Models of Complex Networks

Continuous assessment activity #2

Lorenzo Venieri - April 2022

General short explanation:

I implemented functions to generate all the studied network models for different parameters. The decisions taken in the implementations will be discussed later for the respective models.

For the small networks (N<1000) I plotted a view of the network (with nx.draw(G)) just to visualize what I've generated, together with a histogram of its degree distribution. For the large networks (N>=1000) I just plotted the histogram. For the networks with a degree distribution that follows a power-law I plotted also the histogram in log-log scale, because it's the only way to see the tail of the distribution.

To estimate the exponent of the power law distribution of degrees of the networks with this property (BA and CM_SF) I used two methods: linear regression of log(P(k)) as a function of log(k), and Maximum Likelihood Estimation, following what is explained in A2-Details-Exponent.pdf.

Regarding the linear regression, when p[b]=0 (the probability of a node having degree between x[b-1] and x[b], that are the edges of the b-th bin) it was not possible to take the logarithm of p[b], so I discarded the couples (b,p[b]), with bin b and probability p[b], whenever p[b]=0 for the fitting of the exponent.

Regarding the MLE, I observed that the output was usually not really satisfying, so based on what I read in A.Clauset, C. R. Shalizi and M. E. J. Newman: "Power-law distributions in empirical data": "...in our own experiments we have found it to give results accurate to about 1% or better provided xmin≥6.", I've tried to use MLE filtering out all the nodes with degree<6 to do the estimation with kmin=6, and the quality of the estimation improved considerably.

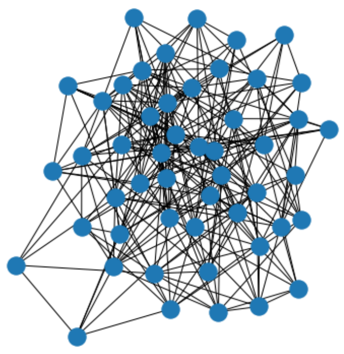
Lastly, just a comment about the notation used to store the networks: I saved all the networks and the relative images with the notation: “Name_N_parameter1_parameter2_etc” e.g “BA_10000_2_5” for a Barabási-Albert network with 10000 nodes, m=2 and number of initial nodes=5.

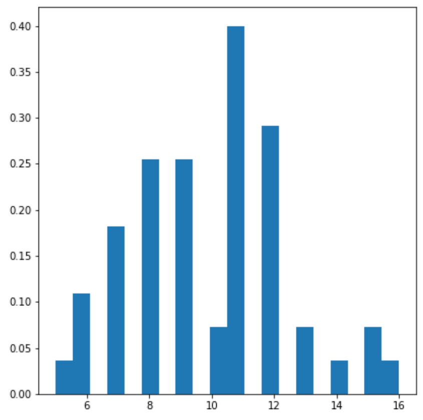
The networks in the delivered files are in pajek format.

Erdös-Rényi (ER) networks:

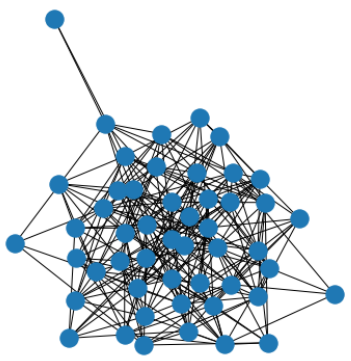
I implemented both the G(N,K) model by Erdős & Rényi and the G(N,p) model by Gilbert.

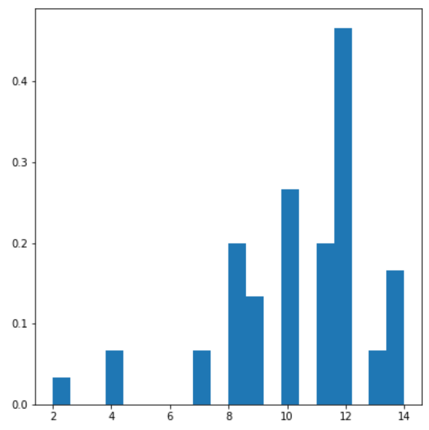
ER model with N=50 and <k>=10, G(N,K) implementation



