POLYTECHNIC OF TURIN

Master's Degree
in Data Science and Engineering

Network Dynamics and Learning

Homework 3

Riccardo Bosio 291299

Anno Accademico 2021-2022

1 Introduction

This document contains my solution to the Homework 3's exercises of the Network Dynamics and Learning course.

At this link https://github.com/riccardobosio/NDL, it is possible to see the text of exercises and the code used.

I have discussed the solution with Beatrice Macchia.

2 Functions used

The function $infection$ is used to simulate if a determined individual will be infected or not. It takes as inputs the number of infected neighbors m and the probability β that the infection is spread from an infected individual to a susceptible one. The individual's probability of getting infected is $1-(1-\beta)^m$. The function picks up a random number between 0 and 1 and returns 1 (infected) if that number is greater than the probability of infection. Otherwise it returns 0 (not infected).

I use the same mechanism to simulate if an infected person will recover during the time step: *recovery* takes as input the probability p of an infected individual to recover and returns 1 or 0.

The function *epidemic_simulation* takes as input: the graph G on which to simulate the epidemic, the list *state* with the initial condition of the population, the number of weeks to simulate *duration*, the probability β that the infection is spread from an infected individual to a susceptible one, the probability p of an infected individual to recover and the list *vaccination* where each element corresponds to the percentage of vaccinated population reached that specific week. The output is a matrix with number of rows equal to the number of weeks and number of columns equal to the number of individuals. This matrix *stats* keeps trace of the status of the population for each week.

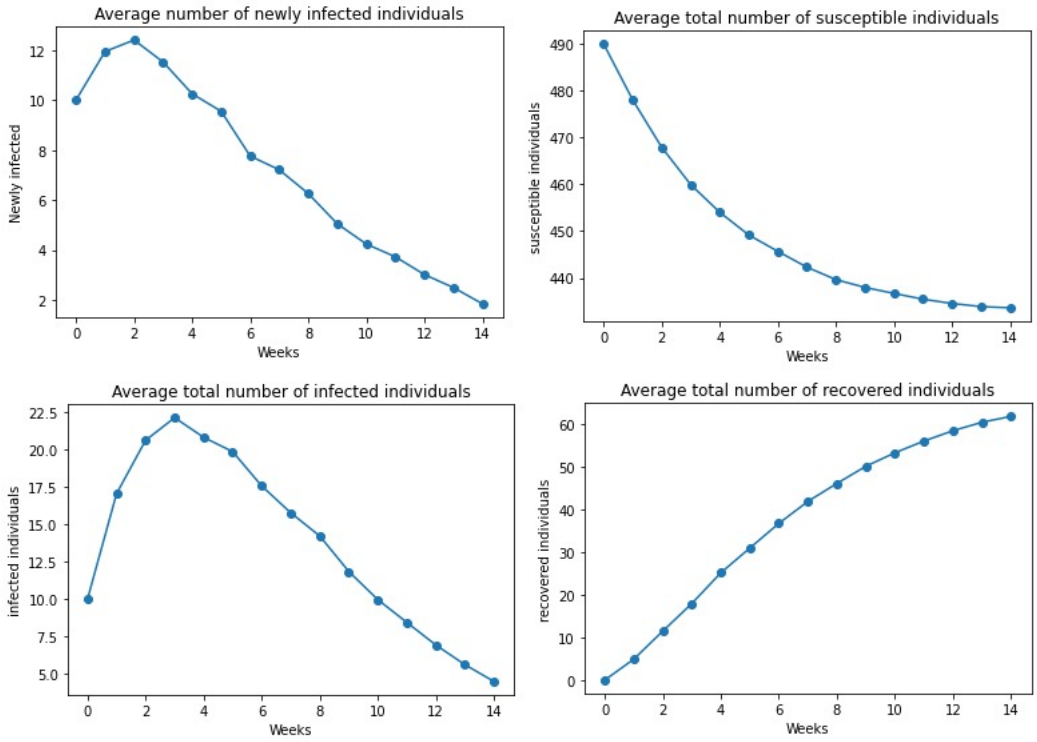
epidemic_simulation repeats this process for each week: first of all it calculates the number of new vaccines, then it randomly selects the individuals that will receive the vaccine and it updates their status. This must be done at first because the vaccination is assumed to take effect immediately once given, i.e. if person a is vaccinated in week 10, then a is no longer susceptible during that week, and can therefore not infect any other individual. Moreover, if an infected individual becomes vaccinated it is assumed that he will not be able to infect another individual. After managing vaccinations, for each node of the graph, the function uses $infection$ or *recovery* to find the new status of that individual.

