Worms and Stochastic Epidemic Models

by

NQGB3

September 2021

A Dissertation submitted in part fulfilment of the

Degree of Master of Science: Statistics

Department of Statistical Science

Faculty of Mathematical & Physical Sciences

University College London

Word Count: 10583

CONTENTS

List of Figures 2
List of Tables 2
Abstract 3
Acknowledgement 4
1 Introduction 5
2 Background Knowledge 8
2.1 The SIR Model 8
2.2 Random Geometric Graphs 9
2.3 SIR on Random Geometric Graph Network without Boundary 10
3 Simulation Process 11
3.1 Movement of the Population 11
3.2 Transmission of Virus 13
4 Simulation Results 16
4.1 Analysis for Different Proportions of Infection with Movements 16
4.2 Analysis for Different Proportions of Infection without Movements 22
4.3 Comparisons between Simulations with and without Movements 23
5 Conclusions 25
5.1 Summary 25
5.2 Limitations 25
5.3 Further Work 26
References 29
A Table for an Overview of the Average Infection Proportion 30
B R Code for the Simulations 31

LIST OF FIGURES

2.1 Diagram of Markov Chain of SIR Model 8
2.2 An example of a random geometric graph 9
2.3 An example of two points near the border connected 10
2.4 An example of a random geometric graph without boundary 10
3.1 Diagram of Markov Chain of the state of points 11
3.2 An example of movements of points 13
3.3 An example showing the change of states of points during a simulation 14
4.1 Line plot of average infection proportions against r for varied R_0 19
4.2 Improved line plot of average infection proportions against r for varied R_0 20
4.3 Rough heatmap of average proportions with movements 21
4.4 Rough heatmap of average proportions without movements 22
4.5 Improved heatmap of average proportions without movements 22
4.6 Line plot of average infection proportions against r for varied R_0 24

LIST OF TABLES

4.1 Part of the records of numbers of infectives in general simulations 16
4.2 Estimated rough range of infection distance under varied ratios (90%) 17
4.3 Statistics for every possible infection distance under varied ratios (90%) 17
4.4 Rough range of infection distance under varied ratios (90%) 17
4.5 Estimated rough range of infection distance under varied ratios (50%) 18
4.6 Average proportions of infectives with varied r and R_0 (50%) 18
4.7 Rough range of infection distance under varied ratios (50%) 18
4.8 Estimated rough range of infection distance under varied ratios (5%) 19
4.9 Rough average proportions of infectives for varied r and R_0 (5%) 19
4.10 Improved average proportions of infectives with varied r and R_0 (5%) 20
4.11 Improved range of infection distance under varied ratios (5%) 21
4.12 Summary of estimated range of r under R_0 for three distinct infection proportions (90%, 50%, 5%) considering movements 21
4.13 Summary of estimated range of r under R_0 for three distinct infection proportions (90%, 50%, 5%) without movements 23

ABSTRACT

As the world is becoming increasingly mobile, people nowadays are constantly nagged by the tremendous prevalence of cyber virus such as worms. Our aim is to explore how the transmission radius of devices (r) and the infectiousness of worms (R_0) will influence on the eventual proportion of infectives (P_I), and how the movement of the devices affects the conditions under which the worms spread or die out. In this dissertation, we investigate and visualise the stochastic transmission of worms while some of the target devices keep moving in a closed unbounded network. To simulate the process of the opportunistic transmission, we build a composite model by combining a simple stochastic epidemic model (SIR) with a wireless ad hoc network based on random geometric graphs (RGG), then vary r and R_0 to obtain different P_I. We conclude that movements will encourage the prevalence of the epidemic in our tested region $r \leq 0.2, R_0 \leq 3$, and with larger r and R_0 (hence larger P_I) the influence of movements will be greater.

Keywords: Worms, SIR model, Random Geometric Graphs

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my supervisor, Dr. Chak Hei (Hugo) Lo for his invaluable advice, continuous support and great patience. I would also like to thank Dr. Simon Harden who provides us the precious topic. Finally, I would like to appreciate all the support I received from my family and friends.

1. INTRODUCTION

The digital virus world experienced a major upheaval in the late 1990s with the appearance of the computer *worms* family, which is one of the most aggressive and dominant type of computer viruses nowadays. The first representative of this family is the *Melissa* virus, which was discovered in 1999 [1].

Worms are self-replicating malicious computer programs which can propagate independently through computer networks without any human intervention as soon as they have breached the system. They are able to detect bugs in operating systems, probe IP addresses to attack and finally infect computers just like regular biological viruses [2–4]. They will control an infected device and use it as a host to scan and infect other devices. After these new devices are controlled, the worms will continue to replicate themselves using recursive methods, scan and infect other devices based on these devices, and the spreading behaviour will persist [5, 6]. As the worms copy themselves without any intervention or the host program and distribute themselves based on the law of exponential growth, they are able to infect and control a considerably large number of devices in a short time [6]. Not like a virus that causes damage mainly to local computer, a worm can harm a network consuming network bandwidth [7].

As the modern world is becoming increasingly mobile, traditional ways of connecting computer devices via physical cables have proven inadequate [8]. In recent years, more and more people are using mobile computing devices with some short-range wireless communication technologies like WLAN and Bluetooth [9] which are widely used to connect to the Internet and any other devices. Obviously, wireless networking has brought a lot of advantages. It allows users to roam freely without worrying getting disconnected to the Internet, and can be deployed rapidly as accepting a new user to enter the network is just a matter of authorisation as soon as all the infrastructure is built [10]. At the same time, however, the convenience for a device to join the wireless network will also put itself in danger from attacking by hackers with various mobile malware such as worms and Trojan horses. For example, Bluetooth allows some mobile worms to spread amongst vulnerable devices by exchanging files which have been infected within a certain communication radius. And as the infected devices travel with their owners, they may leave a trail of infected bystanders [11, 12].

Mathematical modelling of the transmission of infectious diseases such as flu, malaria was initiated by Bernoulli in 1760 [13–15]. The compartmental models are then developed by the works of Ross in 1916 [16], Kermack and McKendrick in 1927 [17], and Kendall in 1956 [18]. One of the basic compartmental models is SIR model (Susceptible, Infected, Removed) [19], which contains three non-linear ordinary differential equations. The model is deterministic and there is no probability but it cannot be solved to get an explicit formula solution. However, we can use simple tools of calculus to obtain a great deal of information from the implicit solutions. The SIR model is derived with a few strong assumptions. Many further epidemic models are extended in all kinds of aspects based on this basic SIR form by relaxing some of the assumptions. In this way the mathematical modelling will become far more sophisticated. However, in this study we will not try to solve an arithmetic solution for the underlying ordinary differential equations of SIR model but we will apply a stochastic framework to look into the dynamics of the transmission of mobile worms. It will be more realistic and easy to track the worms but also it can be far more complicated to simulate and analyse.

In real-life circumstances, the devices are not all connected and the worms will not appear on a device out of nowhere when it is not connected to any others so a simple SIR model is not sufficient to simulate the spread of the worms amongst the devices in a network. They are actually spatially distributed so it would be better to somehow put them into a graph to show their distribution and connectivity. One of the most intensively studied graph is Classical Erdös-Rényi random graph which is initiated by P. Erdös and A. Rényi in 1959 [20], consisting of a graph with N vertices (devices) and a random number of edges (connections) which are selected from all the edges of the complete graph given N vertices, where each edge is independently chosen with probability p. We see that in Erdös-Rényi random graph can show the distribution and connections in some ways but the status of edges of adjacent vertices are completely independent which is not the case when showing the spatial distribution of the computer devices in reality. For example, if device A is close to device B, device B is close to C, then there is a quite high chance that device A is near and connected with device C. Therefore, Erdös-Rényi random graph is inappropriate in this case and to accurately indicate the edges, we can instead use a more realistic graph, the

random geometric graph (RGG).

The study of random plane networks, which are now commonly referred to as *infinite* random geometric graphs, is initiated by Gilbert in 1961 [21]. In the finite-space case where a Poisson point process (completely random process) is performed, the topic is famous as *continuum percolation*, which is the subject of a monograph by Meester and Roy in 1996 [22] and is widely studied and extended in the recent years [23, 24]. For the *finite* random geometric graphs, some early work was done by Hafner in 1972 [25], and more researchers started to work on these graphs in the 1990s [26–28] and most recently [29, 30]. In 1988, Waxman proposed generalised random geometric graphs by introducing a probabilistic connection function [31] and the graphs are further studied in the 21st century [32, 33]. A random geometric graph [34] consists of a set of points that are randomly scattered over a region of space with some statistical distribution, and the edges connect any pair of points separated by a distance that is less than a fixed specified value. An edge in an RGG can well represent the connection between two nearby devices through which a virtual worm might be transmitted. To simplify our model, we only consider the region of a 2-dimensional space $[0, 1]^2$ in our study and ignore the possible difference on the elevations of devices in the real world. Also, we assume that all devices have identical wireless communication radius so that we can use such undirected graphs with the same connection distance.

The past decade has seen the widespread adoption of portable wireless devices and continuous epidemicity of computer viruses especially worms [35]. It is therefore a fairly important topic to learn about the worms and how they are transmitted in a wireless network. By understanding the underlying mechanisms that worms spread we are able to find corresponding techniques for their controls, and there are a cascade of questions for us to figure out. For example, given a single initial infective, under what conditions will the virus dies out at the early stage of an epidemic or the virus manages to spread and infect most of the population, and what factors will affect the transmissions mostly under what circumstances?

A lot of studies have worked on the stochastic spread of infectious epidemics [8, 36, 37] and some have investigated the transmissions on random networks [38, 39] or wireless as hoc networks [40]. However, as we mentioned above that the world is being increasingly mobile, most people who carry portable devices are likely to move intermittently rather than stay at the same places all the time, and their devices are under constant risk of getting infected by cyber viruses during their movements. Those previous studies did not take into account the movements of the population hence overlooking the potential influence of the movements on the worms spread. To learn about the effect of the movements, we can study the case where there are always a small amount of devices moving within a region, analyse the proportions of infectives for various kinds of viruses and compare them with the cases where all devices stay still. Besides, to keep the number of devices constant in our analysis, we can introduce a closed unbounded region where a device, which is going to move across and disappear at a boundary of the region, will appear at the opposite side, just like it is moving on the earth and the region we studied is the 2-dimensional extension of a sphere.

Therefore, in this study, we will mainly focus on constructing a composite model based on SIR model and RGG to simulate the distribution of real-life devices, the way they move and the spread of worms through the population, and hence exploring how the movements affect the conditions under which the virus will cause an extensive epidemic (90% infection proportion), moderate prevalence (50% infection proportion), or die away early (5% infection proportion). We will also analyse how the communication radius of a device (r) and the basic reproduction ratio of a worm R_0 will influence on the resulting proportion of infectives (P_I).

In Section 2, we first briefly review the necessary background knowledge including the SIR model and the random geometric graphs and then introduce an unbounded network based on SIR and RGG. Next, in Section 3, we propose a detailed procedure to simulate the process of virus transmission when devices are moving through the population. Also, we make some necessary assumptions about the proportion of moving points, the time took to perform a single movement and etc., and theoretically state how the virus jumps among devices with some graphs at the middle of an epidemic to visualise the process. In Section 4 we list the results after running the simulations. For the case where points move, the simulation is extremely time-consuming so we only use relatively large steps when adjusting the parameters. We investigate the communication radius required given 5 representative R_0 to reach 90%, 50% and 5% infection proportions respectively and attach a very rough heatmap. For the case where points do not move, we

traverse R_0 and r in sensible regions and generate slightly accurate heatmap. Then we compare the difference of corresponding estimated range of infection distance between these two cases. Finally, we conclude that movements will encourage the prevalence of epidemic than the scenario without movements given identical R_0 and r. As the epidemic develops further or when the virus has a smaller R_0 or larger r, limiting the movements of the population will restrain the epidemic to a greater extent. When the devices have a relatively short communication range r (within the range from 0.08 to 0.20 we analysed here), movements will promote the spread of virus, that is, more devices will get infected.

2. BACKGROUND KNOWLEDGE

2.1. The SIR Model

Compartmental models are widely used in mathematical modelling of epidemics. The simplest model is called SIR model, which divide the population into one of three categories: *susceptible* (S), *infected* (I), *removed* (R). Susceptible individuals are devices which do not have the virus but are likely to catch it, after which the devices will be moved to the infected compartment. Infected individuals are devices which have the virus and are able to spread it to susceptible individuals. Removed individuals are treated as recovered and immune (updated with corresponding patches) thus playing no role in the transmission of the virus afterwards. In this paper, we consider a closed population N, which means that the population will keep constant throughout the infection and the devices will not break down. We denote the number of susceptible, infected and removed individuals at time t by $S(t), I(t)$ and $R(t)$ respectively, so $N = S + I + R$ [38, 41]. The Markov process of SIR model is as below:

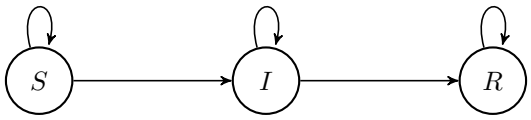


Figure 2.1: Diagram of Markov Chain of SIR Model

Assume that the infection rate is proportional to the product of S and I (the principle of homogeneous mixing), so in a deterministic model where $S(t), I(t), R(t)$ are treated as continuous variables, we have the following differential equations [38]:

$$\frac{dS}{dt} = -\frac{\beta SI}{N}$$
$$\frac{dI}{dt} = \left(\frac{\beta S}{N} - \delta\right) I$$
$$\frac{dR}{dt} = \delta I$$

where β represents the rate at which infectives make close contacts with others (and S/N represents the probability that such a contact is with a susceptible), and the individuals recover at a rate of δ implying an exponentially distributed infectious period with mean $1/\delta$. Given the facts that the total population is constant and all infectives come from susceptibles got infected and all removals come from infectives recovered, the third equation can be derived from the first two equations.

