# ECE 232 Project 1 Report

July 24, 2024

Name: Kelly Furuya

## 1 Part 1: Generating Random Networks

### 1.1 Create random networks using Erdos-Renyi (ER) model

**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.**

We can create undirected random netowrks with a specific number of nodes and edge probability using the *sample_gnp* function with $n = 900$ and can use the resulting network to plot the degree distribution using the *degree_distribution* function. Figures 1-5 show the resulting degree distributions.

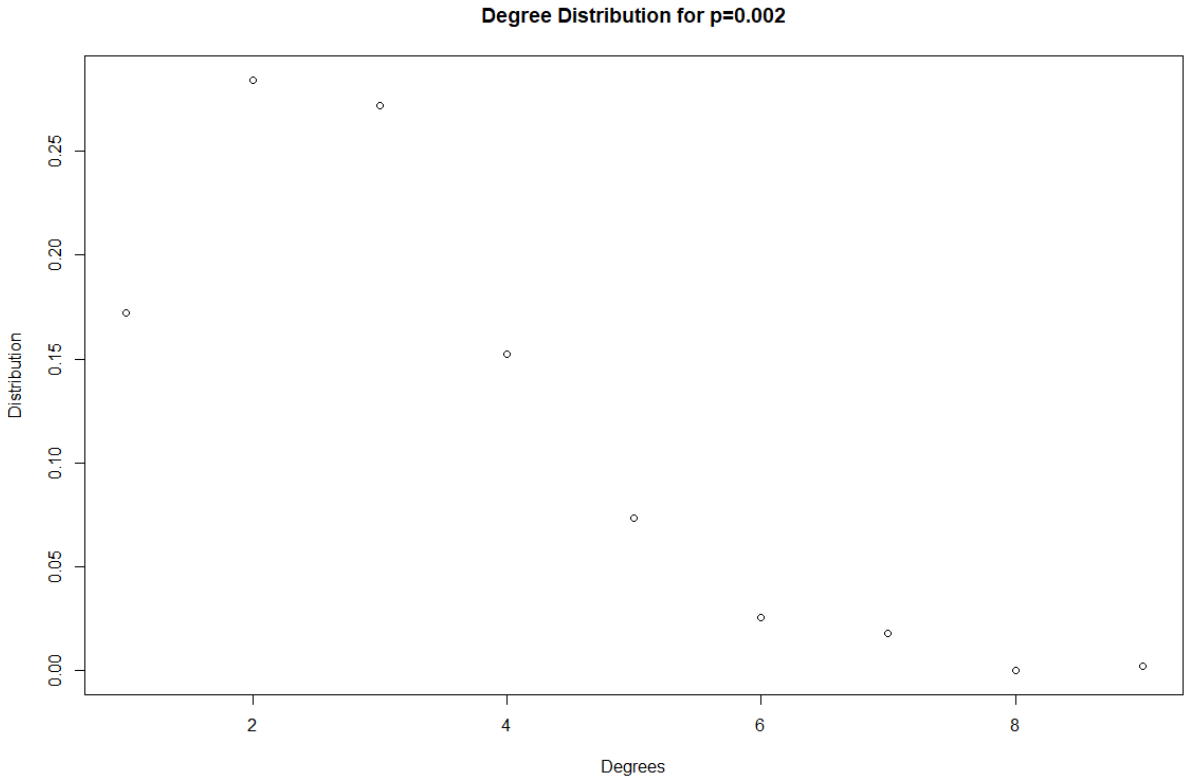


Figure 1: p=0.002

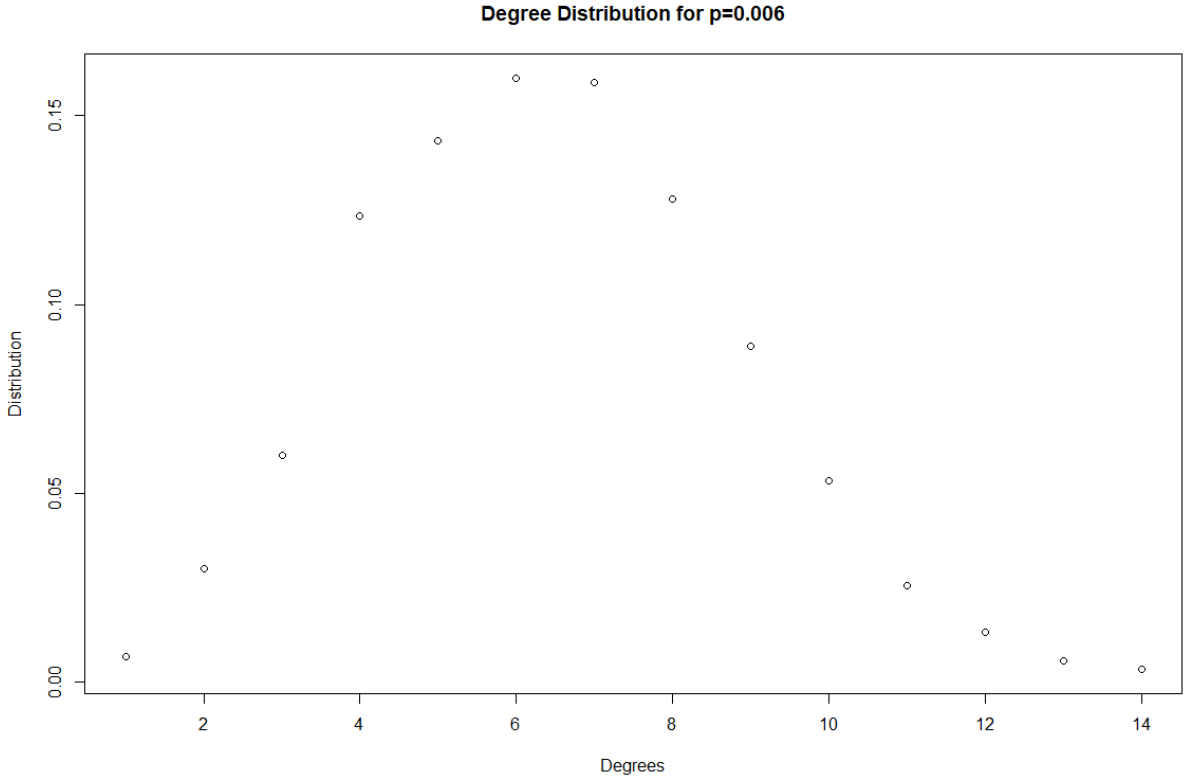


Figure 2: p=0.006

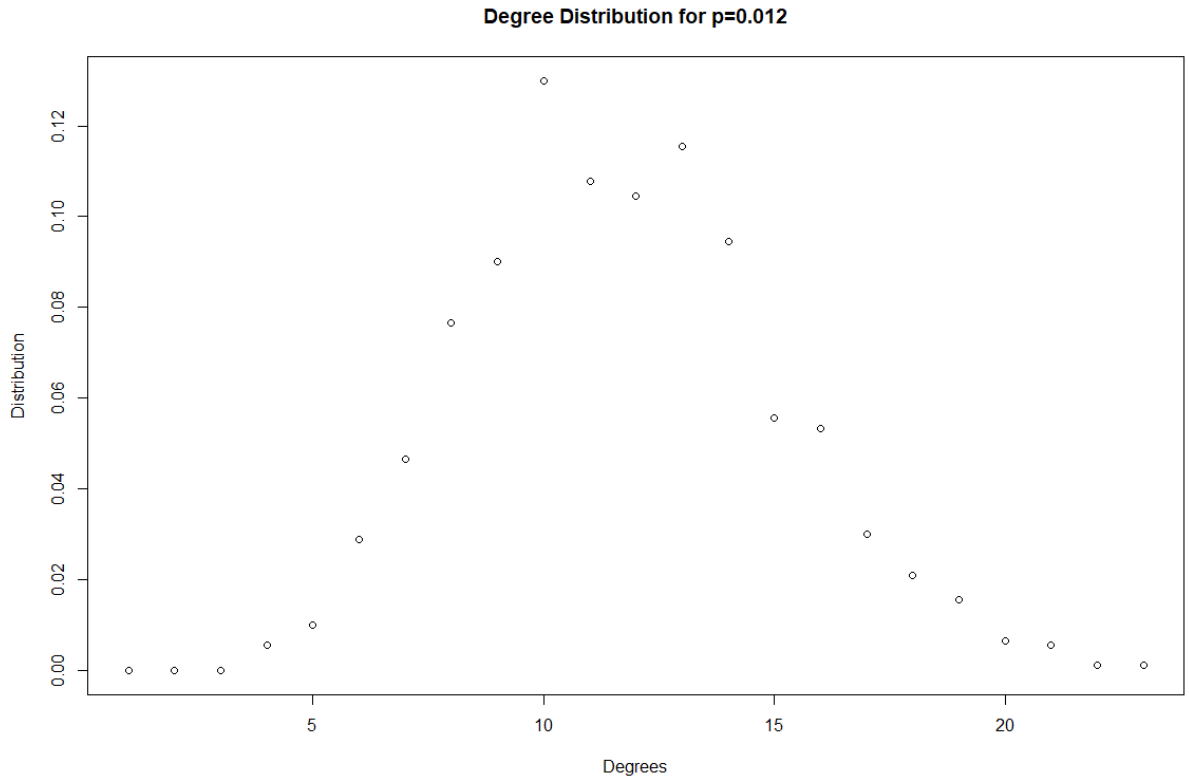


Figure 3: p=0.015

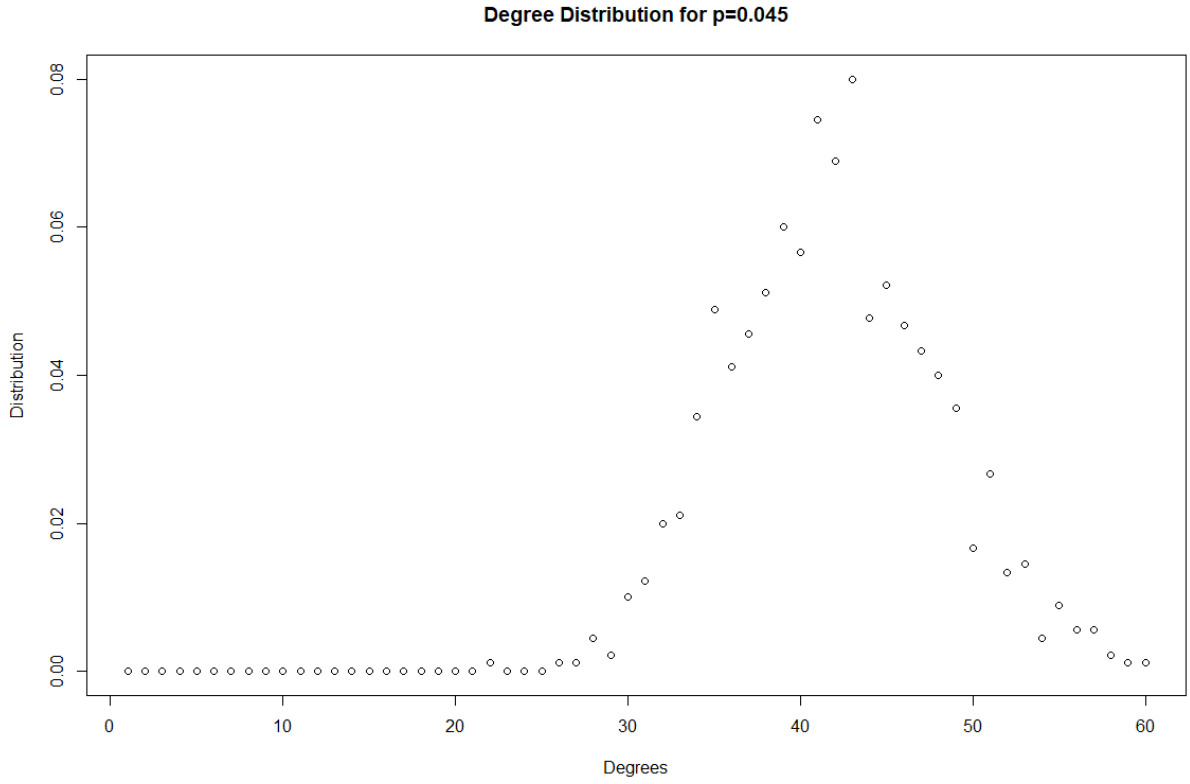


Figure 4: p=0.045

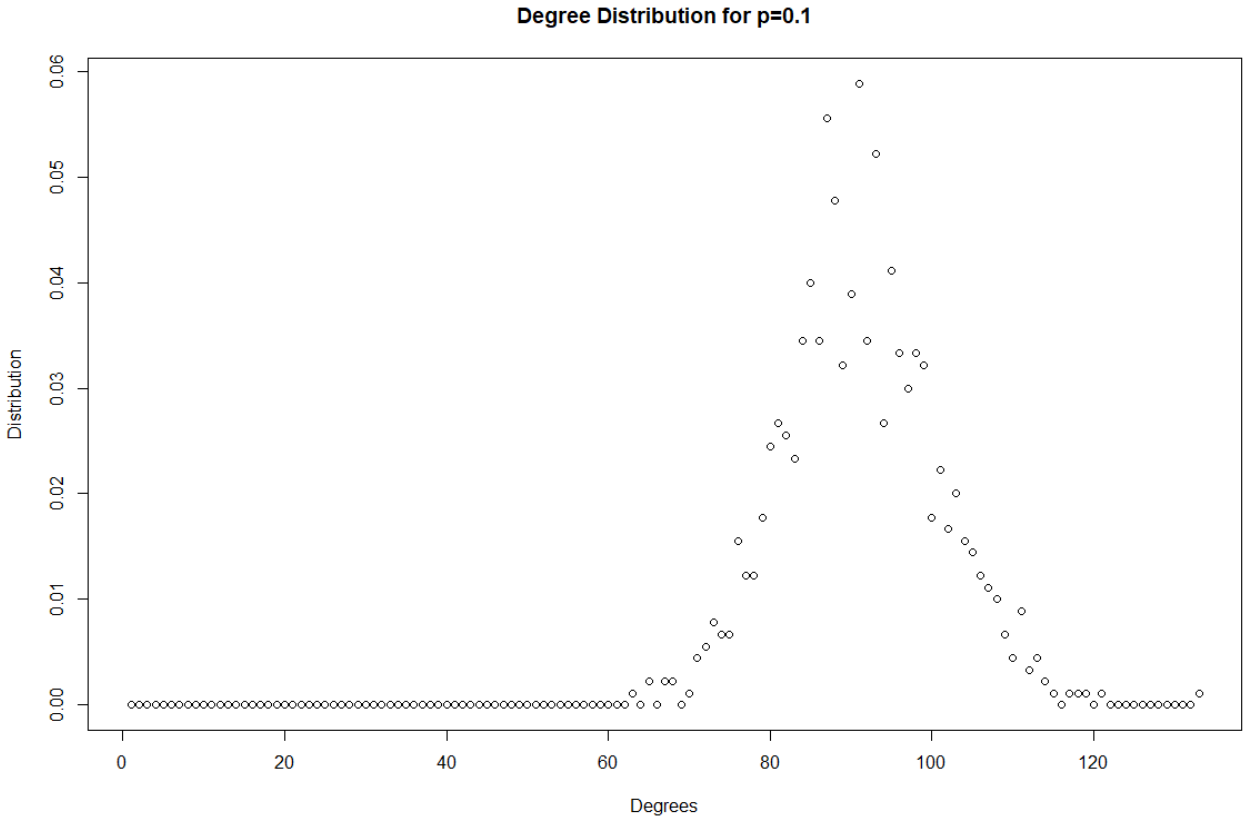


Figure 5: p=0.1

Erdos-Renyi networks tend to create a binomial degree distribution for larger networks but not always for smaller networks. The degree distribution can be represented as the equation:

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

where $n$ is the number of nodes, $k$ is the degree, and $p$ is the probability of an edge.

We know this since there are $\binom{n-1}{k}$ ways of choosing other nodes to connect to a given node. If the probability of one edge is $p$, then the odds of a given node having a degree of at least $k$, or at least $k$ edges, is $p^k$. Lastly, if we want that node to have a degree of exactly $k$, then we need to figure out the odds of no other edges connecting to it. If $p$ is the probability of an edge existing, then $1-p$ is the probability of an edge not existing and $n-1-k$ is the number of nodes not connected. Therefore, $(1-p)^{n-1-k}$ is the probability of no other edges connecting to that given node. This resulting equation matches the general formula for a binomial distribution:

$$P_k = \binom{n}{k} p^k q^{n-k} \tag{2}$$

All the graphs in this project also visually match the description for a binomial distribution as they all generate a single distinct peak.

The theoretical mean of a binomial distribution is the product of the number of trials and the probability of sucess. In this case, the number of trials is the number of nodes that can form an

edge with a given node, $n-1$, and the probabillity is $p$. The theoretical variance is the mean times the probability of failure which would be $(n-1)p(1-p)$

| **Probability** | **Mean** | **Theoretical Mean** | **Variance** | **Theoretical Variance** |
|---|---|---|---|---|
| 0.002 | 1.831111 | 1.798 | 1.987018 | 1.794404 |
| 0.006 | 5.444444 | 5.394 | 5.559758 | 5.361636 |
| 0.012 | 10.731111 | 10.788 | 10.897585 | 10.568544 |
| 0.045 | 40.837778 | 40.455 | 35.537615 | 38.634525 |
| 0.1 | 90.137778 | 89.900 | 87.920929 | 80.910000 |

This table shows the calculated and the theoretical mean and variance for each probability. The actual and theoretical values are fairly close to each other within a range of about 10%.

The actual values were found using the *mean* and *var* functions on the degrees of each network.

**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?**

In order to see if all random realizations of the ER network are connected, we can run multiple iterations of *sample_gnp* and use the *is_connected* function to see if any of the resulting networks are not connected. Once we do this, we can see that when $p = 0.002, 0.006, 0.012$, the networks are not always connected while when $p = 0.045, 0.1$, the networks are always connected.

In order to estimate the probability that a generated network is connected, we can again generate multiple graphs for each probability and then keep track of how many iterations are connected to compute the probability.

| **Edge Probability** | **Connected Probability** |
|---|---|
| 0.002 | 0 |
| 0.006 | 0.01 |
| 0.012 | 0.96 |
| 0.045 | 1 |
| 0.1 | 1 |

As expected, the probabilities for $p = 0.002, 0.006, 0.012$ are all below 1, meaning that not all generated networks will be connected, while the probabilities for $p = 0.045, 1$ are both 1, meaning that all generated networks will be connected. We can se that when $p = 0.002$, the graph will never be connected and that when $p = 0.006$, there is only a 1% chance that the network will be connected. This roughly makes sense as if the probability of an edge forming between any two given nodes is particularly low, then that means there is a much smaller chance that all of the nodes in that network will have a path between each other.

Next, we want to measure the diameter of the GCC for each probability. We can make use of the *diameter* and *largest_component* functions to do so which will find the longest of all the shortest paths between any two given nodes.

| **Probability** | **Diameter** |
|---|---|
| 0.002 | 23 |
| 0.006 | 9 |
| 0.012 | 5 |
| 0.045 | 3 |
| 0.1 | 3 |

This table shows that the higher the probability, $p$, the lower the diameter of the GCC. A higher edge probability generally implies that the network will contain many more edges, providing more opportunities for shorter paths to generate between any given nodes, decreasing a GCC's diameter.

**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 $p = O(\frac{lnn}{n})$. For $n = 900$, sweep over values of $p$ from 0 to a $p_{max}$ that makes the network almost surely connected and create 100 random networks for each $p$. $p_{max}$ 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.**

In order to find the normalized GCC sizes and their averages, we will sweep from $p = 0$ to 0.0015 at increments of 0.0005 and generate 100 new networks for each step. The upper bound of 0.0015 was chosen since we already know that $p = 0.012$ gives a 96% chance of creating a fully connected network which would have a normalized GCC size of 1. Past whatever probability that nearly guarantees a fully connected network, the normalized GCC size wiill always be 1, making it relatively uninformative to calculate and graph for any of those probabilities.

We first check if the generated network is fully connected or not. If it is, then we already know that it contains all of the nodes in the network. If it is not connected, we use the *components* and *largest_components* functions, and extract the *csize* value from the *components* result to find the size of the GCC. All of the sizes are then added up and used to calculate the average while the normalized size is found by dividing the size of the component by the total number of nodes in the network, 900 in this case.

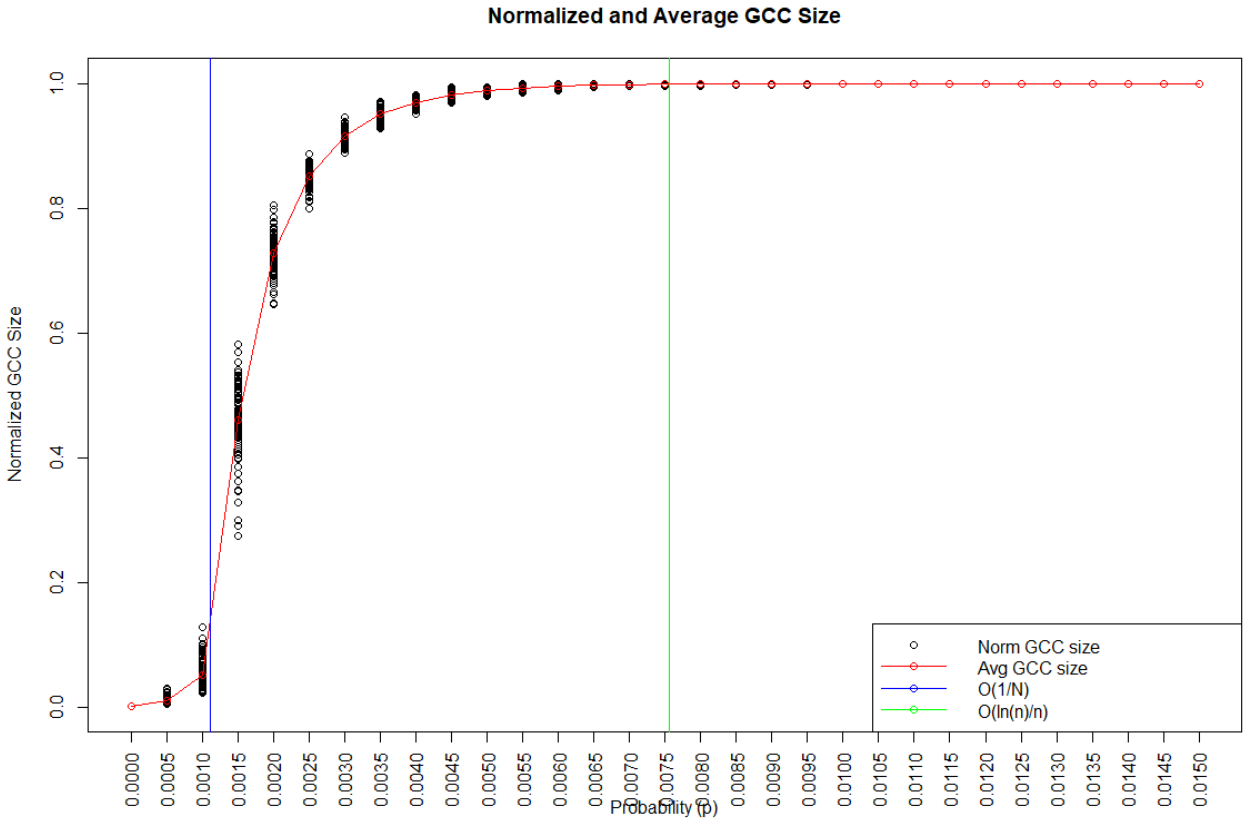


Figure 6: Normalized and Average GCC size vs Probability with $O(\frac{1}{n})$ and $O(\frac{\ln n}{n})$

Figure 6 shows the normalized GCC sizes for each probability along with the average GCC size in red. The blue and green lines indicate where $p = \frac{1}{n}$ and $\frac{\ln n}{n}$. The overall shape of the plot shows that the GCC size starts out very low at low probabilities before rapidly increasing and then leveling off. The blue line shows that the GCC size begins to greatly increase when $p = \frac{1}{n}$ while when $p = \frac{\ln n}{n}$, the GCC size stabilizes and levels off with little to no increase afterwards.

### c.1 Empirically estimate the value of p where a giant connected component starts to emerge (define your criterion of "emergence")? Do they match with theoretical values mentioned or derived in lectures?

We will define the "emergence" of a giant connected component as when a significant portion of the nodes in a network are connected in a single cluster. In this case, we can define it as when at least half of all nodes are in the largest connected cluster. For these undirected random networks, it would be when the normalized size of the largest connected cluster reaches 0.5. Based on the graph produced above in Figure 6, $p$ would be somewhere around 0.00175.

The theoretical value should be when $p > 1/n$. In this case, $1/n = 0.0011$ which is 0.00065 off from the empirically calculated value.

### c.2 Empirically estimate the value of p where the giant connected component takes up over 99% of the nodes in almost every experiment.

In order to empirically estimate the value of $p$ where the GCC contains over 99% of the nodes, we need to zoom in on the plot created in Figure 6.

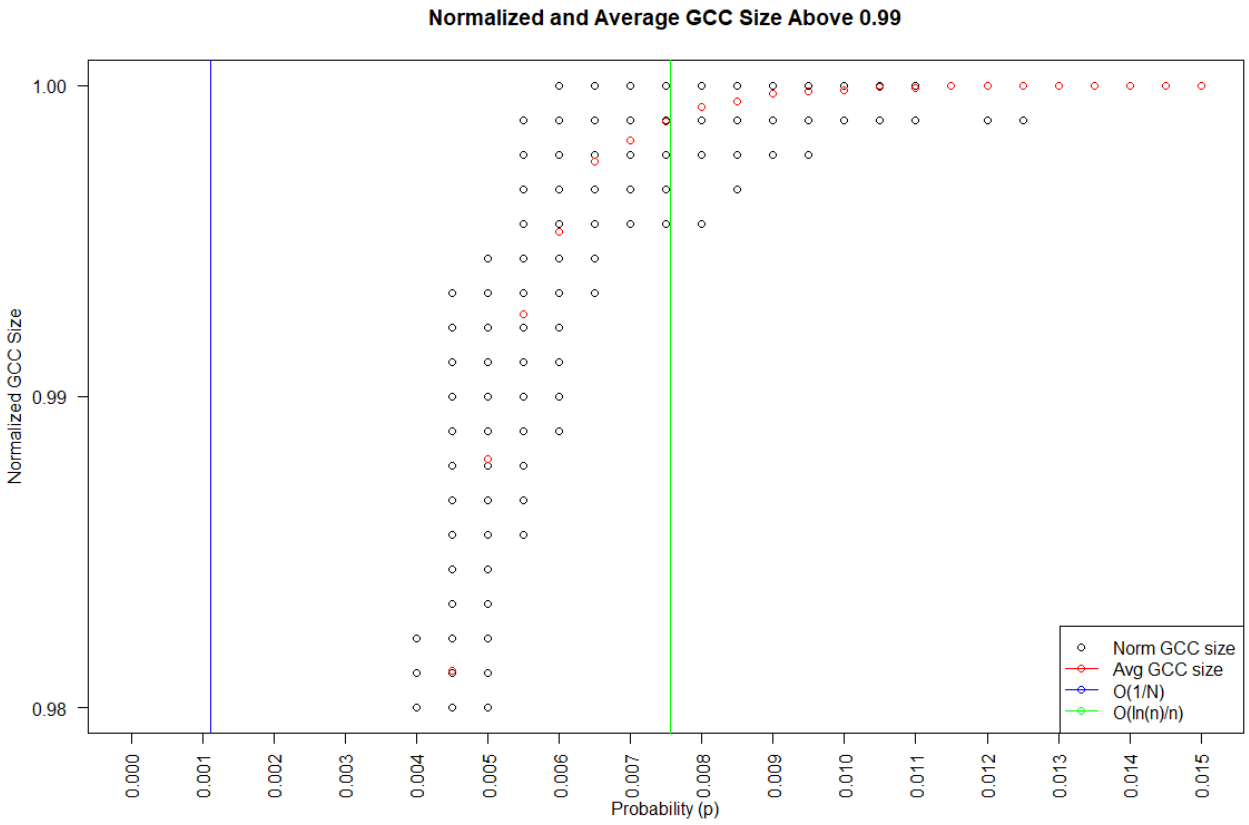


Figure 7: Closer look at the normalized and average size of the GCCs

We can see from Figure 7, that nearly all of the normalized GCC sizes are above 0.99 by the time $p$ reaches 0.006. By 0.007, all of the GCCs contain over 99% of the nodes in the generated graphs for that run. Each graph will end up slightly different from the last, and it is possible for a graph to be generated with a normalized GCC size below 0.99. However, Figure 7 still supports the idea that when $p = 0.007$, over 99% of the nodes will be contained in the GCC in almost every experiment.