Finally there is *multiple_simulation* that is able to perform different simulation and keep trace of the result of each one. The input are the same of *epidemic_simulation* plus the number of simulations *num*. The function calls *epidemic_simulation num* times.

3 Problem 1.1

To solve this problem I set the variables as requested and call the function *multiple_simulation* without passing the *vaccination* hyperparameter.

Figure 1: Results of exercise 1.



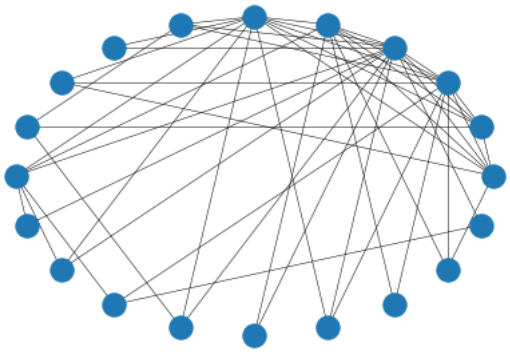
The results are shown in figure 1.

4 Problem 1.2

In the previous problem I simulated an epidemic on a symmetric k-regular graph. Now it is required to build a function that is able to create a random graph according to the preferential attachment model. The function *generate_random_graph* takes as input *nodes* as the number of nodes and *degree* as the degree of the final graph. The output is the obtained graph G.

I tested *generate_random_graph* passing $nodes = 20$ and $degree = 5$ and the result can be seen in figure 2. Then I ran the fuction passing $nodes = 1000$ and $degree = 20$ and the obtained degree is 20.

Figure 2: Example graph.



5 Problem 2

This problem is similar to problem 1.1 but instead of passing a symmetric k-regular graph, I simulate the epidemic on a random graph created using the function created for problem 1.2. The results are reported in figure 3.

6 Problem 3

Here I simulate multiple epidemics considering the vaccinations. The problem gives me the list where each element corresponds to the percentage of vaccinated population reached that specific week. I assign that list to *vaccination* when calling *multiple_simulation*. In figure 4 there are the results.

7 Problem 4

The goal is now to estimate the social structure of the Swedish population and the disease-spread parameters during H1N1 pandemic. First of all, I follow the instructions to create an algorithm that is able to do a gradient-based search over the parameters space k (the degree of the graph), β and ρ in order to find the set of parameters that best matches the real pandemic.

The different steps of the algorithm are implemented in *searchTop* that uses also the function $newly\_infected$ to compute the average new infected per week. Then I use $findParams$ to call the algorithm multiple times and manage improvements in the obtained parameters. The final $RMSE$ obatained is equal to 2.83 and the associated top parameters are $k = 12$, $\beta = 0.15$ and $\rho = 0.4$.

In figure 5 I compare the result of my simulations (in blue) with the ground truth given by the problem (in orange). Moreover in figure 6 the results in terms of susceptible, infected, recovered and vaccinated individuals are reported.

Figure 3: Results of exercise 2.

