

COE 322 FINAL PROJECT

DISEASE PROPAGATION MODELING USING C++

December 10, 2018

Austin Benny (ab62524)

University of Texas at Austin
austinbenny@utexas.edu

Contents

1	Introduction	2
2	Joe: Individual person	2
2.1	Code	3
2.2	Output	4
2.3	Results	4
2.4	Discussion	4
3	Population	5
3.1	Shortcomings	6
3.2	Code	7
3.3	Output	7
3.4	Results	8
3.5	Discussion	10
4	Conclusion	12
5	Appendix	14
5.1	Person.h	14
5.2	Population.h	15
5.3	final.m	19
5.4	repeat.m	20
	References	21

1 Introduction

In this project, students are tasked with building an explicit simulation of disease propagation using C++. The person is an object, and the population is an object that is a collection of person objects. We begin by infecting one person and tracking their status over a number of days. This simple model is scaled up to contain a collection of people where each individual's status is monitored. An infection and random interactions are then introduced to make the simulation more realistic. The program will then track the population from day to day, running indefinitely until none of the population is sick. Since there is no re-infection, the run will always end.

The foundational model of the project is as follows:

1. Sick \Rightarrow they can infect other people
2. Susceptible \Rightarrow they can be infected
3. Recovered \Rightarrow they have been sick, but no longer carry the disease and cannot be infected
4. Inoculated \Rightarrow they do not carry the disease and can not be sick

More details about the project can be found [here](#).

2 Joe: Individual person

For the first section of the project, a model of one person with probability of infection is introduced. The default number of days a person can be sick is set to 5 days and remains that way to the end of the program. The main methods serve to infect a person, and to track their state. The other methods serve to gauge the person's status. The methods of the `Person` class are as follows:

1. `status_string()` : returns a description of the person's state
2. `update()` : update the person's status to the next day
3. `infect(n)` : returns a description of the person's state
4. `is_stable()` : report whether the person has been sick and is recovered

2.1 Code

Below is the main code for the first section of the experiment, the `Person.h` & `Population.h` files along with all the MATLAB codes are in the appendix.

```
#include<iostream>
#include<vector>
#include<random>
#include<string>

#include "person.h"//header files for class person and population
#include "population.h"

using namespace std;

int main(){

    float tolerance = .15;//probability Joe will be infected
    cout << "Default probability of infection is " << tolerance << endl;
    int day = 0;

    person joe;
    joe = person();

    while(joe.is_stable() != true){
        day++;
        joe.update();
        float bad_luck = (float) rand()/(float)RAND_MAX;
        if (tolerance > bad_luck){//if tolerance is 80%, there is an 80%
            chance that it is greater than bad_luck
            joe.infect();
            cout << " Joe on day, " << day << " "; joe.status_string();
            cout << endl;
        }
        else{
            cout << "Joe on day, " << day << " "; joe.status_string();
            cout << endl;
        }
    }
    return 0;
}
```

2.2 Output

```
.  
.   
.   
Joe on day, 12  is susceptible  
  
Joe on day, 13  is susceptible  
  
Joe on day, 14  is susceptible  
  
Joe on day, 15  is sick and will be for 5 more days  
  
Joe on day, 16  is sick and will be for 4 more days  
  
Joe on day, 17  is sick and will be for 3 more days  
  
Joe on day, 18  is sick and will be for 2 more days  
  
Joe on day, 19  is sick and will be for 1 more days
```

2.3 Results

Probability(%)	10	20	30	40	50	60	70	80	90	95
Days	25	10	10	7	7	7	7	5	5	5

Table 1: Affect on days as probability is varied

2.4 Discussion

As Table 1 shows, as the probability of infection is increased in 10% increments, the number of days it takes Joe to become inoculated decreases because he gets infected faster. So if the tolerance is 10% I would expect the `if(tolerance > bad_luck)` condition to be `true` later on in the run-time of the program; whereas, if the tolerance were 70% I would expect the if condition to be met sooner than if the tolerance were 10%. As the probability of infection is 80%, Joe immediately gets infected and the number of days will be 5 because it takes Joe 5 days by default to be recovered after sick. I would expect the number of days to be constant at 5 after the probability of 80% because if the probability generated at the iteration that corresponds to when Joe gets infected, increasing the probability won't change anything. This is because the program is using a quasi-random number generator, so essentially the random numbers created are the same for every new run of the program. To better show this result, please see Figure 1.

