Testing Frequency in an Noncompliant Population

Original Authors: Nick Cunha, Sam Wisnoski

Section I: Modeling Question

Modeling Question:

- **How often do we need to test in order to have the peak number of infected people lower than 20?**
- This is a **design** question as it aims to discover exactly how often tests must be administered in order to achieve a certain peak number of infected persons.

Background Information:

In March of 2020, the world shut down as schools, businesses, and public places closed in an attempt to slow the spread of COVID-19. As essential workers and students returned to their careers and schools, many administrators across the world were asking the question: "How often should we covid test the people returning to work?". Accurately Covid testing dozens or hundreds of people on a regular basis is expensive, but public health is of the utmost safety. In this paper, we sweep over hundreds of different testing frequencies ranging from every hour to every month in order to calculate the safest testing frequency while simultaneously avoiding wasting tests. Additionally, throughout the pandemic, many people broke quarantine, refused to isolate, or take precautions when infected. In order to capture this in our results, the reaction to testing positive will vary from individual to individual.

Importance/Interest:

In any pandemic, the goal is to limit the spread and destruction of a disease. However, the only way to truly limit the spread of disease is complete isolation - which is impossible in many scenarios as essential care workers and other critical members of society must continue to work in order for society to function. One way to protect these workers is to enforce regular testing, a practice many systems enacted upon during the initial rise in covid. These systems often lacked proper research, and chose a testing frequency with no particular backing. This led to tests not being administered frequently enough and infections rising, or tests being administered too frequently and money wasted. Our project aims to find the perfect testing frequency --- with the information our model produces, we can make better informed decisions; maximizing safety during an outbreak while minimizing cost.

Research Synthesis:

Sources → ↓ Topics ↓	California Institute of Technology	Center for Disease Control	Federal Drug Administration	Other Sources
Effectiveness of at home COVID-19 tests	Caltech.edu 15 of the 17 study participants had high and presumably infectious levels of virus for at least a day prior to getting a positive antigen test.	N/A	Fda.gov at-home COVID-19 antigen tests are generally expected to detect the SARS-CoV-2 virus at least 80% of the time when someone is infected.	N/A
Recommended quarantine length after a positive test	Caltech.edu Isolation may end after Day 5 if both of the following criteria are met: You have had no fever for at least 24 hours (without the use of fever-reducing medicine); Other symptoms are not present, or symptoms are mild and improving.	CDC.gov stay home for at least 5 days and isolate from others in your home.	N/A	covid19.ca.gov Isolate for at least 5 days.
How often should you test	N/A	CDC.gov FDA recommends 2 negative antigen tests for individuals with symptoms or 3 antigen tests for those without symptoms, performed 48 hours apart.	N/A	Mass.gov By testing more frequently, you might detect the virus that causes COVID-19 more quickly and could reduce the spread of infection.

Parameters:

Deterministic Parameters:

- **The rate of infection is 0.0002 (chance of an infected person infecting another person they have contact with per timestep (one hour).** Although this value is relatively arbitrary, we chose a value that closely represents how Covid acts within the real world. When we generate our model, our infection peaks then slowly begins to drop off as more people recover and gain immunity. In choosing a value of 0.0002 (per hour) or 0.0048 (per day), our model accurately represents reality. Additionally, this is a deterministic value in order to represent the physical world - the rate of infection of a disease is constant, however, the circumstances that surround the infection are not. This is why we have developed a random

adjacency matrix in order to represent a realistic population, where people will have different connections with each person in their community. Furthermore, to properly simulate the actual act of infection, the action function contained within simulate_absir_mod randomly determines infection with each iteration. In short, we chose this rate of infection since it simulates the real world both in population dynamics and in the spreading of disease.

- **The rate of recovery is 0.006 (chance of an infected person recovering per timestep (one hour).** We chose this value after considerable research into the recommended quarantine regulations provided by Caltech.edu, CDC.gov, and covid19.ca.gov, which all recommend a quarantine of 5 days and the loss of symptoms. However, the minimum quarantine is five days, but an average infection lasts a full week or more with the inclusion of the asymptomatic/non-infectious period. To mirror the real world and simulate a week-long infection, we chose a recovery rate of 0.006 per hour, which correlates to a recovery rate of 0.144 per day or an average of seven days to recover.
- **testing frequency:** As we decided to sweep testing frequencies, we simply needed an array that swept over all reasonable testing values. For this purpose, we created an array that swept 800 values, from testing once every hour to once every 800 hours. Within this array, we selected several common testing frequencies to mark on our graph - once a day (24 hours), twice a week (84 hours), once a week (168 hours), once every ten days (240 hours), once every twenty days (480 hours), and once every thirty days (720 hours)
- **adjacency matrix**: We developed an adjacency matrix with randomized edge weights to represent a complex society with many different types of connections and interactions. Within this matrix, people are able to have a connection ranging from 0 to 3 with every other person in this system. However, while this is a random matrix, it is a deterministic parameter. It is only generated a single time, and the values are set in stone each time we run the simulation.

Random Parameters:

- **Our quarantine distribution uses a beta distribution that generates values between zero and seven, with the distribution skewed left and centered on five**. This distribution mimics the quarantine regulations provided by Caltech.edu, CDC.gov, and covid19.ca.gov, all which recommended quarantining for at least five days. Additionally, in order to mimic personal decisions, we made this value a random parameter. Upon testing positive for Covid, each person will be assigned a value based on this parameter. This represents how they react to quarantine, as reported by not everyone follows guidelines. While most people will end up quarantining for five days, many others will quarantine for either less than or more than five days. This simulates the real world, as reported in "Misrepresentation and Nonadherence Regarding COVID-19", 22.5% of people broke quarantine. We made this a beta distribution as we needed hard boundaries (0-7 days) and a skewed distribution, two things that a beta distribution does well.

State Variables:

While all state variables are initially deterministic, ie, we determine their size and we determine who starts susceptible vs. infected vs. recovered, they all evolve randomly. Both actions in which the state variables

change, where susceptible people become infected or infected people recover, are random. Therefore, I would argue that all three state variables, in each timestep beyond their initialization, are random.