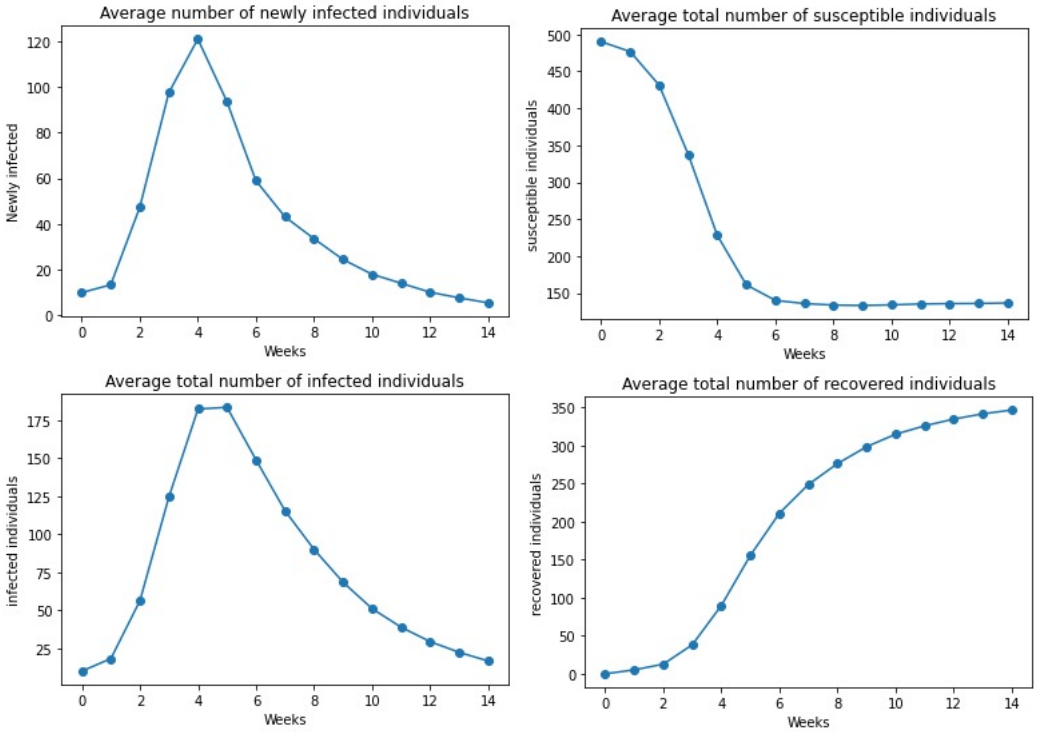


Figure 4: Results of exercise 3.

