Project 1

Winston Hu 206069558
Blaine Arihara 504566191

ECE232E

July 9, 2023

Part 1. Generating Random Networks

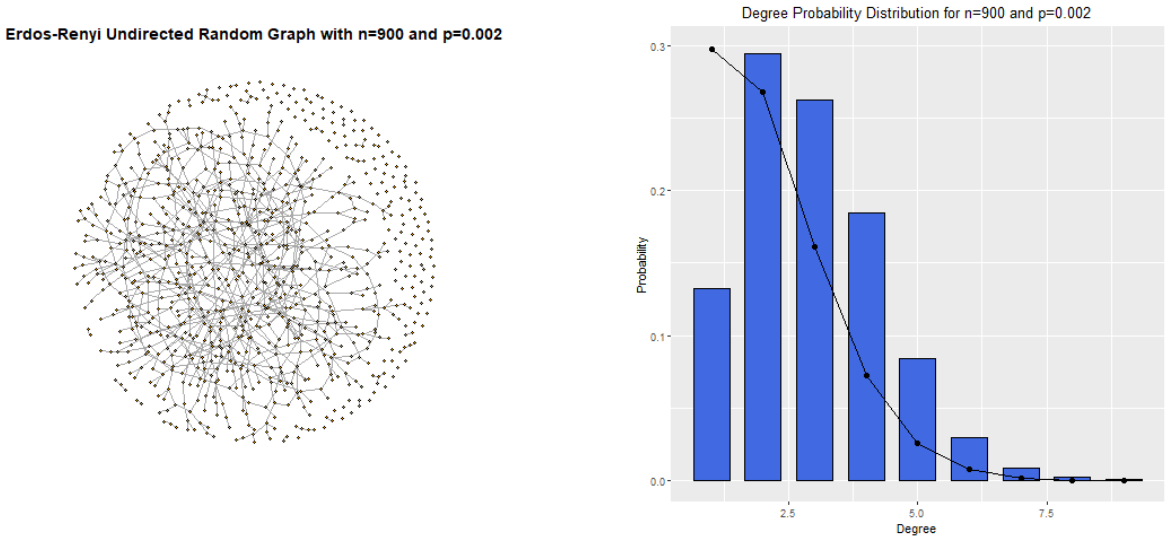
Question 1. Create random networks using the Erdos-Renyi (ER) model

Part (a). Create undirected random networks with n = 900 nodes, and the probability p for drawing an edge between two arbitrary vertices 0.002, 0.006, 0.012, 0.045, and 0.1. Plot the degree distributions. What distribution (linear/exponential/gaussian/binomial or something else) is observed? Explain why. Also, report the mean and variance of the degree distributions and compare them to the theoretical values.

The Erdos-Renyi model is a model that starts with the existence of set number of unconnected nodes. It then takes one node and looks at its pairing with every other node in the set and creates an edge with a paired node according to a fixed probability. After every pair is considered, the same process is carried out for another node and all of its unique pairs. The process is repeated until every node has had a chance to have an edge created with every other node in the set. This model is used to generate the following undirected random graphs with the following probabilities of edge-formation: p = 0.002, 0.006, 0.012, 0.045, and 0.1. For example, p = 0.002 means that as every pair of nodes is considered, there is a 0.2% chance that an edge is created between the nodes. This project was carried out in the programming language of R, which contains powerful libraries and in-built functions to facilitate statistical processing. The *igraph* library is installed and used to simulate and generate many of the networks investigated in this project. The *erdos.renyi.game()* is used to create all the graphs of n = 900 nodes for Question 1.

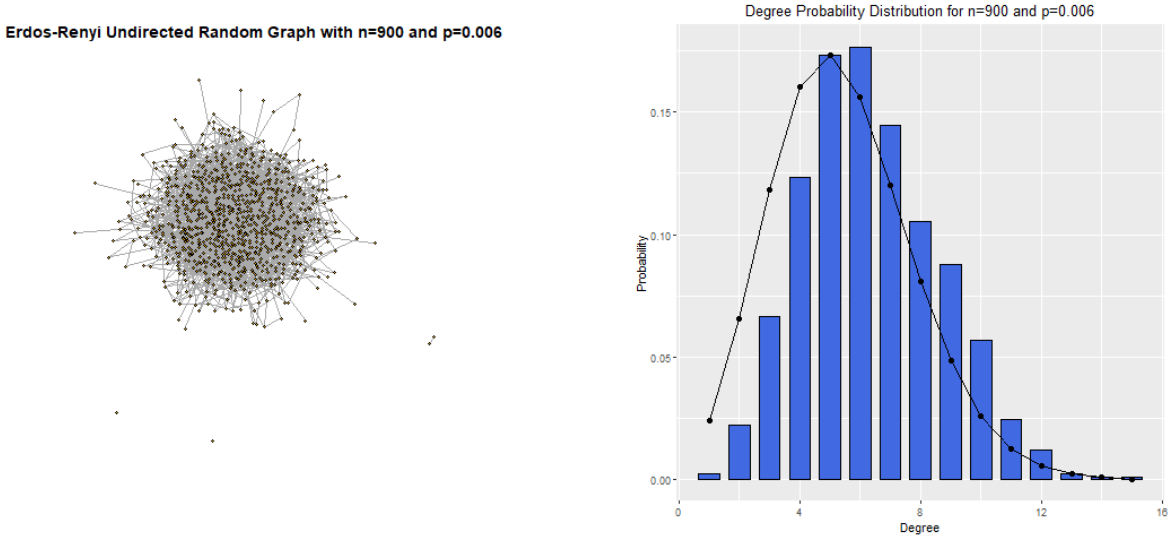
Below are shown five pairs of plots in Figures 1-5 corresponding to the five probabilities that illustrate a visualization of the random undirected networks that they create and their probability degree distribution. It can be seen that in Figure 1 there is a sparsely connected network that is created due to the low probability or likelihood that an edge would be created between each arbitrary pair of nodes. As the probability of edge-formation increases, the networks are observed to grow in density as edges are increasingly likely to be created. The degree distribution in Figure 2b starts to show characteristics consistent with a binomial distribution, where the black line overlaying the distribution represents the theoretically expected probabilities at each degree given the number of nodes and the probability of successes or edge-formation, according to equation 1 below:

$$P_{(x:n,p)} =^{n} C_x P^x (1-p)^{n-x} \tag{1}$$



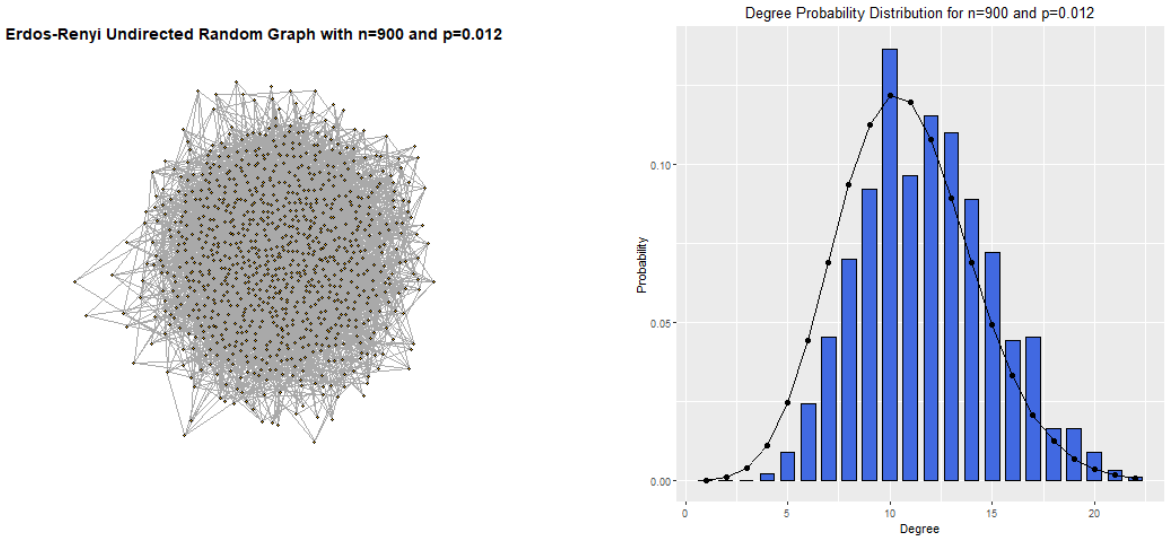
(a) (b)

Figure 1: Undirected random network generated using the Erdos-Renyi Model with n = 900 and p = 0.002. (a) network visualization. (b) Probability density distribution with theoretical binomial distribution.



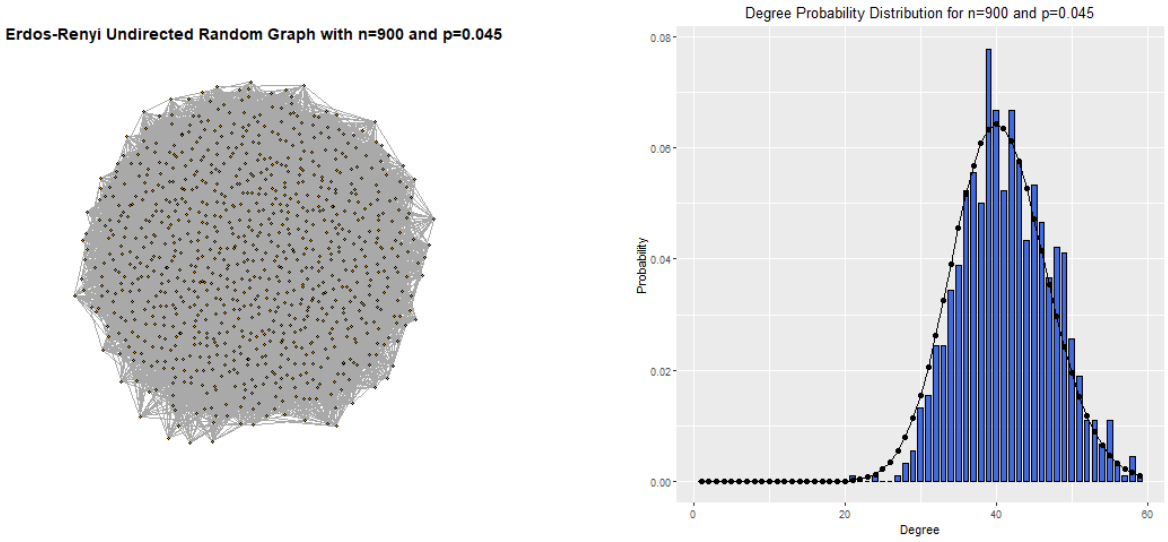
(a) (b)

Figure 2: Undirected random network generated using the Erdos-Renyi Model with n = 900 and p = 0.006. (a) network visualization. (b) Probability density distribution with theoretical binomial distribution.



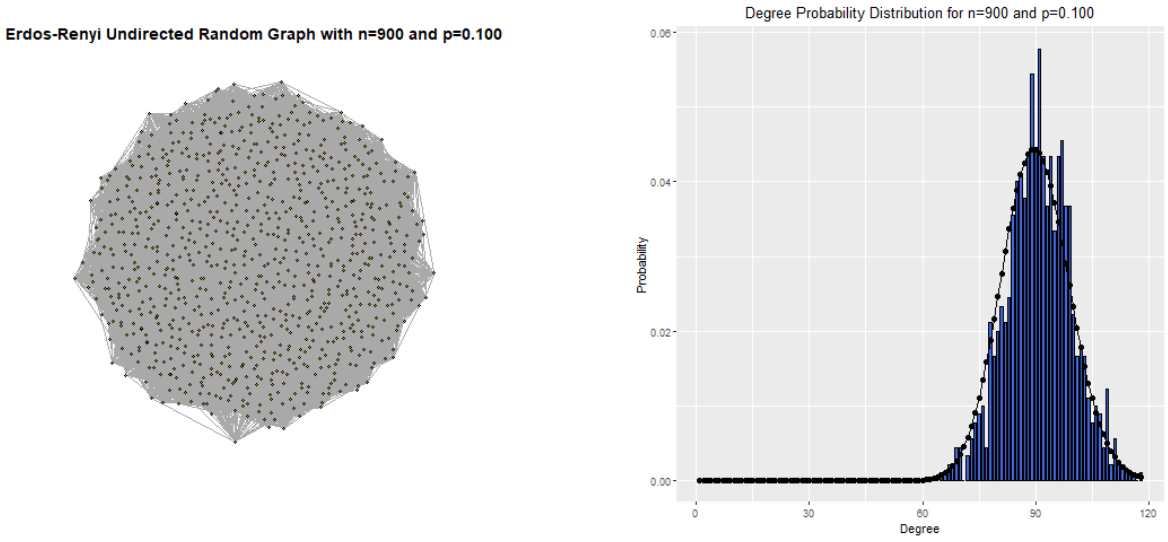
(a) (b)

Figure 3: Undirected random network generated using the Erdos-Renyi Model with n = 900 and p = 0.012. (a) network visualization. (b) Probability density distribution with theoretical binomial distribution.



(a) (b)

Figure 4: Undirected random network generated using the Erdos-Renyi Model with n = 900 and p = 0.045. (a) network visualization. (b) Probability density distribution with theoretical binomial distribution.



(a) (b)

Figure 5: Undirected random network generated using the Erdos-Renyi Model with n = 900 and p = 0.100. (a) network visualization. (b) Probability density distribution with theoretical binomial distribution.

The theoretical mean and variance are also calculated according to equation 2 and equation 3. These theoretical mean and variances are calculated for each value of p and compared to their simulated mean and variance in Table 1. It can be seen that the simulated and theoretical values closely agree, suggesting that the probability degree distribution of the Erdos-Renyi model follows a binomial distribution. This is because the formation of edges between nodes is equally applied and equally probable for every possible pair of nodes, consistent with the binomial distribution assumptions that every trial is mutually independent of each other and has the same probability of success.

$$\sigma = n \cdot p \tag{2}$$

$$\sigma^2 = n \cdot p \cdot (1 - p) \tag{3}$$

Table 1: Observed vs. theoretical mean and variance of the previously generated networks for n = 900.

p	**Simulated mean**	**Theoretical mean**	**Simulated variance**	**Theoretical variance**
0.002	1.94	1.80	1.85	1.80
0.006	5.36	5.40	5.14	5.37
0.012	10.86	10.80	10.51	10.67
0.045	40.47	40.50	36.61	38.68
0.100	89.94	90.00	75.18	81.00

Part (b). For each p and n = 900, 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?

To determine if all random realizations of the ER networks are connected, the *is.connected()* function is used to check if a generated graph is connected. To numerically estimate the probability that a generated network is connected, 1000 graphs are generated and checked for being connected or not. The sum of the graphs connected are divided by the total to find the probability that a randomly generated graph is connected. The results are contained in Table 2. It can be seen that for probabilities of 0.006 and below, there are roughly only 2% of graphs that will be fully connected, which is demonstrated in Figure 2a where a few nodes are visibly disconnected. Probability of 0.012 shows a 98% of full connection, with higher probabilities having numerically estimated 100% connection.

Also shown in Table 2 are the characteristics of the Giant Connected Component (GCC) of the last iteration of the graph for each p. The number of nodes, edges, and diameter of that GCC are tabulated. At probability of 0.006, the GCC was composed of 900 nodes, indicating that every node in the entire set had some path to every other node. While the number of nodes of the GCC reaches a maximum at the number of nodes in the set, its edges increases as more edges are created between the nodes within the GCC. The GCC diameter also decreases as the GCC becomes more dense, agreeing with the conceptual understanding that as more connections are created amongst nodes comprising the GCC, the lower is the least number of steps that it would take to reach one node to another.