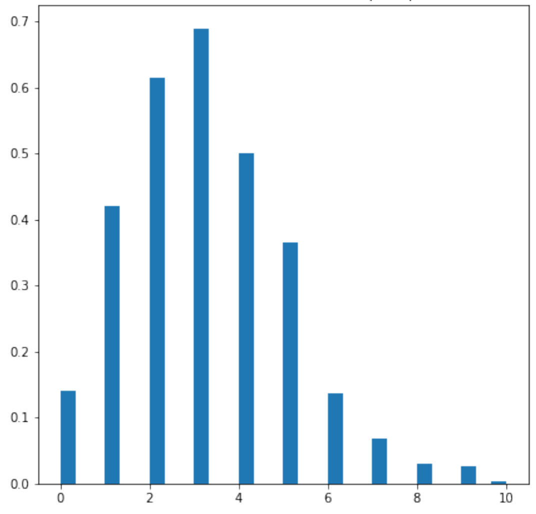


ER model with N=50 and <k>=10, G(N,p) implementation

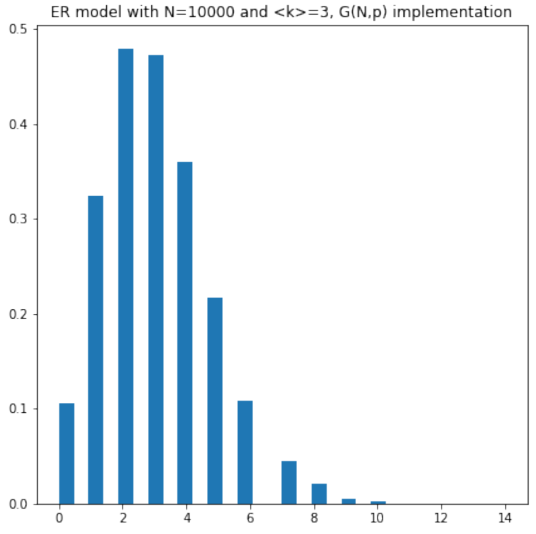




ER model with N=1000 and <k>=3, G(N,p) implementation

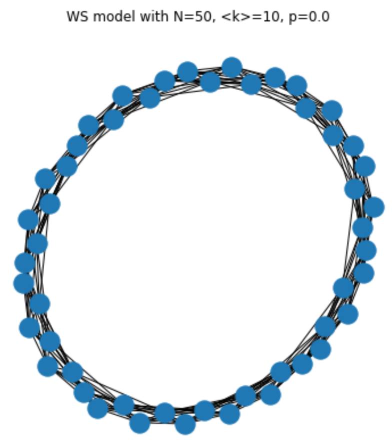


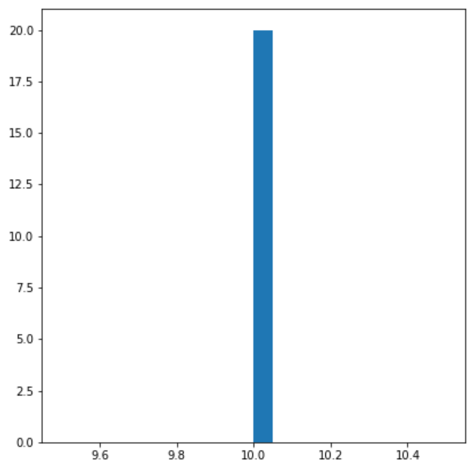
ER model with N=10000 and <k>=3, G(N,p) implementation



As seen in the theory, we can see that the degree distribution for the graphs with N>=1000 follows a Poisson distribution.

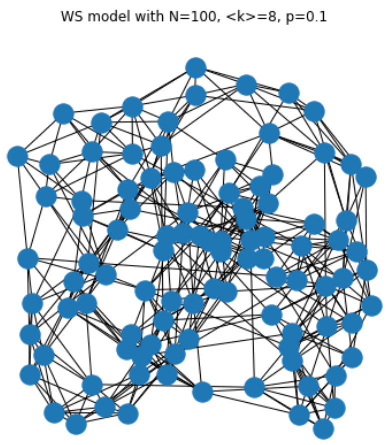
Watts-Strogatz (WS) model:

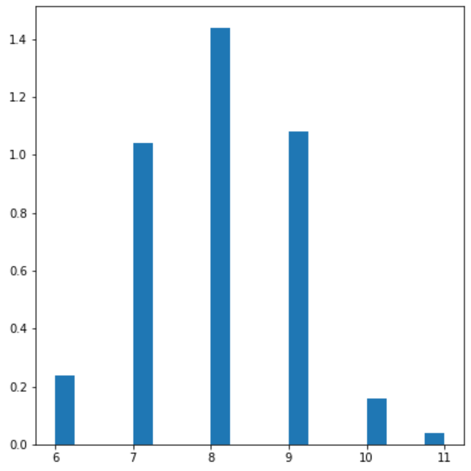




With p=0 the network is a perfect regular ring lattice, with all the nodes with k=<k>= 10.

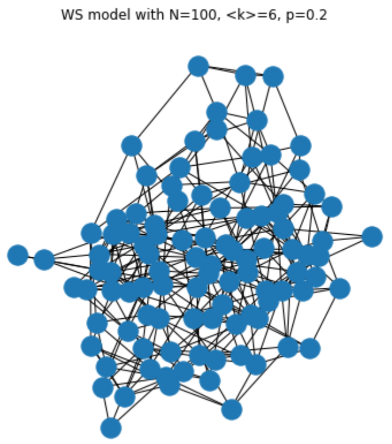
average path length and clustering: 2.959, 0.667

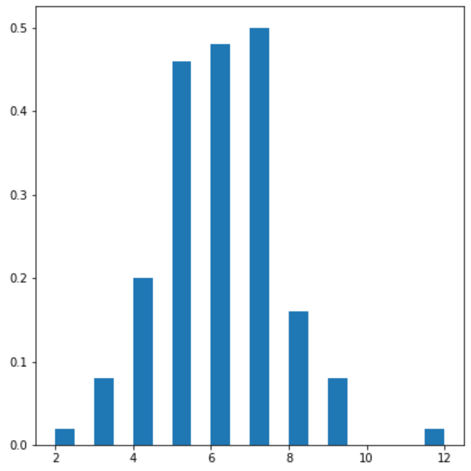




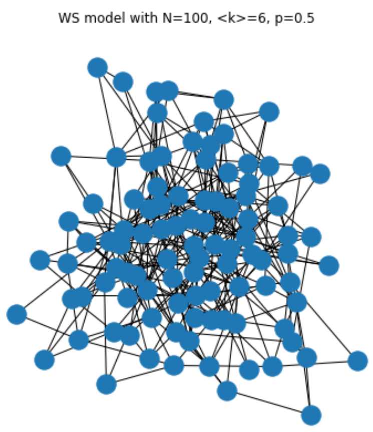
Average path length and clustering: 2.819, 0.411

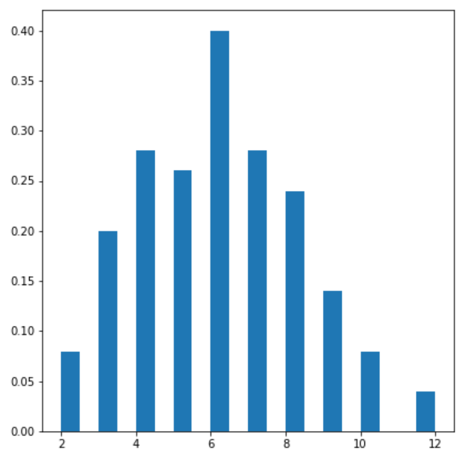
As seen in the theory: as p increases the average path length and clustering get smaller.



