Hand-in 4 FRTN30 Gustaf Sundell

Gustaf Sundell, gu0147su-s

May 2021

Acknowledgements

I have briefly discussed this hand-in with Douglas Ihre and Fredrik Sidh.

1 Preliminary parts

As per the instructions, the preliminary parts consist in simulating an epidemic, and to generate a random graph with preferential attachment.

1.1 Simulate epidemic on given graph

The given graph (or population) is a k-regular graph with $k = 4$ and $|\mathcal{V}| = n = 500$ nodes/individuals. As this is a "perfect" (i.e. not random), undirected k-regular graph, this means each node has 4 in- and out neighbours. The code skeleton provided in Hint 2 in the instructions proved very helpful to implement this graph's adjacency matrix, W.

The simulation was implemented by closely following the instructions. The nodes in the graph update their states at discrete time steps, once per simulated week. The state space of the nodes is, in this problem, $\mathcal{A} = \{S, I, R\}$, but to ease the computations the states were given numerical values, such that $\{S, I, R\} = \{0, 1, 2\}$. This was then used to speed up computations, as matlab returns booleans as 0 or 1. The probability of a susceptible node to be infected at time $t + 1$, given that it has m infected neighbours at time t, as well as the probability that a node that is infected at time t recovers at time $t + 1$ are both given Equation 1, taken directly from the instructions. Throughout the hand-in, β and ρ will signify these given probabilities, n will always be the number of nodes in the graph being discussed, k the average degree of that graph (more on that in the next section) and $n_{infected}$ will signify the initial number of randomized nodes that will be infected before the simulation starts.

$$\begin{aligned}&\mathbb{P}\left(X_i(t+1) = \mathrm{I} \mid X_i(t) = \mathrm{S}, \textstyle\sum_{j\in\mathcal{V}} W_{ij}\delta^{\mathrm{I}}_{X_j(t)} = m\right) = 1 - (1-\beta)^m \\ &\mathbb{P}\left(X_i(t+1) = \mathrm{R} \mid X_i(t) = \mathrm{I}\right) = \rho\end{aligned} \tag{1}$$

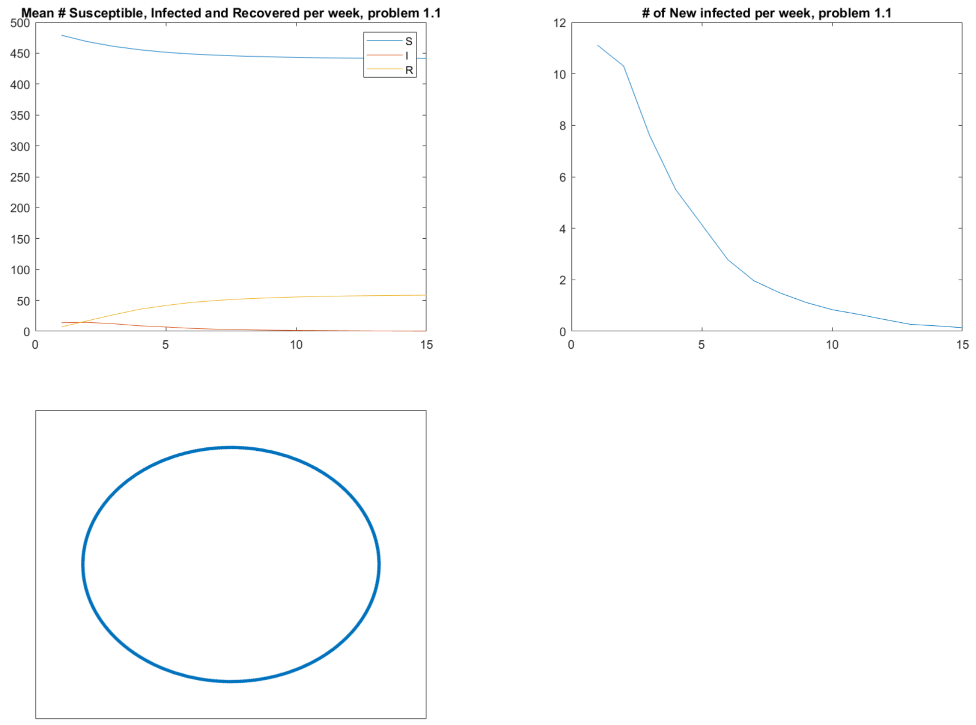

Figure 1: SIR plot, new infected plot as well as graph for problem 1.1

The simulation was implemented in matlab, creating the function SIR_simulation, which returns the mean of $\#S(t)$, $\#I(t)$, $\#I_{new}(t)$ and $\#R(t)$ as vectors, where t stands for time. Time will throughout this hand-in be discrete and measured in weeks. The values of the 4 measurements at week 15 (of course varying across simulations) were approximately $\#S(15) = 441.42$, $\#I(15) = 0.22$, $\#I_{new}(15) = 0.14$, $\#R(15) = 58.37$, which are deemed close enough to the values indicated reasonable in the instructions. The plots of these measures across the weeks (averaged across simulations) are shown in Figures 1. The same figure also shows the graph given for this problem.