Table 2: Probability of graphs being connected and characteristics of their Giant Connected Component (GCC)

p	Probability Connected	Is Connected?	GCC # of Nodes	GCC # of Edges	GCC Diameter
0.002	0.00	False	626	732	26
0.006	0.02	False	895	2437	9
0.012	0.98	True	900	4806	5
0.045	1.00	True	900	17985	3
0.100	1.00	True	900	40115	3

Part (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 $p = O(\frac{1}{n})$ and $p = O(\frac{ln(n)}{n})$. For n = 900, sweep over values of p from 0 to a pmax that makes the network almost surely connected and create 100 random networks for each p. pmax should be roughly determined by yourself. Then scatter plot the normalized GCC sizes vs p. Plot a line of the average normalized GCC sizes for each p along with the scatter plot.
i. 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?
ii. Empirically estimate the value of p where the giant connected component takes up over 99% of the nodes in almost every experiment.

To plot the average normalized GCC sizes against the probability of edge-formation, a vector of probabilities spanning from 0 to $p_{max} = 0.018$ with a step size of $p_{step} = 0.0001$ was created. Then for every probability p, 100 iterations were ran to calculate the average GCC size and plotted against each other to generate the plot shown in Figure 6. Given that $n = 900$, the boundary values provided in the problem statement are $p = O(\frac{1}{n}) = 0.00111$ and $p = O(\frac{ln(n)}{n}) = 0.00756$. These values accurately bound the interesting non-linear behavior of GCC size vs. edge-formation probability observed in Figure 6. For question (i), it can therefore be proposed that the GCC starts to develop around the p value of 0.001. The criterion of emergence is determined by visually inspecting where the GCC sizes begin to starkly increase. For (ii), it can also be visually estimated that the GCC begins to take over 99% of the nodes around where the curve starkly plateaus. This probability of where GCC size first surpasses 0.99 of the network size is found to be 0.0052.

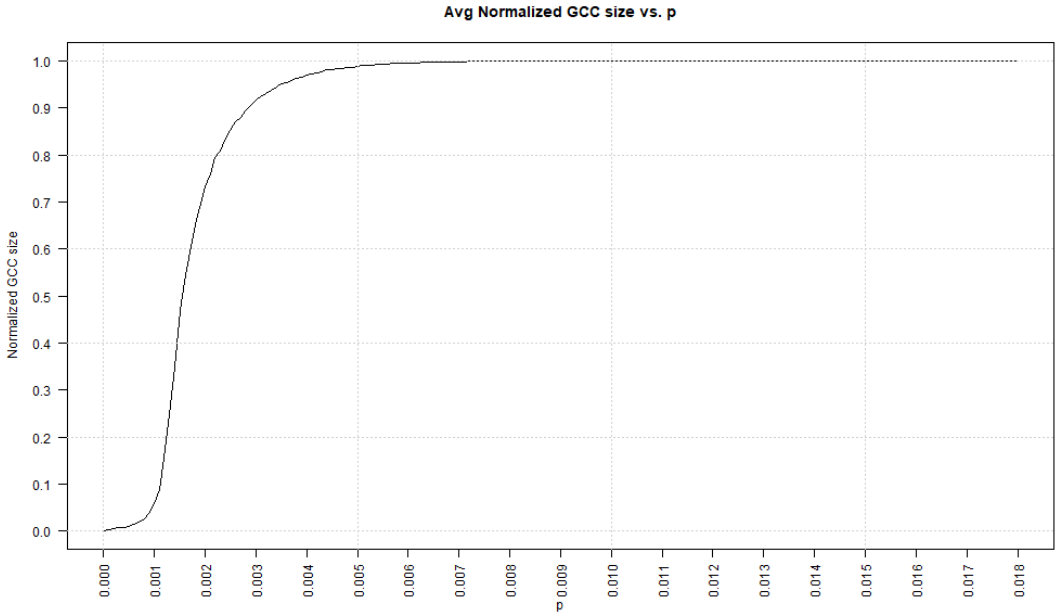


Figure 6: Average GCC size vs. probability of edge formation

Part (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 = 1.15, 1.25, 1.35, and show the results for these three values in a single plot.
iv. What is the relation between the expected GCC size and n in each case?

To generate the plots for this question, the probabilities of edge-formation was dependent upon n according to $p = \frac{c}{n}$, where n ranged from 100 to 10000, and 100 iterations were ran for every value of p to calculate its respective expected GCC size. The parameter c essentially served as a scalar that increased the general probabilities of edge-formation as c increased. This can be seen from the general increase in GCC sizes from Figure 7 to Figure 9 where the GCC sizes at the 10000th node increases in orders of magnitude from c = 0.5 to 1.35. As n increase for any value of c, the GCC sizes also increase as a result of the dominating effect of increasing n despite a decreasing p. n acts as an upper bound to the GCC size, because the GCC size cannot grow any bigger than the number of nodes in the network. And as n increases in size, there becomes more nodes available for the GCC to accrete. But as n increases, p decreases according to $p = \frac{c}{n}$. However, p does not decrease fast enough to impede the generation of a GCC. It does however have the interesting effect of linearizing the relationship between GCC and the number of nodes in a network.

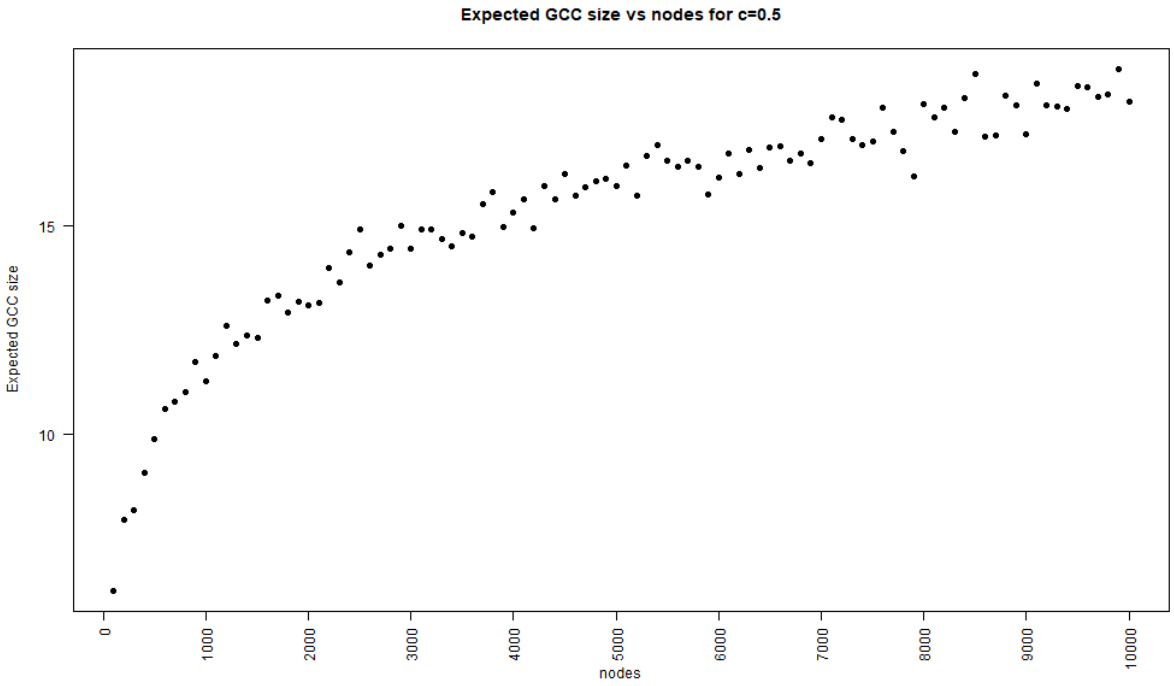


Figure 7: Expected GCC size vs. Number of Nodes
c = 0.5

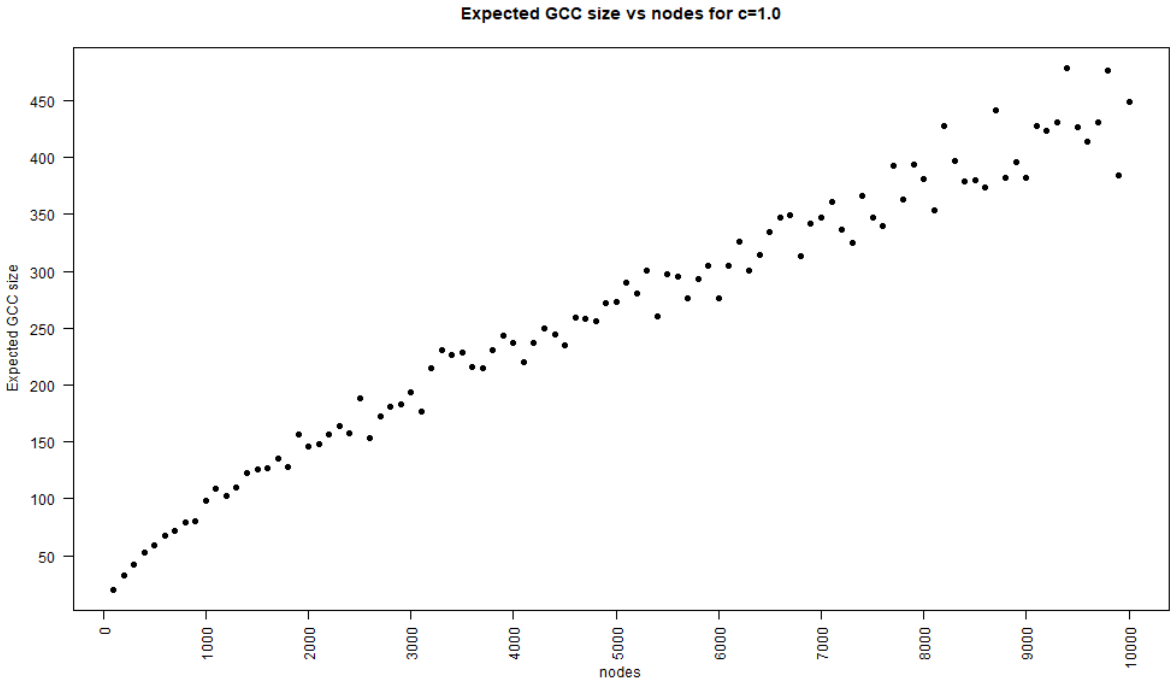


Figure 8: Expected GCC size vs. Number of Nodes
c = 1.0

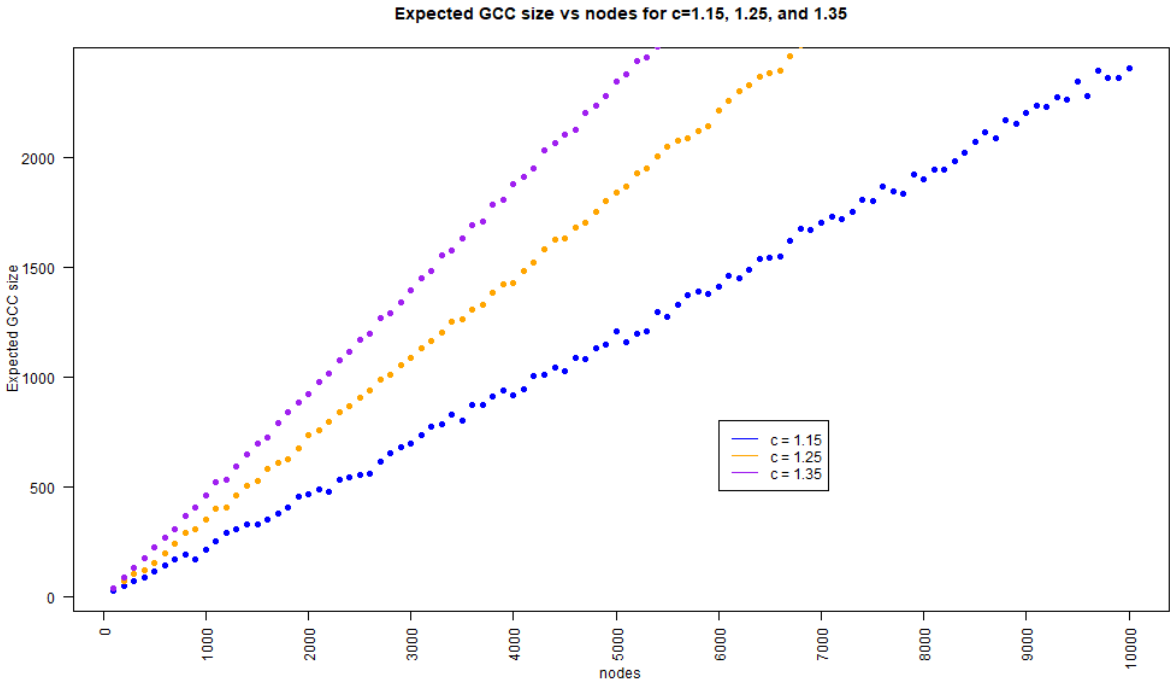


Figure 9: Expected GCC size vs. Number of Nodes
c = 1.15, 1.25, and 1.35

Question 2. Create networks using preferential attachment model

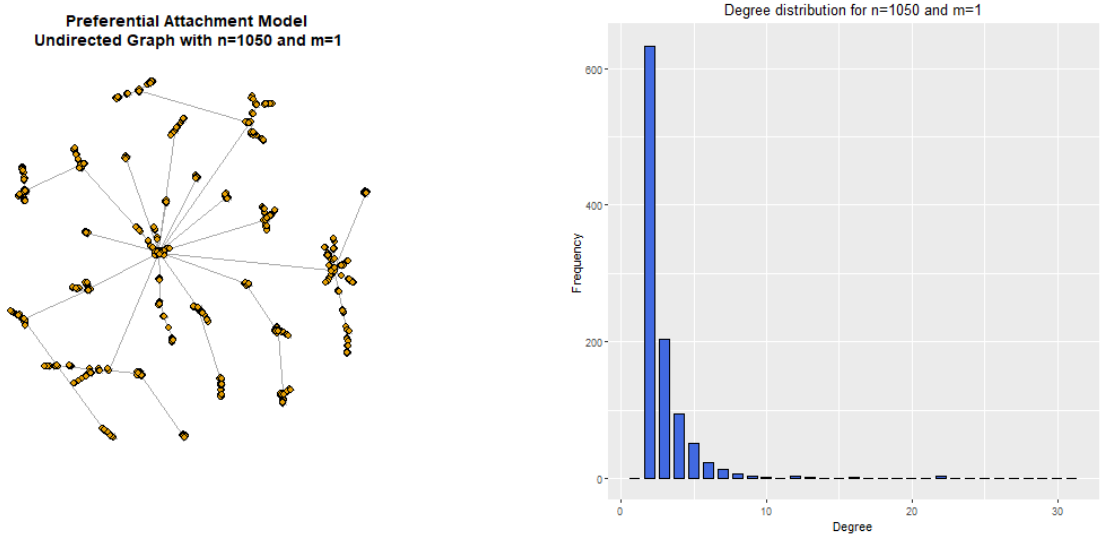
Part (a). Create an undirected network with n = 1050 nodes, with preferential attachment model, where each new node attaches to m = 1 old nodes. Is such a network always connected?

The preferential attachment model differs from the Erdos-Renyi model in that it does not give every possible node pair an equal probability of establishing an edge. Instead, the preferential attachment model introduces nodes into the network one-by-one with a probability of creating an edge with an existing node proportional to its current degree according to Equation 4, where k_i is the degree of node i. The Barabasi-Albert model creates edges to existing nodes with a probability according to Equation 5, where $\sum_j k_i$ is the sum of all existing nodes j.

$$P[i] \sim k_i^{\alpha} + \alpha \tag{4}$$

$$P_i = \frac{k_i}{\sum_j k_i} \tag{5}$$

The Barabasi-Albert model is a variant of the preferential attachment method, which is used for generating the graphs for this question. The model also allows for every new node to bring into the network m-number of edges. The *barabasi.game()* function is used to specify how many nodes the graph should consist of and how many edges each node will bring with it. Figure 10 shows an example of a graph with n = 1050 and m = 1 generated using the preferential attachment model. Its degree frequency distribution is shown in (b) demonstrating how creating new edges to existing nodes proportional to their current degree reflects the "rich get richer" effect observed in systems that follow the power law.

