Network Science Analytics: Assignment 1

Question 1

a. We have an undirected graph, unweighted with no self-loops.
The vector **k** whose elements are the degree k_i of node i can be expressed as followed:

$$\boldsymbol{k = A1}$$

Where **A** is the adjacency matrix of the graph and **1** the column vector with only 1 as its elements.

Question 1

b. Let's note m the number of edges in the graph and n the number of nodes.
We know that, with the previous notations:

$$\sum_{i=1}^{n} k_i = 2m$$

So we have:

$$\boldsymbol{k}^T \mathbf{1} = 2m$$
$$m = \frac{\mathbf{1}^T \boldsymbol{A}^T \mathbf{1}}{2}$$

Question 1

c. The matrix **N** whose elements N_{ij} are the number of common neighbors between i and j is simply the matrix for which the $(i,j)^{th}$ element is the number of path of length 2 between nodes i and j.

The we have:

$$\boldsymbol{N = A^2}$$

Question 2

We have a bipartite network.

Let's call *l* the number of edges going from a node of type 1 to a node of type 2.
Obviously, we then have the number of edges going from a node of type 2 to a node of type 1 also equal to *l*.

As the network is bipartite, we have:

$$c_1 = \frac{2l}{n_1}$$

$$c_2 = \frac{2l}{n_2}$$

So,

$$l = \frac{n_2 c_2}{2} = \frac{n_1 c_1}{2}$$

Then:

$$c_2 = \frac{n_1}{n_2} c_1$$

Question 3

a. A triangle in a graph is a path of length 3 that starts at node i and finishes at node i.
Then, the number of triangles in a graph is given by the trace of the adjacency matrix power 3. (sum of all the paths of length 3 starting and finishing at the same node). We have to divide this number by 2 because the graph is undirected and then taking the triangle in a certain order of edges or in the other does not change anything. We also have to divide this number by 3 because the starting node can be any of the three nodes of the triangle.

Then :

$$\Delta(G) = \frac{tr(\boldsymbol{A}^3)}{6}$$

Question 3

b. The adjacency matrix is symmetric. It is then diagonalizable according to the spectral theorem.

We have

$$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^T$$
$$\boldsymbol{A}^3 = \boldsymbol{P}\boldsymbol{D}^3\boldsymbol{P}^T$$

With **D** a diagonal matrix with **A**'s eigenvalues on the diagonal and **P** an orthogonal matrix.
Then,

$$tr(\boldsymbol{A}^3) = tr(\boldsymbol{P}\boldsymbol{D}^3\boldsymbol{P}^T) = \boldsymbol{tr}(\boldsymbol{D}^3) = \sum_{i \in V} \lambda_i^3$$

Finally:

$$\Delta(G) = \frac{\sum_{i \in V} \lambda_i^3}{6}$$

Question 3

c. The number of triangles in which node i participates in is the number of paths of length 3 starting and finishing at i. It is then the (i,i)th element of the matrix $\mathbf{A}^3$, divided by two (as it is an undirected graph). Let's note this element A_{ii}^3.

If we note, for $i \in V$, λ_i the eigenvalues of $\mathbf{A}$ and $\boldsymbol{u_i}$ its eigenvectors associated with λ_i, we have by simple matrix multiplications:

$$A_{ii}^3 = \sum_{j \in V} \lambda_j \, u_{ji}^2$$

Where u_{ji} is the i^{th} element of the eigenvector u_j associated with the eigenvalue λ_j.

Finally:

$$\Delta_i = \frac{A_{ii}^3}{2} = \frac{\sum_{j \in V} \lambda_j \, u_{ji}^2}{2}$$

Question 4

We consider the random graph G_{np} with average degree c.

a. Let's compute the expected number of triangles in the graph for a fixed n.
As we are in the random graph model, having an edge between two nodes is of probability p and it is an independent event from the events consisting of having an edge between two other nodes. Let's call X_{ij} the random variable equal to 1 if there is an edge between nodes I and j, 0 otherwise.

Then, having a triangle in a graph involving nodes i, j and k corresponds to the event:

$$X_{ij} = 1 \cap X_{jk} = 1 \cap X_{ki} = 1$$

It is an intersection of three independent events, each of probability p.
Then, the probability of having a triangle in the graph is equal to p^3.

There are n nodes in the graph, so there are $\binom{n}{3}$ different possible triangles in the graph.

The expected number of triangles in a random graph with n nodes is then $\binom{n}{3}p^3$.

In the limit of large n, we have:

$$\binom{n}{3} \sim \frac{n^3}{3!}$$

And:

$$p^3 = \left(\frac{c}{n-1}\right)^3 \sim \frac{c^3}{n^3}$$

Then,

$$\binom{n}{3} p^3 \sim \frac{n^3}{3!}\frac{c^3}{n^3} = \frac{c^3}{6}$$

We have the desired result. In the limit of large n, the expected number of triangles in the graph is constant and equal to $\frac{c^3}{6}$.

Question 4

b. We use for this question exactly the same kind of reasoning as for the previous one. We compute the expected number of connected triplets in the graph for a fixed n.
Having a connected triplet in the graph involving nodes i, j and k, with an edge between i and j and between j and k, corresponds to the realization of the event:

$$X_{ij} = 1 \cap X_{jk} = 1$$

This event has a probability p^2.