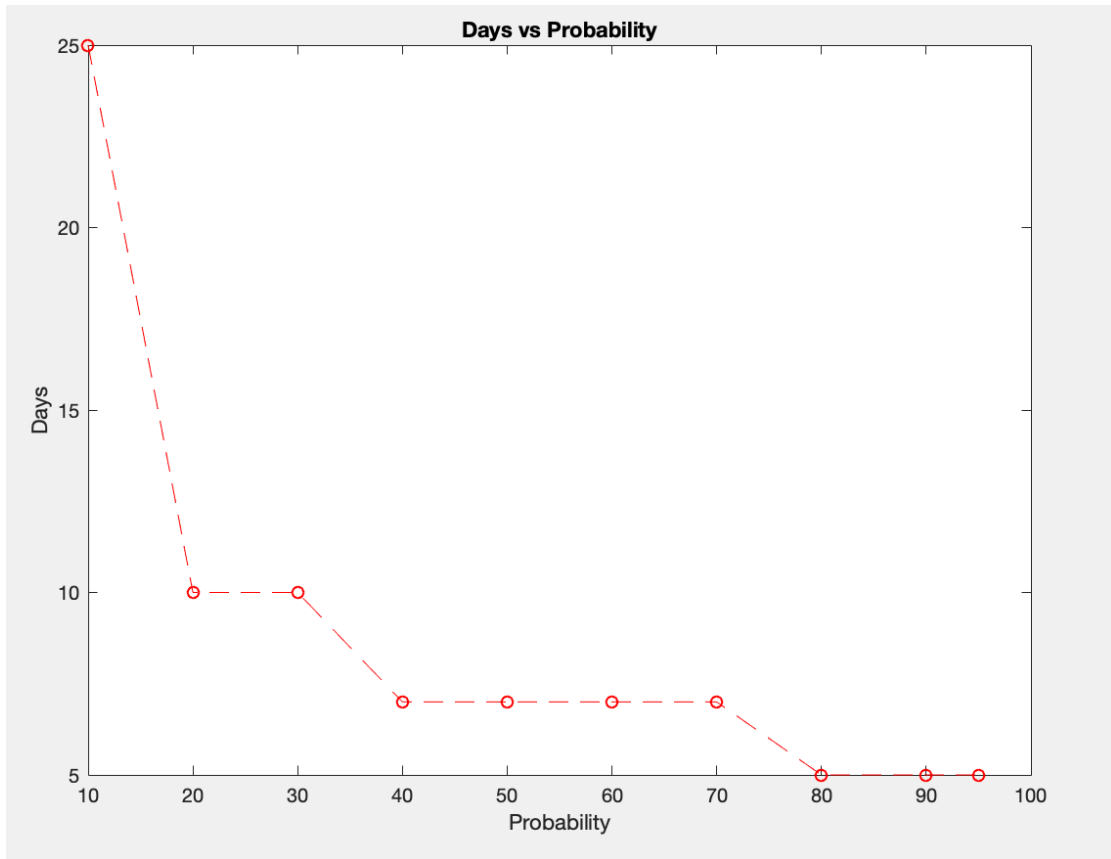


Figure 1: As the probability is increased, the number of days Joe isn't recovered should decrease. This is proved by the simulation.

3 Population

The bulk of the report centers around how the disease spreads in the population. To model the spreading, a form of the The SIR model proposed by Kermack and McKendrick in 1927 comes is employed. The program starts out with infecting one random person in a vector of **Person** objects of a user-defined **size** passed to the population class. This random person has the ability to infect their neighbors if a certain **probability** $\in [0, 1]$ is met. If the neighbors are recovered or inoculated, that neighbor cannot be infected. Each infected person has contact with **n** other people in the population per day. Before infecting anyone, a certain **percentage** of the population is inoculated.

