ECE 232E Project 1 Report

Random Graphs and Random Walks

Di Jin (305026178)

Tao WU (504946672)

Yangyang Mao (504945234)

Yufeng Huang (704944399)

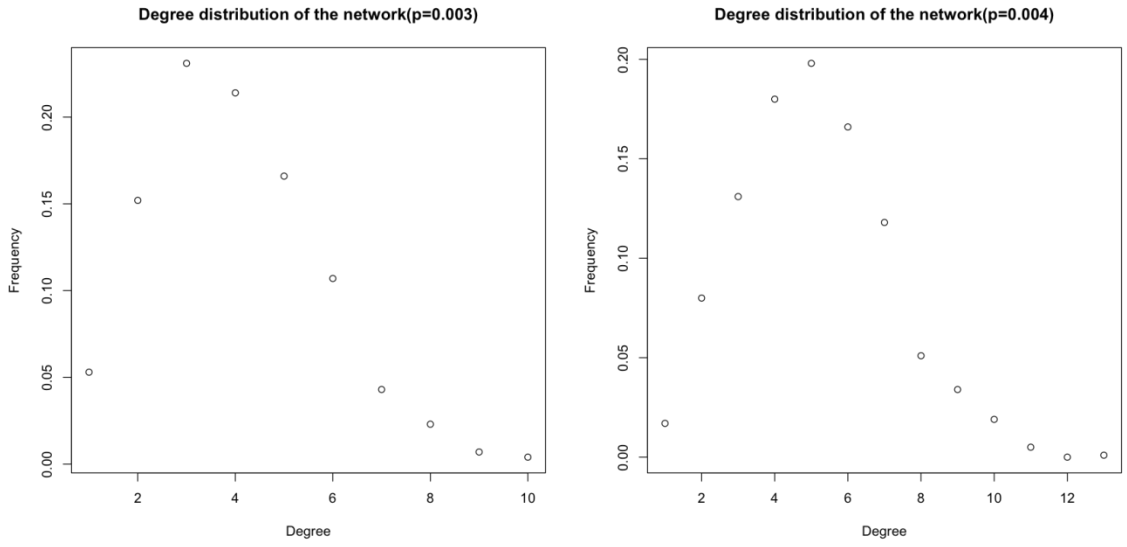
4/22/2018

Part 1 Generating Random Networks

1. Create random networks using Erdös-Rényi (ER) model

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

We generate 5 undirected random networks by using Erdos-Renyi model: g1(p=0.003), g2(p=0.004),g3(p=0.01), g4(p=0.05),g5(p=0.1), and then plot the degree distribution of these networks, as shown in figure 1.1.



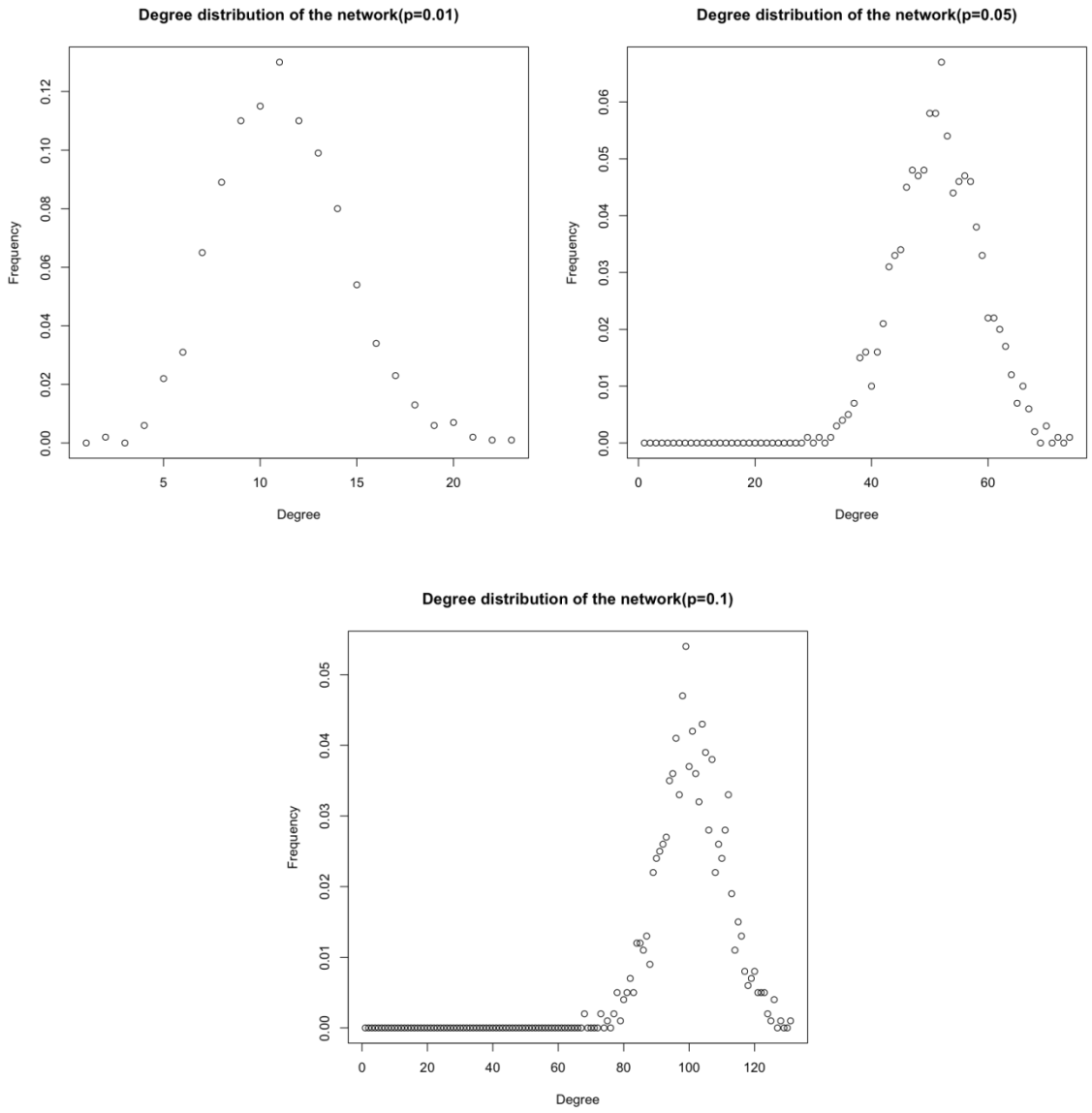


Figure1.1.1

In the degree distribution of the 5 networks, we can observe that with p incre asing, the degree is getting larger, and the frequency of each degree is gettin g smaller which means there are more values of degrees.

We calculated the mean and variance of each network:

```
-----g1-----
mean for g1 is 3.222
variance for g1 is 3.14386
-----g2-----
mean for g2 is 4.122
```

```
variance for g2 is 4.385502
-----g3-----
mean for g3 is 9.984
variance for g3 is 9.877622
-----g4-----
mean for g4 is 49.714
variance for g4 is 51.63384
-----g5-----
mean for g4 is 100.304
variance for g5 is 90.8164
```

And we calculated their theoritical value:

p=0.003: mean= np =3, variance is np(1-p)=2.991

p=0.004: mean= np =4, variance is np(1-p)=3.984

p=0.01: mean= np =10, variance is np(1-p)=9.9

p=0.05: mean= np =50, variance is np(1-p)=47.5

p=0.1: mean= np =100, variance is np(1-p)=90

Thus, we can conclude that the mean and variance we obtain from the generated networks are closer to the theoritical value.

b) For each p and n = 1000, answer the following questions: Are all random realizations of the ER network connected? Numerically estimate the probability that a generated network is connected. For one instance of the networks with that p, find the giant connected component (GCC) if not connected. What is the diameter of the GCC?

```
g1 connectivity: FALSE
g2 connectivity: FALSE
g3 connectivity: TRUE
g4 connectivity: TRUE
g5 connectivity: TRUE
```

We can find that g1 and g2 are disconnected, g3, g4, g5 are connected, then we numerically estimate the probability that a generated network is connected:

```
g1 Connectivity: 0
g2 Connectivity: 0
g3 Connectivity: 0.980198
g4 Connectivity: 0.990099
g5 Connectivity: 0.990099
```

So we can conclude that the Numerical connectivity of g1 and g2 is 0, g3 is 0.98198, g4 and g5is 0.99099. So we need to find GCC of g1 and g2.

```
For g1: GCC is 935, and diameter is 13

For g2: GCC is 982, and diameter is 12
```

c) It turns out that the normalized GCC size (i.e., the size of the GCC as a fraction of the total network size) is a highly nonlinear function of p, with interesting properties occurring for values where p = O(ln n n). For n = 1000, sweep over values of p in this region and create 100 random networks for each p. Then scatter plot the normalized GCC sizes vs p. Empirically estimate the value of p where a giant connected component starts to emerge (define your criterion of "emergence")? Do they match with theoretical values mentioned or derived in lectures?

We scatter plot the normalized GCC sizes vs p, as shown in Figure 1.1.2:

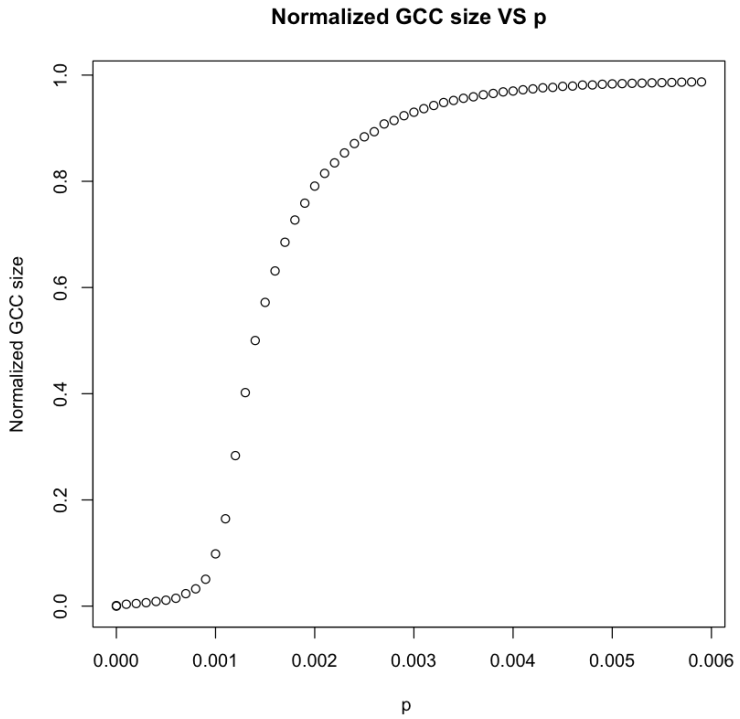


Figure1.1.2

Value of p where a giant connected component starts to emerge is 0.0054739 13 in the experiment. The theoretical value of p should be ln n/n = 0.006907 8. Thus they are basically equal, but the practical value is smaller than the th eoretical value.

d)

i. Define the average degree of nodes c = n × p = 0.5. Sweep over number of nodes, n, ranging from 100 to 10000. Plot the expected size of the GCC of ER networks with n nodes and edge-formation probabilities p = c/n, as a function of n. What trend is observed?

ii. Repeat the same for c = 1.

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

We Plot the expected size of the GCC of ER networks with n nodes and edg e-formation probabilities p = c/n, as a function of n.