### d

### d.1 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 edgeformation probabilities $p = c/n$, as a function of $n$. What trend is observed?

Much like before, we will need to create several randomly generated networks in order to calculate the expected GCC for a network with $n$ between 100 and 10000 and $p = c/n$. We iterate over the

total number of nodes in steps of 100 to decrease the total time it takes to calculate the values. To find the expected GCC size, we average the size of the GCC for each $n$ and $p$ combination.

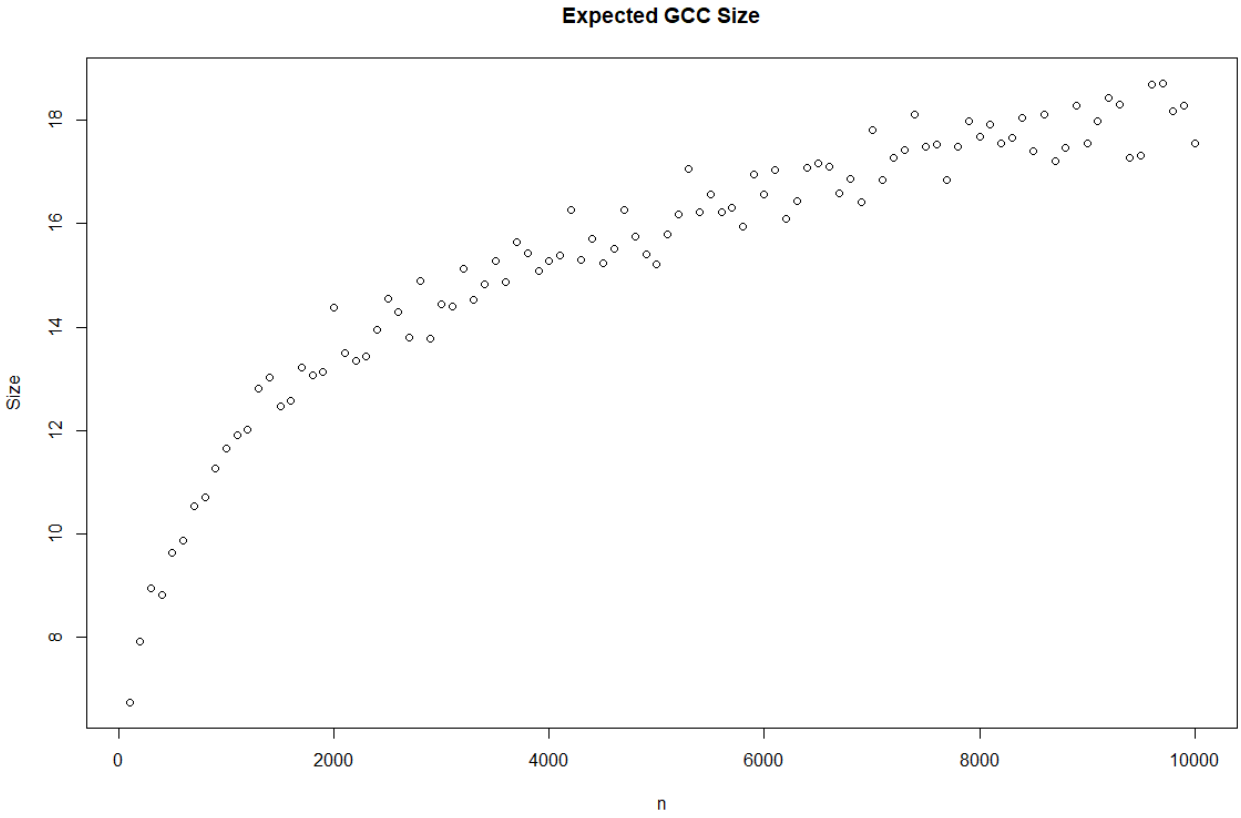


Figure 8: Expected GCC size vs number of nodes

Figure 8 shows that as the number of nodes increases, so does the expected size of the GCC. This makes sense if we consider the equation originally given to us that the average degree of a node is $c = n * p = 0.5$. As the number of nodes increase, the probability of an edge forming must decrease. Figure 6 already showed that smaller probabilities result in larger GCC sizes, supporting the results in Figure 8. In general, the increase in size is greatest at the lower end of $n$ before becoming a more gradual and constant slope.

### d.2 Repeat the same for $c = 1$.

We repeat the process as above except replace $c = 0.5$ with $c = 1$.

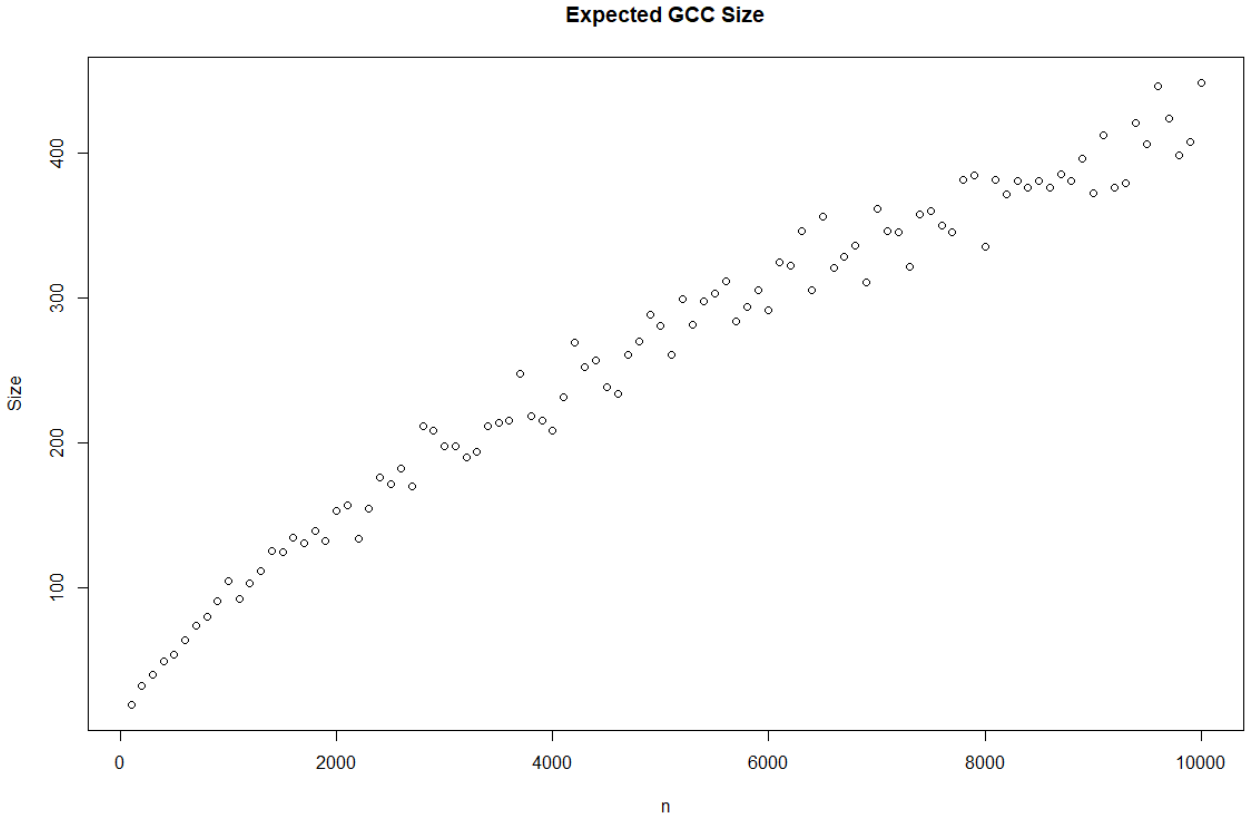


Figure 9: Expected GCC size with $c = 1$

The resulting graph in Figure 9 still shows the general trend that more nodes translates into a larger expected GCC size. However, the distribution appears much more linear as opposed to the plot in Figure 8 with a lower initial slope for the lower number of nodes and a slightly higher constant slope once the graph stabilizes.

### d.3 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.

We repeat the process again for the three values of $c$.

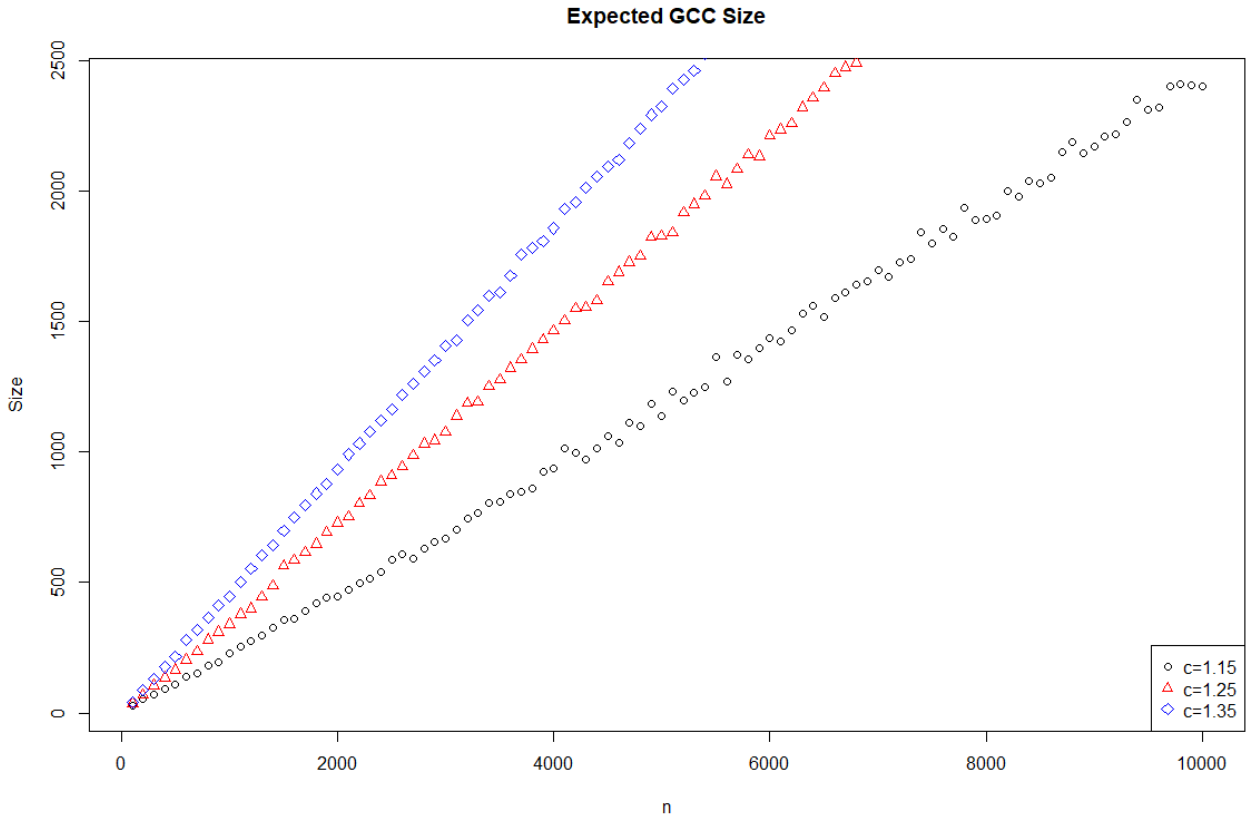


Figure 10: Expected GCC size for $c = 1.15, 1.25, 1.35$

Figure 10 shows that all three values of $c$ create very linear graphs where the slope increases as $c$ increases.

### d.4 What is the relation between the expected GCC size and $n$ in each case?

For $c = 0.5$, the relationship between the expected GCC size and $n$ looks almost logmarithic because of the rapid increase followed by the constant slope after a certain $n$. $c = 1, 1.15, 1.25, 1.35$, however, all have a linear relation between GCC size and $n$.

## 1.2 Create networks using preferential attachment model

**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?**

For this graph, we will use the function *sample_pa* to create an undirected network with a preferential attachment model. The difference between this model (called a Barabasi-Albert model) and the Erdos-Renyi model is that it will assign different probabilites between nodes based on the degree of a node instead of assigning all edges the same probability of forming. The Barabasi-Albert model is sometimes described as "the rich get richer" for it's preference to form edges with nodes of higher degrees. Using the same process as question 1 of testing multiple versions of this graph for connectivity, we can see that this graph is always connected. This makes sense as, unlike the

Erdos-Renyi method, there is no probability of an edge not forming, just varying probabilities of which nodes it will connect.

**b Use fast greedy method to find the community structure. Measure modularity. Define Assortivity. Compute Assortativity.**

The *cluster_fast_greedy* function uses the fast greedy algorithm to map the community structure of a network. We can then use the *modularity* function to find the modularity of the community structure which represents how well a network is connected within clusters. A higher modularity indicates that a network is well connected within communitites but nodes are weakly connected to nodes outside of their cluster.

We will define assortivity as the tendency or preference of nodes in a network to be connected to other nodes that have similar attributes. A positive assortativity indicates a correlation between nodes of similar degree while a negative assortativity indicates a correlation between nodes of different degree. We can use the *assortativity* function to calculate it.

| **Modularity** | **Assortativity** |
|---|---|
| 0.935789 | -0.08745464 |

**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?**

We repeat the same process as above except for a graph with $n = 10500$ nodes.

| **n** | **Modularity** | **Assortativity** |
|---|---|---|
| 1050 | 0.935789 | -0.08745464 |
| 10500 | 0.9785821 | -0.03125745 |

The table above shows that the modularity and assortativity both increased when the number of nodes increased. This implies that the network with more nodes is more densely connected within clusters and is more likely to connect to nodes of similar degree.

**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, we convert the x and y values to their logarithmic equivalent values.

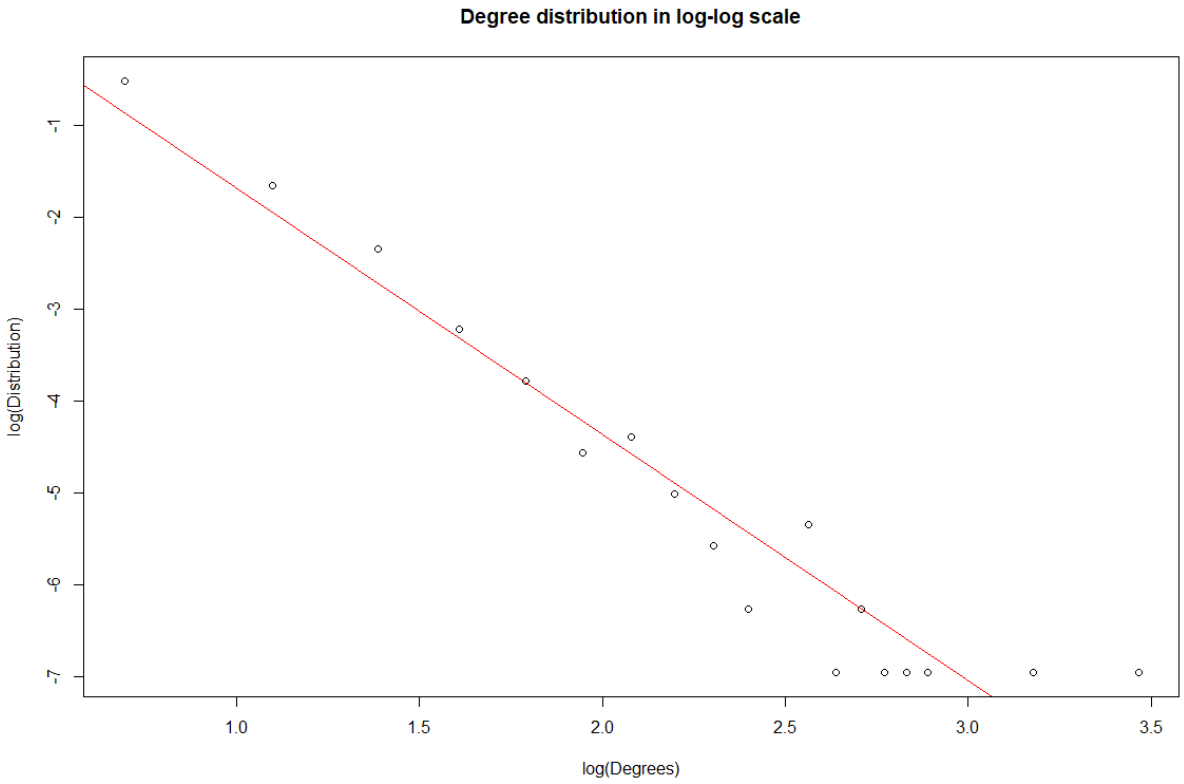


Figure 11: Log-Log Scale Degree Distribution for $n = 1050$.

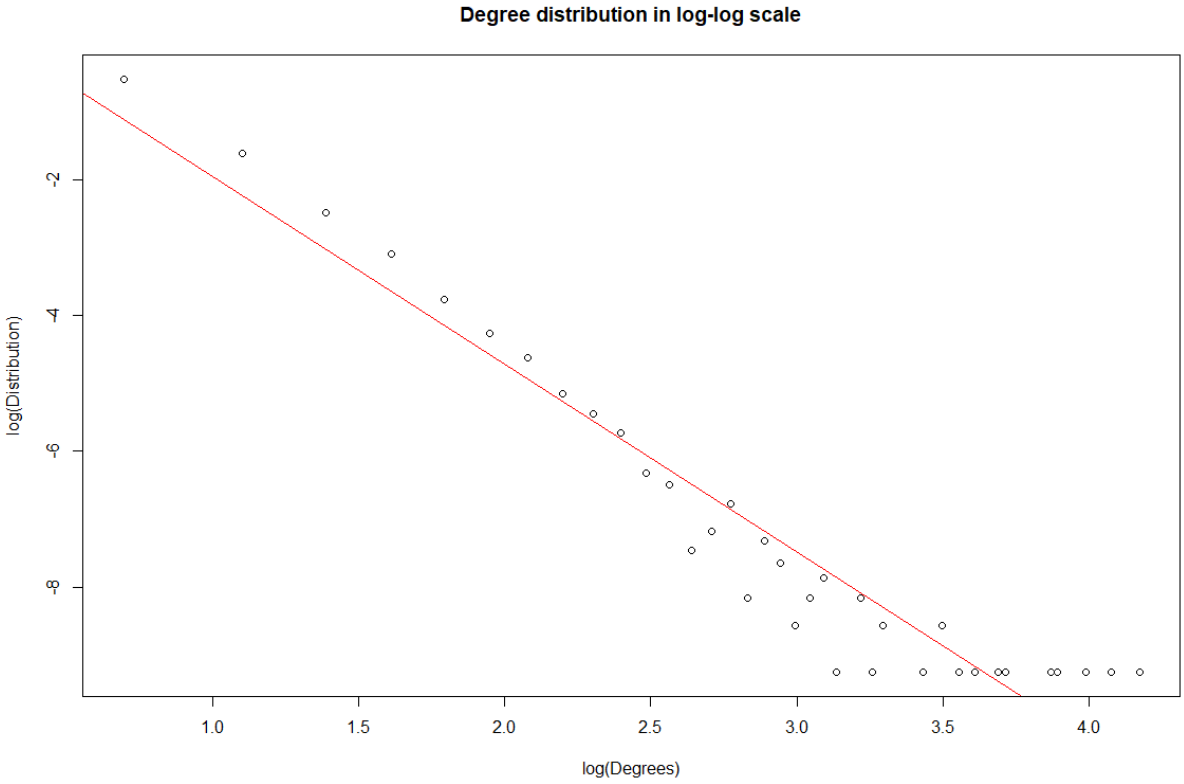


Figure 12: Log-Log Scale Degree Distribution for $n = 10500$

Once plotted, we can see in Figure 11 and 12 that the distributions are linear with a negative slope when the axis are in a log scale. This indicates that they follow the Power Law meaning that they follow the equation:

$$f(cx) = a(cx)^{-k} \tag{3}$$

where $k$ represents the slope of the graph. Using linear regression (red line), we can then calculate the slope.

| **n** | **Slope** |
|---|---|
| 1050 | -2.680574 |
| 10500 | -2.765738 |

Even if we try to empirically estimate the slope of the graph by looking at the linear regression line, we would get that the slope for $n = 1050$ is about -2.5, and the slope for $n = 10500$ is about -2.75 which are both very close to the actual slopes.

**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?**

To find the neighbor of a node, we can use the *adjacent_vertices* function which returns all of the nodes that the input node connects to. Then *sample* can select a random node as $j$.

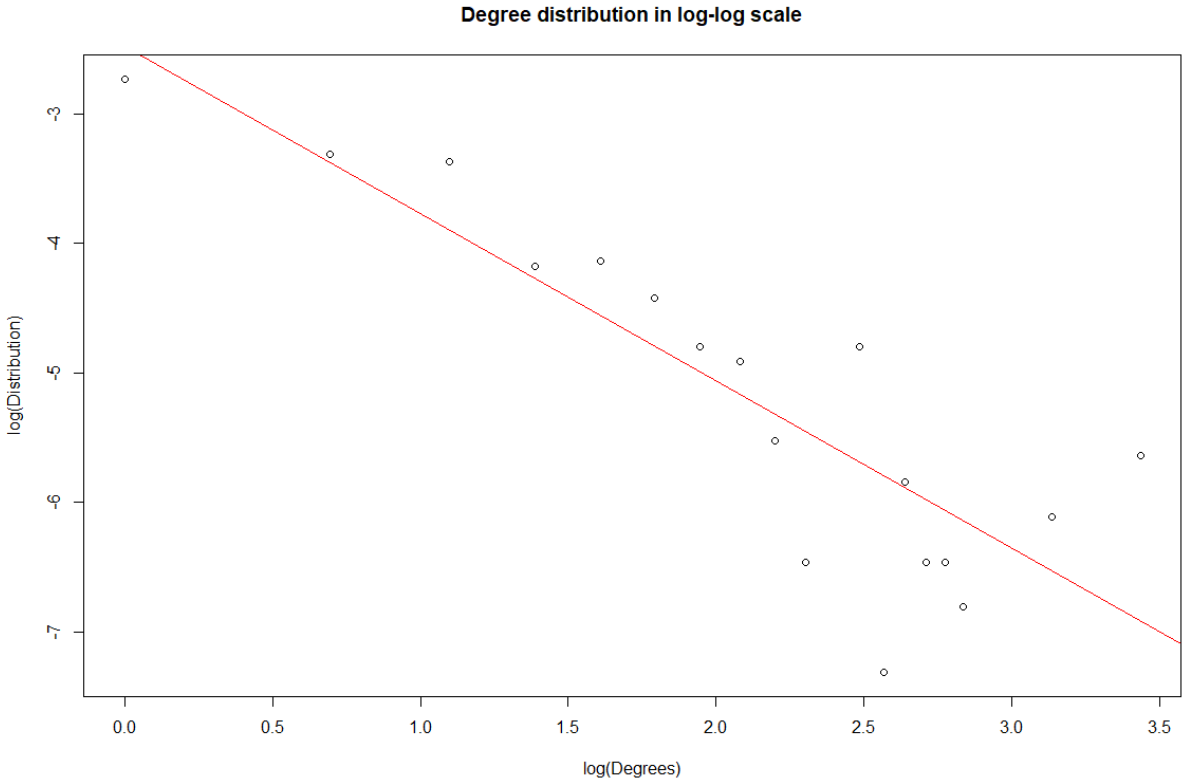


Figure 13: Degree distribution for randomly selected adjacent nodes; $n = 1050$

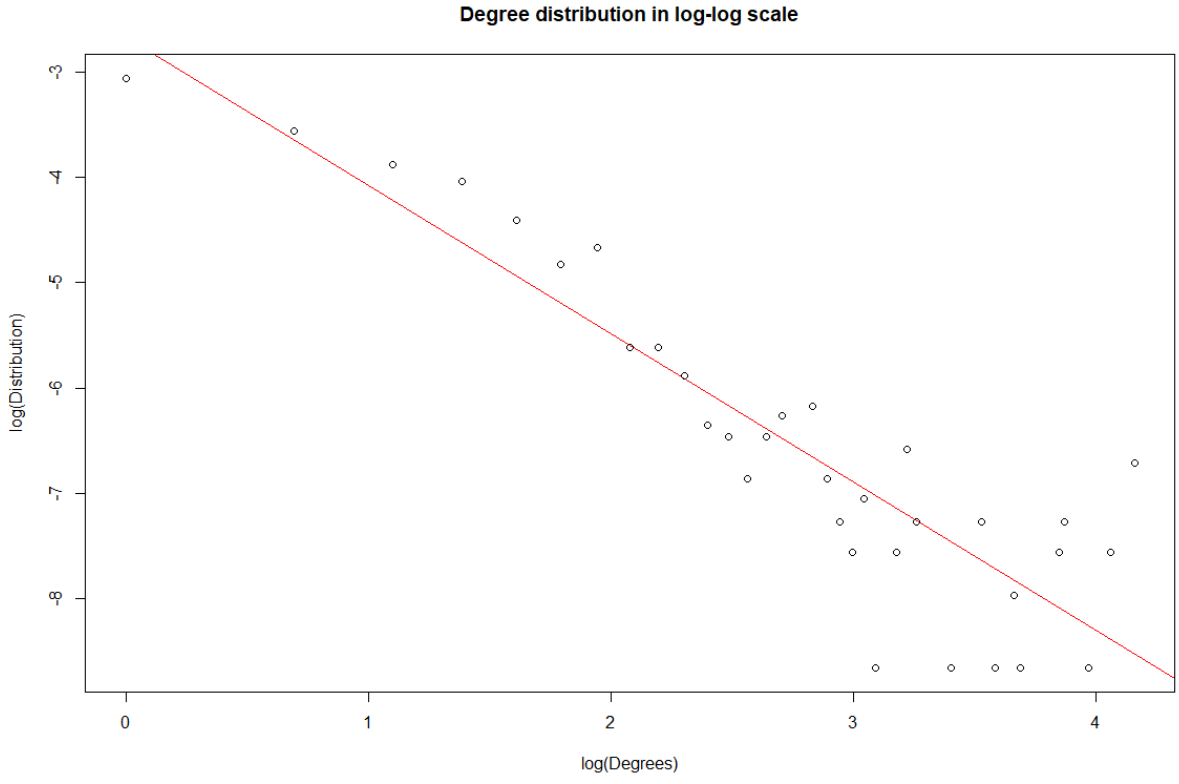


Figure 14: Degree distribution for randomly selected adjacent nodes; $n = 10500$

Figures 13 and 14 show that these degree distributions still follow the Power Law and produce a linear distribution, but the points are slightly more spread out than the original networks. We can then use the same process as before to calculate the slopes and then compare them to original networks'.

| **n** | **Original Slope** | **Random Slope** |
|---|---|---|
| 1050 | -2.680574 | -1.291583 |
| 10500 | -2.765738 | -1.406908 |

This table shows that the randomly selected nodes produce a degree distribution with a shallower slope indicating that the random selection selects nodes in a much more even distribution and is slightly less likely to favor nodes of a certain degree over another.