In the random graph, we can have $3\binom{n}{3}$ connected triplets (the coefficient 3 comes from the fact that for 3 nodes, three different connected triplets exist).

The expected number of connected triplets in a random graph with n nodes is then $3\binom{n}{3}p^2$.

In the limit of large n, we have:

$$\binom{n}{3} \sim \frac{n^3}{3!}$$

And:

$$p^2 = \left(\frac{c}{n-1}\right)^2 \sim \frac{c^2}{n^2}$$

Then,

$$3\binom{n}{3}p^2 \sim 3\frac{n^3}{3!}\frac{c^2}{n^2} = \frac{1}{2}nc^2$$

We have shown that the expected number of connected triplets in the random graph grows with n and is equal to $\frac{1}{2}nc^2$.

Question 4

c. The clustering coefficient for the graph is then equal to, for a fixed n:

$$C = \frac{3\binom{n}{3}p^3}{3\binom{n}{3}p^2} = p$$

We know that $p = \frac{c}{n-1}$.

In the limit of large n, we then have:

$$C = p = \frac{c}{n-1} \sim \frac{c}{n}$$

This value is the same than the one found in class. The two reasonings lead to the same result.

Question 5

a. In terms of α and of the geodesic distance d_{ij} between pairs of nodes, we have:

$$x_i = 1 + \sum_{j \in V, j \neq i} \alpha^{d_{ij}}$$

Question 5

b. An algorithm to compute this centrally measure x_i for node i

- We start by computing the adjacency matrix **A** of the graph.
- We create an empty list already_viewed_nodes and initialize a variable sum to zero.
- Starting with k=1, we compute the power k of the matrix **A** and for each coefficient in the i^{th} column, if it is not zero and if the node corresponding to the line is not in the list already_viewed_nodes, we add α^k to our sum and the node corresponding to the line to the list already_viewed_nodes. We do this procedure again with k = k + 1 as long as each node is not in the list already_viewed_nodes.
- Finally, we just have to add 1 to this sum and we have obtained x_i.

The operation which is the costliest in this algorithm is the calculation of the matrix power, which is in $O(V^2)$. We have to compute matrix multiplications $\max_{j \in V, j \neq i} d_{ij}$ times (this maximum is the length of the path from i to the node the further away).

The complexity of calculating x_i for one node i is then $O\left(V^2 \max_{j \in V, j \neq i} d_{ij}\right)$.

The complexity of calculating x_i for all V nodes in the graph is then $O\left(V^3 \max_{i \in V}\left(\max_{j \in V, j \neq i} d_{ij}\right)\right)$.

Question 6

G is the entire graph and G_A and G_B are the two subgraphs connected by the edge (A,B).

We have:

$$C_A = \frac{n}{\sum_{j \in G} d_{Aj}}$$
$$C_B = \frac{n}{\sum_{j \in G} d_{Bj}}$$

Then,

$$n = C_A \sum_{j \in G} d_{Aj} = C_B \sum_{j \in G} d_{Bj}$$

$$C_A \left(\sum_{j \in G_A} (d_{Bj} - 1) + \sum_{j \in G_B} (d_{Bj} + 1) \right) = C_B \sum_{j \in G} d_{Bj}$$

$$C_A \left(\sum_{j \in G_A} d_{Bj} + \sum_{j \in G_B} d_{Bj} - n_A + n_B \right) = C_B \sum_{j \in G} d_{Bj}$$

$$C_A \sum_{j \in G} d_{Bj} + C_A(-n_A + n_B) = C_B \sum_{j \in G} d_{Bj}$$

$$C_A(n_B - n_A) = (C_B - C_A) \sum_{j \in G} d_{Bj}$$

$$\frac{C_A(n_B - n_A)}{C_B - C_A} = \frac{n}{C_B}$$

$$\frac{n_B - n_A}{n} = \frac{C_B - C_A}{C_A C_B}$$

$$\frac{n_B}{n} - \frac{n_A}{n} = \frac{1}{C_A} - \frac{1}{C_B}$$

Finally, we get the desired expression:

$$\frac{1}{C_A} + \frac{n_A}{n} = \frac{1}{C_B} + \frac{n_B}{n}$$

Question 7

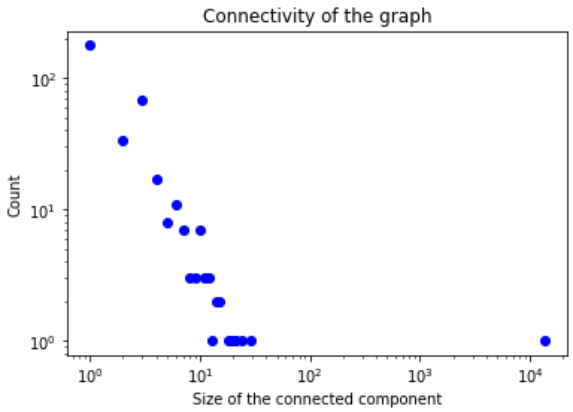
The code is available here (line 14 to 176): https://github.com/JeremiAssael/Network-Science-Analytics/blob/master/Assignment%201/assignment_1.py

a. The graph is stored as a directed graph, but because it is a collaboration network, we are going to use the undirected version of the network. We are using the NetworkX package.

(1) The undirected graph has 5242 nodes and 14496 edges.

(2) After checking with the appropriate method, the graph is not connected.

