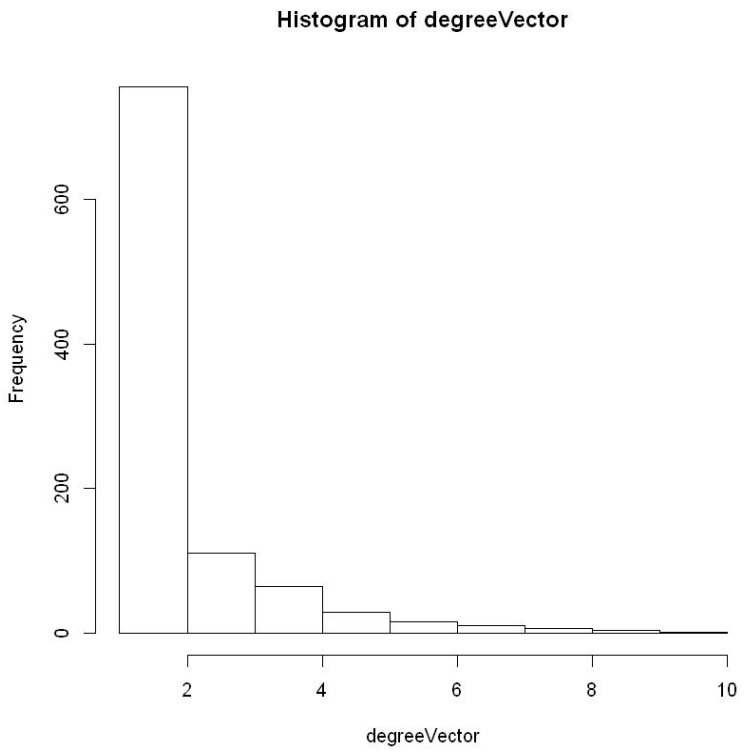
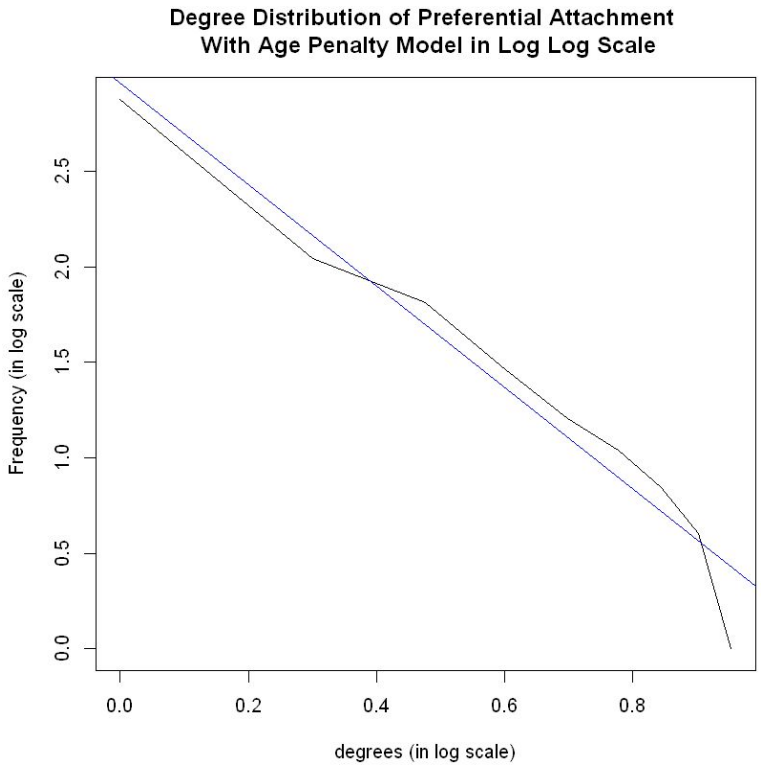
c, α, a, d, β and b are parameters and they can be set via the following arguments: pa.exp (α, mandatory argument), aging.exp (β, mandatory argument), zero.deg.appeal (a, optional, the default value is 1), zero.age.appeal (b, optional, the default is 0), deg.coef (c, optional, the default is 1), and age.coef (d, optional, the default is 1).

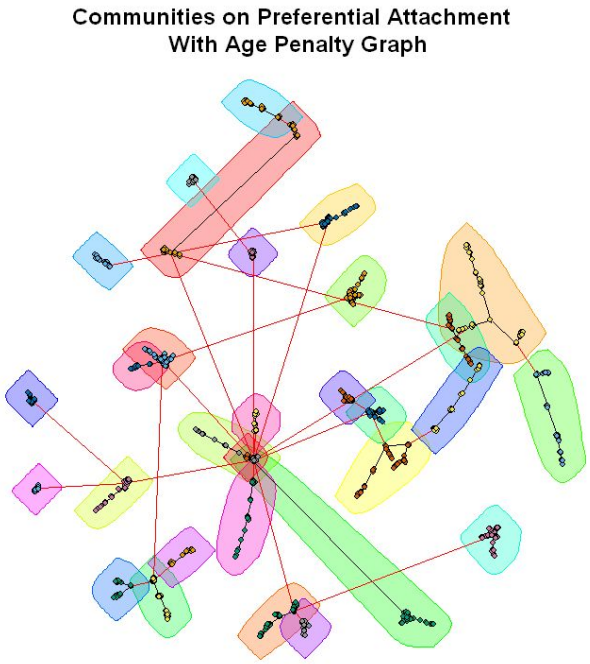
The network generated in this model follows power law dynamics with the power law exponent $\gamma = 4.7396$. Networks with this value of γ have both the first and second moment as finite.

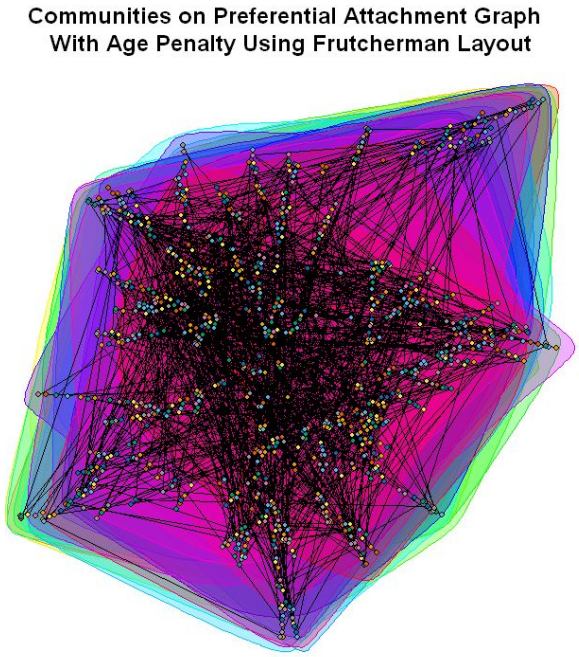
b. Use fast greedy method to find the community structure. What is the modularity?

On observing the modularity and the community structure of the model, we observe the presence of hubs similar to what we observed in the Barabasi model. The high modularity observed indicates the presence of dense clusters with sparse connections between the clusters.

Number of Edges	999
Is This Graph Connected?	YES
Modularity	0.935871306742183
Number of Communities	35
Community Sizes	42 39 37 38 37 33 34 33 36 31 31 31 31 34 31 30 30 29 33 27 26 25 25 25 21 23 24 22 22 23 21 21 19 19 17

Degree Distribution (Histogram)	
Degree Distribution (Log Log Scale)	

Slope of Log Scale Distribution / Power Law exponent	-4.7396
Communities in Graph	

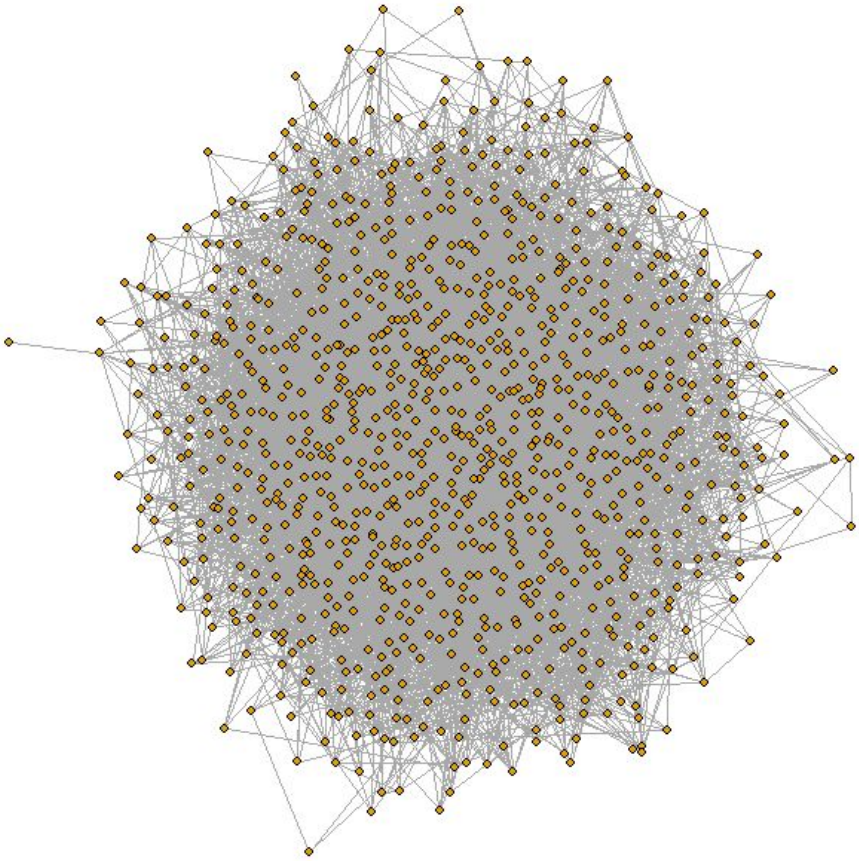
Communities in Graph (Fruchterman Layout)	