As we can see from the bolded items above, there are 4 parameters to vary at 3 relative conditions (low, medium, and high) for a maximum of 64 set of data or graphs able to be created from the simulation. The simulation can be described as

$$f(\textit{Probability}, \textit{Percentage}, n, \textit{Size}) = [S, I, R, IN]$$

Where the right-hand side corresponds to the number of susceptible, infected, recovered, and inoculated people predicted by the initial conditions. 64 graphs is time consuming and most of those graphs will not inform anybody. For example, It is obvious that if the percentage of inoculated people in my population is high (95%) and the probability of transfer of the infection is low (5%) the number of infected people in my population (with all things held constant) will be relatively low. So running this simulation will not be worthwhile; instead, in the Discussion section attention is given to the peculiar and enlightening trends from varying the parameters, especially the phenomenon of herd immunity.

3.1 Shortcomings

At certain sections in the simulation, the limitations of C++ become irksome. For example, when the population size is medium (≈ 100 people), and the inoculation percentage is relatively high ($\approx 80\%$), the program will only randomly infect ≈ 50 people. For example, If the simulation has a size of 100 and the inoculation percentage is 80% as given above, the program will only infect 50 people instead of 80. For extremely high ($\approx 95\%$) the program fails to move to the inoculation stage due to a segmentation fault. The output of the `repeat.m` MATLAB script demonstrates that for the array of random numbers between the population size (100 in this case), a little less than half of the numbers in the array are repeated again.

```
%Size of A is 75

A = [84,87,78,16,94,36,87,93,50,...
     22,63,28,91,60,64,27,41,27,73,...
     37,12,69,68,30,83,31,63,24,68,...
     36,30,3,23,59,70,68,94,57,12,43,...
     30,74,22,20,85,38,99,25,16,71,...
     14,27,92,81,57,74,63,71,97,82,6,...
     26,85,28,37,6,47,30,14,58,25,96,83,46,15];

.
.
.
>> repeat
Size of B is 50
```

B is a vector that employs the built-in MATLAB function `unique()` that extracts only the elements in A that are non-repeating. So, A, the array of size 75 has 15 repeating elements. Since the simulation criteria prohibit inoculating someone who is already inoculated, the 15 repeating numbers are ignored. This reduction in the final people inoculated is most likely because the seed for the random number generator is not changed or updated at every trial. Therefore, even if the size of the population is increased to 1000 people, the number of inoculated will be $\approx 50\%$ of that or around 500 people. This is a limitation of not only C++ but any programming language with a quasi-random number generator.

3.2 Code

```
#include<iostream>
#include<vector>
#include<random>
#include<string>

#include "person.h"//header files
#include "population.h"

using namespace std;

int main(){

    int pop_size = 100; //size of population
    vector<person> group(pop_size, person());
    population pop(group);

    return 0;
}
```

3.3 Output

```
The probability is: 0.85
The percentage of population to inoculate is : 0.05
Population size is: 100
The number of people contacted per day: 6
-----
Day: 1
```



```
Infected: 1
Recovered: 0
Susceptible: 94
Inoculated: 5
```

```
-----
Day: 2
Infected: 13
Recovered: 0
Susceptible: 82
Inoculated: 5
-----
```

```
.
.
.
```

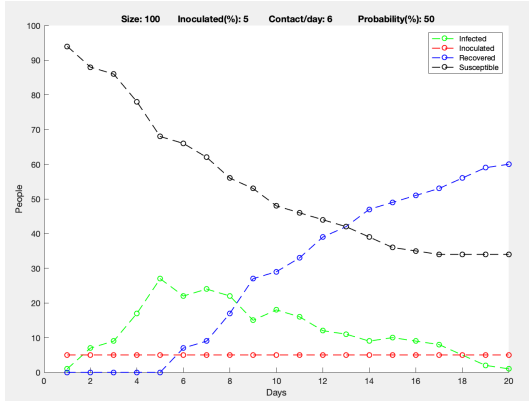
```
Day: 11
Infected: 9
Recovered: 52
Susceptible: 34
Inoculated: 5
```

```
-----
Day: 12
Infected: 5
Recovered: 56
Susceptible: 34
Inoculated: 5
```