(i) There are 355 connected components.
(ii) Here is the distribution of the number of connected components over their sizes (number of edges). We observe one giant connected component at the bottom right of the plot.



(3)

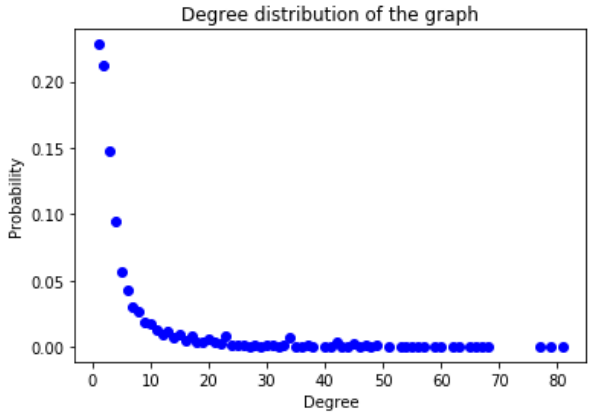
(i) The largest connected component has 4158 nodes and 13428 edges.
(ii) 79.3% of the nodes of the graph belong to the GCC and 92.6% of the edges of the graph belong to the GCC.

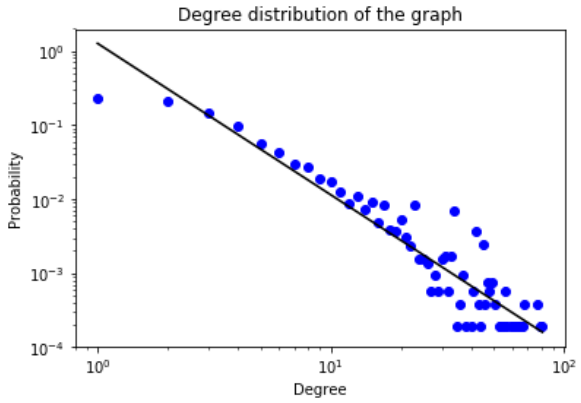
This ratio and the connectivity distribution show that a large part of researchers are linked in this collaboration network. There are small parts of the graphs which are not connected, which show the existence of small groups of researchers not linked to the others.

Question 7

b.
(1) After computations, we find that the minimum degree of the nodes of the graph is 1, the maximum is 81, the median is 3 and the mean is 5.53.
This allows us to understand that no researcher has published alone as the minimum degree is one: there is at least one co-author for each paper. At least one researcher has strong links in the community as it has collaborated with 81 other researchers. Half of the researchers have collaborated with at most 3 other people and the other half with at least 3 other people.

(2)





The first plot has cartesian axes and the second one has logarithmic axes.

As we can fit a straight line through the degree distribution on the logarithmic scale, we have a power-law degree distribution.
We can write that:

$$log\big(p(k)\big) = log(c) - \alpha\, log(k)$$

With p(k) the fraction of node with degree k.

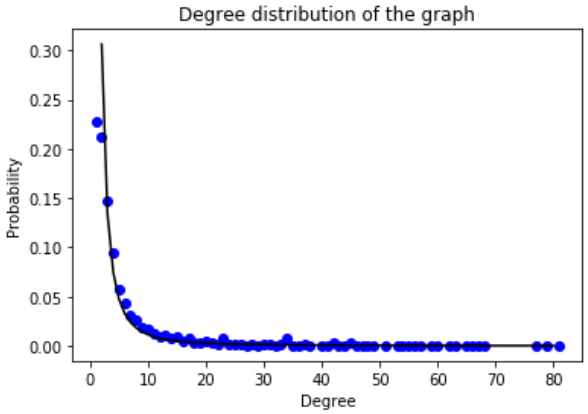
Using Python (Numpy), we found that:

$$log(c) = 0.23$$
$$c = 1.26$$
$$\alpha = 2.04$$

Then the power law is

$$p(k) = ck^{-\alpha} = 1.26k^{-2.04}$$

This power law fits well our observations, as we can see on the graph below (with the degree distribution in blue and the computed power law in black).

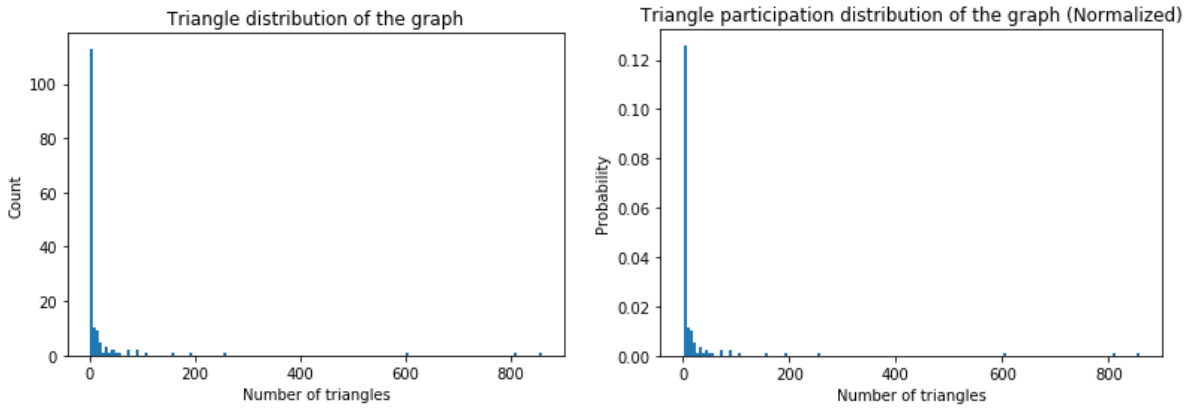


Question 7

c. We are only considering here the GCC of the graph.