From the above differential equations we see that, at the very beginning of an epidemic, the number of infectives increases if and only if

$$\frac{dI}{dt} > 0 \implies \frac{\beta S}{N} - \delta > 0 \implies \frac{S}{N} > \frac{\delta}{\beta} = \frac{1}{R_0}$$

where the basic reproduction ratio $R_0 = \beta/\delta$ represents the expected number of infectives directly generated by a single infected individual during their infectious period with all other individuals being susceptible in the population. R_0 is an very important parameter in the SIR model which can tell whether an epidemic is in progress or not. If $R_0 > 1$ an epidemic will occur, and if $R_0 < 1$ the infection outbreak at the beginning will simply die away, and we consider a virus with large R_0 to be *contagious*. In most cases, the value of R_0 are known or can be estimated. For instance for Severe acute respiratory syndrome coronavirus 2, it was suggested to be 2.2-2.7 [42]. Using R_0 we can solve the above ODEs and the solution of I with respect to S can be written as [38]:

$$I(S) = -S + \frac{1}{R_0} ln(S) + 1.$$

2.2. Random Geometric Graphs

A random geometric graph (RGG) is one of the simplest spatial graphs. A well-defined mathematical model is as follows [34]. Let $||\cdot||$ be *the Euclidean* norm which are defined by:

$$||(x_1, ..., x_d)||_2 := \left(\sum_{i=1}^{d} |x_i|^2\right)^{1/2}.$$

Let f be some specified probability density function on $\mathbb{R}^d$, and let $X_1, X_2, ...$ be independently and identically distributed d-dimensional variables with common density f. Let $\chi_n = \{X_1, X_2, ..., X_n\}$, and let r be a positive parameter. A random geometric graph can be denoted by $G(\chi_n; r)$, which is an undirected graph with vertex set χ and undirected edges connecting all pairs $\{X_i, X_j\}$ with $||X_j - X_i|| \leq r$. In short, a random geometric graph is a region of space where a set of points randomly distributed with some probability distribution (such as Uniform distribution, Poisson distribution), and any pair of points within some designated distance are connected with an undirected edge. In this paper we mainly focus on the case where $d = 2$ (2-dimensional, disk graphs) and f is the density of the uniform distribution on $[0, 1]^2$. See Fig. 2.2 for an example graph with $n = 100, r = 0.15$.

Figure 2.2: An example of a random geometric graph

We notice that almost all individuals are connected given above parameters but there is also an isle at the lower left corner and an isolated point at the bottom of the graph, by which we can foresee the extreme situation that some individuals will survive at the end of an epidemic even if the virus is highly contagious as those susceptibles have no connection with any infectives throughout the process of infection.

In reality, different wireless devices may use different transmit power which means that the existence of a wireless link from node i to node j does not imply the coexistence of a link from node j to node i. Therefore the resulting communication graph is directed [40]. In this study, however, we assume that all devices have the same transmit power and the same corresponding transmission distance r, thus the resulting network can be presented as a 2-dimensional RGG as Figure 2.2 shows. RGGs have been sporadically used in real-life networks modelling [43] and are also widely applied in the study of continuum percolation and to the modelling of wireless ad hoc networks [24, 44–48]. There are also well-known random graphs like Erdös-Rényi random graphs $G(N, p)$ with N vertices. The way they establish edges among vertices is not using a designated distance r like RGGs, instead for each possible edge in a graph, the probability that the edge is present is p [20, 49]. Thus, the number of edges $K = pN(N-1)/2$ and the graph is characterised by the connectivity degree $\alpha = 2K/N = pN$ (we consider $N \approx N - 1$ when investigating large systems with large N), i.e. the average number of edges per vertex [50]. Erdös-Rényi random graphs are also being extensively used in many fields such as social network analysis [51].

2.3. SIR on Random Geometric Graph Network without Boundary

Given the above definitions of SIR model and random geometric graphs, we will combine these two concepts in this section. Suppose we have n individuals uniformly distributed in the region $[0,1]^2$ with effective infection distance r (in this case r can be interpreted as the communication radius of a device as this radius is also the effective transmission distance of the virus), and every device must be in one and only one of the three compartments: S, I and R. Besides staying at the current state, a device must follow the following state transition: $S \to I \to R$ and there is no turning back (See Fig. 2.1). In this study, we will treat the region of interest as the whole world instead of an endemic case, that is, the region is not bounded. In Section 2.2, we have seen the case where the graph is bounded and clearly there is no edge going through the border of $[0,1]^2$ in Fig. 2.2. When it comes to a graph without boundary, a pair of points lying near the opposite border may be linked.

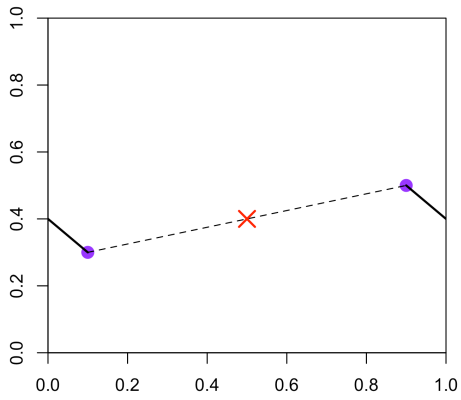


Figure 2.3: An example of two points near the border connected

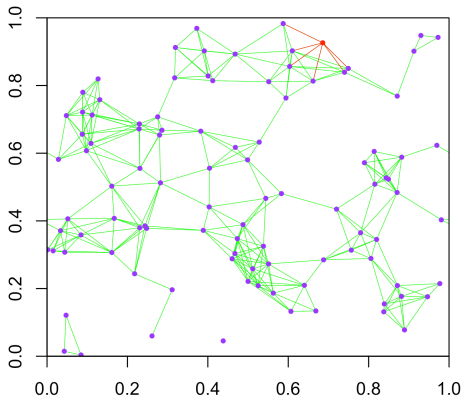


Figure 2.4: An example of a random geometric graph without boundary

For instance, in Fig. 2.3 with $r = 0.3$, two points are located at $(0.1, 0.3)$ and $(0.9, 0.5)$ separately and the Euclidean distance through the middle should be $d = \sqrt{(0.9-0.1)^2 + (0.5-0.3)^2} = 0.825 > 0.3$, thus it seems that no edge should exist between these points; however, considering the Euclidean distance across the border we get that $d = \sqrt{[0.9-(0.1+1)]^2 + (0.5-0.3)^2} = 0.283 < 0.3$, thus there is supposed to be a connecting edge going across one side and coming out from the opposite side as the figure shows. To find all possible edges across the border, we can generate 8 duplicates of the original random geometric graph in the region $[0,1]^2$ and place the duplicates around it (so the entire region of the 3×3 graph is $[-1,2]^2$). Then we calculate the Euclidean distance of each pair of points located in the square ring between $[1-r, 1-r]^2$ and $[1+r, 1+r]^2$, and add an edges connecting the pair of points within a specified distance r. See Fig. 2.4 for the above example graph in Fig. 2.2 without boundary. We see that the distribution of the points and all edges not crossing the borders are identical, while there are clearly some extra edges across the borders. For example, the point near $(0.0, 0.6)$ is connected to the point near $(1.0, 0.6)$ by an edge going through the left and right borders. There are also several edges connecting some points close to the left and right borders respectively but not for the upper and bottom borders as we can see there appears to be no such points close to each other regardless of the upper and bottom borders.

In such random geometric graph networks with SIR model, it is only necessary to calculate the edges for the points in state S and I, and only the edge connecting two points which are in state S and I respectively will lead to an event of infection. As those points in category *Removed* have recovered and will not contribute to the infection, we do not need to calculate the edges of them even if they are close to some other susceptibles and infectives (See Fig. 3.3 when recovered devices appear). By not considering removed devices, we will save a substantial amount of time.

3. SIMULATION PROCESS

Based on the theoretical part discussed in the previous section, we propose a detailed procedure for the simulation of the infection process considering the movements of points:

1. Generate an bounded random geometric graph based on the given number of vertices $n = 100$ and some random distance r on a unit square using function *sample_grg* in R package *igraph*. (We can set a seed at the beginning to reproduce the simulation collecting and checking data, if necessary)
2. Connect the points near the boundary (Remove the boundary. See Section 2.3).
3. Mark all the points as susceptibles (S) and then pick out one random individual marking as an infective (I).
4. Move part of the points randomly until an event of infection or recovery occurs (the infection and recovery rate are dependent on β and δ).
5. Recalculate the edges connecting S and I for the new graph after points having moved.
6. Repeat Step 4-5 until there is no more infective in the whole graph indicating the end of an epidemic.
7. Track the number of infectives and any other data of interest.

To analyse the process of infection for different combinations of r and R_0, we usually repeat the above simulation for multiple times. In this chapter we will talk about the details in the process of a single simulation, such as the way how the devices move and how the virus is transmitted in the population.

3.1. Movement of the Population

We can divide all the devices into two categories. One is still and has a small probability p_1 to start moving, and the other is moving and is very likely to keep moving with probability p_2. Hence, the state transition can be described as below:

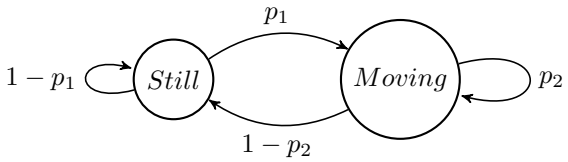


Figure 3.1: Diagram of Markov Chain of the state of points

Assuming that there are n devices in total and x of them are moving when the chain is stationary, we have

$$p_2 x + p_1(n - x) = x \implies x = \frac{p_1}{1 + p_1 - p_2} n$$

that is, $\frac{p_1}{1+p_1-p_2}$ of the population is moving, which is used as the initial number of moving devices in our model to assure approximately identical number of devices are moving at each time point. In our study where $n = 100, p_1 = 0.01, p_2 = 0.9$, we set the initial number to 9. Then we randomly choose 9 devices out of the whole group and mark the status as *Moving*, and the other 91 devices are marked *Still*. For every movement of a single device, the angle θ and the distance d follow the distributions below:

$$\theta \sim U(0, 2\pi), \quad d \sim N(0.3, 0.1)$$

which means a device will randomly move to anywhere within a circle of radius following a normal distribution with mean 0.3 and variance 0.1, then the change of its coordinates will be $(\Delta x, \Delta y) = (d\ cos\theta, d\ sin\theta)$. From Section 2.3 we know that we are using the geometric graphs

without boundary, thus any points moving outside a border will appear at the opposite border (this also ensures the size of the population is constant). For example, a device at $(0.1, 0.5)$ will move towards left for 0.3 unit, as there is no boundary the coordinates will be $(0.8, 0.5)$ instead of $(-0.2, 0.5)$. Let

$$a = \beta E_i, \quad b = \delta i$$

where a can be interpreted as overall infectious rate and E_i is the number of edges from points in I to the points in S; b can be interpreted as overall recovery rate and i is the number of devices in the state I at a time point. It is quite reasonable because the overall infectious rate will obviously increase as the number of edges connecting I and S increases, and an event of recovery will be more likely to occur with the number of infectives going up. In this study, we set δ as a fixed number 2, then we can adjust the value of R_0 by modifying β. For instance if we want $R_0 = 2$ then $\beta = \delta R_0 = 4$.

Assume that the points keep moving as the events occur continuously and independently, and the occurrence of events follow a Poisson distribution, that is, the time interval at the i_{th} event between two events, T_i, is a variable following an Exponential distribution with a rate $\lambda = a + b$. Also, we assume that the time taken for a single movement of points at the i_{th} event, T_i', follow a continuous Uniform distribution with mean 1 and fluctuating range 0.2, that is,

$$T_i \sim Exp(a + b), \quad T_i' \sim U(0.8, 1.2).$$

However, if we simply use this exponential distribution (of which the expectation is $1/(a + b)$ where normally a and b are positive and larger than 1 when there is at least one infective), the time interval would be fairly small (< 1) and as a result, in most cases, every single movement of points is followed by an event which is unreasonable. For example, at the initial stage of the infection where there is only one infective and the rest of the population is susceptible, we know that there must be several movements of points until the second infective appears as the infection cannot be so quick at the very beginning of the pandemic. Therefore, we need to multiply a fixed number C to the exponential distribution so the time interval will look more sensible. Also, we assume that when it approaches to the peak of the pandemic when there are the largest number of infectives, an event of infection or recovery will definitely happen, thus the fixed number cannot be too large as well to meet the assumption. Now we consider a general case with medium parameters, where $r = 0.16, \beta = 2(R_0 = 1)$. At first $i = 1, b = 2, a = 12$, so $T_i \sim Exp(14)$ and the expectation is $1/14$. Assume there are expected to be at least 5 movements before the second event occurs, then we have

$$\frac{1}{14} C \geq 5 \times 1.2 = 6 \implies C \geq 84.$$

At the peak of the pandemic in this case where there are 44 infectives, $i = 44, b = 88, a = 116$, so $T_i \sim Exp(204)$ and the expectation is $1/204$. As mentioned before, we assume that every event must follow a single movement, so we have

