Virus Transmission Simulation

INFO 6205

Spring 2021

Team 20

Team member:

Jialiang Xiao (NUID: 001569847)

Zhilong Zhu (NUID: 001525601)

Shunqing Li (NUID: 001539980)

1. Introduction

The rapidly spread of COVID-19 has caused a global pandemic that affects the life of almost every human on Earth and has a significant impact on our societies and our ways of life. This deadly virus has killed millions of people around the world and the number is still going up. However, this virus is not undefeatable, with methods like masks, quarantining and social distancing, the transmission of the virus can be productively controlled. Especially with the vaccine becoming available, the stop of COVID-19 becomes one step closer. While facing the same challenge from COVID-19, different countries reacted differently and resulted in completely different situations comparing infection rates and death rates. Despite how gruesome this pandemic is, COVID-19 is not the first or the last virus that can dramatically impacts human's health and life. It becomes important to study and understand how viruses are transmitted and how can the transmission be better controlled.

2. Aim of the Project

This project is to simulate the transmission of viruses in a controlled situation over a period of time. Various factors are used to study how different methods impact the transmission of different viruses. A well-functioned user interface will be built to facilitate data entering and data graphing. The output of the simulation will be used to analyze efficiency and accuracy and produce possible best course of action when dealing with viruses.

3. Complete Project Details

The program allows user to input characteristics of virus like name, infection rate, infection rate of super spreader, super spreader rate and safe distance which is the range of the transmission of the virus. The program also allows user to input people setting which includes the number of total test subjects, initial infected ratio, initial super spreader ratio, range of move, distance of single move. Policy setting allows user to input numbers like testing ratio, ratio of people wearing masks, ratio of vaccination, mask's effectiveness and vaccine's effectiveness.

Base on the input from user in people setting, a number of dots are first evenly put in a set boundary as people to simulate an environment for virus transmission. When the simulation begins, a random number of infected people is randomly assigned and a number of people are randomly selected to be wearing masks base on user's input of mask ratio. For every simulation day, people are randomly selected to be tested or vaccinated. The number of testing and vaccination are based on the input of user. All people are to move in random direction and random steps in the beginning of every simulation day. Every person's coordinates and close contacts are being tracked. Close contacts of infected people or super spreaders has a probability of getting infected which is based on whether their possible infector is normal infected or super spreader. The probability of getting infected can be lower base on the status of masks and vaccination. Close contacts that got infected have a chance to be super spreader base on the ratio of use input. Every simulation day, a number of people are randomly selected to do testing. People that are tested positive will be put in isolation and their close contacts will be put in quarantine. If the close contact is infected, they will be put in isolation as well.

Usage of R and K number

The R number which means how many people can one infected person infect on average of the COVID-19 is on average 3. The K number, the dispersion parameter, of COVID-19 is around 0.1 and SARS has a similar K number of 0.16 which means almost 80 percent of the newly infected people are infected by 10 percent of all infected people. Because of the nature of K number, we divide all infected people into two groups: normal infected people and super spreaders and they have different ratios and different infection rates. Base on R number and K number, we calculate the infection rate of normal infected people is 0.0105 and infection rate of super spreaders is 0.3768. The ratio of super spreaders in all infected people is 10 percent.

4. Implementation

People class

Coordinates: The current coordinates of the person
States: Every person has a list of states showing their current health state.
Contacted: Every person has a list of people who are within the safe distance of every simulation day.

```
public class People {
    private Coordinates coordinates;
    private List<StateEnum> states;
    private List<People> contacted;
```

Random coordinates generator

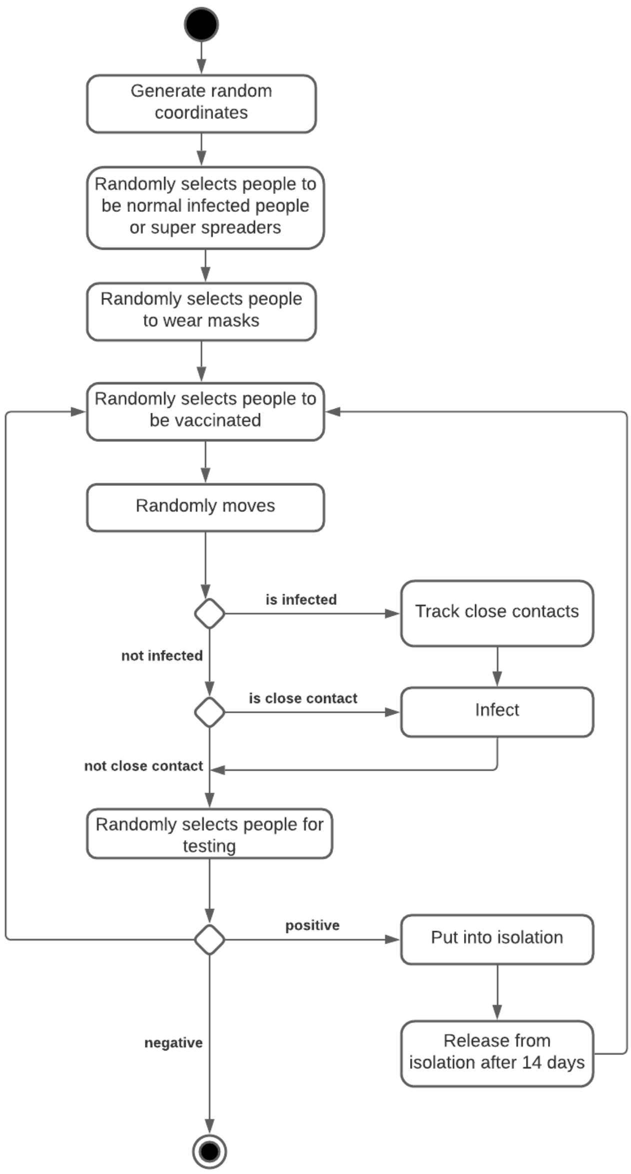
At the beginning of every simulation, random coordinates are generated and given to all people in the simulation pool.

```
public Coordinates generateRandomCoordinates(int bound){
    int x = r.nextInt(bound);
    int y = r.nextInt(bound);
    Coordinates coordinates = new Coordinates(x,y);
    return coordinates;
}
```

States

The list of states that people class has includes the following nine states.

```
public enum StateEnum {
    NORMAL( name: "Healthy", i: 0),
    CONTACTED( name: "ContactedWithPatients", i: 1),
    CONTACTEDS( name: "ContactedWithSuper", i: 2),
    CONFIRMED( name: "Infected", i: 3),
    SUPER( name: "SuperSpreader", i: 4),
    QUARANTINED( name: "Quarantined", i: 5),
    VACCINATED( name: "VaccineInjected", i: 6),
    MASK( name: "WearMask", i: 7),
    INFECTED( name: "InfectedByVirus", i: 8);
```