**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.**

Similar to when we calculated the expected size of a GCC, we need to create multiple iterations of the network and record all of the degrees in order to find an average for each step.

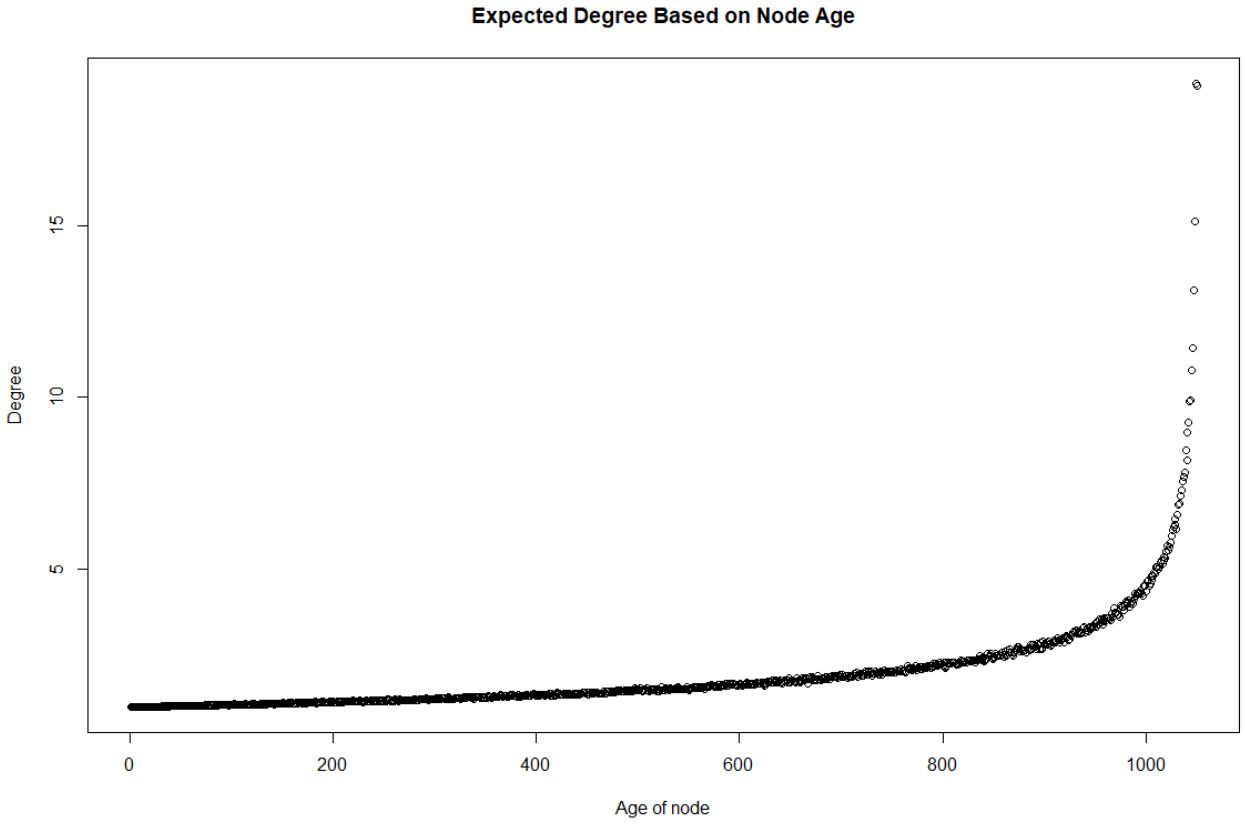


Figure 15: Expected degree of a node based on its age

Figure 15 shows a clear trend that a section of the oldest nodes have a significantly higher degree than all other nodes, consequently resembling a Power Law graph when not graphed in

log scale. This trend can be explained by Barabasi-Albert networks tending to form edges with nodes of a higher degree. Once the first edge is formed, the next node will have an equal chance of selecting either of the two initial nodes. After that second edge, whichever node ended up with a degree of 2 is more likely to be selected to form the next edge with the new node. This process repeats, compounding on itself, resulting in the distribution shows in Figure 15 where the oldest nodes accumulated the most edges.

We can test the accuracy of the graph since we know that the expected degree of a node, $k$, created at time step $i$, at time step $t$ follows the equation:

$$k_{i,t} = m(\frac{t}{i})^{\frac{1}{2}} \tag{4}$$

where $m$ is the number of edges formed at each step. If we calculate the values for $i = 200, 600, 1000$ we get that the expected degrees will be $k = 2.236, 1.291, 1$. Keeping in mind that the graph's x-axis indicates the age of the node (meaning the opposite of the time step when the node was created), we can see that the values more or less line up with the graph.

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

We repeat each of the processes with the new $m$ values.

**a.** Are the networks always connected?

Yes, they are still always connected just like the original network with $m = 1$. Since $m$ controls the number of edges generated at each step, a higher $m$ means a higher likelyhood that a path will be generated between any two given nodes.

**b. & c.** Use fast greedy method to find the community structure, modularity, and assortativity. Repeat for $n = 10500$.

| **n** | **m** | **Modularity** | **Assortativity** |
|---|---|---|---|
| 1050 | 1 | 0.935789 | -0.089745464 |
| 1050 | 2 | 0.549584 | -0.05231068 |
| 1050 | 6 | 0.2508181 | -0.02894907 |
| 10500 | 1 | 0.9785821 | -0.03125745 |
| 10500 | 2 | 0.5311988 | -0.01353158 |
| 10500 | 6 | 0.2512942 | 0.0008323023 |

The table above shows that as $m$ increase, the modularity decreases while the assortativity increases. Since more edges are formed at each step when $m$ increase, there are more chances for nodes to form edges that connect clusters, also increasing assortativity. Interestingly, when the number of nodes increased for $m = 2$, modularity actually went down slightly, constrasting the results from 2c.

If we examine the plots of the networks, we can see just how much the increase in nodes and $m$ can affect the overall structure.

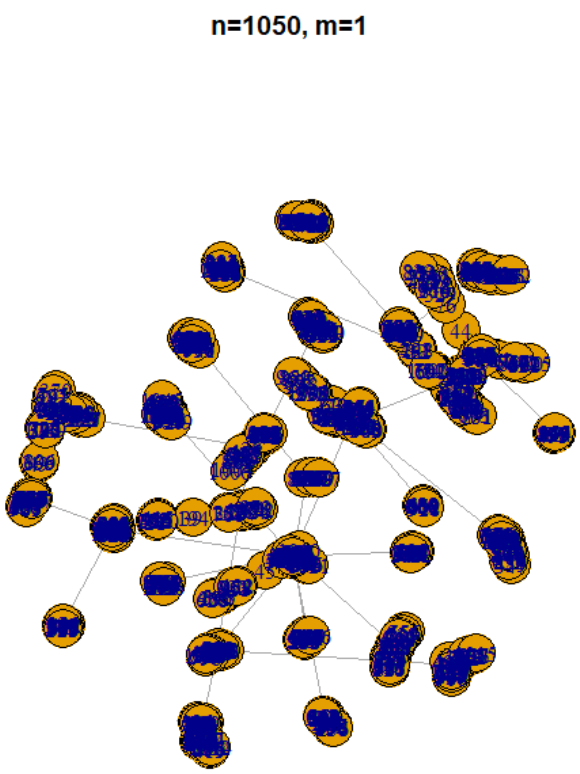


Figure 16: Nework structure for $n = 1050$, $m = 1$

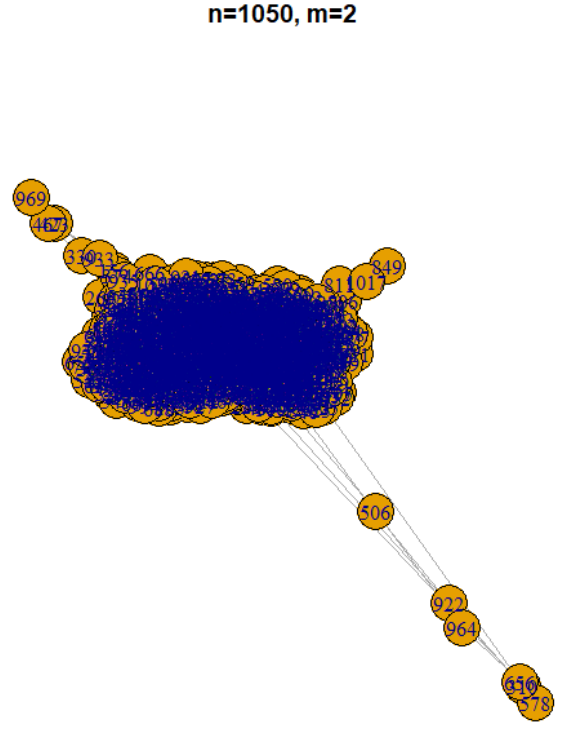


Figure 17: Nework structure for $n = 1050$, $m = 2$

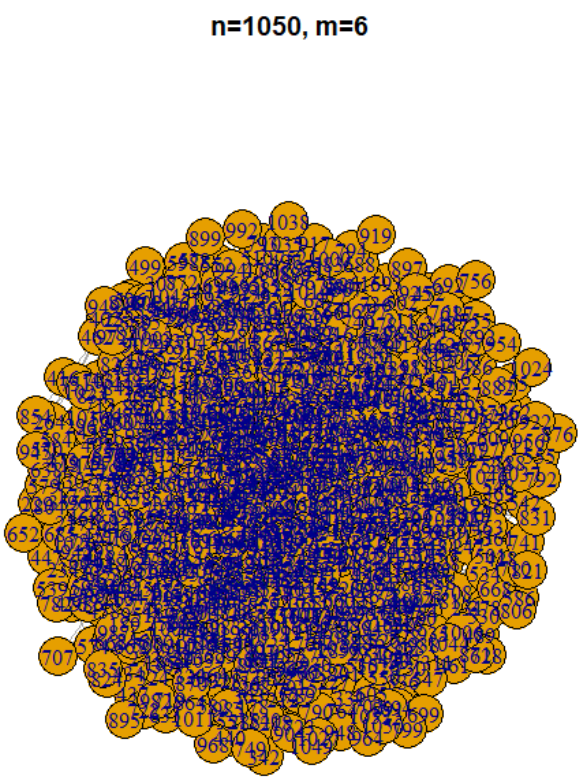


Figure 18: Nework structure for $n = 1050$, $m = 6$

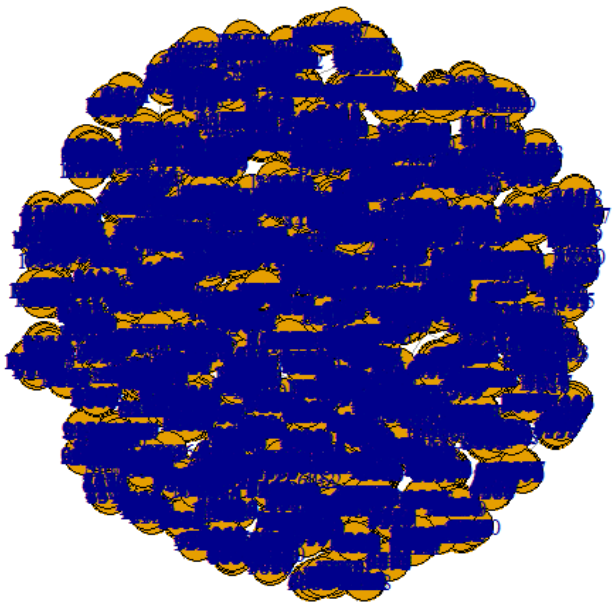


Figure 19: Nework structure for $n = 10500$, $m = 1$

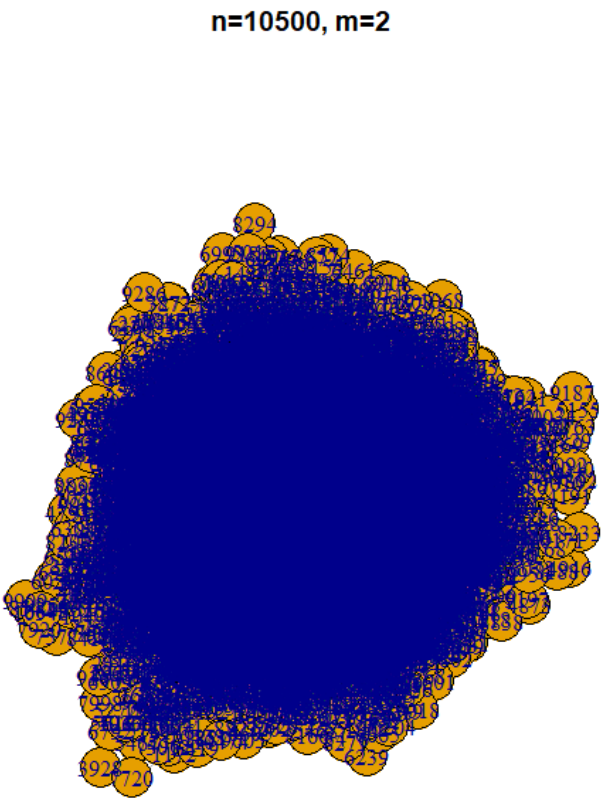


Figure 20: Nework structure for $n = 10500$, $m = 2$

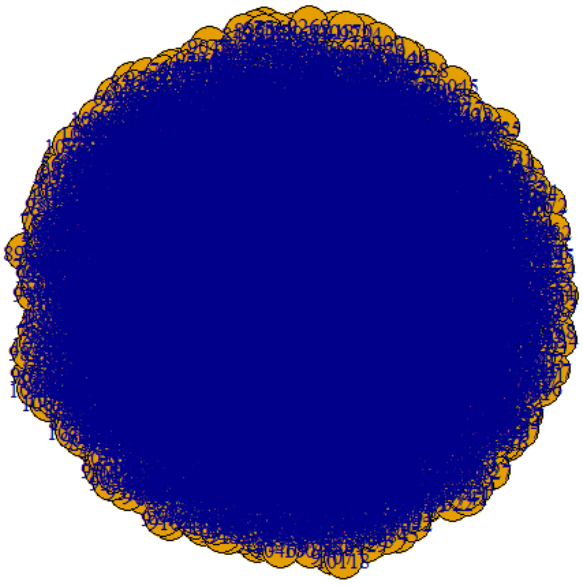


Figure 21: Nework structure for $n = 10500$, $m = 6$

Figures 16 to 21 clearly show that, as $m$ increases, the structure of the network becomes much denser with fewer and fewer community structures outside of the main one. This follows the decrease in modularity (more likely to connect to nodes outside of the node's community), and increase in assortativitiy (more even distribution of node degrees means edges are more likely to form between any two given nodes even if in different communities).

**d.** Plot the degree distributions in a log log scale and find the slope.

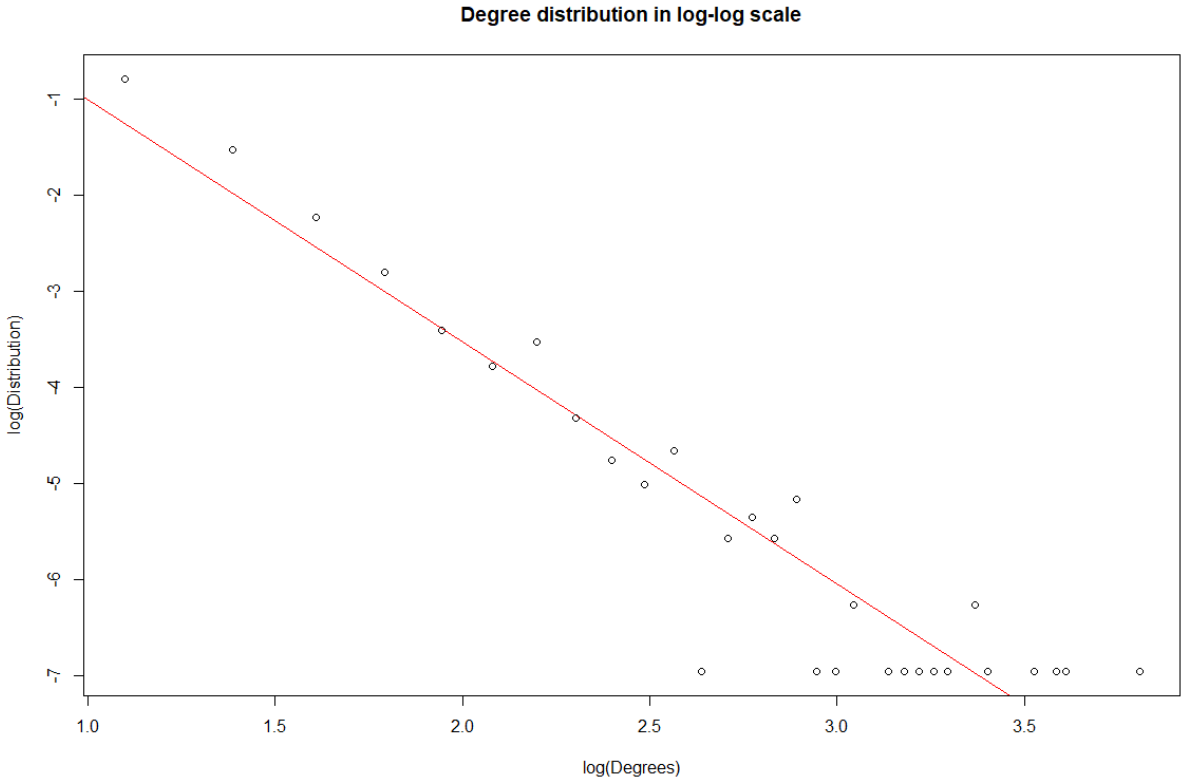


Figure 22: Log scale degree distribution for $n = 1050$, $m = 2$

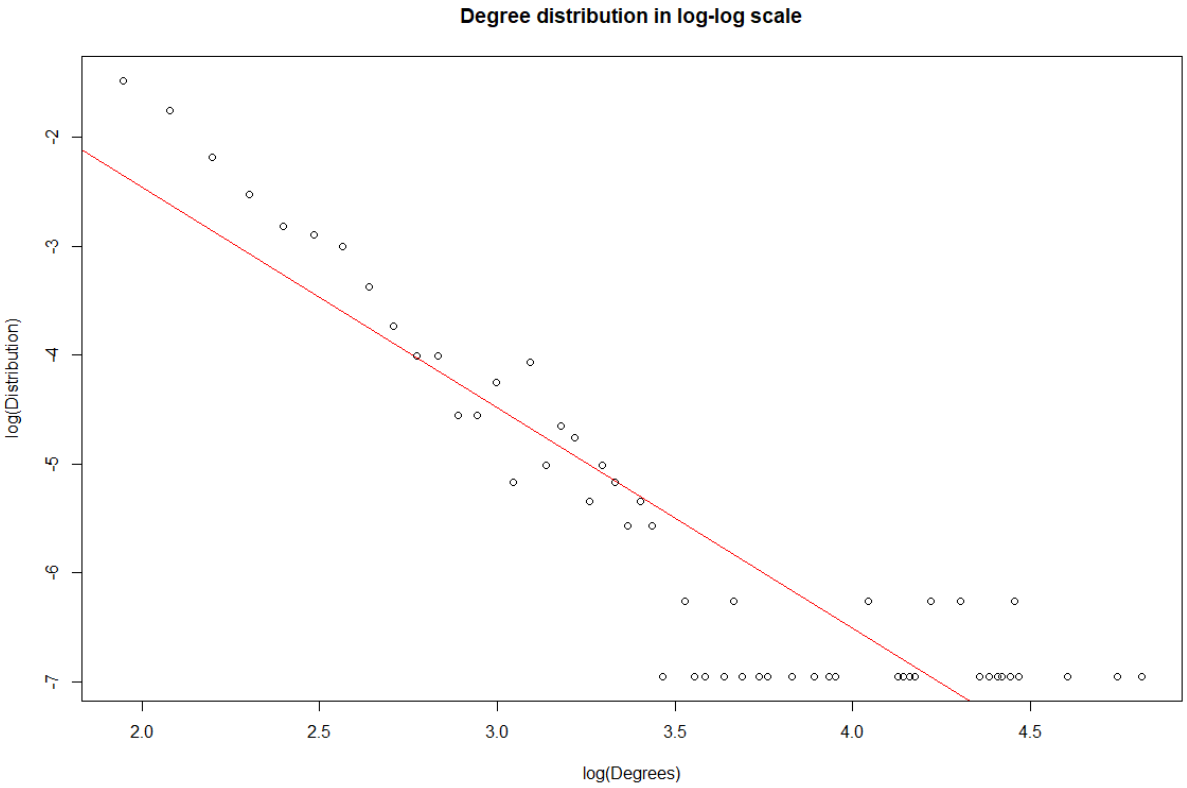


Figure 23: Log scale degree distribution for $n = 1050$, $m = 6$

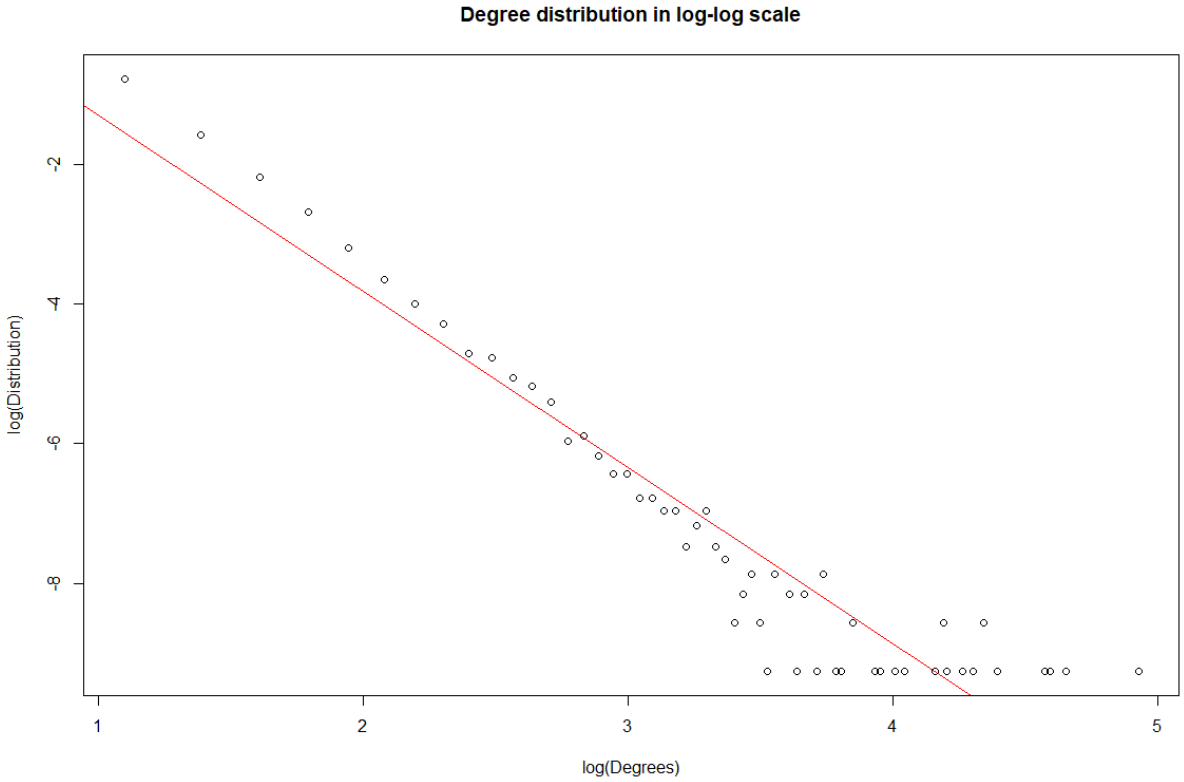


Figure 24: Log scale degree distribution for $n = 10500$, $m = 2$

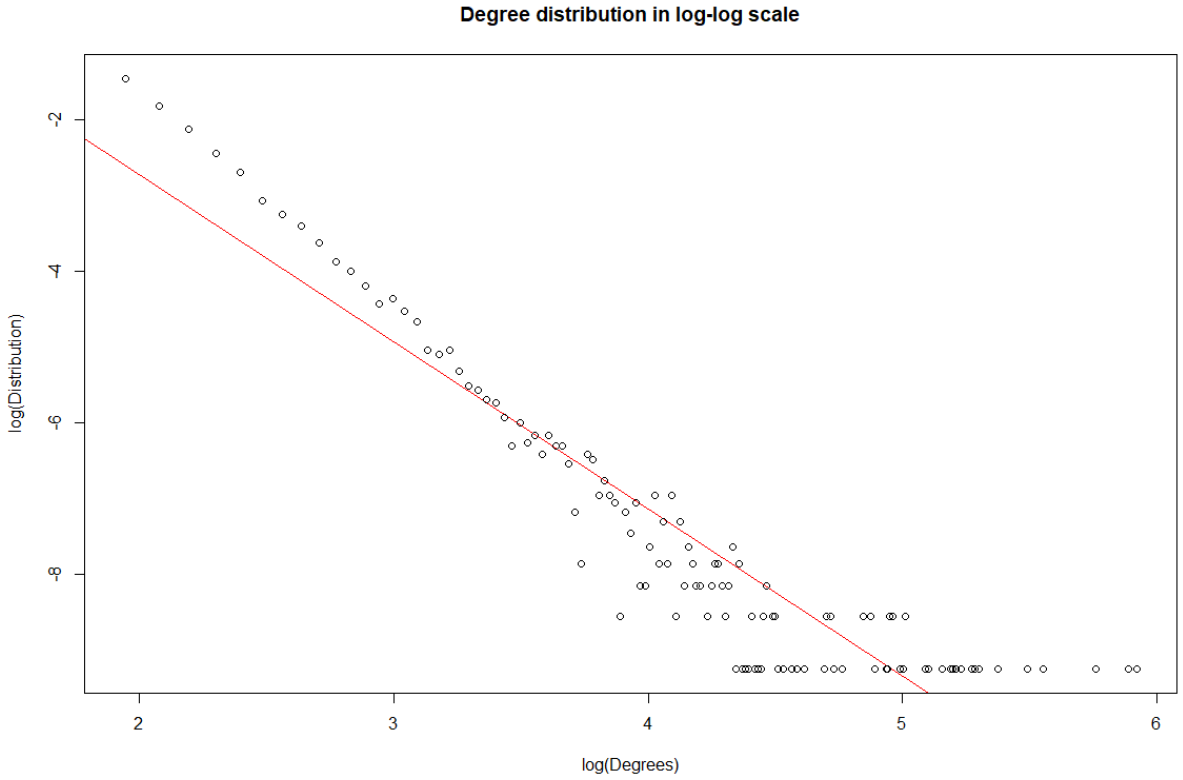


Figure 25: Log scale degree distribution for $n = 10500$, $m = 6$

| **n** | **m** | **Slope** |
|---|---|---|
| 1050 | 1 | -2.680574 |
| 1050 | 2 | -2.52103 |
| 1050 | 6 | -2.0267 |
| 10500 | 1 | -2.765738 |
| 10500 | 2 | -2.522121 |
| 10500 | 6 | -2.211665 |

The log scale degree distributions are still linear for all of the networks. The slopes tended to become less steep as $m$ increased which goes along with the idea that a decrease in modularity and an increase in assortativity would result in a slighly more even degree distribution.