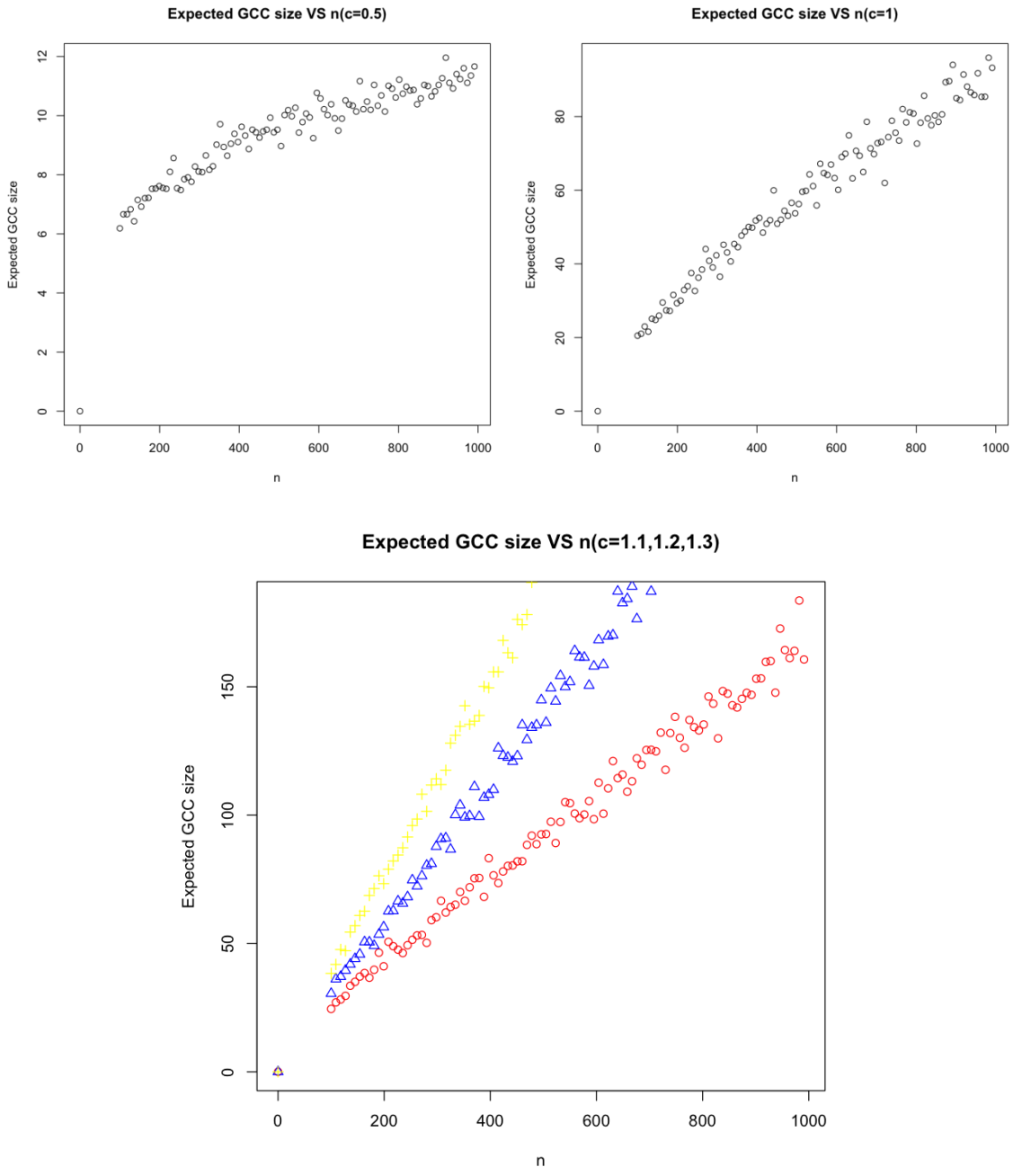


Figure1.1.3

As shown in Figure 1.1.3, expected GCC size is linearly proportional to the number of nodes. In the last figure, the yellow scatter represents c=1.3, blue represents c=1.2, red represents c=1.1. So larger c will lead to larger expecte d GCC size.

2. Create networks using preferential attachment model

a) Create an undirected network with n = 1000 nodes, with preferential attachment model, where each new node attaches to m = 1 old nodes. Is such a network always connected?

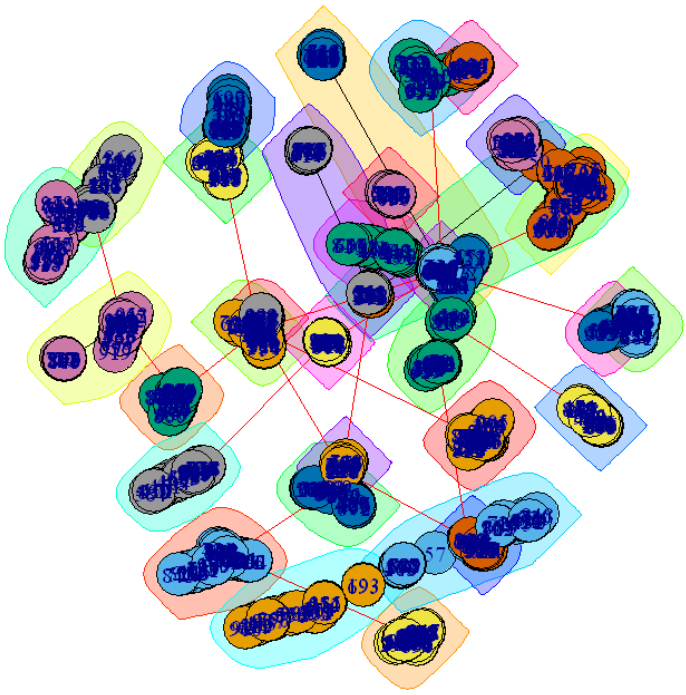
After using preferential attachment model (barabasi.game) to create an undirected network with node n =1000, and make sure m is equal to 1 old nodes this time. We use is.connected function to check if the network is connected. The result shows that, this network is always connected.

```
g1000 connectivity: TRUE
```

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

Then we try to use fast greedy method to find the community structure. We display the community size and modularity of this network (0.9350286) below. Also, the graph of community structure is attached for a direct view.

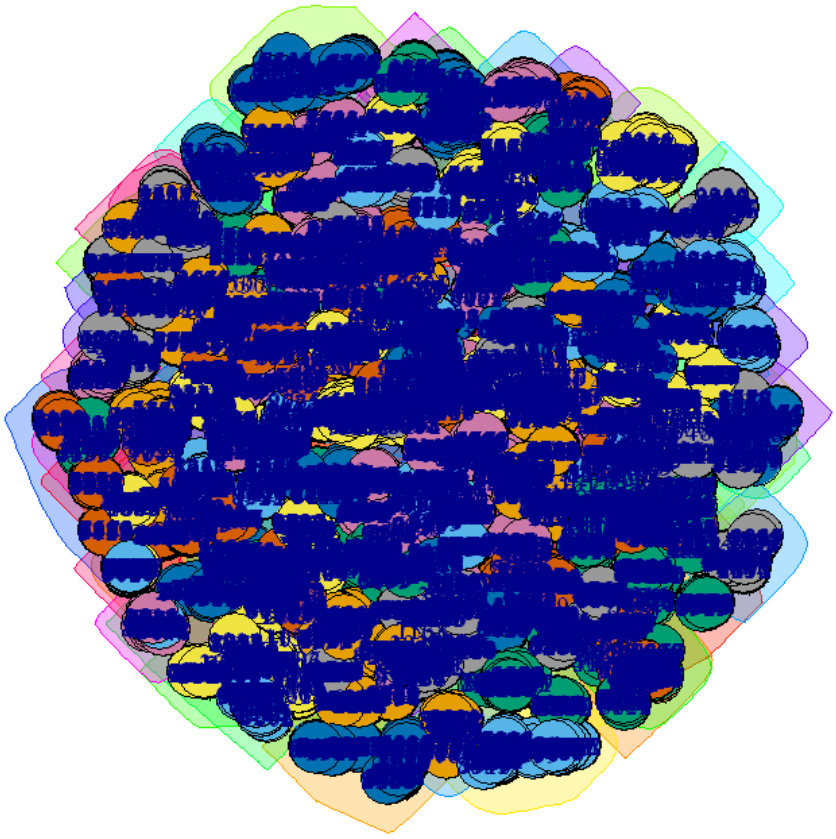
```
Community sizes
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
48 45 41 48 39 40 45 37 40 39 35 33 34 30 33 31 29 30 30 27 27 24 27 24 24 28
27 28 29 30 31 32
19 26 17 18 17 15
[1] 0.9350286
```



c) Try to generate a larger network with 10000 nodes using the same model. Compute modularity. How is it compared to the smaller network's modularity?

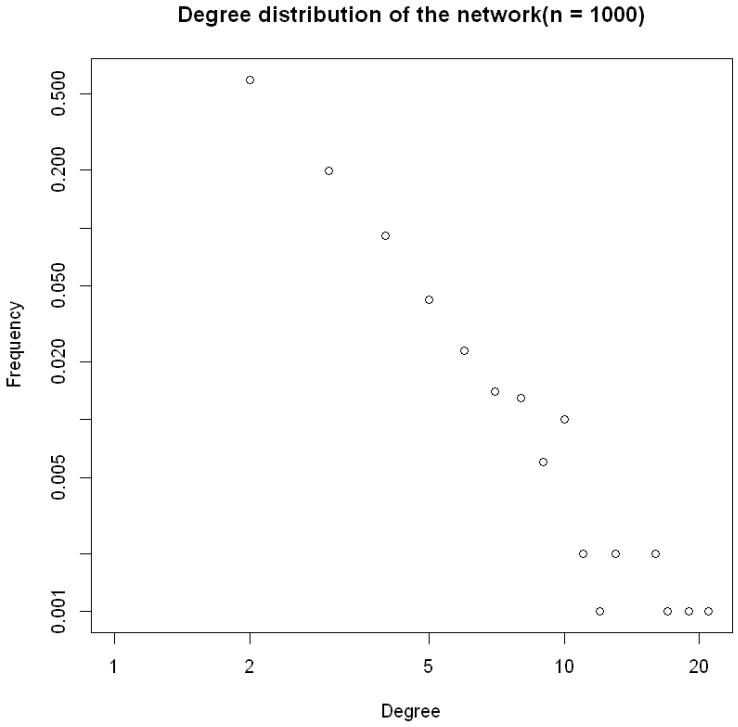
After that, we changed the total number of nodes from 1000 to 10000. Modularity is changed to 0.9780909. We can see that with the increase of the number of nodes, the modularity is increased too. Since the number of nodes is fairly large, modules look like more concentrate.

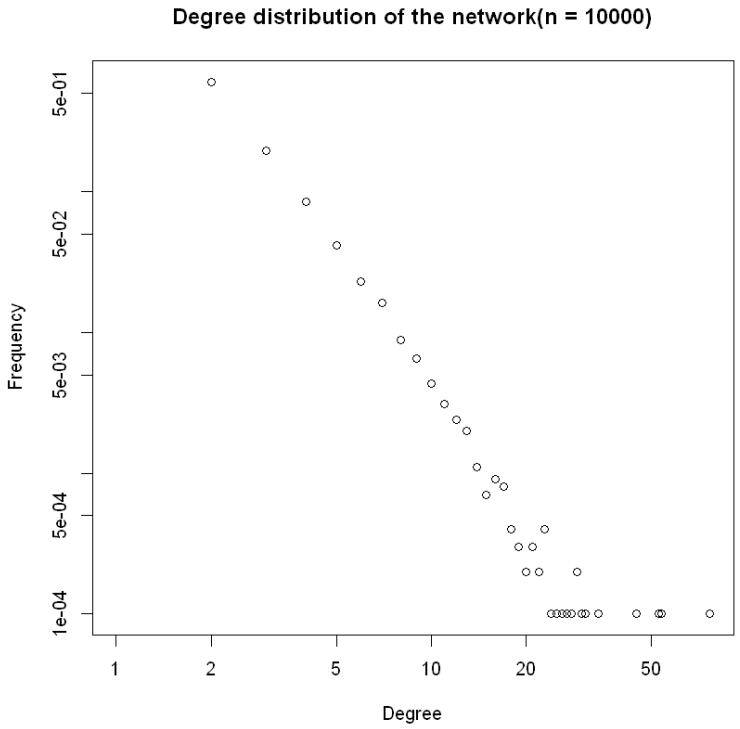
```
g10000 connectivity: TRUE
Community sizes
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
260 140 162 195 129 146 136 143 126 154 128 133 127 177 124 133 158 118 139 116
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
138 120 123 111 111 116 117 117 114 116 134 137 111 108 105 110 118 104 114 120
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
102 100 110 101 104 107  95  89 107  90  90 102  88  94  86  82  93  85  82  77
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
 75  70  72  70  69  85  75  68  72  72  67  66  66  63  66  67  60  61  59  59
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
 59  63  56  61  52  52  50  55  50  48  50  47  55  47  58  48  47  49  42  50
101 102 103 104 105 106 107 108 109 110 111 112
 40  41  40  38  37  41  32  33  33  34  30  28
[1] 0.9780909
```



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

Now we have two networks: one has 1000 nodes, the other one has 10000 nodes. We plot the log-log scale graph of the degree distribution for both networks





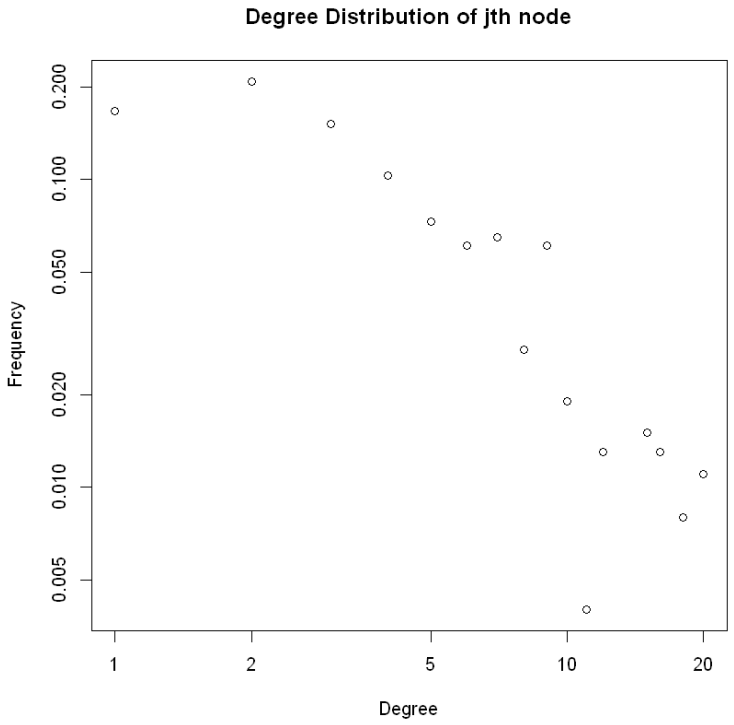
Based on the plot we get, we estimate the slope for both of these networks:

```
Call:
lm(formula = a ~ c)

Residuals:
     Min       1Q   Median       3Q      Max
-0.09918 -0.07524 -0.04229  0.03444  0.41281

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.207697   0.067271   3.087  0.00803 **
c           -0.014253   0.005783  -2.465  0.02727 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1299 on 14 degrees of freedom
Multiple R-squared:  0.3026,    Adjusted R-squared:  0.2528
F-statistic: 6.074 on 1 and 14 DF,  p-value: 0.02727
```

```
Call:
lm(formula = d ~ e)

Residuals:
     Min       1Q   Median       3Q      Max
-0.04889 -0.04078 -0.02534 -0.00980  0.53351

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.074312   0.029219   2.543   0.0159 *
e           -0.002112   0.001088  -1.941   0.0608 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1022 on 33 degrees of freedom
Multiple R-squared:  0.1025,    Adjusted R-squared:  0.07532
F-statistic: 3.769 on 1 and 33 DF,  p-value: 0.06078
```

We can see that the slope is -0.014253 when number of nodes is 1000. Slope is -0.002112 when number of nodes is 10000.