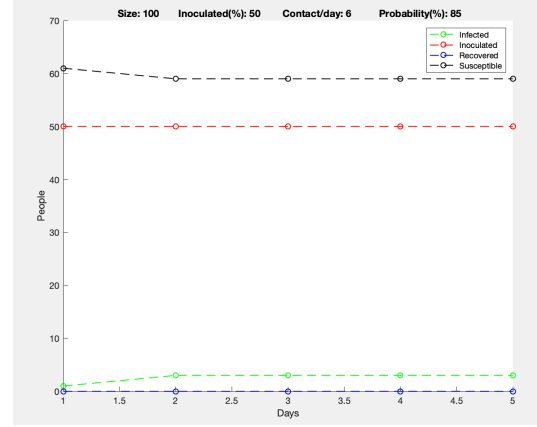
Disease ran its course in the 13 day

3.4 Results

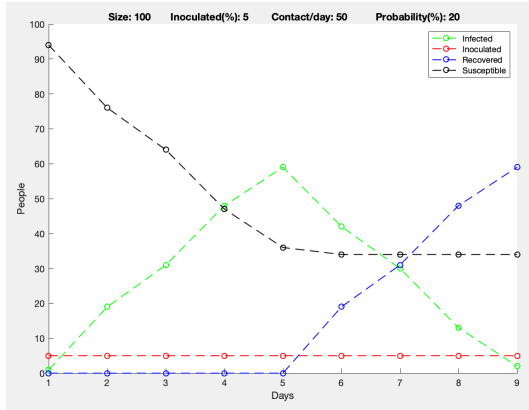
Below are the plots created from the data by varying several parameters of the program with the population size held constant (100 people). Please note, that since the program begins with `while(count_infected() != 0)`, after the `while()` conditions is met, the program stops getting data on the number of inoculated, recovered, and sick. So the final marker on the x-axis represents the conditions of the people the day before everyone was recovered.



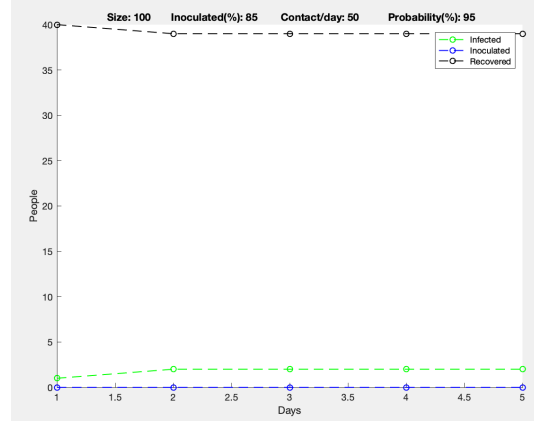
(a) Low inoculation & Moderate transfer



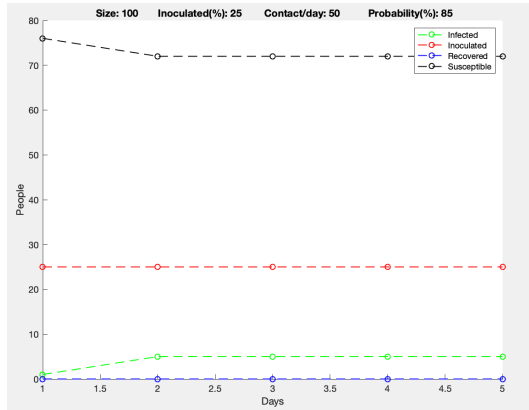
(b) Half inoculation & high transfer



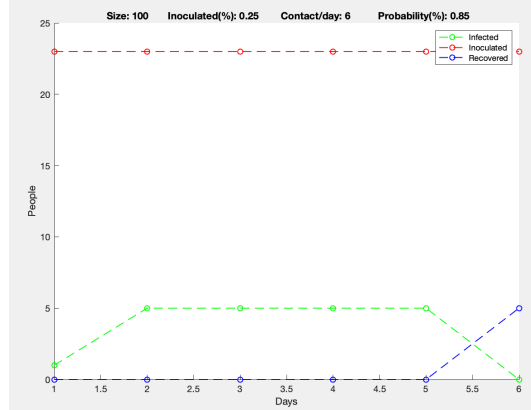
(c) Low inoculation & low transfer & high contact/day