Random Walk on Networks

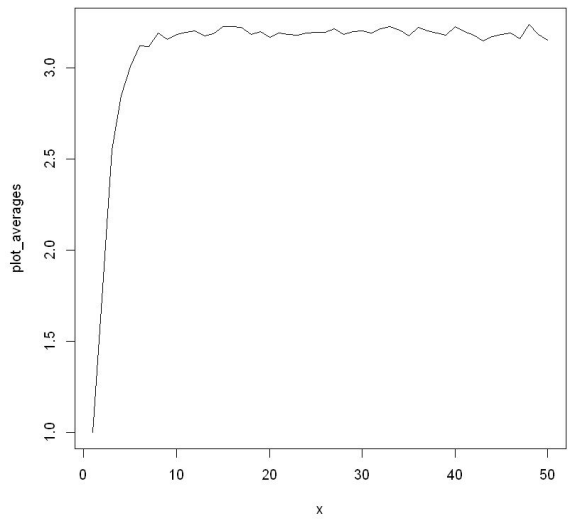
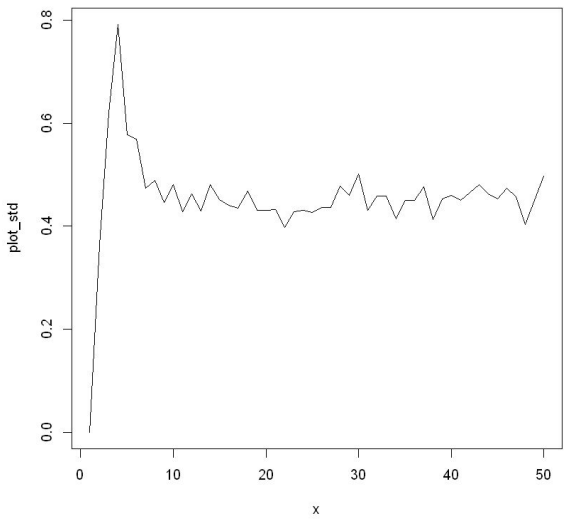
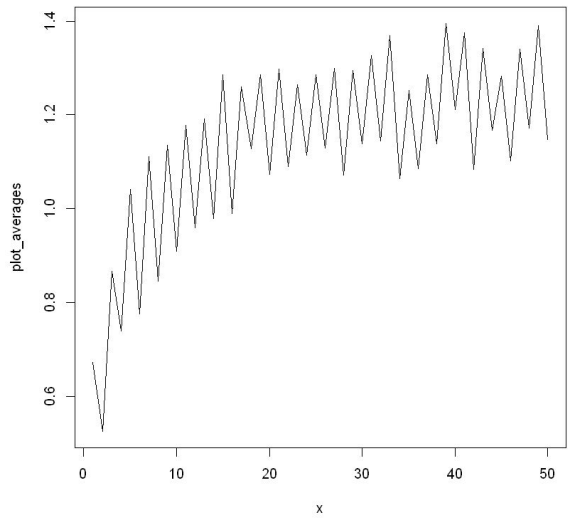
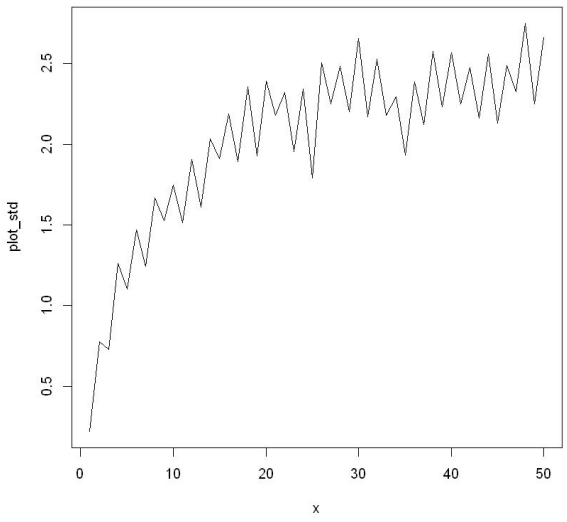
1. Random Walk on Erdos-Renyi Networks

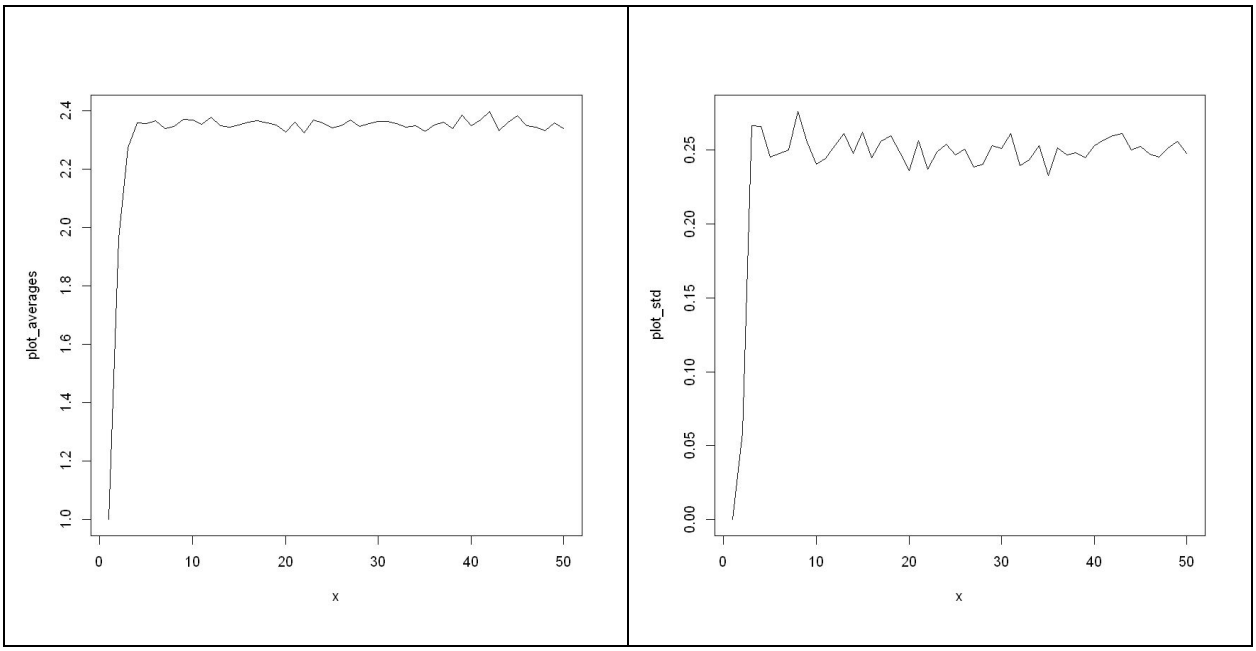
(a) Create an undirected random network with 1000 nodes, and the probability p for drawing an edge between any pair of nodes equal to 0.01.

We created the ER model using the erdos.renyi.game function. The network generated looks as follows:



(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) ⟨s(t)⟩ of the walker from his starting point at step t. Also, measure the standard deviation σ2(t) = ⟨(s(t) − ⟨s(t)⟩)2⟩ of this distance. Plot ⟨s(t)⟩ v.s. t and σ2(t) v.s. t. Here, the average ⟨·⟩ is over random choices of the starting nodes.