(1) Using the triangles method of NetworkX, we find that there is a total of 47779 triangles in the GCC. (We divided the summed output of the method by three, as this method gives the number of triangles each node is involved in, thus counting three times each triangle).

(2)



Most nodes are involved in only a small number of triangles, suggesting that there are quite well-separated communities in the graph: each researcher have their small group of co-authors. The nodes involved in a high number of triangles suggest that the researchers associated to these nodes are working with a lot of different researchers, meaning they have a large community. This idea was also suggested by the degree distribution.

Question 7

d. We are still focusing on the GCC.
In question 3, we find that:

$$\Delta(G) = \frac{\sum_{i \in V} \lambda_i^3}{6}$$

With λ_i being the eigenvalues of the adjacency matrix.

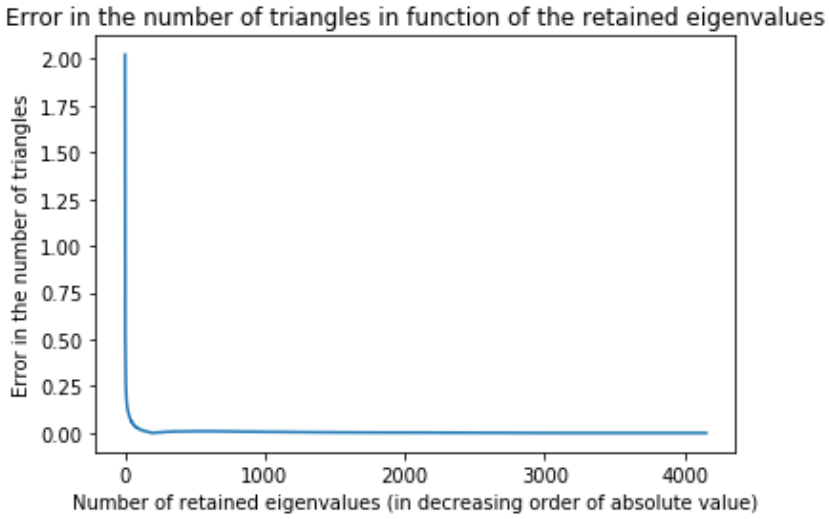
The computation of the eigenvalues can be the bottleneck as the numerical methods used to compute the eigenvalues are in $O(n^3)$ with n being the dimensions of the symmetric adjacency matrix. With a large matrix, it will take some time to compute all the eigenvalues. For me, it takes only a few seconds to compute the eigenvalues of the adjacency matrix of the GCC.

(1) We can approximate the number of triangles by computing it with only the largest eigenvalues (in absolute value) of the adjacency matrix because as the power-law degree distribution is decreasing and has a bigger density for the lowest degrees, the biggest eigenvalues (in absolute value) of the adjacency matrix are going to have the biggest weight in the sum of all eigenvalues. Indeed, the biggest eigenvalues represent the nodes with the highest degrees. This weight will even be bigger since we are going to take each eigenvalue power 3. The smallest eigenvalues (below 1), are going to have an even lower weight when we are going to take them power 3.

(2) In the case where all eigenvalues are used, we find, using the above formula, that there are 47808 triangles in the GCC. This number is a bit different (0.06% of difference) that the one obtained with the NetworkX method, which was 47779. I can't find the way triangles are computed in the NetworkX package and thus I cannot explain the difference. This difference being negligible, for the computation of errors in the following paragraph, I am going to use 47808 as the total number of triangles.

We can plot the error in the number of triangles against the number of eigenvalues retained in the adjacency matrix. We, of course, are going to retain the eigenvalues in decreasing order (in absolute value).

We get the following graph:



We can see that the error decreased really fast when we increase the number of considered eigenvalues. Let's recall that there are 4158 eigenvalues in total.
If we consider that a reasonable approximation can be obtained with an error of 10%, we just have to retain the 27 biggest eigenvalues, in absolute value (error of 10.2%).
If we consider that a reasonable approximation can be obtained with an error of 5%, we just have to retain the 57 biggest eigenvalues, in absolute value (error of 5.1%).
If we consider that a reasonable approximation can be obtained with an error of 1%, we just have to retain the 142 biggest eigenvalues, in absolute value (error of 1.0%).

Question 8

Generating the random graph with n=1000 and p=0.009, we get a graph with 1000 nodes and 4520 edges. The code is available here (line 180 to 223): https://github.com/JeremiAssael/Network-Science-Analytics/blob/master/Assignment%201/assignment_1.py

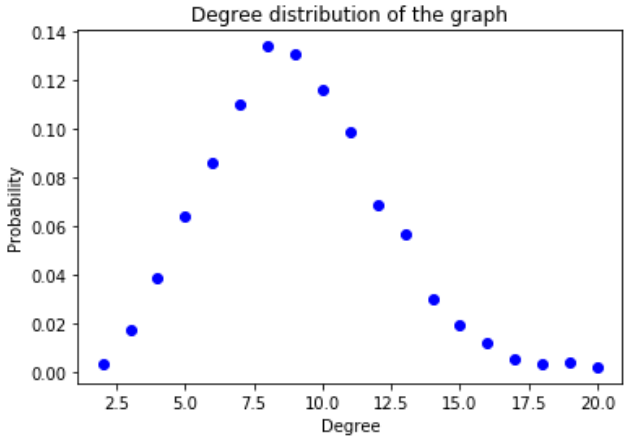
a. The mean degree of the graph is 9.04.
In the random graph model, the expected mean degree is equal to $(n-1)p$, so to 8.991. We found a mean degree consistent with the theory. This theory is based on the fact that the probability that the graph have exactly m edges follows a Binomial distribution.