Initial process

The initial process uses the coordinate generator to assign every person a random coordinate to begin with. This process also uses random numbers to randomly select people to be infected or to become super spreader base on the input ratio.

```
Random r = new Random();
Day d = new Day();
CoordinateGenerator coordinateGenerator = new CoordinateGenerator();
NonIsolation nonIsolation = new NonIsolation();
try {
    for (int i = 0; i < numberOfPeopleInit; i++) {
        Coordinates coordinates = coordinateGenerator.generateRandomCoordinates(coordinateRange);
        double roll = r.nextDouble();
        if (roll > initRatioOfInfected) {
            People people = new People(coordinates);
            nonIsolation.addToNonIso(people);
        } else if (roll > initRatioOfSuper) {
            People people = new People(coordinates, StateEnum.INFECTED);
            nonIsolation.addToNonIso(people);
        } else {
            People people = new People(coordinates, StateEnum.SUPER);
            nonIsolation.addToNonIso(people);
        }
    }
} catch (Exception e) {
    System.out.println("init fail : NonIsolation Build Error");
}
```

Vaccinate

This method randomly selects people to be vaccinated and the selected people won't be vaccinated again if they are already vaccinated.

```
public void vaccinate() {
    this.countTodayVaccinated = 0;
    for (People p : this.nonIsolation.getPeopleNonIsoHash().keySet()) {
        double r = this.random.nextDouble();
        if (r <= this.vaccinateRatio) {
            if(!p.isVACCINATED()){
                p.addState(StateEnum.VACCINATED);
                this.countTodayVaccinated += 1;

            }
        }

    }
    this.countTotalVaccinated += this.countTodayVaccinated;
}
```

Masking

This method randomly selected people to be wearing masks base on the input ratio.

```java
public void masking() {
    for (People p : this.nonIsolation.getPeopleNonIsoHash().keySet()) {
        double r = this.random.nextDouble();
        if (r <= this.maskRatio) {
            p.addState(StateEnum.MASK);
        }

    }
}
```

Testing

The testing method and doTest method combined randomly select people to be tested base on the input ratio every simulation day. People that are tested positive whether they are normal infected people or super spreaders and their close contacts is put in isolation.

```java
public void testing() {
    this.tempRemove.clear();
    this.countTodayTest = 0;
    this.countTodayTestPatient = 0;

    for (Iterator<People> it = this.nonIsolation.getPeopleNonIsoHash().keySet().iterator(); it.hasNext(); ) {
        double r = this.random.nextDouble();
        People p = it.next();
        if (r <= this.testRatio) {
            this.countTodayTest += 1;

            if (doTest(p)) {
                it.remove();
                this.countTodayTestPatient += 1;
            }
        }
    }

    for (People p : this.tempRemove) {
        this.nonIsolation.getPeopleNonIsoHash().remove(p);
    }
private boolean doTest(People p) {
      this.nonIsolation.getPeopleNonIsoHash().remove(p);
   if (p.isINFECTED() || p.isSUPER()) {
       for (Iterator<People> it = p.getContacted().listIterator(); it.hasNext(); ) {
           People pC = it.next();
           putPeopleInIso(pC); //身边人都要隔离
           //todo 从NonIsolation消失 check
           if (!this.tempRemove.contains(pC)) {
               this.tempRemove.add(pC);
           }
       }
       return true;
         for (People pContacted : p.getContacted()) {
             putPeopleInIso(pContacted);
         }
   } else {
       // do nothing
       return false;
   }
```

Put people in isolation

This method put people that have been close contacted with the positive people and haven't been infected into isolation.

```java
private void putPeopleInIso(People p) {
    //this.nonIsolation.getPeopleNonIsoHash().remove(p);
    if (p.isINFECTED() || p.isSUPER()) {
        // kicked out the simulation without putting in Isolation

    } else if (p.isNORMAL()) {
        this.isolation.addToIsolation(p);
    }
}
```

Put people back in non-isolation

This method put people in isolation that are not infected back into non-isolation pool after 14 days.

```java
public void putPeopleBackInNonIso() {
    List<People> tempP = this.isolation.getPeopleOut();
    for (People p : tempP) {
        this.nonIsolation.addToNonIso(p);
    }
}
```

Virus spread

This method simulates the virus spread from all infected people including normal infected people and super spreaders. Close contacted people of the infected people are infected base on whether they are contacted with normal infected people or super spreaders.

```java
public void virusSpread() {
    this.todayContacted.clear();
    this.countTodayContacted = 0;
    this.countTodayNewPatients = 0;

    //todo 新建templist 装所有病人， check
    List<People> templist = new ArrayList<>();
    for (Iterator<People> it = this.nonIsolation.getPeopleNonIsoHash().keySet().iterator(); it.hasNext(); ) {
        People people = it.next();
        if (people.isINFECTED() || people.isSUPER()) {
            templist.add(people);
        }
    }
    for (People p : templist) { //从templist搞
        if (p.isINFECTED()) {

            p.getContacted().clear();//保证contacts 是当天的
            infectedSpread(p);
        } else if (p.isSUPER()) {

            p.getContacted().clear();//保证contacts 是当天的
            superSpread(p);
        }
    }
    this.countTodayContacted = this.todayContacted.size();
```

Infected spread

This method gets the list of contacted people which are people within the safe distance of normal infected people and let the contacted people be infected.

```
private void infectedSpread(People p) {
    for (People targetP : this.nonIsolation.getPeopleNonIsoHash().keySet()) {
        if (p.equals(targetP)) {

        } else {
            double distance = p.calcDistance(targetP);
            if (distance < this.safeDistance) {
                p.addContacts(targetP);
                if (!this.todayContacted.contains(targetP)) {
                    this.todayContacted.add(targetP);
                }
               this.countTodayContacted += 1;

                if (targetP.isNORMAL()) {
                    this.infect(targetP, this.sInfected);
                }
```

Super spread

This method gets the list of contacted people which are people within the safe distance of super spreaders and let the contacted people be infected.

```
private void superSpread(People p) {

    for (People targetP : this.nonIsolation.getPeopleNonIsoHash().keySet()) {
        if (p.equals(targetP)) {

        } else {
            double distance = p.calcDistance(targetP);
            if (distance < this.safeDistance) {
                p.addContacts(targetP);
                if (!this.todayContacted.contains(targetP)) {
                    this.todayContacted.add(targetP);
                }
               this.countTodayContacted += 1;

                if (targetP.isNORMAL()) {
                    this.infect(targetP, this.sSuper);
                }
```