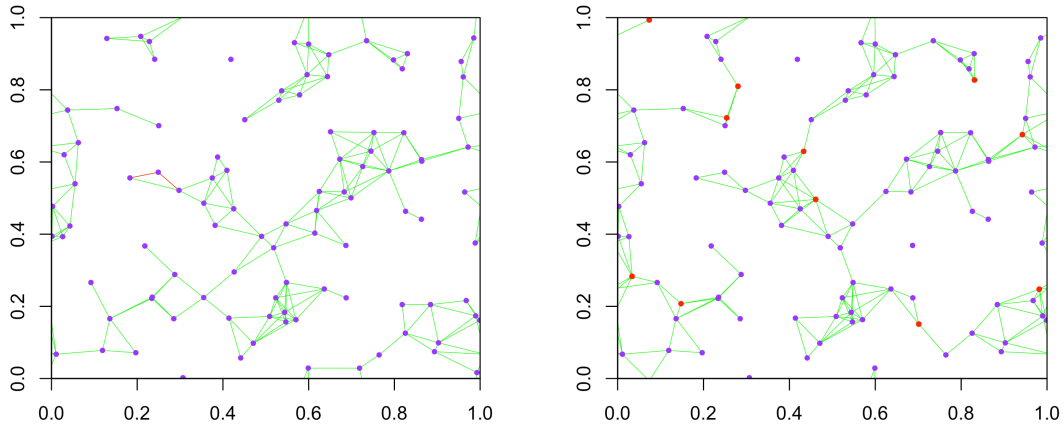
$$\frac{1}{204} C \leq 1 \times 0.8 = 0.8 \implies C \leq 163.2.$$

Above all, we have $84 \leq C \leq 163.2$ and we want to choose a number in this range to make the time interval sensible. When it comes to the case with large parameters saying $r = 0.2, R_0 = 3$, the value of a and b must be much larger and we do not want T_i to be too small. Besides, we can accept there needs to be more than 5 times of movement at the start so we can pick a fancy number as large as possible in the above range. Finally, we choose $C = 160$, hence

$$T_i \sim 160\ Exp(a + b), \quad T_i' \sim U(0.8, 1.2).$$

For convenience, we assume that the time interval is fixed during the movement of points until the next event of infection or recovery happens (between two events there will be many times of movements and every time the points move, the value of a and b change thus the distribution of the time interval is supposed to change simultaneously). For every movement of points happened, we subtract a uniformly distributed time in $(0.8, 1.2)$ from the time interval until it becomes zero or a negative number, and then we recalculate a, b and the new corresponding time interval.

In Fig. 3.2, we use two graphs to show the movements of the points, where $n = 100, r = 0.14$. Figure (a) shows the initial positions of the points when no event has happened. At first we randomly choose 9 points out of 100 to move, and the exponentially distributed time interval is set to be 4.99. After every single movement, the moving states of a point will change following the transition diagram in Fig. 3.1. When 6 times of movements have passed and the time interval has been reduced to -0.93 (first time to reach negative, just before the first event happens), there are 11 points in total which has moved at least once. Figure (b) shows the ending states of the points, where the points moved are marked as red dots and still points are coloured blue. By comparing (a) and (b) we clearly see that only the red points moved and the corresponding edges changed, any other points kept still and so did their edges.



(a) Distribution of points before movement (b) Distribution of points after movement

Figure 3.2: An example of movements of points

3.2. Transmission of Virus

Considering movements of the points and based on the SIR model on unbounded random geometric graph network, we are discussing how the virus will spread in the population in this section. We assume that the worm virus will propagate rapidly and take control of the device as soon as it invades a device, and at the same time it is ready to infect other devices.

In our study there is initially only one infective device and all the others are susceptible. At the initial stage when the movements of points have finished, an event of either infection or recovery will happen as mentioned in Section 3.1, of which the probabilities are $a/(a+b)$ and $b/(a+b)$ respectively, i.e. $Pr(I) = a/(a+b)$, $Pr(R) = b/(a+b)$, which are proportional to overall infectious rate a and overall recovery rate b respectively. If an infection event happens, we will randomly choose a susceptible connected to the initial infected device and change its state from S to I (the initial infective must be connecting to some other devices or else $E_i = 0, a = 0$, which means it is impossible for an infection event to happen). If a recovery event happens, the initial infective will recover and there will be no more infective, so it is the end of this simulation. Now suppose we are at somewhere in the middle of the epidemic where there exist several S, I and R simultaneously. If an infection event happens, we will record a possible infection list for all the susceptibles connected to every single infective. Note that the same susceptible may appear multiple times in the list if this susceptible is connected to many infectives at the same time so that it has a higher chance to get infected. Then we randomly sample one device from the list and move it from S to I. If a recovery event happens, we will randomly sample an infective and mark it as R. When it approaches to the end of the epidemic where there is no susceptible, we have $E_i = 0, a = 0, Pr(I) = 0, Pr(R) = 1$ so only events of recovery will occur then.

Now consider a case with $n = 100, r = 0.14, R_0 = 3$, we can visualise the transmission of virus by Fig. 3.3 below. The figure shows 6 symbolic stages (a) - (f) during the epidemic, and $times \in \mathbb{N}$ in the sub-captains indicate the number of events including both infection and recovery that have happened. For instance $times = 0$ means no event has happened which refers to the initial stage where there is only one initial infective. The largest possible value for $times$ is $2n - 1$ including $n - 1$ events of infection and n events of recovery. In the figure, blue points represent

susceptibles, red points represent infectives and grey points represent removals. Green lines represent *healthy links* which connect two susceptibles. Red lines represent *dangerous links* which connect susceptibles and infectives. Here we keep the edges connecting two infectives and mark them as red lines, though they did nothing throughout the infection process. If we deleted such edges the whole graph would look empty (also in reality, an infected device will continue to receive virus as long as it is not patched). For figure (e) there would be no connections and it would look quite similar to figure (f), but by adding such edges we can have an intuitive feeling about how many infectives still exist in the population.

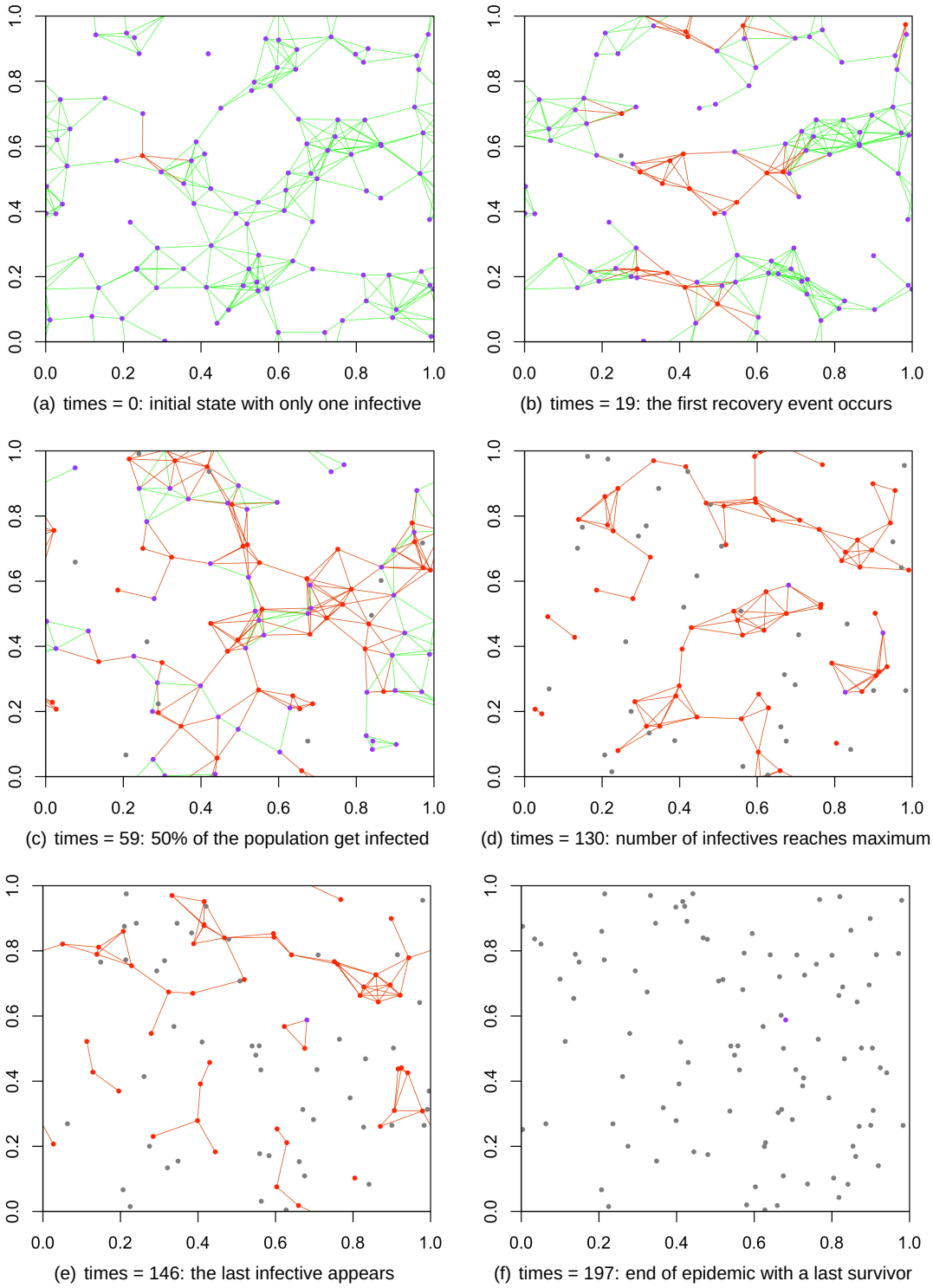


Figure 3.3: An example showing the change of states of points during a simulation

Figure (a) is at $times = 0$ showing the initial stage. The initial infective is located at near $(0.25, 0.55)$ with 5 dangerous links connecting to 5 nearby susceptibles. All other points are well connected as $r = 0.14$ is relatively large in region $[0, 1]^2$, except an isolated device near $(0.4, 0.9)$. We are considering movements in this case, thus the isolated point might still get infected by moving to somewhere else and establishing connections with some infectives. (The numbers of the susceptibles, infectives and removals are: 99S + 1I + 0R)

Figure (b) is at $times = 19$ where the first removal case occurs. The first removed point is located near $(0.25, 0.6)$. It is obvious that the layout of the points has been changed approving the movements of points, also there are still some points staying at the same locations such as those three points near $(0.0, 0.4)$. Besides, there have been some infectives in the middle of the figure and another one is at the upper-right corner. The infectives seem to be clustered and separated, and most of the points and links are healthy. It is at the early stage of the pandemic. (81S + 18I + 1R)

Figure (c) is at $times = 59$ where half of the population has been infected. Half of the population remains susceptible and a few removals are randomly scattered in the figure. Dangerous links are dominant and there are only a few green lines clustered in the right-hand side and upper side. Those dominant infectives might result from a fairly large $R_0 = 3$, and it tends to get every individual infected and removed at last. (50S + 40I + 10R)

Figure (d) is at $times = 130$ where the number of infectives reaches its peak 63. There have been quite a few removals in the figure but most of the population are still infectives. The connections reduced considerably compared to Figure (c) and we can easily find that there are only 3 susceptibles. Now all the links are red and most of the points are also marked red, so it seems that a batch of recovery events will occur afterwards. (3S + 63I + 34R)

Figure (e) is at $times = 146$ where the last infective appears. Now all the devices have been infected except one lucky dog. The only susceptible lies near $(0.7, 0.6)$, and is connecting to two distinct infectives. With clearly rising recovery cases, decreasing dangerous connections and a large quantity of infectives, the epidemic is approaching the end. (1S + 51I + 48R)

Figure (f) is at $times = 197$ where there is no more infective. The last survivor stays in the same position as in Figure (e) and any other points are in state R. There is clearly no edge and that is the end of the simulation. (1S + 0I + 99R)

In this case, the virus is highly contagious with moderate infection distance and considerably large reproduction ratio R_0, hence red spreads all over the graph at somewhere in the middle of the simulation and almost all devices are finally infected and removed. In the above figures, we can visually observe all the features mentioned before, such as connecting edges between points which are close to each other, points in distinct colours representing distinct states (S, I, R), edges crossing the borders in graphs without boundary, continuous movements of points as the infection processes and so on. By visualisation of the simulation process, we can easily check the status of the graph including distributions of the devices and connections, thus looking into the details of the simulation, for example, the reason why there is still some susceptibles at the end or why the epidemic dies away at the very beginning. Then we are going to repeat the above procedures using different parameters to do some analysis.

4. SIMULATION RESULTS

Above we have introduced a model for the transmission of malicious worms amongst wireless enabled devices in a closed region and described detailed simulation methods. To explore the effects of the basic reproduction ratio R_0 and the effective infection distance r on the prevalence of a virus (which can be described using the proportion of the population that ever got infected, i.e. % of the removals in the end), we can perform an appropriate amount of simulations and track the number of infectives. In this study, we will find the range of infection distance required to reach a specified proportion of infection, given different $R_0 \in \{0.5, 1.0, 1.5, 2.0, 3.0\}$ when the population size $n = 100$ (Considering movements is so time-consuming that we have to decrease the number of r and R_0 to be checked, so we choose only 5 R_0 here).

4.1. Analysis for Different Proportions of Infection with Movements

Now we are observing the transmission of virus with movements and under what conditions the infection proportion will reach 90%, 50% and 5% respectively. First in order to give a general idea about the approximate range for each cases, we ran 10 iterations for each combination of R_0 and r starting from 0.06 with rough steps 0.02 and computed the average proportions which are listed in Appendix A. The size of this overview is fairly small so there might be some unpleasant simulations in which the virus dies away and only infects less than 5% of the devices even when R_0 is very large but in fact they are supposed to get most of the population infected. Those simulations will greatly affect the accuracy of the average proportion thus we may take into consideration the post-average after removing those *improper simulations* to estimate the infection level for each case. By observing the average and post-average, if a step of 0.02 probably contain the proportion of our interest (such as 50%, 90%), we will perform a series of further simulations using step 0.05 to shrink the range. We will stop increasing the infection distance if the post-average is already very close to 100%. Note the post-averages mentioned are only for reference and we have more precise criteria to determine the appropriate range of infection distance.