(d) High inoculation & high transfer & high contact/day



(e) Moderate inoculation & and high transfer



(f) Moderate inoculation & high transfer

Figure 2: Varying probability of transfer (transfer), contact per day (contact/day), and percentage of inoculated (inoculation)

3.5 Discussion

Figure 2a represents the control result; this graph represents the results when the disease is given an opportunity to run through the population without a major impedance such as a high proportion of inoculated in the population. Since the percentage of inoculated is 5%, 5 people in the population will be inoculated and stay that way throughout the end of the program, represented by the red dotted line. The number of infected rises steadily as patient-zero is given an opportunity to infect his neighbors, and since a relatively small number of the population are inoculated, more and more people will be infected. Day 5 marks the inflection point for the infected population, the green dotted line steadily decreases from a high of about 30 people. As a consequence of this, the number of people recovered increases and the number susceptible people decrease as shown by 2a. The number of infected decreases after day 5 because at that point, around half the population is either infected, inoculated, or recovered and the program only has a 50% chance of infecting a susceptible person.

Figure 2b represents the case where a half the population is infected and the disease has a high probability of transfer. In this case, the program infects 3 people on day 2 and fails to infect anybody else. Analogously, Figure 2e represents the case where more than half the population is inoculated and the disease has a high probability of transfer and looks very similar to Figure 2b.

On the converse of that, Figure 2f represents the case where a moderate proportion of the population is infected and the disease has a high probability of transfer. This result is odd, because even though only 25% of the population are inoculated, only a total of 5 people are ever infected through the course of the program. I would expect the script, when randomly choosing a `Person` to infect from the population, would have a good probability of selecting another soul at every iteration since only 25% are inoculated.

To further investigate this result, the number of people contacted per day was increased from 6 to 50. Figure 2c represents the other control case where a low percentage of the population are inoculated and the disease has a low probability of transfer. This trends in this graph are similar to the ones in Figure 2a with one stark difference. Since the number of people contacted per day is so high, one day in this population produces a large number of infected as evidenced by the peak number infected at 60. This peculiar result is driven home by Figure 2d; this graph looks very similar to Figure 2e as the number of infected, even when one person comes in contact with 50 other people, peaks at 5. Furthermore, the percentage of the population inoculated is set to 25% so I would expect the program to infect more than 5 people before eradicating the disease.

Figure 2b, 2d, 2e, and 2f are a consequence of what epidemiologists colloquial refer to as “herd immunity”. Herd immunity is a form of indirect protection from infectious disease that occurs when a decent percentage of a population has become inoculated, thereby providing a measure of protection for individuals who are not immune [2]. In a

population in which a large number of individuals are immune, chains of infection are likely to be disrupted, which stops or slows the spread of disease [3]. The greater the proportion of individuals in a community who are immune, the smaller the probability that those who are not immune will come into contact with an infectious individual [2]. For example, since 25% of the population are inoculated in the population representing Figure 2d, the other 75% are protected by herd immunity, hence the low total of infected persons. The health of the inoculated disrupts the chain of infection going around this population.

Holding the number of people contacted per day and the probability of transfer constant at 6, and 85%, the percentage of the population inoculated is varied. This relationship can be described as follows:

$$f(Percentage) = [Infected]$$

Table 2 contains the results.

Inoculated(%)	10	15	20	25	30	35	40	50	60
Peak Infected	27	5	16	5	7	12	8	4	5

Table 2: Affect on the highest number of infected as probability is varied

As the trend-line in Figure 3 clearly shows, as the percentage of inoculated is increased, the peak number of infected decreases. The figure also shows that after about 35% the peak number of infected never rises over that value of 12. Therefore, it is reasonable to conclude that if 30-35% of the population is inoculated, the disease will have a difficult time propagating even under a high probability of transfer. It is also import to note that the program predicted a peak number of infected of 5 and 7 at 15% and 25% respectively. However, this can be attributed to the randomness of the random number generator.

