APPM 3170 Fall 2021
Tehya Laughlin and Claire Ely
Final Project Part Two

For the simulation we did not use time steps. We ran the simulation until there was a count of 0 infectious nodes, t_∞, *or until there were 0 susceptible nodes,* of which, the n_r at t_∞ would be 100%. All n_r values, and subsequent count of recovered nodes, implies that the rest are still susceptible, and that there are no infectious nodes, or that the rest are entirely infected. As a consequence, the number of recovered nodes as a function of roh_0 with the variable k, increases with k. This is because k is a factor that multiplies against the probability "q" within the simulation, which determines the recovery of a node. If "q" is increased, it means that a node is more likely to recover, so by the end of the simulation, there are more recovered nodes.

Problem 3 - Recovered Nodes vs. k-value

Approach:

Using the SIR simulation, we ran nested for loops to create a 5 by 100 matrix, and iterated the k value. We used the average number of recovered nodes as a function of roh_0. The table represented shows the roh_0 value using k and the average degree, as well as the average number of recovered does at the end of the simulation. The plots have error bars equivalent to the standard deviation of the 100 runs at each roh_0 value. The plots have the average n_r on the y-axis, but the tables show the average number of recovered nodes as the function of roh_0.

InVS13:

roh_0	1.6	3.2	4.8	6.4	8
Avg. |R(t)|	1.25	1.57	4.66	6.39	13.63

InVS15:

roh_0	3.9	7.8	11.7	15.6	19.5
Avg. |R(t)|	1.61	8.87	16.15	25.16	20.81

LH10:

roh_0	3.0	6.0	9.0	12.0	15.0
Avg. |R(t)|	28.07	44.61	61.37	64.30	65.74

LyonSchool:

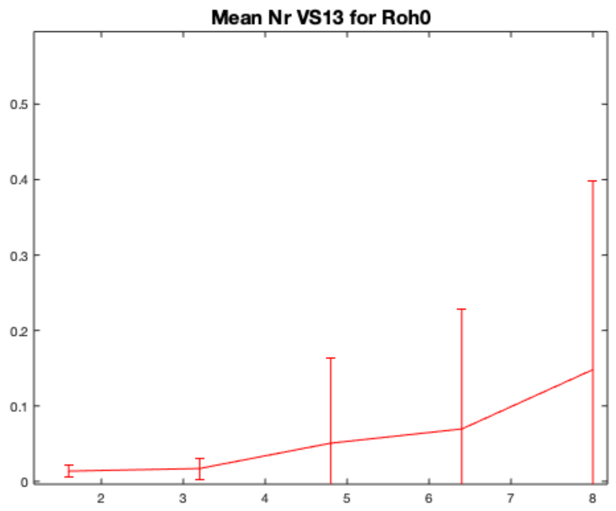
roh_0	6.9	13.8	20.7	27.6	34.5
Avg. |R(t)|	183.54	203.66	216.79	214.78	204.23

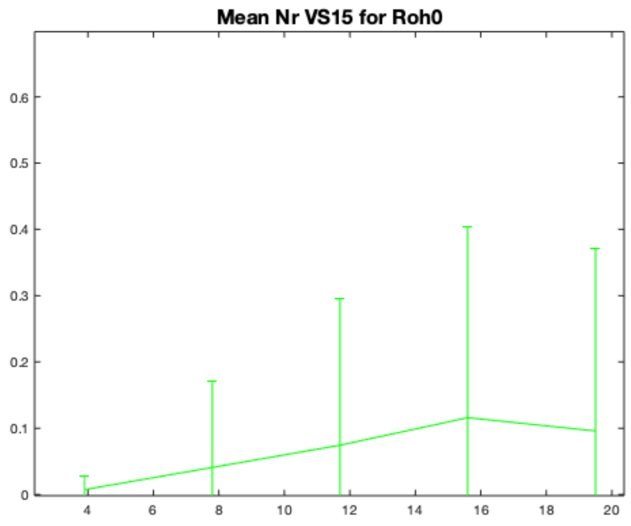
SFHH:

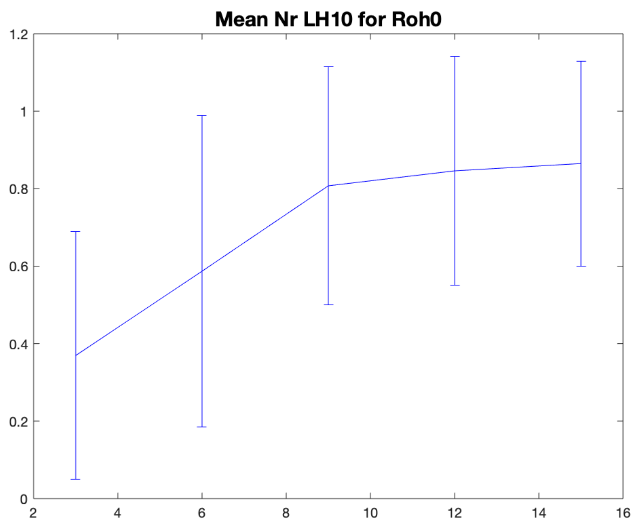
roh_0	4.8	9.6	14.4	19.2	24.0
Avg. |R(t)|	2.35	15.60	90.63	170.45	179.63

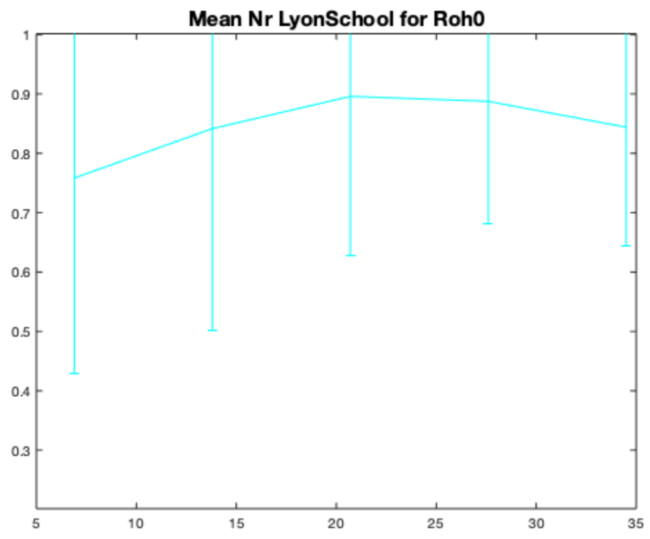
Thiers13:

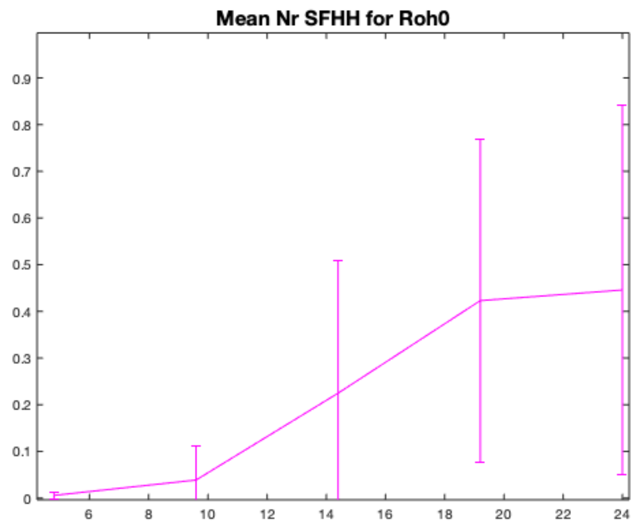
roh_0	3.5	7.0	10.5	14.0	17.5
Avg. |R(t)|	1.56	4.57	26.63	46.77	93.60

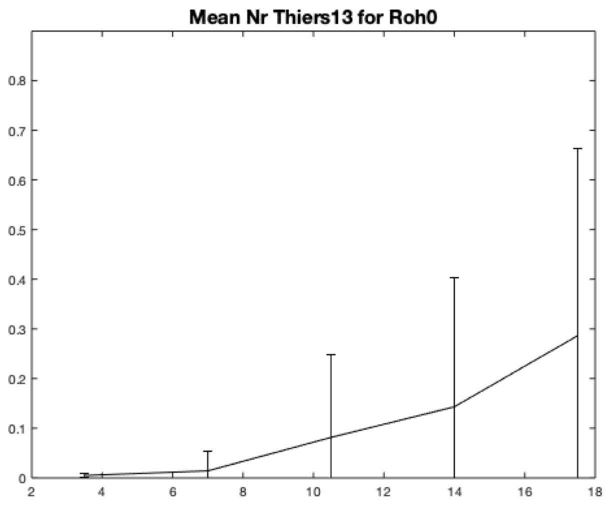












Analysis:

For the well connected graphs like SFHH and the LyonSchool, as k increased, the number of recovered nodes also increased. The simulation of the epidemic for k = 5 showed that many people, nodes, got infected and recovered. When the k is smaller, like 1 or 2, most locations had only a small number of recovered nodes at the end, showing there could be a majority of infected population, because the simulation also stops when there are no more susceptible nodes. For the Lyon School, this was not the case. The Lyon School has 242 nodes, yet at k = 1, the number of nodes recovered was more than half, at 75%. And at k = 5 for Lyon School, 84% of the population had the infection and recovered. For some of the epidemics, the roh_0 values were similar to the number of nodes recovered, but of Thiers13, which has been shown to mimic InVS15 in degree, density, and cc behavior, also followed this trend despite having more nodes. The location LH10 also had steady increase in the number of nodes recovered as k increased, going from 37% to 87% recovered for k = 1 and 5, respectively. As the mu value, which encodes how fast an infected node recovers from the virus, is dependent on the k-value, the larger the roh_0 value (directly proportional to the k value), the faster the recovery from the virus. As such, the proportion of recovered nodes is higher for a higher k-value.

Code: This function makes an array for the number of recovered nodes:

```
function Arr = AvgRnodePerK(A,sz)
% this function fills a row array with the average number of recovered
% nodes for each k
Arr = zeros(1,5);
sums = sum(A,2);
    for k = 1: 5
        Arr(k) = (sums(k)/100)* sz;
    end
end
```

Problem 4 - Fraction of Recovered Nodes vs. k-value

Approach:

For each k, there are 100 simulation runs. We wrote a program that runs through each 100 simulations at each k and counts the number of n_r values that are greater than 0.2. Then the program returns the fraction of n_r values that meet the condition of the 100 runs. The values if the tables are fractions of the $|n_r > .2|$ / 100. The tables below contain 0 if there are no n_r values greater than 0.2 in any of the 100 simulation runs.

Fraction N_r: $|(n_{rr} > .2)|$ /100 : is the fraction of n_r that are greater than .2 for the 100 simulations

InVS13:

roh$_0$	1.6	3.2	4.8	6.4	8
Fraction N_r	0	0	0.08	0.09	0.21

InVS15:

roh$_0$	3.9	7.8	11.7	15.6	19.5
Fraction N_r	0.01	0.07	0.09	0.13	0.1

LH10:

roh$_0$	3.0	6.0	9.0	12.0	15.0
Fraction N_r	0.55	0.67	0.87	0.89	0.92

LyonSchool:

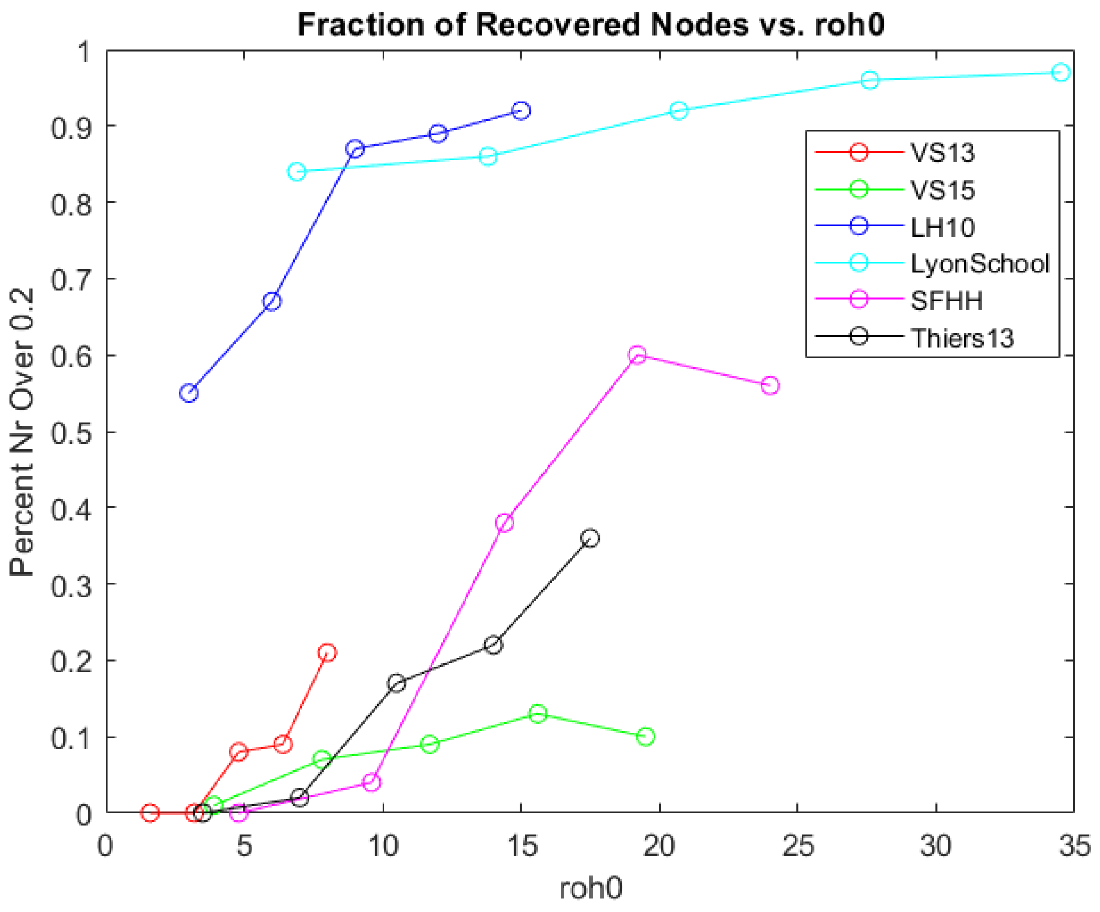
roh$_0$	6.9	13.8	20.7	27.6	34.5
Fraction N_r	0.84	0.86	0.92	0.96	0.97

SFHH:

roh$_0$	4.8	9.6	14.4	19.2	24.0
Fraction N_r	0	0.04	0.38	0.60	0.56

Thiers13:

roh_0	3.5	7.0	10.5	14.0	17.5
Fraction N_r	0	0.02	0.17	0.22	0.36



Analysis:

As k increases, there are generally more epidemics where the fraction of recovered nodes is greater than 20%. For the Lyon School and LH10, the fractions are much greater than the other locations. However, the roh_0 value for Lyon School is double the roh_0 value for LH10. Their connectivity must be in play then, as the average cc of LH10 and Lyon School are around 1, with LH10 being 0.3 higher. The densities of LH10 and Lyon School are also higher than the other graphs, with LH10 having a density of 0.4 and Lyon School having a density of 0.28. For reference, the densities of the other locations are less than 0.2.

For SFHH, the graph with the highest number of edges, is also the graph with the lowest average cc, which explains why the epidemics aren't on the level of LH10 despite being seemingly more "connected."

The only locations where an increase in k didn't correspond to a completely upward trend in percent recovered greater than 20% was for InVS15 and SFHH, where the change from k=4 to k=5 resulted in a slight decrease in percent recovered.

Code:

Function that counts the fraction of n_r

```
function frac = fractionOfNR(A)
% this function counts the number of return SIR values that are greater
% than .2. Then is adds that fraction to an array, which is the returned
frac = zeros(1,5);
for k = 1: 5
     count = 0;
    for i = 1: 100
        if A(k,i) > .2
            count = count +1;
        end
    end
    frac(k) = (count / 100);
end % end for
end % end function
```

Problem 5 - Average recovered nodes where the final fraction is greater than 20%

Approach:

The problem asks to average the number of recovered nodes for each epidemic where the n_r value is greater than 0.2. The code runs through k, from 1 to 5. At each k value, it runs from 1 to 100 in the array holding the SIR return values, and checks to see if the value is large enough, and then adds the number of recovered nodes to a sum. At the end of those 100 simulations, it divides the sum and stores that average value in an array.