Question 8

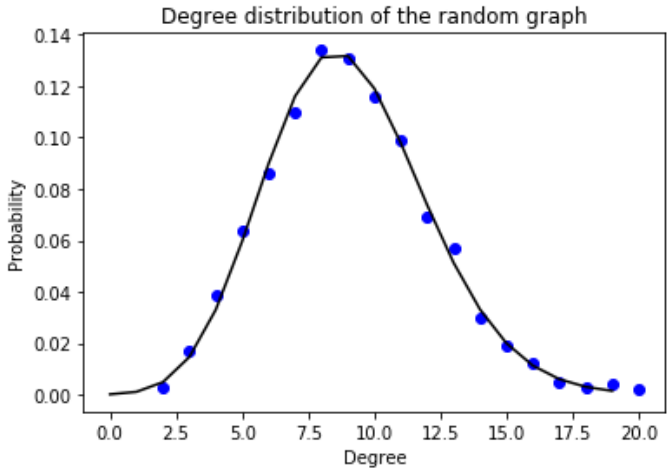
b. With the appropriate method of the NetworkX package, we find that the graph is connected.
The probability to have isolated nodes and then a graph with is not connected is inferior or equal to ne^{-c}, c being the mean degree of the graph. Here, this quantity is equal to 0.12. It was then likely to get a connected graph (although it would also have been possible to get a non-connected one).

Question 8

c. Here is the degree distribution of the graph, which has a mean degree of 9.04.



The degree of a node effectively tends toward a Poisson distribution. With parameter c=9.04, we have a good fit of this Poisson distribution to the real degree distribution, the Poisson distribution being in black.



The peak of this degree distribution is around the mean degree, which is logic in a random graph model.

Question 9

The code is available here (line 228 to 457): https://github.com/JeremiAssael/Network-Science-Analytics/blob/master/Assignment%201/assignment_1.py

a. We note the initiator matrix:

$$A_1 = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

With a = 0.99, b = 0.26, c = 0.53.

To answer to the two following questions, we are going to use the theorems on Stochastic Kronecker Graphs generated using this special matrix form.

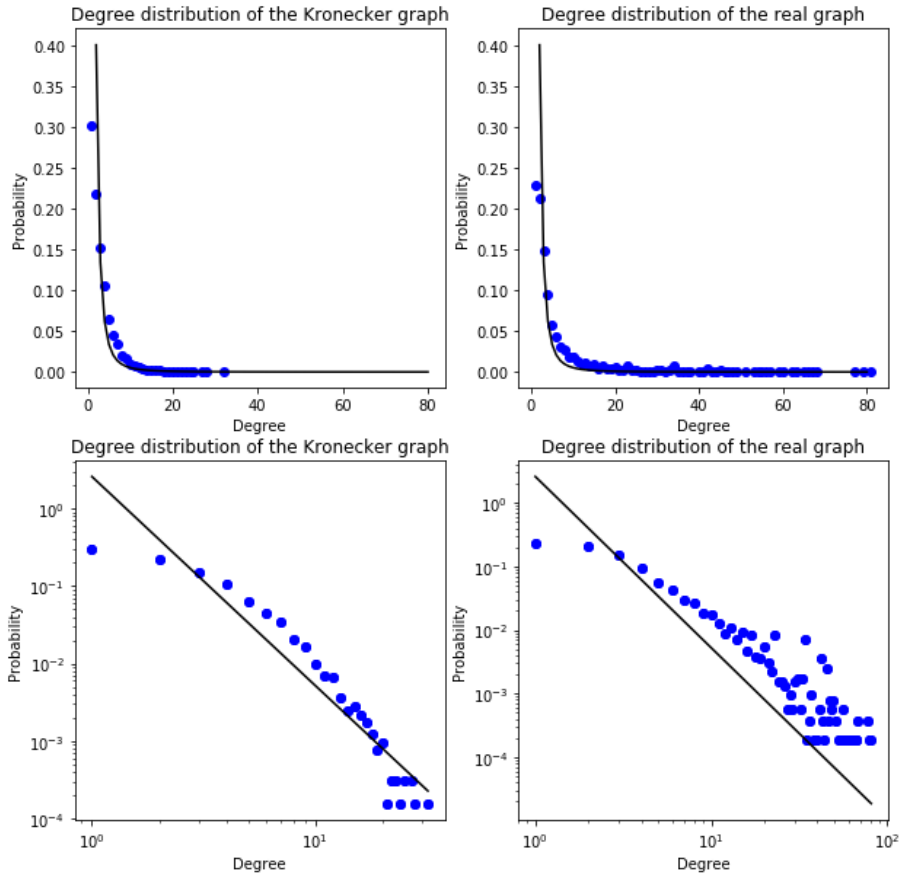
(i) Because b + c = 0.79 < 1, the obtained graph is not connected. We can easily check this with Python for different values of k (different iterations of the Kronecker product).