Average Distance (n=1000)	Standard deviation (n=1000)
	
Average Distance (n=100)	**Standard deviation (n=100)**
	
Average Distance (n=10000)	**Standard deviation (n=10000)**

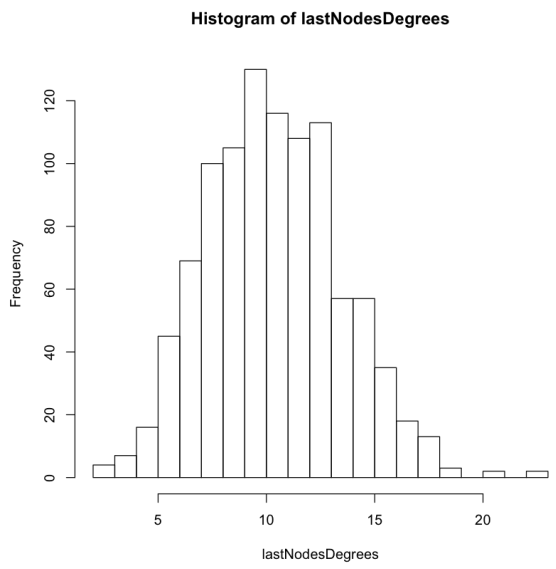
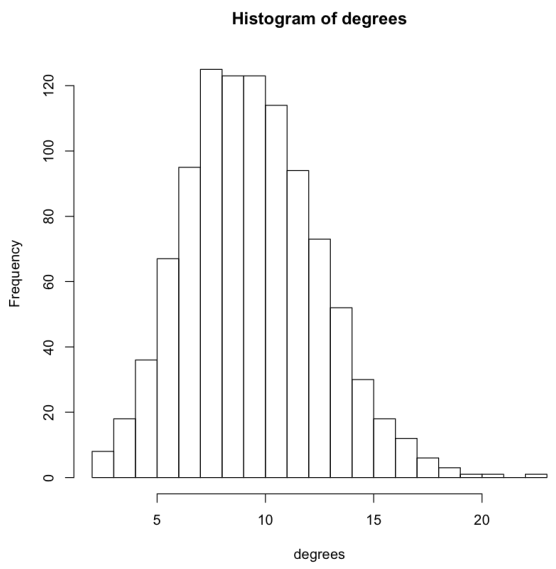


(c) Measure the degree distribution of the nodes reached at the end of the random walk. How does it compare to the degree distribution of graph?

The degree distribution for both the cases is plotted in the graphs below. We see that both the graphs look similar. To get more insights on this, we have fit both the degree distributions in a Poisson distribution and see the values of parameter lambda(here z) of the distribution. Poisson distribution:

$$P(deg(v) = k) = \frac{z^k e^{-z}}{k!}$$

where z is the average degree.

Degree distribution of nodes reached at the end of the random walk	Degree distribution of graph
	
Lambda(here z) from Poisson distribution fit for the above graph of degree distribution of nodes reached at end of random walk is **4.17**	Lambda(here z) from Poisson distribution fit for the above graph of degree distribution of nodes is **1.998**

(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 results for 100 and 10,000 nodes for the part (b) have been included under part (b) for easier comparison.

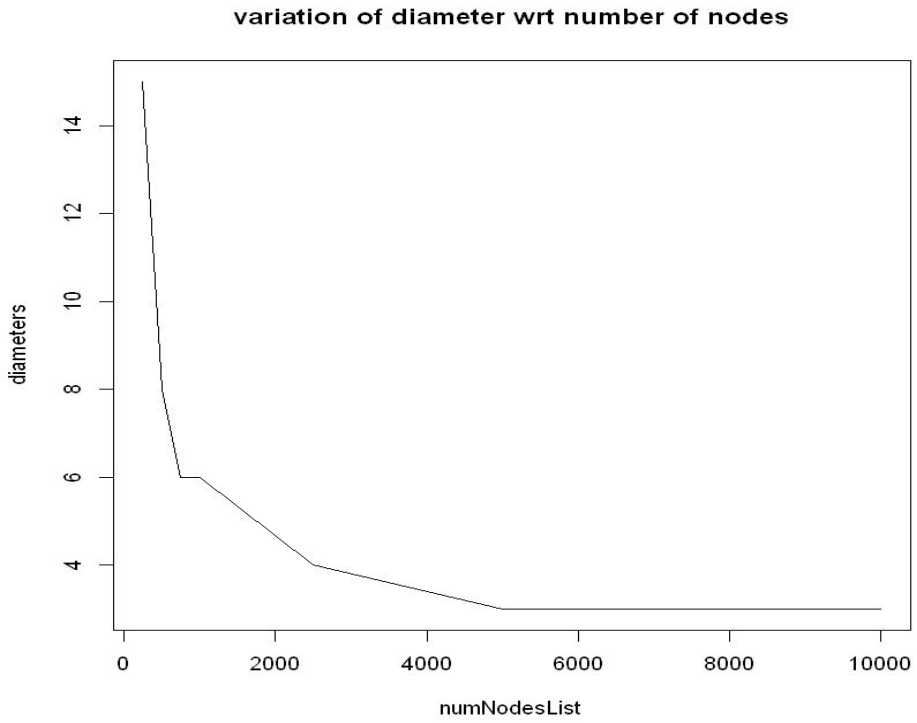
Role of diameter

The diameters of ER Graphs for nodes 100, 1000, 10000 and probability 0.01 are 11,5,3 respectively(in a decreasing order).

From the average distance and standard deviation graphs, we see that the graphs for n=1000 and n=10000 converge in less that 10 steps of the random walk while the graphs for n=100 do not converge even in 50 steps of the random walk.

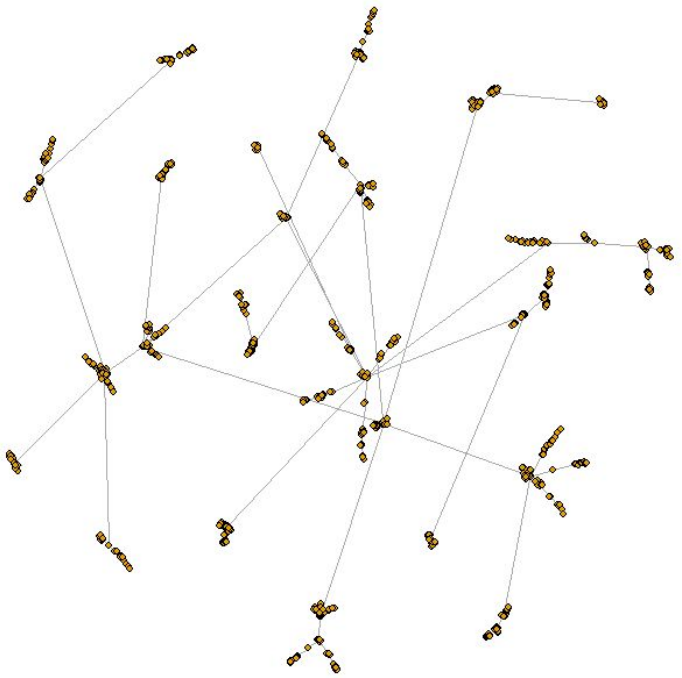
Considering both these facts, we can conclude that the number of steps at which the average distance and standard deviation graphs converge depend on the

diameter of the graph and that graphs with smaller diameters reach this convergence point in lesser steps of the random walk.

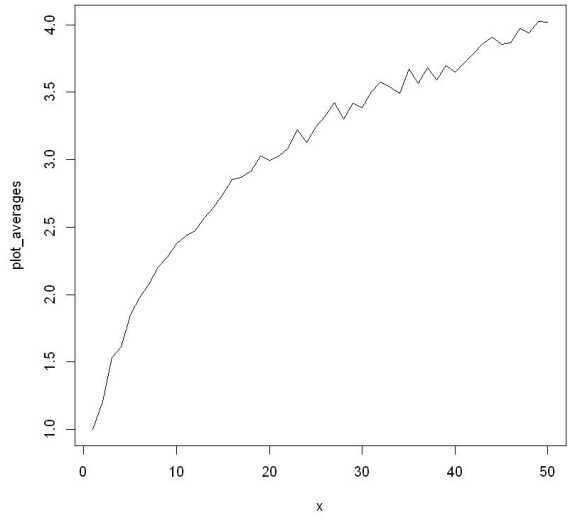
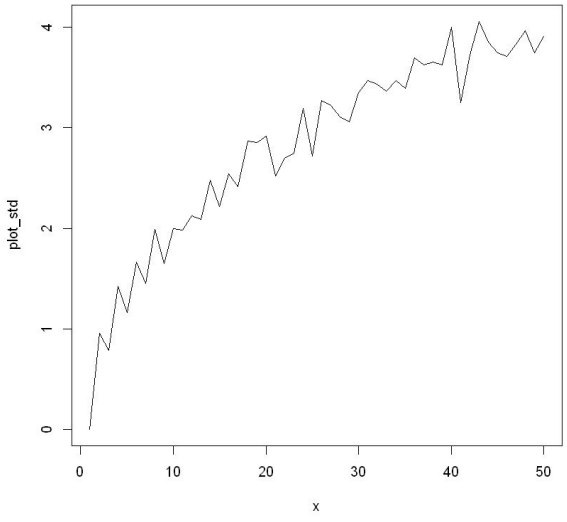