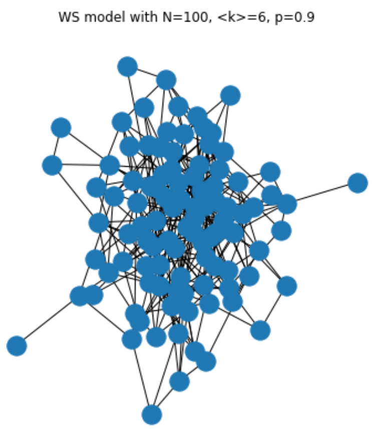


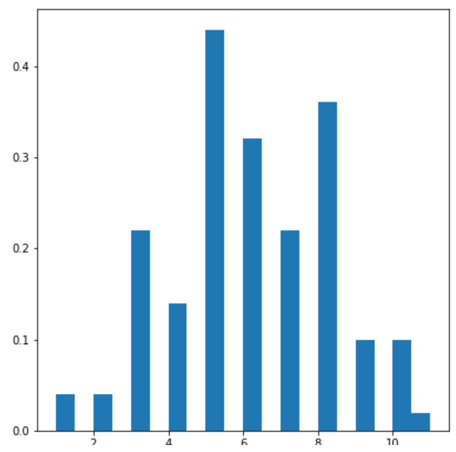
Average path length and clustering: 2.870, 0.177



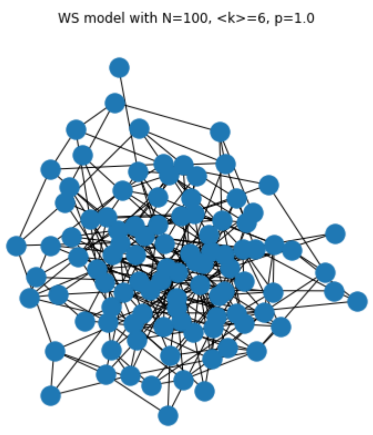


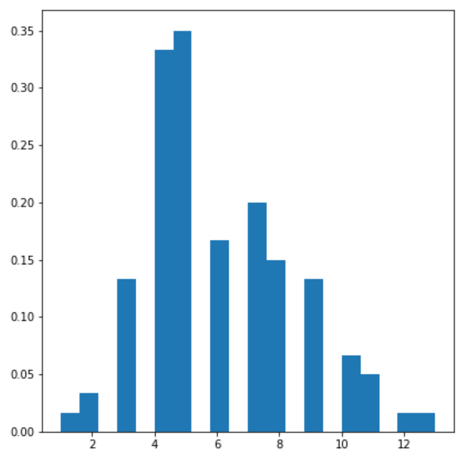
Average path length and clustering: 2.741, 0.087





Average path length and clustering: 2.759, 0.055



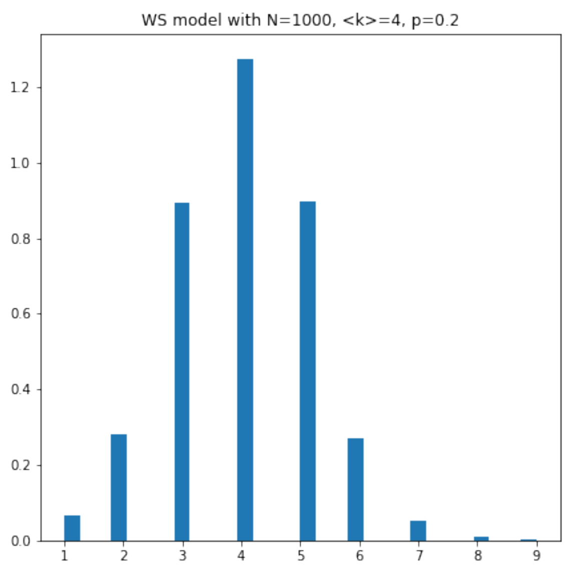


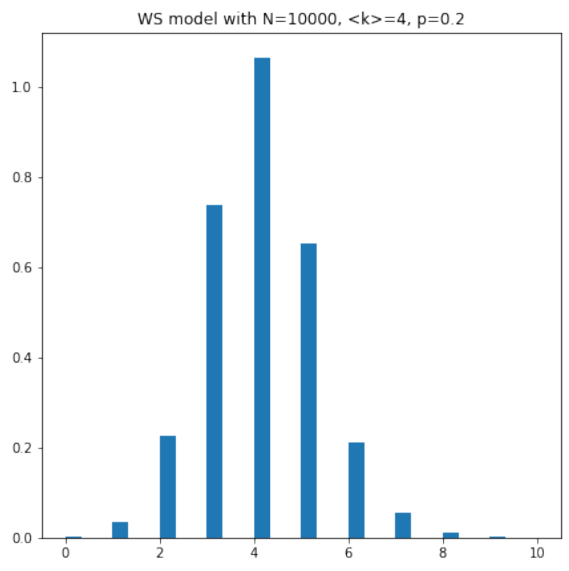
Average path length and clustering: 2.729, 0.060

WS_10000_4_02:

Plots of the big network:

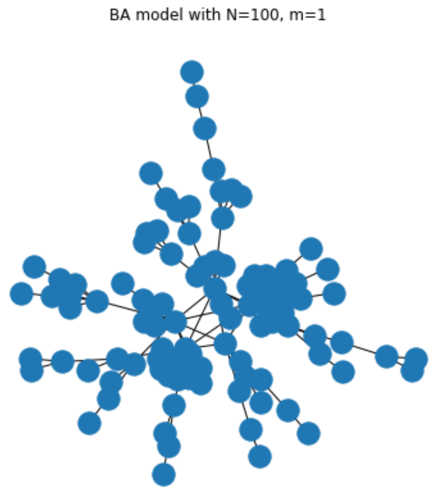
WS_1000_4_02: average path length and clustering: 5.910, 0.150

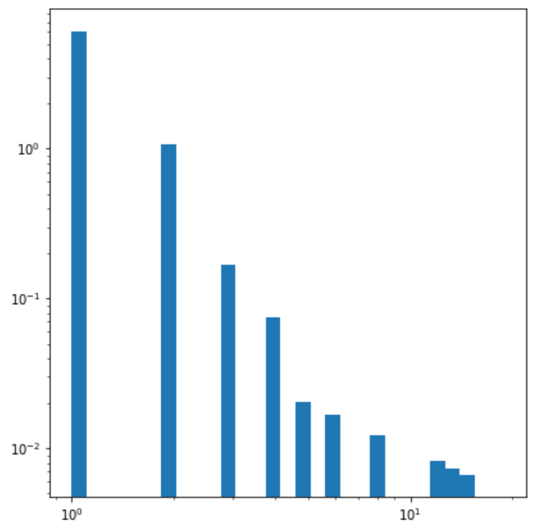


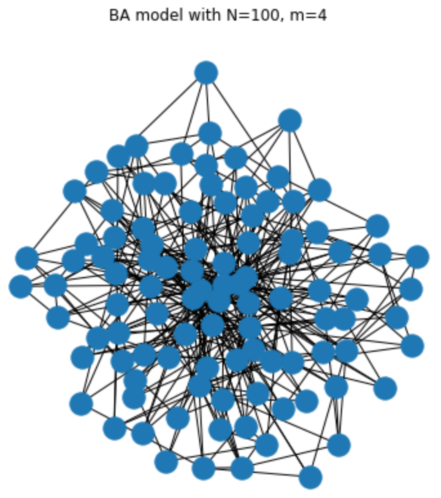


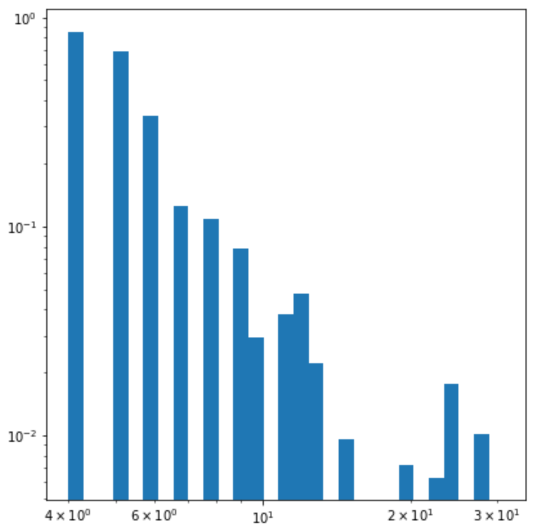
Barabási & Albert (BA) preferential attachment model

A comment about the implementation: to generate the initial clique I used the function to generate ER models with p=1.