Infection Proportion - 90%

First we are discussing how large the infection distance r should be to get most (90%) of the population infected for different R_0. Since we need to simulate many times when almost all devices get infected and it could be really time-consuming, we are only aiming to find a rough range with intervals being 0.005. If under a specific condition (with selected R_0 and $r = r_{90\%}$, i.e. $r_{R_0,90\%}$) we repeat a limited number of simulations, and for all the iterations excluding improper ones the infection proportions reach 90%, we say that the infection distance $r_{90\%}$ is required under R_0 so that an overall 90% infection proportion is approved. The iteration with over 90% infectives is marked as *Success*, otherwise it is marked as *Failure*. From the table in Appendix A, we list the exact numbers of infectives of 10 iterations for part of the cases as below:

	Iteration Number									
Ratio-Distance	1	2	3	4	5	6	7	8	9	10
0.5 - 0.18	2	100	1	75	98	94	99	91	1	98
0.5 - 0.2	100	100	100	100	1	99	99	1	99	98
1.0 - 0.155	2	100	89	91	97	1	98	89	98	5
1.0 - 0.16	99	88	97	100	99	99	100	100	97	98
1.5 - 0.14	97	92	93	86	92	78	99	97	51	98
1.5 - 0.145	99	96	93	1	94	96	96	1	98	98
2.0 - 0.14	99	97	99	96	96	95	1	84	92	96
2.0 - 0.16	100	100	99	100	100	100	1	100	100	100
3.0 - 0.13	98	97	99	1	97	90	99	56	91	95
3.0 - 0.135	99	99	1	99	92	93	96	96	98	98

Table 4.1: Part of the records of numbers of infectives in general simulations

For $R_0 = 0.5$, when $r = 0.18$ the 4_{th} iteration has only 75% infected population; when $r = 0.2$

all iterations are successful except two iterations have merely 1% which should be ignored. So we have $r_{0.5,90\%} \in (0.18, 0.2)$. For $R_0 = 1$, when $r = 0.155$ the 3_{th} and 8_{th} iterations have 89% infection population (< 90%); when $r = 0.16$ the 2_{nd} iteration (88%) failed. As the proportions when $r = 0.16$ are already quite large and the average is 97.7% which is fairly close to 100%, we stop increasing the infection distance at 0.16. So we have $r_{1.0,90\%} > 0.16$ and we can guess that $r_{1.0,90\%}$ lies in the next 0.02 interval since 0.16 has almost met our standard, i.e. $r_{1.0,90\%} \in (0.16, 0.18)$,. Similarly we can estimate other $r_{90\%}{}'s$ for $R_0 = 1.5, 2.0, 3.0$ which are listed below:

Ratio	0.5	1.0	1.5	2.0	3.0
Range	(0.18, 0.2)	(0.16, 0.18)	(0.14, 0.145)	(0.14, 0.16)	(0.13, 0.135)

Table 4.2: Estimated rough range of infection distance under varied ratios (90%)

Now we are testing the ranges listed above to find a interval of 0.005 for each ratio. Based on our criteria, there is no need to redo the simulations for the lower bounds as at least one unacceptable case has already appeared and the corresponding $r_{90\%}$ must be above the lower bound. Hence, we are actually looking for the upper bound of infection distance for each ratio. Here we repeat simulations more times for accuracy, say 30 times, for each possible range and then we have the following results:

Ratio-Distance	Improper No.	Success No.	Failure No.
0.5 - 0.185	9	19	2
0.5 - 0.19	10	20	0
1.0 - 0.165	5	25	0
1.0 - 0.17	7	23	0
1.5 - 0.145	6	12	12
1.5 - 0.15	6	19	5
1.5 - 0.155	7	21	2
1.5 - 0.16	5	25	0
2.0 - 0.145	4	25	1
2.0 - 0.15	4	26	0
3.0 - 0.135	5	22	3
3.0 - 0.14	3	27	0

Table 4.3: Statistics for every possible infection distance under varied ratios (90%)

For $R_0 = 0.5$, when $r = 0.185$ it failed twice; when $r = 0.19$ all proper iterations are successful, so $r_{0.5,90\%} \in (0.185, 0.19)$. For $R_0 = 1.0$, when $r = 0.165, 0.17$ there is no failure for both, so $r_{1.0,90\%} \in (0.16, 0.165)$. For $R_0 = 1.5$, when $r = 0.145$ we have 12 failed iterations whilst all proper iterations are successful in Table 4.1 which might be due to chance, so our estimated range for $R_0 = 1.5$ in Table 4.3 is proved inappropriate. We see that when $r = 0.15, 0.155$ there are still 5 and 2 failed iterations respectively, whilst all proper iterations are successful when $r = 0.16$, therefore, we get $r_{1.5,90\%} \in (0.155, 0.16)$. We can do the same tricks on $R_0 = 2.0, 3.0$ and then the rough ranges of r are shown below:

Ratio	0.5	1.0	1.5	2.0	3.0
Range	(0.185, 0.19)	(0.16, 0.165)	(0.155, 0.16)	(0.145, 0.15)	(0.135, 0.14)

Table 4.4: Rough range of infection distance under varied ratios (90%)

Using the above methods, we have obtained rough ranges of r. When the basic reproduction ratio $R_0 = 0.5$ we require a quite large infection distance over 0.185 to get most of the devices infected; however, when $R_0 = 3.0$, a small infection distance within (0.13, 0.135) would be enough for an epidemic. As R_0 goes up, the required r keeps decreasing. In other words, as a virus getting more contagious, it will need less infection radius to cause an epidemic. Moreover, in Table 4.3 we find it interesting that as R_0 rises, the number of improper cases seems to be decreasing, which means the more contagious a virus is, the less likely it will die out at the beginning.

Infection Proportion - 50%

Next we are discussing how large the infection distance r should be to get half (50%) of the population infected for different R_0. In this subsection we also consider a rough interval of 0.005 unit long for convenience as before, but here we will use average proportion of infectives taking into consideration the improper iterations to assess whether a virus with specified R_0 and r will infect half of the devices. When analysing the cases for 50% there will be quite a few proportions which are distributed at a relatively low level around 10% or even at 1%, it would be inappropriate to just ignore these and they should be treated as one of the contributing factors for the 50% case. By observing the averages around 50% in Appendix A, we find that they are quite monotonic so it is very straightforward to find approximate range of r for 50% infection proportion. We can use the starting point of the range which first reaches 50% as lower bound and use the ending point of the range which last leaves 50% as upper bound. For instance for $R_0 = 0.5$, when d increases from 0.15 to 0.155 the proportion first hits 50% and when d rises from 0.16 to 0.165 the trend of the proportion last goes through the line of 50%, so we have $r_{0.5,50\%} \in (0.15, 0.165)$. Here are the estimated ranges of $r_{50\%}{}'s$:

Ratio	0.5	1.0	1.5	2.0	3.0
Range	(0.15, 0.165)	(0.135, 0.14)	(0.125, 0.13)	(0.125, 0.13)	(0.12, 0.125)

Table 4.5: Estimated rough range of infection distance under varied ratios (50%)

Based on the estimated rough ranges, we can expend them until the proportions are strictly monotonic with at least 2 cases less than 50% and at least 2 cases more than 50%. Also, we can moderately increase the times of iterations to make our results more stable. Now we repeat the simulations 20 times for every sensible infection distance given some ratios. Here are the average proportions for the iterations centred around 50%:

	Infection Distance											
Ratio	0.115	0.12	0.125	0.13	0.135	0.14	0.145	0.15	0.155	0.16	0.165	0.17
0.5								18.95	28.15	46.5	64.2	58.05
1.0				24.45	43.35	54.4	57.4					
1.5		18.85	29.8	34.05	61	68.1	74.4					
2.0	11.4	37.05	53.9	60.45								
3.0	24.95	42	53.9									

Table 4.6: Average proportions of infectives with varied r and R_0 (50%)

For $R_0 = 0.5$, based on our estimated rough range in Table 4.5 we performed the simulations for $r \in \{0.15, 0.155, 0.16, 0.165\}$, after which we found that we got only one case with the proportion greater than 50%, thus we extended our testing range to 0.17. For $R_0 = 1.5$, the estimated range (0.125, 0.13) does not include 50% so we extended the range until 0.145 to get 2 cases greater than 50%. We notice that almost all proportions in Table 4.6 are monotonically increasing which perform excellently (except there appears to be a dip at $R_0 = 0.5, r = 0.17$ but it does not really matter as it is far greater than 50% and so does the former proportion at $r = 0.165$, thus 50% is very likely to lie between 0.16 and 0.165. A further check with more iterations might be suggested to confirm the exact range). This notable feature make it a lot easier for us to determine the range of $r_{50\%}$. Here we have the rough range of infection distance for 50% infection proportion:

Ratio	0.5	1.0	1.5	2.0	3.0
Range	(0.16, 0.165)	(0.135, 0.14)	(0.13, 0.135)	(0.12, 0.125)	(0.12, 0.125)

Table 4.7: Rough range of infection distance under varied ratios (50%)

By just a glance at the results, we can draw a same conclusion as above subsection that r goes down as R_0 rises given a fixed infection proportion. Besides, the infection distance r clearly declines for each R_0 compared to the above section with 90% infection proportion, which means to get less devices infected the infection distance has to be smaller based on the same R_0.

Infection Proportion - 5%

Finally we are going to explore the range of infection distance r when there is only a small proportion (5%) of the population being infected (the virus dies out at a very early stage). Since the proportion of interest is fairly small, the occasions where a virus dies young will occur very frequently, we should take them into account and hence we decide to continue using the overall average proportion of infectives for each combination. Also, when analysing such small proportions a big step (e.g. 0.005) on the change of infection distance may result in a big change of the calculated average which may not be accurate enough for the 5% case. To avoid this, we will consider a more precise interval of 0.002 unit long. Furthermore, we are expected to encounter lots of so-called improper iterations mentioned above, which can lead to unstable fluctuation. We can reduce it by increasing the number of iterations and we are allowed to do so as the simulations will terminate quickly with relatively short running time. The first step is to observe a rough range of the infection distance for each ratio. By investigating the table in Appendix A, we can generally conclude the rough range of $r_{5\%}{}'s$ as below:

Ratio	0.5	1.0	1.5	2.0	3.0
Range	(0.1, 0.12)	(0.08, 0.1)	(0.08, 0.1)	(0.08, 0.1)	(0.08, 0.1)

Table 4.8: Estimated rough range of infection distance under varied ratios (5%)

The second step is to run corresponding simulations with slight more iterations, say 30, and calculate their averages. Below are the table and the line graph for the averages:

	Infection Distance											
Ratio	0.08	0.082	0.084	0.086	0.088	0.09	0.092	0.094	0.096	0.098	0.1	0.102
0.5	/	/	/	/	/	/	/	/	/	/	2.03	3.37
1.0	2.4	2.73	2.3	5.07	3.57	3.73	4.1	4.07	3.6	5.5	6.13	6.23
1.5	2.57	3.5	3.5	3.43	3.33	4.1	4.9	5.07	5.27	5.83	5.23	/
2.0	3.2	3.37	4.2	4.53	5.27	4.93	6.93	5.67	6.43	5.74	6.73	/
3.0	2.5	3.03	5.27	6.8	6.43	6.13	9.23	7.83	7	6.73	8.83	/

(continue)

Ratio	0.104	0.106	0.108	0.11	0.112	0.114	0.116	0.118	0.12	0.122	0.124	0.126
0.5	2.3	3.43	4.3	4.4	5.27	3.67	4	5.1	6.53	6.17	3.73	6.67
1.0	5.93	8	7.23	5.77	/	/	/	/	/	/	/	/

Table 4.9: Rough average proportions of infectives for varied r and R_0 (5%)

