Stochastic SIR: Variable Transmission Rates

Jordan Kaisman

April 2024

1 Introduction

We simulate an agent based stochastic SIR epidemic model in which interactions between agents have unique transmission rates. To achieve this, we first initialize each agent with a transmission score, chosen randomly from some given distribution. Then whenever an infected and susceptible agent interact during the simulation, the transmission rate for that particular interaction is taken to be the average of their respective transmission scores. To simulate interactions, we rely on the fact that the probability of an agent interacting with multiple other agents within the same time step becomes negligible for sufficiently small time steps. Thus, we may assume that each individual interacts with at most one other individual at each iteration. These pairs are random.

2 Initialization

We must first determine the distribution from which to sample the transmission scores. Clearly, scores must be positive. It is reasonable that there should also be an upper bound on scores, representing a default transmission frequency in the absence of any safety precautions. We therefore seek a distribution defined on an interval. There are many such choices, and we make the decision to use a mixture of two truncated normals on the interval. The rationale is that the population can be roughly grouped into those who take safety precautions and those who do not. Within each group, scores follow a truncated normal distribution.

We choose the interval to be $(0, 1)$, an arbitrary decision.

Then letting $N_1 \sim N(\mu_1, \sigma_1^2)$ and $N_2 \sim N(\mu_2, \sigma_2^2)$, we can construct T_1 and T_2 as their respective truncations on the interval $(0, 1)$. T_1 and T_2 will have their own means μ_3 and μ_4 which Matlab can compute numerically. If $g(x)$ is the density of T_1 and $h(x)$ is the density of T_2, we can define the density of their mixture M as

$$f(x) = wg(x) + (1 - w)h(x)$$

where $w \in [0, 1]$ is a weight parameter. Then by linearity of expectation,

$$\begin{aligned} E[f(x)] &= E[wg(x) + (1-w)h(x)] \\ &= wE[g(x)] + (1-w)E[h(x)] \\ &= w\mu_3 + (1-w)\mu_4 \end{aligned}$$

This allows us to control the mean of our mixture by changing w. In particular, we can set the mean to be $a = \frac{2.3}{7}$, which is the transmission rate used in the class notes for both the deterministic and stochastic simulation. This ensures that our scores, and thereby our rates, will on average be the same as the fixed rate in the original model.

Substituting a for $E[f(x)]$ in the above equation yields

$$\begin{aligned} a &= w\mu_3 + (1-w)\mu_4 \\ a &= \mu_4 + w(\mu_3 - \mu_4) \\ \frac{a - \mu_4}{\mu_3 - \mu_4} &= w \end{aligned}$$

We will experiment with different choices of parameters $\mu_1, \mu_2, \sigma_1, \sigma_2$.

3 Governing Algorithm

At each time step, we randomly pair all N individuals. Note that we are assuming N to be even. We treat all possible pairings as being equally likely, so that we are drawing from the uniform distribution on all
$(N-1)!! = (N-1)(N-3)\cdots(5)(3)(1)$ choices. This would require a lot of memory, so the pairing is actually done by randomly choosing the pair for agent 1 from among all $N-1$ other agents, and then choosing the pair for the next agent from among all $N-3$ remaining agents, and so on. We implement an in place pairing using an array of the indices called free. For each agent i, we randomly select j, $i < j \leq N$. Then $free[j]$ is available to be paired with i, so we swap it with $free[i+1]$ and then increment i by 2 in order to reach the next free agent. At each iteration i, the consecutive indices less than i hold the existing pairs, and all indices greater than i hold free agents. We also maintain an array called risk to keep track of S/I pairs. If node i is infected and happens to pair with a susceptible node j, we set $risk[i] = j$.

Terminology: The pairing problem has a graph theoretic interpretation. We can consider agents as nodes and pairings as edge constructions. This is why agents are referred to as nodes in comments in the code.

Once the pairs are established, we proceed to update states.

1. First, we restrict our attention to the S/I pairs. For each such pair, we determine the transmission rate a and infect the susceptible agent with probability $p = a \cdot \Delta t$.