1.2 Generate a Random Graph

The preferential attachment algorithm aims to create an undirected random graph having an average degree k which is pre-specified. For the implementation in this hand-in, this is only doable for $k \in \mathbb{Z}^+$. The algorithm implemented is based heavily on the instructions, and some of the code was borrowed from Excercise 13, and so I will not go in to much more detail.

The implementation resulted in a function called gen_population, and takes the parameters n and k, being $|\mathcal{V}|$ and the average degree respectively. It also has an optional argument to plot the result or not (most of the functions in this hand-in do). The function returns the adjacency matrix of the generated graph. Figure 2 depicts the graph generated by the algorithm with $n = 1000$ and $k = 8$. It is also verified in the matlab script that the average degree indeed is 8. The same figure also shows a histogram over the degrees in the graph. It is as visible, very skewed, with a few nodes having very high degree. As mentioned in a lecture by Giacomo, the preferential attachment algorithm results in the rich getting richer!

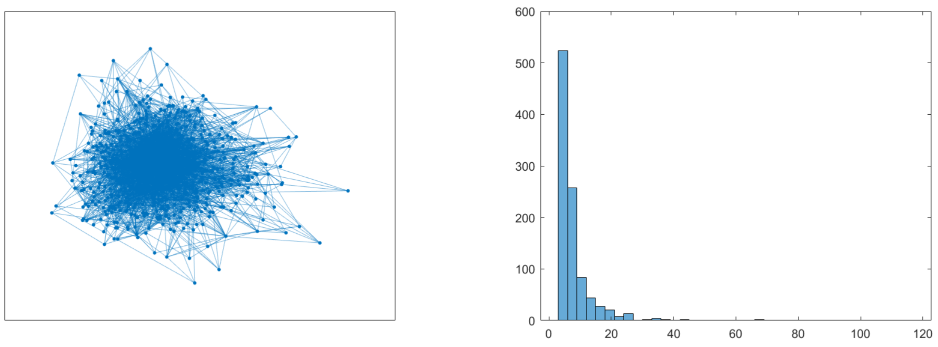

Figure 2: Random graph generated by the preferential attachment algorithm with $n = 1000$ and $k = 8$. Also histogram of resulting degrees.

2 Simulate a pandemic without vaccination

Now that the two main algorithms were in place and put into matlab-functions, creating a new random population and simulating the epidemic comes down to calling these two funcions. As specified in the instructions, we allow the function gen_population to create the graph and return the adjacency matrix, and then simulate the epidemic for the specified number of weeks. As per the instructions, we set $n = 500$, $k = 6$, $\beta = 0.3$, $\rho = 0.3, n_{infected} = 10$, and $N = 100$. The results of the simulation are presented in Figure 3. Looking at these plots, it is clear that the I_{new}-curve shows the increments in the "I-curve" in the SIR-plot.