(ii) Because (a + b)(b + c) = 0.9875 < 1, the graph does not have a giant connected component of size Θ_n. Again, this propriety can also be easily checked with Python for different values of k. For instance, a trial with k=12 gives a GCC with only 5881 nodes and 10274 edges, whereas the dimensions of Θ_{12} are 8192×8192.

Question 9

b. The Kronecker graph has been computed for k=12.
It has 6403 nodes and 10573 edges. The graph has been generated with the methods describe in the assignment, using a Bernoulli random variable with p = $(\Theta_{12})_{i,j}$ to generate each edge (if the random variable is equal to 1, the edge is generated, otherwise, it is not).

(i) Let's first compare the power-law distribution of the two graphs.



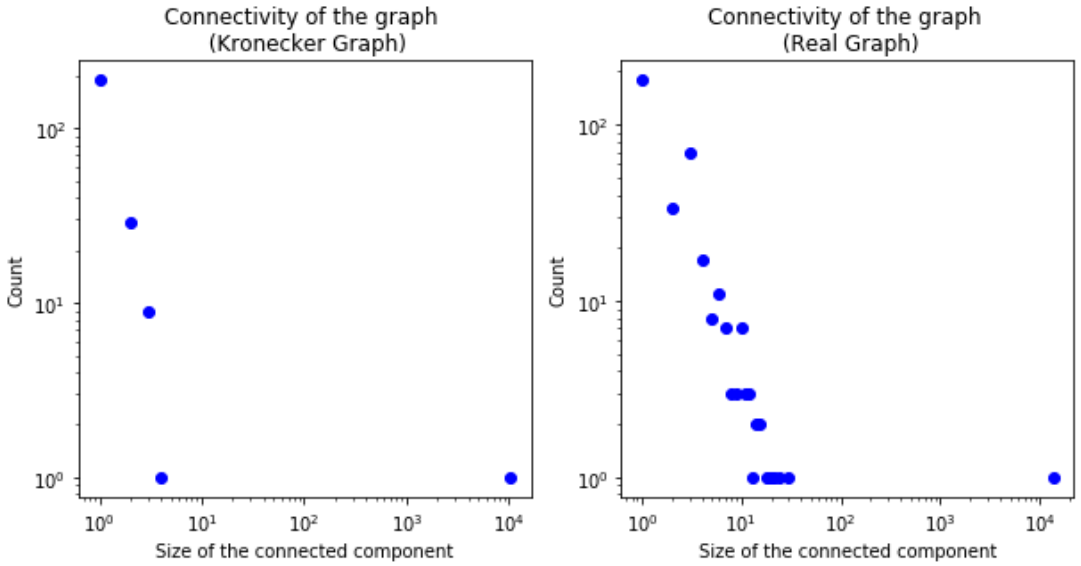
We fit the line on the logarithmic-scaled plot of the degree distribution of the Kronecker case. We then compute the parameters to obtain the power-law using the same technique as in Question 7. We then get:

$$p(k) = ck^{-\alpha} = 2.60k^{-2.70}$$

Of course, the parameters are not the same than the real network ones but, as we draw the straight line obtained on the logarithmic-scaled Kronecker graph's degree distribution on the logarithmic-scaled real graph's degree distribution, we observe that the trend is very similar. Also, plotting the distribution $p(k)$ obtained on the Kronecker graph on the degree distribution of the real one, we also observe that it fits really well. These two drawings have been done on the plots above.
In the real graph, there is also more nodes with a high degree, whereas in the generated graph the higher degree is 32. This is due to the fact that we have more nodes and less edges in the Kronecker graph than in the real one.

The degree distribution of the two graphs is then really similar (in trend, if we compute some metrics they are obviously different (mean degree, median…)).

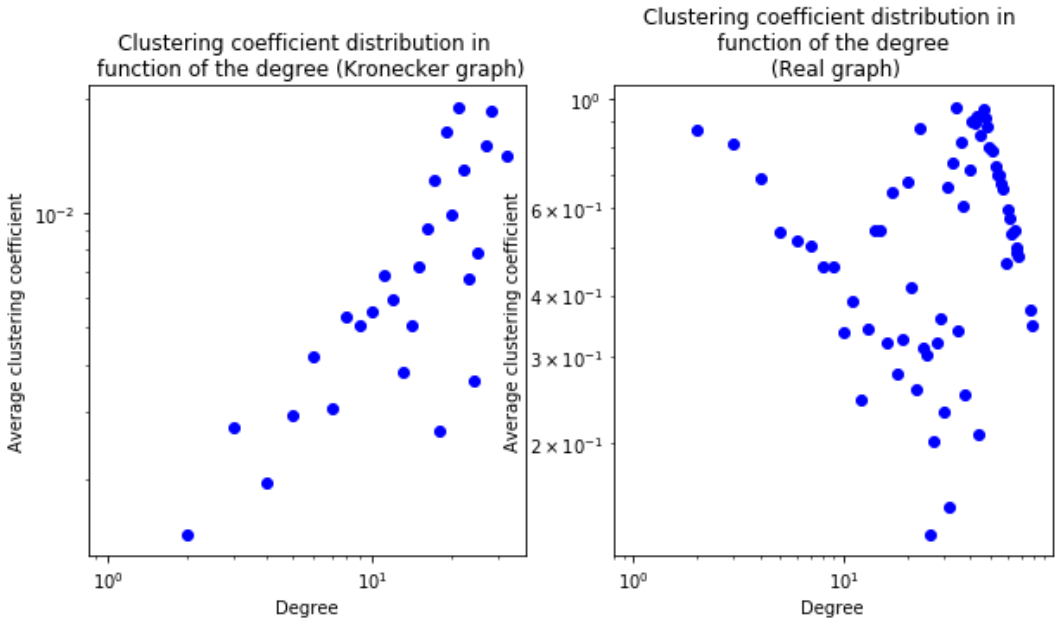
(ii) Let's now look at the connected components of the two graphs.
In the generated Kronecker graph, we have a smaller number of connected components (228 CC against 355 for the real graph). This is mainly due to the fact that there is a smaller number of edges in the generated Kronecker graph. (Absolute values are however not really interesting as they depend on the number of iterations of the Kronecker product).
For the Kronecker graph, 92,4% of the nodes belong to the GCC and 97.4% of the edges. This is much higher than for the real graph where 79.3% of the nodes of the graph belonged to the GCC and 92.6% of the edges of the graph belonged to the GCC.