**e.** Randomly pick an adjacent node, plot the degree distributions, and find the slopes.

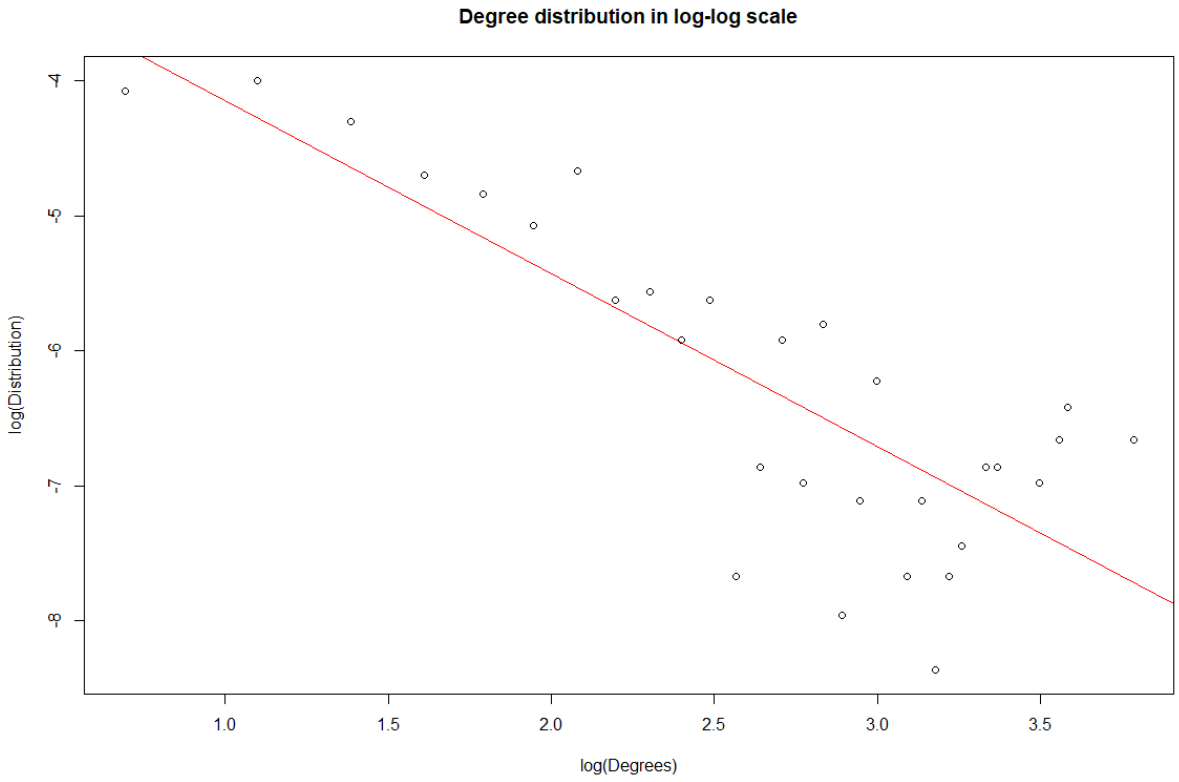


Figure 26: Log scale degree distribution for $n = 1050$, $m = 2$

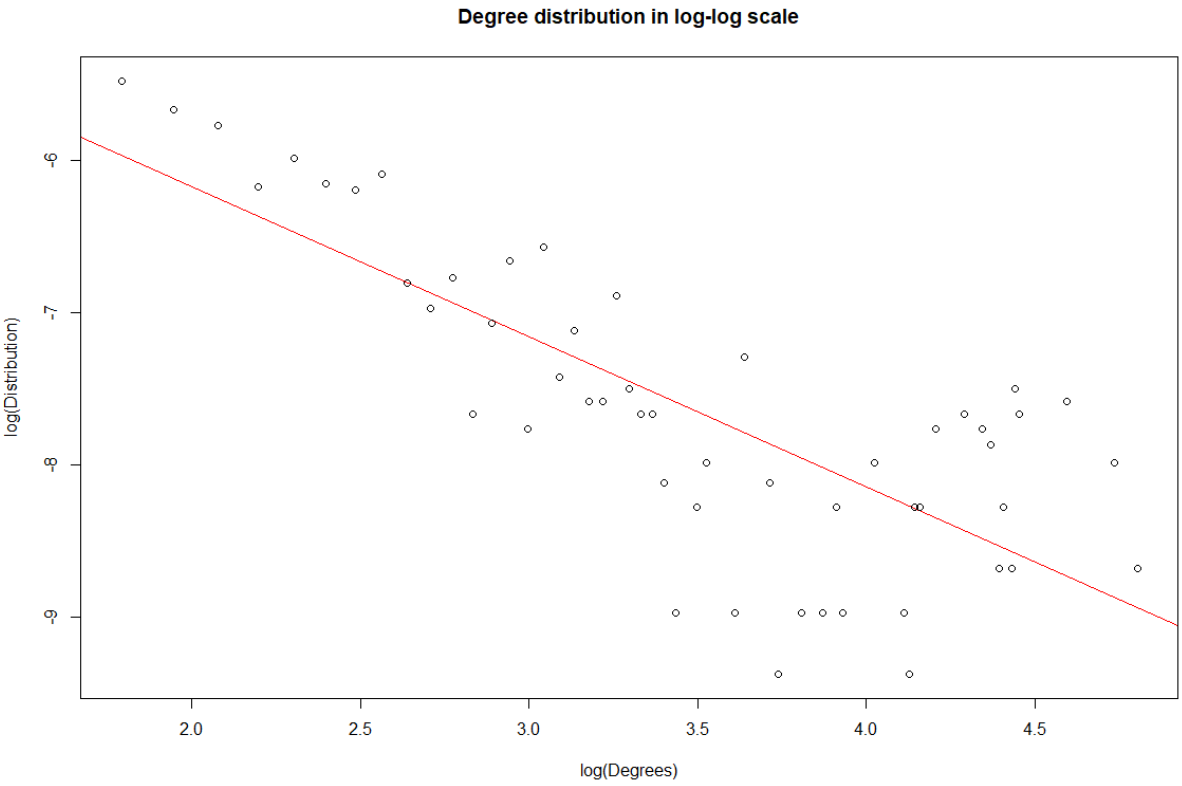


Figure 27: Log scale degree distribution for $n = 1050$, $m = 6$

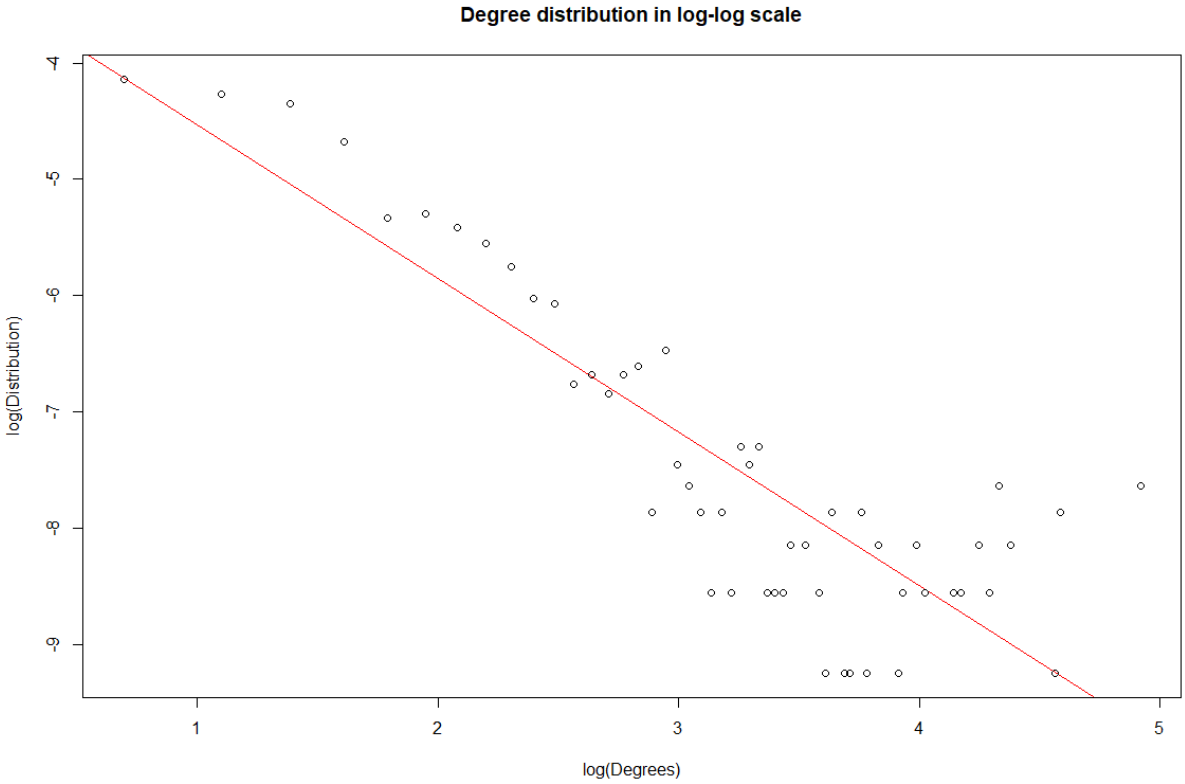


Figure 28: Log scale degree distribution for $n = 10500$, $m = 2$

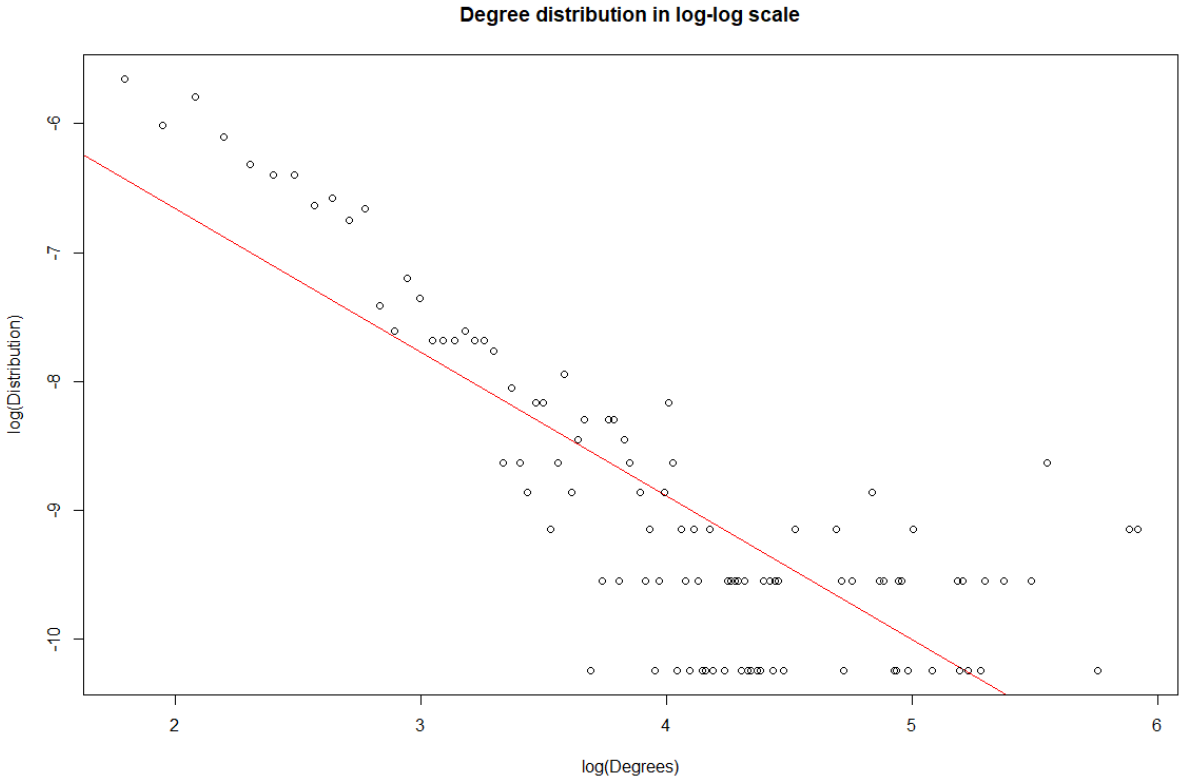


Figure 29: Log scale degree distribution for $n = 10500$, $m = 6$

| n | m | Slope |
|---|---|---|
| 1050 | 1 | -1.291583 |
| 1050 | 2 | -1.281989 |
| 1050 | 6 | -0.9843672 |
| 10500 | 1 | -1.406908 |
| 10500 | 2 | -1.31982 |
| 10500 | 6 | -1.115884 |

With the random neighbor networks, the slope of the log scale degree distribution becomes shallower when $m$ increases, indicating that the degree distribution becomes more even as the number of edges increases.

**f.** Estimate the expected degree of a node at a time step.

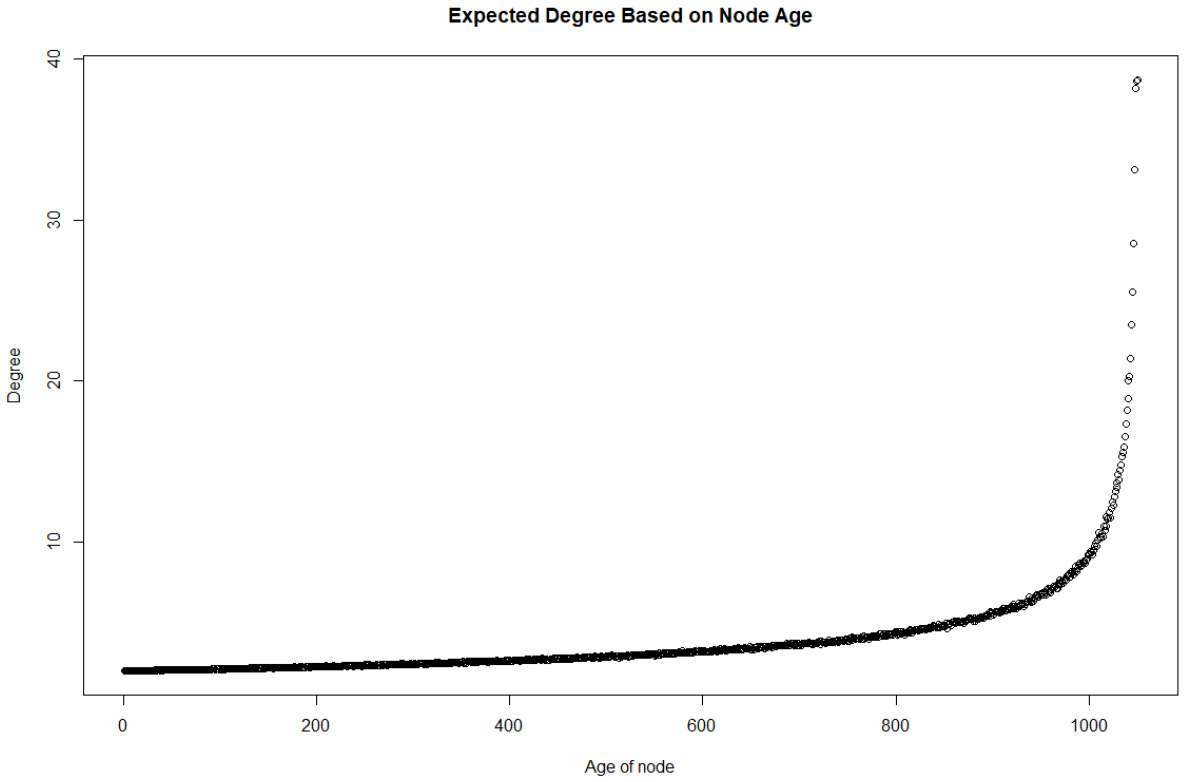


Figure 30: Expected degree at a given time step for $n = 1050$, $m = 2$

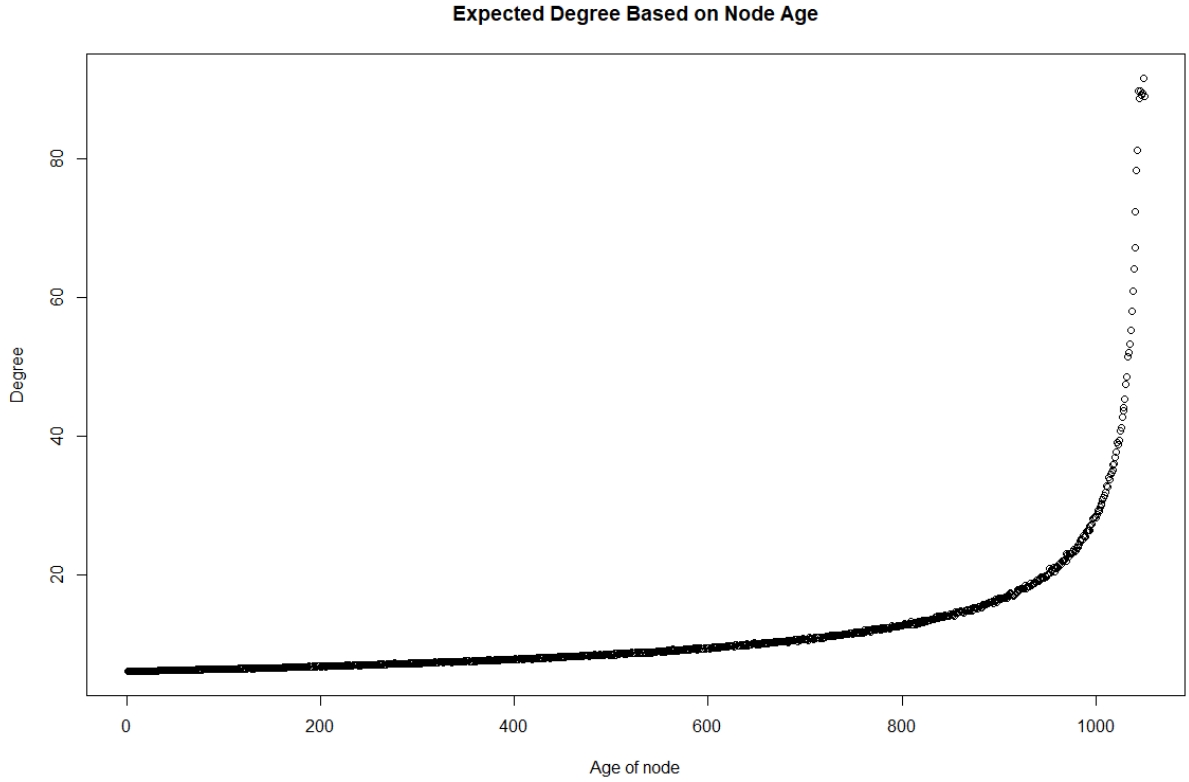


Figure 31: Expected degree at a given time step for $n = 1050$, $m = 6$

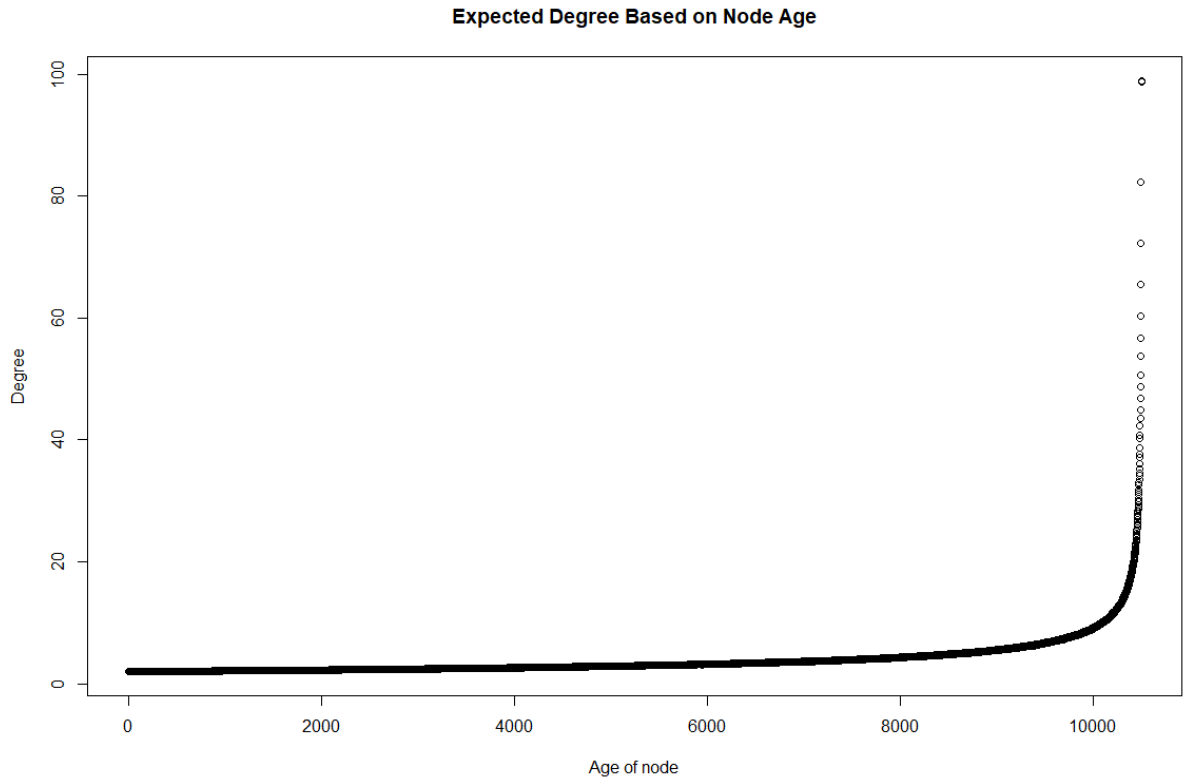


Figure 32: Expected degree at a given time step for $n = 10500$, $m = 2$

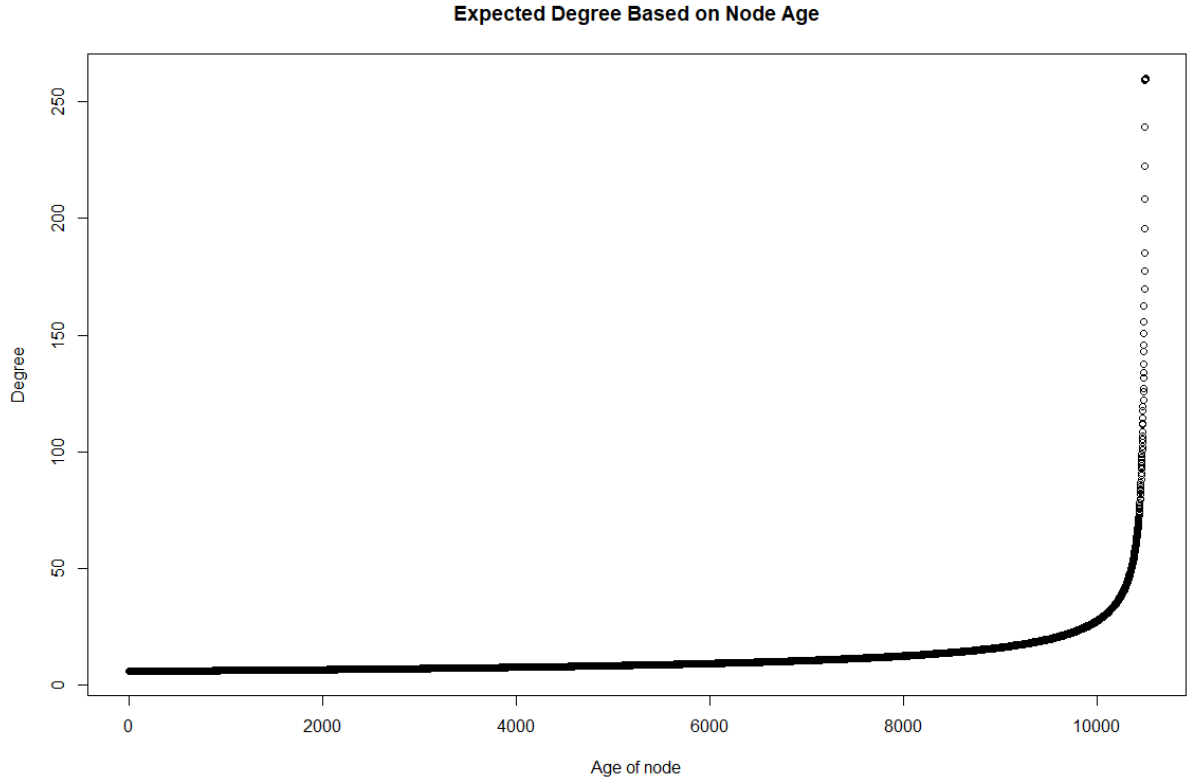


Figure 33: Expected degree at a given time step for $n = 10050$, $m = 6$

We can see from Figures 30 to 33, that as $m$ increase, the degree of a node at any given time step also increases. The overall shape of the graph remains the same regardless of $m$.