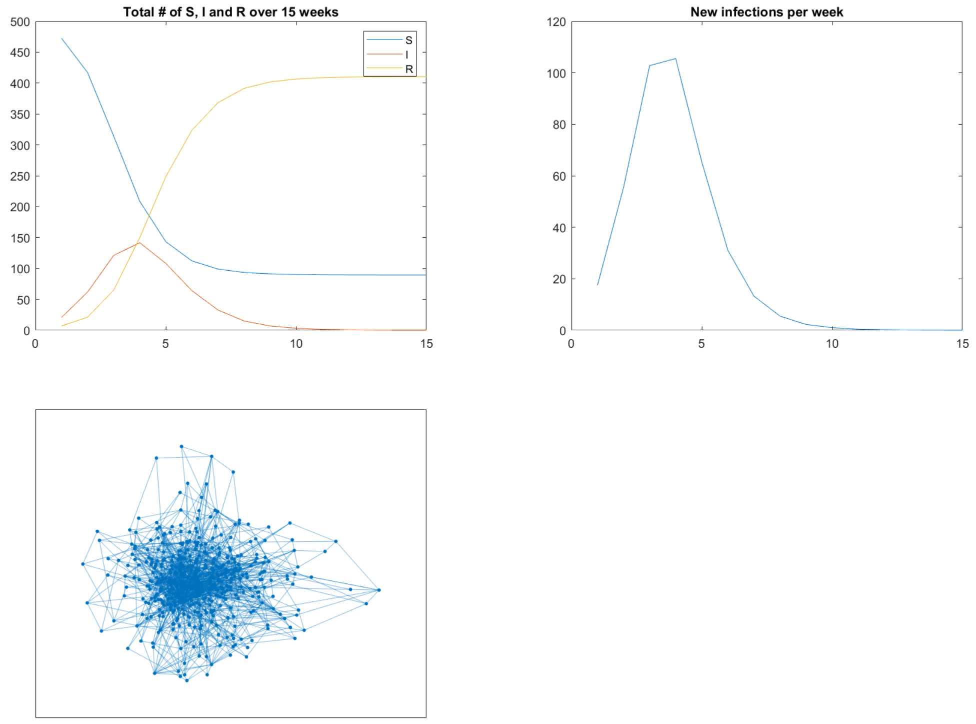

Figure 3: SIR plot, new infections plot and graph for problem 2.

3 Simulate a pandemic with vaccination

In this portion of the hand-in, a vaccination scheme is introduced. The vector containing the vaccination scheme was given in the instructions, and explains how many percent of the total population should be vaccinated at each week. The vaccination scheme vector was loaded into matlab and normalized to reflect actual percentages. The new function SIRV_simulation was created, where the previous function was refined to administer vaccines to the nodes. The state space was therefor extended to $\mathcal{A} = \{S, I, R, V\} = \{0, 1, 2, 3\}$. As before, the states were given numerical values. The results are presented in Figure 4. The vaccination scheme was implemented in the simulation as per the instructions. The input parameters for this simulation are identical to the ones in the last problem, only differing in that this problem also took the vaccination scheme as an argument.

The SIRV plot demonstrates the number of nodes that are in either of the four states (averaged across simulations) for every simulated week. The second plot shows newly vaccinated and infected people each week, again averaged across simulations.

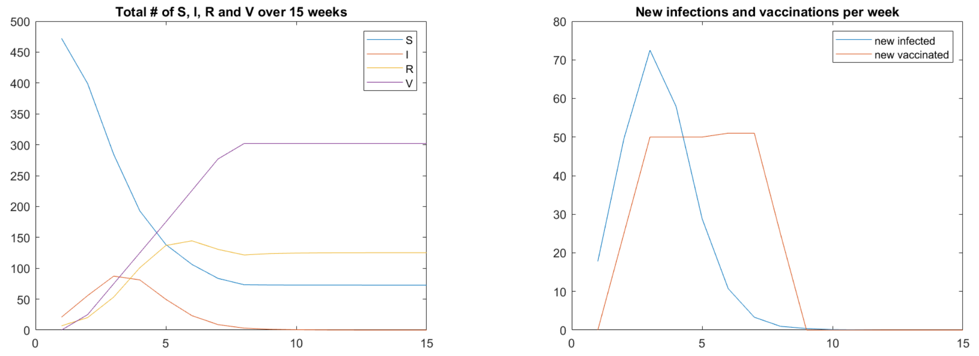

Figure 4: SIRV plot and new infecteds and vaccinations for problem 3.

4 The H1N1 epidemic in Sweden 2009

This problem is reminiscent of Problem 3, as we again simulate a population with the preferential attachment algorithm, as well as introducing a (new) given vaccination scheme. The in-paramaters differ somewhat however. The aim of this problem is to create a simulation yielding data close to the observed data from the actual H1N1 epidemic in Sweden 2009. Specifically, the parameters that are unknown and sought are k, β and ρ. The data to be fitted is given as $I_0(t) = [1, 1, 3, 5, 9, 17, 32, 32, 17, 5, 2, 1, 0, 0, 0, 0]$, which is the observed new infections per week during the epidemic. It is noteworthy that this vector is of length 16, whereas the simulations are run for 15 weeks. This is because the first element of the given vector is interpreted as week 0, i.e. the number of initial infected. Thus, the first element (which is 1) is assigned to the familiar parameter $n_{infected}$, a number that will not vary across simulations. Hence the given cost function will be minimized for $I_0(2 : end)$, and not the full vector, i.e. only for $t = 1 : 15$, rather than the actual span of the given vector, which is $t = 0 : 15$. The other parameters are set to $N = 10$ and $n = 934$.