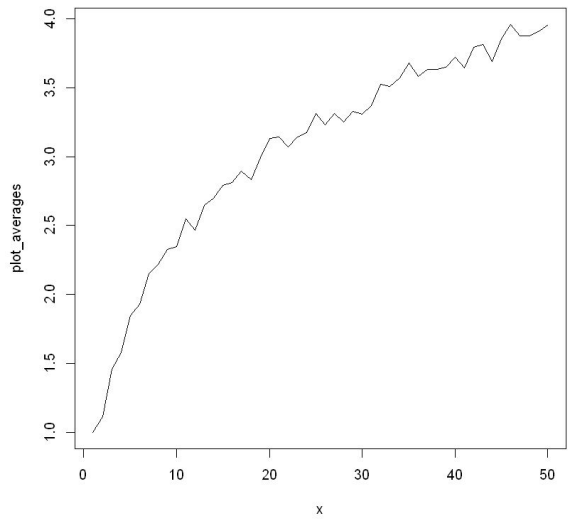
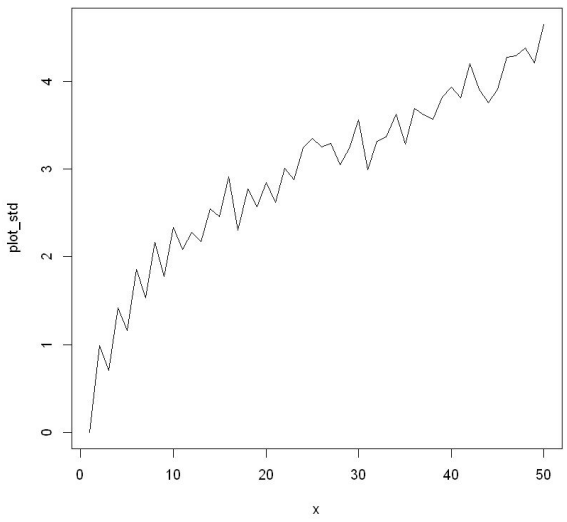
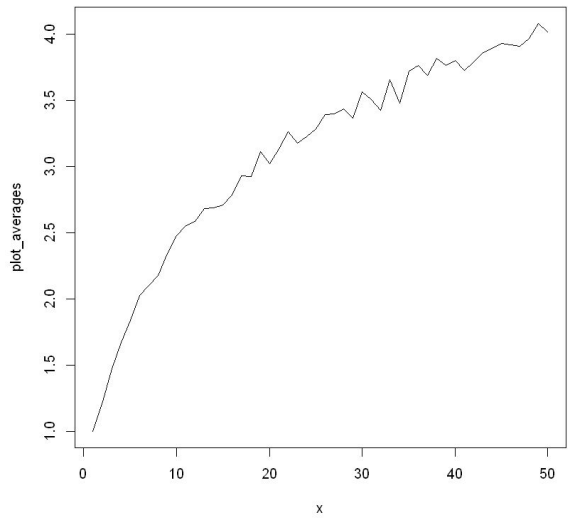
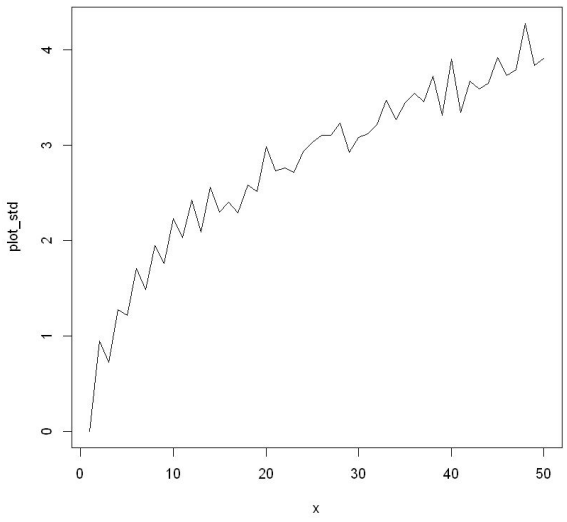
2. Random Walk on Networks with Fat-Tailed Degree Distribution

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



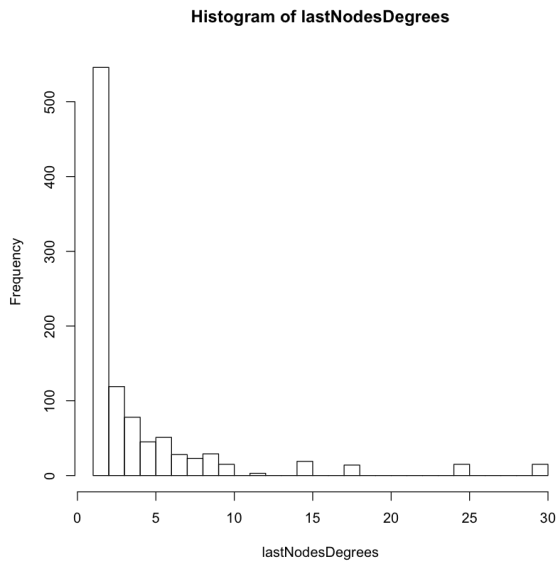
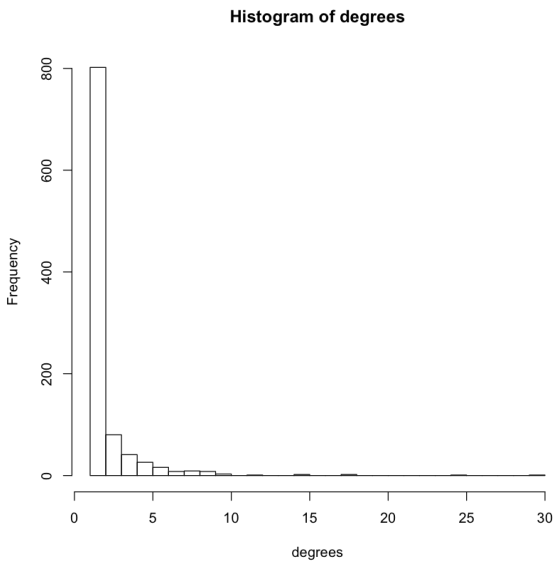
(b) Let a random walker start from a randomly selected node. Measure and plot ⟨s(t)⟩ v.s. t and σ2(t) v.s. T.

Average Distance (n=1000)	Standard deviation (n=1000)
	

Average Distance (n=100)	Standard deviation (n=100)
	
Average Distance (n=10000)	**Standard deviation (n=10000)**
	

(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?

Ans: The degree distribution for both the cases is plotted in the graphs below. We see that both the graphs look similar. To get more insights on this, we have fit both the degree distributions in a power law equation and see the exponent from the power law fit.

Degree distribution of nodes reached at the end of the random walk	**Degree distribution of graph**
	
Exponent from Power Law Fit for above graph of degree distribution of nodes reached at end of random walk is **2.057**	Exponent from Power Law Fit for above graph of degree distribution of nodes is **2.628**

(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?

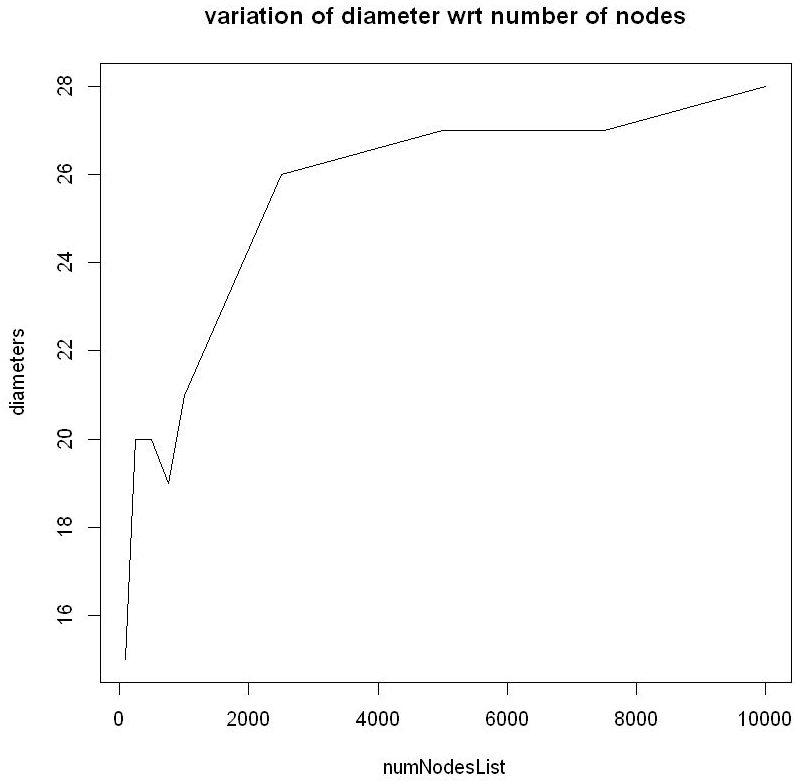
The results for 100 and 10,000 nodes for the part (b) have been included under part (b) for easier comparison.