**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.**

In order to create a network with the same degrees as another but with different edges, we can use the *sample_degseq* function which will take the degree sequence and generate a network. There are multiple methods that this function can use, but we willl be testing the "simple.no.multiple" and "vl" methods. The first method tries to avoid creating loops and multiple edges while the second method implements an algorithm by Viger and Latapy. This algorithm first generates a simple undirected graph from the degree sequence then performs operations to ensure the graph is connected.

We already know how to plot a network, but in order to mark the communities, we will use the *cluster_fast_greedy* function to figure out the communities.

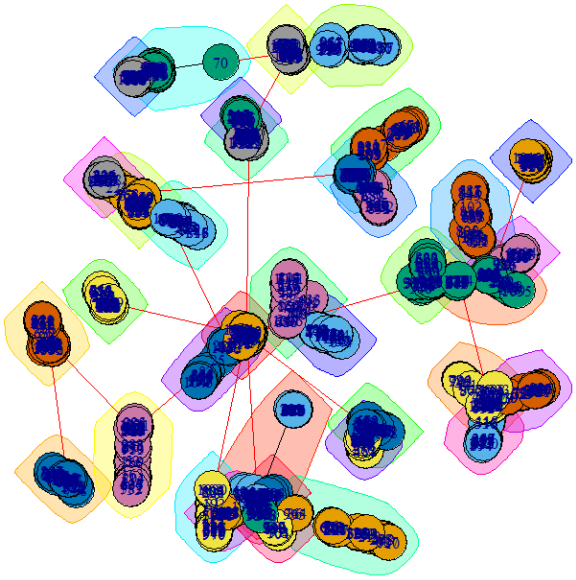

Figure 34: Initial network with $n = 1050$ and $m = 1$

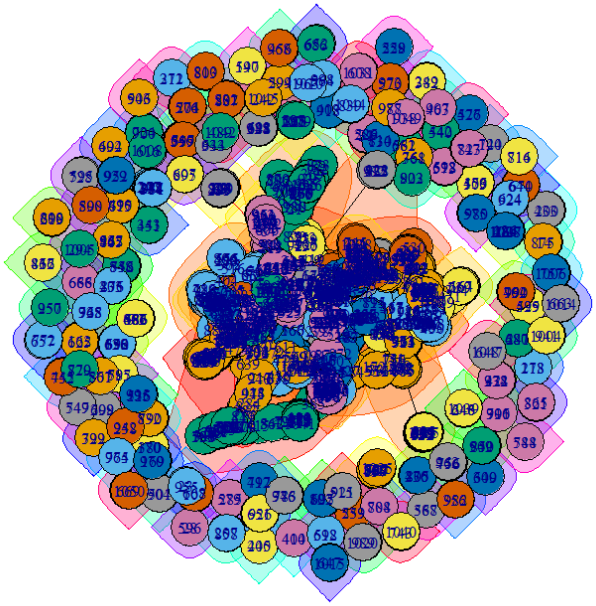

Figure 35: Created network using "simple.no.multiple" method

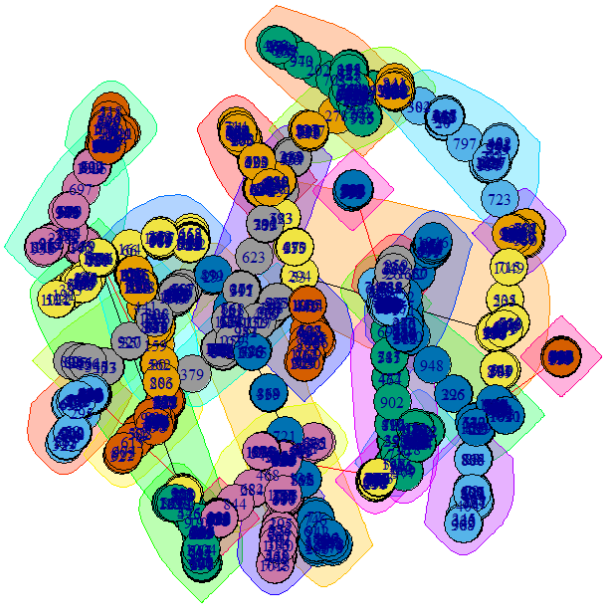

Figure 36: Created network using “vl” method

Figures 34 to 36 show the networks generated by each method. The initial network in Figure 34 shows several communities that are weakly connected to each other to form a fully connected network. Figure 35 shows an interesting structure where there is a central cluster of nodes that is weakly connected to an outer ring of nodes. This was generated using the “simple.no.multiple” method. Figure 36 shows a network somewhat similar to Figure 34 and was generated using the “vl” method.

| **Method** | **Modularity** |
|---|---|
| Initial | 0.9343494 |
| simple.no.multiple | 0.8477614 |
| vl | 0.9359606 |

This table shows that the “vl” method produces a modularity very similar to the initial network while the “simple.no.multiple” method produces a lower modularity. We can clearly see that the network produced in Figure 35 contains many more communities than the other two networks, and these communities do not appear to be well connected within themselves except for the large cluster in the very center.

Based on these results, if the goal is to create a well connected network with fewer communities, then the original preferential attachment method and the “vl” stub matching methods appear to be better over the “simple.no.multiple” stub matching method.

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

**a 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?**

We are given the information that "the probability that a newly added vertex connects to an old vertex is proportional to" the equation:

$$P[i] \ (ck_i^{\alpha} + a)(dl_i^{\beta} + b) \tag{5}$$

where $k_i$ is the degree of node $i$, and $l_i$ is the age of the node $i$. We can use the function *sample_pa_age* and plug in the provided values which will produce an undirected graph whose edges are formed based on the aforementioned probability. If we plug in these values, we get the equation:

$$P[i] \ (k_i + 1)(l_i^{-1}) \tag{6}$$

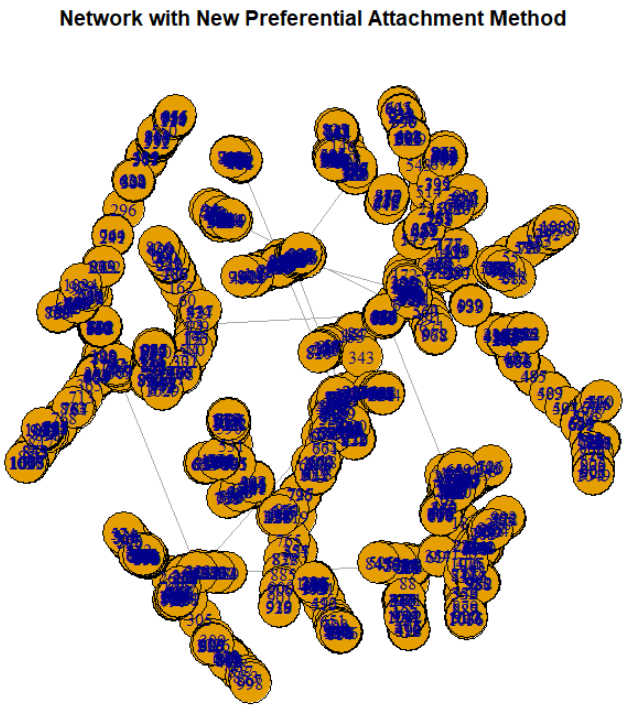


Figure 37: Plot of the network created with the age-dependent probability

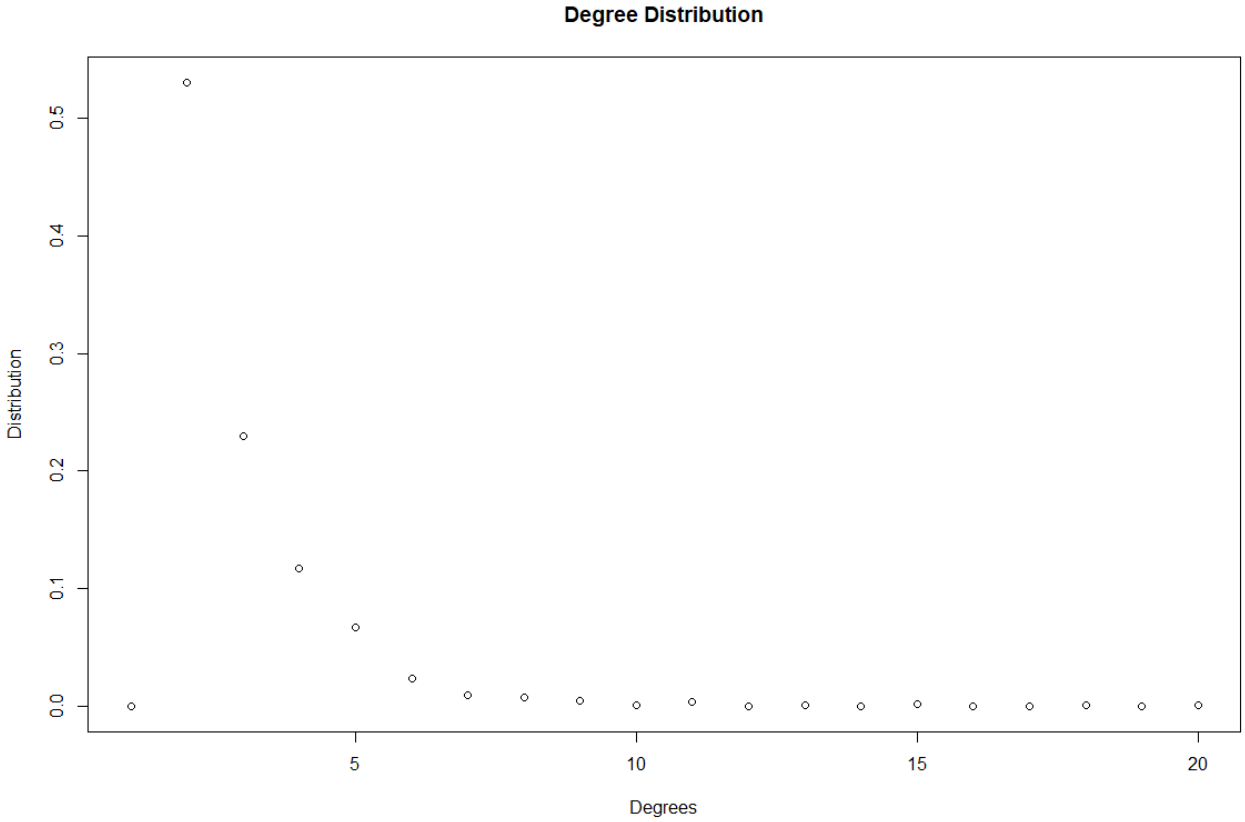


Figure 38: Degree distribution for the age-dependent preferential attachment network

We can see from Figure 37 that the resutling network does not look drastically different from the ones produced in earlier parts with distinct clusters loosely connected to each other. The degree distribution in Figure 38 also follows a binomial distribution.

In order to calculate the Power Law exponent, a.k.a. the slope of the log scale degree distribution, we will need to convert the data into log scale.

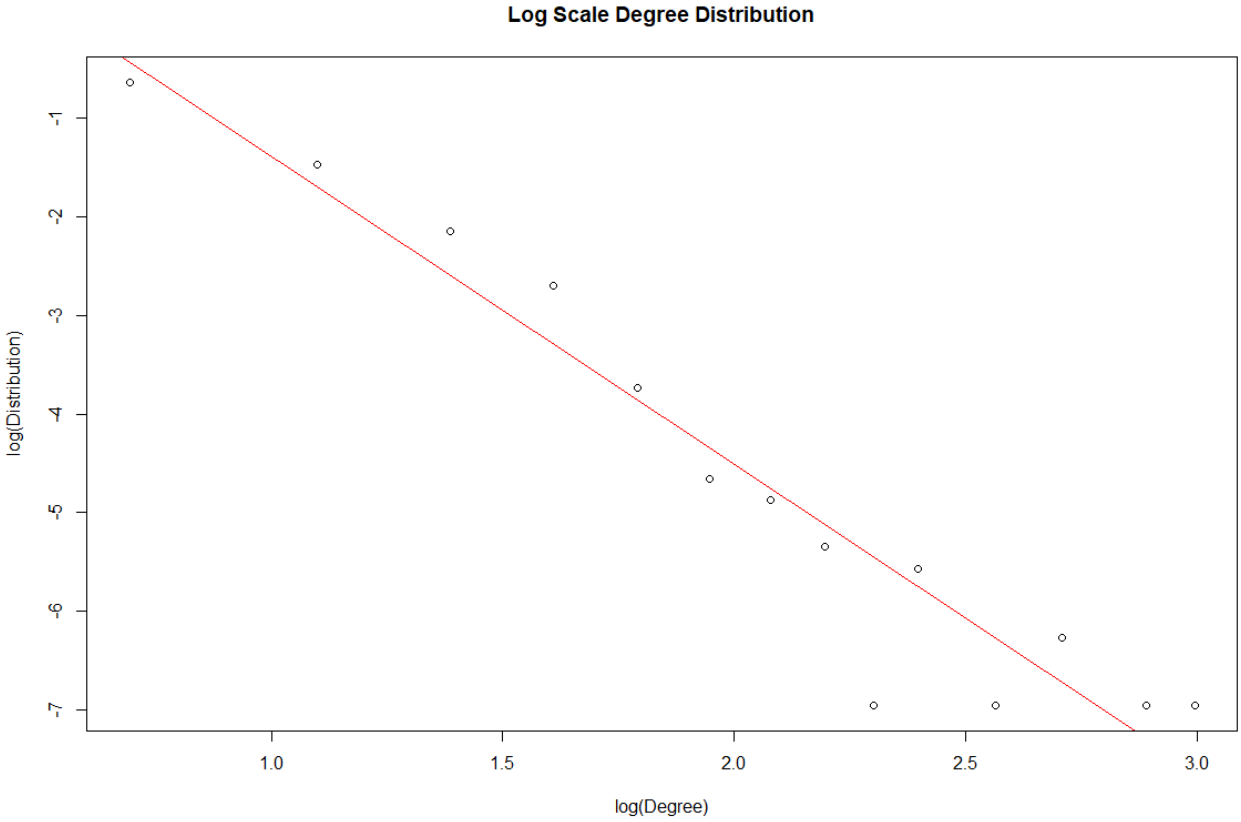


Figure 39: Log scale degree distribution with linear regression

We can then determine that the slope of the graph is -3.120922 which is a much steeper slope than what was produced in any of the earlier models. This indicates that there is a much larger probability for a certain degree or set of degrees than the rest. This is very noticeable when viewing the regular degee distribution plot in Figure 38 which shows a probability of over 0.5 for a degree of about 2 while every degree distribution graph created in 1a stayed below 0.25 for the distribution.

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

We repeat the same procedure as 2b.

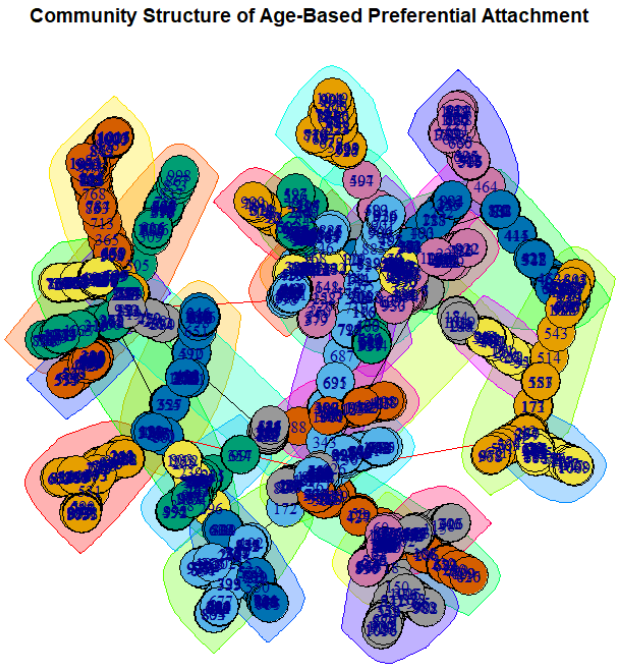


Figure 40: Community structre plot for the age-based preferential attachment model

The modularity is 0.9362246 which is very similar to the modularity calculated in 2b implying that the two networks have a similar level of connectivity within and between clusters.

# 2 Random Walk on Networks

## 2.1 Random walk on Erdos-Renyi networks

**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.**

We will repeat the process from Part 1's 1a except with $p = 0.015$.

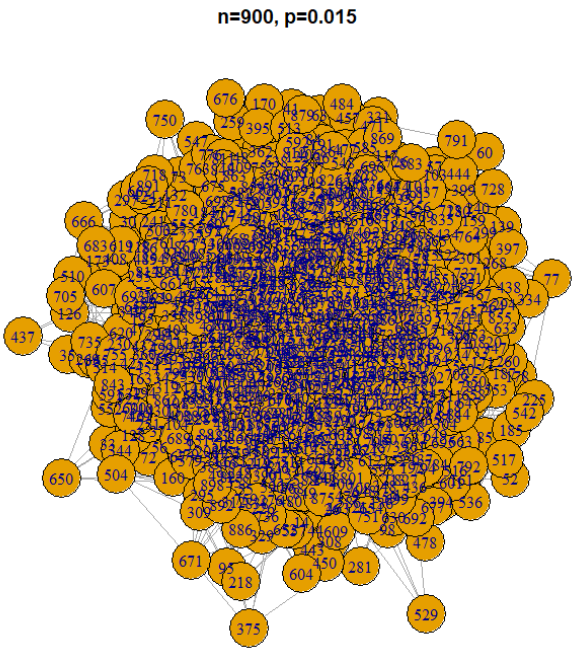


Figure 41: Network with $n = 900$ nodes and $p = 0.015$

**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 $\langle\rangle$ is over random choices of the starting nodes.**

This part is asking to randomly generate a path from a starting node to some end node after taking $t$ steps, where each step represents moving via an edge from one node to another. In order to do this, we start with the network generated in 1a and select a single node using *sample*. We then "walk" from that node to a neighboring node using a specifically created transition matrix to ensure no teleportation. We repeat this $t$ times to generate an entire list of nodes that the walker travelled. In order to find the average distance and the variance, we need to use the function *distances* to calculate the length of the shortest path from the starting node to each end node for each step before using that data to calculate the mean and variance. We had the walker take 100 steps per run and performed 1000 runs.

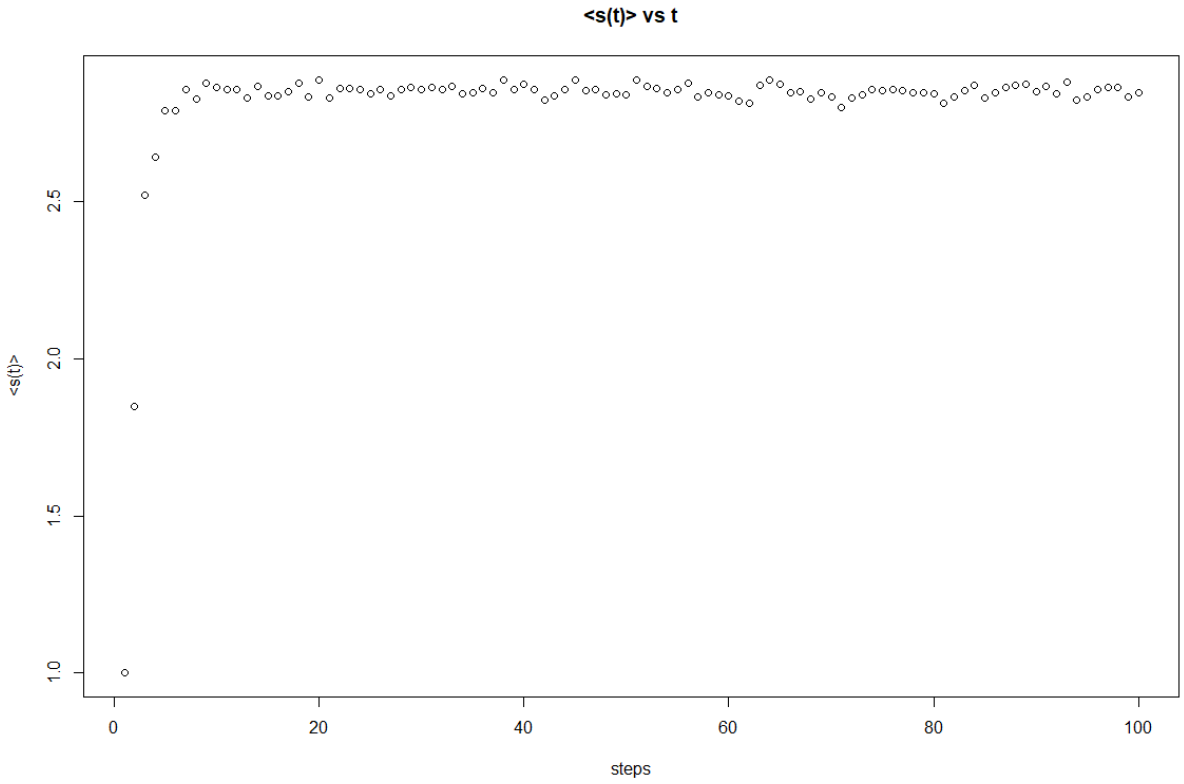


Figure 42: Average length of the shortest path from the start node to the end node at $t$ steps

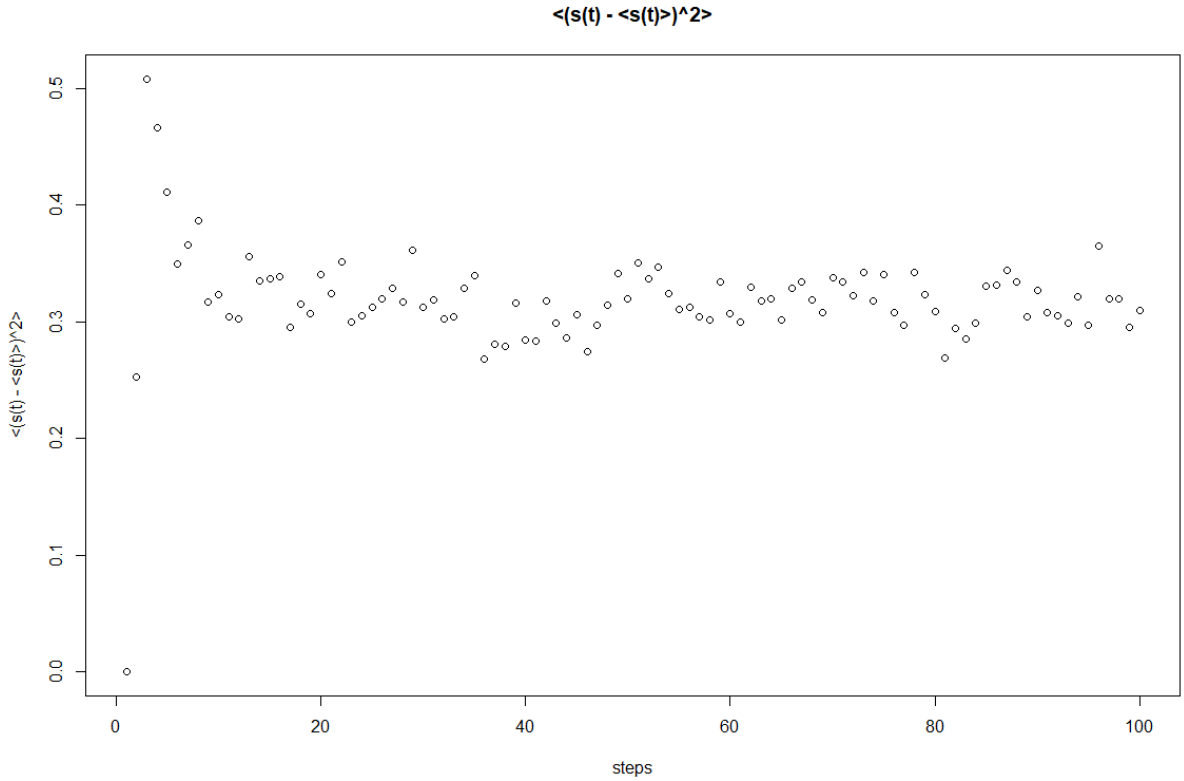


Figure 43: Variance of the shortest path from the start node to the end node at $t$ steps

Figure 42 shows that the average shortest path between the start node and the end node starts out very small before ramping up very quickly and then plateauing just as quickly at around 2.9. This means that the graph is connected well enough that after 100 steps, on average, only 3 nodes need to be transversed to travel between the start and end nodes.

Figure 43 shows the variance of each of the paths just discussed. We can see that the variance also starts out low before rapidly increasing. However, instead of plateauing at the top of the peak, it drops down slightly before leveling off. This means that after the first step, the distance between the start and end nodes can vary greatly from the average for that step before the walker reaches a point where they begin to primarily travel along nodes that are all a similar distance from the start node.