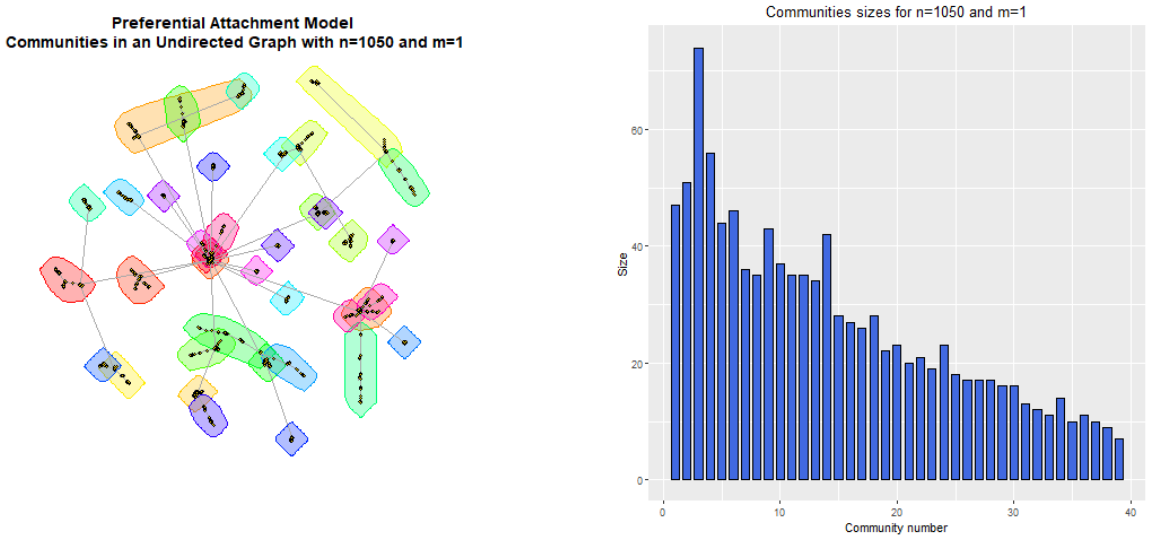


(a) (b)

Figure 10: Undirected random network generated using the Preferential Attachment Model with n = 1050 and m = 1. (a) Network visualization. (b) Probability density distribution.

Part (b). Use fast greedy method to find the community structure. Measure modularity. Define Assortativity. Compute Assortativity.

The *cluster_fast_greedy()* function implements the fast greedy modularity optimization algorithm for finding community structure. It takes in a given graph and returns groupings of nodes that exhibit connections dense enough to comprise a subgraph, also called communities. An example of this is shown in Figure 11a, where the color-coded groupings indicate identified communities. The community sizes distribution is shown in Figure 11b.



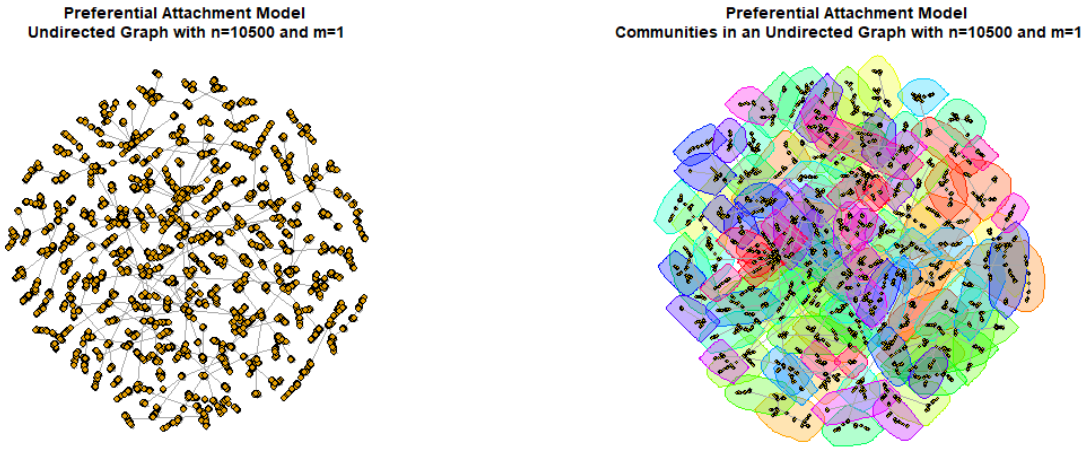
(a) (b)

Figure 11: Undirected random network generated using the Preferential Attachment Model with n = 1050 and m = 1. (a) Network visualization. (b) Probability density distribution.

Modularity is a measure of how modular the division of a graph is into given subgraphs. The modularity score ranges from -1 to 1, where the higher modularity scores indicate stronger division into communities, while lower scores indicate a more random or homogeneous structure. The *modularity()* function returns a modularity score and can be used to measure how effective the fast greedy algorithm was in determining the communities that it returned. For the graph and community structure in Figure 11, its modularity score is: 0.929. On a scale from -1 to 1, this suggests an effective identification of well-defined communities. It can be seen in the figure that the groups of nodes determined to be communities have dense internal connections and sparse connections to other communities.

Part (c). Try to generate a larger network with 10500 nodes using the same model. Compute modularity and assortativity. How is it compared to the smaller network's modularity?

Running the same model with n = 10500 produces the network and community structure shown in Figure 12, and the community size distribution can be seen in Figure 13. The modularity score is also computed for this grpah and compared to the graph in part (a) and (b) with n=1050. It can be seen that the modularity score increased for a graph with an order of magnitude more nodes.



(a) (b)

Figure 12: Undirected random network generated using the Preferential Attachment Model with n = 10500 and m = 1. (a) Network visualization. (b) Community structure.

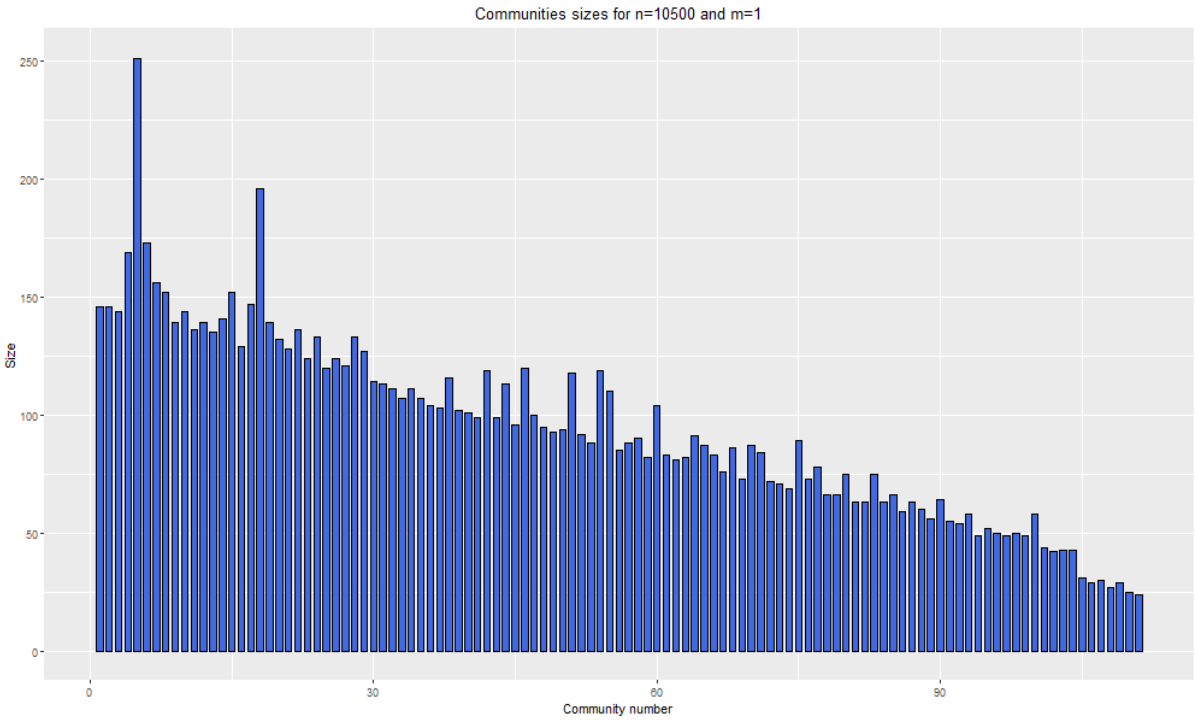


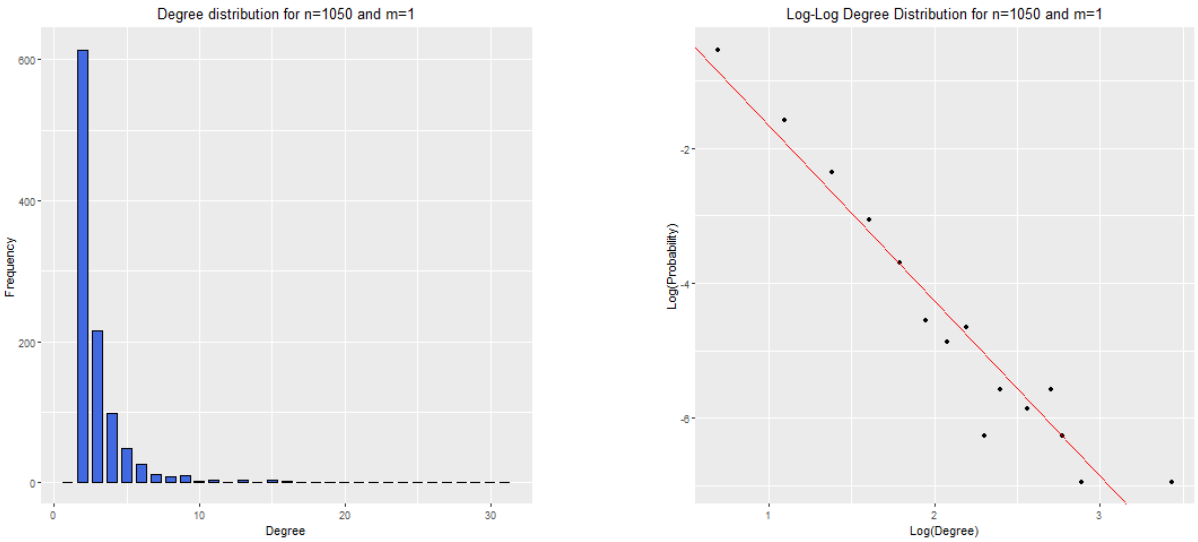
Figure 13: Community sizes

Table 3: Observed vs. theoretical mean and variance of the previously generated networks for n = 900.

n	Community Size	Modularity Score
1050	34	0.929
10500	111	0.979

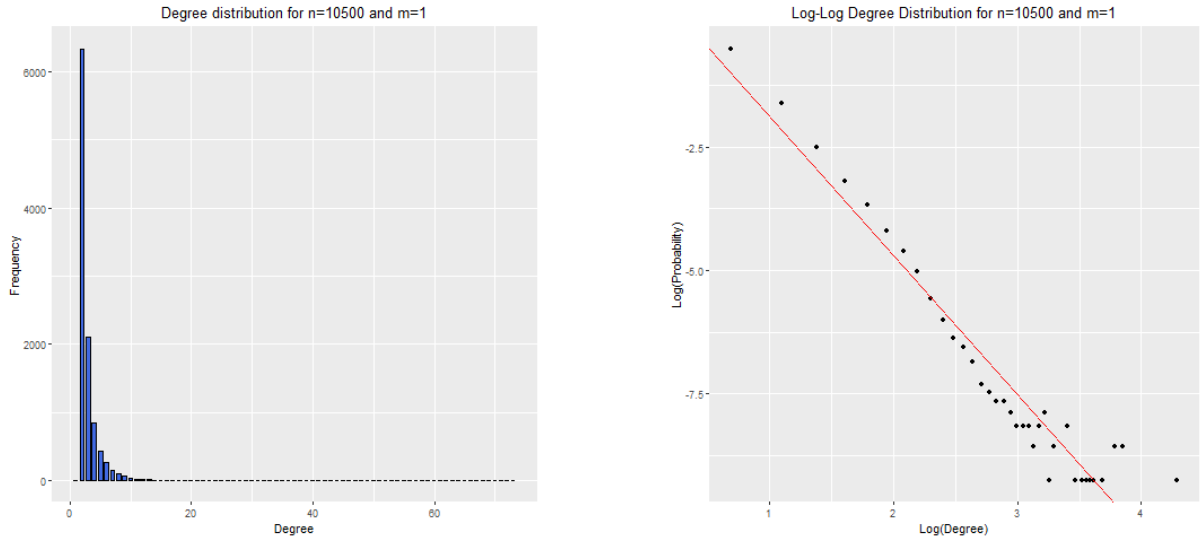
Part (d) Plot the degree distribution in a log-log scale for both n = 1050, 10500, then estimate the slope of the plot using linear regression.

To plot the degree distribution in a log-log scale, a graph was created and its degree distribution was found. Then both axes of the degree distribution were transformed into log-scale and then plotted. R's in-built linear regression function, $lm(x \sim y)$, was used to pull out the slope and intercept of the best fit line, shown in Table 5. The same steps were taken for both cases of n and their resulting plots shown in Figure 14 and Figure 15. It can be seen that both cases of n continue to show a power law relationship in their degree distribution, and is further supported by the close adherence to a linear relationship when transformed into the log-scale.



(a) (b)

Figure 14: Undirected random network generated using the Preferential Attachment Model with n = 1050 and m = 1. (a) Degree frequency distribution. (b) Log-Log plot of the same degree distribution with a linear regression best fit line.



(a) (b)

Figure 15: Undirected random network generated using the Preferential Attachment Model with n = 10500 and m = 1. (a) Degree frequency distribution. (b) Log-Log plot of the same degree distribution with a linear regression best fit line.

Table 4: The slope and intercept of the linear regression best fit for each case of n.

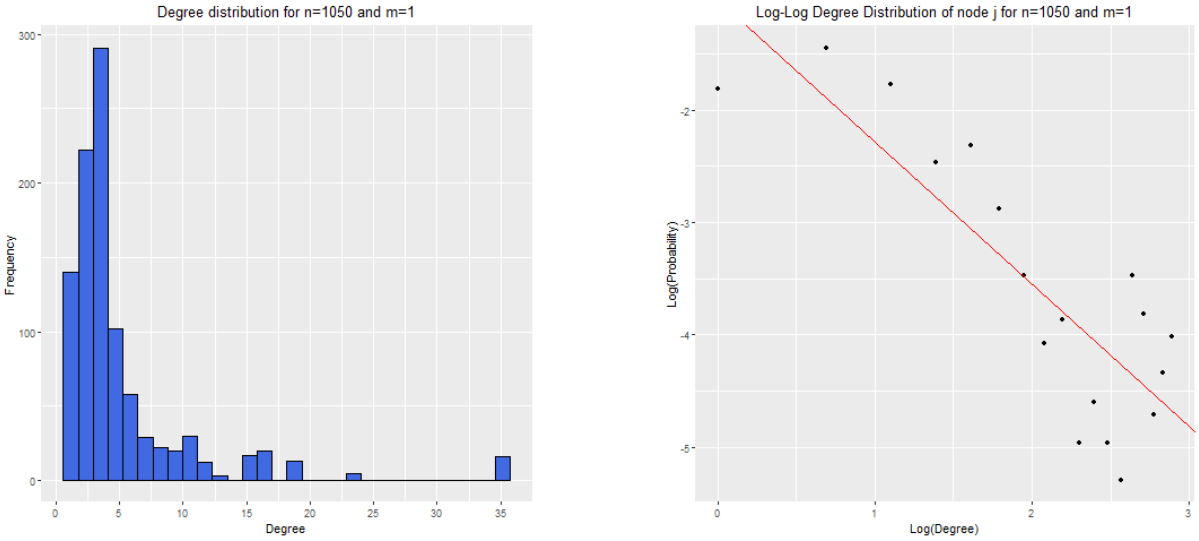
n	Slope	Intercept
1050	-2.601	0.941
10500	-2.818	0.942

Part (e) In the two networks generated in 2(a) and 2(c), 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. Is the distribution linear in the log-log scale? If so, what is the slope? How does this differ from the node degree distribution?.