Role of diameter

The diameters of preferential attachment network for nodes 100, 1000, 10000 and m=1 are 11,23,28 respectively(in an increasing order).

From the average distance and standard deviation graphs, we see that the graphs do not converge even in 50 steps. The graph for n=100 converges in approximately 200 steps. The graphs for n=1000, n=10000 require even more steps.

Considering both these facts, we can conclude that the number of steps at which the average distance and standard deviation graphs converge depend on the diameter of the graph and that graphs with smaller diameters reach this convergence point in lesser steps of the random walk. This conclusion is same as what was observed in ER graphs.



For preferential attachment graphs, the diameter of the graph is approximately equal to log N where N is the number of nodes of the graph.

3. PageRank

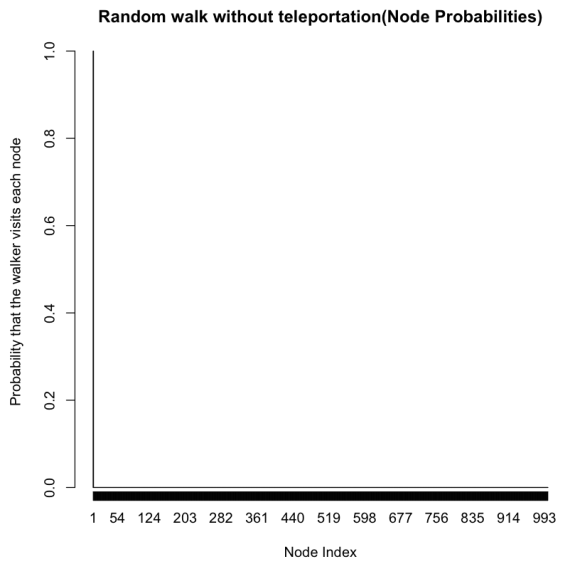
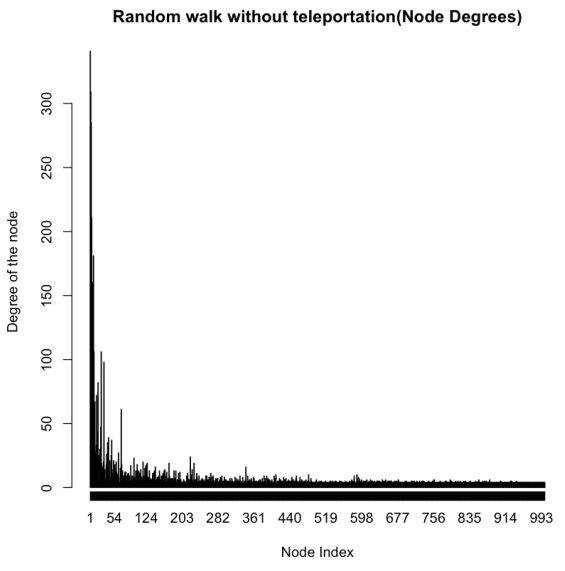
(a) Create a directed random network with 1000 nodes, using the preferential attachment model, where m = 4. Note that in this directed model, the out-degree of every node is m, while the in-degrees follow a power law distribution. Measure the probability that the walker visits each node. Is this probability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

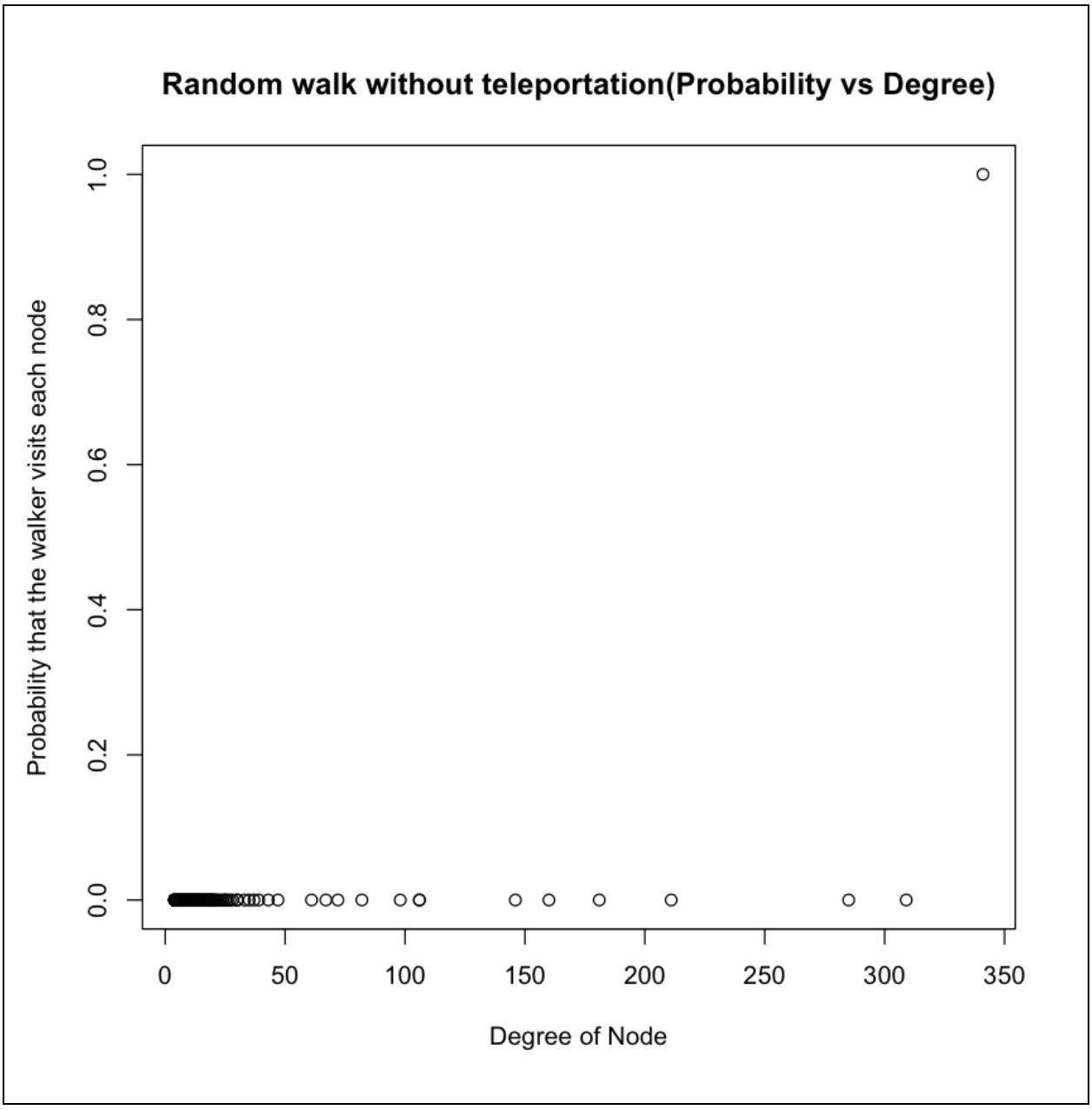
In this task, we have first created a directed graph with 1000 nodes with m=4.
We will use random walk on this graph to simulate PageRank.
We use a simple random walking technique in this task without any teleportations.

We perform random walks for multiple number of times(100) each time with a new random starting node. Each random walk is performed for 100 steps. We store how many times each node appears as the last node of the random walk.

For this task, we have used the basic random walk function and transition matrix without any modifications to them. As the walker proceeds in this random walk from node to node, he visits some nodes more often than others; intuitively, these are nodes with many links coming in from other frequently visited nodes. The idea behind PageRank is that pages visited more often in this walk are more important.

The probabilities that the walker visit each nodes and the degrees of the nodes are shown in the table below.

Probabilities of the nodes	**Degree of the nodes**
	



As this is a preferential attachment graph, the degree of the first node is high, it has no outgoing links and it has many incoming links. Similarly the degrees of initial nodes in the graph is more as compared to the nodes which are added at a later time.

Now if we observe the probability that the walker visits each node, we see that the probability of the first node is 100% while the rest of the nodes have a probability of 0%.

As the in-degree of first node is high, a large number of nodes will have this node as the next step in random walks. So there is a high chance that the walker comes to the first node in the random walk. As the out-degree is 0 and there is no teleportation, the walker has nowhere to go and hence once the walker lands on the first node, it remains there.

Hence, in all the iterations, the last node of the walk is the first node and hence it has a probability of 100%. Thus, we see that this high probability is a direct effect of the high degree of the node.

(b) In all previous questions, we didn't have any teleportation. Now, we use a teleportation probability of α = 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?

In this task, we do a random walk with teleportations. We perform random walks for multiple number of times(100) each time with a new random starting node. Each random walk is performed for 100 steps. We store how many times each node appears as the last node of the random walk.