**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?**

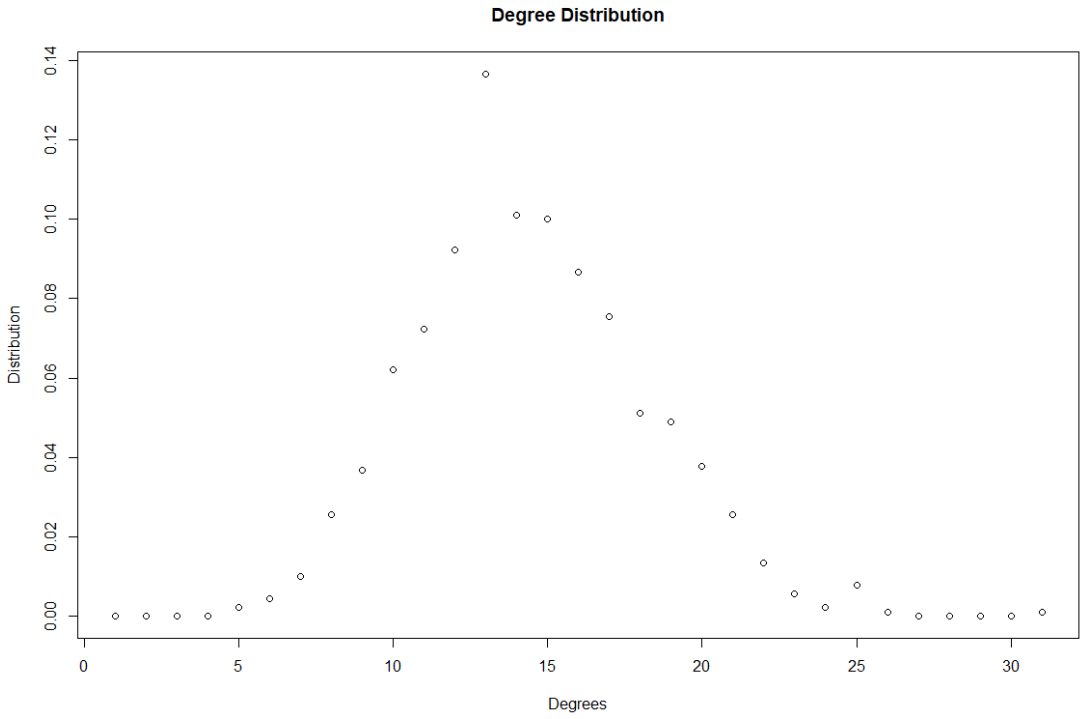


Figure 44: Degree distribution of the network

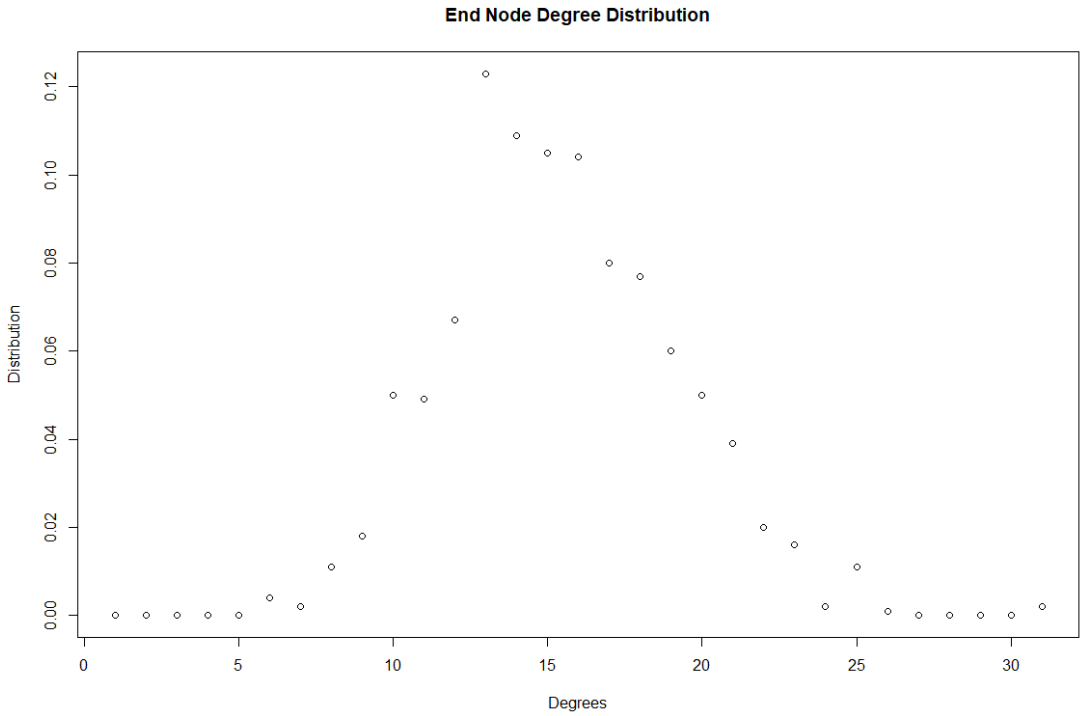


Figure 45: Degree distribution of the end nodes after 100 steps and 1000 runs

As Figures 44 and 45 show, the two degree distributions are strikingly similar implying some kind of relationship between the two. The primary difference is that the normal degree distribution has a slightly higher number of nodes of degree around 12 or 13 as the peak almost reaches 0.14 vs 0.12 for the end node distribution.

**d 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?**

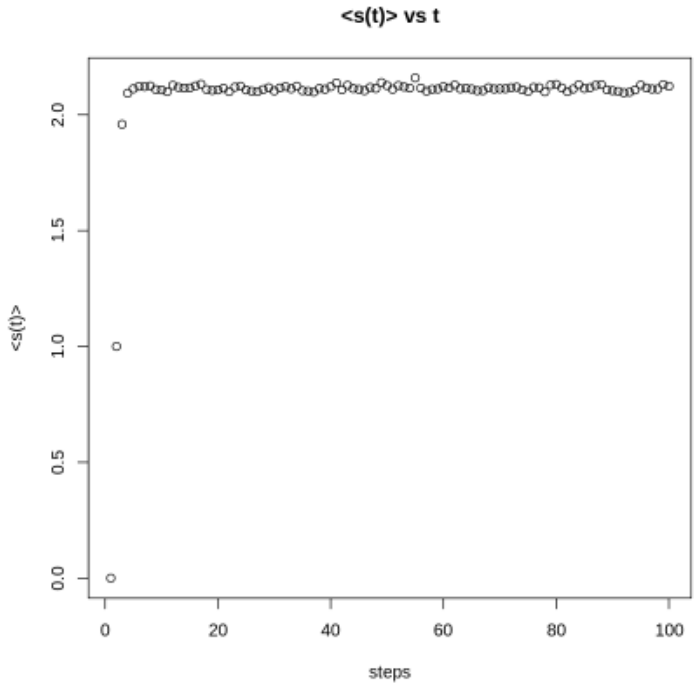


Figure 46: Degree distribution of the network for $n = 9000$

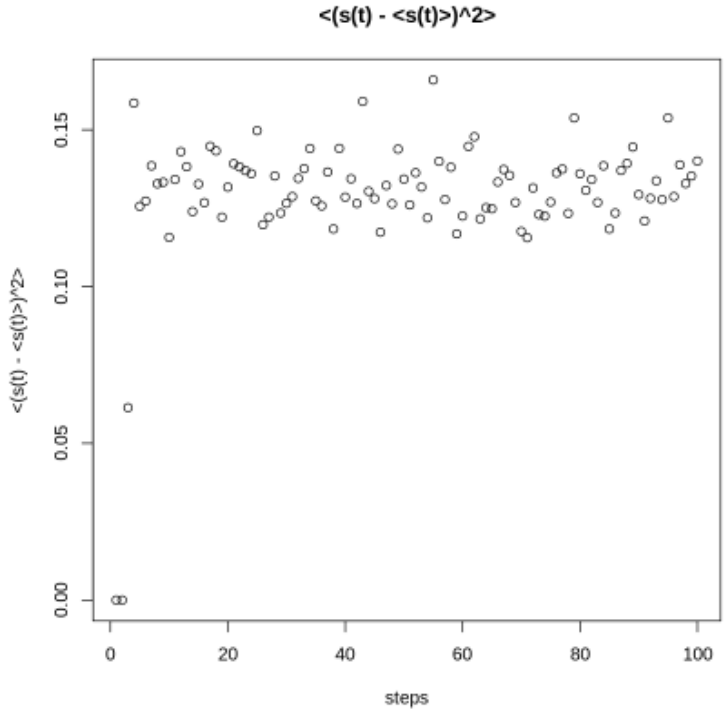


Figure 47: Degree distribution of the end nodes for $n = 9000$ after 100 steps and 1000 runs

The average distance when $n = 9000$ is slighly lower than the lower node count, around 2.1, meaning that the shortest path between the start node and the end node at step $t$ is only about 2 nodes. The variance is also significantly smaller. It stabilized around 0.14, which is about half the variance for $n = 900$.

We then use the *diameter* function to calculate the diameter for both graphs.

| **n** | **Diameter** |
|---|---|
| 900 | 5 |
| 9000 | 3 |

The graph with 900 nodes has a larger diameter than the graph with 9000 nodes. The smaller diameter helps explain the reduction in the average shortest path and the variance as it implies that the larger network is more connected than the smaller one, allowing for shorter paths to form between any two given nodes.

## 2.2 Random walk on networks with fat-tailed degree distribution

**a Generate an undirected preferential attachment network with 900 nodes, where each new node attaches to $m = 1$ old nodes.**

Repeat the same process as 2a in Part 1.

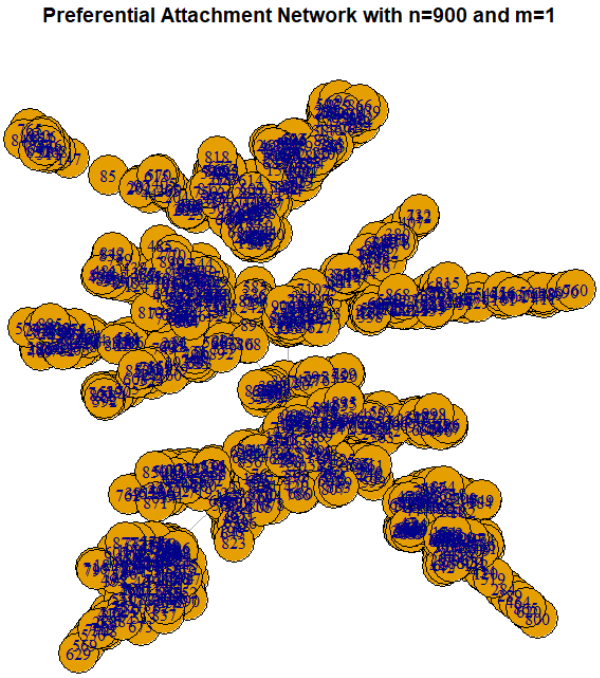


Figure 48: Network generated with preferential attachment, $n = 900$ and $m = 1$

**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. $t$**

Repeat the same process as 1b.

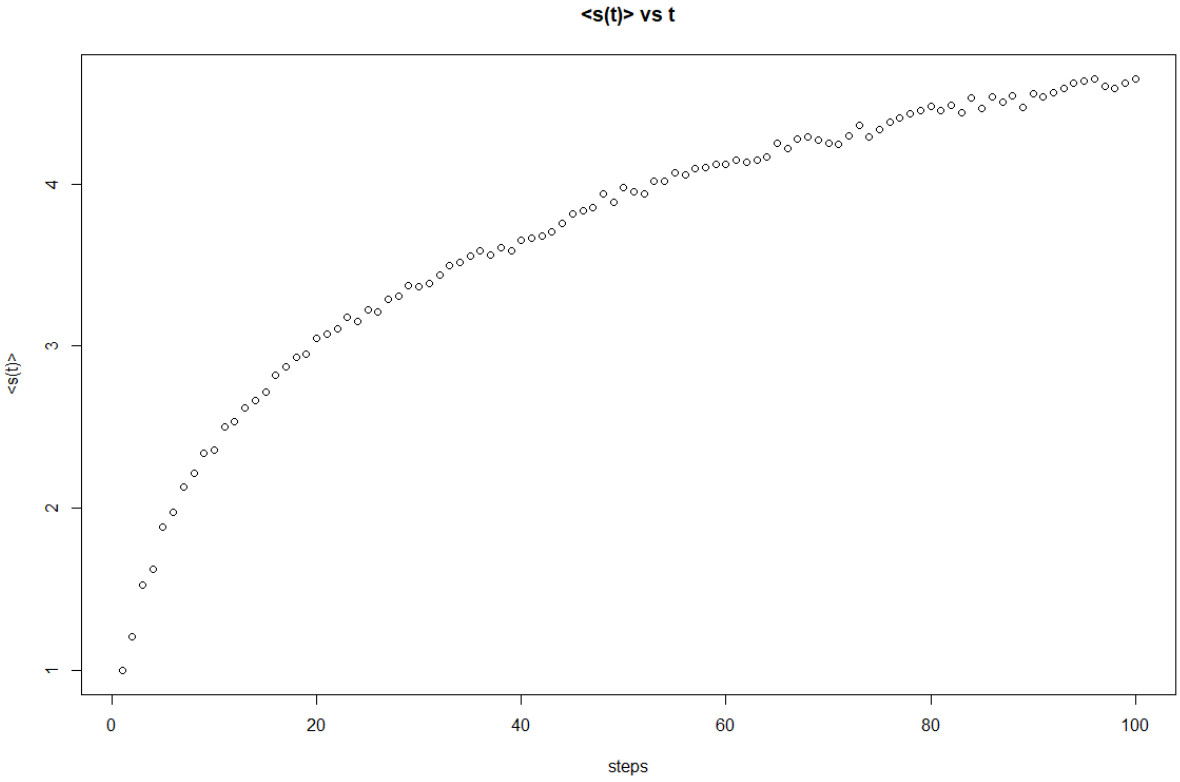


Figure 49: Average length of the shortest path from the start node to the end node at $t$ steps

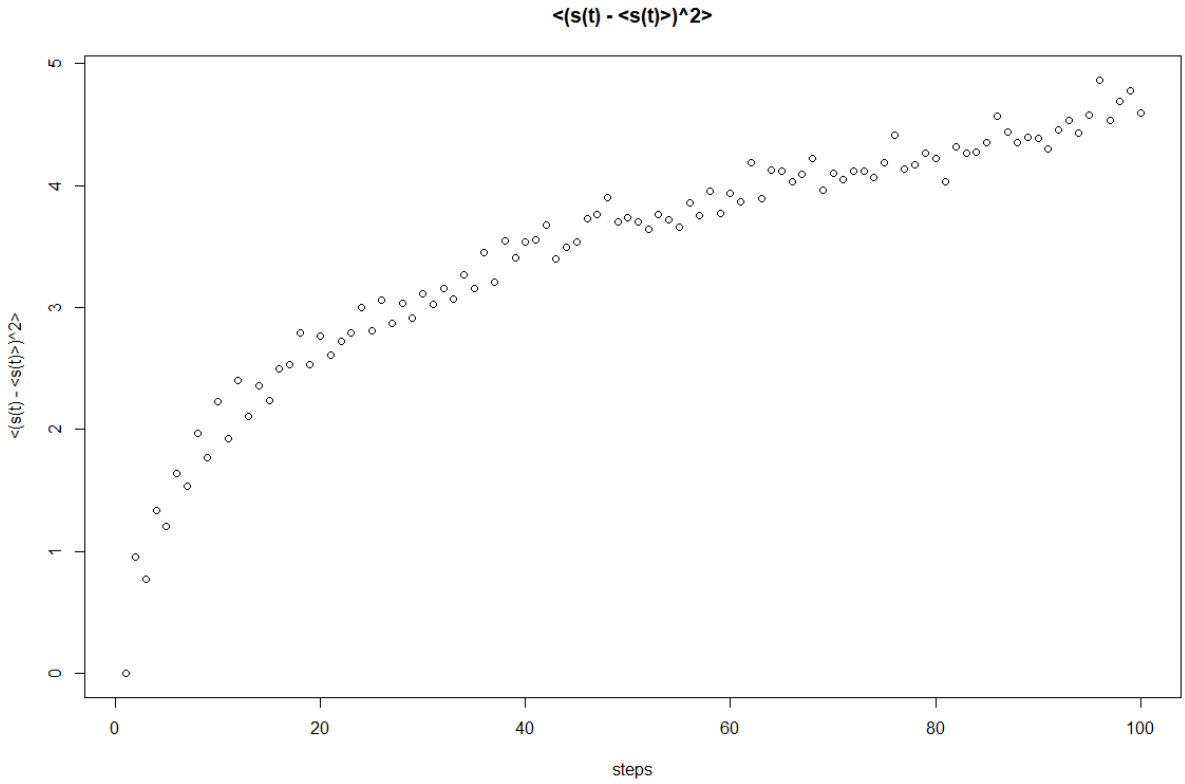


Figure 50: Variance of the shortest path from the start node to the end node at $t$ steps

We can see that unlike the previous graphs in Figures 42, 43, 46, and 47, the graphs in Figures 49 and 50 do not plateau. Instead they both have a fairly steep slope until about 10 steps in before the slope begins to level off and remain constant. This means that the average shortest distance gradually increases as the walker takes more steps which also results in the variance increasing as the steps increase.

**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?**

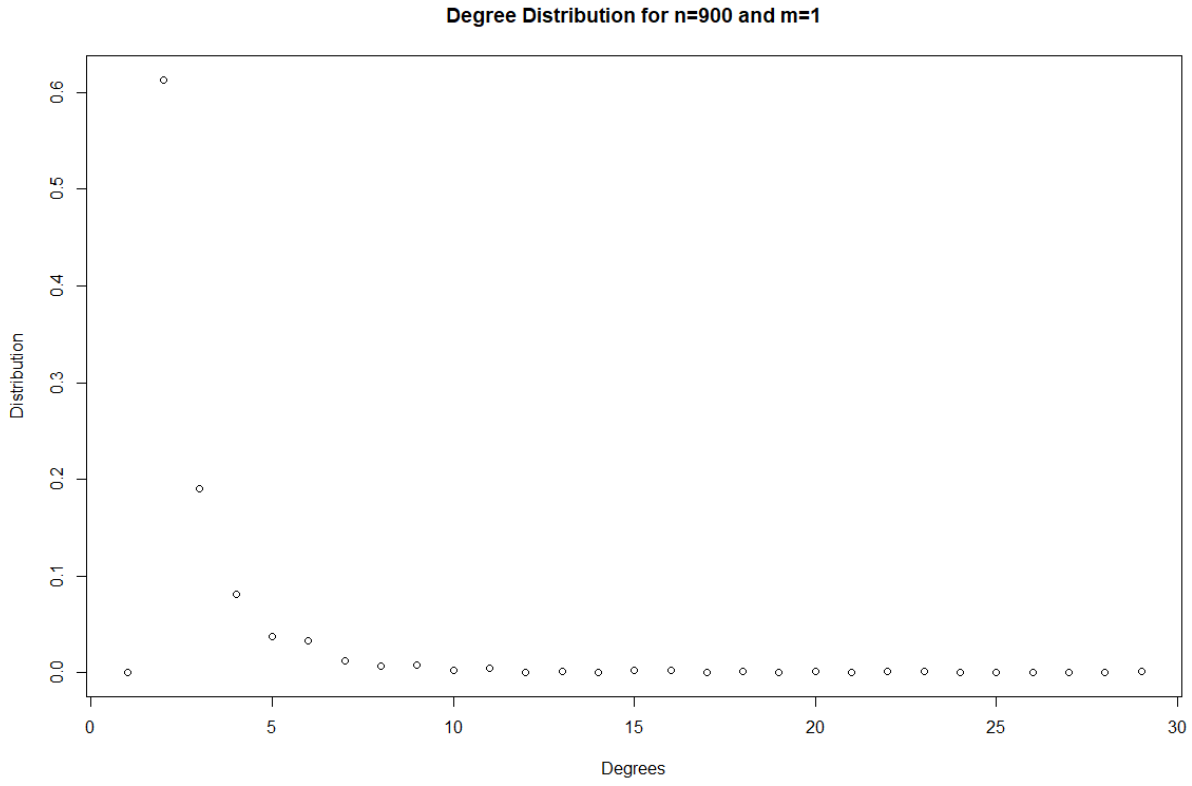


Figure 51: Degree distribution of the network for $n = 900$ and $m = 1$

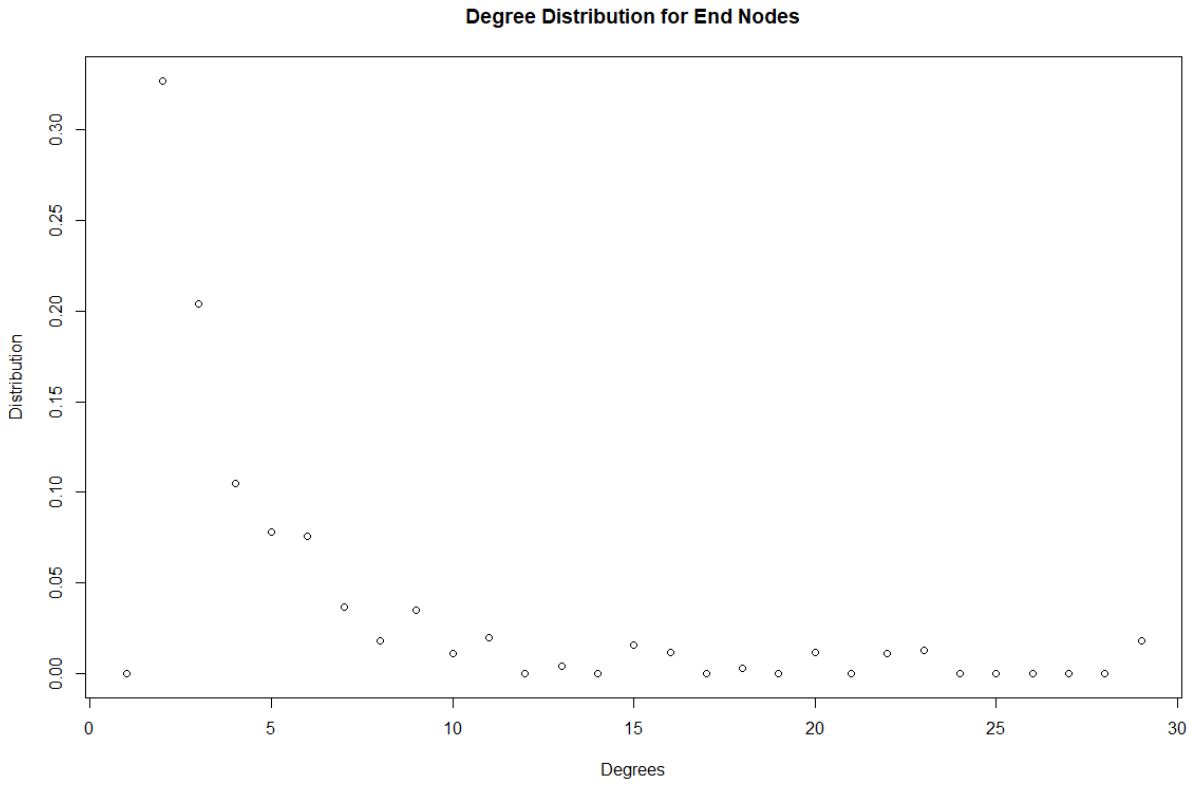


Figure 52: Degree distribution of the end nodes for $n = 900$ and $m = 1$

We can see that the two distributions have a similar shape just like in 1c. The end node degree distribution has a lower peak but the rest of the graph matches up.

**d Repeat 2(b) for preferential attachment networks with 90 and 9000 nodes, and $m = 1$. Compare the results and explain qualitatively. Does the diameter of the network play a role?**

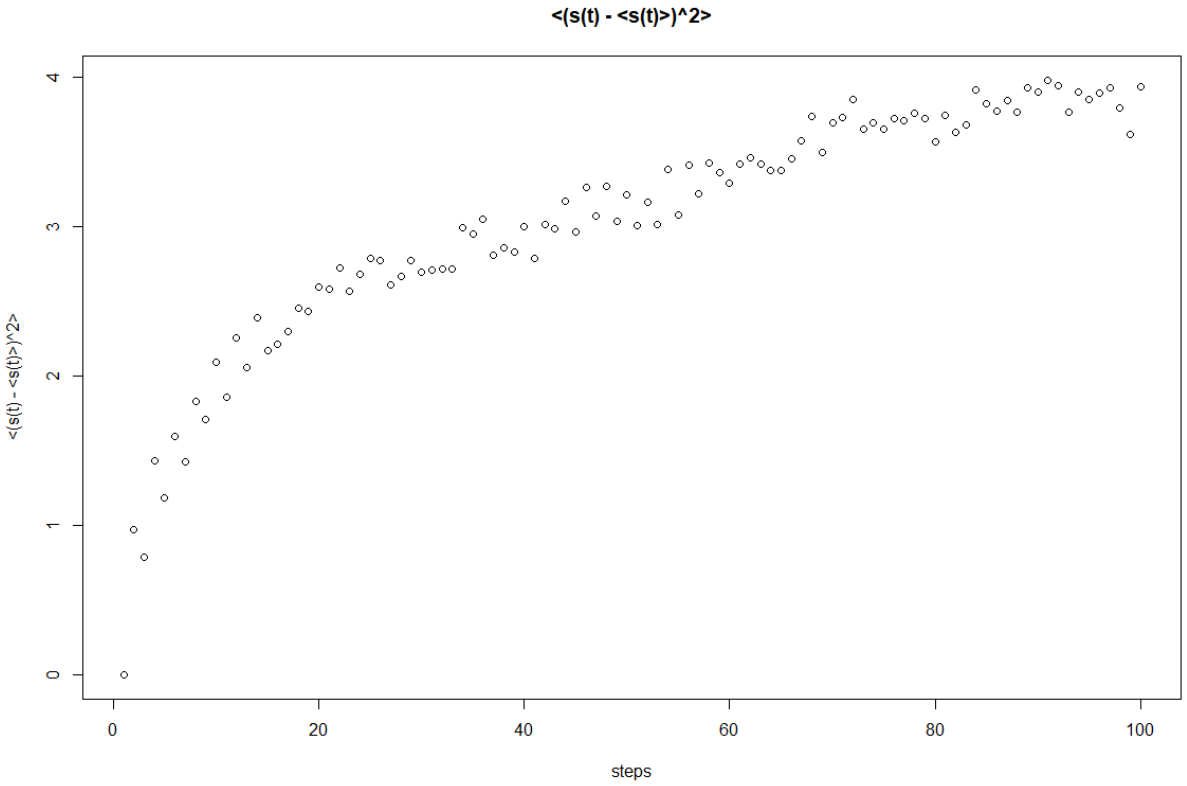


Figure 53: Variance of the shortest path from the start node to the end node at $t$ steps for $n = 90$

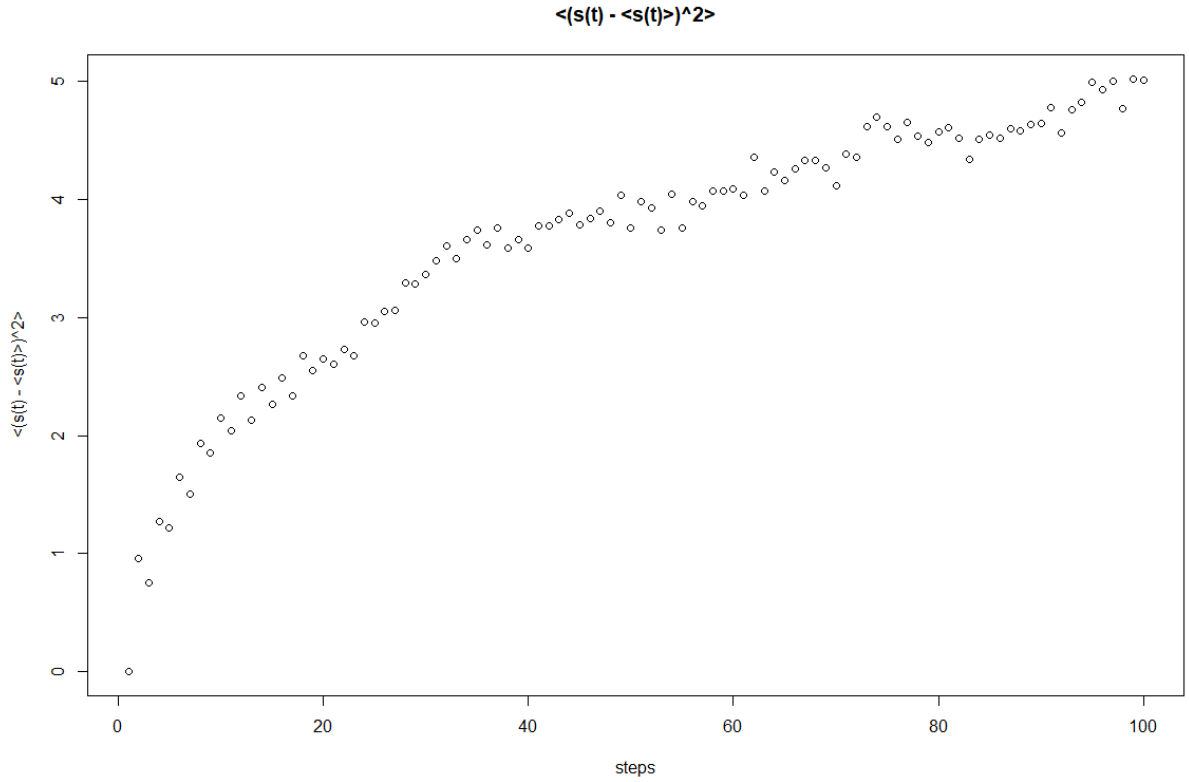


Figure 54: Variance of the shortest path from the start node to the end node at $t$ steps for $n = 9000$