The plots for this question were generated by taking the graphs generated in parts (a) and (c), and using R's sample() function to take random $node_i$ within the network. The igraph *neighbors()* function was used to find the indices of all neighbors connected to $node_i$, where a random neighbor $node_j$ was selected, and the degree of that node was found. 1000 iterations of this were ran for each graph to calculate the expected degree of a random neighbor $node_j$ connected to a randomly selected $node_i$. The resulting graphs are shown in Figures 16 and 17.

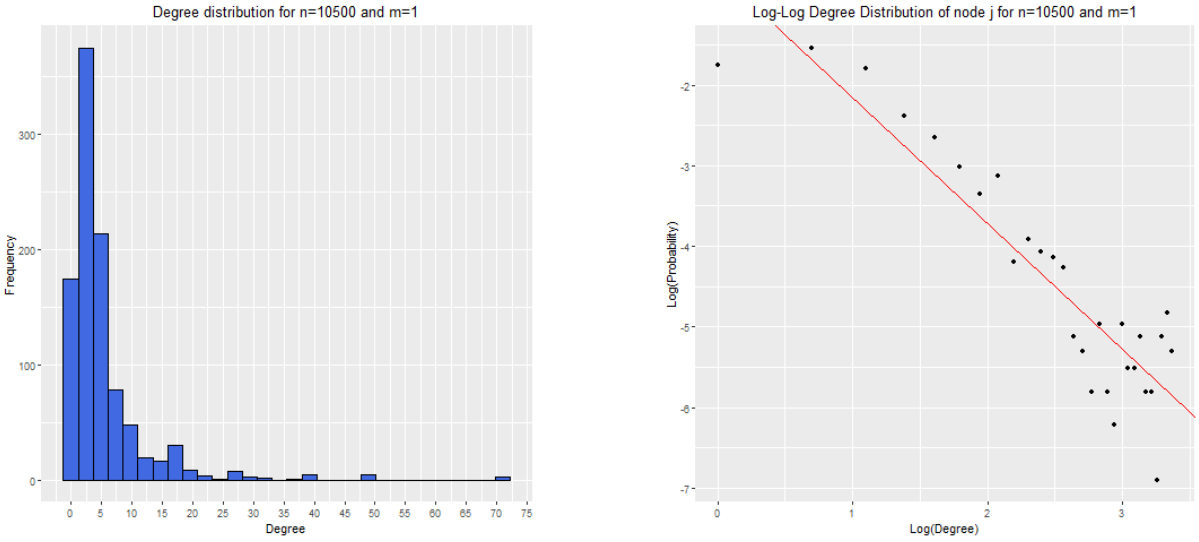
It can be seen that this random sampling of node degree captures a degree distribution that is still linear in the log-scale. This method still captures a noticeably higher representation of some higher degree nodes, seen at the far tail end of the histogram. This could be due to the single-step random walk having a high probability of selecting a node that walks into the node with the highest degree of connections to neighboring nodes. This same reasoning could explain the lower representation of lower degree nodes which have a lower probability of being randomly walked into because of their lower degree

of connections to neighboring nodes. The slopes of the linear regression can also be seen in Table 5, which show a greater slope value, indicating that while the degree distribution still exhibits a power law, it is not fully capturing the distribution of degrees throughout the network. The random-walk step is likely introducing some bias into the collection of this data.



(a) (b)

Figure 16: Undirected random network generated using the Preferential Attachment Model with n = 10500 and m = 1. (a) Degree frequency distribution. (b) Log-Log plot of the same degree distribution with a linear regression best fit line.



(a) (b)

Figure 17: Undirected random network generated using the Preferential Attachment Model with n = 10500 and m = 1. (a) Degree frequency distribution. (b) Log-Log plot of the same degree distribution with a linear regression best fit line.

Table 5: The slope and intercept of the linear regression best fit for each case of n.

n	Slope	Intercept
1050	-1.314	-0.912
10500	-1.660	-0.430

Part (f) Estimate the expected degree of a node that is added at time step i for 1 i 1050. Show the relationship between the age of nodes and their expected degree through an appropriate plot. Note that the newest added node is the youngest.

To estimate the degree of a node that is added at every time step, 1000 iterations were ran to collect the degree distributions of every node as it was added to the network, and the their sum total at every time step was divided by the number of iterations to produce the expected degree of the i^{th} node from i = 1 to 1050. Figure 18 shows the results of this approach, where the average or expected degree is on the y-axis, while the x-axis shows the age of the nodes, meaning that moving to the right is moving through nodes that were added earlier in the generation of the networks. The plot shows that the earlier that the node was added, the more connections it was able to accrue over time, while the nodes added later had less time to accrue as many nodes. This behavior is expected from the power law formula that determines the increasing probability that a connection is made to a node with increasing connections. The rich node gets richer.

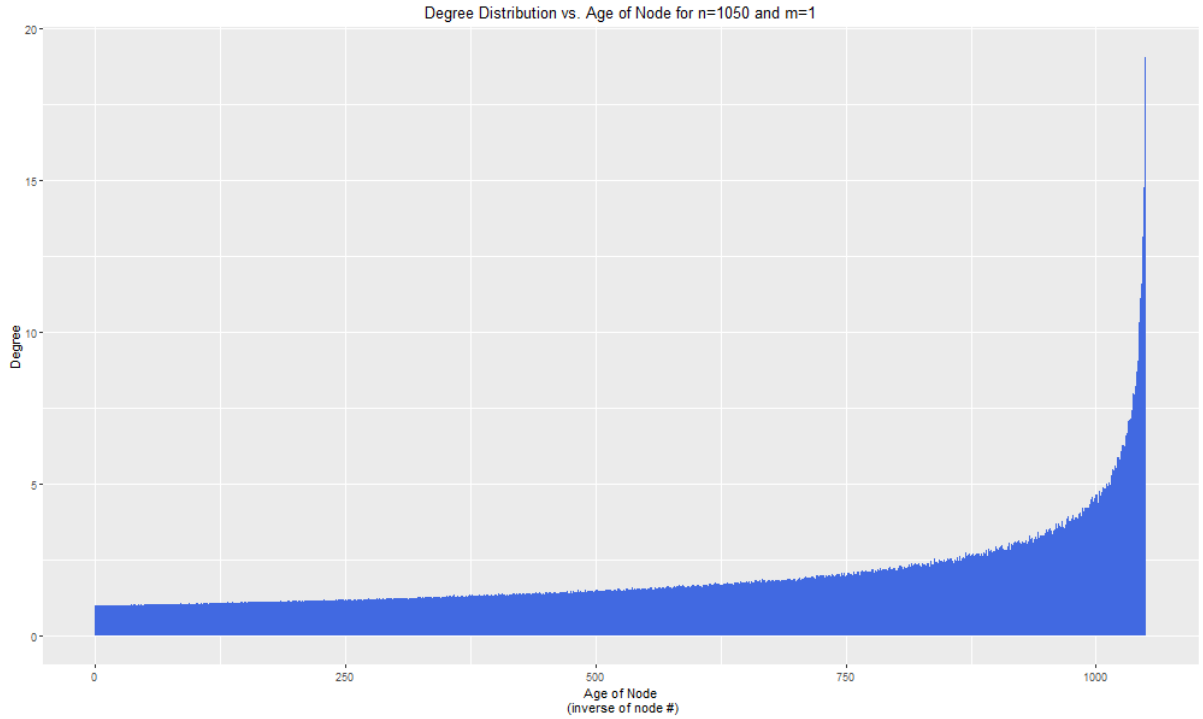


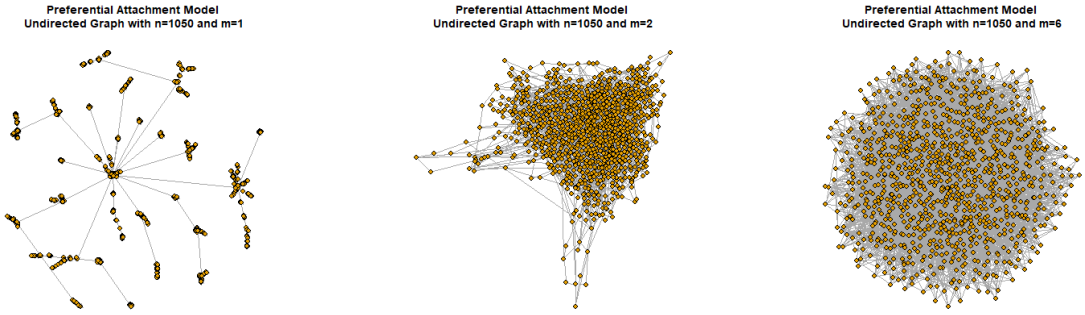
Figure 18: Expected degree vs. Age of the node

Part (g) Repeat the previous parts (a-f) for m = 2, and m = 6. Compare the results of each part for different values of m.

In this section, every plot that was previously generated for m = 1 was created again for m = 2 and m = 6, with all three plots juxtaposed next to each other to observe general differences in trends, if any.

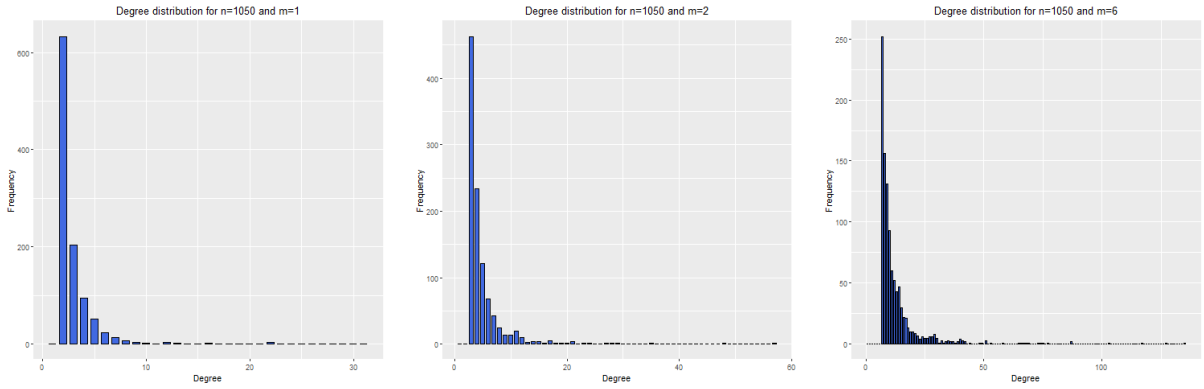
Part (g)(a)

It can be seen in Figure 19 how the networks of the same size increase in their density the more edges that they bring with them to the network. This is no surprise and makes sense that the general power law shape of their degree distributions would also not change.



(a) m = 1 (b) m = 2 (c) m = 6

Figure 19: Network Visualization network with n = 1050.

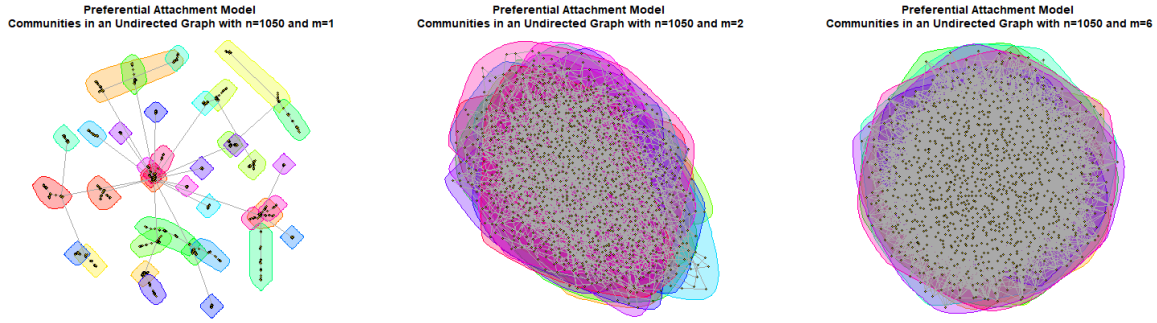


(a) m = 1 (b) m = 2 (c) m = 6

Figure 20: Degree distribution of network with n = 1050.

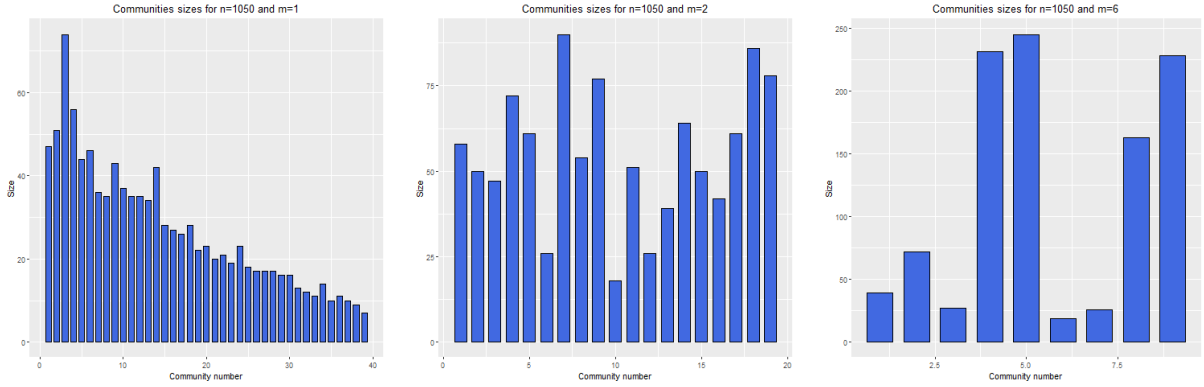
Part (g)(b)

The overlapping of communities becomes more apparent as the network becomes more dense with higher m values, as can be seen in Figure 21. This is corroborated by Table 6 which shows that the modularity diminishes significantly in denser networks as it becomes more difficult to discern any boundaries between possible communities, if they even exist. As the nodes within a network become more connected amongst each other and the diameter of the network decreases, then the entire network becomes a large community.



(a) m = 1 (b) m = 2 (c) m = 6

Figure 21: Community structure of network with n = 1050.



(a) m = 1 (b) m = 2 (c) m = 6

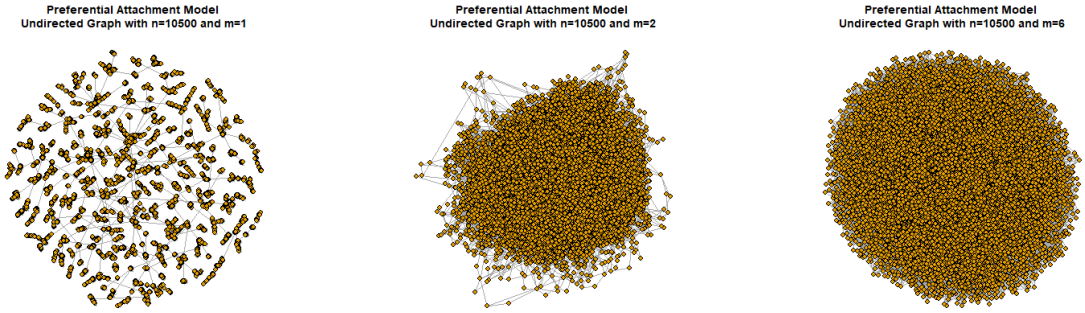
Figure 22: Community sizes of network with n = 1050.