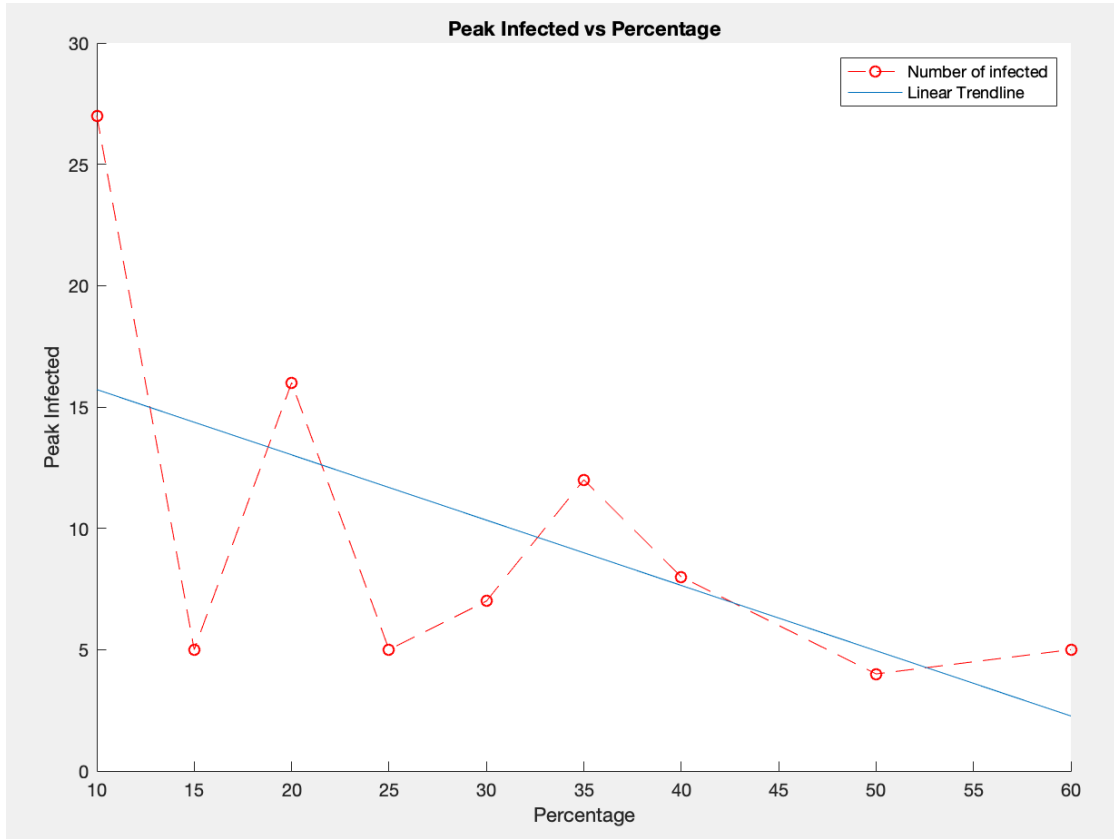


Figure 3: Linear trend-line showing the inversely proportional relationship between percentage of infected and peak number of infected.

4 Conclusion

Not everybody in a population has access to vaccines and excellent medical care, but as this simple SIR simulation has shown, if a decent proportion of the population (in this case $\approx 30\%$) is inoculated they can act as a shield for their susceptible counterparts. When infectious diseases are spreading, it is vital for those members of that population who can be inoculated to be inoculated. If the members of the population are lethargic and *Laissez-faire* a large number of people will be infected and the population will take considerable amount longer to eradicate the infection as evidenced by Figure 2a.

It is also important to note that as infected members of the population contact an excessive amount of people per day, the number of infected people in the population will naturally have a huge chance of increasing as evidenced by 2c. To thwart this result, quarantines and sick days are employed.

Each person is different; socio-economic status and access to healthcare, proclivity

towards disease, concentration of infected people in the vicinity, variations in individuals understanding of propagation, and many other factors must be employed to create a useable model. Even though this model is too simplistic for real world epidemics, it provides an excellent introduction to the propagation of disease and computational models in epidemiology.