Deterministic State Variables:

- None

Random State Variables:

- Susceptible Persons: Susceptible persons are a random variable, they are updated regularly when a susceptible person becomes randomly infected based on a Bernoulli distribution.
- Infected Unquarantined Persons: Infected persons are a random variable - with each timestep, each susceptible person has a random chance to be infected based on their connections within the adjacency matrix and the infected rate. Since this chance is decided randomly but evenly, infected persons are a random variable based on a Bernoulli distribution.
- Recovered Persons: Recovered persons are a random variable - with each timestep, each infected person has a 0.006 chance to recover. Since this chance is decided randomly but evenly, recovered persons are a random variable based on a Bernoulli distribution.
- Quarantined Persons: As discussed above, the quarantine distribution is randomized as a beta distribution.

References:

- Caltech.edu
- CDC.gov
- Covid19.ca.gov
- Fda.gov
- Mass.gov
- Misrepresentation and Nonadherence Regarding COVID-19

Section II: Model and Methodology

Description:

Our Model and Methodology section is broken into seven sections: Creating Adjacency Matrix, Creating Beta Distribution, Running a Single ABSIR Model, Sweeping Over Testing Frequencies, Sweeping over Various Distributions, Adressing our Modeling Question, and Model Validation. Each section goes into further detail about the choices we made when building that section, but a broad overview of model and methodology is below:

1.) Create Adjacency Matrix: Generate an adjacency matrix with randomized edge weights to accurately represent an average population.

2.) Create Beta Distribution: Generate a beta distribution and multiply it by a certain factor of seven so that it represents values 0-7 rather than values 0-1, and it will peak at the recommended number of days to isolate (five days).

3.) Run a Single ABSIR Model: Using the simulate_absir function runs a single realization, demonstrating the mechanics of the ABSIR function.

4.) Sweep over testing frequencies: Sweep over hundreds of different testing frequencies to find the relationship between testing frequency and peak number of infected persons.

5.) Create Multiple Realizations: Using the simulate_absir function to run multiple realizations, allowing us to analyze the mean data of the realizations for a more accurate data set.

6.) Adressing our Modeling Question: We examine our results from the mean of the realizations to determine exactly how often tests must be administered to lower the peak number of infected people below twenty.

7.) Model Validation: Comparing our model against other research studies for both parameter sweeps and ensuring our models make sense with respect to the real world.

Underlying Assumptions:

- Every person who receives a positive test will immediately go into quarantine with no delay
- A quarantined person will have zero interaction with anyone else during their isolation period
- The maximum edge weight values in the population matrix are completely arbitrary
- The Covid tests are always accurate
- Persons who were previously infected cannot be reinfected
- The infection rate is constant for every person
- No person in the model is vaccinated
- Every person has an identical immune system (recovery rate)

I. Create adjacency Matrix with randomized edge weights

The first step of this project is to create a randomized adjacency matrix. In order to run the ABSIR function, we need to provide a connections matrix, where each cell provides the "connection" between two people. For example, the 7th column of the 3rd row represents the connection between person 3 and person 7. Because the connection between person A and person B is the same as the connection between person B and person A, the matrix is symmetric. A higher number in the cell represents a closer connection, and zero in the cell represents no connection.

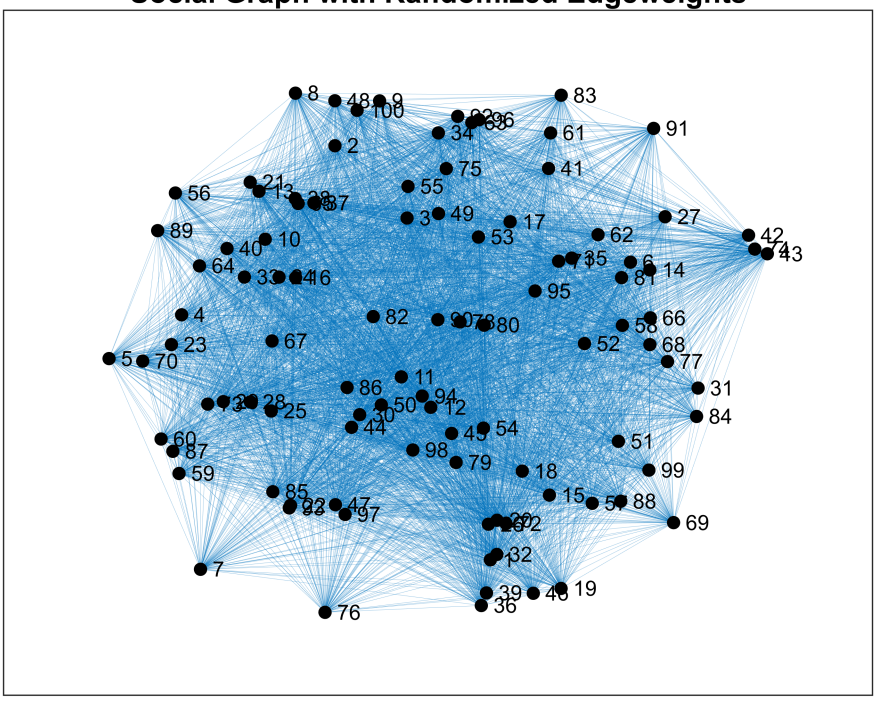
We decided to mimic a population of 100 people. Additionally, these connections are randomized to simulate the real world. In a real world community, people have varying connections between each other person in the community.

```
% create the randomized connections
% determine the size of the population
PopSize = 100;
```