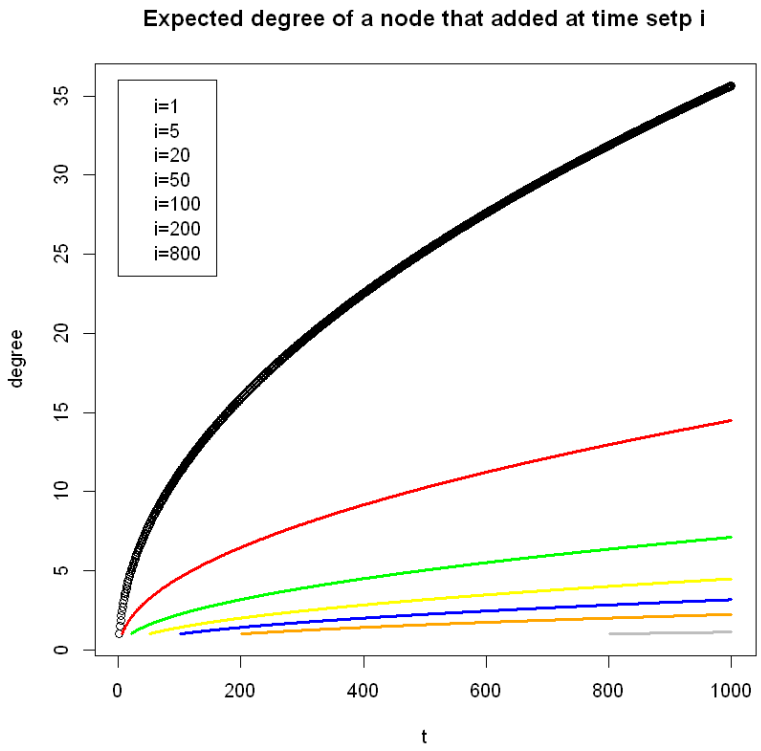
e) You can randomly pick a node i, and then randomly pick a neighbor j of that node. Plot the degree distribution of nodes j that are picked with this process, in the log-log scale. How does this differ from the node degree distribution?



From the graph we can also see that when the degree is getting larger, the frequency is decreasing. This trend is similar to what we have in the part d

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

Then we estimate the expected degree of a node that is added at time step i for $1 \leq i \leq 1000$. The relationship between the age of nodes and their expected degree is shown below:



From the figure above we can easily see that degree is increasing with the increase of the edge. This trend is obvious when the node is old. When i equal to 800, the increase of degree is not as rapid as when I smaller than 800. It is more like a line.

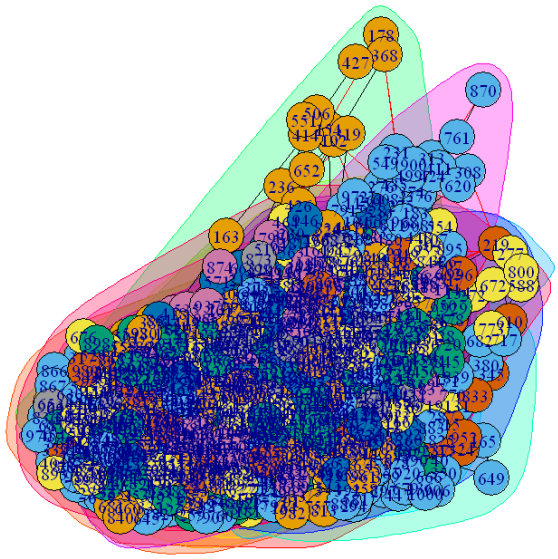
g) Repeat the previous parts for m = 2, and m = 5. Why was modularity for m = 1 high?

Here we change the value of out-degree of node vertices. And then repeat what we have done before. It is obvious that the when m is increase the modularity is decreasing. I think this is because when m is small, it will be more easily to find relationship among one node and others. But when m is large, neighbors of each node are increased. More and more nodes are connected. It will be harder to find a single module.

When m = 2:

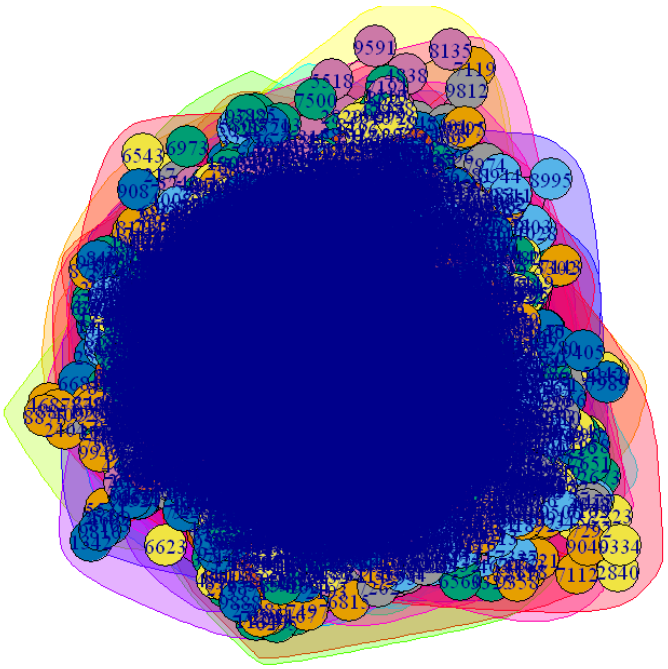
```
g1000 connectivity: TRUE

Community sizes
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 61 103  47  27  49  23  23  22  54  38  45  34  44  59  30  41  77  48  92  83
[1] 0.5224992
```

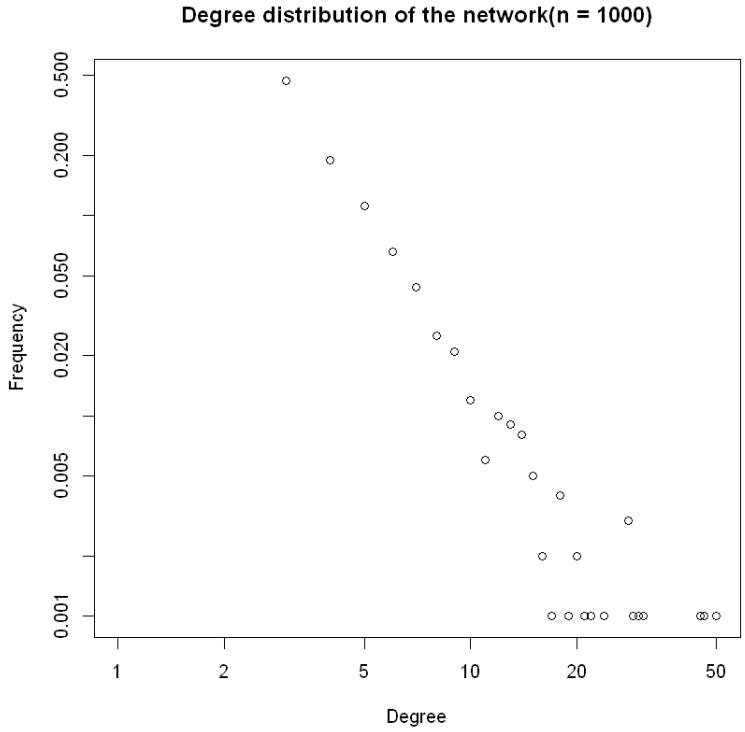
When there are 1000 nodes in the network, the modularity is 0.22



```
g10000 connectivity: TRUE
Community sizes
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
209 119 118 162 505 187  86  86  80 140 183 281 138  72  77 201 213 117 398  59
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
663 708  91 245 167 396  28 376 247 262 302 364 770 353 622 975
[1] 0.5342031
```

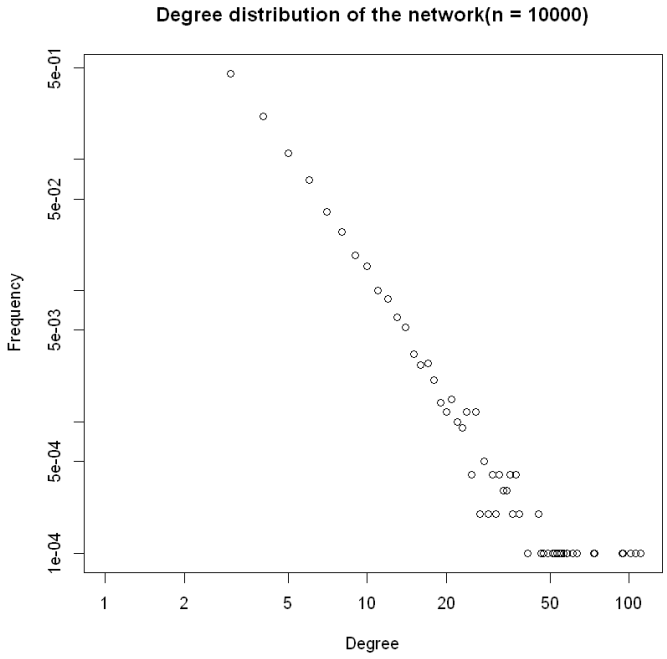
When there are 10000 nodes in the network, the modularity is 0.534




Degree distribution of the network(n = 1000)
Frequency
0.500
0.200
0.050
0.020
0.005
0.001
1
2
5
10
20
50
Degree

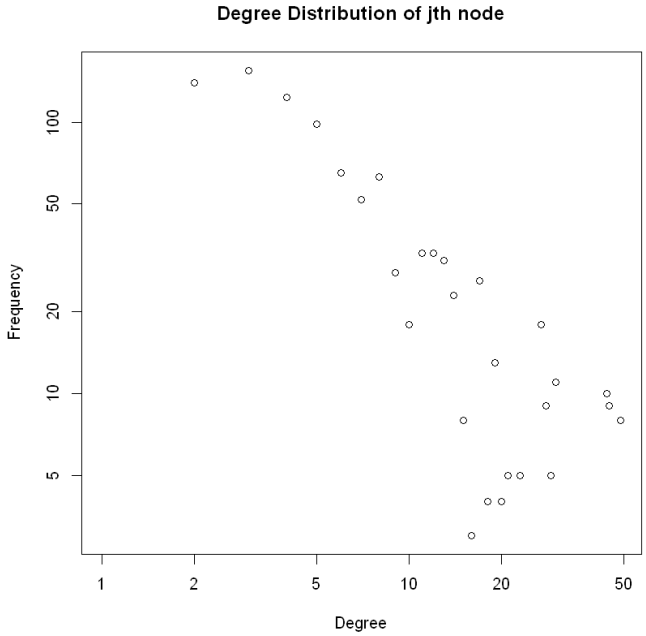
```
Call:
lm(formula = a ~ c)

Residuals:
     Min       1Q   Median       3Q      Max
-0.05567 -0.04381 -0.02948  0.00151  0.38349

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.09720    0.03003   3.237  0.00329 **
c           -0.00323    0.00132  -2.447  0.02145 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.08702 on 26 degrees of freedom
Multiple R-squared:  0.1872,    Adjusted R-squared:  0.156
F-statistic:  5.99 on 1 and 26 DF,  p-value: 0.02145
```

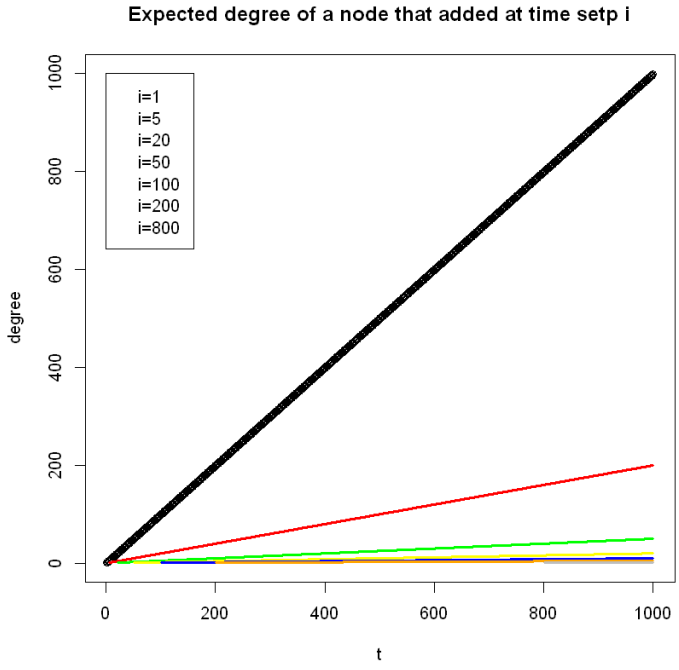
The slop for this log-log scale figure is about -0.0033.




```
Call:
lm(formula = d ~ e)

Residuals:
     Min       1Q   Median       3Q      Max
-0.03172 -0.02675 -0.01734 -0.00151  0.40475

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0468069  0.0144794   3.233  0.00207 **
e           -0.0007857  0.0003154  -2.491  0.01579 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0639 on 55 degrees of freedom
Multiple R-squared:  0.1014,    Adjusted R-squared:  0.08504
F-statistic: 6.205 on 1 and 55 DF,  p-value: 0.01579
```

The slop for this log-log scale figure is about -0.0007857.