Table 6: The slope and intercept of the linear regression best fit for each case of n.

n	**m**	**Modularity**
1050	1	0.929
1050	2	0.523
1050	6	0.247

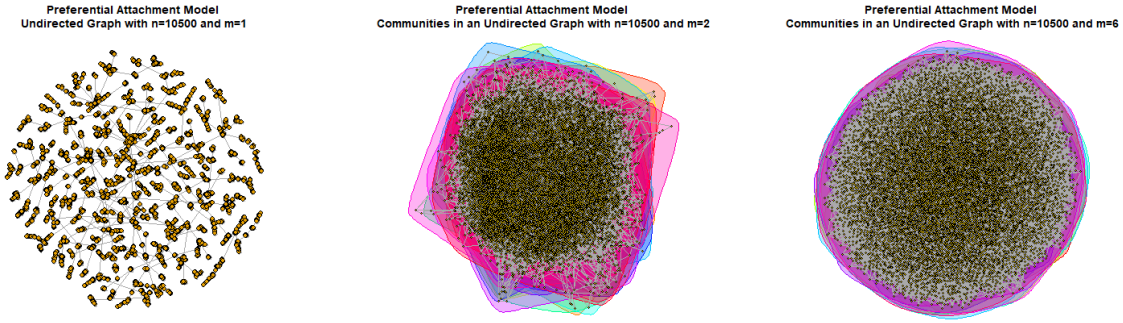
Part (g)(c)

When generating larger networks with n = 10500, the results are similar regarding modularity and community structure. The modularity scores are on par with a network of n = 1050 where modularity decreases with increasing m. And the community structure is again more ambiguous and difficult to discern as the network becomes more dense.



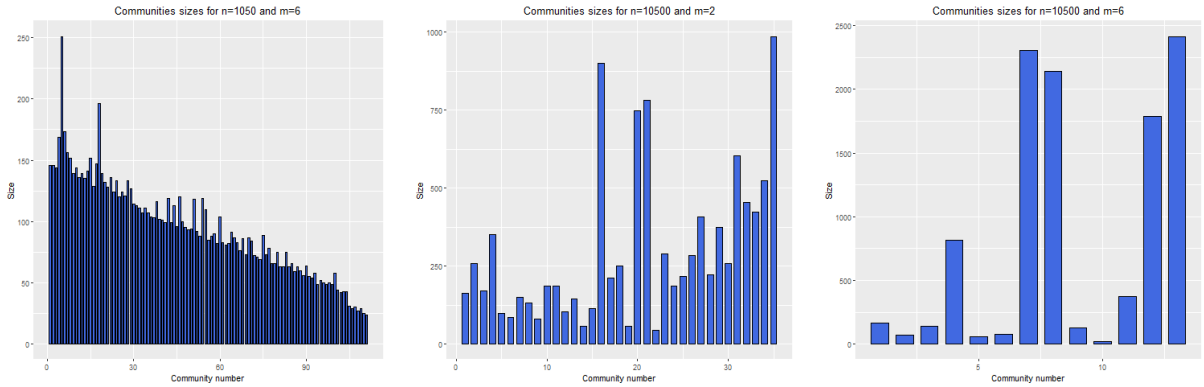
(a) m = 1 (b) m = 2 (c) m = 6

Figure 23: Network visualization of network with n = 10500.



(a) m = 1 (b) m = 2 (c) m = 6

Figure 24: Community structure of network with n = 10500.



(a) m = 1 (b) m = 2 (c) m = 6

Figure 25: Community sizes of network with n = 10500.

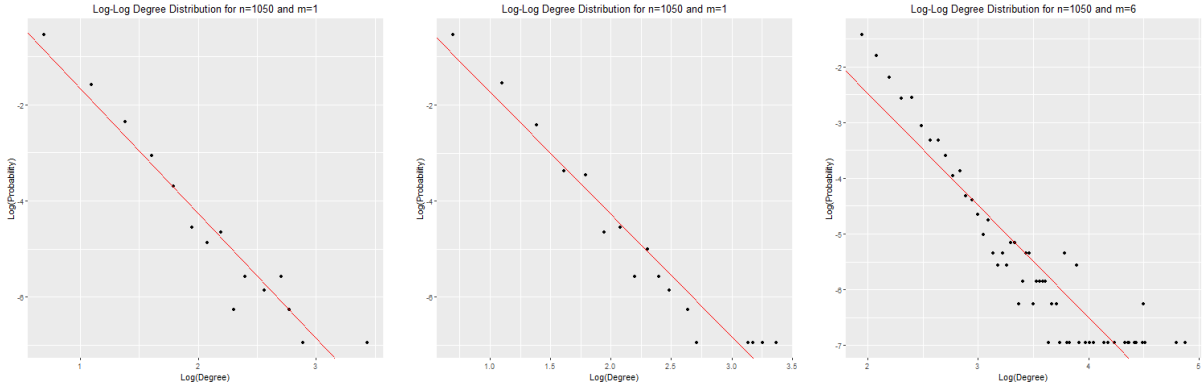
Part (g)(d)

The log-log plots of the degree distributions for m = 2 and 6 follow the same general trend as m = 1. The log scale degree distributions all exhibit linear behavior suggesting that their degree distributions follow

Table 7: The slope and intercept of the linear regression best fit for each case of n.

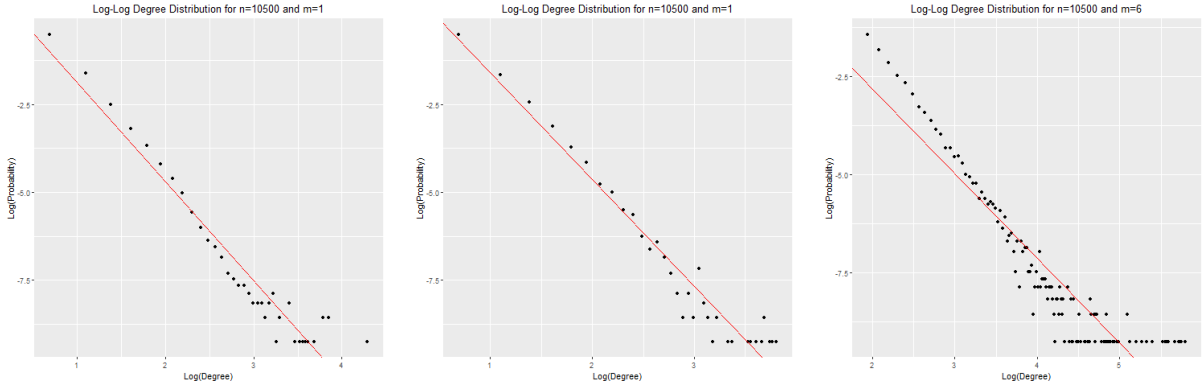
n	m	Modularity
1050	1	0.972
1050	2	0.534
1050	6	0.249

the power law, while the slopes are shown in Table 8 showing that the slopes of the linear regression do become steeper indicating that the exponentiation of the degree distribution is greater when each node brings more edges.



(a) m = 1 (b) m = 2 (c) m = 6

Figure 26: Log scale of density distribution of network with n = 1050.



(a) m = 1 (b) m = 2 (c) m = 6

Figure 27: Log scale of density distribution of network with n = 10500.

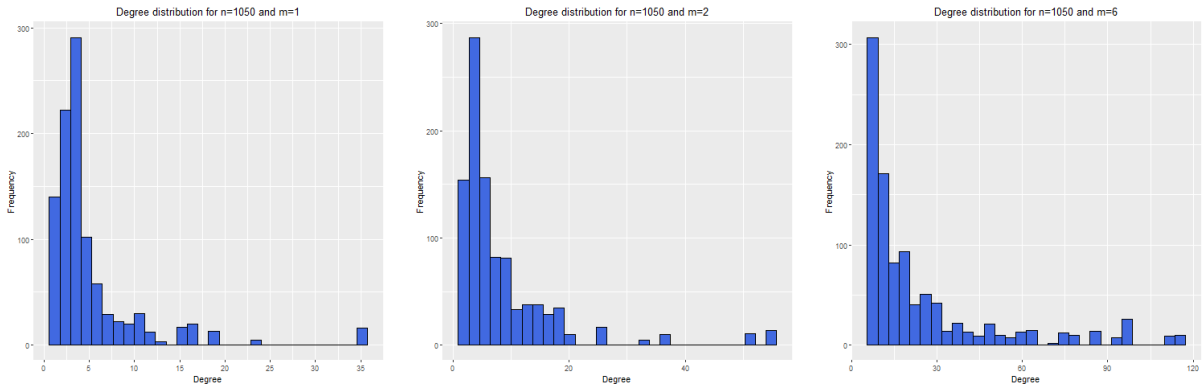
Part(g)(e)

It can be seen from Figure 28 and Figure 30 that the general trends are the same for sparser and denser networks. Selecting a random node in the network and then find the density distribution of a random neighbor of that node yields generally the same results that reflect higher probabilities of having relatively low degree nodes, which would serve as a rough sampling of the degree distribution of the overall network. The same observations can be made as were made in Part (e), where there seems to appear a sampling

Table 8: The slope and intercept of the linear regression best fit for each case of n.

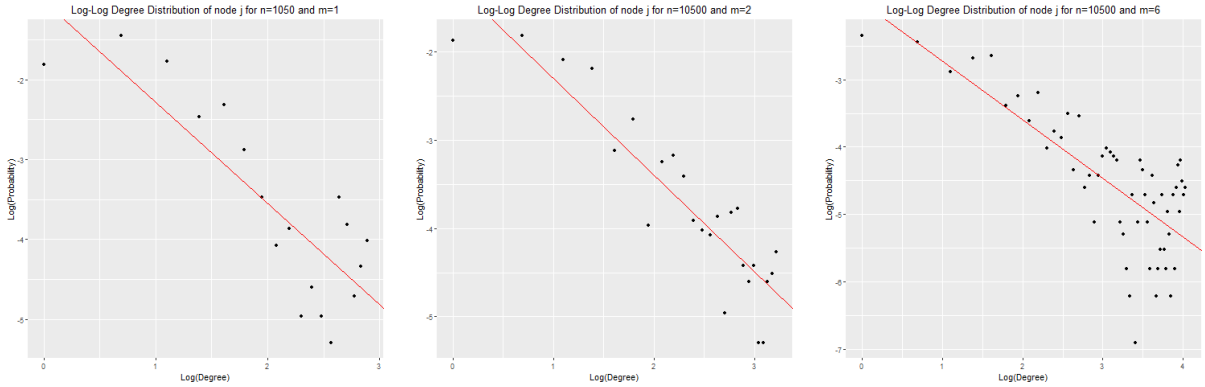
n	m	Slope	Intercept
1050	1	-1.314	-0.912
1050	2	-2.549	0.9821
1050	6	-2.011	1.545
10500	1	-1.660	-0.430
10500	2	-3.036	1.443
10500	6	-2.155	1.490

bias where performing a random one-step walk gives higher degree nodes higher probabilities of being walked into. This might explain the smattering of relatively higher representation at the tail ends of the degree distribution.



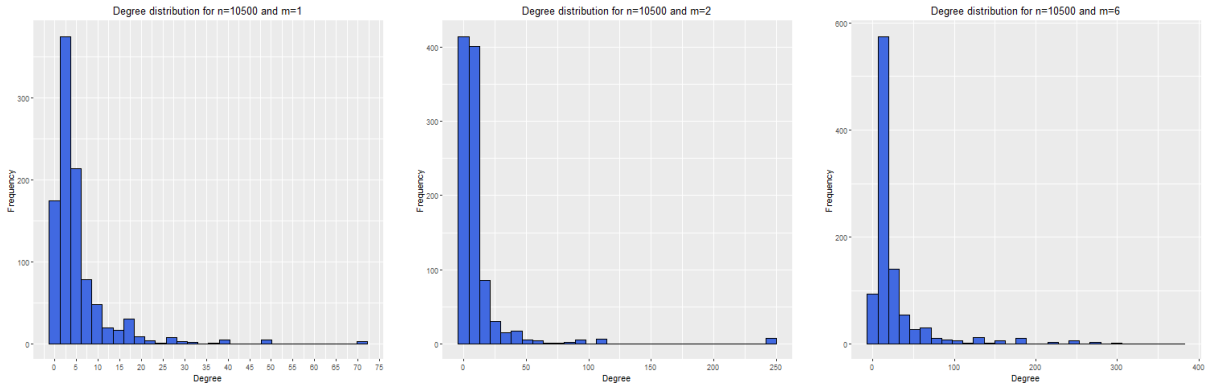
(a) m = 1 (b) m = 2 (c) m = 6

Figure 28: Expected degree distribution of $node_j$ in network with n = 1050.



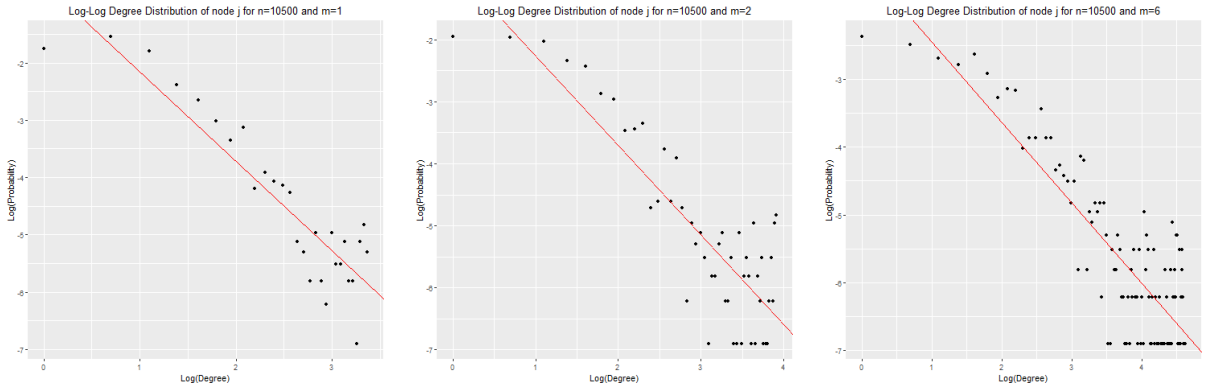
(a) m = 1 (b) m = 2 (c) m = 6

Figure 29: Log scale of expected degree distribution of $node_j$ in network with n = 1050.



(a) m = 1 (b) m = 2 (c) m = 6

Figure 30: Expected degree distribution of $node_j$ in network with n = 10500.

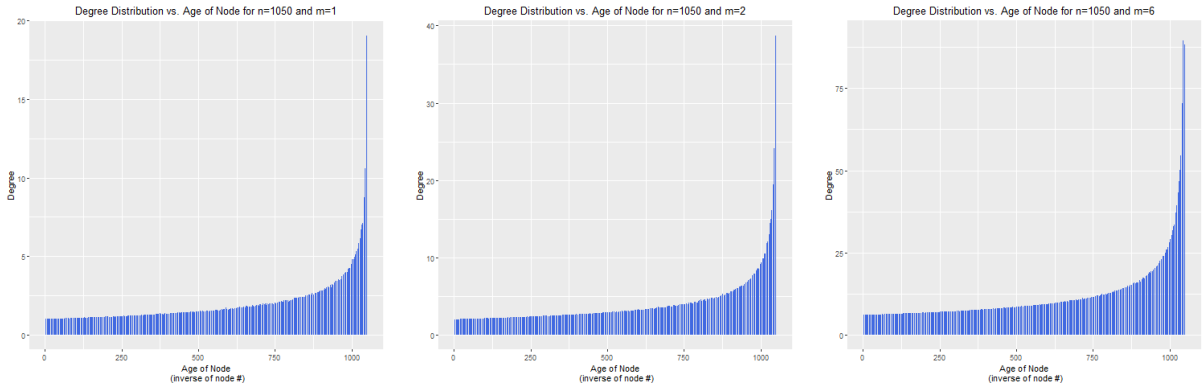


(a) m = 1 (b) m = 2 (c) m = 6

Figure 31: Log scale of expected degree distribution of $node_j$ in network with n = 10500.