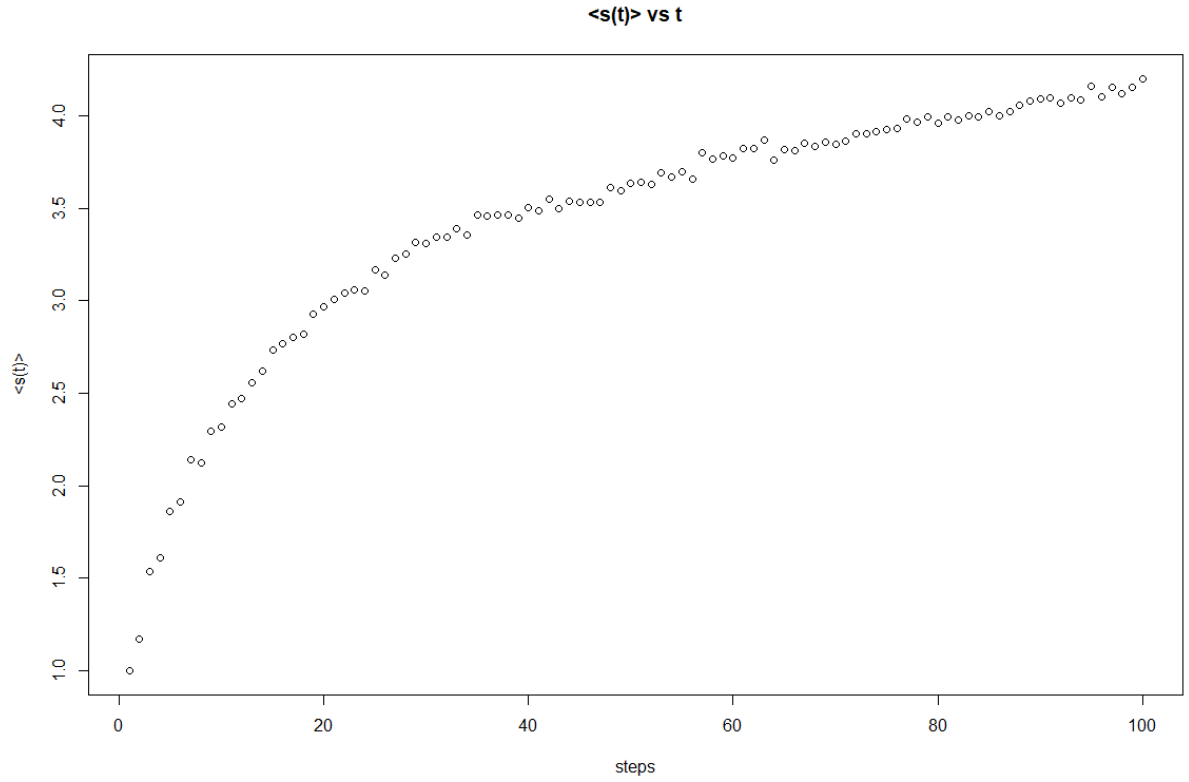


Figure 55: Average length of the shortest path from the start node to the end node at $t$ steps for $n = 90$

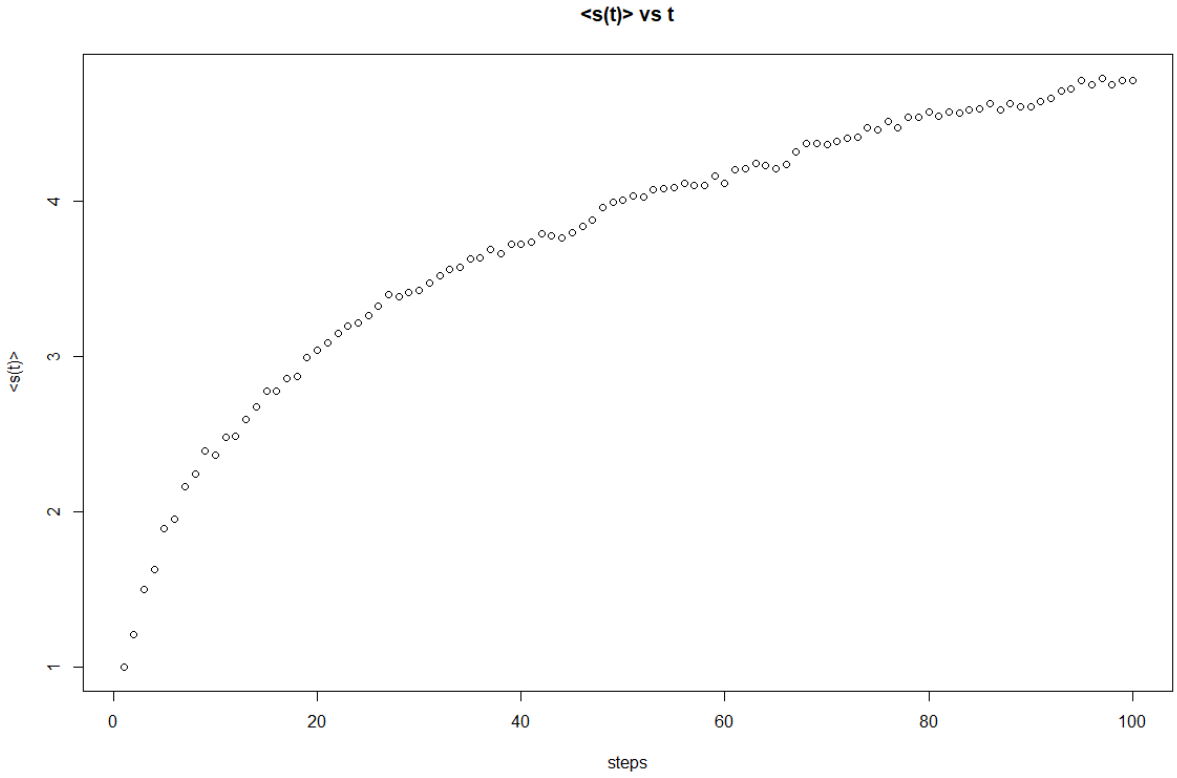


Figure 56: Average length of the shortest path from the start node to the end node at $t$ steps for $n = 9000$

| n | Diameter |
|---|---|
| 90 | 11 |
| 9000 | 24 |

If we compare the graphs in Figures 53 through 56 to the graphs in Figures 49 and 50, we can see that as the number of nodes increases, so does the average shortest path and the variance. Viewing the diameter, we can see that the larger network also has a larger diameter. The larger diameter and longer shortest paths relate to each other as a large diameter implies that the network is not as well connected as another graph of the same size with a smaller diameter.

## 2.3 PageRank

**a 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 start by creating two networks with $n = 900$ nodes and $m = 4$ number of edges created at each step. Then we use the function *as_edgelist* to get all of the edges from one of the networks before using *sample* to randomize the order of the edges. Then we can use *permutate* to mix the order of the nodes to create a brand new edge list. This new list is then added to the other graph to create a new, better connected one.

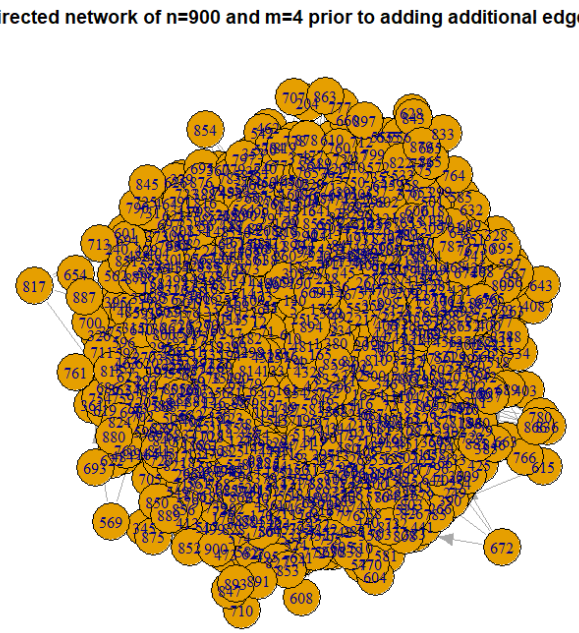


Figure 57: Original network prior to adding new edges.

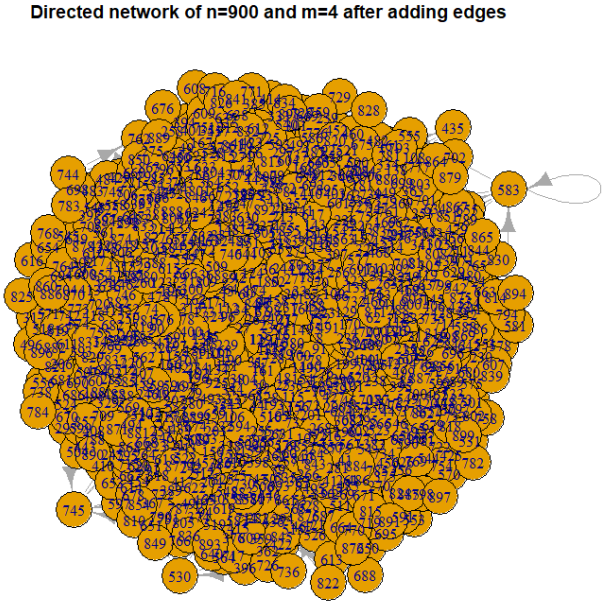


Figure 58: Network after adding new edges.

The sheer number of nodes makes it a little difficult to see, but the second graph in Figure 58 contains twice as many edges as the graph in Figure 57.

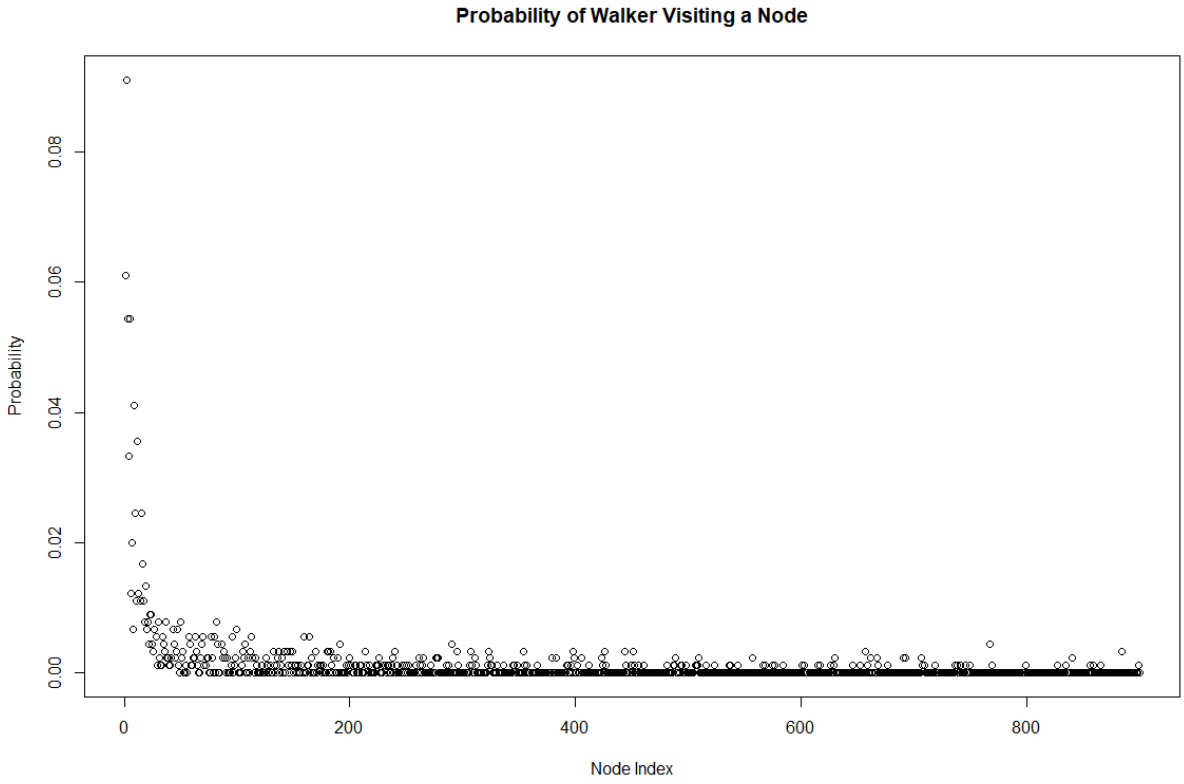


Figure 59: Probability of the walker reaching each node.

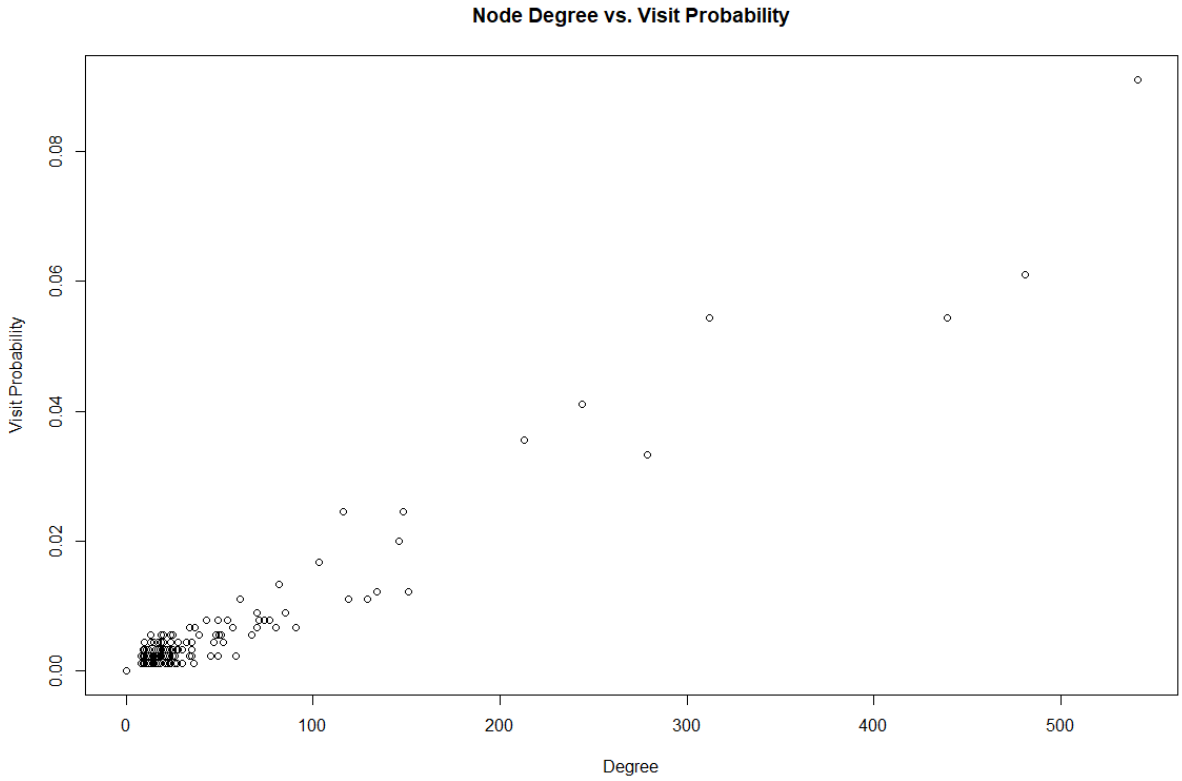


Figure 60: Probability of the walker reaching a node of a certain degree.

The lower index nodes have a higher probability of being reached as they have more incoming edges than outgoing edges. The only outgoing edges would be the ones added by the second edge list The graph in Figure 60 supports this as the walker has a higher chance of ending up in a node with a high degree which would likely be the older nodes.

**b In all previous questions, we didn't have any teleportation. Now, we use a teleportation probability of $\alpha = 0.2$ (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 $\alpha$?**

In order to determine when the walker should teleport, we can use any number of random number generating functions and set the parameters in a convenient way. In this case, we used *runif* to generate a value between 0 and 1 and set the walker to teleport if the generated value is below 0.2.

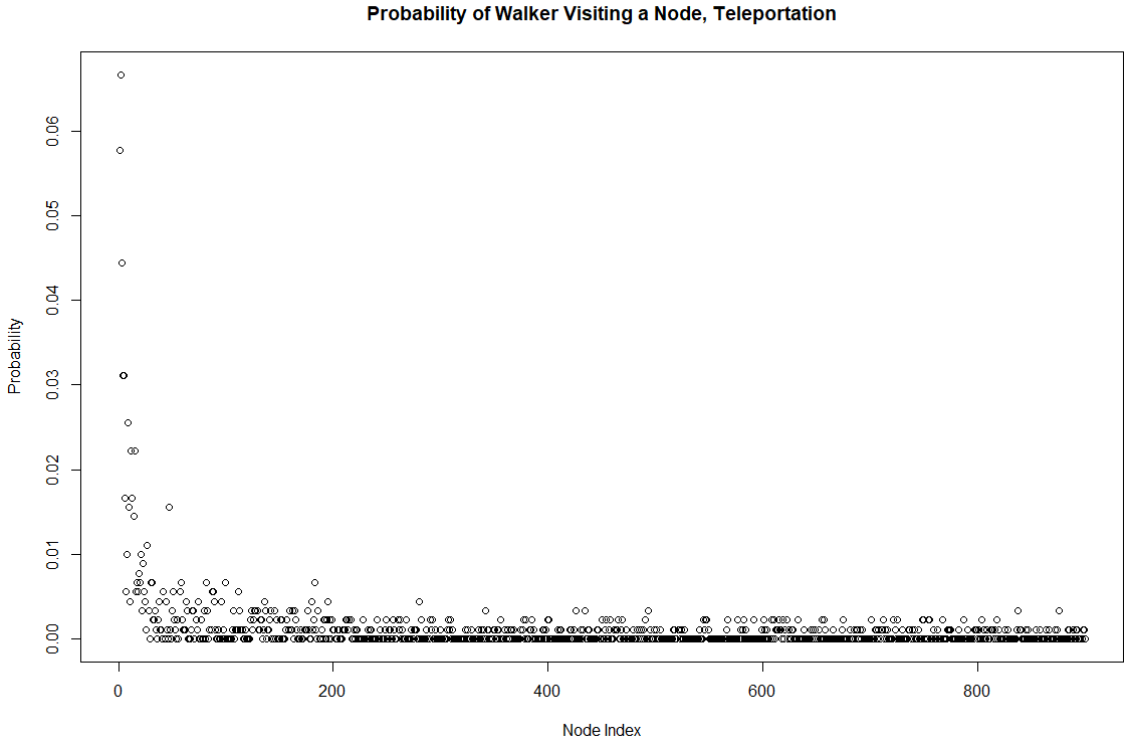


Figure 61: Probability of the walker reaching each node with teleporation as an option.

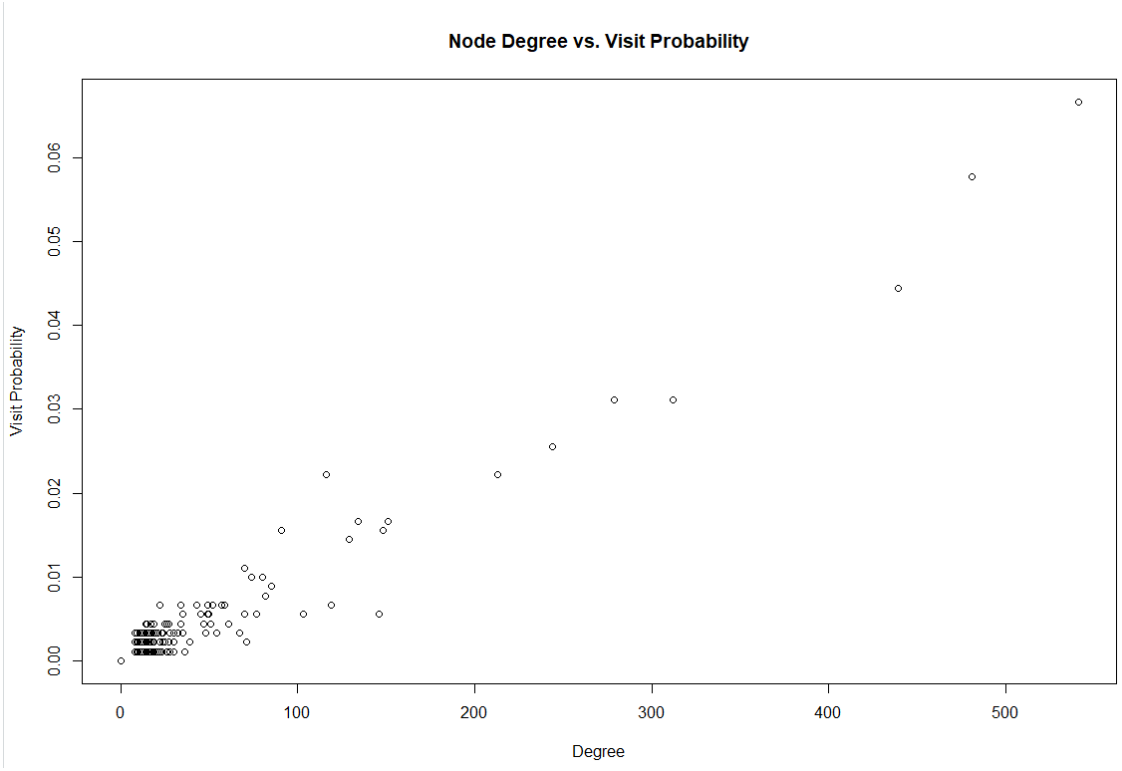


Figure 62: Probability of the walker reaching a node of a certain degree.

We can see that teleporation enables the walker to travel to newer nodes and nodes with lower degrees more often. The teleporation helps deal with the issue of older nodes having more incoming edges than newer nodes by enabling the walker to ignore edges and directions and simply move to a random node.

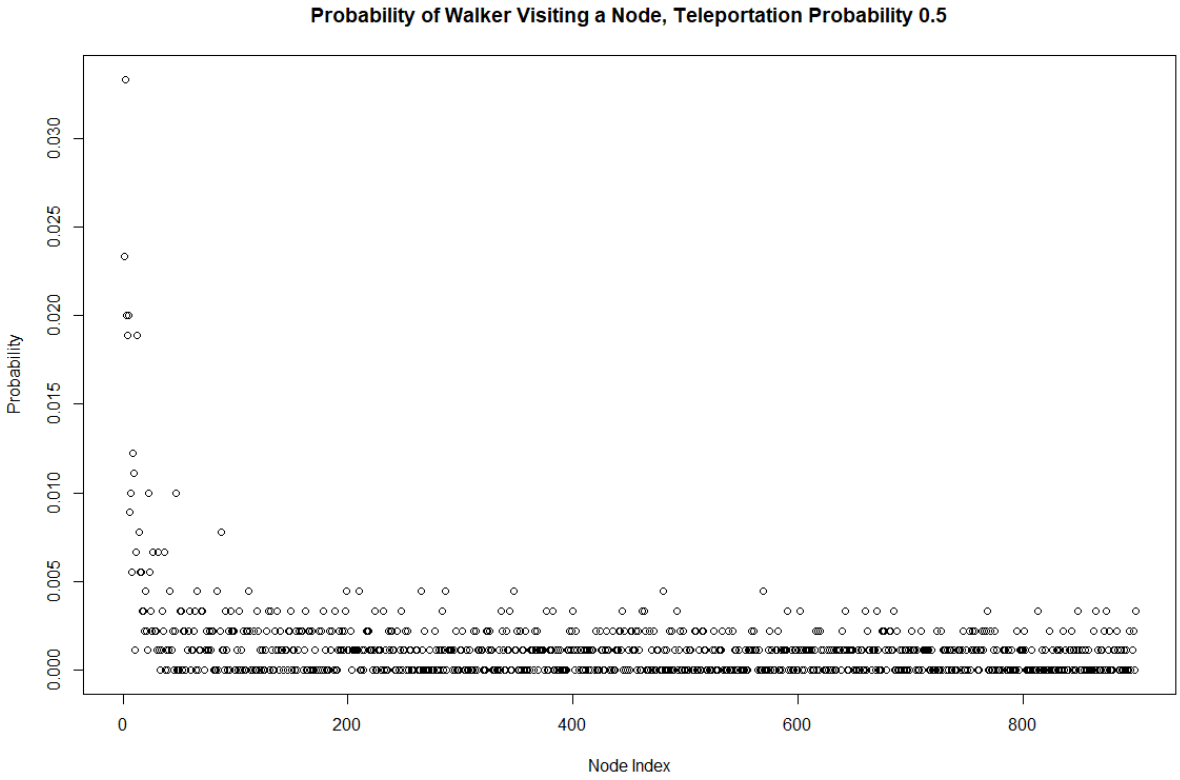


Figure 63: Probability of the walker reaching each node with $\alpha = 0.5$.