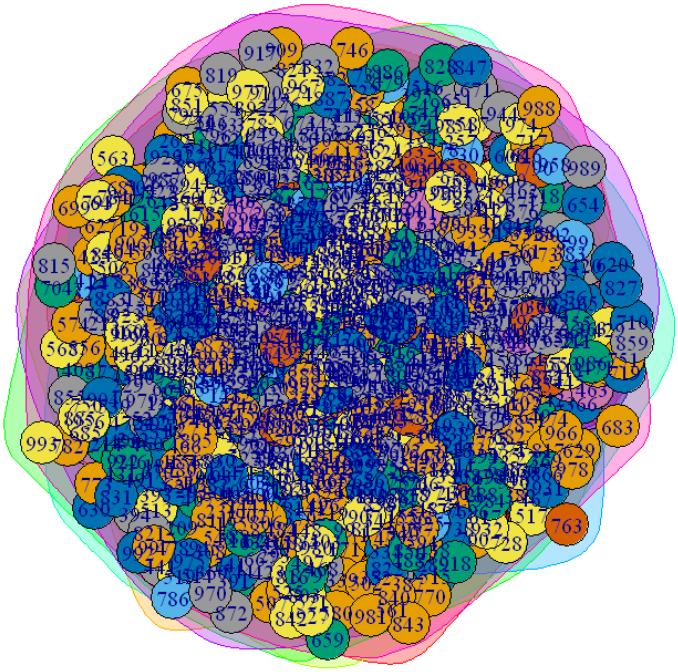




When m = 5:

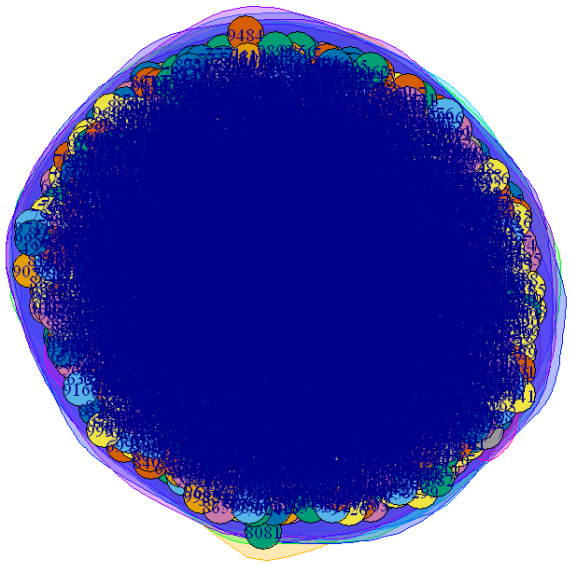
```
g1000 connectivity: TRUE
```

```
Community sizes
  1   2   3   4   5   6   7   8   9
 49  27 104 208 209  29   9 158 207
[1] 0.2818202
```



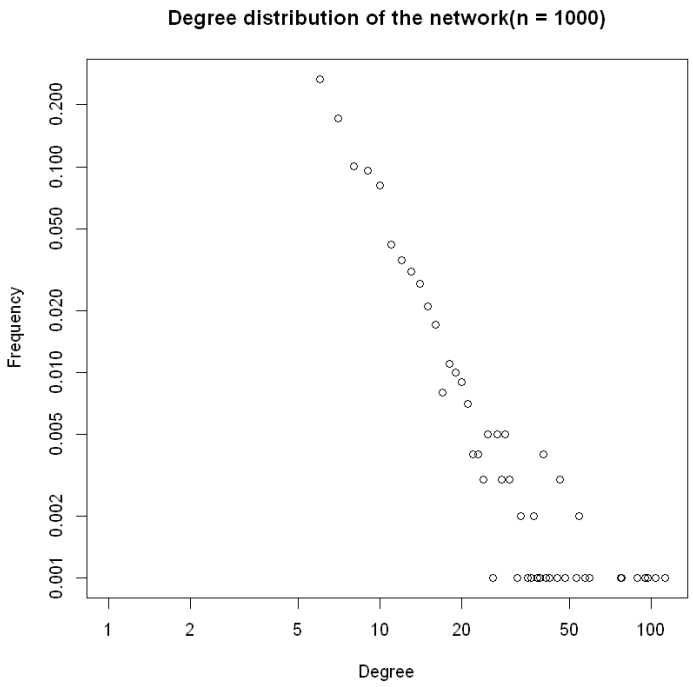
When there are 1000 nodes in the network, the modularity is 0.2818202

```
g10000 connectivity: TRUE
Community sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 186   65  356   53   65   48 1994  103  130  921  495 1568 1926 2069    8    7
  17
   6
[1] 0.2771717
```

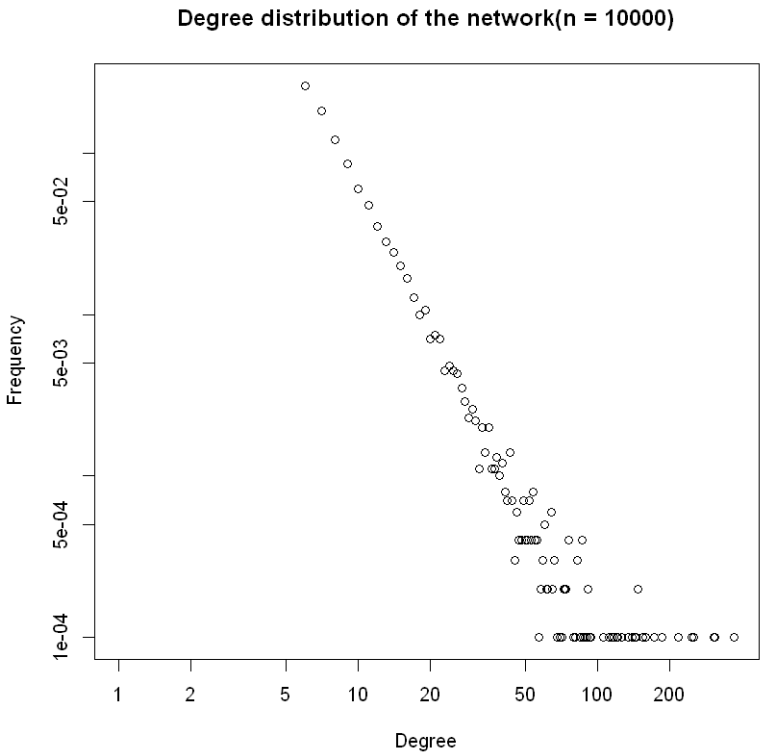


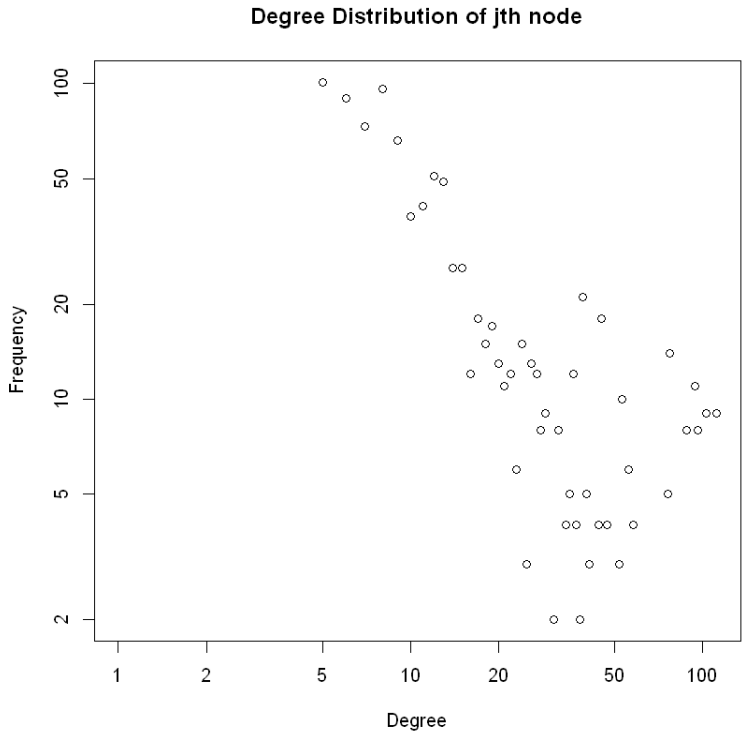
When there are 10000 nodes in the network, the modularity is 0.2771717

```
Call:
lm(formula = a ~ c)

Residuals:
      Min        1Q    Median        3Q       Max
-0.028062 -0.023305 -0.016035  0.001673  0.222916

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0485923  0.0108895   4.462 5.04e-05 ***
c           -0.0007514  0.0002361  -3.183  0.00259 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04437 on 47 degrees of freedom
Multiple R-squared:  0.1773,    Adjusted R-squared:  0.1598
F-statistic: 10.13 on 1 and 47 DF,  p-value: 0.002586
```

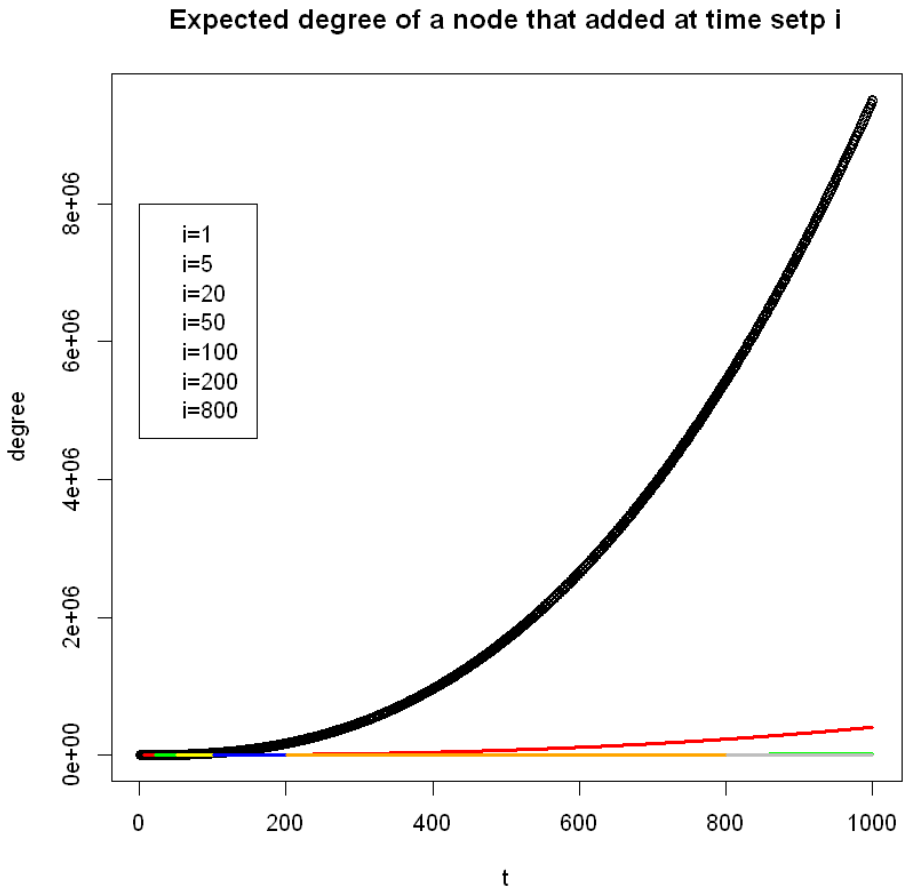



```
Call:
lm(formula = d ~ e)

Residuals:
      Min        1Q    Median        3Q       Max
-0.014841 -0.012736 -0.010018 -0.001675  0.242468

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0203614  0.0050942   3.997 0.000124 ***
e           -0.0001381  0.0000503  -2.747 0.007165 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03424 on 98 degrees of freedom
Multiple R-squared:  0.07148,   Adjusted R-squared:  0.062
F-statistic: 7.544 on 1 and 98 DF,  p-value: 0.007165
```