Part (g)(f)

There is strong agreement amongst all three plots in Figure 32 that the age of a node correlates with having a higher expected degree than every node in the graph, where the expected degree distribution exponentially increases with increasing age. The more edges that each node introduces unsurprisingly adds more nodes to the degree distribution, which can be seen in the plots.

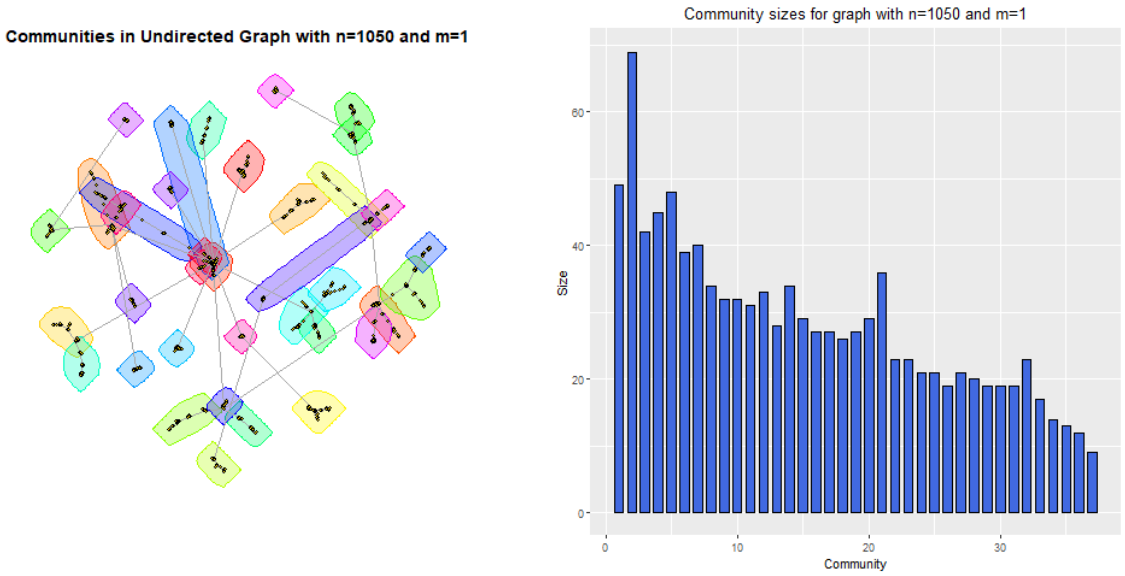


(a) m = 1 (b) m = 2 (c) m = 6

Figure 32: Degree distribution vs. Age of node for network with n = 1050.

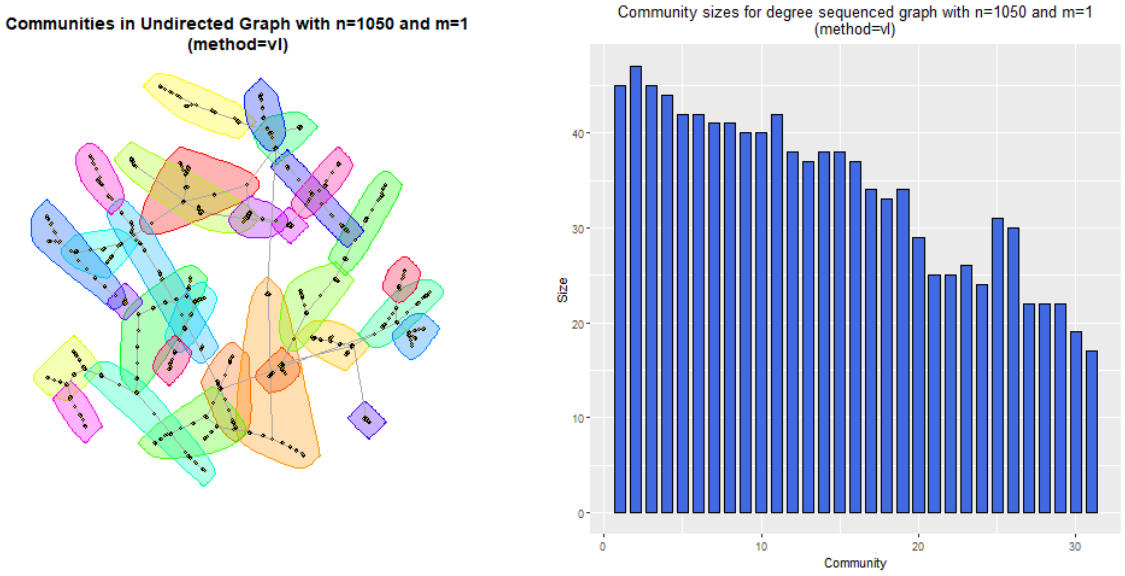
Part (h) Again, generate a preferential attachment network with n = 1050, m = 1. Take its degree sequence and create a new network with the same degree sequence, through stub-matching procedure. Plot both networks, mark communities on their plots, and measure their modularity. Compare the two procedures for creating random power-law networks.

To generate the graphs for this question, an original graph of n = 1050 and m = 1 is generated, shown in Figure 33. This graph's degree sequence is then fed into the *sample_degseq()* function to generate another graph through a stub-matching method. A stub-matching method is an approach of creating a graph by starting with a given set of vertices and disconnected edges (or stubs), and then matching the stubs to create a randomized network. A couple different methods of stub matching are shown below. In Figure 34, the vl or Viger-Latapy method is used which appears to generate an apparently organic graph generation that exhibits highly modular communities. Another method shown in Figure 35 shows a resulting interwoven web of communities alongside a large portion of nodes that were not given an edge. This is reflected in its degree distribution where a fraction of nodes have interconnections, while the remainder have no connections.



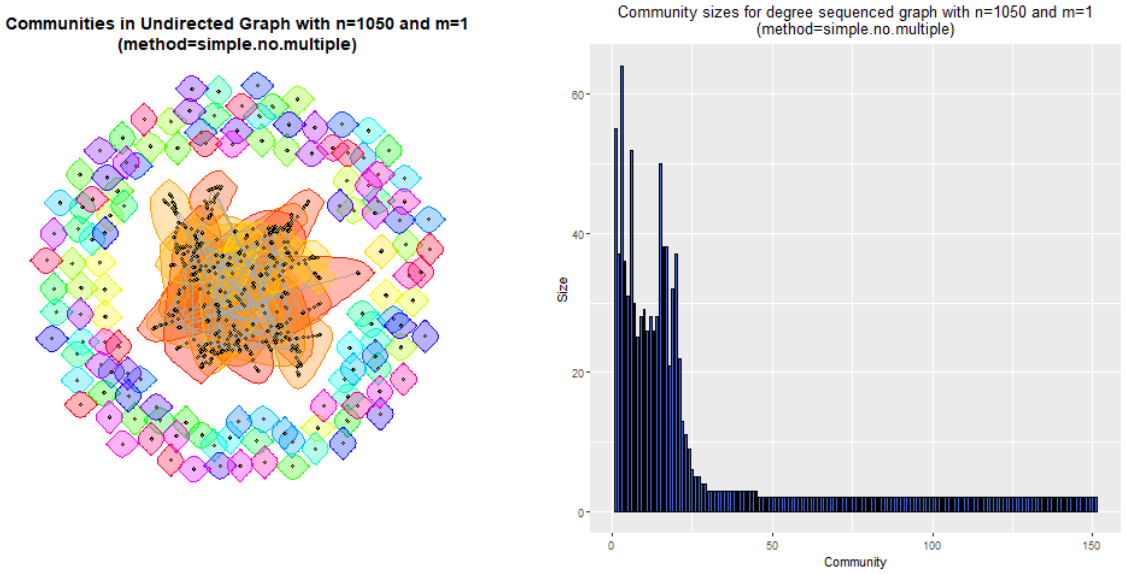
(a) (b)

Figure 33: Preferential attachment model network with n = 1050. (a) Community structure. (b) Community sizes.



(a) (b)

Figure 34: Stub-matched network with n = 1050 using the ”vl” method. (a) Community structure. (b) Community sizes.



(a) (b)

Figure 35: Stub-matched network with n = 1050 using the "simple no multiple" method. (a) Community structure. (b) Community sizes.

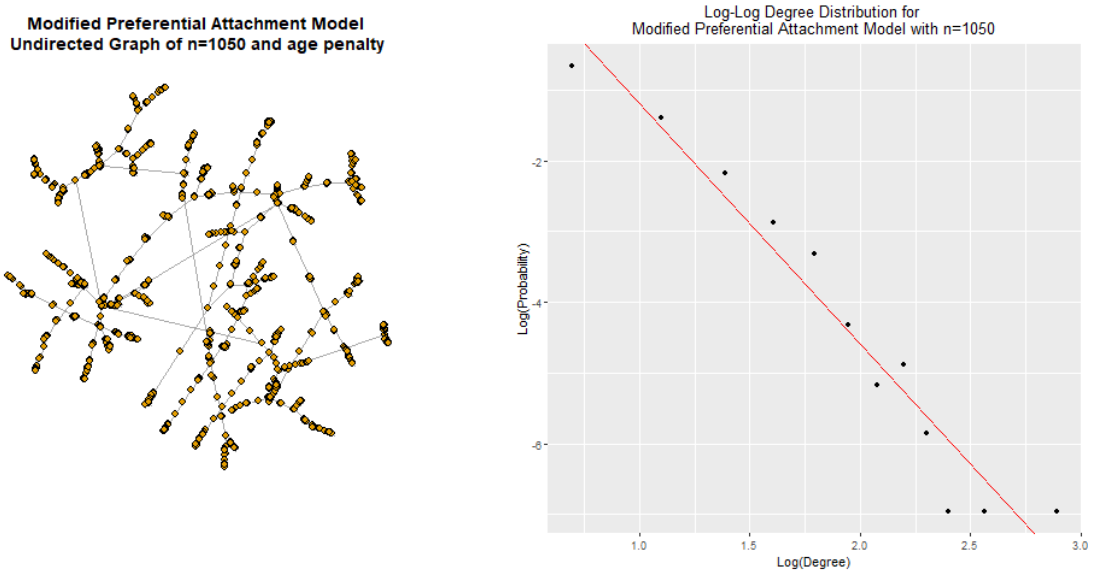
Question 3. Create a modified preferential attachment model that penalizes the age of a node

Part (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:

$$P[i] \sim (c \cdot k_i^{\alpha} + a)(d \cdot l_i^{\beta} + b), \tag{6}$$

where k_i is the degree of vertex i in the current time step, and l_i is the age of vertex i. Produce such an undirected network with 1050 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 *sample_pa_age()* function is used to generate the modified preferential attachment graph using the given parameters and formula for probability of edge-formation in Equation 6. Figure 36 shows the resulting plot and the log scale degree distribution with the slope of the linear regression being -3.319, equal to the power law exponent.



(a) (b)

Figure 36: Modified preferential attachment model (a) Network visualization. (b) Log-scale degree distribution.

.

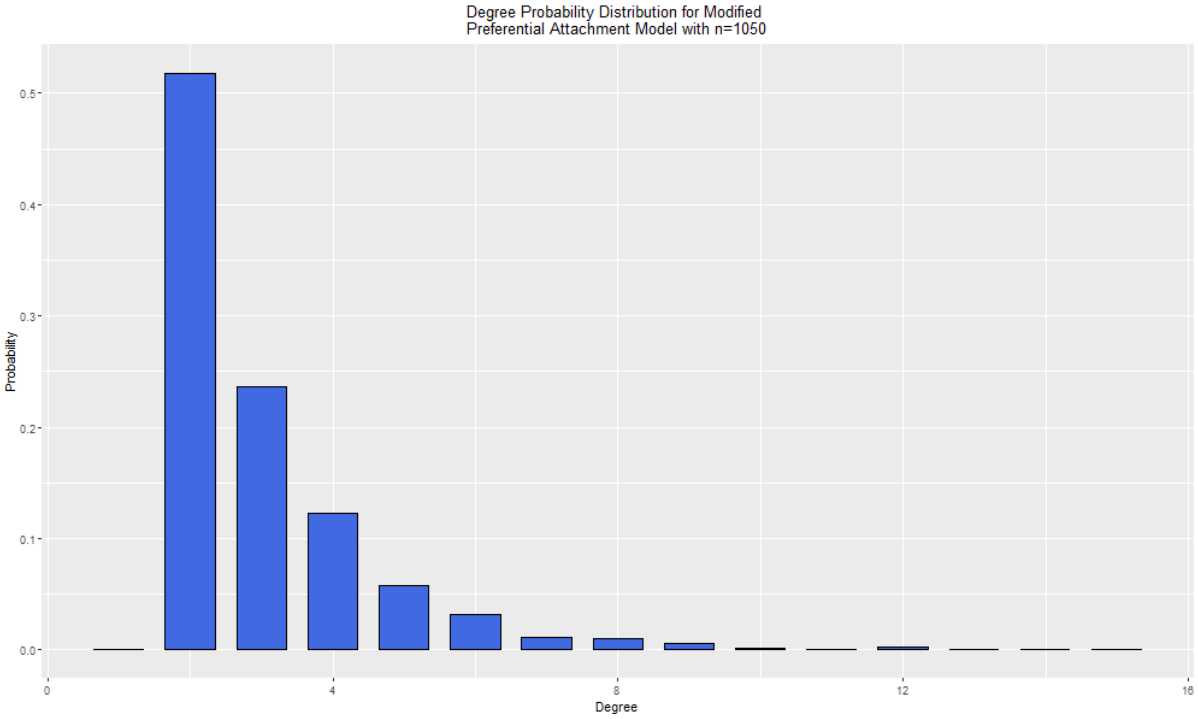


Figure 37: Modified preferential attachment degree distribution.

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

The community structure the graph in part (a) is show in Figure 38. The modularity of these communities is 0.938, indicating a strong discernment between communities. This score is on par with the modularity of 0.929 from Question 2 part (b) with was a similar preferential attachment network of n = 1050, but without age penalization. The modularity of age penalization is slightly higher, which could be explained by the inhibition of edge-formation to older nodes. This inhibition might be loosening the bias towards the largest cluster and allowing for more nodes to find connections to other clusters.

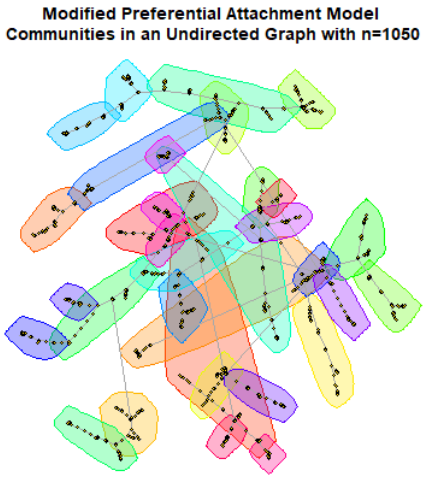


Figure 38: Modified preferential attachment community structure.

Part 2. Random Walk on Networks

Question 1: Random walk on Erd¨os-R´enyi networks

Part (a) Create an undirected random network with 900 nodes, and the probability p for drawing an edge between any pair of nodes equal to 0.015.