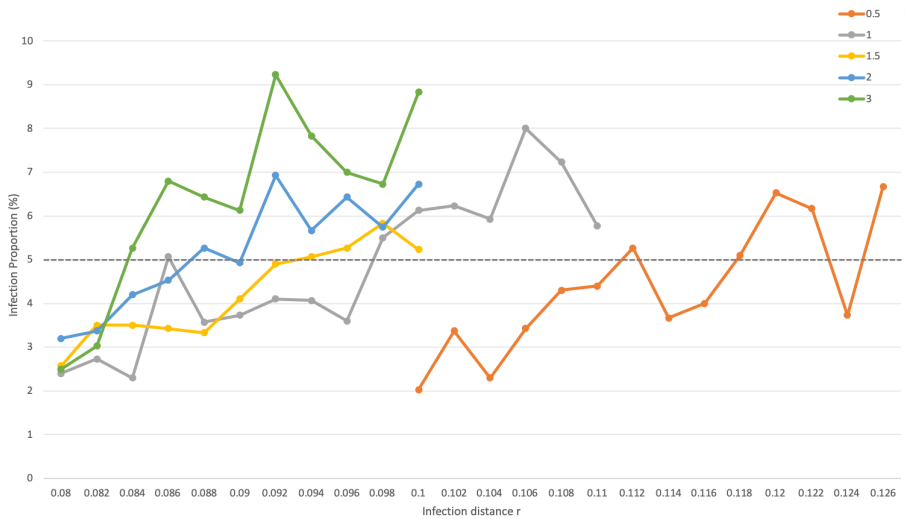


Figure 4.1: Line plot of average infection proportions against r for varied R_0

For $R_0 = 0.5$, the proportion does not exceed 5% until $r = 0.112$ with the first peak 5.27%, then it goes below 5% and re-appears above 5% at $r = 0.118$ (5.1%), followed by the second peak at $r = 0.12$ (6.53%). So far it seems that $r_{0.5,5\%}$ is smaller than 0.12. To ensure that we did a few more simulations following $r = 0.12$ and found the proportion is only 3.73% at $r = 0.124$ which is abnormal (for $r > 0.124$ the following several proportions are all way greater than 5% so $r_{0.5,5\%} < 0.126$, and we only show $r = 0.126$ (6.67%) here due to the size limit of the figure). The trend of the lines are absolutely not satisfying so we need to further increase the number of iterations for a smaller estimated range based on Table 4.9 and Figure 4.1. For each R_0, We can guess a median range of infection distance of 0.002 unit long which might include 5% and extend it to 0.01 unit long and perform further simulations with 100 iterations at step 0.002 (that is, six rs need to be checked for each $R0$). For $R_0 = 0.5$ we can guess 5% appears at somewhere near the middle peak (0.12), say, (0.118, 0.12). For $R_0 = 1.0$ the trend looks more monotonic (just ignore the peak at $r = 0.086$ which obviously should not be above 5% as its following 5 proportions are all below 5%) and 5% is very likely to appear between (0.096, 0.098). Similarly we can guess that the median ranges for $R_0 = 1.5, 2.0, 3.0$ are (0.092, 0.094), (0.086, 0.088) and (0.082, 0.084) respectively. The following table contains the average infection proportions of 100 iterations for the extended estimated median ranges and the corresponding line graph is also attached below:

	Infection Distance					
Ratio	0.114	0.116	0.118	0.12	0.122	0.124
0.5	3.91	5.32	5.27	5.65	6.02	5.77

	Infection Distance					
Ratio	0.092	0.094	0.096	0.098	0.1	0.102
1.0	3.82	4.23	3.76	4.84	4.76	5.77

	Infection Distance					
Ratio	0.088	0.09	0.092	0.094	0.096	0.098
1.5	4.9	4.16	5.04	5.09	5.98	5.3

	Infection Distance					
Ratio	0.082	0.084	0.086	0.088	0.09	0.092
2.0	3.38	4.09	4.44	5.52	5.14	6.13

	Infection Distance					
Ratio	0.078	0.08	0.082	0.084	0.086	0.088
3	2.95	3.17	3.38	4.9	5.83	5.92

Table 4.10: Improved average proportions of infectives with varied r and R_0 (5%)

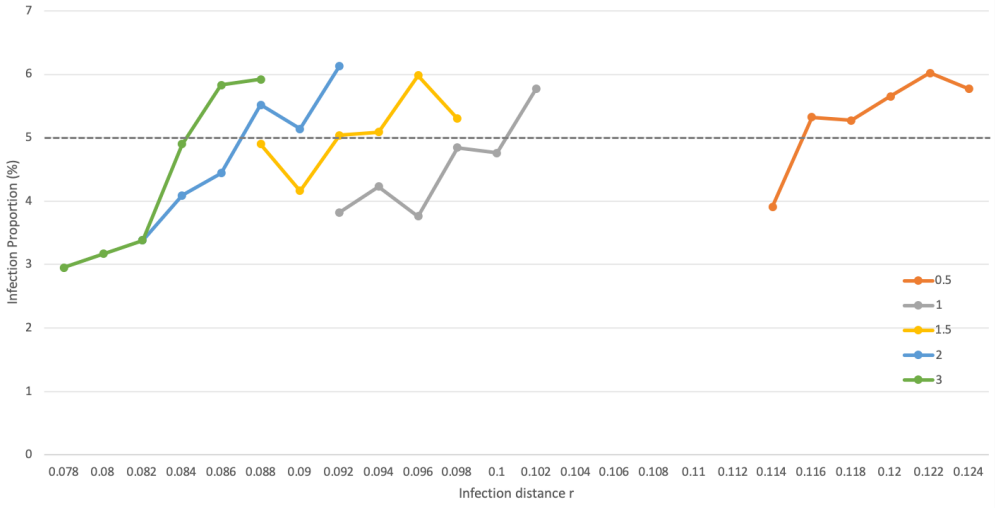


Figure 4.2: Improved line plot of average infection proportions against r for varied R_0

In Figure 4.2 the lines look much better and more monotonic now with more clear upward trends. For all R_0 there is only one piece of range that goes through the dashed line of 5% and it can be regarded as an improved range. It is unavoidable that there are still a few turning points thus the improved ranges might not be so accurate. We may need to continue rising the number of iterations but due to the limit of computational power, that is all we can do currently. The resulting improved ranges of infection distance are listed in Table 4.11:

Ratio	0.5	1.0	1.5	2.0	3.0
Range	(0.114, 0.116)	(0.1, 0.102)	(0.09, 0.092)	(0.086, 0.088)	(0.084, 0.086)

Table 4.11: Improved range of infection distance under varied ratios (5%)

Brief Summary

We have now performed quite a lot simulations using different r and R_0 for three proportions of interest (90%, 50%, 5%) considering movements of the population and estimated corresponding range of infection distance r which are summarised in Table 4.12 below:

	Ratio (R_0)				
Infection Proportion	0.5	1.0	1.5	2.0	3.0
90%	(0.185, 0.19)	(0.16, 0.165)	(0.155, 0.16)	(0.145, 0.15)	(0.135, 0.14)
50%	(0.16, 0.165)	(0.135, 0.14)	(0.13, 0.135)	(0.12, 0.125)	(0.12, 0.125)
5%	(0.114, 0.116)	(0.1, 0.102)	(0.09, 0.092)	(0.086, 0.088)	(0.084, 0.086)

Table 4.12: Summary of estimated range of r under R_0 for three distinct infection proportions (90%, 50%, 5%) considering movements

From the summary table, it turns out that for infection proportion $P_I = 90\%$ and $R_0 = 0.5$ the required r can be as large as 0.19, while for $P_I = 5\%$ and $R_0 = 3.0$ the required r is only at most 0.086 which is almost the half of the former one. We conclude that different infection proportions P_I and basic reproduction ratios R_0 will have a great effect on the required infection distance r. It is obvious that as P_I goes up and R_0 goes down, r will gradually increase.

In addition, we notice that for $R_0 = 0.5, 1.0, 1.5, 2.0, 3.0$, as P_I decreases from 90% to 5%, the middle point of the corresponding range of r decreases by 0.0725, 0.0615, 0.0665, 0.0605 and 0.525 respectively showing a slightly downward trend (The trend is somehow have a hump at $R_0 = 1.5$ and may confirm a monotonically decreasing trend by doing more iterations). It means that given a larger R_0, r will increase more slowly as P_I goes up, that is, for a more contagious virus, the required infection distance will increase at a slower speed to get more population infected.

Below is a very rough heatmap where we can visually observed the trend of average P_I of 10 iterations as r vary in (0.07, 0.2) with step 0.01 and R_0 vary in (0.2, 3.0) with step 0.2. The heatmap is really time-consuming, and to save time we did not run simulations for the rectangle region $R_0 \in [1.6, 3], r \in [0.17, 0.2]$. We know from Table 4.12 that P_I must be greater than 90% given $R_0 > 1.5, r > 0.16$ so we can assume $P_I = 91\%$ in this region and we are only interested in the 90% boundary. The rest of the graph has taken 106.92 concussive hours, and a larger P_I means longer simulation time. A dent $P_I = 75.3\%$ at $R_0 = 2.4, r = 0.16$ might be due to chance as we have only 10 iterations. The heatmap needs improving by rising the number of iterations.

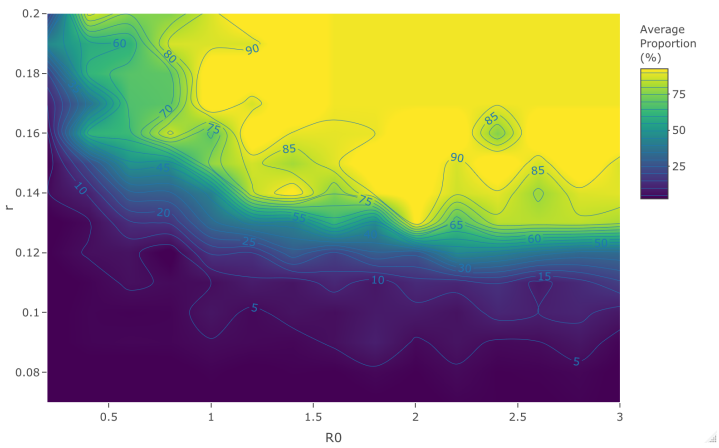


Figure 4.3: Rough heatmap of average proportions with movements

4.2. Analysis for Different Proportions of Infection without Movements

In the previous section we have considered the case where there would always be a group of devices moving in the population and the results have been shown in Table 4.12. Now we will see what if the devices do not move. The process of the simulation without movements is pretty much the same as the process with movements that we have mentioned in 3. Step 1-3 are identical, however in Step 4, now we will not move the points but instead keep the original distribution of points and edges as the events of infection or recovery happen, until all infectives appeared during the simulation are removed. Since we do not need to move points and recalculate the edges anymore which can save a considerably large amount of time, we can increase the number of iterations and run the simulations using different initial geometric graph.

We know that when considering movements a single initial RGG would be sufficient because most of the points move during the simulation and edges keep changing continuously so every device is likely to get infected in a single iteration. However, when ignoring movements we may need more than just one RGG because the edges will not change throughout the whole simulation in this case. It means that in a single iteration the infectives will only be generated from the susceptibles that live in the same isle with the initial infective, that is, the susceptibles that somehow connected to the initial infective via some points and edges in the RGG. Hence, the resulting infection proportions might be wired sometimes. For instance, suppose we have a small r (less connected RGG) and a large R_0 (highly contagious), and we are using two RGGs with identical r: one is somehow extremely discrete by chance where almost all the points stay alone and disconnected, but the other has way more tiny clusters, say, formed by 2-3 points. Obviously, the largest possible number of infectives will be the number of devices in a single isle, and with a large R_0 the points in the same isle as the infective will get infected. So the average proportions for the latter RGG is very likely to be larger than that for the former one, no matter how many iterations you choose to run - the average level of P_I has been decided as soon as an RGG is generated. Therefore, we should consider more RGGs when points are not moving.

To find out the effect of R_0 and r on the infection proportions, we can try to plot a heatmap for that as the simulations without movements are much quicker so we can create steps for R_0 as well (not just 5 levels as before) and make the steps for r even smaller. First we can have a general look at the approximate ranges of R_0 and r covering 5%-90%. We generate 5 RGGs and run 100 iterations for each RGG given $R_0 \in [0.5, 3.0]$ with step 0.1 and $r \in [0.06, 0.2]$ with step 0.01, and then calculate the average proportions for every combination of R_0 and r and the heatmap is shown in Figure 4.4 below, where x-axis is R_0 and y-axis is r.

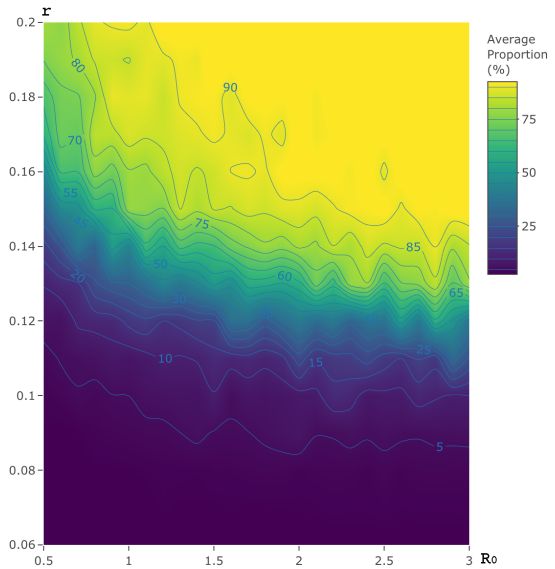


Figure 4.4: Rough heatmap of average proportions without movements

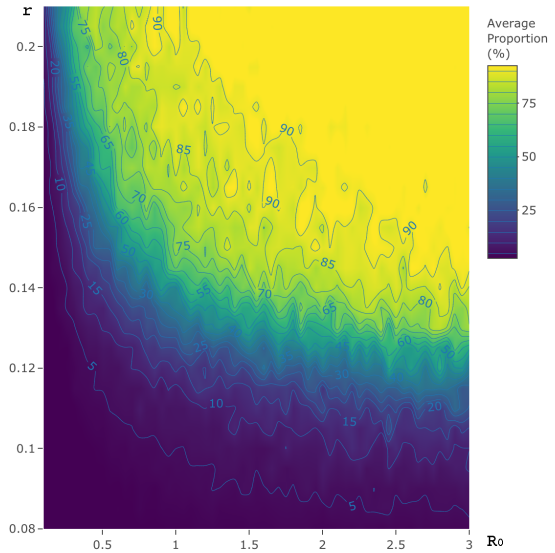


Figure 4.5: Improved heatmap of average proportions without movements

We notice that the 5% contour line lies above $r = 0.08$ so we can set the lower bound of r as 0.08. Also the 90% point of $R_0 = 1$ seems to be at around $r = 0.2$, so we can slightly increase the current upper bound 0.2 to 0.21 (the larger r is, the more computations required, so a slightly increase should do), that is, the new range of r is $[0.08, 0.21]$. Besides, we notice that the many contours disappear at y-axis (e.g. 5% contour line disappears at around $(0.5, 0.11)$) indicating that even for $R_0 < 0.5$, P_I can still reach 5%, even 50% or more, so we probably need to reduce the lower bound for R_0, say, 0.1. And 90% contour line stops at about $(3.0, 0.15)$, which means we do not need to rise the upper bound, that is, the new range of R_0 is $[0.1, 3.0]$. Using the new ranges of r and R_0 with new steps being 0.005 and 0.05 respectively, we have an improved version of heatmap with more precise ranges of R_0 and r as shown in Figure 4.5 above.