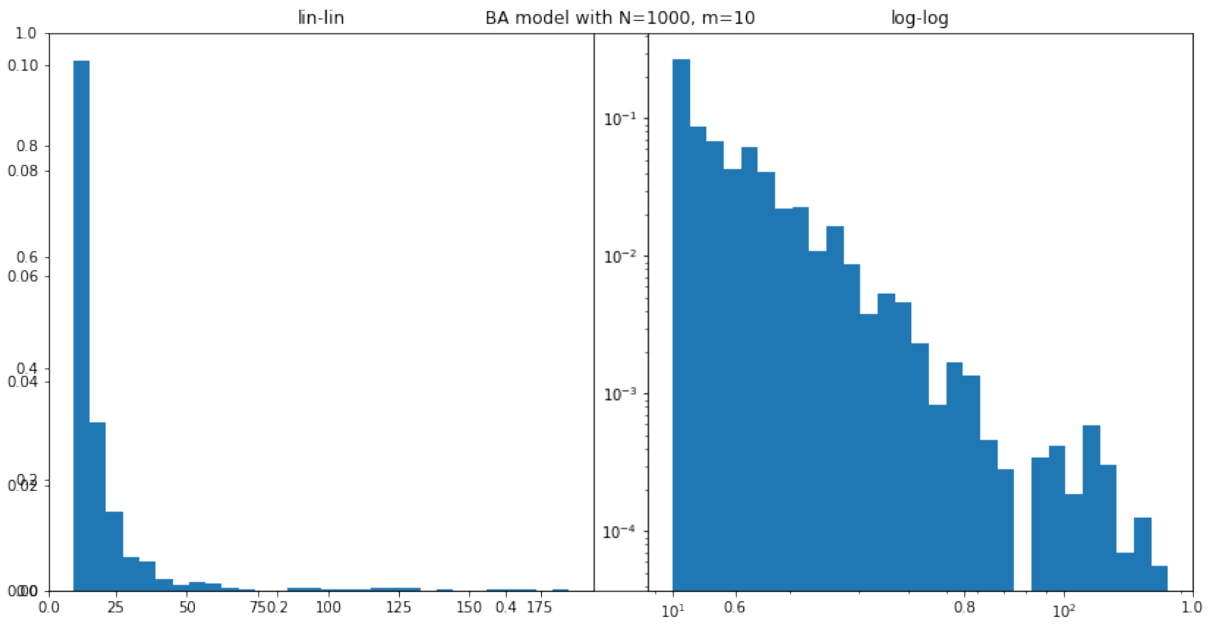




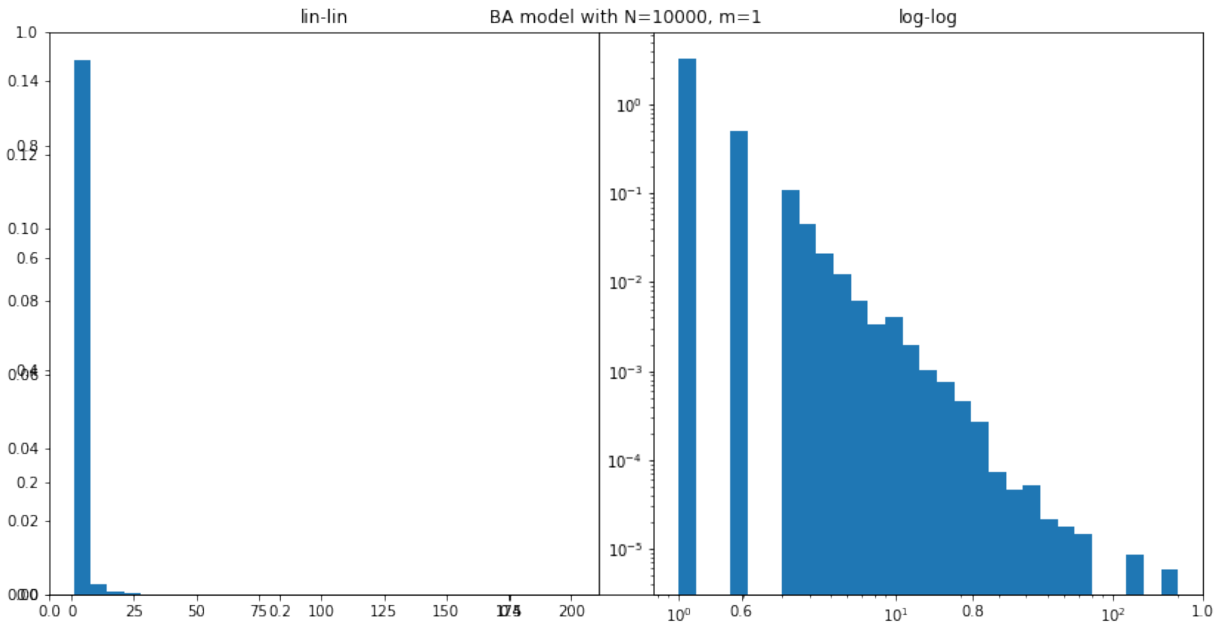




Plots of the big networks:



BA networks have degree distribution following a power-law with exponent gamma = 3.
MLE gamma estimation, with k_min>=1: 2.836298844094058
MLE gamma estimation, with k_min>5: 2.836298844094058
regr coefficient: -1.7842647789974184 , gamma = 2.7842647789974184



BA networks have degree distribution following a power-law with exponent gamma = 3.
MLE gamma estimation, with k_min>=1: 1.9368897583500508
MLE gamma estimation, with k_min>5: 2.8270433828536565
regr coefficient: -1.7078800986596399 , gamma = 2.70788009865964

Other examples with the relative gamma estimations can be found in the notebook (source code) and in the Img/BA/ folder.

Configuration Model (CM):

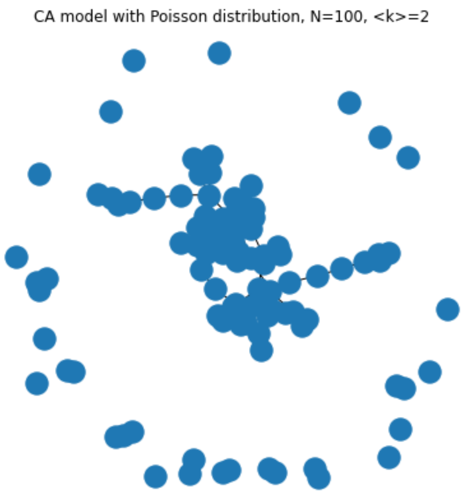
The function to build a CM model takes as input a given distribution, but the sum of the degrees of its nodes must be even. To construct this distribution I wrote two functions: Poisson_even and SF_even.

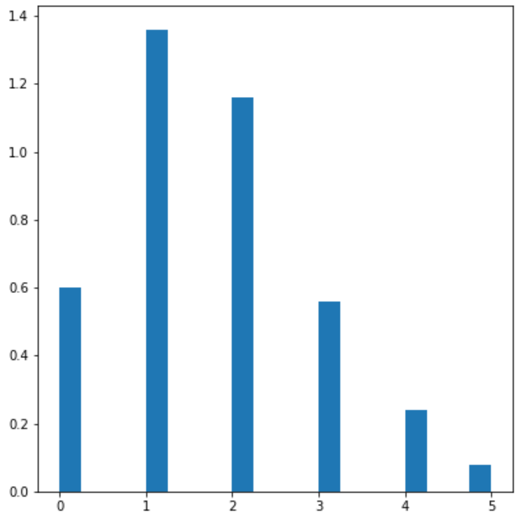
My implementation of CM follows these steps:

1. Generate the graph and its nodes
2. Construct the list of stubs (half edges)
3. Shuffle the list with the fisher_yates function I implemented
4. Loop over the list of stubs to build the edges, to avoid self loops and multi-edges we store them in a list (bad_edges) instead of adding it as edges of the graph
5. We reshuffle the list bad_edges trying to obtain valid edges to add to the graph. We add the valid edges found to the edges of the graph
6. Repeat point 5 until the list bad_edges is empty or a given maximum number of reshuffles has been done.