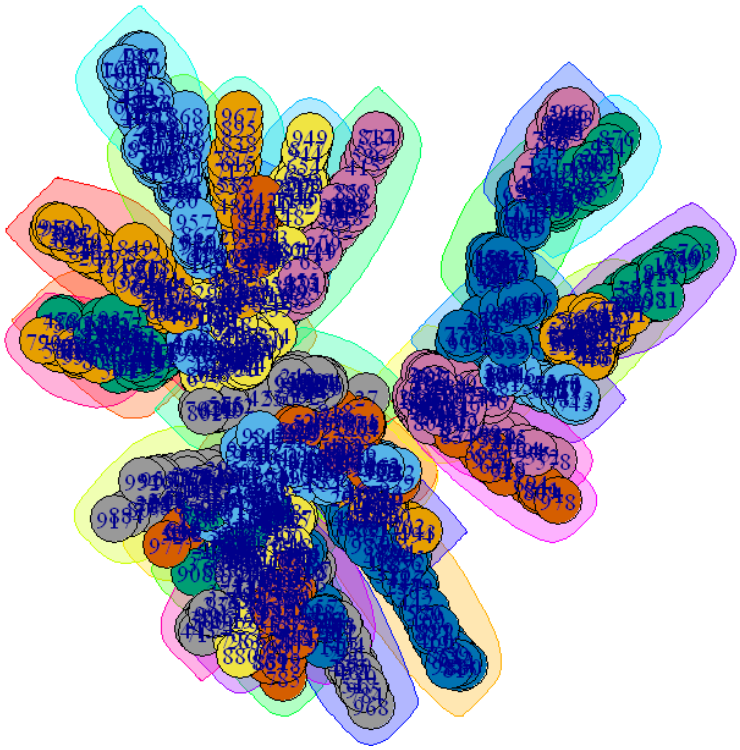
("998 y values <= 0 omitted from logarithmic plot")



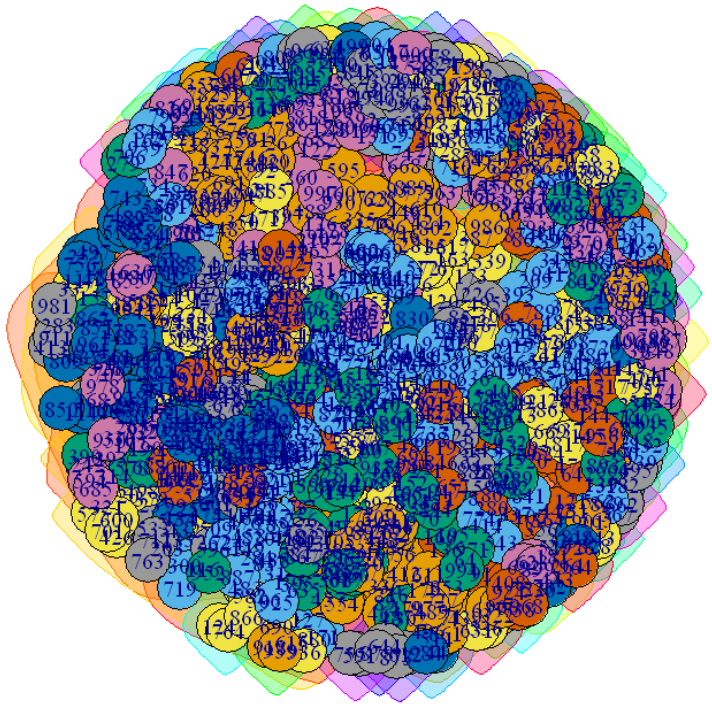
h) Again, generate a preferential attachment network with n = 1000, m = 1. Take its degree sequence and create a new network with the same degree sequence, through stub-matching procedure. Plot both networks, mark communities on their plots, and measure their modularity. Compare the two procedures for creating random power-law networks.

Finally we generate a preferential attachment network with n = 1000, m = 1. Take its degree sequence and create a new network with the same degree sequence

0.930596762929097



0.843294746197651



We find that although we use the same degree sequence for both network, but the modularity is not the same. When we use a modified preferential attachment model, the modularity is 0.93, but when we use same degree to generate network ,we got a modularity at 0.84. We can also see it directly from the plot. The module is not the same.

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

a) In this part, we are required to use the preferential attachment model to produce an undirected network with 1000 nodes. In detail, we use the aging.prefatt.game function to create preferential attachment which sets alpha = 1, beta = -1, and m = 1. Since the default of a, c, d equals to 1 and b equals to 0, we don't need to set these numbers in our preferential function. Then we plot the degree distribution, and get the following figure.

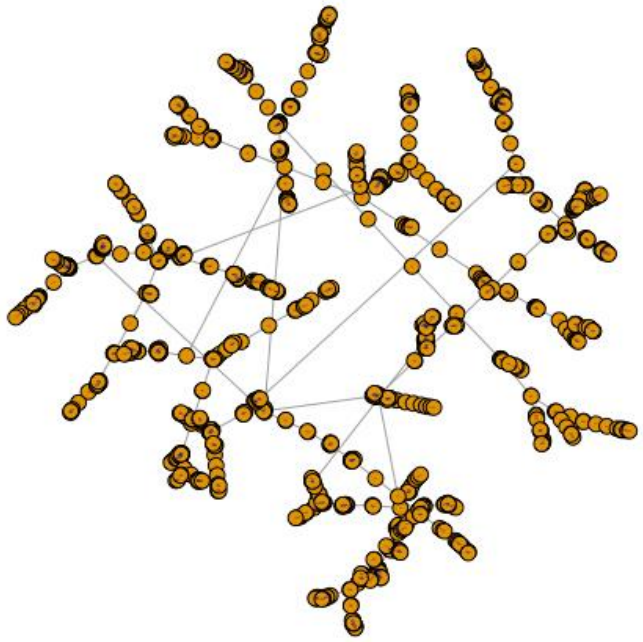


Figure1.3.1: 1000 nodes with m = 1

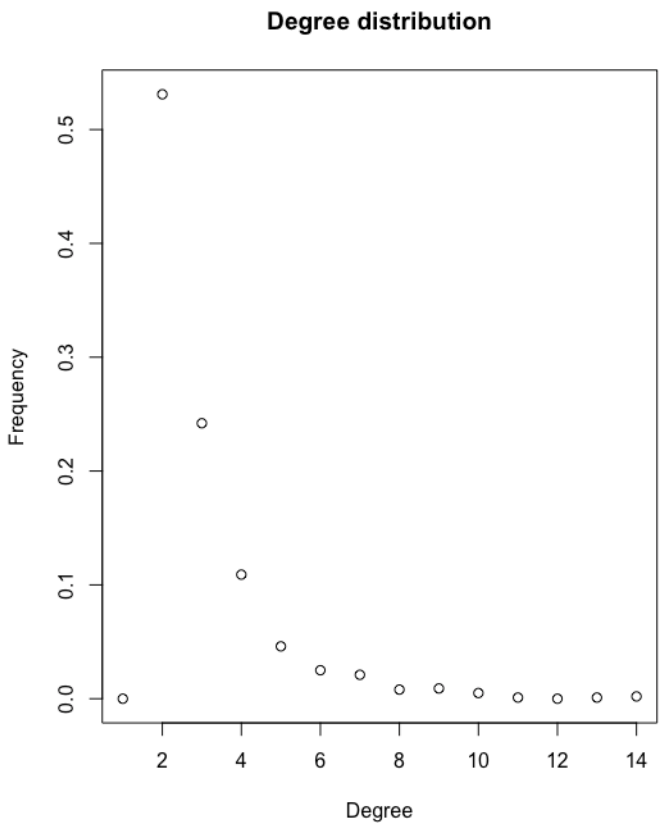


Figure1.3.2: degree vs frequency

The above figure shows that with the increase of degree, the frequency decreases until close to zero. Using the power.law.fit function, we can calculate the power law exponent is 4.711233.

b) To find the community structure, we will use the fastgreefy function to calculate it in a straight way. The result shows the modularity of the graph is 0.935452.

Part 2 Random Walk on Networks

1. Random walk on Erdös-Rényi networks

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

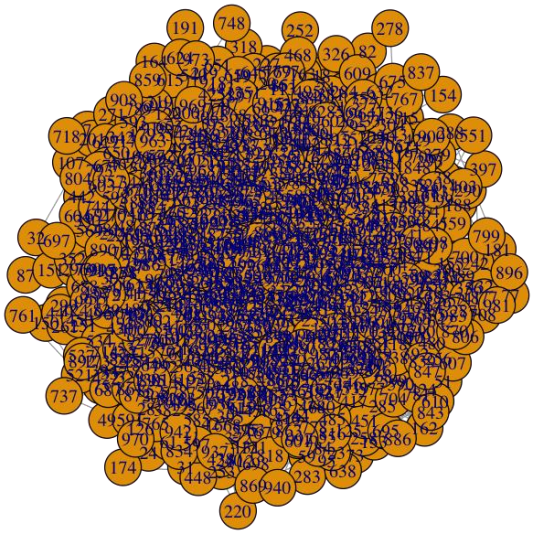


Figure 2.1.1

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 standard deviation $\sigma^2(t) = \langle (s(t) - \langle s(t) \rangle)^2 \rangle$ of this distance. Plot $\langle s(t) \rangle$ vs t and $\sigma^2(t)$ vs t. Here, the average $\langle \cdot \rangle$ is over random choices of the starting nodes.

We plot $\langle s(t) \rangle$ vs t and $\sigma^2(t)$ vs t, as shown in Figure 2.1.2

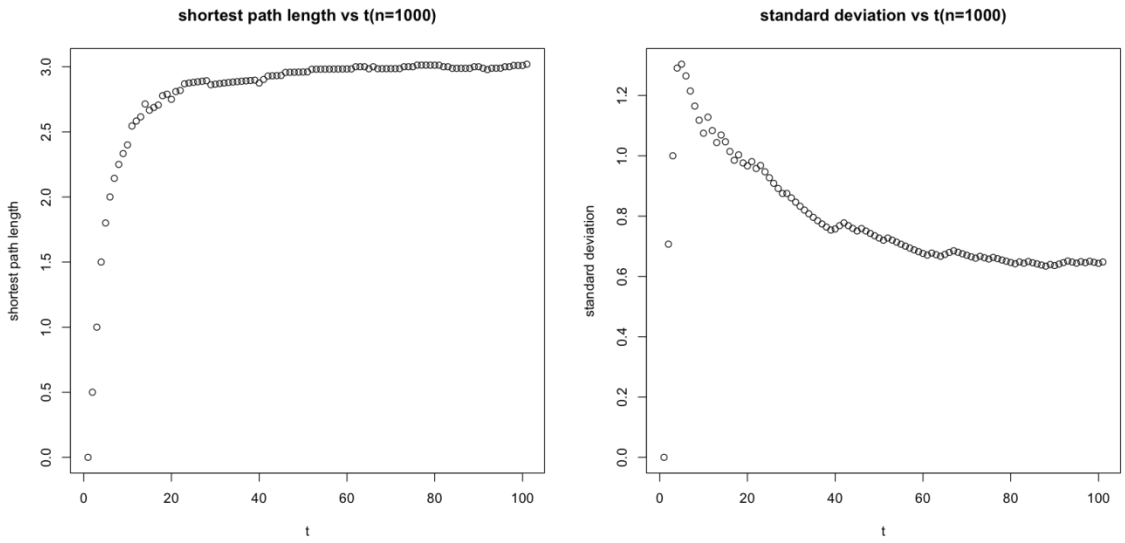


Figure 2.1.2

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

We plot the degree distribution of the nodes reached at the end of the random walk and the degree distribution of graph, as shown in Figure 2.1.3

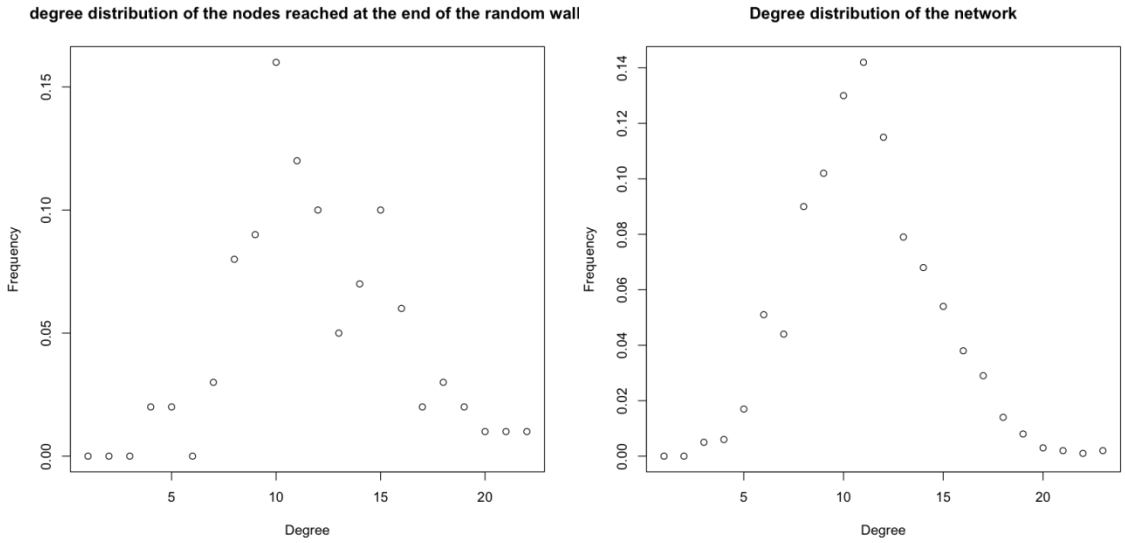


Figure 2.1.3

The degree distribution of the nodes reached at the end of the random walk is similar to the degree distribution of graph.

d) Repeat (b) for undirected random networks with 100 and 10000 nodes. Compare the results and explain qualitatively. Does the diameter of the network play a role?