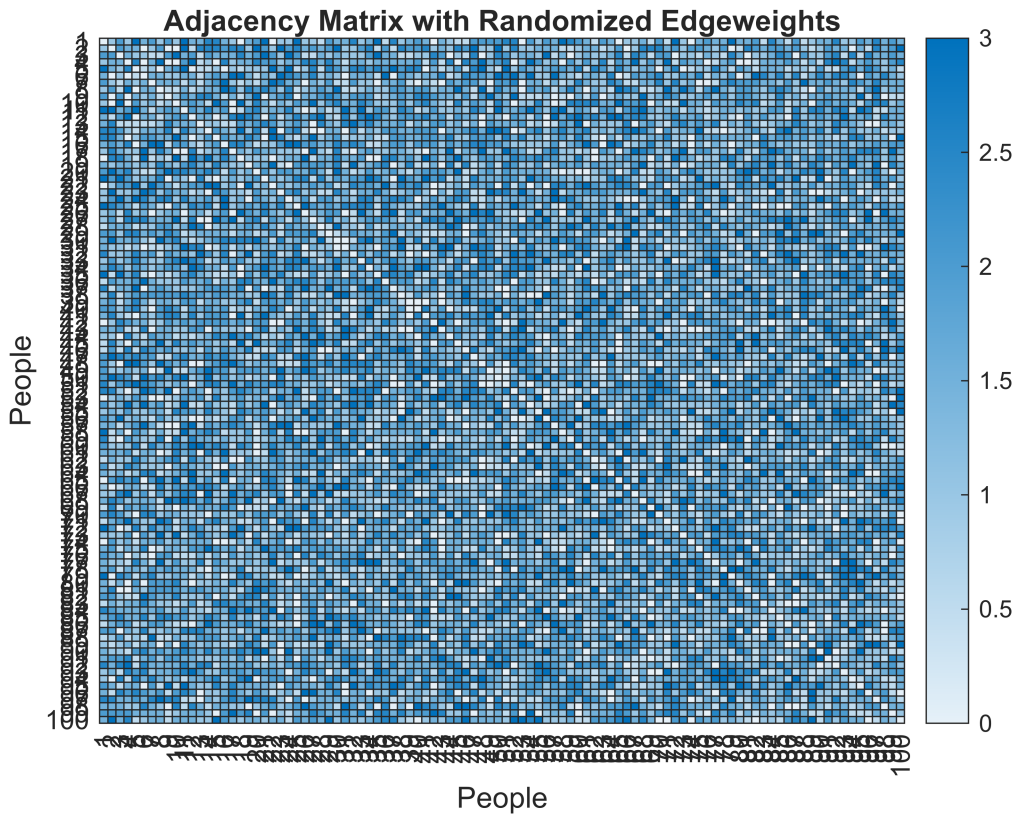
```
M_pop = (randi(4, PopSize) - ones(PopSize, PopSize));
% make the diagonal 0
M_pop(1:1+size(M_pop,1):end) = 0;
% make the matrix symmetric by taking the average of the corresponding
% values
M_pop = (M_pop + M_pop')/2
```

```
M_pop = 100×100
         0    1.0000    2.0000    1.0000    3.0000    1.5000    2.0000    1.0000 ···
    1.0000         0    2.0000    1.5000    1.5000         0    1.0000    2.0000
    2.0000    2.0000         0    1.5000    1.0000    1.0000    0.5000    0.5000
    1.0000    1.5000    1.5000         0    2.5000         0    1.5000    3.0000
    3.0000    1.5000    1.0000    2.5000         0    0.5000    1.0000         0
    1.5000         0    1.0000         0    0.5000         0    1.5000    1.5000
    2.0000    1.0000    0.5000    1.5000    1.0000    1.5000         0    1.5000
    1.0000    2.0000    0.5000    3.0000         0    1.5000    1.5000         0
    2.0000         0    2.0000         0    1.5000    1.0000    1.5000    0.5000
    1.0000         0    3.0000         0    2.0000    1.0000    1.5000    0.5000
     ⋮
```

```
% Visualize the graph structure
Graph_pop = graph(M_pop);
figure()
h = plot(Graph_pop, 'NodeColor', 'k', 'LineWidth', 0.1);
% NB. I use a graph layout algorithm that makes the distance between nodes
% inversely-proportional to the edge weight. This will tend to "clump" the
% pods together, which will make interpreting the graph visualization
% easier.
layout(h, 'force', 'WeightEffect', 'inverse')
title('Social Graph with Randomized Edgeweights')
```



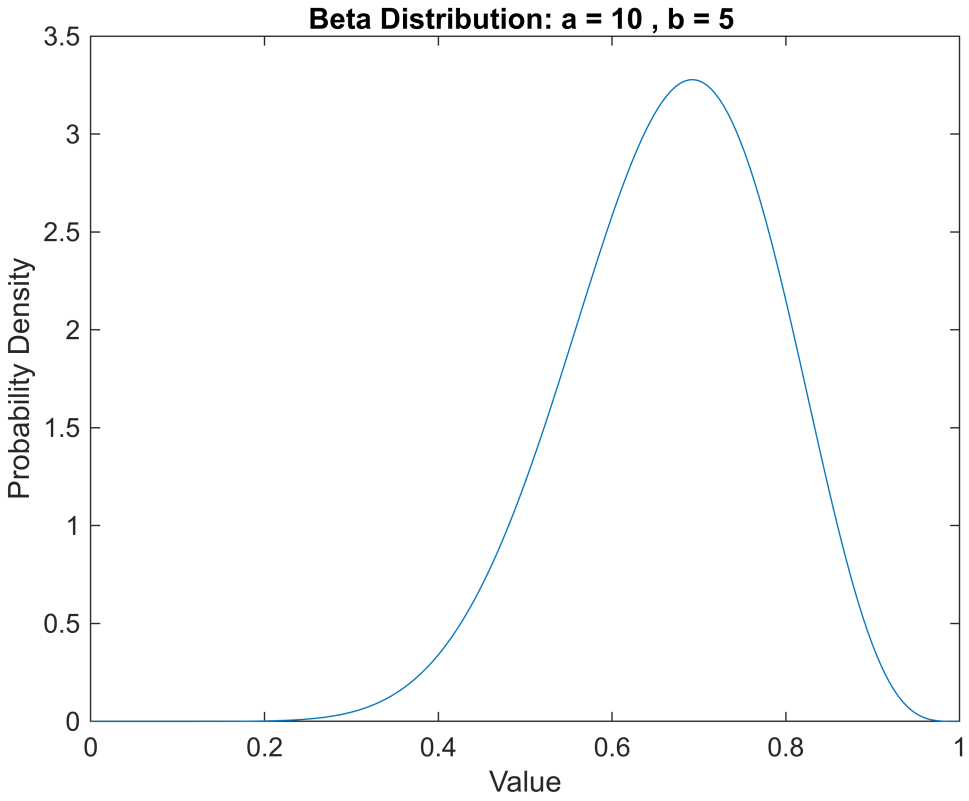

```
%Visualize the matrix as a heatmap
figure()
heatmap(M_pop)
title('Adjacency Matrix with Randomized Edgeweights')
xlabel('People');ylabel('People');
```