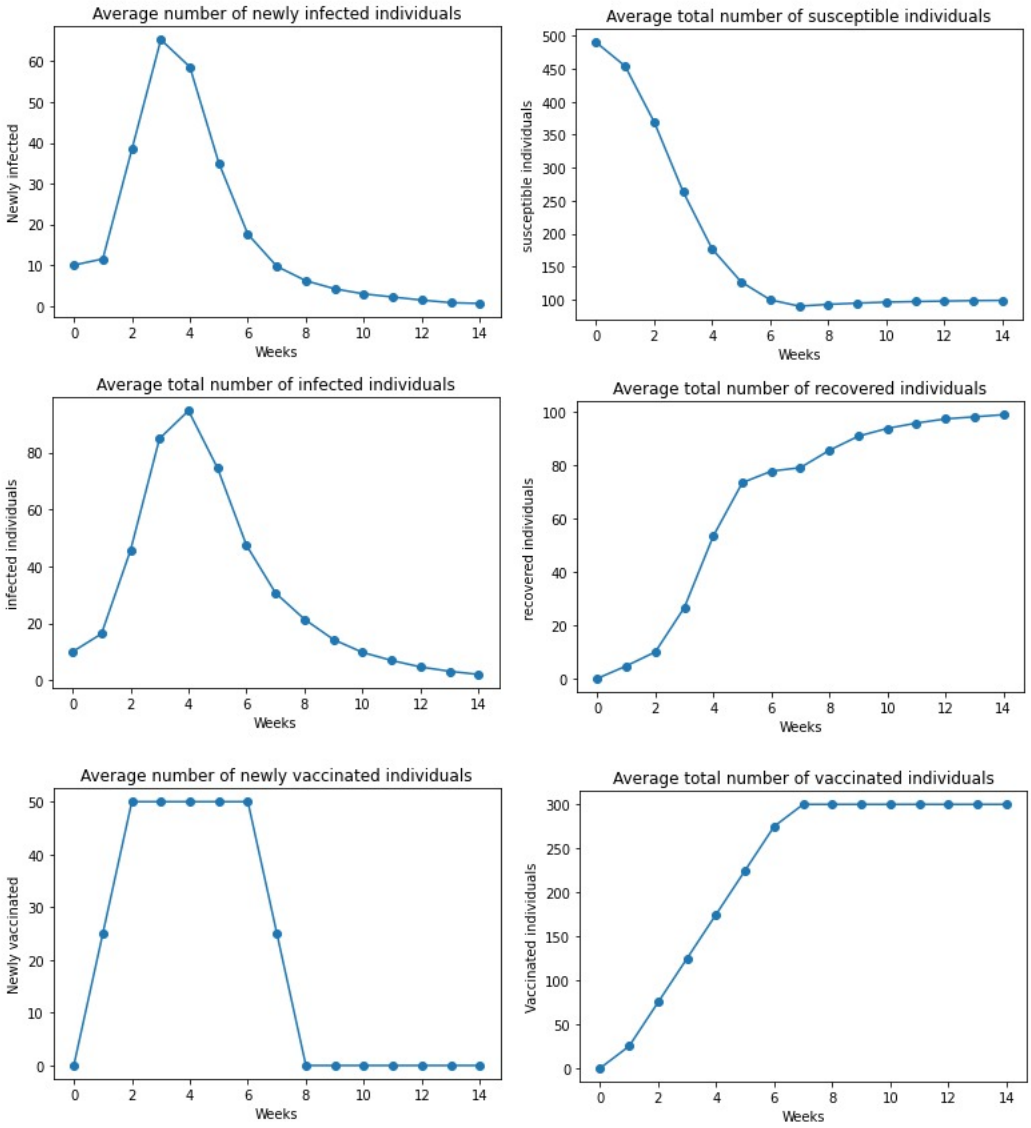


Figure 5: Results of exercise 4.

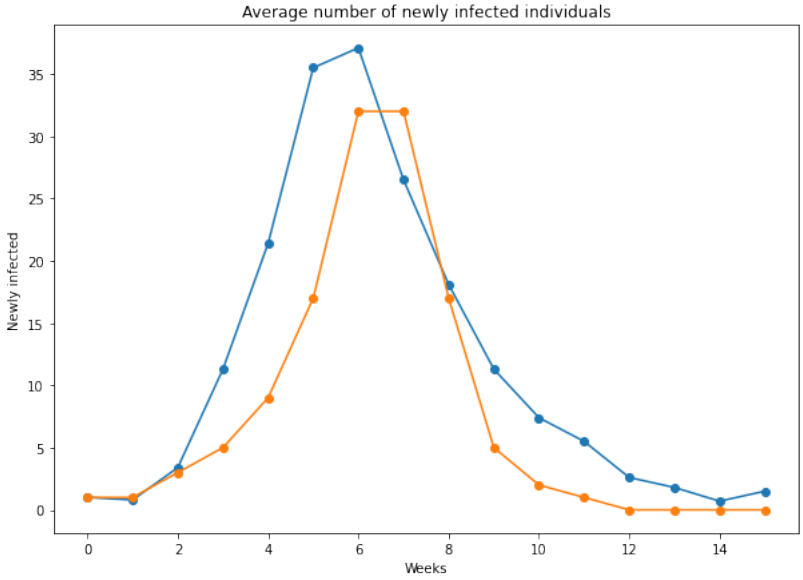


Figure 6: Results of exercise 4.