The ER graph with 900 nodes with probability 0.015 is shown in Fig. 39.

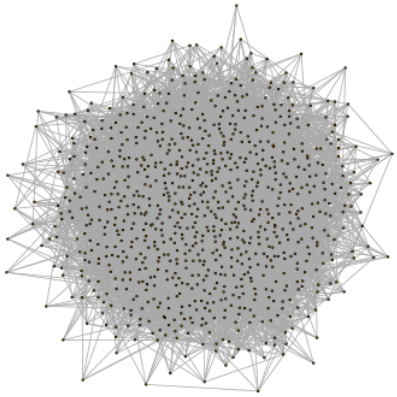
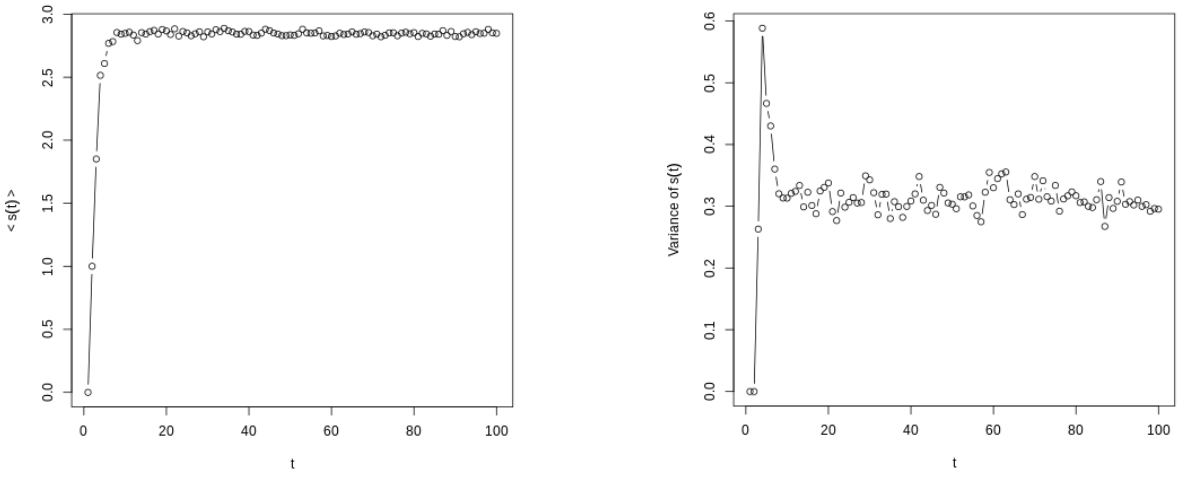


Figure 39: Generated ER network

Part (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) $\langle s(t) \rangle$ of the walker from his starting point at step t. Also, measure the variance $\sigma^2(t) = \langle (s(t) - \langle s(t) \rangle) 2 \rangle$ of this distance. Plot $\langle s(t) \rangle$ v.s. t and $\sigma^2(t)$ v.s. t. Here, the average $< . >$ is over random choices of the starting nodes

We perform a random walk that starts at a randomly selected node in GCC and run it for 100 steps. For each step $t \leq T$, we measure the shortest distance $s(t)$ from the starting node. In order to get $< s(t) >$ and $\sigma^2(t)$, we calculate the average of the variance of $s(t)$ over 1000 such random walks.

Observe from Fig. 40, $\langle s(t) \rangle$ and $\sigma^2(t)$ that reach a steady state around $t_{steady-state} = 10$ where $\langle s(t) \rangle \approx 2.75$ and $\sigma^2(t) \approx 0.32$.

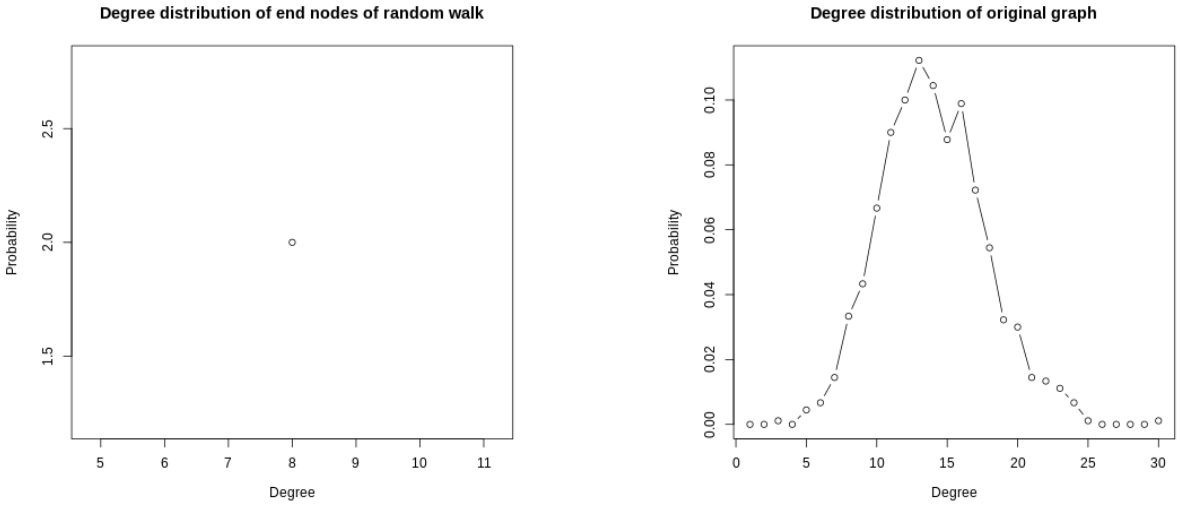


(a) $s(t)$ v.s. t (b) $\sigma^2(t)$ v.s. t

Figure 40: Variation of average and variance of the shortest distance to the walker from starting node with the number of steps (n = 900, p = 0.015)

Part (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?

From Fig. 41, the degree distribution from random walk and original network basically follow a similar binomial distribution, as expected for E-R networks.

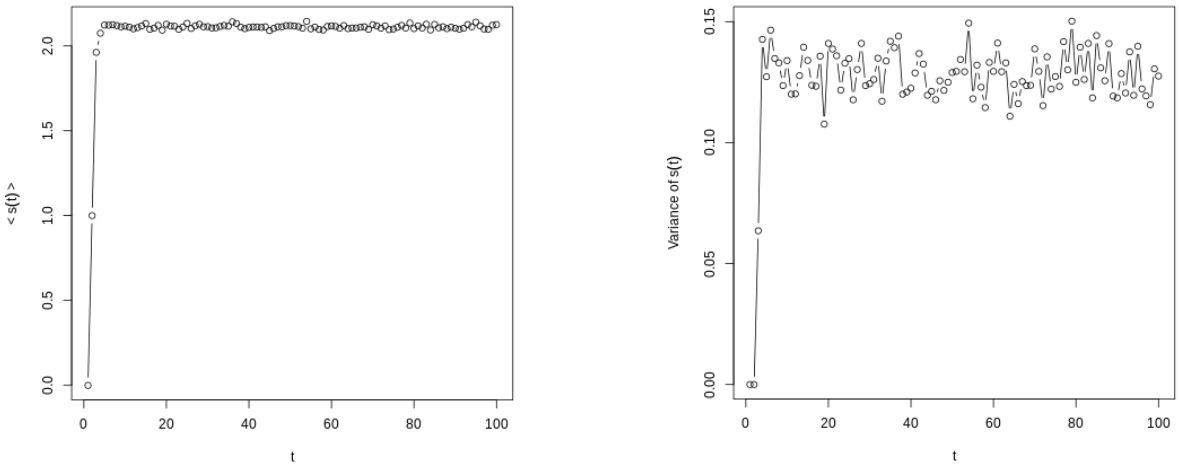


(a) Degree Distribution of end nodes in random walk (b) Degree Distribution of nodes in original network

Figure 41: Degree of distribution of the original network and the end nodes of the random walk (n = 900, p = 0.015)

Repeat 1(b) for undirected random networks with 9000 nodes. Compare the results and explain qualitatively. Does the diameter of the network play a role?

In this part, we run a similar experiment as Sec. , but for an E-R network with n = 9000 and p = 0.015. $\langle s(t) \rangle$ and $\sigma^2(t)$ for this graph is shown in Fig. 42. From Fig. 42, when steady state is reached, $\langle s(t) \rangle \approx 2.15$ and $\sigma^2(t) \approx 0.135$. For bigger network, the diameter is 3 which is smaller than the original network's diameter (4). Comparing Fig. 42 and Fig. 40, the bigger network reaches the ready state much faster than then the original network.



(a) $s(t)$ v.s. t (b) $\sigma^2(t)$ v.s. t

Figure 42: Variation of average and variance of the shortest distance to the walker from starting node with the number of steps ($n = 9000, p = 0.015$)

Question 2: Random walk on networks with fat-tailed degree distribution

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

An undirected preferential attachment network is created with n = 900 and m = 1, which is shown in Fig. 43.

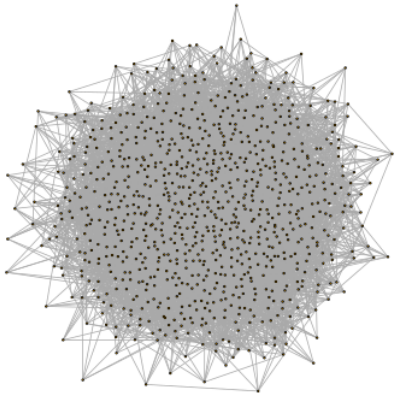
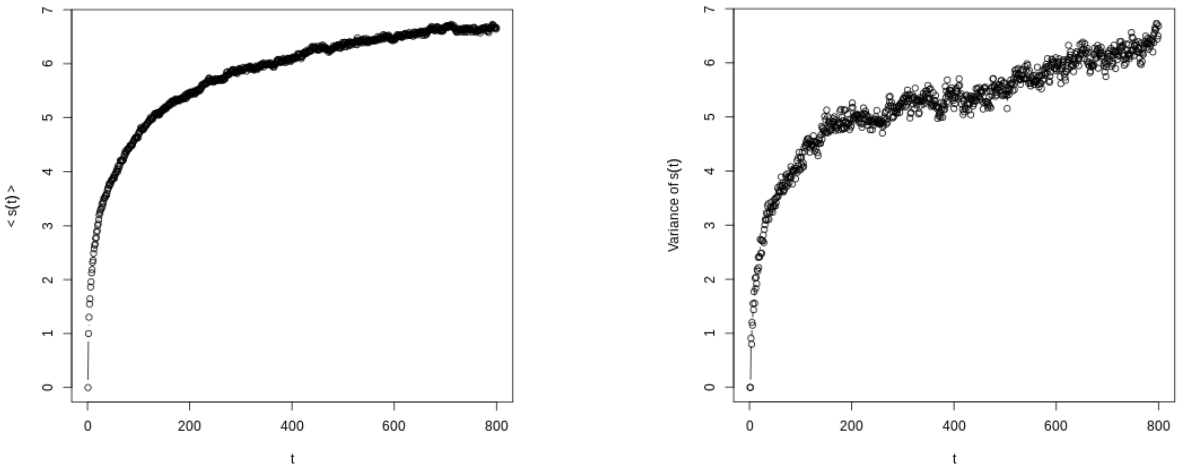


Figure 43: Generated ER network

Part (b) Let a random walker start from a randomly selected node. Measure and plot $\langle s(t) \rangle$ v.s. t and $\sigma^2(t)$ v.s. t.

We perform a random walk that starts at a randomly selected node in GCC and run it for 800 steps. For each step $t \leq T$, we measure the shortest distance $s(t)$ from the starting node. In order to get $< s(t) >$ and $\sigma^2(t)$, we calculate the average of the variance of $s(t)$ over 1000 such random walks.

Observe from Fig. 44, $\langle s(t) \rangle$ and $\sigma^2(t)$ that reach a steady state around $t_{steady-state} = 750$ where $\langle s(t) \rangle \approx 6.8$ and $\sigma^2(t) \approx 6$.



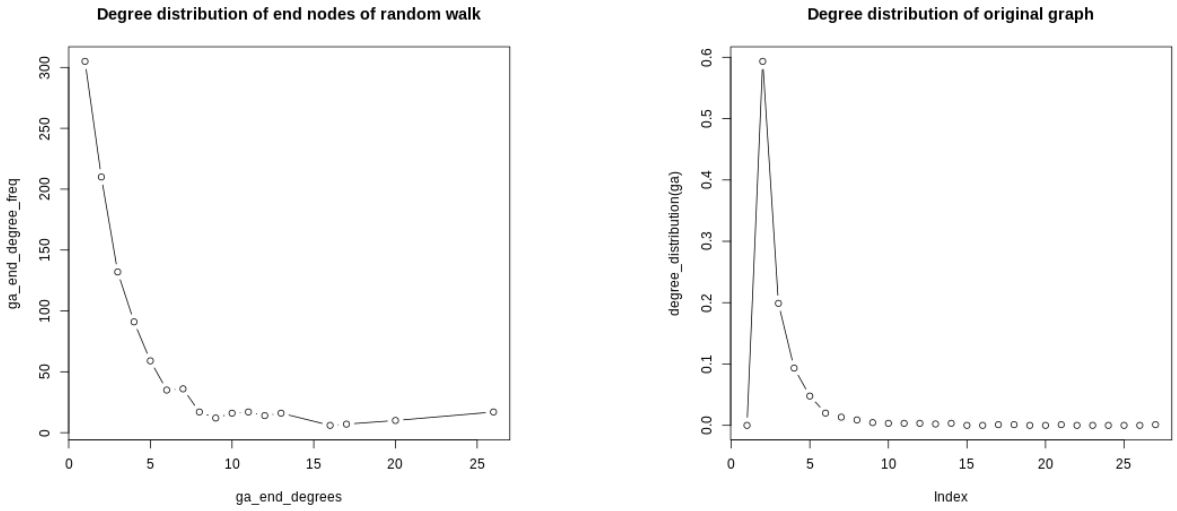
(a) $s(t)$ v.s. t

(b) $\sigma^2(t)$ v.s. t

Figure 44: Variation of average and variance of the shortest distance to the walker from starting node with the number of steps (n = 900, m = 1)

Part (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?

We calculate the degree of the node at the end node of the random walk as Sec. . The degree distribution of these end nodes and the degree distribution of the original professional attachment network are shown in Fig. 45. Both distribution roughly follow a similar power law distribution, as expected for professional attachment networks.