WE Repeat (b) for undirected random networks with 100 and 10000 nodes, t he result is shown as in Figure 2.1.4:

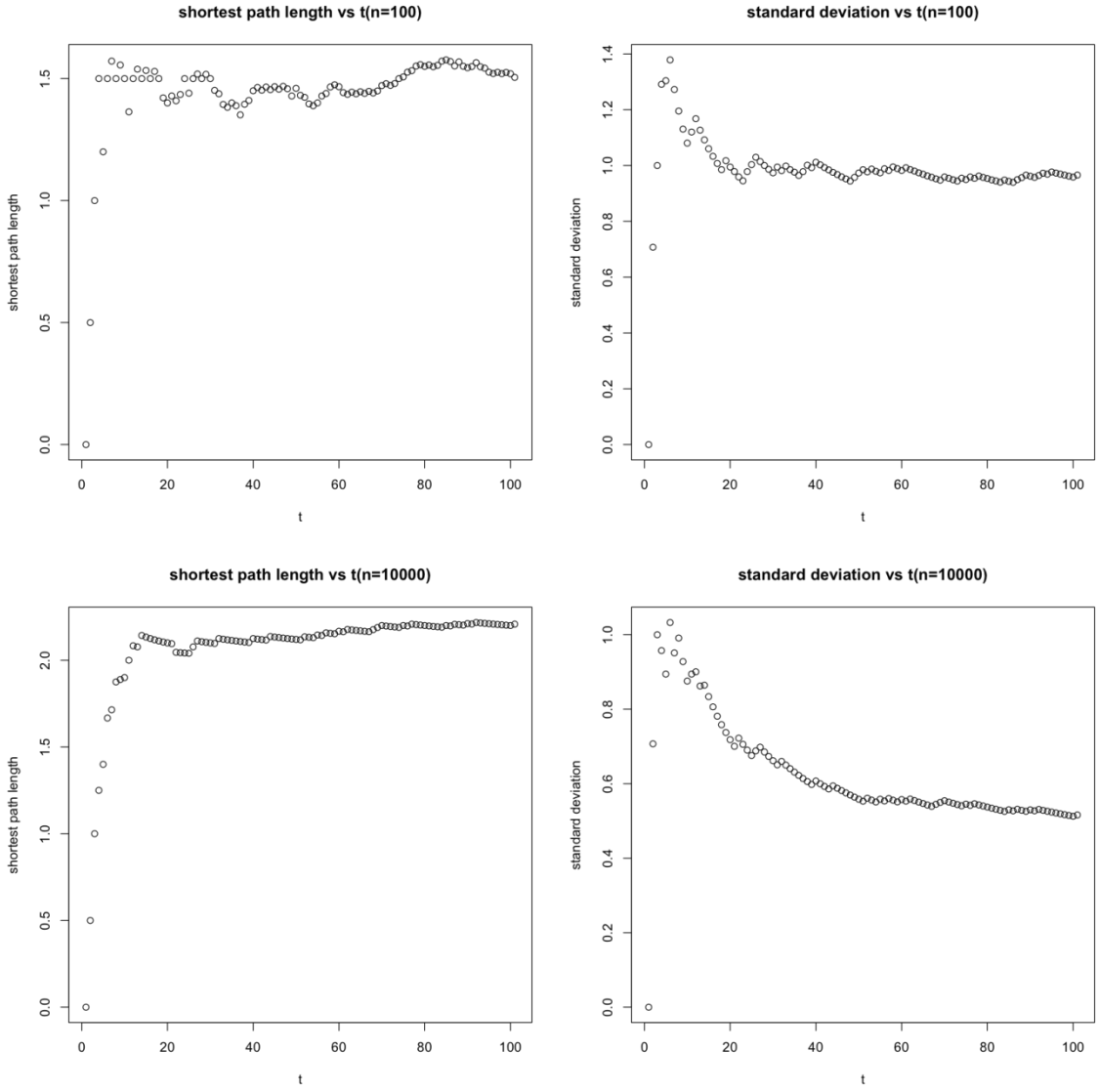


Figure 2.1.4

nodes	100	1000	1000
diameter	6	5	3

As we can see, the diameter of the network is decreasing with the increasing of number of nodes. We could see from Figure 2.1.4, for the graph with 100 nodes, the distribution converges slowly. Thus, the smaller the diameter is, the Average distance and Standard variance converge more quickly. In addition, the distribution of each step for smaller diameter seems to be closer, which means the variance of Standard variance distance becomes smaller.

2. Random walk on networks with fat-tailed degree distribution

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

```
In [12]: g <- barabasi.game(1000, m=1, directed=F)
         plot(g,vertex.size=5, vertex.label.cex=0.1)
```

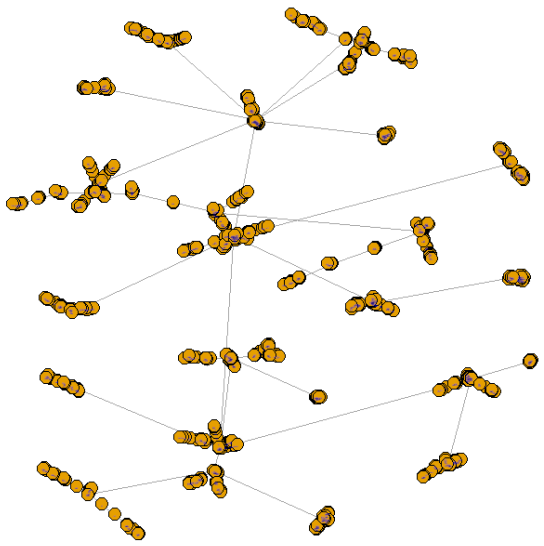


Figure 2.2.1

Firstly, we use BA model to create an undirected preferential attachment network with 1000 nodes and m=1. The plot is shown above.

b) Let a random walker start from a randomly selected node. Measure and plot <s(t)> vs t and $\sigma^2(t)$ vs t.

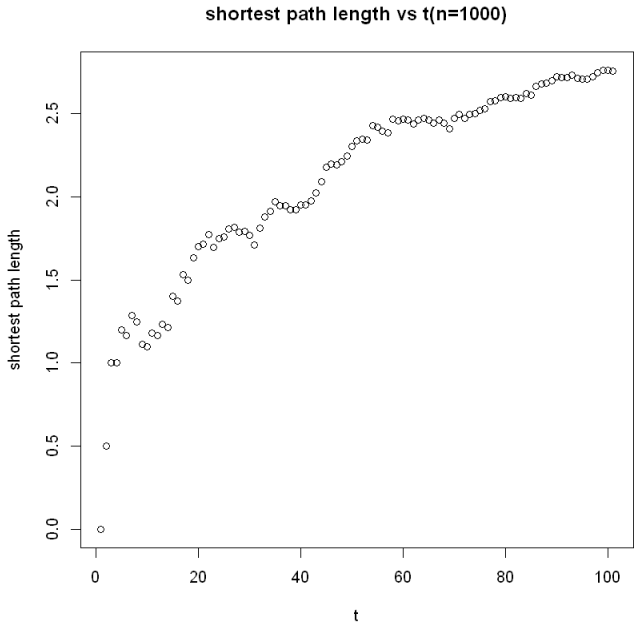


Figure 2.2.2

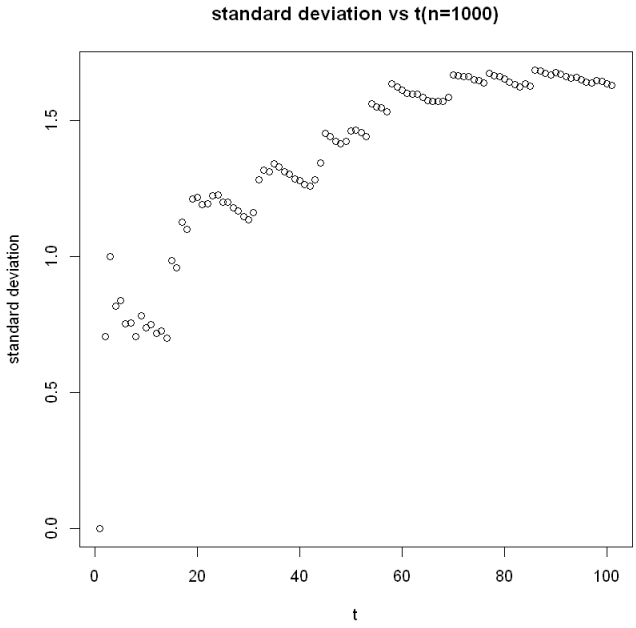


Figure 2.2.3

The figure 2.2.2 and 2.2.2 show the plots of shortest path length and standard deviation vs t. We run 100 steps and set start node to be random, checking the results of given networks. And in each step, we save the shortest path (average distance) and the standard derivation. We can find that overall, the average distance and standard deviation increase with larger t. When t becomes larger, the increasing rate has decreased and experience a slowly rise at a large t. While for standard deviation, although it seems become larger when t increase, it experiences a small decrease in some small range of t.

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?

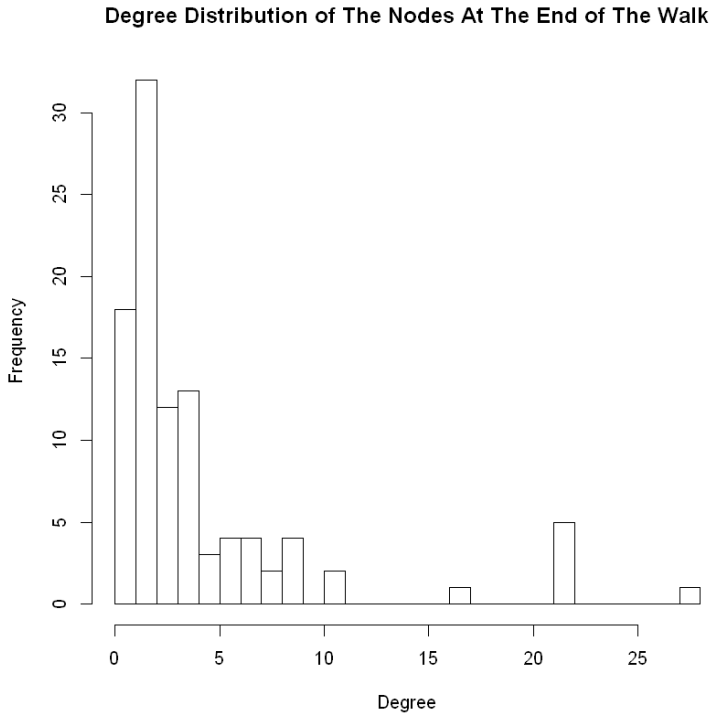

Degree Distribution of The Nodes At The End of The Walk
Frequency
0
5
10
15
20
25
30
0
5
10
15
20
25
Degree

Figure 2.2.4

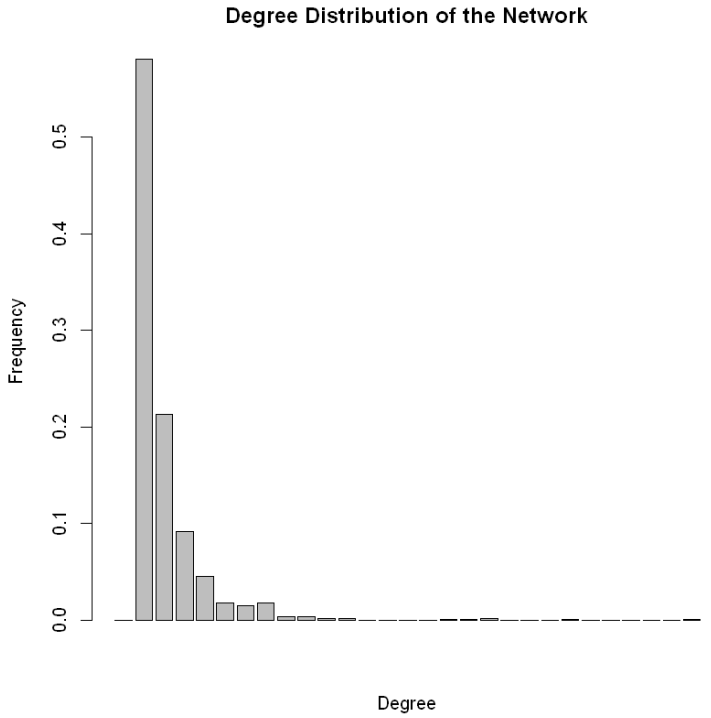

Degree Distribution of the Network
Frequency
0.0
0.1
0.2
0.3
0.4
0.5
Degree

Figure 2.2.5

Figure 2.2.4 and 2.2.5 show that distribution of the nodes reached at the end of the random walk and of the graph. We set the walk step to be 1000 here. We can find that degree distribution of the nodes reached at the end of the random walk is similar with the degree distribution of the graph. They both peak at degree 1 which in this model most of the nodes has degree 1. This may because that for degree 1 has the largest proportion and the walker will visit these nodes with a high probability than other degree. And we can find the two shapes of the distribution is similar and this may also indicate that the walker would visit all the nodes in the networks during its walk.