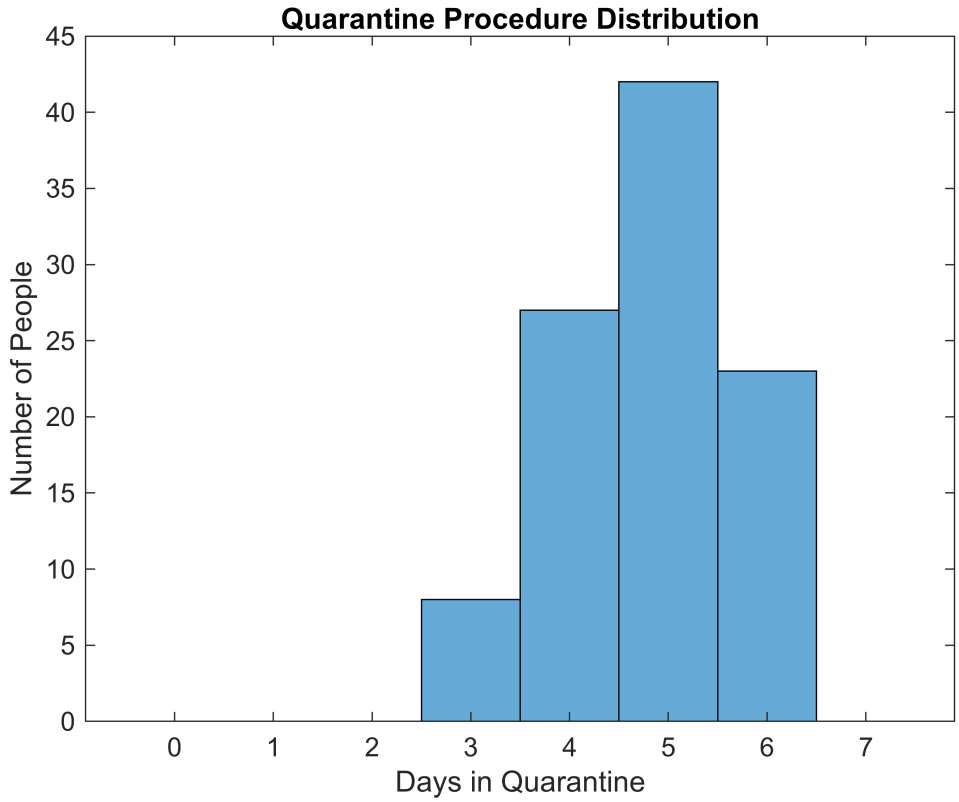
II. Create Beta distribution

Next, we need to generate our Beta distribution. Based on our research, we found that most companies recommended a five day quarantine period. However, not everyone will follow these guidelines; many people will quarantine for more, and many people will quarantine for less. Our distribution generates values between 0 and 7 and is a parameter for the function simulate_absir_mod. Inside the function, each time someone tests positive, they are assigned a random number of hours between 0 and 168 (or zero to seven days) according to the distribution. This number represents how many hours they will quarantine for. During this quarantine period, their connections in the adjacency matrix drop to zero. After this time period is over, regardless if they have recovered, they "re-enter" society and their connections return to normal.

```
% Initialize distribution
a = 10; % First shape parameter
b = 5; % Second shape parameter
pd_beta = makedist('Beta', 'a', a, 'b', b);

% Visualize
x = linspace(0, 1, 200);
pr = pd_beta.pdf(x);
figure();
plot(x, pr)
xlabel('Value')
ylabel('Probability Density')
title('Beta Distribution: a = 10 , b = 5 ')
```




```
B = random(pd_beta, 1, 100); % generate sample using distribution

X = (7-0)*B+0; % Transform the sample based on our research

% Visualize as a histogram
figure();
histogram(X,[-.5,.5,1.5,2.5,3.5,4.5,5.5,6.5,7.5])
xlabel('Days in Quarantine')
ylabel('Number of People')
title('Quarantine Procedure Distribution')
```