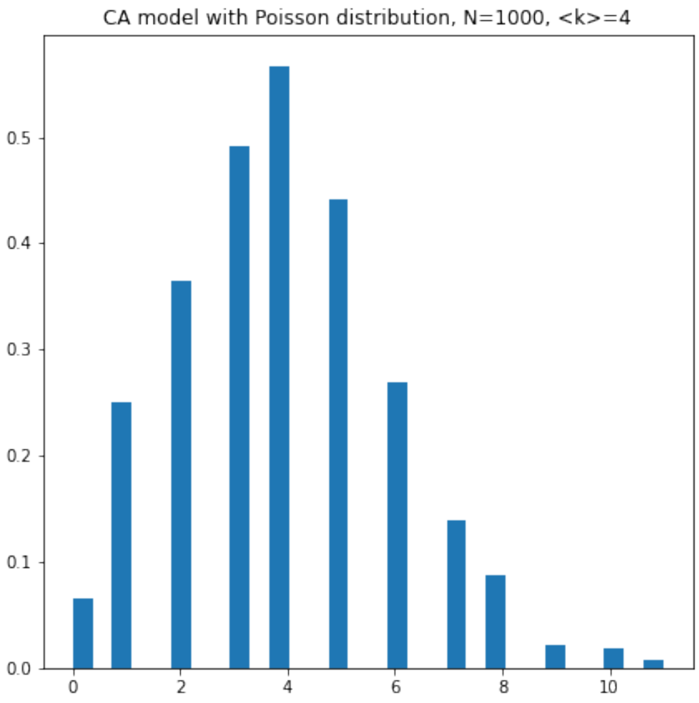
A concern:
I'm not sure that the implementation of CM I proposed is uniformly sampling all the networks without multi-edges or self-loops that have the desired degree distribution: repeating the permutation on the "bad edges" is increasing the probability to sample the networks that have the other edges fixed as given by the first permutation, but without multi-edges or self-loops.
To fix this I tried to implement CM (the function CM_test in the notebook) restarting to build the network from scratch as soon as a not eligible edge is found, until we have a list of edges without self-loops or multi-edges. This way we are sure to sample uniformly from the networks of the desired configuration and without self-loops or multi-edges. The problem is that unless the network is very sparse this algorithm has to generate a huge number of list of edges before finding a good one.

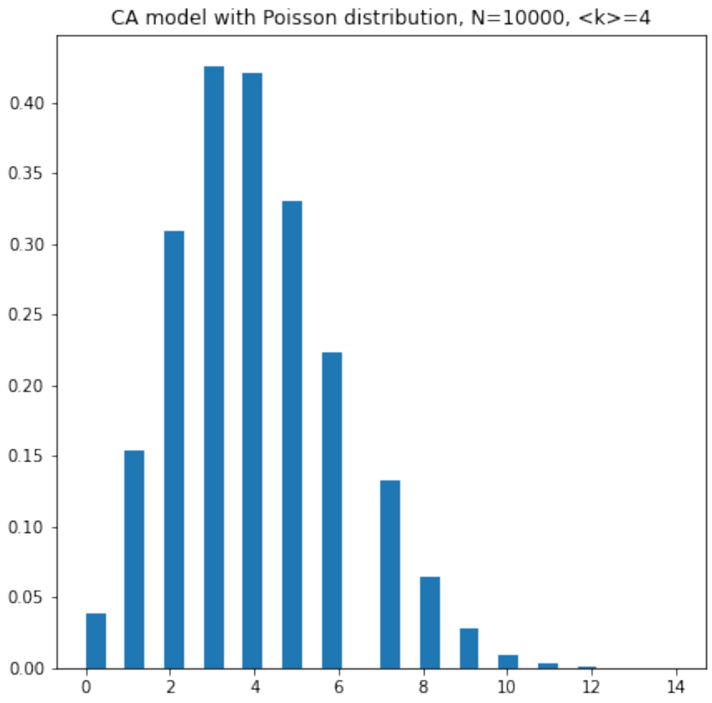
CM (Poisson)



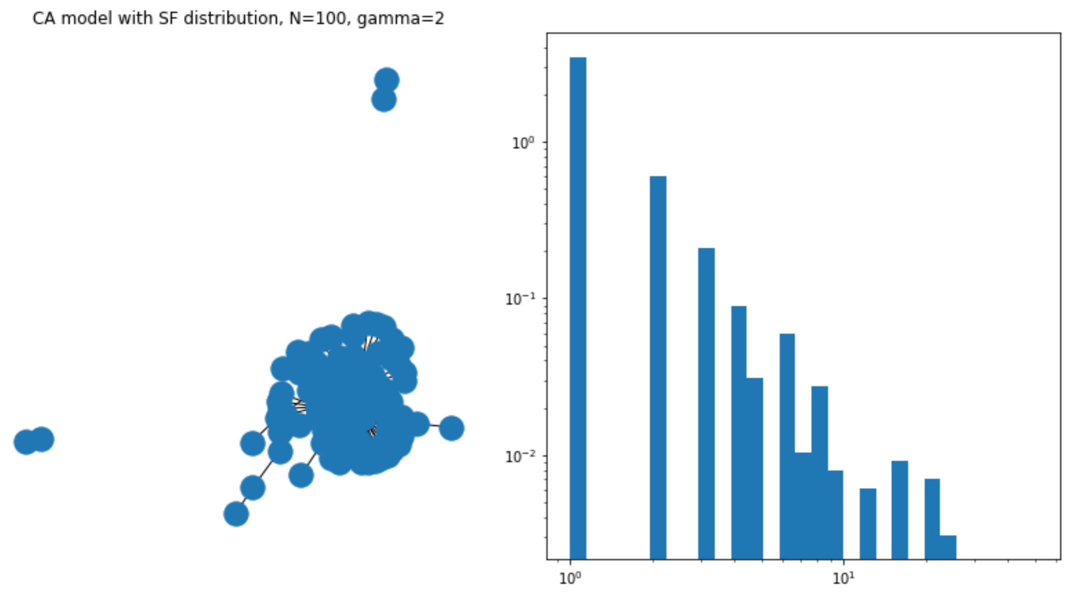


Big networks:

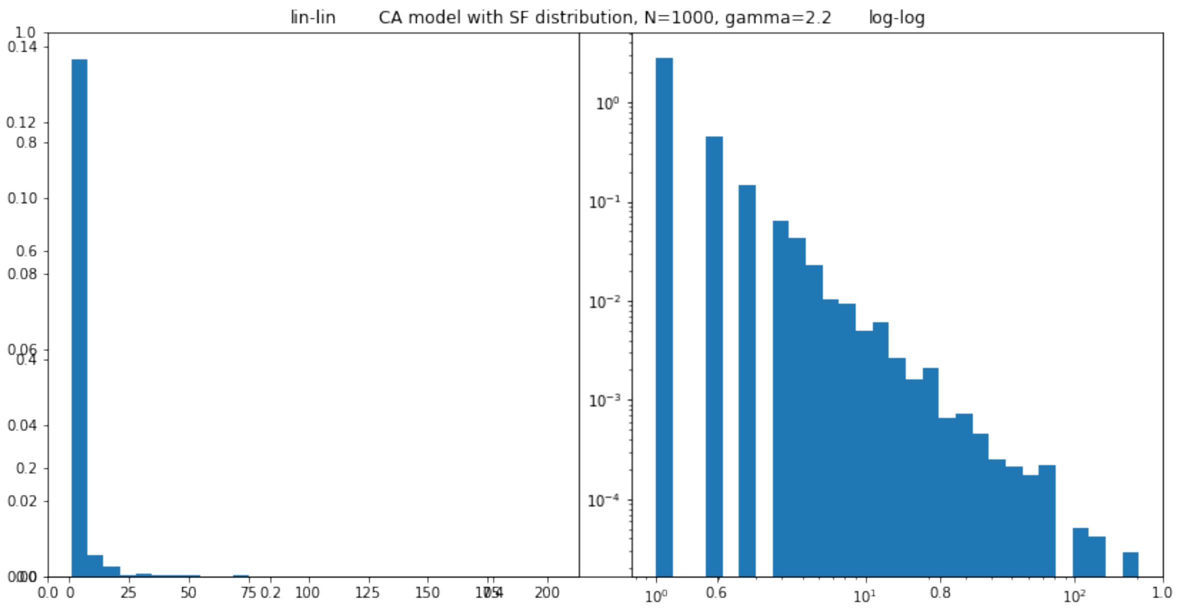




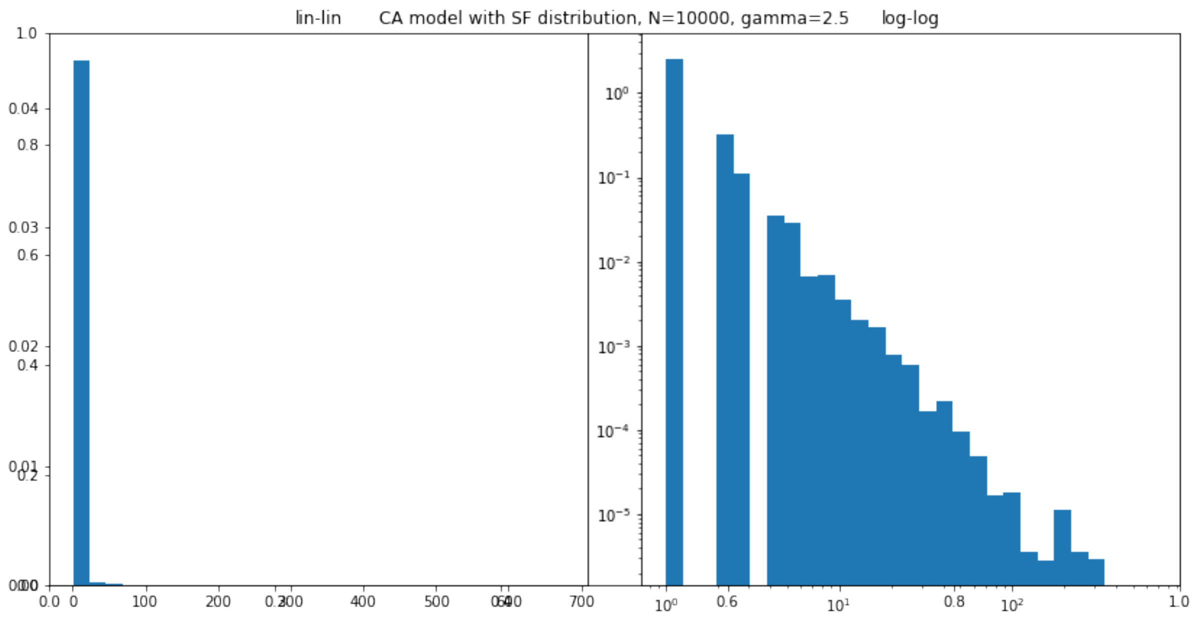
CM (Scale Free)