Although the general shape of the heatmap is quite similar, it appears that the contour lines of the former heatmap are way smoother than those of the improved one. Since we used smaller steps but did not increase the number of simulations, we are expected to have more unstable fluctuations between the formerly used big steps (we tried to use the newly simulated proportions with the original steps to generate a heatmap, and the contour lines are as smooth as before). To make the contour lines smoother, we should improve the accuracy of our simulated proportions by, for example, increasing the number of RGGs used or the number of iterations for every RGG.

By observing Figure 4.5, we can see a clear downward trend as r and R_0 decrease and an obvious demarcation line at around 70%. The whole heatmap is like a waterfall. First, the surface is very flat on the top layer in the upper right (above 90%). Then, the surface becomes rough with some spikes just like the water starts to splash and the droplets scatter into the air when approaching the edge of the cliff (about 90% - 80%). Next, the data decreases rapidly, which is similar to the steep fall over a rocky ledge (about 80% - 60%). Eventually, the surface is back to stable as a plunge pool (below 60%).

Furthermore, we notice that when $R_0 < 0.5$, it is quite hard for a virus to infect a large proportion of population, to reach 60% infection proportion r needs to be at least around 0.18, whilst when $R_0 = 1$, r needs to be only 0.14. When $R_0 > 1.5$, the contours lines below 80% are nearly parallel with the x-axis, which means when the infection distance of some viruses are relatively small such as below 0.14, then their infection proportions will be very close given $R_0 \in (1.5, 3.0)$. Indeed, if the infection distance is so small that not all devices are connected (small r), then no matter how easy the virus will be transmitted to other susceptibles (large R_0), it can only spread within the region which has connection to the infective, where the size of the region is decided by r. When r is large such as 0.2, if R_0 is close to 0 then it is clear that P_I will be at a very low level. However, if we slightly rise R_0 to about 0.2, P_I will rocket up from nearly 0% to almost 60%; if R_0 continues to rise to about 0.5 then P_I will be 80% which is fairly large. Therefore, we see that to infect a relatively large proportion of population, it requires both r and R_0 are at moderate levels.

4.3. Comparisons between Simulations with and without Movements

To compare the two scenarios with and without movements, we estimate the ranges of r for $R_0 = 0.5, 1.0, 1.5, 2.0, 3.0$ that covers 5%, 50% and 90%, then we run simulations for r with steps 0.002 using 5 distinct RGGs and 100 iterations for each EGG, and calculate the corresponding average P_I. We then use the last sub-range that contains the proportion of interest as an estimation of the range of r required under R_0 to reach 5%, 50% and 90% infection proportion (to ensure the chosen r will lead to the corresponding P_I). For example, Under $R_0 = 0.5$, P_I is fluctuating around 90% and the last P_I less than 90% is 87.686% when $r = 0.258$, and the following P_I is 91.226% when $r = 0.26$, so we have $r_{0.5,90\%} \in (0.258, 0.26)$. The results are listed in the following table:

	Ratio (R_0)				
Infection Proportion	0.5	1.0	1.5	2.0	3.0
90%	(0.258, 0.26)	(0.204, 0.206)	(0.172, 0.174)	(0.154, 0.156)	(0.138, 0.14)
50%	(0.164, 0.166)	(0.138, 0.14)	(0.13, 0.132)	(0.124, 0.126)	(0.118, 0.12)
5%	(0.108, 0.11)	(0.098, 0.1)	(0.09, 0.092)	(0.086, 0.088)	(0.084, 0.086)

Table 4.13: Summary of estimated range of r under R_0 for three distinct infection proportions (90%, 50%, 5%) without movements

And the line plot of average proportions P_I against r for 5 distinct R_0 is as below:

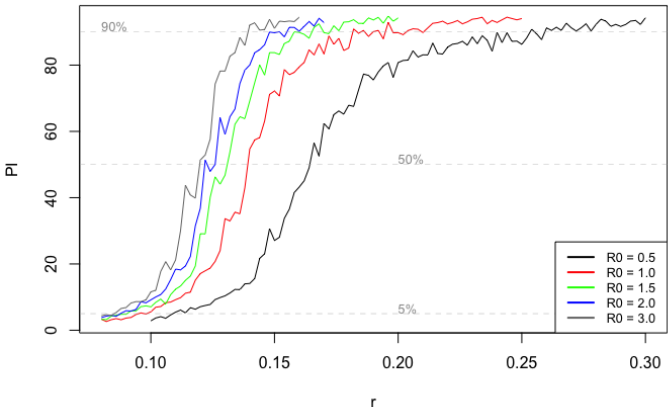


Figure 4.6: Line plot of average infection proportions against r for varied R_0

By investigating Table 4.13, we can draw the same conclusion as the cases considering movements that for a more contagious virus with higher R_0, the required infection distance r will increase less to get more population infected (increased infection distance when P_I increases from 5% to 90%, is from about 0.15 for $R_0 = 0.5$ to about 0.054 for $R_0 = 3.0$).

By Comparing Table 4.12 with Table 4.13 above, we notice that basically the required r will be larger under the same circumstance if not considering movements. When $R_0 = 0.5, P_I = 90\%$, the difference of r between the two scenarios can be about 0.07, whilst when $R_0 = 3.0, P_I = 5\%$ there is almost no difference. After observing the whole table, we see that the larger P_I is and the smaller R_0, the less difference of r will be. When $P_I = 5\%$, there is little different between the two scenarios for every R_0. From Figure 4.5 we have learnt that given $P_I = 5\%, R_0 \leq 0.5$ the infection distance r is always at a low level below 0.12 which means the geometric graph is not well-connected and it can be very hard for the virus to spread no matter whether the population moves or not. As R_0 increases, for the same P_I the difference between r is smaller and it means a virus with smaller R_0 will rely more on the movements of population to spread. For example, by observing the rows of $P_I = 90\%$, we notice that when $r = 0.14, R_0 = 3.0$, P_I are both about 90% for movements and no movements, and when $r = 0.19, R_0 = 0.5$, P_I is above 90% according to Table 4.12 considering movements but definitely less than 90% without movements as it requires $r > 0.258$. Moreover, it seems that devices with larger communication range r will be affected more by the movements. For example we can have a look at the column for $R_0 = 1.5$, when $r = 0.92$, P_I for movements or no movements are both about 5%, when $r = 0.16$, P_I is above 90% for movements but below 90% for no movements.

5. CONCLUSIONS

5.1. Summary

In this study we have built a stochastic epidemic model which is based on SIR epidemic model and random geometric graph without boundary, investigating and visualising the transmission of the virus during a simulation of epidemic where about 9% of the devices keep moving, exploring the underlying associations amongst the basic reproduction ratio R_0, the infection radius r and the infection proportion P_I by varying two parameters (R_0 and r) and recording the resulting P_I, and comparing them with the scenario where all devices stay still. We calculated the approximate range of infection ratio required to infect 5%, 50% and 90% of the population given 5 varied reproduction ratio. In a region of $[0, 1]^2$ with 100 uniformly distributed devices, r should be in (0.185, 0.19) for $R_0 = 0.5, P_I = 90\%$ and (0.084, 0.086) for $R_0 = 3.0, P_I = 5\%$ considering population movements, whilst r have to be at least (0.258, 0.26) for $R_0 = 0.5, P_I = 90\%$ and also in (0.084, 0.086) for $R_0 = 3.0, P_I = 5\%$.

We know for sure that larger communication radius and basic reproduction ratio lead to larger proportion of infectives, and the required infection distance without movements will generally be much greater, which means movements will encourage the prevalence of worms under the same circumstances. And the larger R_0 is and the smaller P_I is, the difference of r between with and without movements will be smaller, indicating that the case with smaller R_0 will be affected more greatly by the population movements. Also, we can infer that movements will rise even more infected cases when the devices have a relatively large communication radius r. (But considering an extreme circumstance where r is so large that every device is connected with each other, then we can foresee that movements will have no contribution to the transmission of the virus. Further studies are needed to explore the threshold.)

The study tells us that we can control the spread of the worms by limiting the movements and communication radius of the devices. As the epidemic develops further or with a smaller R_0 and r, it would be more effective to limit the movements of population. But if the virus has a relatively large basic reproduction ratio, then controlling movements will have less influence on the transmission and we need to find some other ways to suppress the spread of the worms.

5.2. Limitations

However, there do exist some limitations that need to be brought to attention. Firstly, we have made quite a few assumptions to build our model which do not reflect the real life. For instance, we assumed fixed time intervals for the movements of points between two events, but actually the intervals might change after every movement of points as the edges and the number of infected devices are very likely to vary so the actual interval would follow the distribution based on newly calculated edges conditioning the time having past, which would be too complicated and time-consuming. We also assumed identical transmission radius for every device whilst in real life different devices have different communication range.

Secondly, we used different criteria for different P_I to determine the range of required infection distance while sometimes the criteria could be unreliable. For $P_I = 90\%$, we chose the sub-range where there is no observations under 90%; however, we can infer that the more iterations we performed, the larger the expected r would be as more observations mean it would be more likely to have an unadmitted observation given identical R_0 and r. Suppose we have only 1 unadmitted observation out of 10,000 with very large r and R_0 but we still have to increase r which is unreasonable. Therefore, this criterion should only be used when the iteration number is small.

Furthermore, we have mentioned about the time-consuming issues above. When considering movements, we will recalculate the edges connecting all susceptibles and infectives (S to S, S to I, I to I) except removals by checking every pair of eligible points for which the time complexity could be $O(N^2)$. Thus, it would be almost impossible to generate a precise contour plot using this method. We are able to improve this by connecting the edges only between S and I as we need to connect S to S and I to I to visualise the transmission but we do not necessarily need to do so to realise the transmission. The process of transmission is only associated with the number of edges connecting S to I but not any other edges (See 3.1). For example, given $r = 0.13, R_0 = 3.0$ the time cost was 28.85 minutes for 500 iterations if connecting all S and I, while it was 10.68

minutes if connecting only S to I, where the time was reduced by 63%. Even if the efficiency improves to some extent, it is still not high enough for a detailed contour plot. (When ignoring movements, given $r = 0.13, R_0 = 3.0$, it took only 18.40 seconds for 500 iterations, in which 17.96 seconds are for generating an RGG without boundary and 0.44 seconds are for occurrence of 500 events.)

5.3. Further Work

Although our simulations results already revealed the influence of the movements in some ways, we were using a fixed movement pattern throughout our study. Therefore, a future refinement can take into account more complex movement patterns (e.g. not straight line, different movement time interval and distance) by which we can further explore the effect of the frequency, distance and curve of movements.

Besides, we assume all the devices in the region are uniformly distributed but in reality the devices are more likely to distribute in clusters (maybe also move in clusters as one person might bring multiple devices) so our study can be refined by considering inhomogeneity in the distribution of the population. There are also some other common situations in reality like vaccination (installing the patch, antivirus software) and isolation (shut down or move to some places where any communications are blocked) that can be considered in the future.

In addition, one of the most important thing is to improve the running speed of the simulations in the future studies. There must be some more efficient ways to rebuild the geometric graphs after movements and events. One possible way might be to record the moved points in a single time interval between two events and recalculate the edges of only the moved points before every event occurs. Or we may do not need to check every possible pair of devices but make grid of many squares with side length r so one device in a square can only be connected with any other devices in the neighbouring 3×3 squares. The simplest way might be to use more computers or parallel computing with package *foreach* and *doParallel* in R [52].

After improving the efficiency of the simulation in the future work, we can run more iterations for more R_0 and r using smaller steps to calculate more precise infection proportions and hence analyse the cases for more distinct P_I such as 10%, 20% and further explore the variation of required infection distance with varied infection rates and proportions.