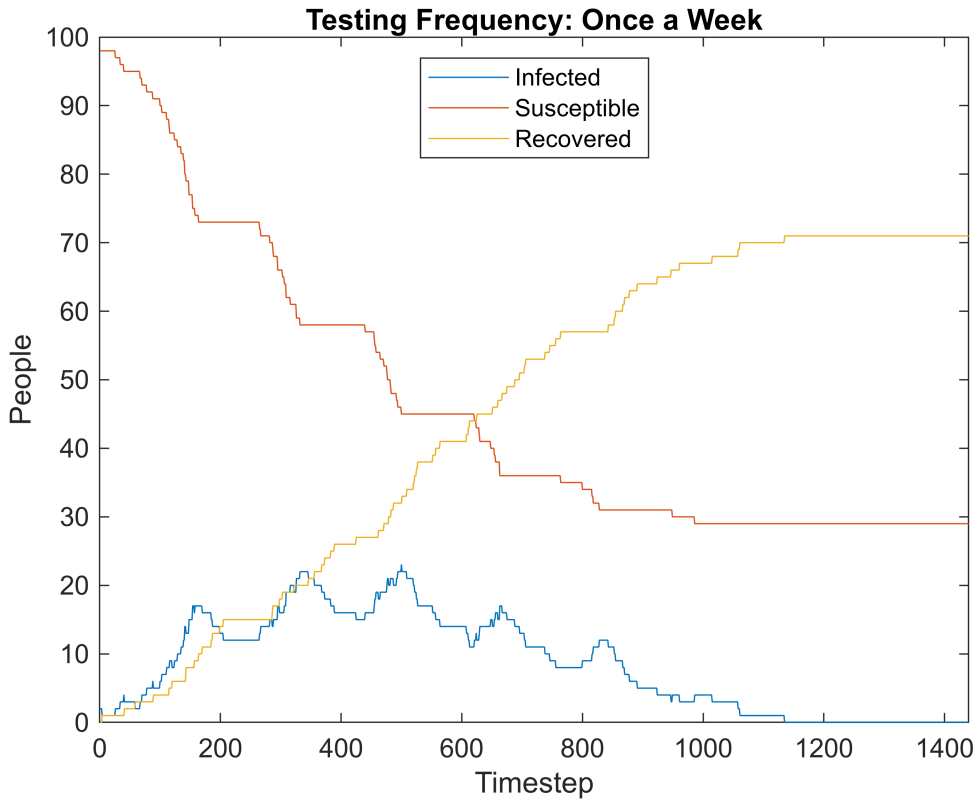


III. Run a Single ABSIR Realization

In order to demonstrate what the function simulate_absir.m does, we are running a single realization with a testing frequency of once a week. This will help us visualize our future results; however, this run of the model has no bearing on our final results.

```
T_simulation = 1440; % Initialize number of timesteps
infection_rate = .0002; % Initialize infection rate
recovery_rate = .006; % Initialize recovery rate
testing_frequency = 168; % Initialize example testing frequency
Iv0 = zeros(100,1); %setting up our base infection matrix
Iv0(1) = 1; Iv0(2) = 1; %infecting two individuals in our base infection matrix

% Run a single instance of simulate_absir_mod
[Sh, Ih, Rh] = simulate_absir_mod(M_pop, Iv0, T_simulation, infection_rate,
recovery_rate, testing_frequency, pd_beta);

% Assert that every person in each timestep for the control is either susceptible,
infected, or recovered
assert(isequal(ones(100,T_simulation),(Sh + Ih + Rh)))

I_count = sum(Ih, 1); %summing our number of infection people each timestep
S_count = sum(Sh, 1); %summing the number of susceptible people each timestep
R_count = sum(Rh, 1); %summing the number of recovered people each timestep
```

```
%plotting results
figure()
plot(1:T_simulation, I_count, 1:T_simulation, S_count, 1:T_simulation, R_count);
axis([0,1440,0,100])
legend('Infected', 'Susceptible', 'Recovered', 'Location','North')
title('Testing Frequency: Once a Week')
ylabel('People')
xlabel('Timestep')
hold off
```