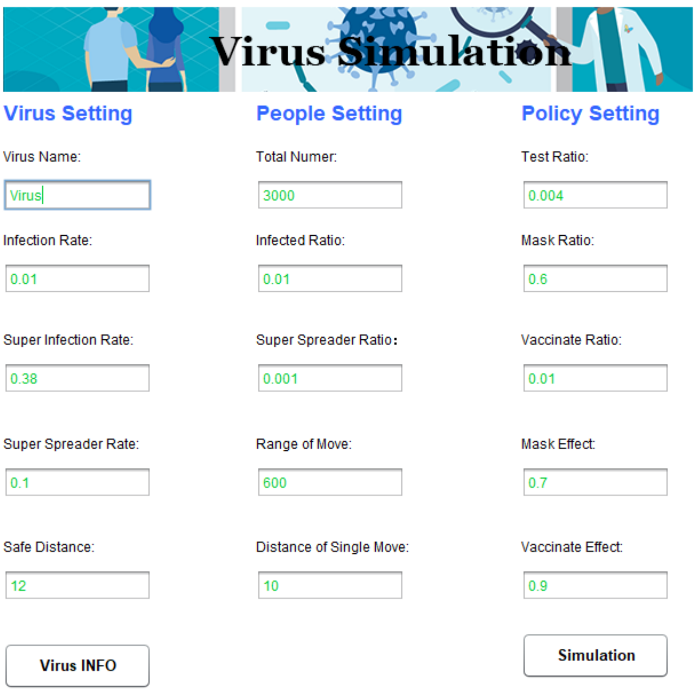
Infect

This method simulates chance of contacted people being infected by infected people by taking a random number and compare it to the infection rate. The infection rate is the result of base infection rate times the effectiveness of masks and vaccine. There is a chance that new infected people become normal infected people or super spreaders.

```
private boolean infect(People targetP, double sValue) {


    double maskE = 1.0d;
    if (targetP.isMASK()) maskE = this.maskEffect;

    double vaccineE = 1.0d;
    if (targetP.isVACCINATED()) vaccineE = this.vaccineEffect;

    double rInfectOrNormal = this.random.nextDouble();
    double rSuperOrNot = this.random.nextDouble();

    if (rInfectOrNormal < sValue * maskE * vaccineE) {
        this.countTodayNewPatients += 1;

        if (rSuperOrNot < this.superRatio) {
            targetP.becomeSuper();
            this.countTotalSuper += 1;
        } else {
            targetP.becomeInfected();
            this.countTotalInfected += 1;
        }
        return true;
    } else {
        return false;
    }
```

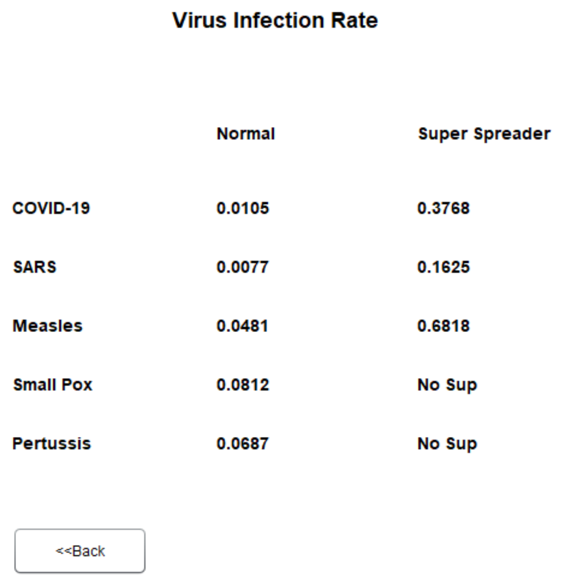
User interface

Default settings are provided and user can change all settings within validation range.



Virus INFO

A table of infection rates of various viruses is provided for user for reference.

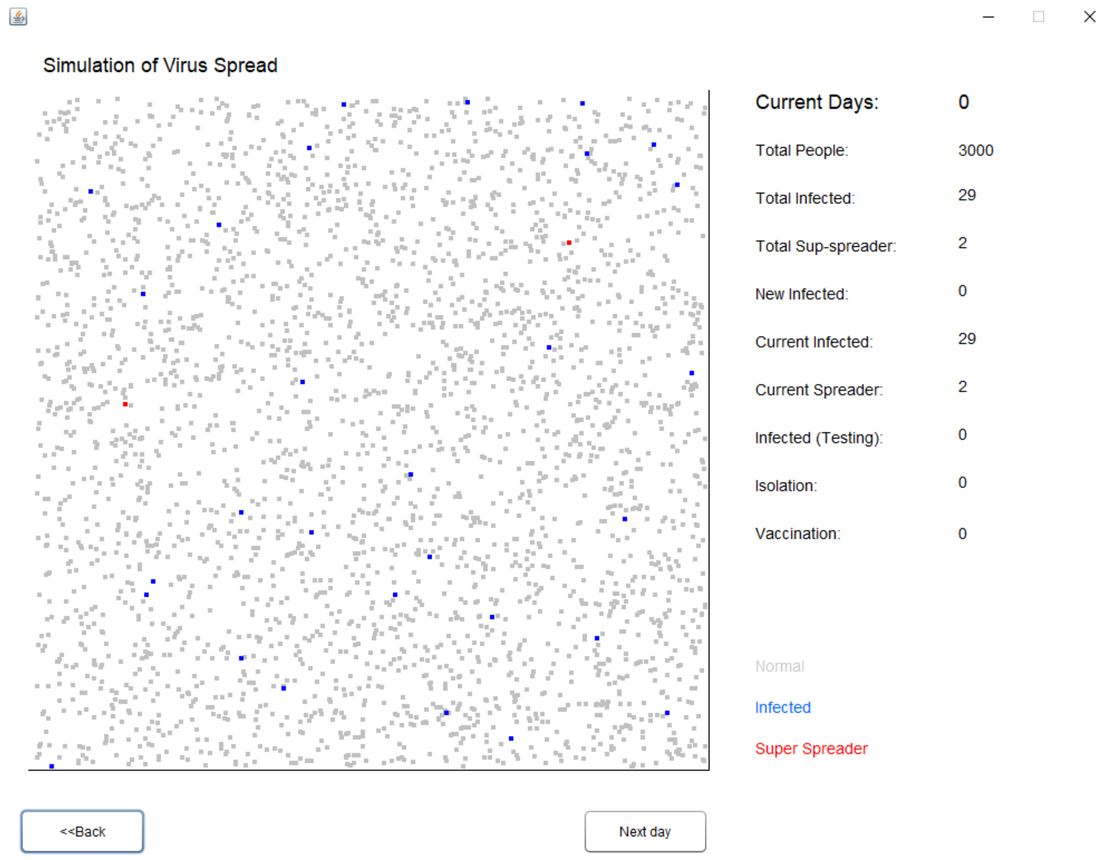


Virus Infection Rate

	Normal	Super Spreader
COVID-19	0.0105	0.3768
SARS	0.0077	0.1625
Measles	0.0481	0.6818
Small Pox	0.0812	No Sup
Pertussis	0.0687	No Sup