REFERENCES

1. Pastor-Satorras, R. *Evolution and Structure of the Internet : A Statistical Physics Approach / Romualdo Pastor-Satorras, Alessandro Vespignani.* eng. ISBN: 9780511610905 (2004).
2. Mishra, B. K. & Keshri, N. Mathematical model on the transmission of worms in wireless sensor network. eng. *Applied mathematical modelling* **37,** 4103–4111. ISSN: 0307-904X (2013).
3. Moore, D., Shannon, C. & claffy k, k. *Code-Red: A Case Study on the Spread and Victims of an Internet Worm* in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment* (Association for Computing Machinery, Marseille, France, 2002), 273–284. ISBN: 158113603X. https://doi.org/10.1145/637201.637244.
4. Staniford, S., Paxson, V. & Weaver, N. *How to Own the Internet in Your Spare Time* in *11th USENIX Security Symposium (USENIX Security 02)* (USENIX Association, San Francisco, CA, Aug. 2002). https://www.usenix.org/conference/11th-usenix-security-symposium/how-own-internet-your-spare-time.
5. Zhang, C., Zhou, S. & Chain, B. M. Hybrid epidemics–a case study on computer worm conficker. eng. *PloS one* **10,** e0127478–e0127478. ISSN: 1932-6203 (2015).
6. Marion, J.-Y. From Turing machines to computer viruses. eng. *Philosophical transactions of the Royal Society of London. Series A: Mathematical, physical, and engineering sciences* **370,** 3319–3339. ISSN: 1364-503X (2012).
7. Kak, A. *Lecture 22: Malware: Viruses and Worms (Lecture Notes on "Computer and Network Security")* Available at https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture22.pdf (2021).
8. Rhodes, C. & Nekovee, M. The opportunistic transmission of wireless worms between mobile devices. eng. *Physica A* **387,** 6837–6844. ISSN: 0378-4371 (2008).
9. Miller, B. A. *Bluetooth revealed / Brent A. Miller, Chatschik Bisdikian.* eng. ISBN: 0130902942 (Prentice Hall, Upper Saddle River, NJ ; London, 2001).
10. Gast, M. *802.11 Wireless Networks: The Definitive Guide, 2nd Edition / Gast, Matthew.* 2nd edition. eng. ISBN: 9780596100520 (2005).
11. Hypponen, M. Malware goes mobile. *Scientific American* **295,** 70–77 (2006).
12. Hameroff, I. Malware goes mobile: worms, viruses, and Trojans coming to a mobile phone near you. (Security). **3,** 44 (2003).
13. Kak, A. *SIR Models of Epidemics – Theoretical Biology | ETH Zurich* Available at tb.ethz.ch/education/learningmaterials/modelingcourse/level-1-modules/SIR.html.
14. White, P. J. & Enright, M. C. Mathematical models in infectious disease epidemiology. *Infectious Diseases,* 70 (2010).
15. Bernoulli, D. Essai d'une nouvelle analyse de la mortalité causée par la petite vérole, et des avantages de l'inoculation pour la prévenir. *Histoire de l'Acad., Roy. Sci.(Paris) avec Mem,* 1–45 (1760).
16. Ross, R. An application of the theory of probabilities to the study of a priori pathometry.—Part I. *Proceedings of the Royal Society of London. Series A, Containing papers of a mathematical and physical character* **92,** 204–230 (1916).
17. Kermack, W. O. & McKendrick, A. G. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* **115,** 700–721 (1927).

18. Kendall, D. G. in *Volume 4 Contributions to Biology and Problems of Health* (ed Neymann, J.) 149–166 (University of California Press, 2020). https://doi.org/10.1525/9780520350717-011.
19. Allen, L. J. S. in *Mathematical Epidemiology* (eds Brauer, F., van den Driessche, P. & Wu, J.) 81–130 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008). ISBN: 978-3-540-78911-6. https://doi.org/10.1007/978-3-540-78911-6_3.
20. Erdös, P. & Rényi, A. On random graphs I. *Publ Math Debrecen* **6,** 290–297 (1959).
21. Gilbert, E. N. Random plane networks. *Journal of the society for industrial and applied mathematics* **9,** 533–543 (1961).
22. Meester, R. & Roy, R. *Continuum percolation* (Cambridge University Press, 1996).
23. Mertens, S. & Moore, C. Continuum percolation thresholds in two dimensions. *Physical Review E* **86,** 061109 (2012).
24. Glauche, I., Krause, W., Sollacher, R. & Greiner, M. Continuum percolation of wireless ad hoc communication networks. *Physica A: Statistical Mechanics and its Applications* **325,** 577–600. ISSN: 0378-4371. https://www.sciencedirect.com/science/article/pii/S0378437103002498 (2003).
25. Hafner, R. The asymptotic distribution of random clumps. *Computing* **10,** 335–351 (1972).
26. Godehardt, E. & Jaworski, J. On the connectivity of a random interval graph. *Random Structures & Algorithms* **9,** 137–161 (1996).
27. Harris, B. & Godehardt, E. in *Classification, Data Analysis, and Data Highways* 54–61 (Springer, 1998).
28. Godehardt, E., Jaworski, J. & Godehardt, D. in *Classification, Data Analysis, and Data Highways* 35–45 (Springer, 1998).
29. Dettmann, C. P., Georgiou, O. & Knight, G. Spectral statistics of random geometric graphs. *EPL (Europhysics Letters)* **118,** 18003 (2017).
30. Trillos, N. G., Gerlach, M., Hein, M. & Slepčev, D. Error estimates for spectral convergence of the graph Laplacian on random geometric graphs toward the Laplace–Beltrami operator. *Foundations of Computational Mathematics* **20,** 827–887 (2020).
31. Waxman, B. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications* **6,** 1617–1622 (1988).
32. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M. & Hwang, D.-U. Complex networks: Structure and dynamics. *Physics reports* **424,** 175–308 (2006).
33. Faloutsos, M., Faloutsos, P. & Faloutsos, C. in *The Structure and Dynamics of Networks* 195–206 (Princeton University Press, 2011).
34. Penrose, M. *Random geometric graphs / Mathew Penrose.* eng. ISBN: 0198506260 (Oxford University Press, Oxford, 2003).
35. La Polla, M., Martinelli, F. & Sgandurra, D. A survey on security for mobile devices. *IEEE communications surveys & tutorials* **15,** 446–471 (2012).
36. Gibson, G. Investigating mechanisms of spatiotemporal epidemic spread using stochastic models. *Phytopathology* **87,** 139–146 (1997).
37. Ball, F. Deterministic and stochastic epidemics with several kinds of susceptibles. *Advances in applied probability* **17,** 1–22 (1985).
38. Isham, V., Harden, S. & Nekovee, M. Stochastic epidemics and rumours on finite random networks. eng. *Physica A* **389,** 561–576. ISSN: 0378-4371 (2010).
39. Yulmetyev, R. *et al. Fluctuations and noise in stochastic spread of respiratory infection epidemics in social networks* in *AIP Conference Proceedings* **665** (2003), 408–420.
40. Nekovee, M. Worm epidemics in wireless ad hoc networks. eng. *New journal of physics* **9,** 189–189. ISSN: 1367-2630 (2007).
41. Beckley, R. *et al.* Modeling epidemics with differential equations. eng. *Tennessee State University Internal Report* (2013).

42. Sanche, S. *et al.* High Contagiousness and Rapid Spread of Severe Acute Respiratory Syndrome Coronavirus 2. eng. *Emerging infectious diseases* **26,** 1470–1477. ISSN: 1080-6040 (2020).
43. Banavar, J. R., Maritan, A. & Rinaldo, A. Size and form in efficient transportation networks. *Nature* **399,** 130–132. https://doi.org/10.1038/20144 (1999).
44. Glauche, I., Krause, W., Sollacher, R. & Greiner, M. Distributive routing and congestion control in wireless multihop ad hoc communication networks. *Physica A: Statistical Mechanics and its Applications* **341,** 677–701. ISSN: 0378-4371. https://www.sciencedirect.com/science/article/pii/S0378437104005540 (2004).
45. Krause, W., Glauche, I., Sollacher, R. & Greiner, M. Impact of network structure on the capacity of wireless multihop ad hoc communication. *Physica A: Statistical Mechanics and its Applications* **338,** 633–658. ISSN: 0378-4371. https://www.sciencedirect.com/science/article/pii/S0378437104002687 (2004).
46. Xia, W. & Thorpe, M. F. Percolation properties of random ellipses. *Phys. Rev. A* **38,** 2650–2656. https://link.aps.org/doi/10.1103/PhysRevA.38.2650 (5 1988).
47. Balberg, I. Universal percolation-threshold limits in the continuum. *Phys. Rev. B* **31,** 4053–4055. https://link.aps.org/doi/10.1103/PhysRevB.31.4053 (6 1985).
48. Alon, U., Drory, A. & Balberg, I. Systematic derivation of percolation thresholds in continuum systems. *Phys. Rev. A* **42,** 4634–4638. https://link.aps.org/doi/10.1103/PhysRevA.42.4634 (8 1990).
49. Erdös, P. & Rényi, A. in *The structure and dynamics of networks* 38–82 (Princeton University Press, 2011).
50. Newman, M. E. J., Strogatz, S. H. & Watts, D. J. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E* **64,** 026118. https://link.aps.org/doi/10.1103/PhysRevE.64.026118 (2 2001).
51. Scott, J. Social Network Analysis. *Sociology* **22,** 109–127. eprint: https://doi.org/10.1177/0038038588022001007. https://doi.org/10.1177/0038038588022001007 (1988).
52. Jones, M. *Quick Intro to Parallel Computing in R* Available at https://nceas.github.io/oss-lessons/parallel-computing-in-r/parallel-computing-in-r.html (2017).

A. TABLE FOR AN OVERVIEW OF THE AVERAGE INFECTION PROPORTION

Ratio	Distance																
	0.06	**0.08**	**0.10**	**0.12**	0.125	0.13	0.135	**0.14**	0.145	0.15	0.155	**0.16**	0.165	0.17	0.175	**0.18**	**0.20**
0.5	1.9	2.1	2.5	8.5	/	/	/	25.2	18	18.3	55.7	42.7	54.8	72.4	72.9	65.9	79.7
Improper No.*	/	/	/	/	/	/	/	/	/	/	2	3	4	2	2	3	2
Post-average**	/	/	/	/	/	/	/	/	/	/	69.4	60.3	89.7	90.1	90.5	93.6	99.4
1.0	2	2.5	6.2	10.8	11.5	24.1	21.4	54.9	56.3	67.9	67	97.7					
Improper No.	/	/	/	/	/	/	/	3	4	3	3	0					
Post-average	/	/	/	/	/	/	/	77.6	93	96.4	94.6	97.7					
1.5	2.7	2.8	5.5	25.8	32.7	57.4	80.1	88.3	77.2	88.3	79	69.3					
Improper No.	/	/	/	/	/	2	1	0	2	1	2	3					
Post-average	/	/	/	/	/	71.5	88.9	88.3	96.3	98	98.4	98.6					
2.0	2.2	3.4	6.3	41.5	35.6	61.1	76.8	85.5	/	/	/	90					
Improper No.	/	/	/	/	3	1	2	1	/	/	/	1					
Post-average	/	/	/	/	50.1	67.8	95.5	94.9	/	/	/	99.9					
3.0	2.4	2	8.5	46	76.2	82.3	87.1	98.8									
Improper No.	/	/	/	/	0	1	1	0									
Post-average	/	/	/	/	76.2	91.3	96.7	98.8									

*The number of improper simulations. *Improper* refers to the simulations where the virus died out quickly and there are less than 5 individuals in total that got infected. We only calculate this when the average proportion exceeds 50% indicating a widely spread virus. Note this number is just for reference and does not necessarily mean anything.