However, as we can see on the above plots, the distributions of the number of connected components in the Kronecker and real graphs are very similar. We have exactly the same trend, with one giant connected component for both networks and then several much smaller connecter components around this GCC. We even have for the two graphs a peak for the connected components with only one edge, which reach approximately the same count of CC. It is another clue that the trend of both distributions is the same.

The distribution of connected components of the two graphs is very similar even if some metrics like the proportion of the GCC relative to the whole graph can be different.

(iii) Finally, let's study the clustering coefficient.



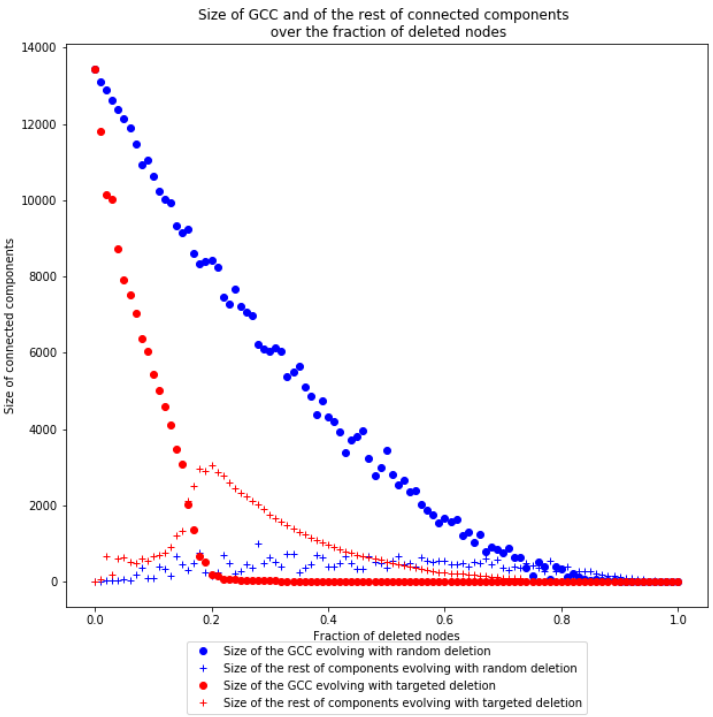
The distribution of the mean clustering coefficient of a node over the degree of the node appears to be totally different in the two graphs. The Kronecker graph has clustering coefficients which are much lower than the real graph (this is quite logical as they are less edges and more nodes). The average clustering coefficient also confirm this idea: it is 0.001885 for the Kronecker graph and 0.529636 for the real one.
This means that the nodes in the real graph have a higher tendency to cluster together than in the generated graph. Relative to the clustering coefficient, the generated graph and the real ones have different properties.

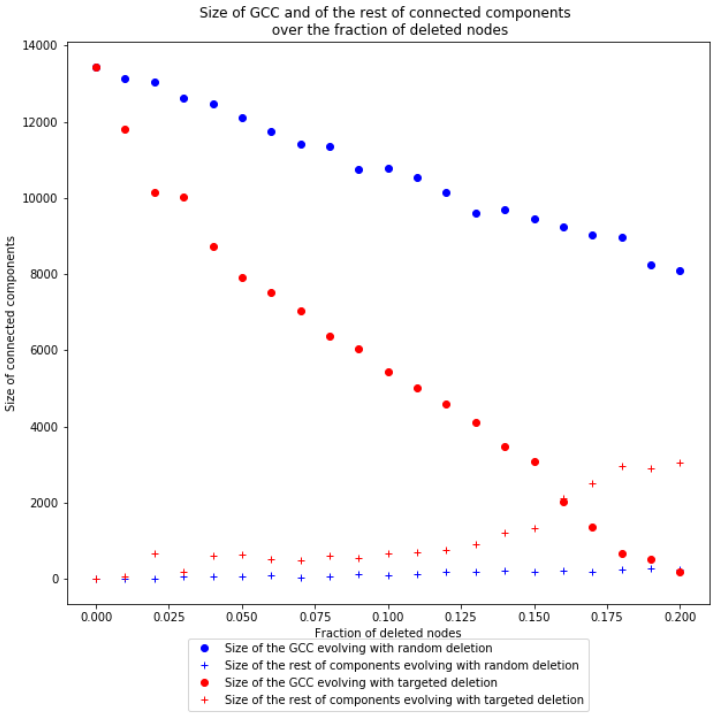
In conclusion, the degree distribution of the generated graph is a good representation of the real one. However, the clusters and connected components of the Kronecker graph does not really represents the real properties of the graph, as the tendency to cluster is totally underestimate (even if we can still spot their existence and their approximative distribution with the generated graph). We could have study other properties for which the Kronecker graph would have better represent the real one, like the node triangle participation.