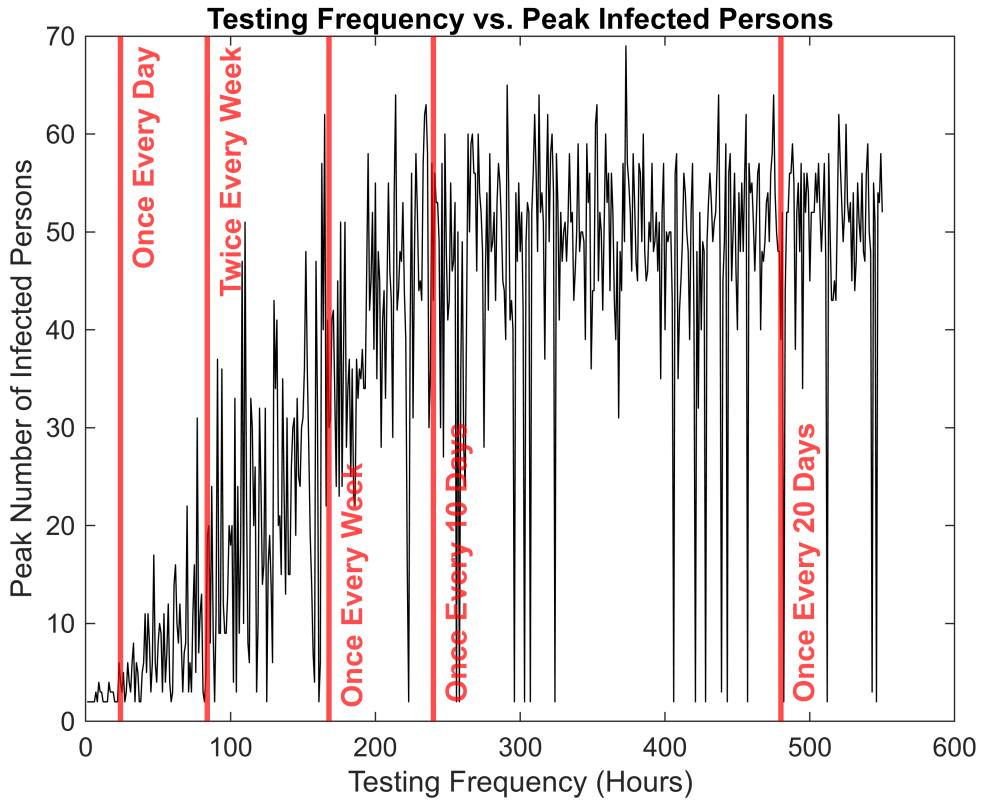


IV. Sweep Over Testing Frequencies

Our next step is to compare different testing frequencies to see which are the most effective. We chose to simulate testing frequencies between once every one hour and once every twenty days. Although both of these are unrealistic extremes, it is nonetheless important to consider a variety of test all values to see the overall relation between testing frequency and the spread of infection.

```
% allocate space for matrices, initialize different_frequencies
i_metric = zeros(1, 550);
different_frequencies = linspace(1,550,550);
I_ensemble = zeros(1,T_simulation);
R_ensemble = zeros(1,T_simulation);
S_ensemble = zeros(1,T_simulation);

% Parameter Sweep over different_frequencies
```

```
for i = 1:length(different_frequencies)

     % Run an instance of simulate_absir_mod
     [Sh, Ih, Rh] = simulate_absir_mod(M_pop, Iv0, T_simulation, infection_rate,
recovery_rate, different_frequencies(i), pd_beta);
     % Sum the infection history and find the peak number of infections
     I_ensemble(:) = sum(Ih, 1);
     i_metric(i) = max(I_ensemble);

     % Assert that every person in each timestep for the control is either
susceptible, infected, or recovered
     assert(isequal(ones(100,T_simulation),(Sh + Ih + Rh)))

     % Assert that the number of recovered people only ever increases
     % Assert that the number of susceptible people only ever decreases
     R_ensemble(:) = sum(Rh,1);
     S_ensemble(:) = sum(Sh,1);
     for t = 2:T_simulation
        assert(R_ensemble(t) >= R_ensemble(t));
        assert(S_ensemble(t) <= S_ensemble(t));
     end
end

% Visualize results with key testing frequencies highlighted
figure();
plot(different_frequencies,i_metric,'k'); hold on;
title('Testing Frequency vs. Peak Infected Persons')
ylabel('Peak Number of Infected Persons'); xlabel('Testing Frequency (Hours)');
xline(24, 'r', {'Once Every Day'}, 'fontsize', 11, 'FontWeight', 'bold',
'linewidth', 2)
xline(84, 'r',{'Twice Every Week'}, 'fontsize', 11, 'FontWeight', 'bold',
'linewidth', 2)
xline(168, 'r',{'Once Every Week
'}, 'fontsize', 11, 'FontWeight', 'bold', 'linewidth', 2)
xline(240, 'r',{'Once Every 10 Days
'}, 'fontsize', 11, 'FontWeight', 'bold', 'linewidth', 2)
xline(480, 'r',{'Once Every 20 Days
'}, 'fontsize', 11, 'FontWeight', 'bold', 'linewidth', 2)
hold off;
```

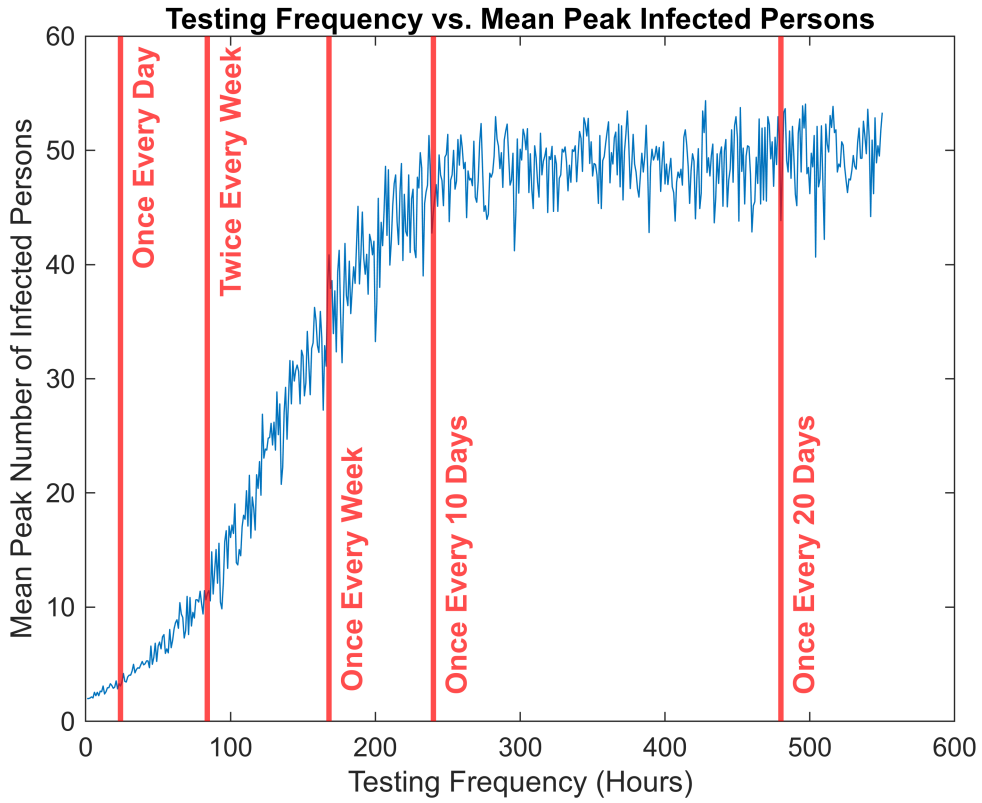


V. Creating Multiple Realizations:

In order to maximize the accuracy of our data, we are running simulate_absir multiple times to create several realizations of our data. This is due to the fact that simulate_absir utilizes randomness within each realization that it runs, as does our quarantine time. To find the most accurate/applicable data, we are finding the mean of our realizations. This allows us to better visualize our data and increase the consistency of our data.

```
% allocate space for matrices
i_metric_mean_data = zeros(20,550);

% run 20 realizations of parameter sweeps
for j = 1:20

    %parameter sweep over different_frequencies
    for i = 1:length(different_frequencies)

        % run a realization of simulate_absir_mod
        [Sh, Ih, Rh] = simulate_absir_mod(M_pop, Iv0, T_simulation, infection_rate,
recovery_rate, different_frequencies(i), pd_beta);
        % Sum the infection history and find the peak number of infections
        I_ensembleCon(:) = sum(Ih, 1);
        i_metric(i) = max(I_ensembleCon);

        % Assert that every person in each timestep for the control is either
susceptible, infected, or recovered
```