With $I_{new}(t)$ denoting the simulated vector of newly infected nodes per week 1 through 15, the parameter optimization problem may be formulated as:

$$(k^*, \beta^*, \rho^*) = \underset{\substack{k \in \mathbb{Z}^+ \\ \beta \in [0,1] \\ \rho \in [0,1]}}{\text{argmin}} \ \text{RMSE} = \sqrt{\frac{1}{15} \sum_{t=1}^{15} \left(I_{new}(t) - I_0(t)\right)^2} \tag{2}$$

Where RMSE is the root-mean-square error between the simulated $I_{new}(t)$ and the observed $I_0(t)$ for $t = 1 : 15$. This is done by implementing a gradient-based search of the parameter space spanned by the triple (k, β, ρ), with Equation 2 demonstrating the constraints.

To conduct the search, we introduce the starting triple $(k_0, \beta_0, \rho_0) = (10, 0.3, 0.7)$, as well as discrete jumps to be taken $(\Delta k, \Delta\beta, \Delta\rho) = (2, 0.2, 0.2)$. Here, I deviate from the instructions. The starting triple worked out alright, but I found better results by starting with bigger Δ than the suggested, and then halving them in the search process. The reason is that larger jumps in the parameters initially makes for a large search space, and when the algorithm has selected the same parameter triple twice in a row, the Δ's are halved. As the optimization problem is probably not convex, this approach aims to avoid getting stuck in local optima, and to zero in on a more precise solution when a solution seems to be decent. The algorithm is designed such that when the Δ's have been halved three times, the search ends and the solution is considered to be found.

For each iteration in the search algorithm, there are 27 triples to evaluate, as there are 27 possible permutation of the "action" $(k \pm \Delta k, \beta \pm \Delta\beta, \rho \pm \Delta\rho$. This is akin to counting in base 3, and to yield all the 27 combinations, I create the matrix try_these_params. Each column of try_these_params is one parameter triple, from which a new population (of constant size) is generated, and on which population the SIRV simulation is run on with the specified vaccination scheme and fix number of iterations. For computational efficiency, I created a third simulation function called SIRV_simulation_streamlined, differing from the one used in Problem 3 in that it only returns the vector of newly infected each week.

Due to this, generating the requested plots for this problem was done by re-simulating the whole epidemic using SIRV_simulation. This means that the fall-out may differ from the fall-out yielded in the optimization algorithm, which is important to keep in mind. As always with simulations and randomness, results will differ between simulations, but it should be acceptable.

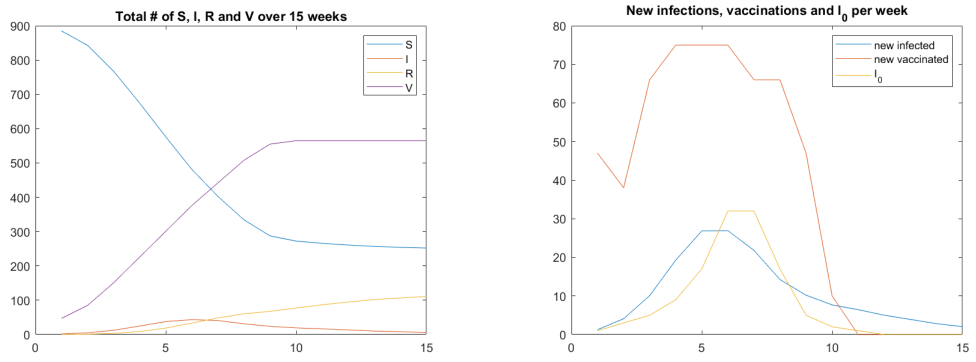

Figure 5: The SIRV plot and newly infected, newly vaccinated as well as I_0 for reference for Problem 4.

The results are shown in Figure 5. The optmial parameters found were $(k^*, \beta^*, \rho^*) = (15, 0.1, 0.5)$. The plots look pretty close, but not perfect. It should be noted that the optimal parameters vary across optimization runs. As mentioned before, playing around with the initial parameter values did not help me a lot in this problem, however it is highly possible that different initial values would help. Furthermore, implementing the preferential attachment algorithm such that k could assume floating point values could also help find a more refined optimum. In the current implementation, when Δk is halved to 0.5, the step taken for the k parameter needs to be rounded to an integer value.