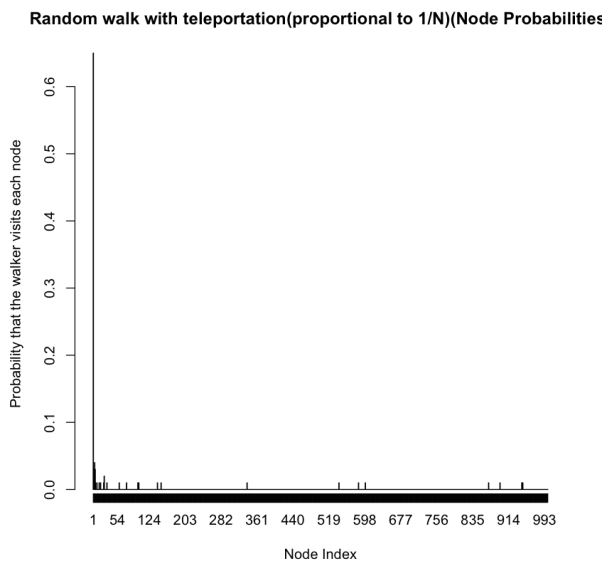
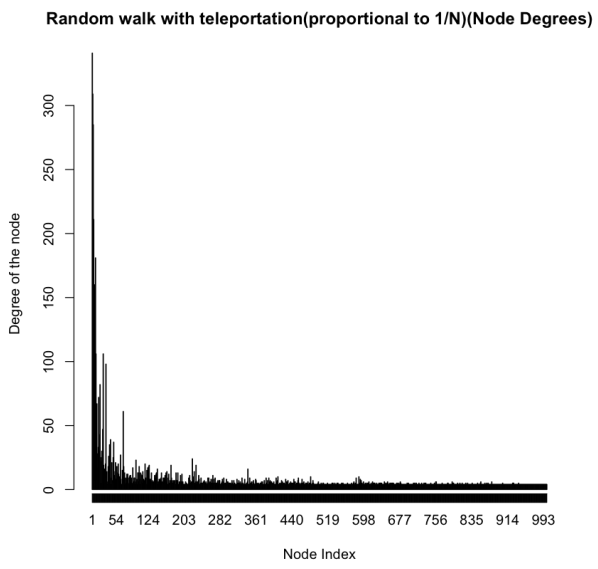
We teleport randomly for 15% of the time depending on the value of α (here =0.15). Whenever we want to randomly teleport to a node, we randomly select any node.i.e. All nodes have equal probability of 1/N to be picked to be teleported to.

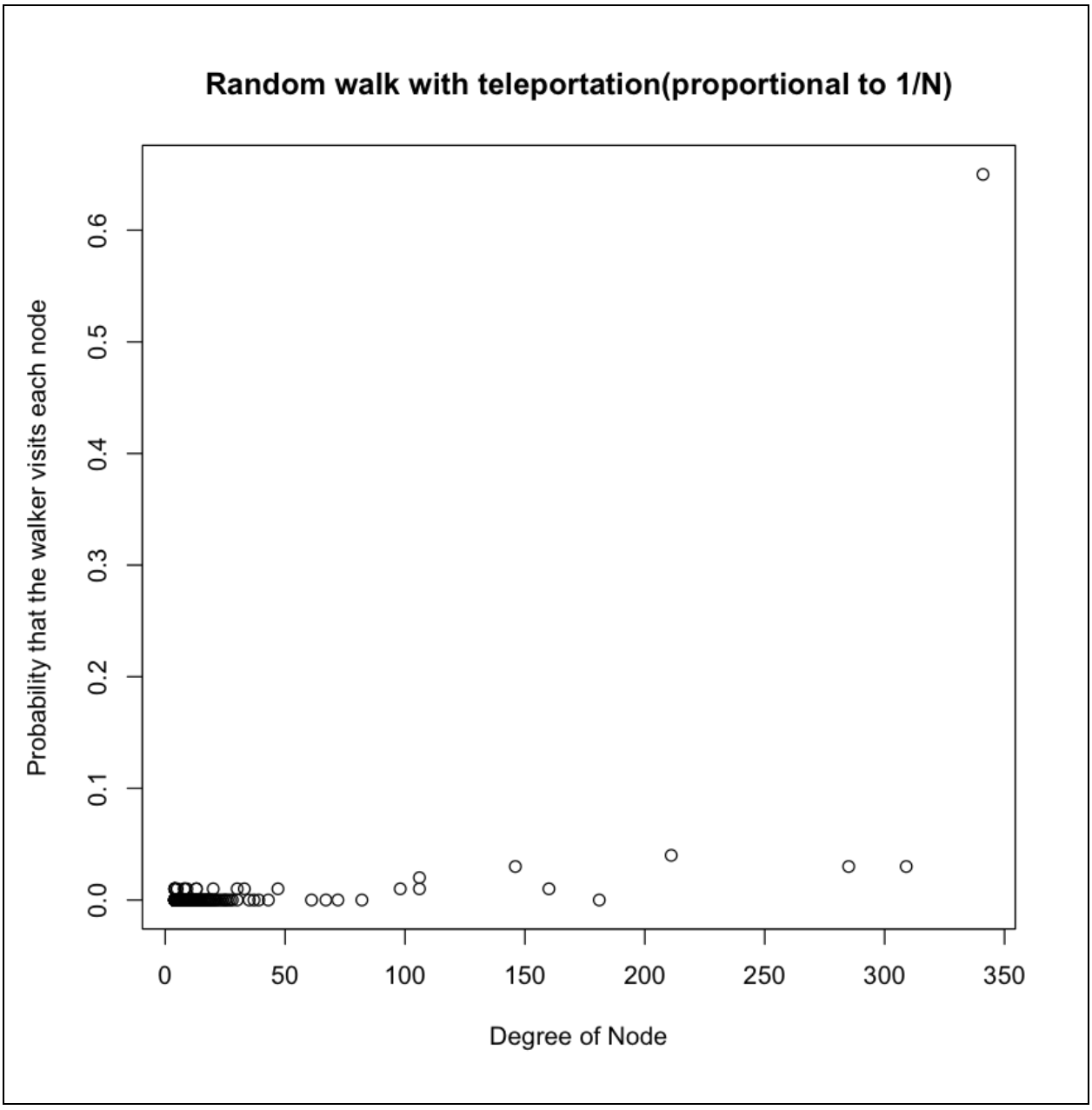
For this task, we have tried with both a modified version of the random walk function and also using a modified transition matrix.

We modify the transition matrix in the following way:
new_transition_matrix = d*transition_matrix + (1-d)*(1/N)*ones
Where,

d=1-alpha
ones is an NxN matrix of 1s

The effect of this on the probabilities that the walker visit each nodes and the degrees of the nodes are shown in the table below.

Probabilities of the nodes	Degree of the nodes
	



As this is a preferential attachment graph, the degree of the first node is high, it has no outgoing links and it has many incoming links. Similarly the degrees of initial nodes in the graph is more as compared to the nodes which are added at a later time.

Now if we observe the probability that the walker visits each node, we see that the probability of the first node is still high. It is not 100% like the previous part and is about 0.15 but is still very high compared to the other nodes. SImilarly the other

nodes which previously had 0% probability now have a very low but non-zero probability.

As the in-degree of first node is high, a large number of nodes will have this node as the next step in random walks. So there is a high chance that the walker comes to the first node in the random walk. As the out-degree is 0, once the walker lands on this node, it will stay there until a teleportation occurs. Hence, in many iterations, the last node of the walk is the first node and hence it has a high probability. When teleportation takes place, random walk again continues from a new node and eventually lands on the first node. Because of this teleportation, nodes adjacent to the first node also have a higher probability now. Sometimes the last step of random walk might be a teleportation step and thus any random node might become the last node of the random walk. These are the nodes have a very low probability.

Thus, we see that this high probability of first node is a direct effect of the high degree of the node. The other nodes leading to the first node also have some probabilities. The very low probabilities of some nodes is the effect of teleportation.

4. Personalized PageRank

(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 (as- sume 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 α = 0.15. Compare the results with 3(b).

Ans: In this task, we do a random walk with teleportations. We perform random walks for multiple number of times(100) each time with a new random starting node. Each random walk is performed for 100 steps. We store the degree of the last node of each random walk.

Here the rules for teleportation are slightly different. We still teleport randomly 15% of the time however the selection of which node to teleport to is different. Instead of teleporting to any random node, we teleport to nodes based on their pagerank values. We have modified the transition matrix based on the pagerank equation in the following way.