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

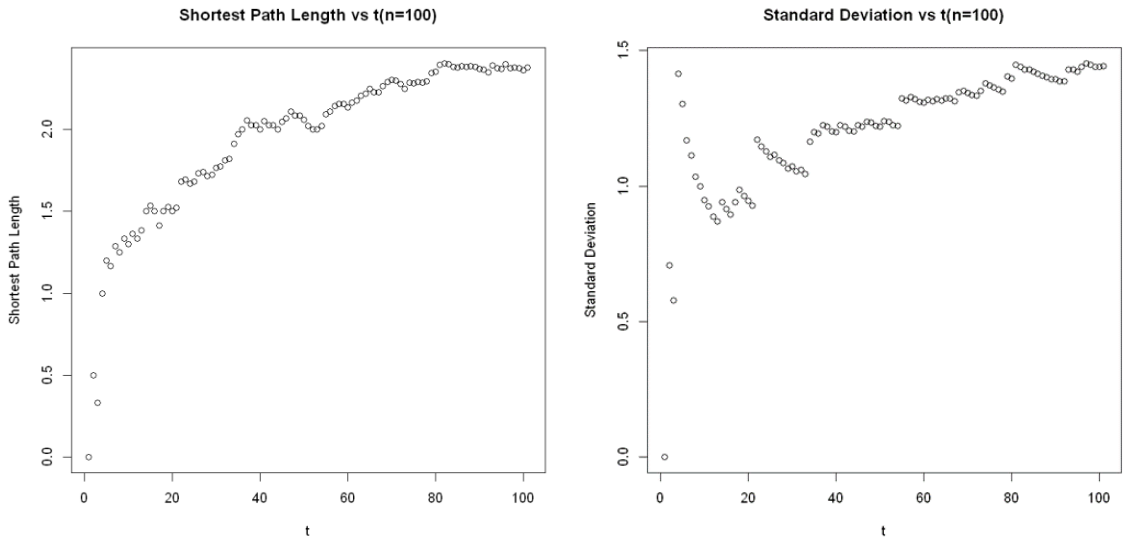


Figure 2.2.6

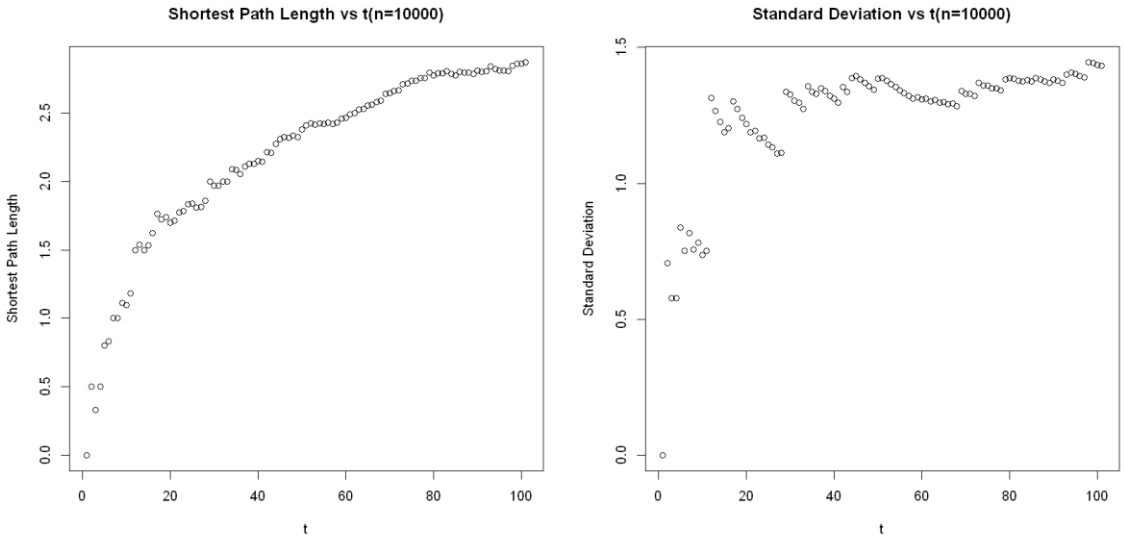


Figure 2.2.7

Number of Nodes	Diameter	Average Distance
100	14	1.95
1000	22	2.10
10000	31	2.19

Table 2.2.1

Figure 2.2.6 and 2.2.7 show that the average distance and standard deviation for n=100 and n = 10000. And the table 2.2.1 shows the diameter and average distance for different number of nodes. We can find that with a larger amounts of nodes, the diameter of the graph of be also larger and so is average distance.

3. PageRank

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

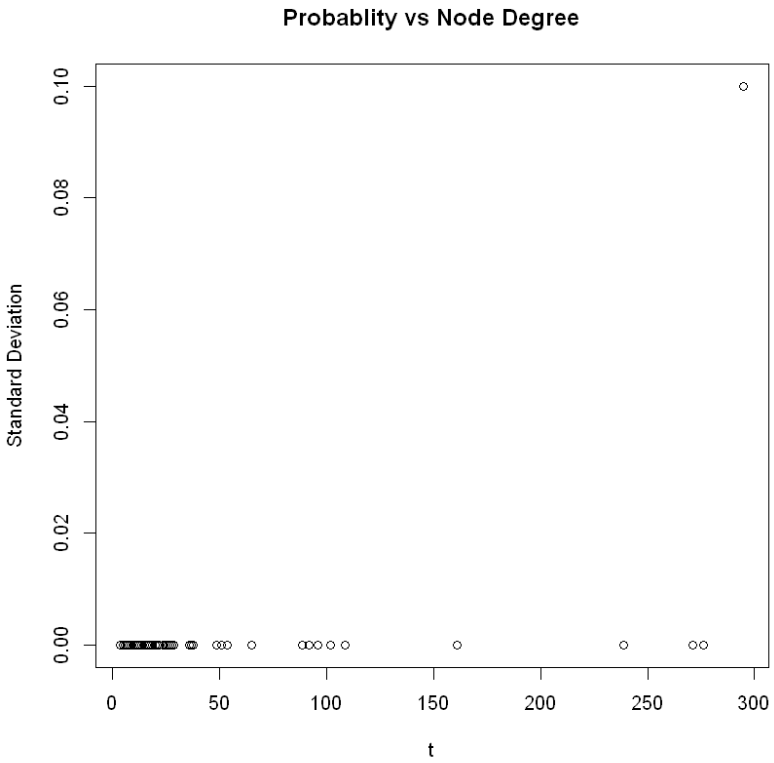


Figure 2.3.1

The figure 2.3.1 shows the probability that the walker visits each node vs degree. Every circle in the figure represents a node. We can find that the node with the largest degree also has the largest probability. This can be accounted that for preferential attachment network means that that the more connected a node is, the more likely it is to receive new links. And for Page Rank, a node will have a higher score if it has more incoming links. So, this generally means that a node with the larger degree will have a larger visit probability.

To have a more intuitive understand of the relationship between the probability and degree. We sum up the occurrences for each degree and got figure 2.3.2.

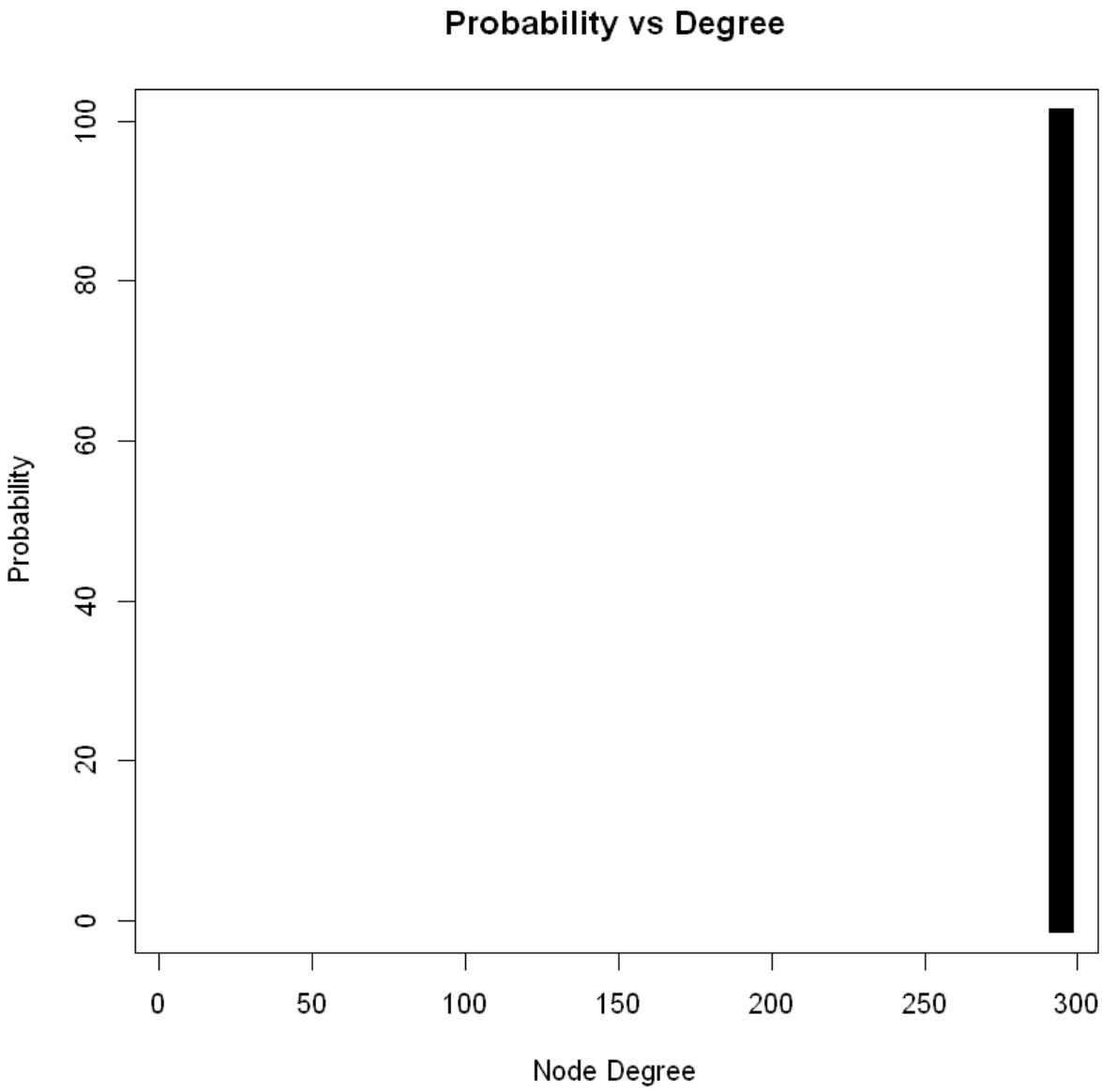


Figure 2.3.2

f) In all previous questions, we didn't have any teleportation. Now, we use a teleportation probability of α = 0.15. By performing random walks on the network created in 3(a), measure the probability that the walker visits each node. Is this probability related to the degree of the node?

In this part we add a teleportation probability into our walker function which means the transportation probability becomes following expression:

$P=\alpha E+(1-\alpha)P$

And similarly, we got the figure 2.3.3 to show the relationship between probability and node degree.

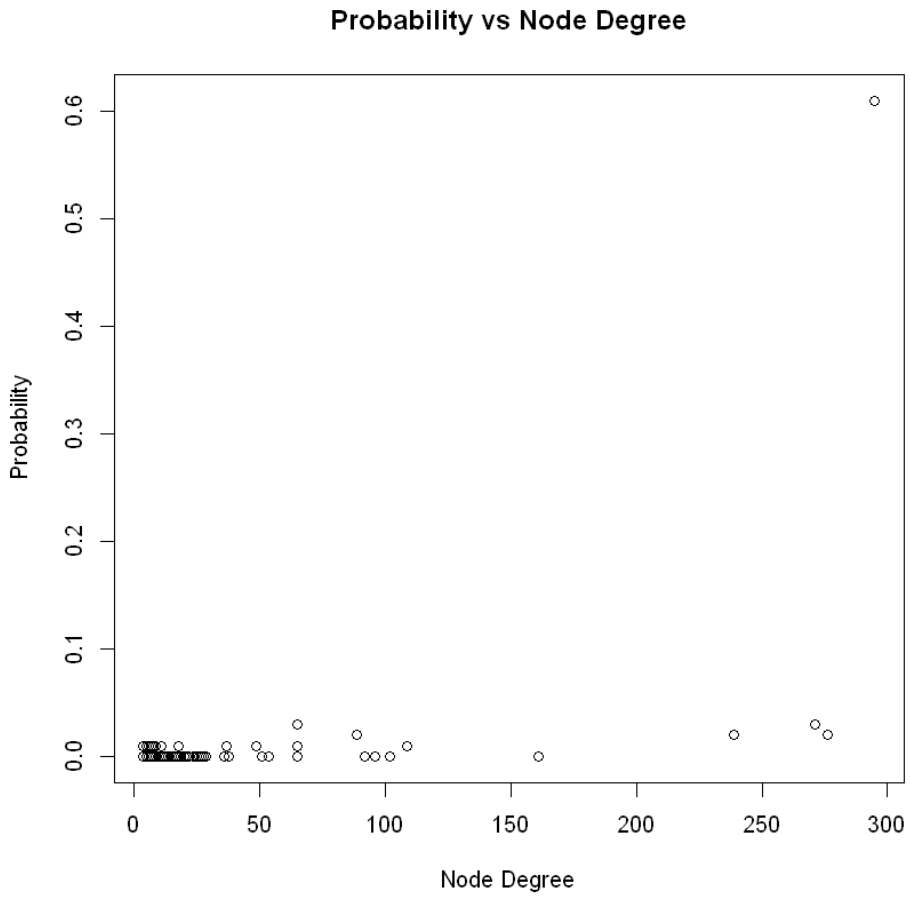


Figure 2.3.3

From figure 2.3.3 we can find that it is similar with figure 2.3.2 while the probability has changed. The largest probability has become around 60% instead of 100% here. This is because we add the teleportation probability, and this will cause a random jump from one node to another node. Instead of stopping at node 1 with the largest degree it can jump to some other nodes in its walk. And also, we plot the probability vs degree in figure 2.3.4.