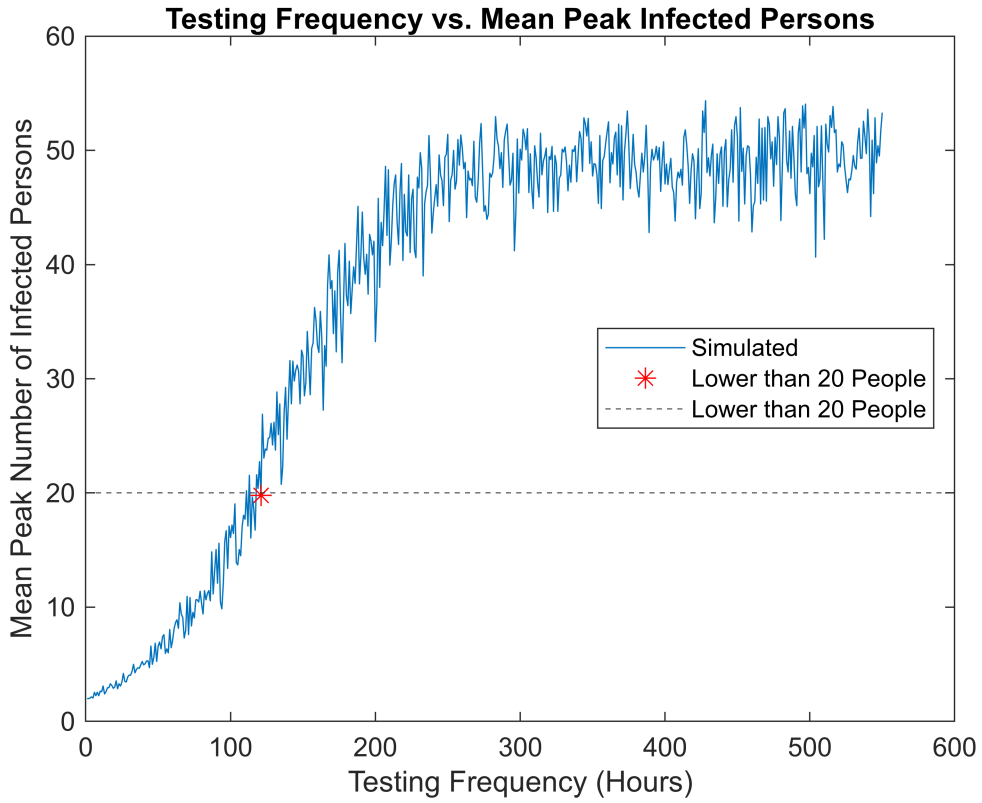
```
        assert(isequal(ones(100,T_simulation),(Sh + Ih + Rh)))

        % Assert that the number of recovered people only ever increases
        % Assert that the number of susceptible people only ever decreases
        R_ensemble(:) = sum(Rh,1);
        S_ensemble(:) = sum(Sh,1);
        for t = 2:T_simulation
           assert(R_ensemble(t) >= R_ensemble(t));
           assert(S_ensemble(t) <= S_ensemble(t));
        end
    end
    % Store peak values in a matrix
    i_metric_mean_data(j,:) = i_metric(:,:);
end
% Find the mean of the peak infections
I_mean = mean(i_metric_mean_data, 1);


% Visualize mean data
plot(different_frequencies,I_mean);
title('Testing Frequency vs. Mean Peak Infected Persons')
ylabel('Mean Peak Number of Infected Persons'); xlabel('Testing Frequency (Hours)');
xline(24, 'r', {'Once Every Day'}, 'fontsize', 11, 'FontWeight', 'bold',
'linewidth', 2)
xline(84,  'r',{'Twice Every Week'}, 'fontsize', 11, 'FontWeight', 'bold',
'linewidth', 2)
xline(168, 'r',{'Once Every Week
'}, 'fontsize', 11, 'FontWeight', 'bold', 'linewidth', 2)
xline(240, 'r',{'Once Every 10 Days
'}, 'fontsize', 11, 'FontWeight', 'bold', 'linewidth', 2)
xline(480, 'r',{'Once Every 20 Days
'}, 'fontsize', 11, 'FontWeight', 'bold', 'linewidth', 2)
```



VI. Adressing our Modeling Question:

In order to answer our modeling, we are determining exactly how often tests must be administered for the peak number of infected people to be lower than twenty.

```
% Plot Testing Frequency vs. Mean Peak Infected Persons
figure()
plot(different_frequencies, I_mean, 'DisplayName', 'Simulated')
xlabel("Testing Frequency (Hours)")
ylabel("Mean Peak Number of Infected Persons")
legend('Location', 'East')
title("Testing Frequency vs. Mean Peak Infected Persons")

% Find the highest frequency that results in a peak number of infected people lower
than 20
g = 20;
h = I_mean < g;
i = different_frequencies(h);
j = I_mean(h);
k = j(end);

% Plot marker at correct frequency
hold on;
plot(i(end), k, 'r*', 'MarkerSize', 8, 'DisplayName', 'Lower than 20 People');
yline(20,'--', 'DisplayName', 'Lower than 20 People');
```




```
% Find exactly how often testing is needed to lower peak below 20
i(end)
```

```
ans = 121
```

VII. Model Validation:

The peer-reviewed research paper "Frequency of Routine Testing for Coronavirus Disease 2019 (COVID-19) in High-risk Healthcare Environments to Reduce Outbreaks", published in Oxford's scientific journal Clinical Infectious Diseases closely mimics the results found with our model. In their model, they create a model of (basic reproduction number or the number of secondary infections) in a relationship with testing frequency. In examining this model and their results, they concluded that routine testing substantially reduces risk of outbreaks, but may need to be as frequent as twice weekly". Similarly, we found that testing on a regular basis significantly lowers the spread of disease... but only if tests are administered more than once a week. If tests are administered at a rate of once a week or less frequently, there is little to change in the spread of disease. Additionally, in order to highly contain the disease, testing should be administered at least twice a week, as this reduces the peak by 75%. Because our model produces very similar results to a model in a peer reviewed, highly renowned scientific journal, we can safely assume our model's results are highly accurate.

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7797732/

Section III: Results

To best answer our question, we need a parameter sweep over various testing frequencies. To best demonstrate the connection between testing frequency and the spread of disease, we chose the metric of peak number of infected persons in correlation with testing frequencies. As our system represents a community of essential workers, it is vital that we "flatten the curve" and slow the infection, making the peak number of infected persons a perfect choice.