5 Appendix

5.1 Person.h

```
#include<iostream>
#include<vector>
#include<random>
#include<string>
#include <cmath>//maybe use for floor()

using namespace std;

class person{
protected://should I make these public?
    int infect_days, age;
    string status;
public:
    person(){
        status = "susceptible";
        age = 0;
    }
    string get_status(){
        return status;
    }
    void set_status(string condition){
        status = condition;
    }
    void infect(){
        if (status == "susceptible"){
            status = "sick";
            infect_days = 5;//default 5 days
        }
    }
    void update(){
        age++;
        if (status == "sick" && infect_days != 1){
            infect_days --;
        }
        else if(status == "sick" && infect_days == 1){
            status = "recovered";
        }
    }
    void status_string(){
        if (status == "sick"){
            cout << " is sick and will be for " << infect_days << " more
                days" << endl;
        }
    }
}
```

```

        else{
            cout << " is " << status << endl;
        }
    }
    bool is_stable(){
        if (status == "recovered" || status == "innoculated"){
            return true;
        }
        else{
            return false;
        }
    }
};

```

5.2 Population.h

```

#include<iostream>
#include<vector>
#include<random>
#include<string>

#include "person.h"

using namespace std;

class population : public person{
public:
    int num_people, sick_number, rand_person, people_contacted,
        next_position, inoculated_number,
        recovered_number, susceptible_number, days, contact;
    float probability, perc;
    vector<int> position_of_infected, people_to_innoculate;
    vector<person> v_persons;

    population(vector<person> men){
        v_persons = men;
        num_people = v_persons.size();
        update_pop();
    }
    void random_infection(){
        int healed, total = 0;
        for (auto &e: v_persons){

```



```

        if (e.get_status() == "recovered" || e.get_status() ==
            "innoculated"){
            healed++;
        }
    }
    total = num_people - healed; //if total = 0 the everyone is recovered
    rand_person = (int)(rand()%num_people) + 1; //random int person in
    vector
    if (total == 0){
        cout << "No more people to infect " << endl;
    }
    else if (v_persons[rand_person].get_status() == "susceptible"){
        v_persons[rand_person].infect();
    }
    else{
        random_infection(); //call function till someone is infected
    }
}

int count_infected(){
    sick_number = 0;
    for (auto &a: v_persons){
        if (a.get_status() == "sick"){
            sick_number++;
        }
    }
    return sick_number;
}

int count_inoculated(){
    innoculated_number = 0;
    for (auto &b: v_persons){
        if (b.get_status() == "innoculated"){
            innoculated_number++;
        }
    }
    return innoculated_number;
}

int count_recovered(){
    recovered_number = 0;
    for (auto &c: v_persons){
        if (c.get_status() == "recovered"){
            recovered_number++;
        }
    }
    return recovered_number;
}

int count_susceptible(){
    susceptible_number = 0;

```

```

    for (auto &g: v_persons){
        if(g.get_status() == "susceptible"){
            susceptible_number++;
        }
    }
    return susceptible_number;
}

void update_pop(){//update whole pop till nobody is sick

    probability = 0.8;//probability of transfer
    perc = 0.5;//percentage of population to inoculate
    people_contacted = 6; // number of people contacted
    next_position = 0;

    cout << " The probability is: " << probability << endl;
    cout << " The percentage of population to inoculate is : " << perc
        << endl;
    cout << " The population size is: " << num_people << endl;

    if (perc <= 1 && perc >= 0){
        int rand_person_innoculate, num_to_innoculate;
        if (perc == 1){
            num_to_innoculate = num_people;
        }
        else if (perc == 0){
            num_to_innoculate = 0;
        }
        else{
            num_to_innoculate = perc * num_people;
        }
        for (int k = 0; k < num_to_innoculate; k++){
            rand_person_innoculate = (int)(rand()%num_people) + 1;
            people_to_innoculate.push_back(rand_person_innoculate);
        }
        for (auto g: people_to_innoculate){
            if (v_persons[g].get_status() == "susceptible"){//assuming
                sick people can only be recovered not inoculated
                v_persons[g].set_status("inoculated");
            }
        }
    }
    else{
        cout << "Please enter a valid percentage [0,1] " << endl;
    }

    random_infection();
    while (count_infected() != 0){