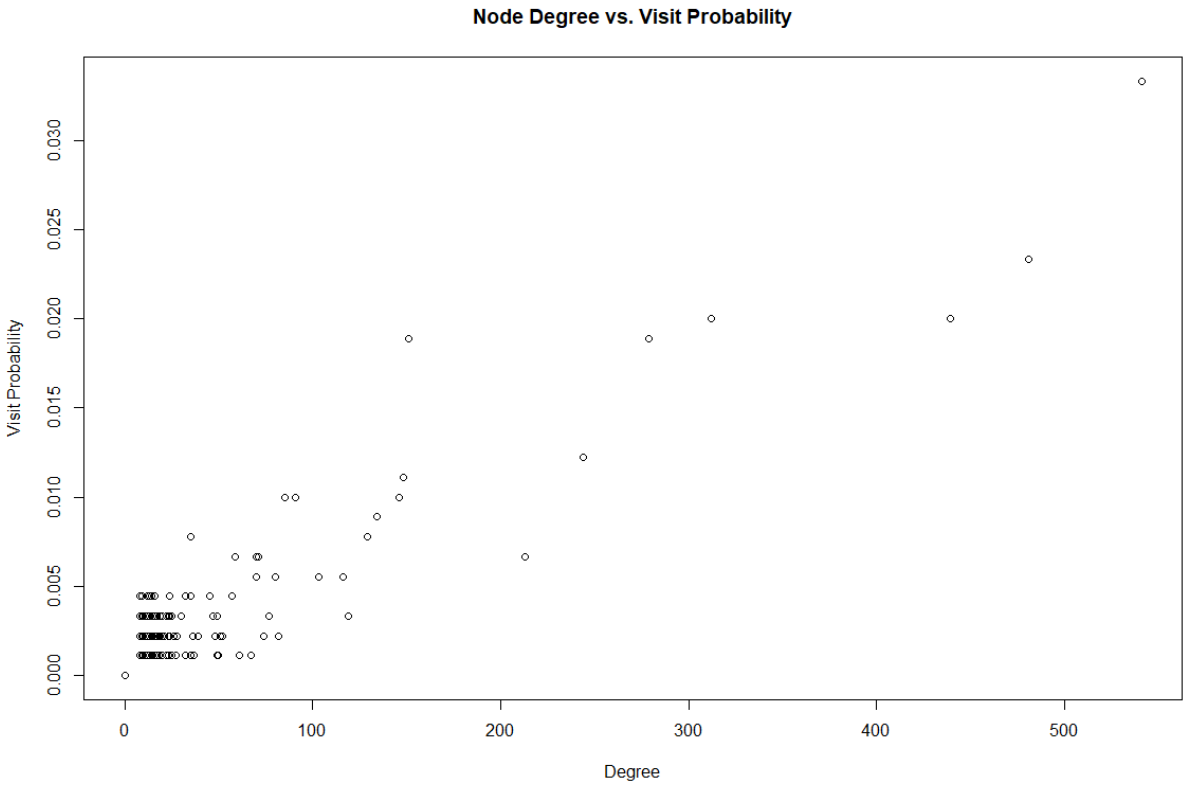


Figure 64: Probability of the walker reaching a node of a certain degree with $\alpha = 0.5$.

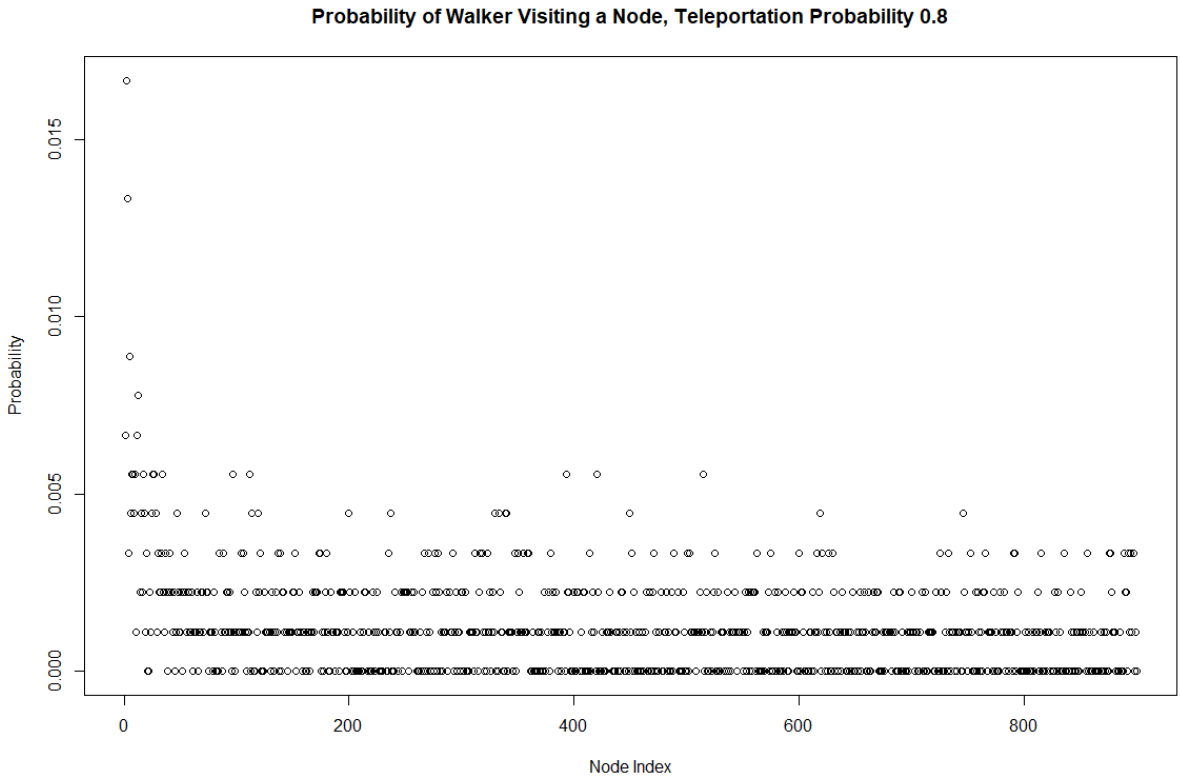


Figure 65: Probability of the walker reaching each node with $\alpha = 0.8$.

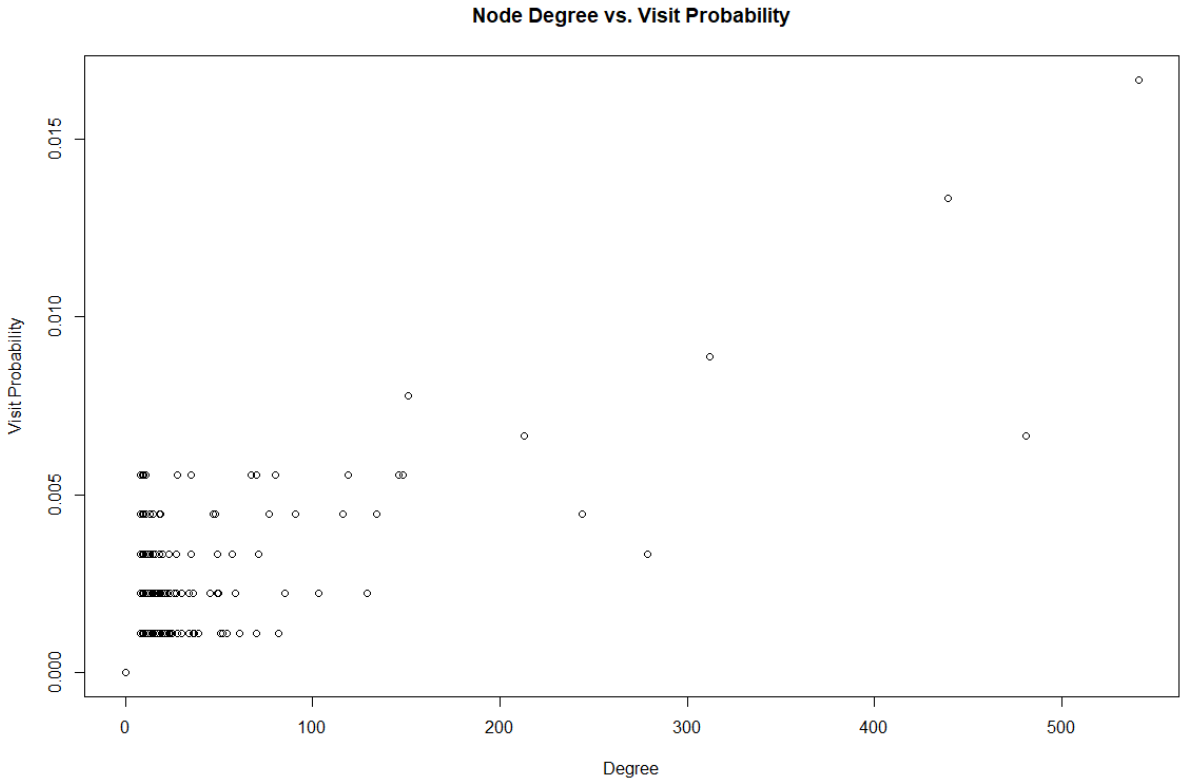


Figure 66: Probability of the walker reaching a node of a certain degree with $\alpha = 0.8$.

Figures 63 through 66 show that as $\alpha$ increases, the probability becomes much more uniform across all nodes. The higher teleporation chance means that each node will end up with more or less the same chance of the walker reaching it at any given step. Also, the visit probability begins to resemble the networks actual degree distribution for much the same reasons.

## 2.4 Personalized PageRank

**a 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 $\alpha = 0.2$. Compare the results with 3(a).**

Instead of using the transition matrix to determine the probability of a walker traveling on a particular edge, we can use the *page_rank* function to create a new probability matrix.

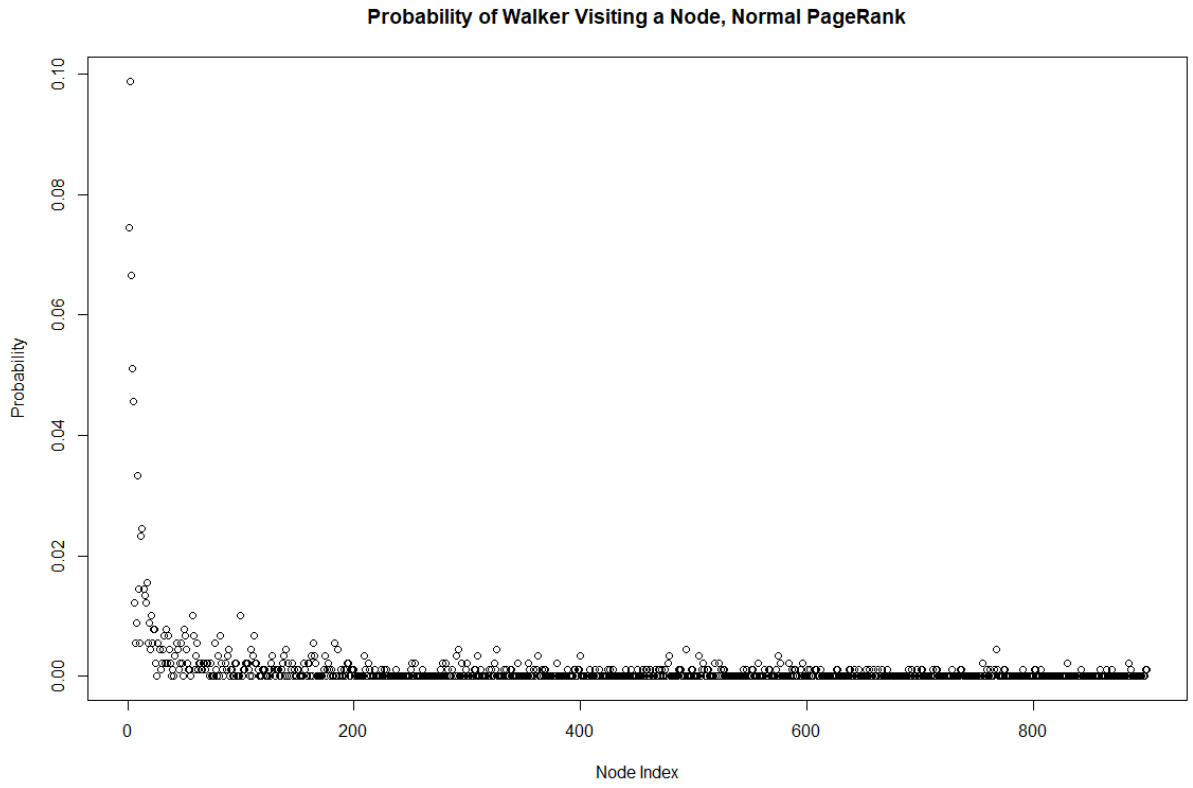


Figure 67: Probability of the walker reaching a node with Normal PageRank values for teleportation.

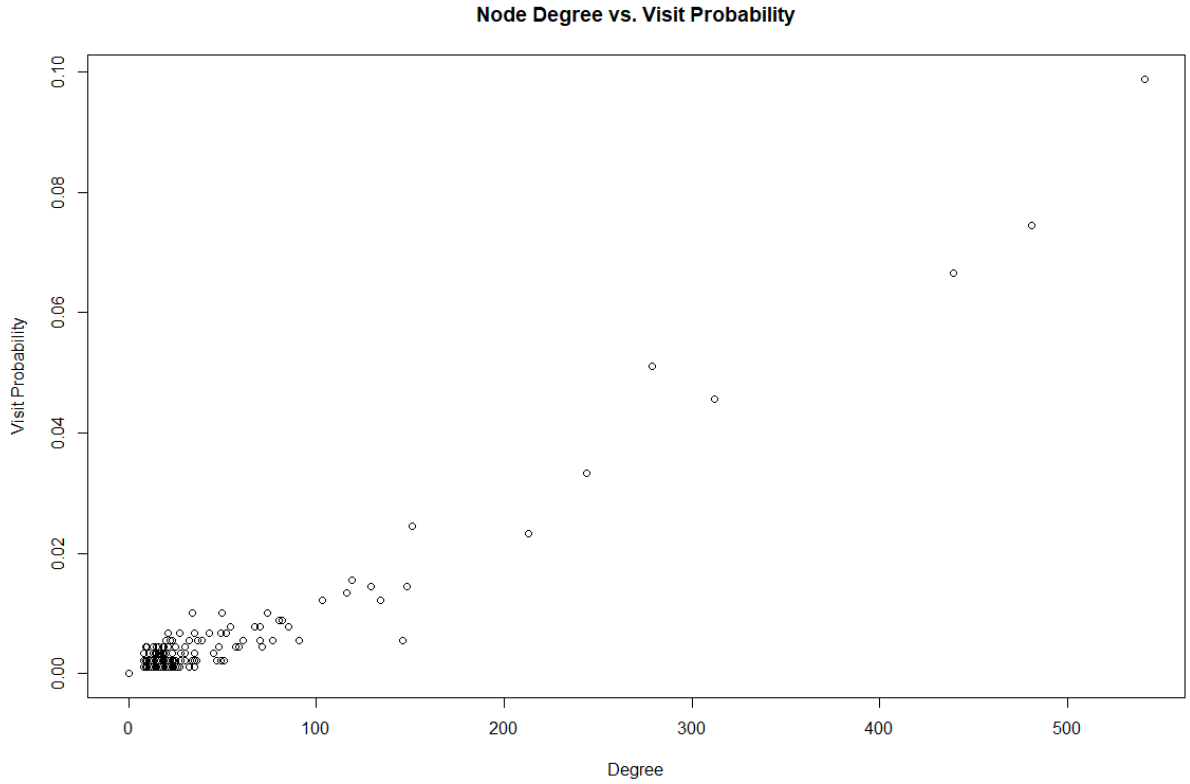


Figure 68: Probability of the walker reaching a node of a certain degree with Normal PageRank values for teleportation.

The normal PageRank values produce a graph with a similar shape to the normal teleporation one in 1a where there is still a high probability of older nodes being visited and a much lower chance of newer nodes being visited. The graph also still follows the Power Law.

**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?**

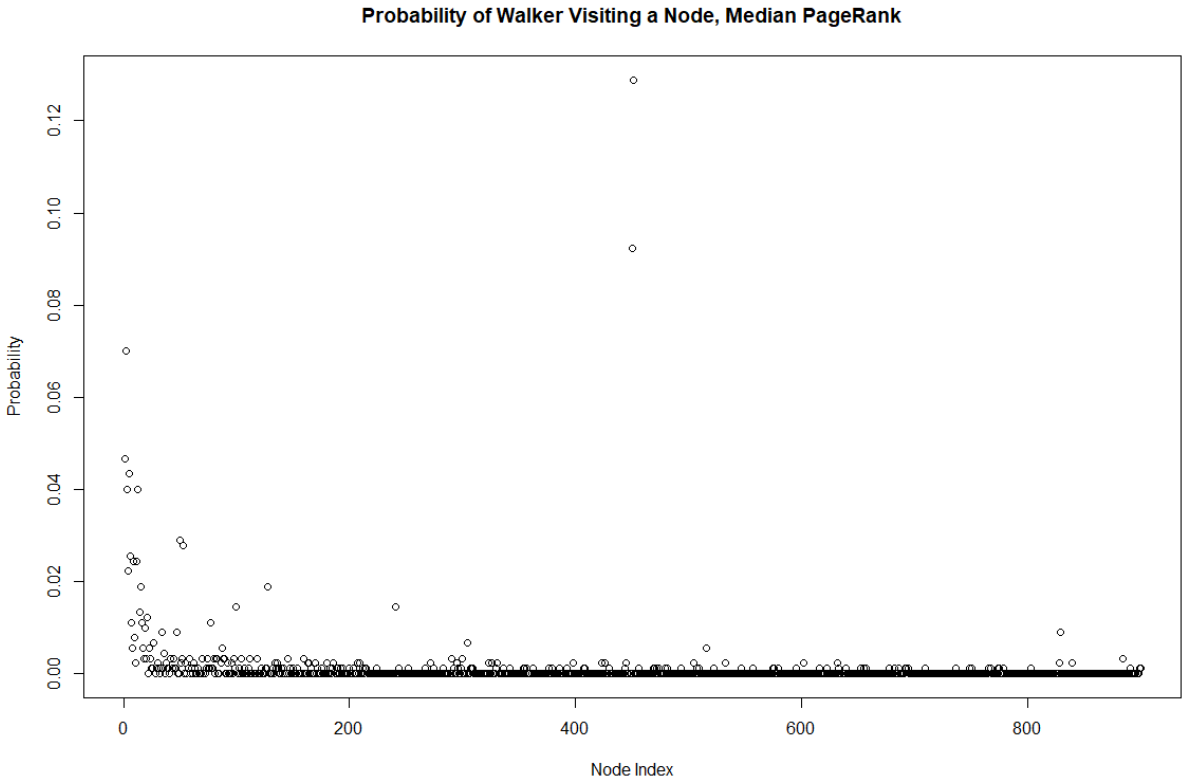


Figure 69: Probability of the walker reaching a node with Median PageRank values for teleportation.

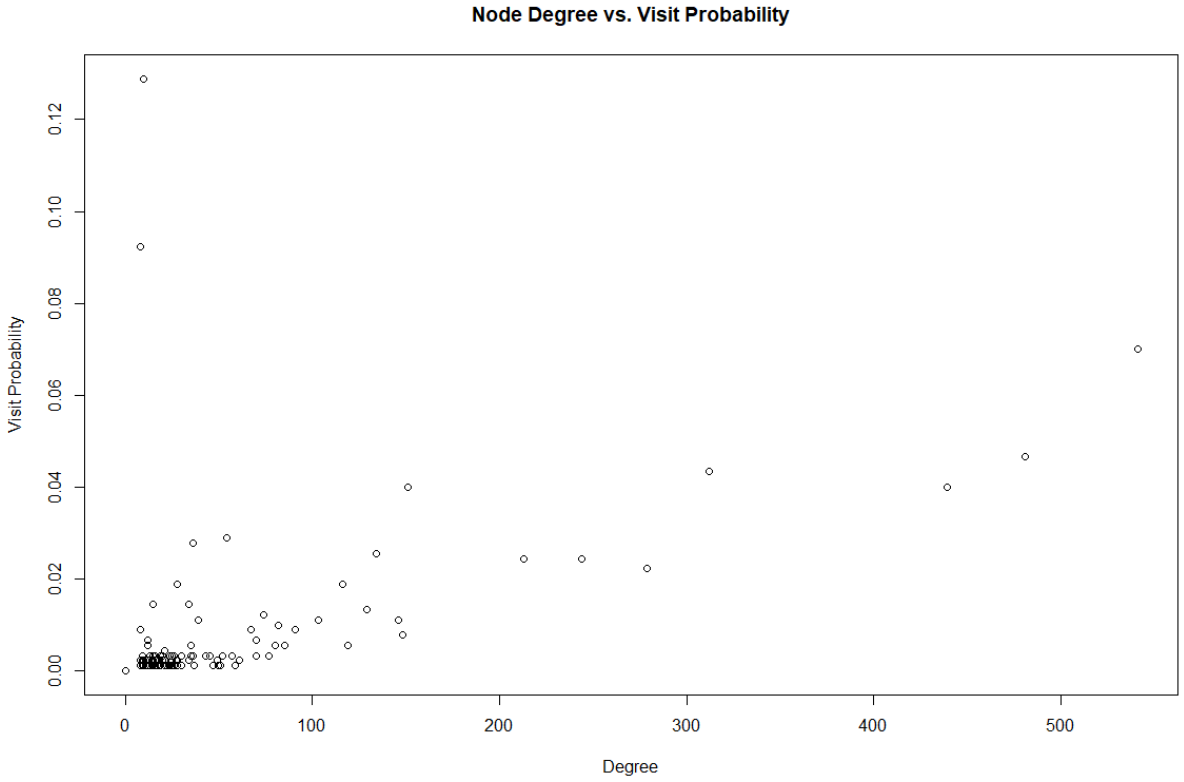


Figure 70: Probability of the walker reaching a node of a certain degree with Median PageRank values for teleportation.

Figures 69 and 70 show the use of PageRank again except if the two median nodes were the only two teleported to and had a probability of 0.5. The resulting graphs are still very simliar to the ones in 4a except they notably have two very distinct points on them that represent when the walker teleports to the two median nodes. They also show that the walker visits slightly newer nodes and nodes with slightly higher degrees more frequently than before.

**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?**

If we try to adjust the PageRank values to account for people's actual tendency when surfing the internet, we would need to combine the values in a PageRank where the chance of visiting any node during teleportation is equal with the values in 4b where there are two nodes that have a higher chance of being teleported to.

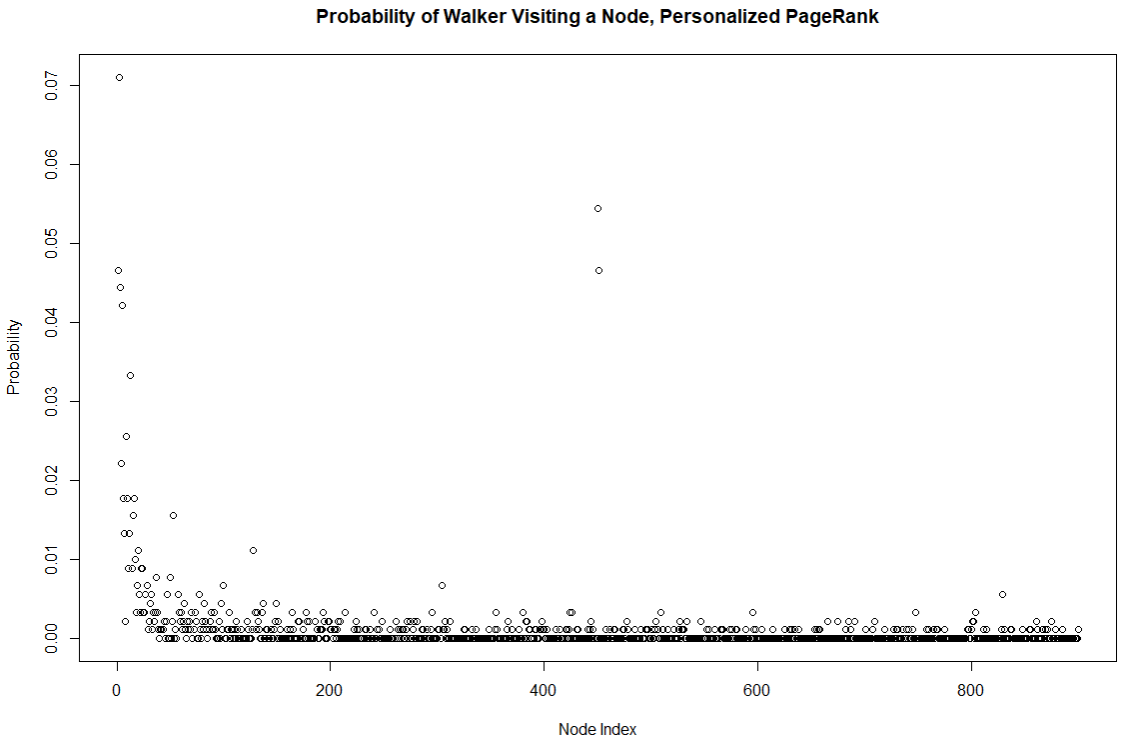


Figure 71: Probability of the walker reaching a node with Personalized PageRank values for teleportation.

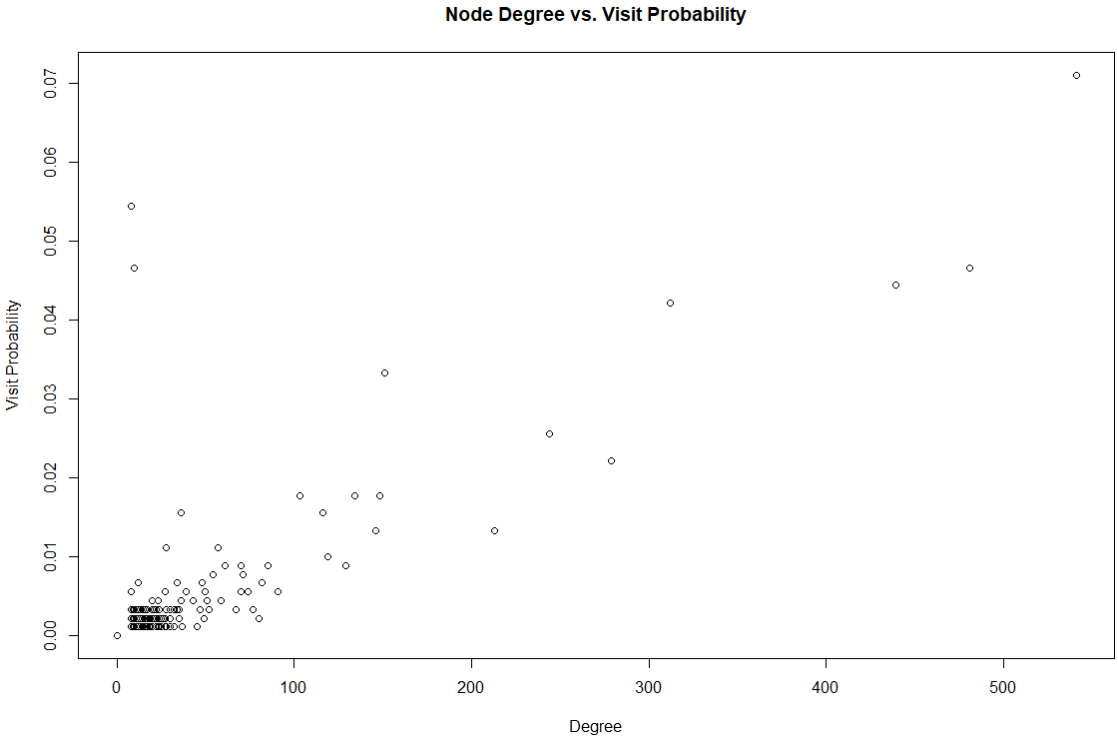


Figure 72: Probability of the walker reaching a node of a certain degree with Personalized PageRank values for teleportation.

The resulting graphs in Figure 71 and 72 show that the results are still similar to the graphs in 4b. The main difference is that the walker has a higher chance of visiting any given node as shown by the line of non-zero probability points graphed across all nodes in Figure 71. Figure 72 also shows a noticeable increase in the probability of the walker vising lower degree nodes as compared to the graph in Figure 68 and 70.