2. Next, we consider each infected individual, and allow them to recover with probability $p = b \cdot \Delta t$.

3. Lastly, we identify all newly infections with the old infections. The reason for the delay is that none of the new infections should be considered in the recovery step above.

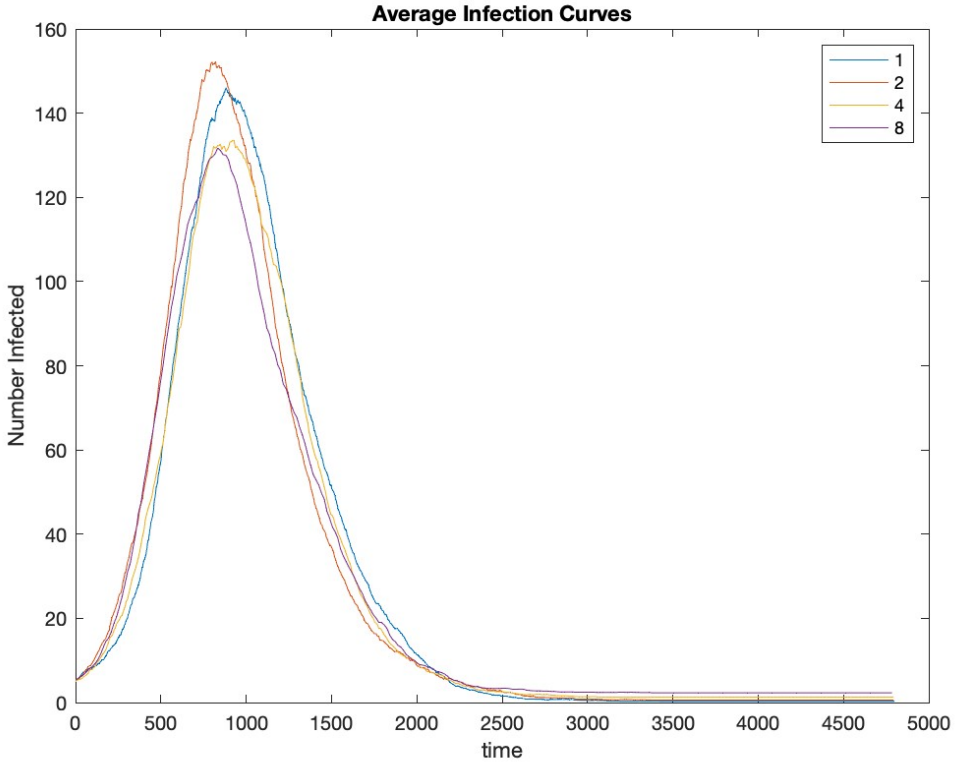
```
    % Algorithm

    for clock = 1: clockmax
        t = clock * dt;
        risk = zeros(1, N);
        free = zeros(1, N);
        for i = 1 : N %initialize array
            free(i) = i;
        end
        for i = 1 : 2: N
            j = i + unidrnd(N-i); % pointer to available index
            k = free(j); % available index
            % Pair node i with node k
            if (status(i) == 1) && (status(k) == 0)
                risk(i) = k;
            end
            if (status(i) == 0) && (status(k) == 1)
                risk(k) = i;
            end

            % Swap
            q = free(i+1); % save index
            free(i+1) = k;
            free(j) = q;
        end

        % Update States
        for i = 1: N
            if (status(i) == 1) % Node Infected
                if (risk(i) ~= 0)
                    j = risk(i);
                    a = (rate(i) + rate(j)) / 2; % average rates
                    p = a * dt;
                    if rand < p
                        status(j) = 3;
                        I = I + 1; %increment infected count
                        S = S - 1; %decrement susceptible
                    end
                end
                p = b * dt;
                if rand < p
                    status(i) = 2;
                    R = R + 1; %increment recovered
                    I = I - 1; %decrement infected
                end
            end
        end

        % Newly Infected
        for i = 1: N
            if (status(i) == 3)
                status(i) = 1;
            end
        end

        % Track Total

        tsave(clock) = t;
        Ssave(clock) = S;
        Isave(clock) = I;
        Rsave(clock) = R;

    end
```