Simulation

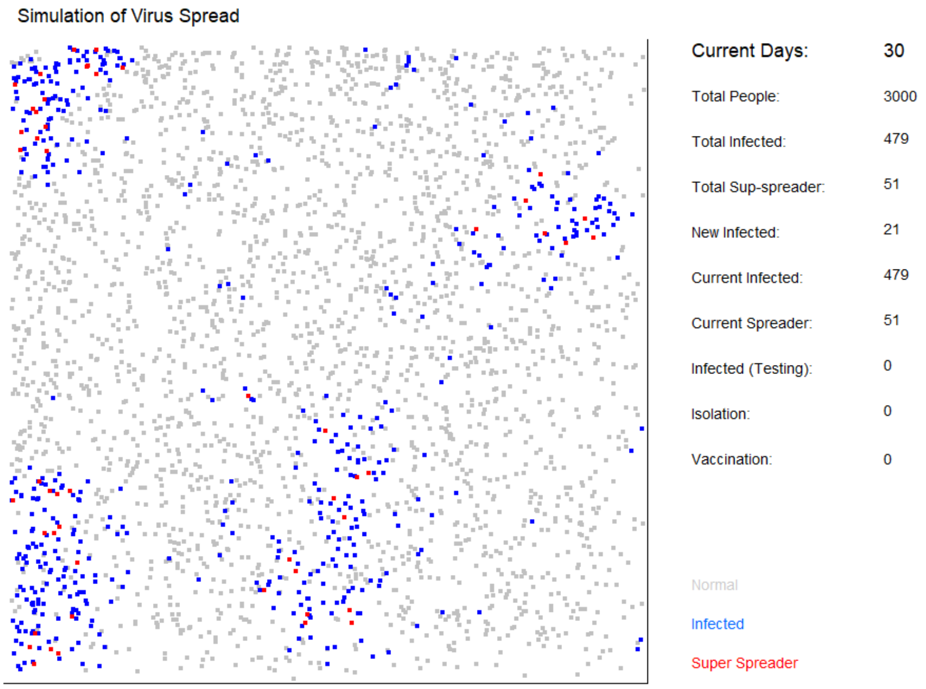
Simulation showing the current state of the virus spread.



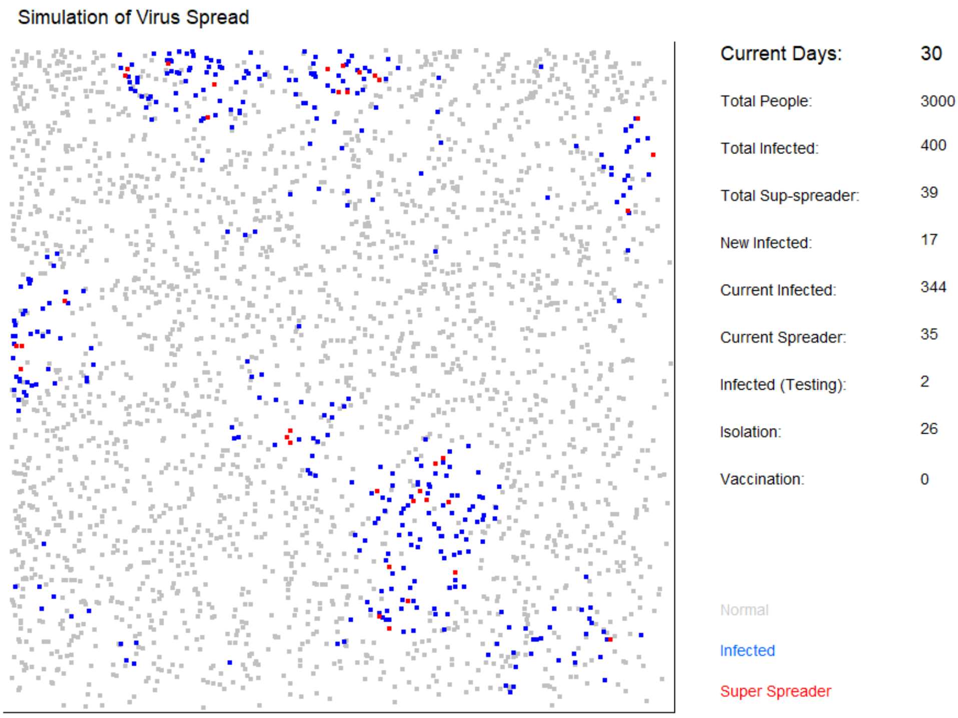
5. Output

Data input for infection rates, super spreader ratio, testing ratio, masks ratio and effectiveness, vaccination ratio and effectiveness are realistic data gathered from real-life situations.

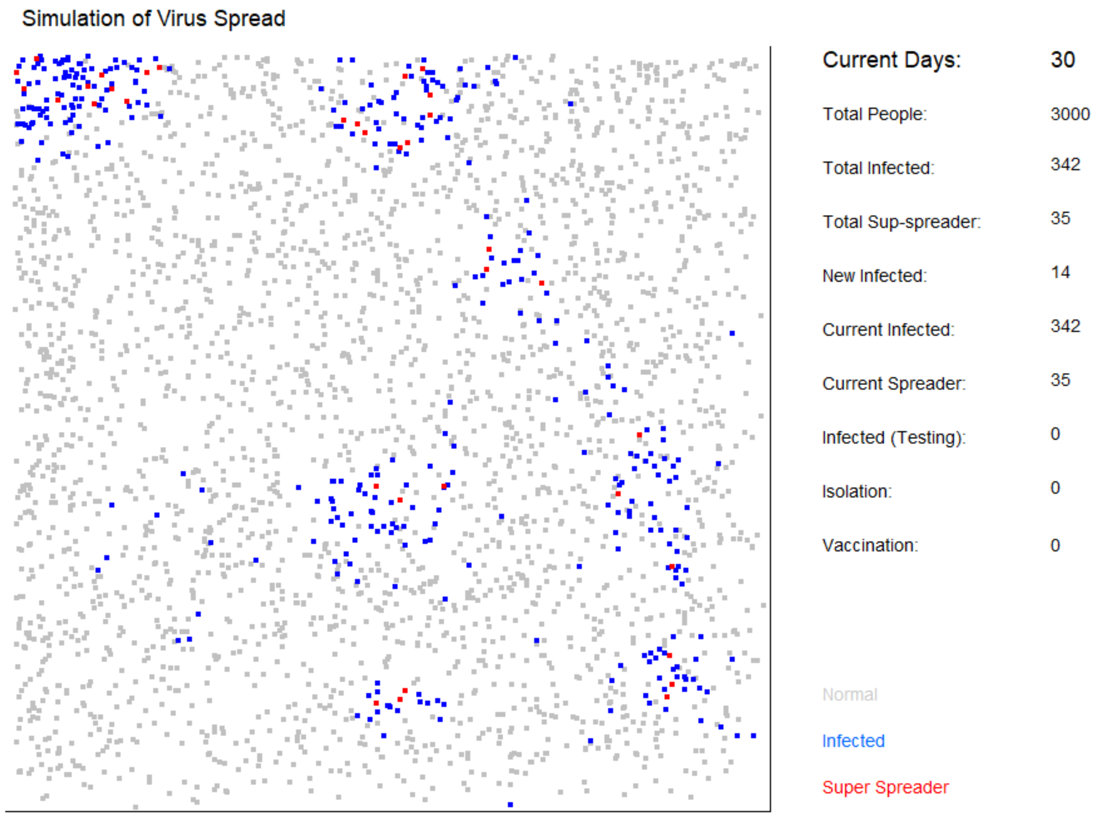
COVID-19 spread in 30 days with no control policy.