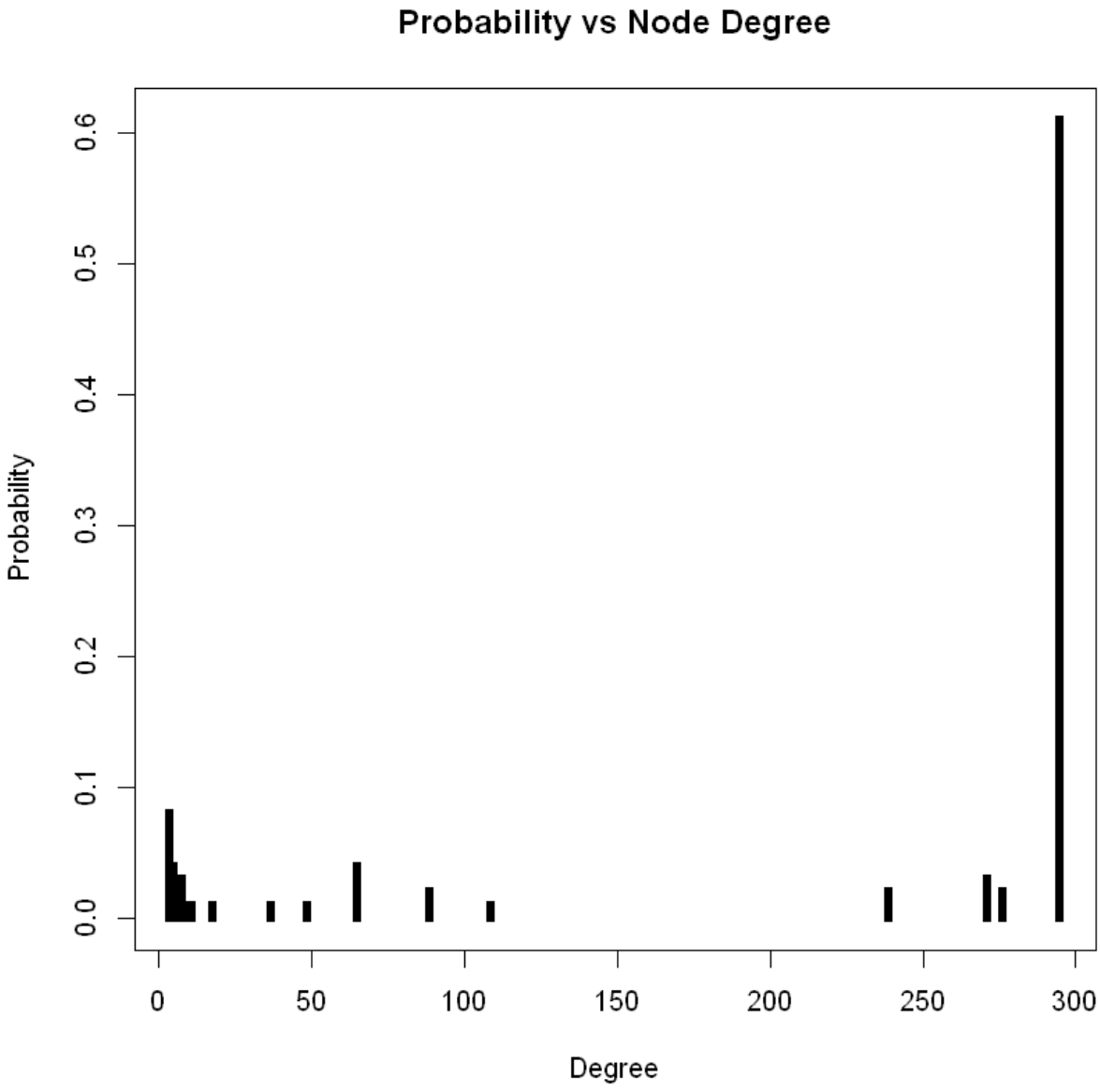


Figure 2.3.4

4. Personalized PageRank

a) Suppose you have your own notion of importance. Your interest in a node is proportional to the node's PageRank, because you totally rely upon Google to decide which website to visit (assume that these nodes represent websites). Again, use random walk on network generated in part 3 to simulate this personalized PageRank. Here the teleportation probability to each node is proportional to its PageRank (as opposed to the regular PageRank, where at teleportation, the chance of visiting all nodes are the same and equal to 1/N). Again, let the teleportation probability be equal to $\alpha = 0.15$. Compare the results with 3(b).

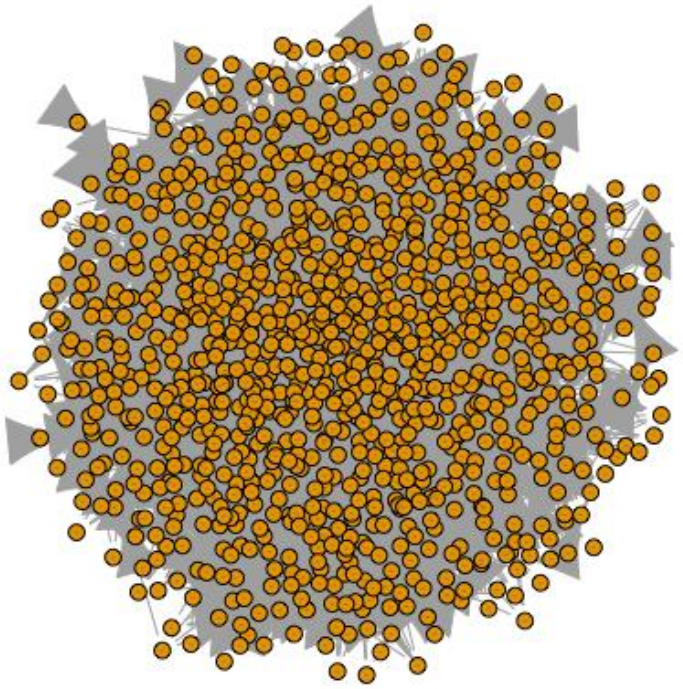


Figure2.4.1: 1000 nodes with m = 4

In this part, we are required to simulate a personalized PageRank using the random walk network generated in part 3. In detail, we will use the page_rank function to calculate the probability which set the teleportation probability is equal to 0.15. Then we get the following figure.

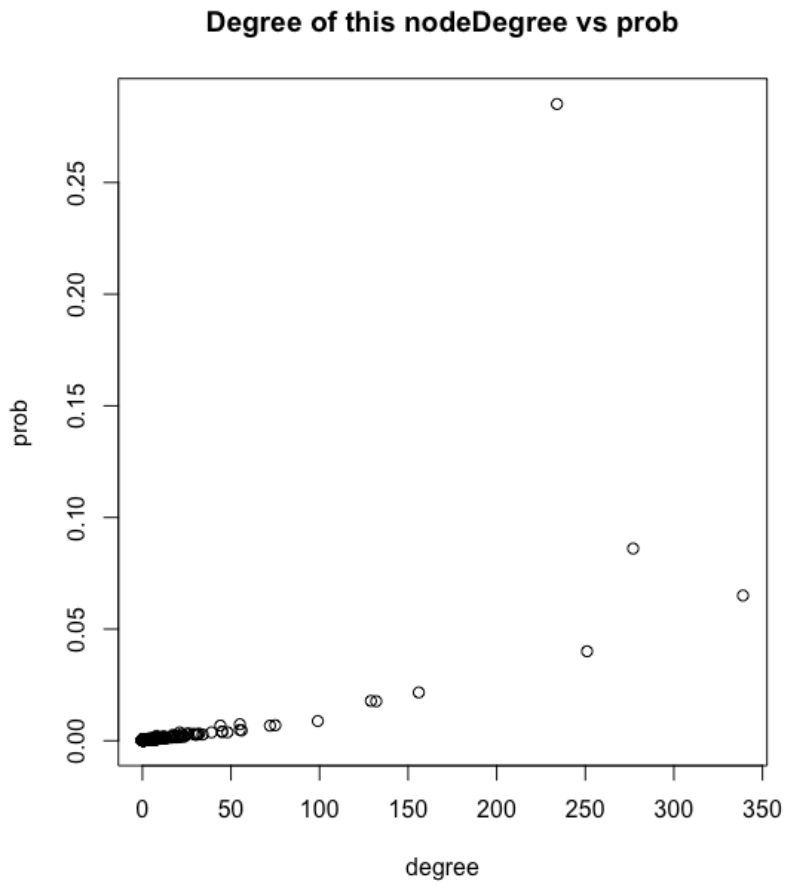
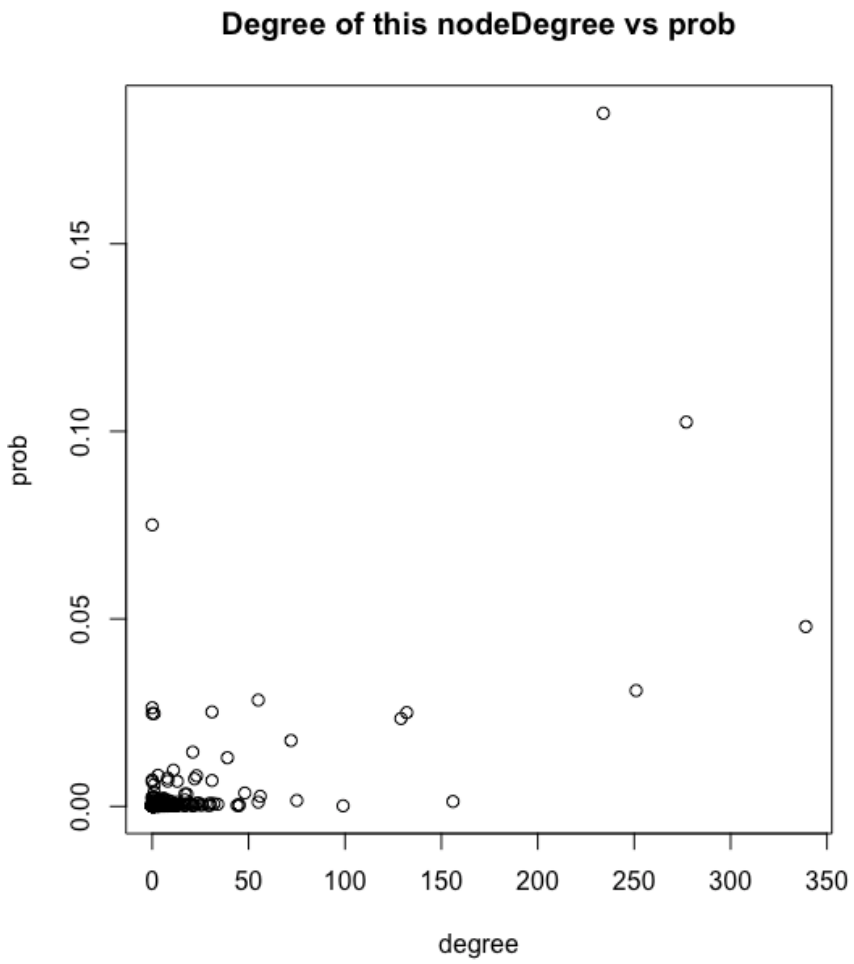


Figure2.4.1: nodeDegree VS prob

Compared the results with 3(b), we can see there is no big difference between two figures. The probability of two figures are all start from 0 and increase with the increase of degrees. Also, both figures all have couple nodes that the probability is higher than 0.005.

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

In this section, we will repeat part (a) but constraining the teleportions land only on those two nodes. The following is our result.



Compared the results with 3(b), we can see the figure above has more diffuse nodes especially in the degree between 0 and 150. In figure 2.31, we find the probability between 0 and 150 is almost close to 0, but in above figure there are lots of nodes spread out in the area which has a 0 ~ 0.05 probability. Otherwise, they are quite similar and both figures still have couple nodes the probability is higher than 0.05.

c) More or less, (c) is what happens in the real world, in that a user browsing the web only teleports to a set of trusted web pages. However, this is against the different assumption of normal PageRank, where we assume that people's interest in all nodes are the same. Can you take into account the effect of this self-reinforcement and adjust the PageRank equation?

The equation of normal PageRank is:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

As described in part (b), if people have different importance and interest to a node is proportional to the node's PageRank, we can change teleportation probability to each node. Then, we can get a new equation:

$$PR(p_i) = \frac{PR(p_i) - d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$