4 Code

https://github.com/jordankaisman/Agent-SIR/blob/main/ASIR.m

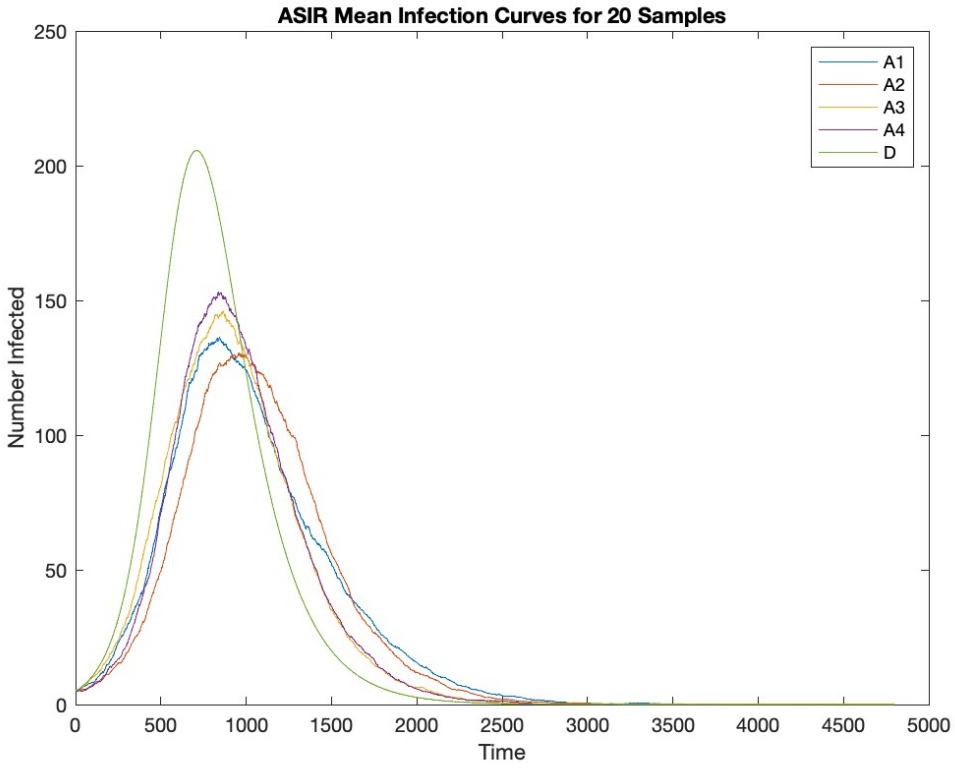
5 Validation

To ensure that the results do not depend on the choice of Δt, we graph the averaged infection curve for different choices of Δt: 1 hour, 2 hours, 4 hours, 8 hours. To generate an average infection curve, we repeat the simulation 25 times and then take the average simulation value at each time step. These particular results were obtained using a mixture generated by $\mu_1 = 0.25, \mu_2 = 0.6, \sigma_1^2 = \sigma_2^2 = 0.07$.



6 Results

We compare the averaged infection curves generated by different choices of the mixture parameters $\mu_1, \mu_2, \sigma_1^2, \sigma_2^2$. To reduce the number of free parameters, we restrict our attention to the case where $\sigma_1^2 = \sigma_2^2$ and fix $\mu_1 = 0.32$, $\mu_2 = 0.33$. The closeness of the two means results in a unimodal distribution. We test the values A1: $\sigma^2 = 0.01$, A2: $\sigma^2 = 0.04$, A3: $\sigma^2 = 0.08$, A4: $\sigma^2 = 0.2$, and also include the deterministic infection curve for reference. All of the agent based SIR infection curves display a lower peak and a thicker tail. This flattening of the curve is more prominent for smaller values of σ^2. Therefore, it is possible that lower variability of transmission rates throughout the population could have a curve flattening effect.



To test a bimodal distribution, we pull the means further apart. In particular, we fix $\mu_1 = 0.25$, $\mu_2 = 0.41$ and again test the values A1: $\sigma^2 = 0.01$, A2: $\sigma^2 = 0.04$, A3: $\sigma^2 = 0.08$, A4: $\sigma^2 = 0.2$. Again we observe that smaller values of σ^2 result in a flatter curve.