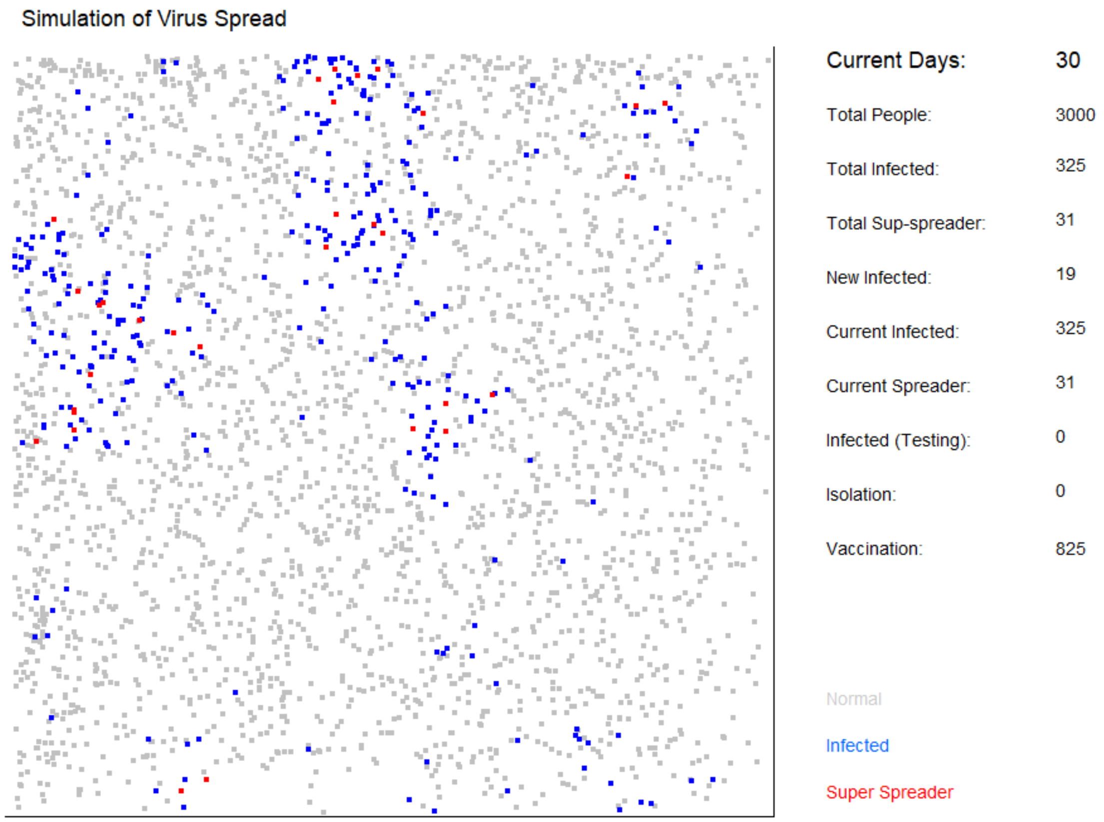
COVID-19 spread in 30 days with testing.



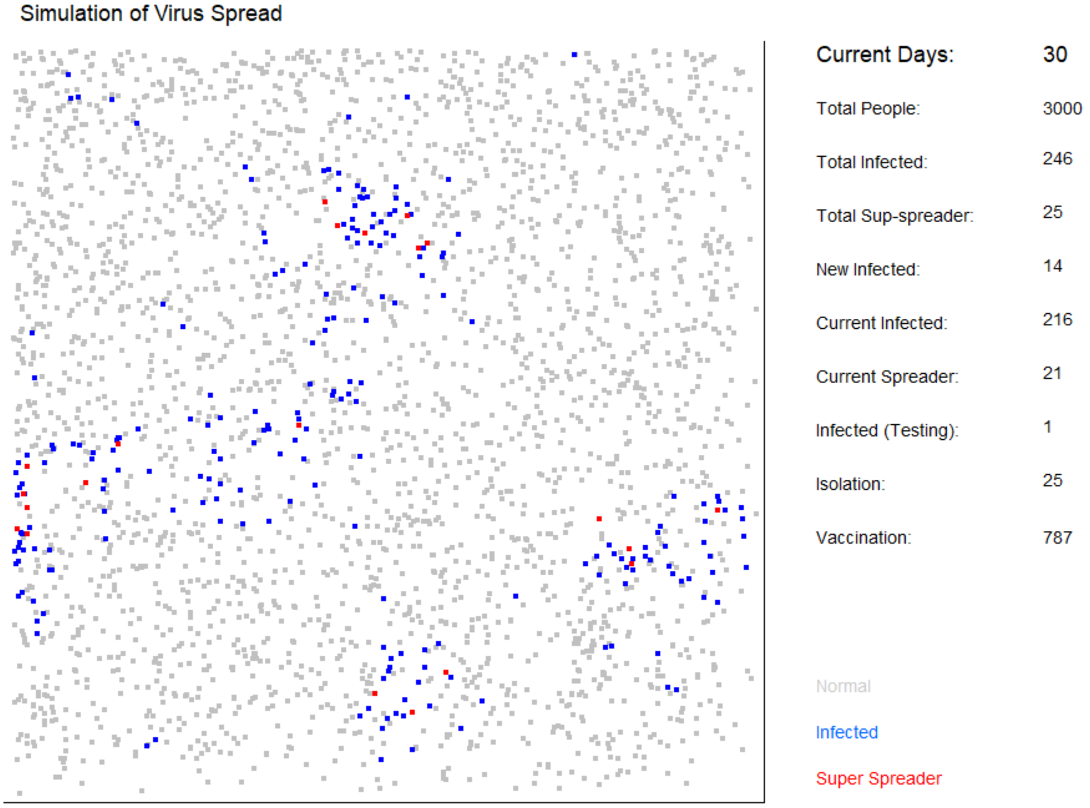
COVID-19 spread in 30 days with masks.