```

```

cout << " -----
" << endl;
cout << " Day: " << days << endl;
cout << " Infected: " << count_infected() << endl;
cout << " Recovered: " << count_recovered() << endl;
cout << " Susceptible: " << count_susceptible() << endl;
days++;
for (int j = 1; j < people_contacted; j++){//number of people
    contacted/day
    contact++;
    for (int index = 0; index < num_people; index++){//find
        position of infected person in vector
        if (v_persons[index].get_status() == "sick"){
            position_of_infected.push_back(index);
        }
    }
    float tolerance = (float) rand()/(float)RAND_MAX; //random
    tolerance [0,1]
    if (probability > tolerance){
        cout << " Contact is made. # " << contact << endl;
        for (auto f: position_of_infected){//checking bounds
            if (f == 0){
                next_position = 1;//move right
                if (v_persons[next_position].get_status() ==
                    "susceptible"){
                    v_persons[next_position].infect();
                }
            }
            else if (f == num_people - 1){
                next_position = f - 1;//move to the left
                if (v_persons[next_position].get_status() ==
                    "susceptible"){
                    v_persons[next_position].infect();
                }
            }
            else{//infect neighbors
                if (v_persons[f - 1].get_status() ==
                    "susceptible"){
                    v_persons[f - 1].infect();
                    cout << " Infecting neighbor" << endl;
                }
                if (v_persons[f + 1].get_status() ==
                    "susceptible"){
                    v_persons[f + 1].infect();
                    cout << " Infecting neighbor" << endl;
                }
            }
        }
    }
}
}

```

```

        }
    }
    for (auto &e: v_persons){//update everybody
        e.update();
    }
}
if (count_infected() == 0){
    cout << "Disease ran its course in the " << days << " day " <<
        endl;
}
}
};

```

5.3 final.m

```

clear all clc

% 4 variables I can manipulate - pop_size, percentage of inoculated,
% people_contacted, probability_trasfer
% This is the MATLAB script I used to get the graphs for the population
% exercise. I varied the parameters but this is the main code I used

recovered = [0,0,0,0,0]; %array of recovered members
susceptible = [49,49,49,49,49]; %array of susceptible members
infected = [1,1,1,1,1]; %array of infected members

inoc_num = 5; %number inoculated
inoculated = [];
days = [];
for I = 1:length(infected)
    inoculated(I) = inoc_num;
end
for j = 1:length(infected)
    days(j) = j;
end

figure
hold on
plot(days, infected, 'g--o', 'LineWidth', 1);
plot(days, inoculated, 'r--o', 'LineWidth', 1);
plot(days, recovered, 'b--o', 'LineWidth', 1);
plot(days, susceptible, 'k--o', 'LineWidth', 1);
title([' Size: ', num2str(100), ' Inoculated(%): ', num2str(0.75) ...
    , ' Contact/day: ', num2str(6), ' Probability(%): ' ...
    , num2str(0.05)]);
xlabel(' Days ');

```

```
ylabel('People');  
legend('Infected', 'Inoculated', 'Recovered', 'Susceptible');  
hold off
```

5.4 repeat.m

```
clear all clc  
  
%quick script that finds all unique elements in array or vector  
  
A = [84,87,78,16,94,36,87,93,50,...  
     22,63,28,91,60,64,27,41,27,73,...  
     37,12,69,68,30,83,31,63,24,68,...  
     36,30,3,23,59,70,68,94,57,12,43,...  
     30,74,22,20,85,38,99,25,16,71,...  
     14,27,92,81,57,74,63,71,97,82,6,...  
     26,85,28,37,6,47,30,14,58,25,96,83,46,15];  
B = unique(A);  
  
disp(['Size of B is ', num2str(length(B))])
```

References

- [1] Kermack WO, McKendrick AG (August 1, 1927) "*A Contribution to the Mathematical Theory of Epidemics*". Proceedings of the Royal Society A. 115 (772): 700–721. doi:10.1098/rspa.1927.0118.
- [2] Fine, P.; Eames, K.; Heymann, D. L. (1 April 2011) "*Herd immunity*": *A rough guide*". Clinical Infectious Diseases. 52 (7): 911–16. doi:10.1093/cid/cir007. PMID 21427399
- [3] Merrill, R. M. (2013) *Introduction to Epidemiology*. Jones & Bartlett Publishers. pp. 68–71. ISBN 978-1449645175.