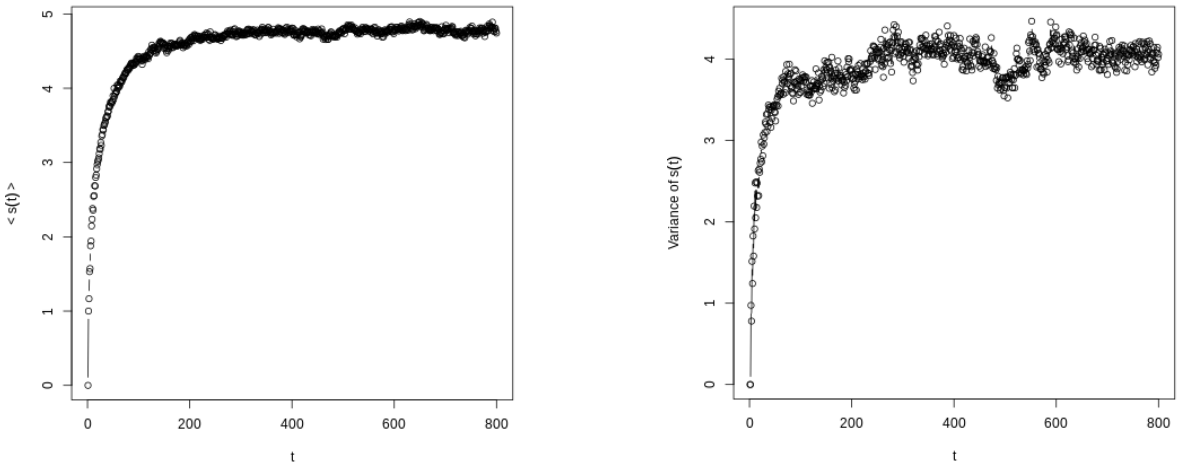


(a) Degree Distribution of end nodes in random walk (b) Degree Distribution of nodes in original network

Figure 45: Degree of distribution of the original network and the end nodes of the random walk (n = 900, m= 1)

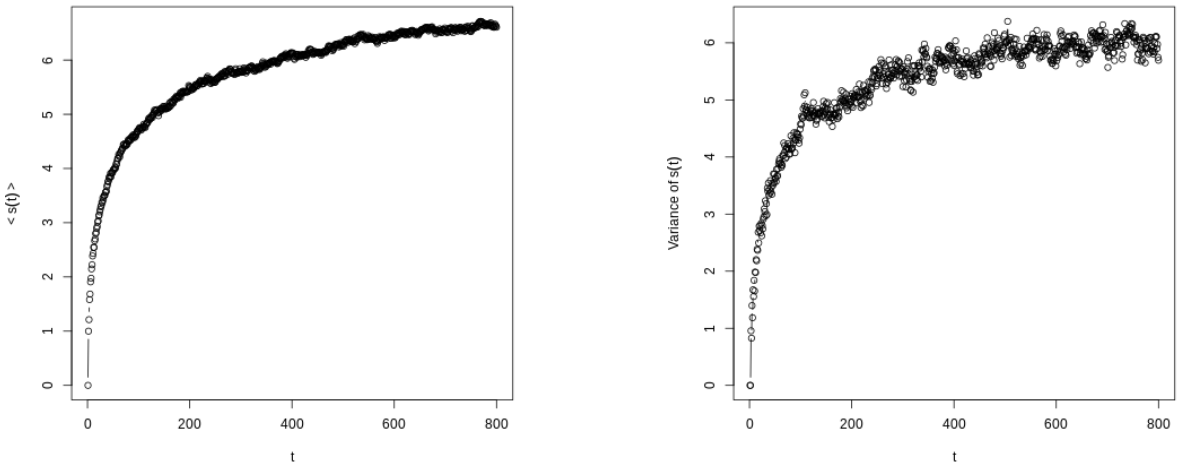
Part (d) Repeat 2(b) for preferential attachment networks with 100 and 10000 nodes, and m = 1. Compare the results and explain qualitatively. Does the diameter of the network play a role?

From Fig. 46 and Fig. 47, we can observe that for small network, $s(t) \approx 4.8$ and $\sigma^2 \approx 4.2$ when it reaches steady state; for bigger network, $s(t) \approx 6.2$ and $\sigma^2 \approx 5.2$ when it reaches steady state. Comparing Fig. 46 and Fig. 47, random walk in smaller PA network reaches steady state faster than the bigger network. The reason behind this might result from the bigger network (18) has larger diameter than the smaller network (12).



(a) $s(t)$ v.s. t (b) $\sigma^2(t)$ v.s. t

Figure 46: Variation of average and variance of the shortest distance to the walker from starting node with the number of steps (n = 90, m = 1)



(a) $s(t)$ v.s. t (b) $\sigma^2(t)$ v.s. t

Figure 47: Variation of average and variance of the shortest distance to the walker from starting node with the number of steps (n = 9000, m = 1)

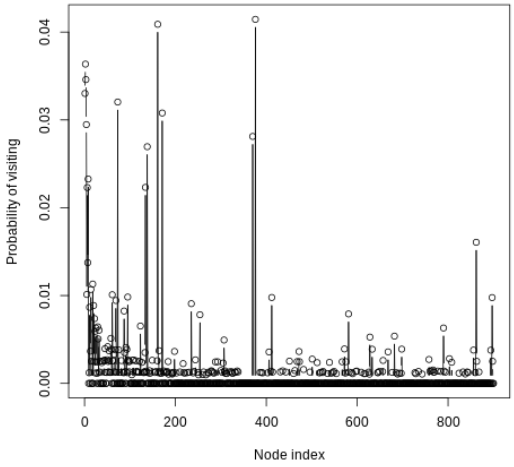
Question 3: The PageRank algorithm, as used by the Google search engine, exploits the linkage structure of the web to compute global "importance" scores that can be used to influence the ranking of search results. Here, we use random walk to simulate PageRank.

Part (a) We are going to create a directed random network with 900 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 900-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. 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?

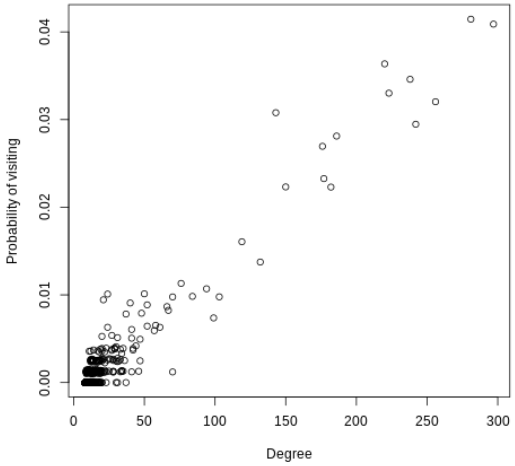
We create two professional attachment networks with $n = 900$ and $m = 4$ using `sample_ga` to create a new professional attachment network without "black-hole". We get the edge list of the second permuted network and then add the edges to the first network. The merged network has 7198 edges.

To get the node-visiting probability in the random walk, we perform 100 random walks with 1000 steps with randomly selected starting node. Moreover, we only start to count visits after each random walk reaches its steady-state. We achieve this by counting ceiling($ln(n)$) steps.

The probability that the random walker visits each node against node index and node degree is shown in Fig. 48. We can obverse that the visiting probability again node degree approximately lies in a line. After we fit them into a line, we get Pearson correlation coefficient of this relationship is 0.960171 and a linear fit gives a slope of 0.0001454878. The Pearson correlation coefficient indicates high-degree node has greater chance being visited.



(a) Probability of random walker visiting each node

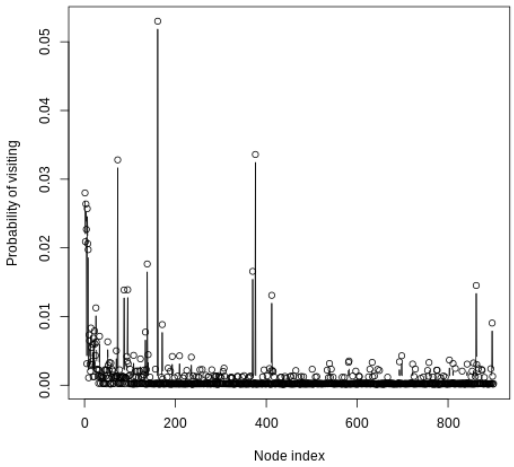


(b) Probability of random walker visiting nodes with given degree

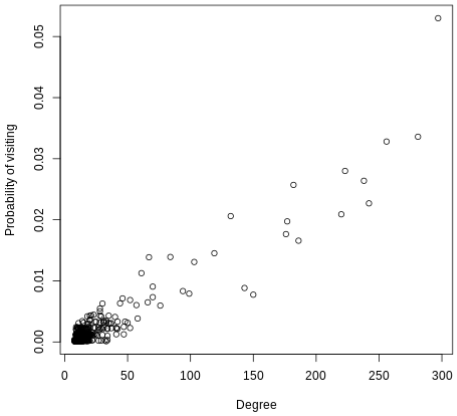
Figure 48: Probability of random walker visitation with the node index and node degree on Merge PA network (n = 900, m = 4)

Part (b) n all previous questions, we didn't have any teleportation. Now, we use a teleportation probability of α = 0.2(teleport out of a node with prob=0.2 instead of going to its neighbor). By performing random walks on the network created in 3(a), measure the probability that the walker visits each node. How is this probability related to the degree of the node and α ?

Similar to the previous section, the random walker has probability of $1 - \alpha$ to visit current node's neighbor. We obtained the probability that the random walker visits each node against node index and node degree in Fig. 49. We fit the probability again node degree in a line and get a slope of 0.000113981 and Pearson correlation coefficient 0.9615753.



(a) Probability of random walker visiting each node



(b) Probability of random walker visiting nodes with given degree

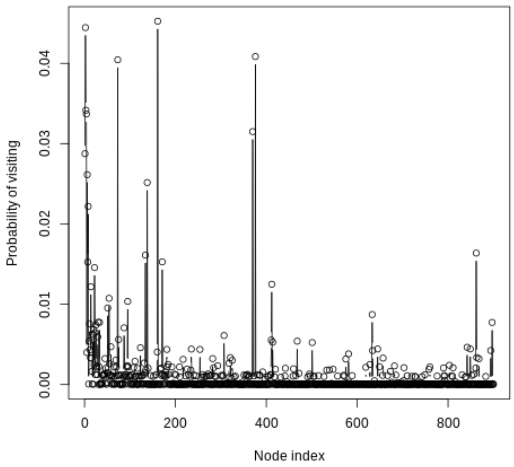
Figure 49: Probability of random walker visitation with the node index and node degree on Merged PA network with teleportation(n = 900, m = 4)

Question 4: 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.

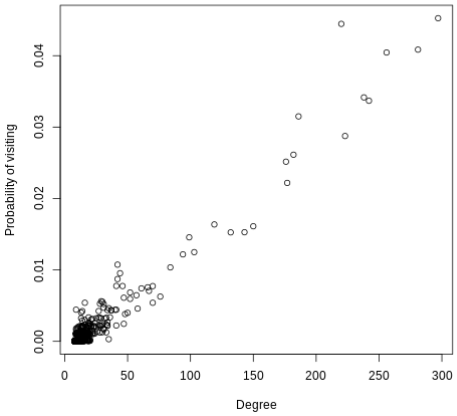
Part (a) Suppose you have your own notion of importance. Your interest in a node is proportional to the node's PageRank, because you totally rely upon Google to decide which website to visit (assume that these nodes represent websites). Again, use random walk on network generated in question 3 to simulate 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 $\frac{1}{N}$). Again, let the teleportation probability be equal to $\sigma = 0.2$. Compare the results with 3(a).

We run the experiment under similar settings as Sec. . We calculate the PageRanks using `page_rank` function to obtain the probability of any node being chosen as the teleportation destiny.

The probability that the random walker visits each node against node index and node degree in Fig. 50. The Pearson correlation coefficient is 0.9606795 and the slope of fitted line is 0.0001393134. Comparing Fig. 50 and Fig. 49, the slope increases in Fig. 50. The reason behind this is that random walker tend to choose the next nodes with higher page-ranks.



(a) Probability of random walker visiting each node



(b) Probability of random walker visiting nodes with given degree

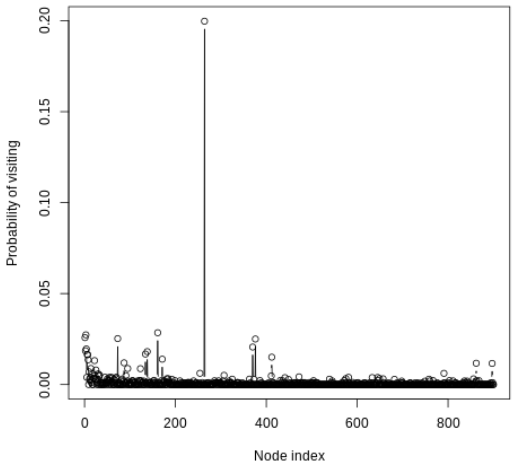
Figure 50: Probability of random walker visitation with the node index and node degree on Merged PA network with PageRank teleportation(n = 900, m = 4)

Part (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?

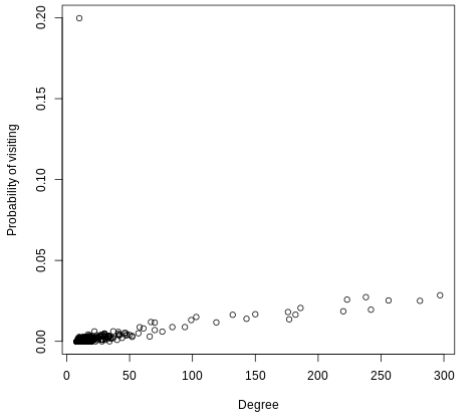
The difference between our current implementation with previous section is that the random walker only teleport the nodes with median PageRanks.

The probability that the random walker visits each node against node index and node degree in Fig. 51. We also obtain the Pearson correlation coefficient is 0.4058206 and the slope of the fitted linear line is 0.0001112607. Comparing Fig. 51 and Fig. 50, Fig. 51 (b) tend to have more dense nodes in the left-lower corner. The reason might be that random walker with median PageRanks tend to have higher probability to visit nodes with less degree.

We compare the impact of median PageRanks on the original PageRank values in Fig. 52. The node index is assigned based on the ascending order of the PageRanks of the original graph. As we can obverse in Fig. 52, the median PageRank shift the spike to the middle, which indicates the nodes with median PageRank values are more likely to be visited.



(a) Probability of random walker visiting each node



(b) Probability of random walker visiting nodes with given degree

Figure 51: Probability of random walker visitation with the node index and node degree on Merged PA network with Median PageRank teleportation(n = 900, m = 4)

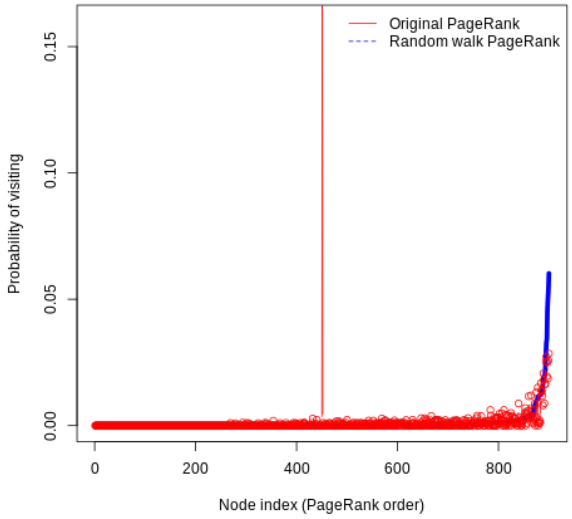


Figure 52: The PageRank value comparison between the original PageRank and Median PageRank

Part (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?

Denote A as the adjacent matrix of directed graph; O_i represents the number of out-going nodes of current node i; P as the transition matrix from PageRank. P_{ij} is defined as the probability of transition from node i to node j.

The teleported random walker is now according to the equation:

$$P_{ij} = (1-\alpha)A_{ij} + \alpha\frac{1}{(N-1)}$$

where N is number of nodes in current graph.

Now, let's consider the trusted portion of the nodes, we can denote them as T. The updated equation is now can be written as:

$$P'_{ij} = \begin{cases} (1-\alpha)\frac{1}{0_i}A_{ij} + \alpha\frac{1}{|T|}, & i \in T \\ (1-\alpha)\frac{1}{0_i}A_{ij}, & i \notin T \end{cases}$$

When the steady state is reached, for each node i, we have

$$\pi_i = \sum_{j=1}^{N} \pi_j P'_{ij}$$

Then the resulting PageRank value for each node i can be written as:

$$\pi_i = \begin{cases} (1-\alpha)\sum_{j=1}^{N} A_{ji}\pi_j + \alpha\frac{1}{|T|}, & i \in T \\ (1-\alpha)\sum_{j=1}^{N} A_{ji}\pi_j, & i \notin T \end{cases}$$