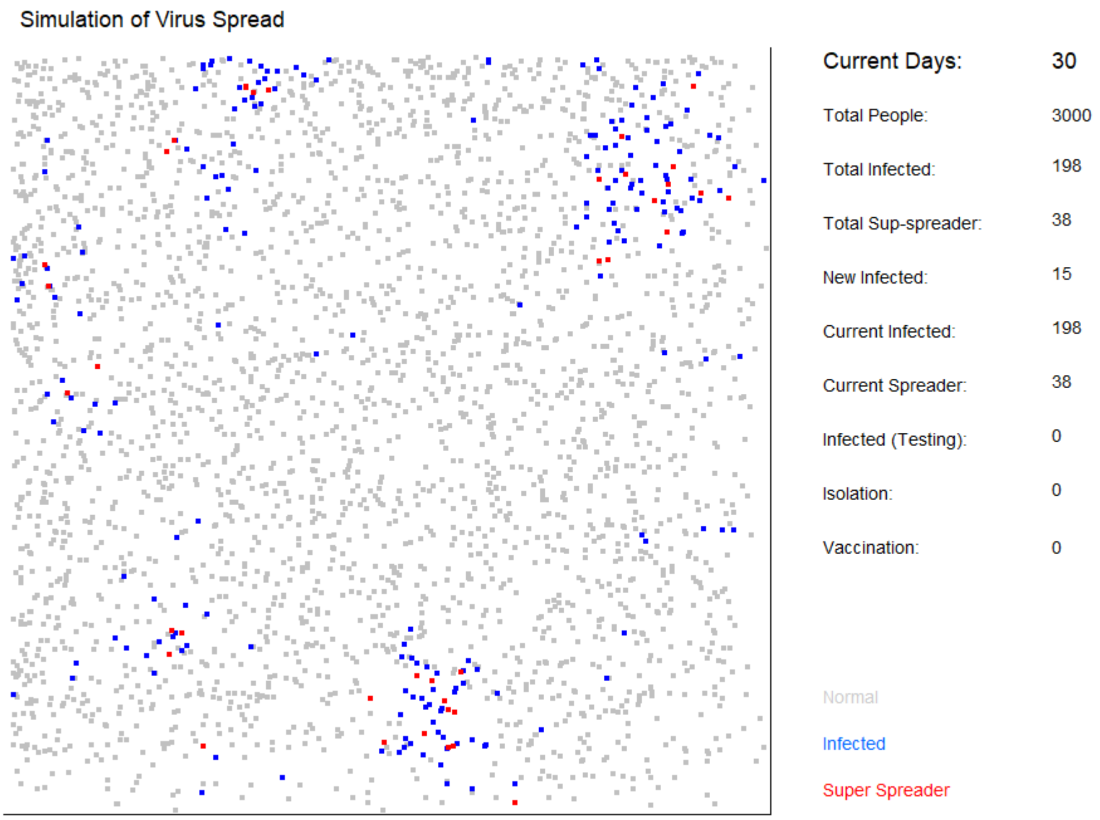
COVID-19 spread in 30 days with vaccine.



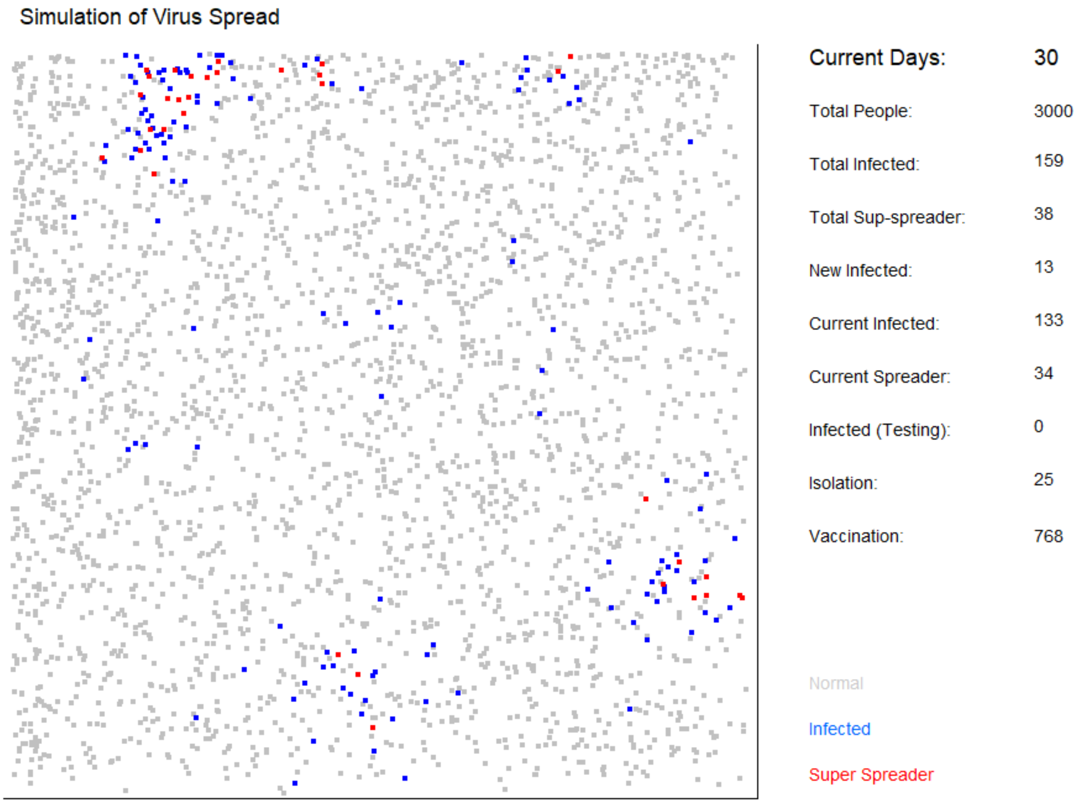
COVID-19 spread in 30 days with all control policies.