The equation for page rank can be written as,

$$PR = d*T*PR + (1-d)*PR$$

$$\Rightarrow PR = [d*T + (1-d)*I]*PR$$

So we have modified the transition matrix in the similar way to,

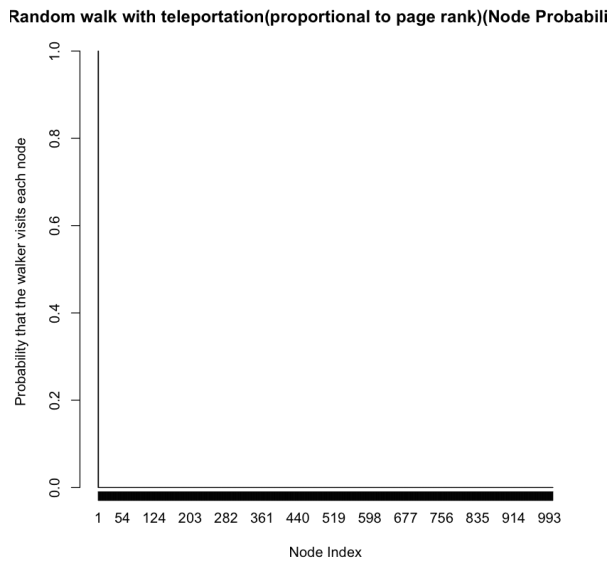
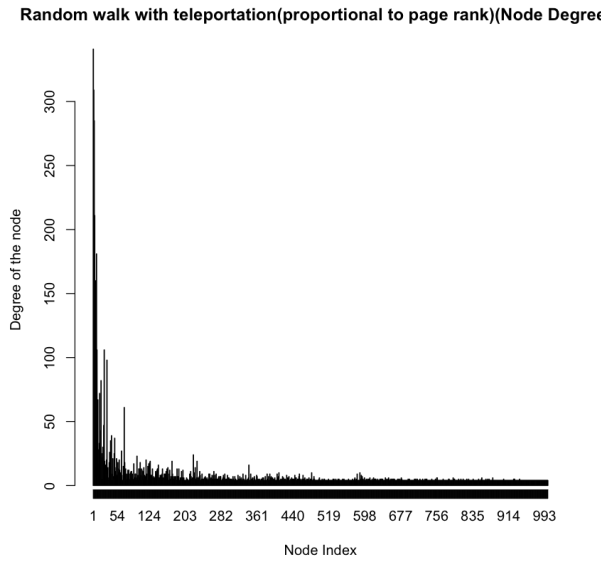
new_transition_matrix = d*transition_matrix + (1-d)*I

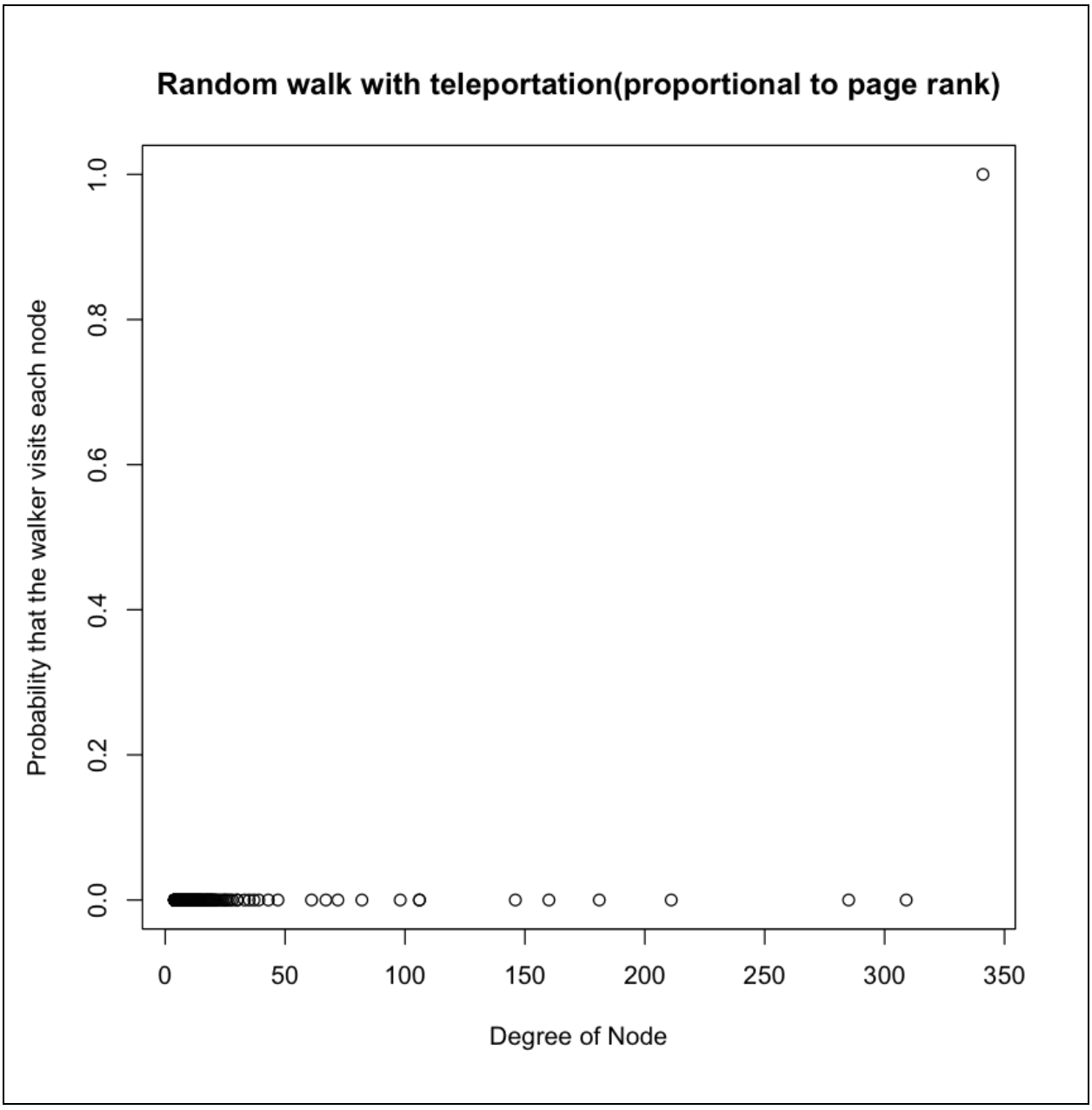
Where,

d is (1-alpha)

I is NxN identity matrix

The effect of this on the probabilities that the walker visit each nodes and the degrees of the nodes are shown in the table below.

Probabilities of the nodes	Degree of the nodes
	



As this is a preferential attachment graph, the degree of the first node is high, it has no outgoing links and it has many incoming links. Similarly the degrees of initial nodes in the graph is more as compared to the nodes which are added at a later time.

Now if we observe the probability that the walker visits each node, we see that the probability of the first node is 100% while the other nodes have a 0% probability.

As the in-degree of first node is high, a large number of nodes will have this node as the next step in random walks. So there is a high chance that the walker comes to the first node in the random walk. As the out-degree is 0, once the walker lands on this node, it will stay there until a teleportation occurs. When teleportation takes place the node to be chosen to be teleported to is based on the page rank equation. In this equation, we have modified the transition matrix in such a way that when teleportation takes place, the node prefers to teleport to itself or rather stay where it is. This happens because we have added extra factor to the diagonal of the transition matrix. The equations for this can be written as,

PR = d*T*PR + (1-d)*PR

=> d*PR = d*T*PR

=> PR = T*PR which is the steady state equation for PageRank

This effectively means that in (1-d)% of the cases, the walker chooses to do nothing, and we are left with only d% of the cases in which the walker does not choose teleportation. In a steady state, d% of the cases is equal to 100% of the cases. So this case effectively acts as a case without any teleportation.

The out degree of the first node(node with highest degree) is 0 so the only way walker can go to another node is through teleportation. However now the teleportation also favors to stay at the same node. Hence, once the walker lands on the first node, it will remain there till the end of the walk. Hence, in all the iterations, the last node of the walk is the first node and hence it has 100 % probability.

Thus, we see that this high probability of first node is a direct effect of the high degree of the node and teleportation based on the pagerank values.

(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?

Ans: In this task, we do a random walk with teleportations. We perform random walks for multiple number of times(100) each time with a new random starting node. Each random walk is performed for 100 steps. We store the degree of the last node of each random walk.

Here the rules for teleportation are slightly different. We still teleport randomly 15% of the time however the selection of which node to teleport to is different. Instead of teleporting to any random node or based on pagerank values, we now teleport to either of the two nodes with median pageranks. For this, we have used the pagerank vector that we have found from part 3(a). This vector has 100% probability

for the first node and 0% probability for the remaining nodes. Hence any two of the nodes having 0% pagerank value will be our median nodes. Through a random sampling, we have taken node 668 and 179 as our median nodes.

For this task, we have tried with both a modified version of the random walk function and also using a modified transition matrix.