Question 10

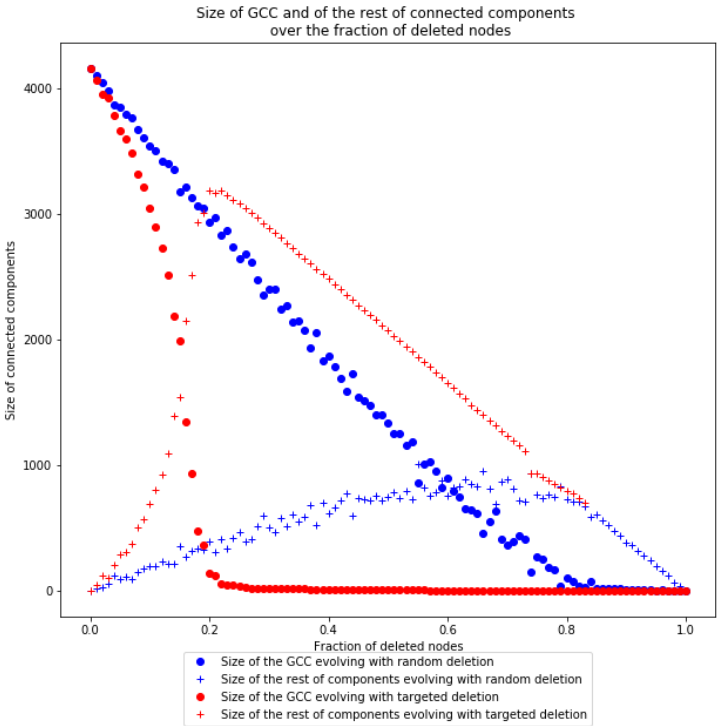
The code is available here (line 462 to 586): https://github.com/JeremiAssael/Network-Science-Analytics/blob/master/Assignment%201/assignment_1.py

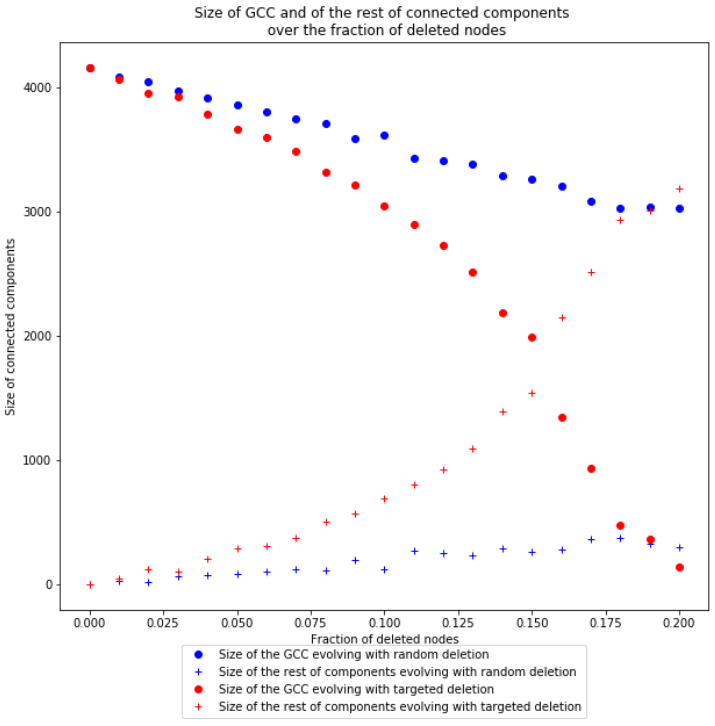
Thanks to the code available on the GitHub link, the following plots were generated, the second one being a zoom on the x-axis range of 0 to 0.2. These two graphs are obtained considering the size is the number of edges.





As some researcher also considered the size being the number of nodes, we plot the two following graphs, in which we don't consider the size of a CC as its number of edges but as its number of nodes. They are very similar to the above ones in trends, as expected.





We can indeed notice that our real network is more robust to random node deletion than to targeted node deletion. The decreasing of the GCC size for the case of random deletion is almost linear whereas in the case of targeted deletion, it looks like a power law, $\alpha\, r^{-k}$, with a very high k. In the targeted deletion case, for a rate of deletion near 18%, we already have the total size of the small CC which is above the size of the GCC whereas it only happened for a rate of deletion near 60% in the case of random deletion.

The random deletion case preserves a connected network for small deletion rate, which is not the case with targeted deletion. We can then effectively conclude that our real network tends to be robust under random removal of nodes and vulnerable under the attacks to high degree nodes.