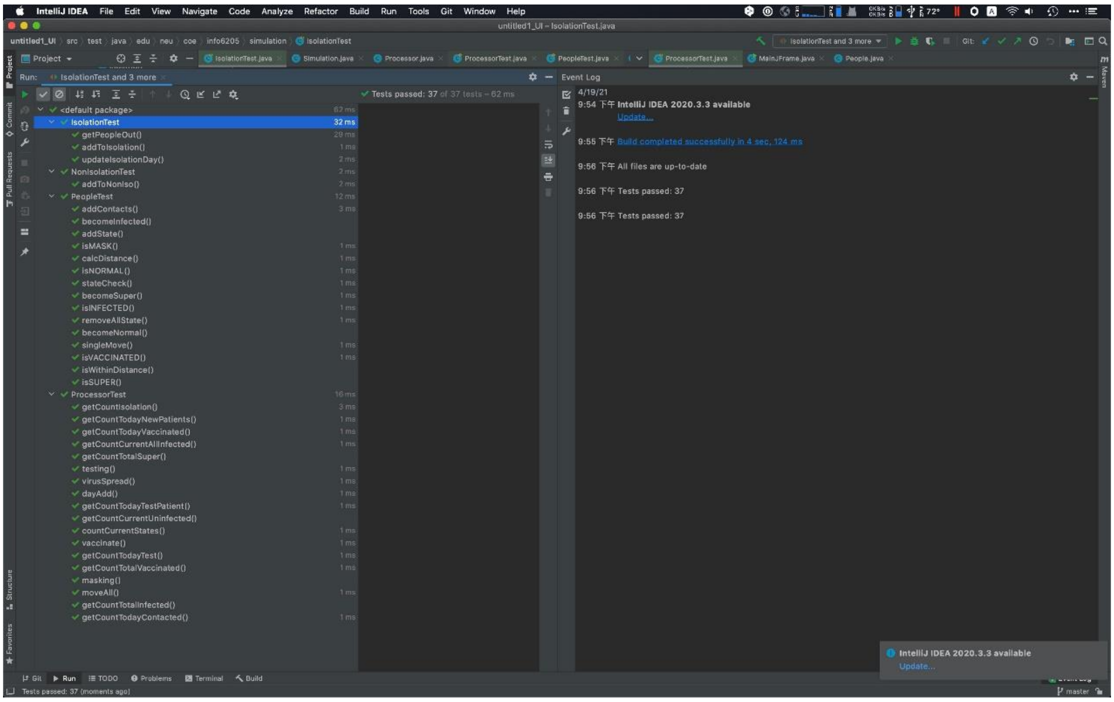
SARS spread in 30 days with no control policies.



SARS spread in 30 days with all control policies.



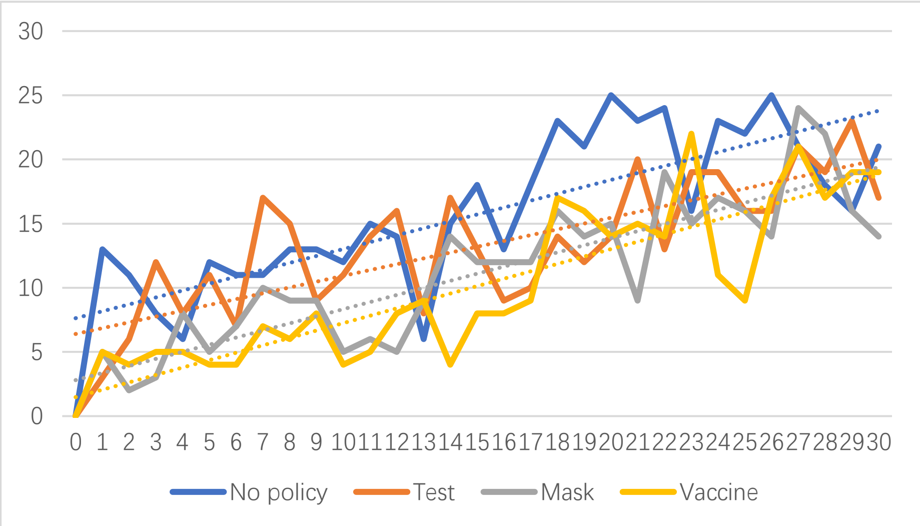
Unit test



6. Mathematical analysis

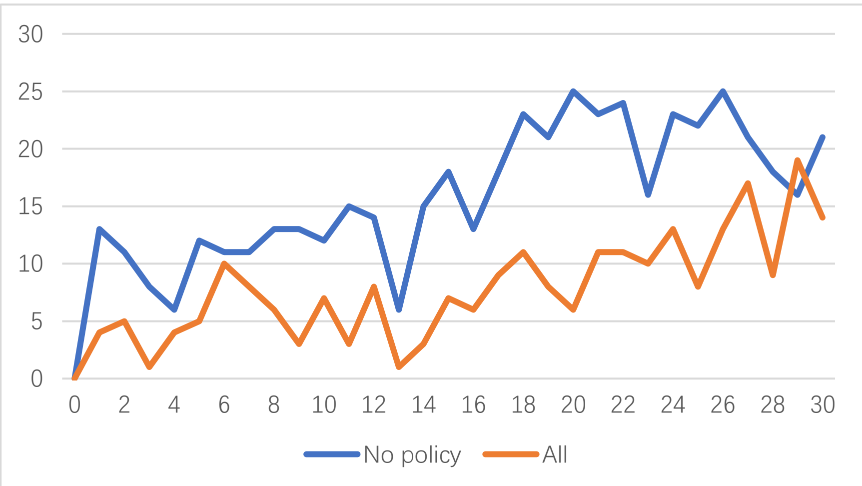
COVID-19 no control policy vs single control policy

Daily new infected people of COVID-19 within a 30-day period are compared between simulation with no control policy, simulation with just testing, simulation with just masks and simulation with just vaccine. As shown in the graph below, because of the randomness of movement and possibility of virus spread, the graph is not exactly linear. Results showing that simulation with no control policy having the obvious highest daily new infected. In the three simulations with single control policy, vaccination has the lowest average daily new infected, the second place is masks and testing have the highest average daily new infected. However, testing has the lowest daily new infected increase rate which means testing is more efficient in the long run.