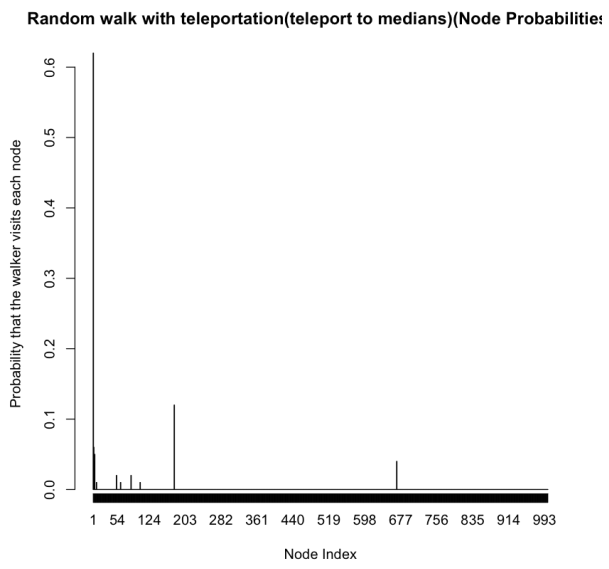
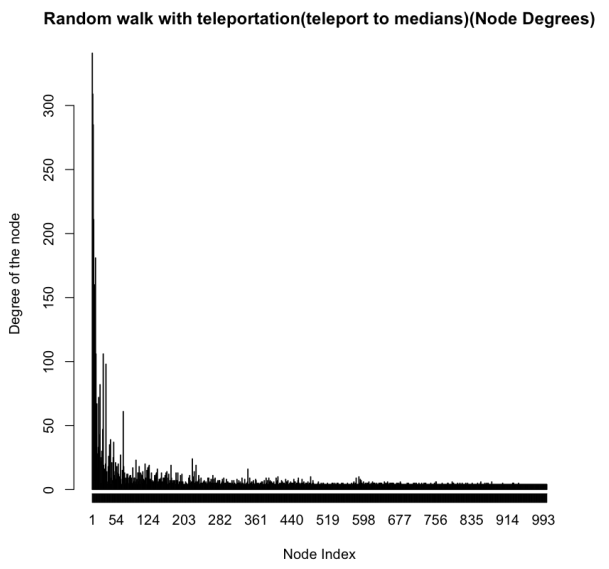
We modify the transition matrix in the following way:
new_transition_matrix = d*transition_matrix + (1-d)*R
Where,

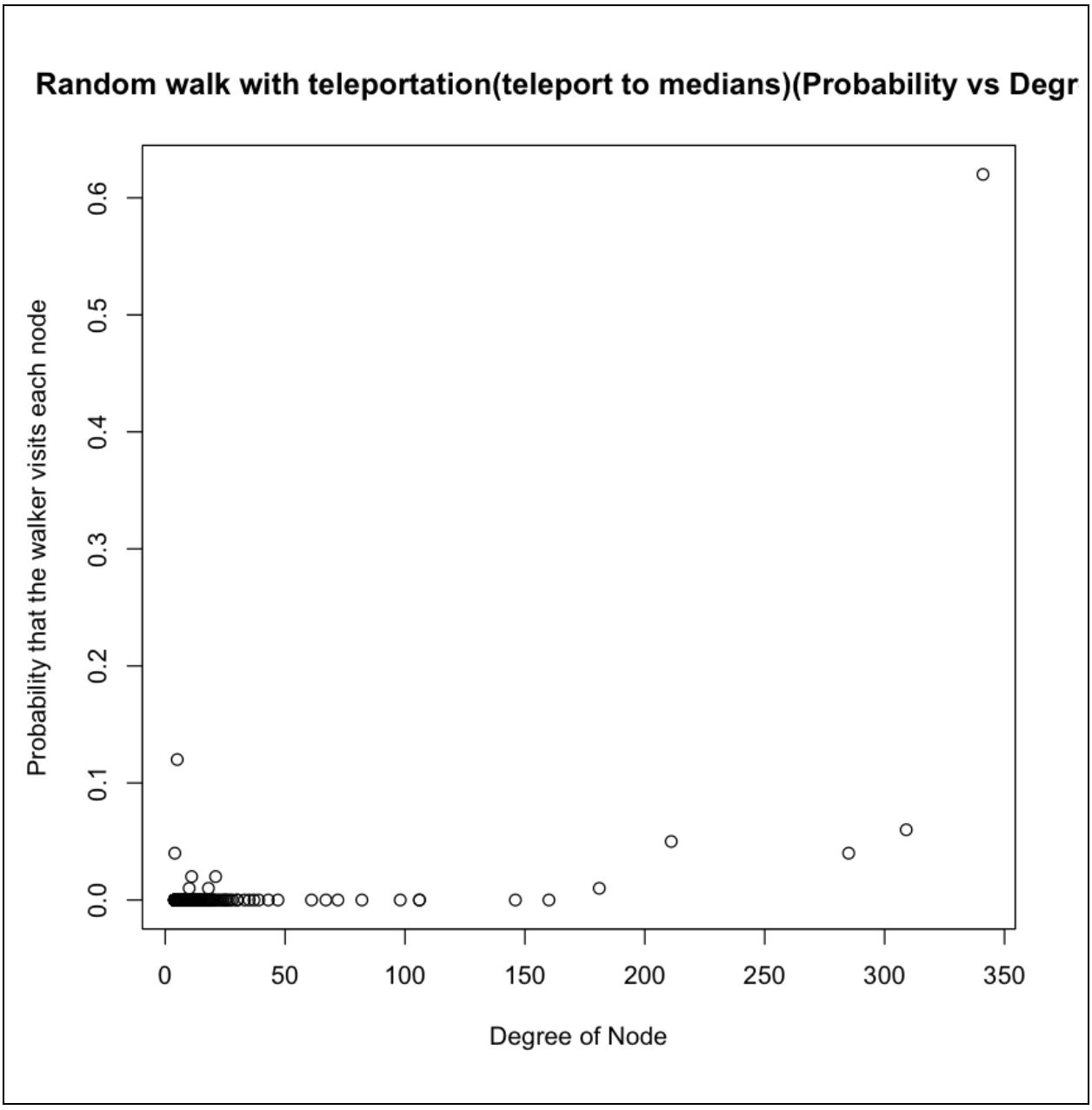
d=1-alpha

R = NxN matrix of [0 0 ... 0.5 ... 0.5 ...0 0
0 0 ... 0.5 ... 0.5 ...0 0
.. ..
.. ..
.. ..
0 0 ... 0.5 ... 0.5 ...0 0
0 0 ... 0.5 ... 0.5 ...0 0]

With value 0.5 at median nodes.

The effect of this on the probabilities that the walker visit each nodes and the degrees of the nodes are shown in the table below.

Probabilities of the nodes	Degree of the nodes
	
Medians = 668 (degree=4), 179(degree=5)	



As this is a preferential attachment graph, the degree of the first node is high, it has no outgoing links and it has many incoming links. Similarly the degrees of initial nodes in the graph is more as compared to the nodes which are added at a later time.

Now if we observe the probability that the walker visits each node, we see that the probability of the first node is still the highest though not 100%. We now also see 2 nodes, our median nodes having higher probability values than before summing up to approximately 0.15 which is our teleportation factor Nodes adjacent to these

median nodes also have some higher values now. The rest of the nodes have 0% probabilities. As the in-degree of first node is high, a large number of nodes will have this node as the next step in random walks. So there is a high chance that the walker comes to the first node in the random walk. As the out-degree is 0, once the walker lands on this node, it will stay there until a teleportation occurs. Hence, in many iterations, the last node of the walk is the first node and hence it has a high probability compared to other nodes. When a teleportation takes place either of the median nodes is chosen. Hence is some iterations this teleportation step might be the last step of the random walk and hence this median nodes have higher probability values than earlier cases. Similar is the explanation for nodes adjacent to these median nodes. Rest of the nodes have a rare chance of being visited only during the initial random start.

Thus, we see that the high probability of first node is a direct effect of the high degree of the node. The probabilities of the median nodes is higher due to the teleportation.

(c) More or less, (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 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?

This self-reinforcement behavior of teleporting to trusted web pages is what happens in real life and this has been simulated in the previous part through having median nodes. We still teleport randomly 15% of the time however the selection of which node to teleport to is different. Instead of teleporting to any random node or based on pagerank values, we now teleport to either of the two nodes with median pageranks with 50% probability for each.
The equation for page rank can be written as,

$$PR = d*T*PR + (1-d)*PR$$

$$\Rightarrow PR = [d*T + (1-d)*I]*PR$$

To adjust the PageRank equation to take this self-reinforcing behavior into account, we modify the above equation as,

$$PR = [d*T + (1-d)*R]*PR$$

Where,

d=1-alpha

R = NxN matrix of [0 0 ... 0.5 ... 0.5 ...0 0

0 0 ... 0.5 ... 0.5 ...0 0

.. ..

.. ..

.. ..

0 0 ... 0.5 ... 0.5 ...0 0

0 0 ... 0.5 ... 0.5 ...0 0]

With value 0.5 at median nodes.