The idea is to see when the simulation stops in terms of recovered versus susceptible nodes for each k. By having a limit on the n_r value, there is a measure of statistical accountability when counting the SIR run as an epidemic of statistical weight.

0 represents no n_r values for that roh_0 value that are greater than 0.2.

Avg. |R(t)| - n_r : The average number of recovered nodes for the epidemics where n_r is greater than 0.2.

InVS13:

roh_0	1.6	3.2	4.8	6.4	8
Avg. |R(t)| - n	0	0	36.750	50.667	57.048

InVS15:

roh_0	3.9	7.8	11.7	15.6	19.5
Avg. |R(t)| - n	47.000	108.714	167.889	186.077	198.700

LH10:

roh_0	3.0	6.0	9.0	12.0	15.0
Avg. |R(t)| - n	49.146	65.896	70.299	72.112	71.359

LyonSchool:

roh_0	6.9	13.8	20.7	27.6	34.5
Avg. |R(t)| - n	218.179	236.628	235.533	223.688	210.516

SFHH:

roh_0	4.8	9.6	14.4	19.2	24.0
Avg. |R(t)| - n	0	127.250	233.974	283.050	319.536

Thiers13:

roh_0	3.5	7.0	10.5	14.0	17.5
Avg. |R(t)| - n	0	86.000	138.882	203.500	257.000

Analysis:

The number of nodes recovered and n_r are related, so if the n_r count doesn't include the lower values, then the average of the number of recovered nodes when only counting the epidemics with higher n_r values, will also be higher. This also means that there is less of a difference between the averages when compared with which k is used, because the lower the k is, the more likely that there are more n_r values lower than 0.2, so the range is limited, which has different weights for different k's.

We see generally that for the lowest k-values (k=1), the time it takes to recover from an infection makes it so that the percentage recovered of the population is lower than 20%. Since the simulation has a clause that also stops the simulation if there are no more susceptible nodes, it is possible that these epidemics are also stopping with a low recovery number because the infection affects most of the population, although that's unlikely.

Code:

Function that makes an array for the average number of recovered nodes based off of the $n_r > 0.2$

```
function avg = fracNRAvgR(A,sz)
% this function counts the number of return SIR values that are greater
% than .2. Then is adds that fraction to an array, which is the returned
avg = zeros(1,5);
for k = 1: 5
     sum = 0;
     count = 0;
    for i = 1: 100
        if A(k,i) > .2
            sum = sum + (A(k,i) * sz); % modifies nr to be nr*n = numR
            count = count + 1;
        end
    end
    avg(k) = (sum / count); %count is not 100, because not all of the nr
are > .2
end
End
```

Simulation Code:
```
function Nr = SIR(A, k)
nNodes = length(A);
beta = 4e-4;
mu = 10*beta/k;
v0=randi(nNodes, 1);
delt = 5e-3/beta;
q=mu*delt;

%initialize w random choice of k for mu
Snodes= ones(1, nNodes);
Inodes = zeros(1, nNodes);
Rnodes = zeros(1, nNodes);
Snodes(v0) = 0;
Inodes(v0)=1;

%while loop before Inodes && snodes ~= 0 (count time stamps)
while (sum(logical(Inodes)) ~= 0) && (sum(logical(Snodes)) ~= 0 )
    infected = find(Inodes);

    for i =1:length(infected) %iterate over each infected node

        Auv = A(infected(i),:); %neighbors
        friends = find(Auv);

        for u =1:length(friends)
            ifriend = friends(u);

            if Snodes(ifriend) > 0  %if the friend at node u is suscep
                p = beta * A(u,i) * delt;
                sick = (rand(1,1) < p);
                if(sick)
                    Inodes(friends(u)) = 1;
                    Snodes(friends(u))= 0;
                    fprintf('\n\t susceptible = %d\t infected = %d\t
recovered %d', sum(Snodes), sum(Inodes), sum(Rnodes));
                end
            end
        end
         %v recovers??
        recovery = (rand(1,1)<q);
        if(recovery)
           Rnodes(infected(i))=1;
           Inodes(infected(i))=0;
        end

    end
end
```

```
%output Nr
Nr = sum(Rnodes)/nNodes;
end
```