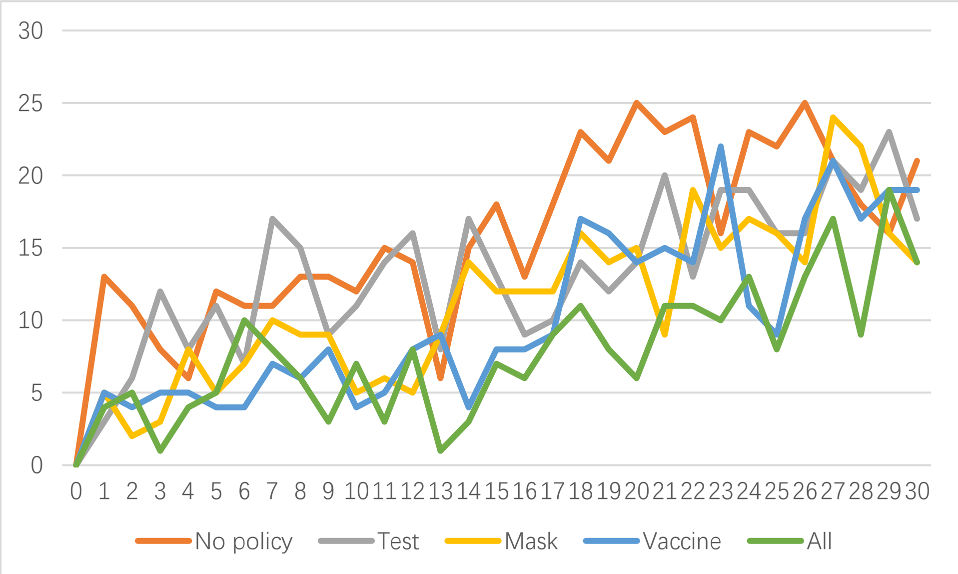
COVID-19 no control policy vs all control policies

As shown in the graph below, daily new infected is significantly decreased by using all control policies including testing, masks and vaccine. The growth rate of simulation with all control policies is also lower than simulation with no control policy. This shows a great difference between letting COVID-19 develop on its own and taking good counter measures against it.



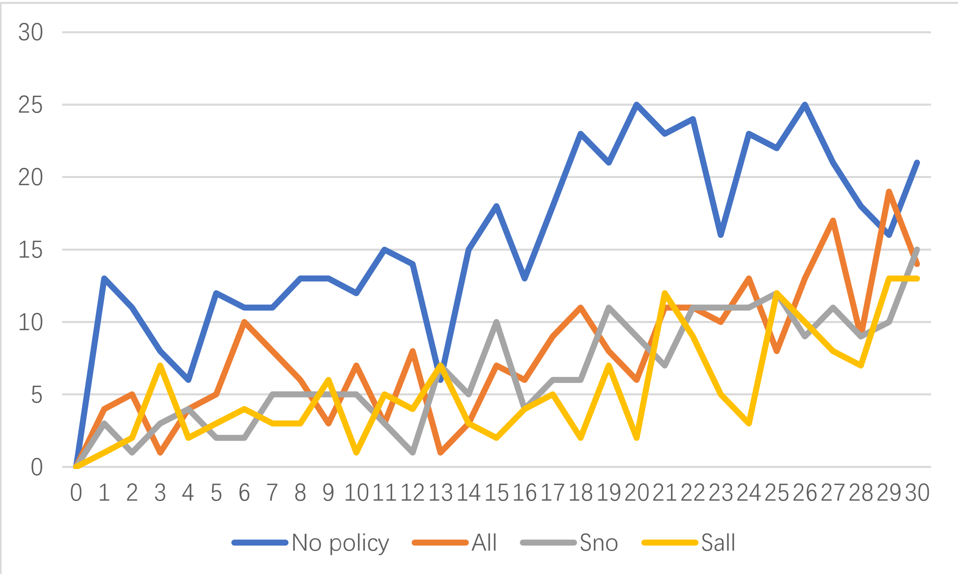
COVID-19 no control policy vs single policy vs all control policies

As shown in the graph below, simulation with all control policies has the lowest average daily new infected. The results suggest taking all control policies together is more efficient than taking any control policy alone.



COVID-19 vs SARS

SARS virus has lower infection rate than COVID-19 virus and they have the similar K value of 0.16 and 0.1. Comparing the simulation results of simulation with no control policy and simulation with all control policies of the both virus, SARS has a lower average daily new infected number than COVID-19, mostly because of its lower infection rate.



Worst-case Growth

We break down the whole simulation process into separate modules, namely vaccinating, virus spreading, and testing.

For module vaccinating, the worst growth is about c*N(where c is one constant and N is the number of people set at the beginning of the simulation), for there is only one loop in this module.

For module virus spreading, the worst growth is about c*N^2(where c is one constant and N is the number of people set at the beginning of the simulation). Because in module virus spreading, it takes one loop to find those patients, and for each patient, it takes one entire loop to find the normals and get them infected.

For module testing, the worst growth is about c*N^2(where c is one constant and N is the number of people set at the beginning of the simulation). Because it takes one loop to test and for each person tested as infected, it takes another loop to get every close contact into the isolation.

Data structure and Invariants

For the whole simulation, the invariants are that:

first, a person could only be normal, infected, or super spreader;

second, a person could only be in a non-isolation area, isolation area, or outside of the simulation.

Entropy

Entropy mainly comes from three aspects: the coordinate of a person, the state of a person, and the area the person belongs. Also, it is affected by the number of people put into the simulation.

7. Conclusion

From the program and algorithm perspective, we make our simulation as realistic as possible by implementing random movement and safe distance and using realistic data input coming from the current and past pandemics. We make the results more accurate by using R and K values for infection rates.

Base on the results, control policies like testing, wearing masks and vaccination are crucial to defeating the virus. Each control policy has its different effect. Base on the results of our simulations we concluded that effectiveness comparisons between the control policies are as follows: Vaccination > Masks > Testing and isolation and the best result comes from combining all three control policies together.

Comparing the simulation results of COVID-19 and SARS, we concluded that with similar K value, even the SARS has a slightly higher K value but with its lower R value its infection rate is much lower than COVID-19 so that SARS spreads slower in the same environment comparing to COVID-19. In the end, any virus, no matter how easy it spreads, can be controlled with proper control policies. As long as people strictly following the policy like wearing masks, keeping social distances and regularly taking tests and with the help of vaccine, the damage of viruses can be controlled to minimum.

Reference

Barrett, A. (2020, November 30). *The K factor: Nevermind R, here's the number we need to understand*. BBC Science Focus Magazine. https://www.sciencefocus.com/news/the-k-factor-nevermind-r-heres-the-number-we-need-to-understand/

Daily COVID-19 tests per thousand people. (2021). Our World in Data. https://ourworldindata.org/grapher/full-list-daily-covid-19-tests-per-thousand

Brooks, J. T. (2021, March 9). *Effectiveness of Mask Wearing to Control Community Spread of SARS-CoV-2*. Infectious Diseases | JAMA | JAMA Network. https://jamanetwork.com/journals/jama/fullarticle/2776536