Metrics and Sweeps:

- Testing every hour.....2 peak people infected
- Testing every day.....3 peak people infected
- Testing twice a week.....11 peak people infected
- Testing every week.....41 peak people infected
- Testing every ten days.....45 peak people infected
- Testing every twenty days.....50 peak people infected

In order to lower the peak infection below twenty people, the testing frequency can be at most once every 121 hours.

Analysis:

How often do we need to test in order to have the peak number of infected people lower than 20?

As expected, **testing on a more frequent basis leads to a lower peak number of infected people**, flattening the curve. In order to lower the peak below 20 people, we need to test at least once every 121 hours. However, since this is only a best estimate, **we reccomend testing twice a week to guarantee that the peak is lower than 20 people.**

Additionally, it should be noted that testing on a more frequent basis only signifigantly lowers the peak over the time period of around 0-10 days. If tests are less frequent than once every ten days, then the number of tests no longer correlates with the peak number of infected people. This makes complete sense, as we modeled an infection that lasts for a period of seven days on average, where the recommended quarantine is five days. If a person is able to become infected, spread the disease, and completely recover without ever needing to test and therefore quarantine, then testing regularly becomes significantly less effective or even entirely ineffective.

Section IV: Interpretation

Original Question:

How often do we need to test in order to have the peak number of infected people lower than 20?

Answer:

In order to lower the peak below 20 people, we need to test at least once every 121 hours. However, since this is only a best estimate, we reccomend testing twice a week to guarantee that the peak is lower than 20 people.

ABT:

The effect of the testing can clearly be seen in the first simulation output graph, where testing was conducted once a week. After an initial infection peak, the infected line sharply decreases as people quarantine themselves. This is followed by slight residual infection peaks as groups of people come out of isolation. The frequency of testing has a large effect on the mean infection peak as shown in our parameter sweep. Testing less frequently leads to a much more widespread infection, until you exceed 10 days without them and the sweep begins to level off. Obviously, if you wish to reduce the spread of the disease as much as possible, you would also be testing as frequently as possible. It is not reasonable however, to expect people to test themselves multiple times a day or even every day. Ideally, everyone would test themselves twice a week, as this would bring us below our 20 person threshold.

Limitations:

The first limitation comes in the form of our population matrix. The minimum edge weights are obviously zero, as the least amount of interaction two people can have is no interaction, but the maximum weights are completely arbitrary and not based on any verification. Additionally, there are probably not enough people with zero connection to accurately model reality, where most people don't know each other. Another limitation is the beta distribution used to determine quarantine length. In real life, many or even most persons will not closely follow the recommended quarantining procedures. Many people may simply disregard them entirely and not isolate at all, or not be informed with the most up to date procedures from the CDC and isolate for an improper amount of time. Finally, having the infection and recovery rates be the same for all persons is a gross inaccuracy, but a necessary one given the scope of this project.

Next Steps in Model Development:

We would like to delve deeper into how these differing quarantine procedures affect the initial outbreak of an infection. More specifically, how often can frequent testing and quarantine completely shut down an infection before it ever has a chance to spread.

```
function [Sh, Ih, Rh] = simulate_absir_mod(M, Iv0, T, infection_rate, recovery_rate, ↙
testing_frequency, quarantine_dist)
% Simulate agent-based transmission model. Uses a graph to represent social
% connectivity between agents.
%
% Note: You may find it helpful to summarize the state history matrices via
% summation. For instance sum(Ih, 1) will return the total number of
% infected persons at each timestep in the simulation.
%
% Inputs
%   M (square matrix): Adjacency matrix
%   Iv0 (column vector): Initial infection state
%   T (integer): Number of timesteps to simulate in hours
%   infection_rate (float): Infection rate (probabilistic)
%   recovery_rate (float): Recovery rate (probabilistic)
%   testing_frequency (float): Hours between tests are taken
%   quarantine dist
%
% Returns
%   Sh (matrix): Susceptible state history
%   Ih (matrix): Infected state history
%   Rh (matrix): Recovered state history

    % Setup
    dim = length(Iv0);  % Dimensions of initial state
    Ih = zeros(dim, T); % Infection history
    Posh = zeros(dim, 1);
    time_to_break = zeros(dim,1);
    Ih(:, 1) = Iv0; % Record the initial state to infection history
    Rh = zeros(dim, T); % Recovery history
    M2 = M;

    % Construct an "action" helper function
    function [I, R] = action(I, R, i)

        % Check if this timestep is a timestep where tests are administered
        if mod(i, testing_frequency) == 0


            Posh(:) = I(:);    % Mark all infected people as testing positive
            Lpos = logical(Posh(:));    % Covert the testing positive people to a ↙
logical array

            % Assign each person who tested positive a quarantine time
            % based on the quarantine_dist
                for j = 1:dim
                    time_to_break(j) = Lpos(j)*random(quarantine_dist, 1, 1)*168;
                end
        end
```

```
        % Lower the quarantine time by a single timestep (hour)
        time_to_break(:,:) = time_to_break(:,:) - 1;

        % If quarantine time is negative, set time to zero
        for k = 1:length(Iv0)
            if time_to_break(k)<0
                time_to_break(k) = 0;
            end
        end

        Lbreak = logical(time_to_break); % Create a logical vector (all non zero quarantine times are "True")
        M(:) = M2(:);   % Reset Matrix
        M(:,Lbreak) = 0; M(Lbreak,:) = 0; % Set all connections of quarantined people to zero

        % Compute infection probabilities based on the social graph
        v_eff = infection_rate * M * I;
        v_pr = v_eff ./ (1 + v_eff);

        % Draw random values
        v_infect = rand(dim, 1) <= v_pr;
        v_recover = rand(dim, 1) <= recovery_rate;

        % Infect non-recovered individuals
        I = I | v_infect & (~R);
        % Recover infected individuals
        R = R | (I & v_recover);
        I = I & (~R);
    end

    % Run simulation
    for i = 2:T
        [Ih(:, i), Rh(:, i)] = action(Ih(:, i-1), Rh(:, i-1), i);
    end

    % Compute susceptible history
    Sh = ones(dim, T) - Ih - Rh;
end
```