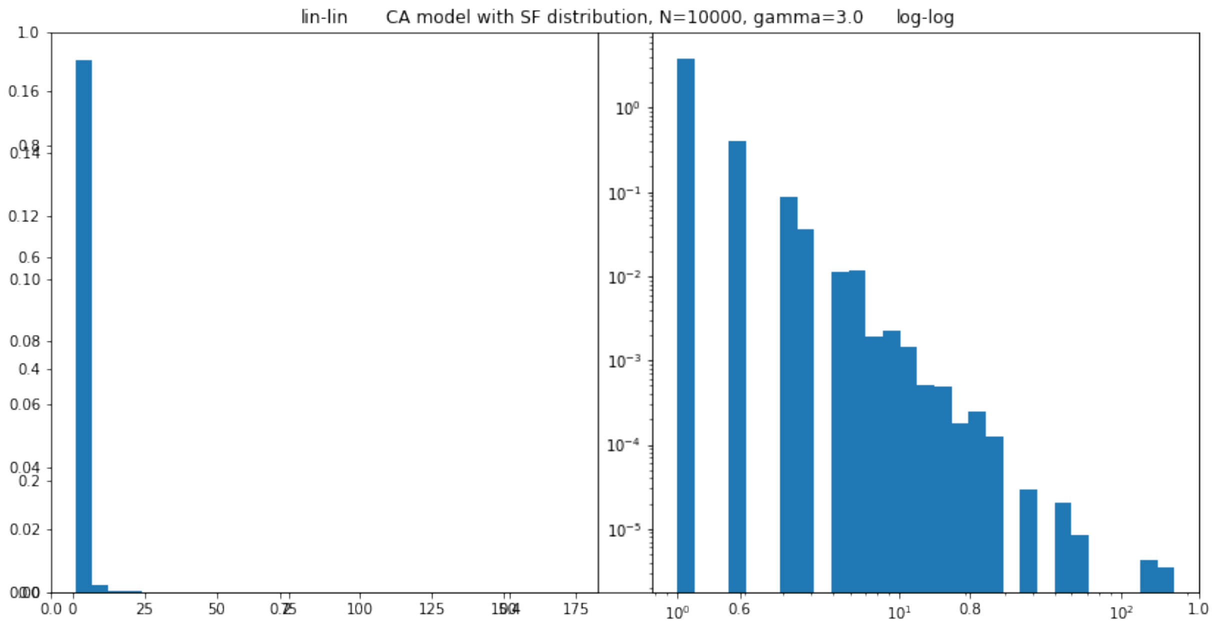
Big networks:



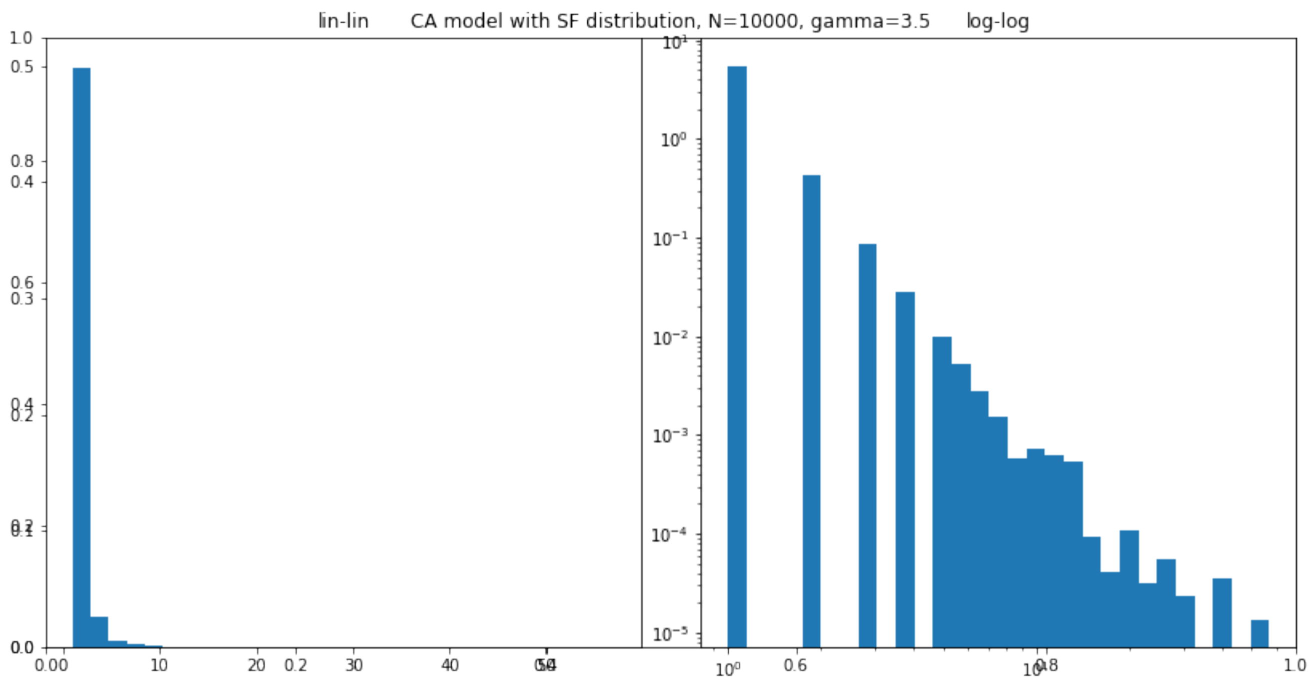
real gamma: 2.2
MLE gamma estimation, with k_min>=1: 1.768190533943788
MLE gamma estimation, with k_min>5: 2.184985934561395
regr coefficient: -1.207612375650401 , gamma = 2.2076123756504007



real gamma: 2.5
MLE gamma estimation, with k_min>=1: 1.8801780904348426
MLE gamma estimation, with k_min>5: 2.3335159115471353
regr coefficient: -1.3642803749835228 , gamma = 2.364280374983523



real gamma: 3
MLE gamma estimation, with k_min>=1: 2.0470731383128244
MLE gamma estimation, with k_min>5: 2.9092245003532757
regr coefficient: -1.8808252990090109 , gamma = 2.880825299009011



real gamma: 3.5
MLE gamma estimation, with k_min>=1: 2.15252498809289
MLE gamma estimation, with k_min>5: 3.174503921866916
regr coefficient: -2.218783605748986 , gamma = 3.218783605748986