** The average proportion of infectives regardless of the improper simulations.

B. R CODE FOR THE SIMULATIONS

```r
require(igraph)
require(xlsx)
require(plotly)

# Run the code using multi-cores if applicable
# require(foreach)
# require(doParallel)
# registerDoParallel(3)



### Get the distance of two points using the difference of x&y coordinates
get.dist = function(x, y)
{
  sqrt(x^2 + y^2)
}



### Move a group of points and recalculate edges without R
move = function(g, id.m, R)
{
  if(length(id.m) == 0) return(g)

  for (i in 1:length(id.m))
  {
    d <- rnorm(1, 0.3, 0.01)
    angle <- runif(1, 0, 2*pi)

    V(g)$x[id.m[i]] <- V(g)$x[id.m[i]] + cos(angle) * d
    V(g)$y[id.m[i]] <- V(g)$y[id.m[i]] + sin(angle) * d

    if (V(g)$x[id.m[i]] < 0) V(g)$x[id.m[i]] = V(g)$x[id.m[i]] + 1
    if (V(g)$x[id.m[i]] > 1) V(g)$x[id.m[i]] = V(g)$x[id.m[i]] - 1
    if (V(g)$y[id.m[i]] < 0) V(g)$y[id.m[i]] = V(g)$y[id.m[i]] + 1
    if (V(g)$y[id.m[i]] > 1) V(g)$y[id.m[i]] = V(g)$y[id.m[i]] - 1
  }

  g <- g %>% delete_edges(E(g)[.inc(1:num)])

  SI <- (1:num)[!(1:num) %in% R]

  SI.num <- length(SI)

  if (SI.num == 1 | SI.num == 0) return(g) # One last infected individual remained

  for (i in 1:(SI.num-1))
  {
    for (j in (i+1) : SI.num)
    {
      dist.x <- abs(V(g)[SI[i]]$x - V(g)[SI[j]]$x)
      dist.y <- abs(V(g)[SI[i]]$y - V(g)[SI[j]]$y)

      if (dist.x > 0.5) dist.x <- 1 - dist.x
      if (dist.y > 0.5) dist.y <- 1 - dist.y

      if (get.dist(dist.x, dist.y) <= r)
      {
        g <- g %>% add_edges(c(SI[i], SI[j]))
      }
    }
  }

  return(g)
}



### Move a group of points and recalculate edges only between S and I
move.simple = function(g, id.m, S, I)
```

```
{
  if(length(id.m) == 0) return(g)

  for (i in 1:length(id.m))
  {
    d <- rnorm(1, 0.3, 0.01)
    angle <- runif(1, 0, 2*pi)

    V(g)$x[id.m[i]] <- V(g)$x[id.m[i]] + cos(angle) * d
    V(g)$y[id.m[i]] <- V(g)$y[id.m[i]] + sin(angle) * d

    if (V(g)$x[id.m[i]] < 0) V(g)$x[id.m[i]] = V(g)$x[id.m[i]] + 1
    if (V(g)$x[id.m[i]] > 1) V(g)$x[id.m[i]] = V(g)$x[id.m[i]] - 1
    if (V(g)$y[id.m[i]] < 0) V(g)$y[id.m[i]] = V(g)$y[id.m[i]] + 1
    if (V(g)$y[id.m[i]] > 1) V(g)$y[id.m[i]] = V(g)$y[id.m[i]] - 1
  }

  g <- g %>% delete_edges(E(g)[.inc(1:num)])

  if(length(S) == 0 | length(I) == 0) return(g)

  for (i in 1 : length(S))
  {
    for (j in 1 : length(I))
    {
      dist.x <- abs(V(g)[S[i]]$x - V(g)[I[j]]$x)
      dist.y <- abs(V(g)[S[i]]$y - V(g)[I[j]]$y)

      if (dist.x > 0.5) dist.x <- 1 - dist.x
      if (dist.y > 0.5) dist.y <- 1 - dist.y

      if (get.dist(dist.x, dist.y) <= r)
      {
        g <- g %>% add_edges(c(S[i], I[j]))
      }
    }
  }

  return(g)
}



### Move points without calculating edges
move.noedge = function(g, id.m)
{
  d <- rnorm(1, 0.3, 0.01)
  angle <- runif(1, 0, 2*pi)

  V(g)$x[id.m] <- V(g)$x[id.m] + cos(angle) * d
  V(g)$y[id.m] <- V(g)$y[id.m] + sin(angle) * d

  if (V(g)$x[id.m] < 0) V(g)$x[id.m] = V(g)$x[id.m] + 1
  if (V(g)$x[id.m] > 1) V(g)$x[id.m] = V(g)$x[id.m] - 1
  if (V(g)$y[id.m] < 0) V(g)$y[id.m] = V(g)$y[id.m] + 1
  if (V(g)$y[id.m] > 1) V(g)$y[id.m] = V(g)$y[id.m] - 1

  return(g)
}



### Choose every entry in vector x with probability p
choosep = function(x, p)
{
  res <- numeric(length = 0)

  if (length(x) == 0) return(res)

  prob <- runif(length(x))
  for (i in 1:length(x)) {
    if (prob[i] < p)
    {
```

```
      res <- append(res, x[i])
    }
  }
  return(res)
}



### Remove all the edges across the boundary (for figures showing unbordered edges)
boundary = function(g, r)
{
  temp.data <- as_data_frame(g, "both")

  id.remove <- numeric(0)

  for (i in 1:length(E(g))) {

    dist.x <- V(g)[temp.data$edges$from[i]]$x - V(g)[temp.data$edges$to[i]]$x
    dist.y <- V(g)[temp.data$edges$from[i]]$y - V(g)[temp.data$edges$to[i]]$y

    if (get.dist(dist.x, dist.y) > r)
    {
      id.remove <- append(id.remove, i)
    }
  }

  g <- g %>% delete_edges(id.remove)

  return(g)
}




for (beta in c(1, 2, 3, 4, 6))
{
  for (r in seq(0.13, 0.135, 0.005))
  {
    record = numeric()
    count = 1

    time.begin <- Sys.time() # Record runing time
    ###################################

    num = 100  # Number of nodes
    r = 0.13   # Infective distance

    ### R0 = beta/delta
    beta = 6   # Infection rate
    delta = 2  # Recovery rate

    ###################################

    # set.seed(2020)
    set.seed(714)
    g1 <- sample_grg(num, r, coords = TRUE)
    # plot(g1, vertex.size = 1, vertex.label = NA)# plot.igraph

    ## Plot layout
    ### 2 3 4 ###
    ### 5 1 6 ###
    ### 7 8 9 ###

    ### g1: graph with boundary
    ### g0: graph without boundary

    g9 = g8 = g7 = g6 = g5 = g4 = g3 = g2 = g1

    V(g2)$x <- V(g1)$x - 1
    V(g2)$y <- V(g1)$y + 1

    V(g3)$x <- V(g1)$x
```

```
  V(g3)$y <- V(g1)$y + 1

  V(g4)$x <- V(g1)$x + 1
  V(g4)$y <- V(g1)$y + 1

  V(g5)$x <- V(g1)$x - 1
  V(g5)$y <- V(g1)$y

  V(g6)$x <- V(g1)$x + 1
  V(g6)$y <- V(g1)$y

  V(g7)$x <- V(g1)$x - 1
  V(g7)$y <- V(g1)$y - 1

  V(g8)$x <- V(g1)$x
  V(g8)$y <- V(g1)$y - 1

  V(g9)$x <- V(g1)$x + 1
  V(g9)$y <- V(g1)$y - 1

  g <- g1 + g2 + g3 + g4 + g5 + g6 + g7 + g8 + g9

  ### Connect the points on the boundary
  id <- which((((V(g)$x > -r & V(g)$x < r)|(V(g)$x > 1-r & V(g)$x < 1+r))
               & (V(g)$y > -r & V(g)$y < 1+r))
              | (((V(g)$y > -r & V(g)$y < r)|(V(g)$y > 1-r & V(g)$y < 1+r)))
              & (V(g)$x > -r & V(g)$x < 1+r))
  id_1 <- sum(V(g)[id] <= num)

  for (i in 1:id_1)
  {
    for (j in (id_1+1):length(id))
    {
      if (get.dist(V(g)[id[i]]$x - V(g)[id[j]]$x,
                   V(g)[id[i]]$y - V(g)[id[j]]$y) <= r)
      {
        g <- g %>% add_edges(c(id[i], id[j]))
      }
    }
  }

  temp.data <- as_data_frame(g, what = "both")
  temp.edges <- unique(t(apply((temp.data$edges-1)%%num+1, 1, sort)))
  temp.data$edges <- list(from = temp.edges[, 1], to = temp.edges[, 2])

  g0 <- g1

  for (i in (length(E(g0))+1) : length(temp.data$edges$from))
  {
    g0 <- g0 %>% add_edges(c(temp.data$edges$from[i], temp.data$edges$to[i]))
  }

  # plot(g0, vertex.size = 1, vertex.label = NA) # Connection Plot Check

  ### Simulation

  iter.num = 20   # Number of iterations for on combination of R0 and r

  num.I <- matrix(nrow = iter.num, ncol = 2*num)

  set.seed(2021)
  # set.seed(714)

  # foreach (iter = 1:iter.num) %do%
  for (iter in 1:iter.num)
    {
    temp.g <- g0

    p1 <- 0.01
    p2 <- 0.9

    # number of points moving
    pt.n <- round(p1/(1+p1-p2)*num)
```

```
      pt.move <- sample(num, pt.n)
      pt.still <- 1:num
      pt.still <- pt.still[!pt.still %in% pt.move]


      S <- as.numeric(V(g0))
      I <- sample(1:num, 1)
      S <- S[S != I]
      R <- numeric(length = 0)

      num.I[iter, 1] <- 1

      times = 0

      while (is.null(I) == FALSE)
      # for (i in 1 : 85) # Terminate the infection process at specified point
      {
        a <- beta * length(E(g0)[S %--% I])
        b <- delta * length(I)

        if(a+b == 0) break
        # if(times == 1000) break

        # time.int <- floor(rexp(1, 1/(a+b)))

        time.int <- rexp(1, a+b) * 160

        while (time.int > 0)
        {
          pt.mtos <- choosep(pt.move, 1-p2)
          pt.stom <- choosep(pt.still, p1)

          pt.move <- pt.move[!pt.move %in% pt.mtos]
          pt.move <- append(pt.move, pt.stom)

          pt.still <- pt.still[!pt.still %in% pt.stom]
          pt.still <- append(pt.still, pt.mtos)

          # pt.move.rec <- append(pt.move.rec, pt.move)

          for (j in pt.move) {
            temp.g <- move.noedge(temp.g, j)
          }

          time.int <- time.int - runif(1, 0.8, 1.2)
        }

        pt.mtos <- choosep(pt.move, 1-p2)
        pt.stom <- choosep(pt.still, p1)

        pt.move <- pt.move[!pt.move %in% pt.mtos]
        pt.move <- append(pt.move, pt.stom)

        temp.g <- move(temp.g, pt.move, R)

        ### If only edges between S and I are needed
        ### Comment the above line and use the line below
        # temp.g <- move.simple(temp.g, pt.move, S, I)


        ### New graph after points moving has been obtained ###

        event <- sample(c(0, 1), size = 1, prob = c( a/(a+b), b/(a+b)))

        ### Infection
        if(event == 0)
        {
          temp.infect.list <- ends(g0, E(g0)[.inc(I)])[!ends(g0, E(g0)[.inc(I)]) %in% I
    ]
          temp.infect.list <- temp.infect.list[!temp.infect.list %in% R]

          if (length(temp.infect.list) == 1){
            temp.infect <- temp.infect.list
```

```
          }
          else {
            temp.infect <- sample(temp.infect.list, 1)
          }

          I <- append(I, temp.infect)
          S <- S[S != temp.infect]
        }

        ### Recovery
        if(event == 1)
        {
          if (length(I) == 1) temp.recover = I
          else temp.recover <- sample(I, 1)
          I <- I[I != temp.recover]
          R <- append(R, temp.recover)
        }
        # break
        time.end <- Sys.time()

        record[count] = time.end - time.begin
        count = count + 1

        times = times + 1

        num.I[iter, times+1] <- length(I)
      }
      # cat("End of iteration", iter, "\n")
    }

    sheet.name <- paste("Ratio", beta/delta, "Dist", r, sep = "")

    write.xlsx(num.I, 'Data_Infect_temp.xlsx', sheetName = sheet.name, col.names = TRUE
    ,
               row.names = TRUE, append = TRUE, showNA = FALSE)

    time.end <- Sys.time()

    cat("End of","Ratio", beta/delta, "Dist", r, "\n")

    cat(time.end - time.begin, "\n")
  }
}

# write.xlsx(record2, 'Time_Record.xlsx', sheetName = "Record2", col.names = TRUE,
#            row.names = TRUE, append = TRUE, showNA = FALSE)



### Code to plot figure in the middle of the virus transmission
### where ‘temp.g‘ is a geometric graph generated by the above code

### Generate a random graph for debugging
# temp.g <- sample_grg(100, 0.15, coords = TRUE)
# plot(temp.g, vertex.size = 1, vertex.label = NA)

g1.f <- temp.g

g1.f <- boundary(temp.g, r)

g9.f = g8.f = g7.f = g6.f = g5.f = g4.f = g3.f = g2.f = g1.f

V(g2.f)$x <- V(g1.f)$x - 1
V(g2.f)$y <- V(g1.f)$y + 1

V(g3.f)$x <- V(g1.f)$x
V(g3.f)$y <- V(g1.f)$y + 1

V(g4.f)$x <- V(g1.f)$x + 1
V(g4.f)$y <- V(g1.f)$y + 1

V(g5.f)$x <- V(g1.f)$x - 1
V(g5.f)$y <- V(g1.f)$y
```

```
434
435 V(g6.f)$x <- V(g1.f)$x + 1
436 V(g6.f)$y <- V(g1.f)$y
437
438 V(g7.f)$x <- V(g1.f)$x - 1
439 V(g7.f)$y <- V(g1.f)$y - 1
440
441 V(g8.f)$x <- V(g1.f)$x
442 V(g8.f)$y <- V(g1.f)$y - 1
443
444 V(g9.f)$x <- V(g1.f)$x + 1
445 V(g9.f)$y <- V(g1.f)$y - 1
446
447 g.f <- g1.f + g2.f + g3.f + g4.f + g5.f + g6.f + g7.f + g8.f + g9.f
448
449 ### Connect the points on the boundary
450 id <- which((((V(g.f)$x > -r & V(g.f)$x < r)|(V(g.f)$x > 1-r & V(g.f)$x < 1+r))
451              & (V(g.f)$y > -r & V(g.f)$y < 1+r))
452             | (((V(g.f)$y > -r & V(g.f)$y < r)|(V(g.f)$y > 1-r & V(g.f)$y < 1+r)))
453             & (V(g.f)$x > -r & V(g.f)$x < 1+r))
454 id_1 <- sum(V(g.f)[id] <= num)
455
456 for (i in 1:id_1)
457 {
458   for (j in (id_1+1):length(id))
459   {
460     if (get.dist(V(g.f)[id[i]]$x - V(g.f)[id[j]]$x,
461                  V(g.f)[id[i]]$y - V(g.f)[id[j]]$y) <= r)
462     {
463       g.f <- g.f %>% add_edges(c(id[i], id[j]))
464     }
465   }
466 }
467
468 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R)])         # Remove Recovery edges
469 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R+1*num)])   # Remove Recovery edges
470 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R+2*num)])   # Remove Recovery edges
471 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R+3*num)])   # Remove Recovery edges
472 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R+4*num)])   # Remove Recovery edges
473 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R+5*num)])   # Remove Recovery edges
474 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R+6*num)])   # Remove Recovery edges
475 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R+7*num)])   # Remove Recovery edges
476 g.f <- g.f %>% delete_edges(E(g.f)[.inc(R+8*num)])   # Remove Recovery edges
477
478
479 g.f.data <- as_data_frame(g.f, "both")
480
481 id <- which(g.f.data$edges$from <= num | g.f.data$edges$to <= num)
482 id.i <- which(g.f.data$edges$from %in% I | g.f.data$edges$to %in% I)
483
484
485 ### Save the figure with higher resolutions
486 # tiff("fig_status_0.tiff", units="in", width=5, height=5, res=300)
487
488 ### Plot the connection considering boundary
489 plot.new()
490 plot.window(xlim=c(0,1), ylim=c(0,1), xaxs="i", yaxs="i")
491
492 for (i in id)
493 {
494   segments(x0 = g.f.data$vertices$x[g.f.data$edges$from[i]],
495            y0 = g.f.data$vertices$y[g.f.data$edges$from[i]],
496            x1 = g.f.data$vertices$x[g.f.data$edges$to[i]],
497            y1 = g.f.data$vertices$y[g.f.data$edges$to[i]],
498            lwd = 0.5, col = "#00FF00")
499 }
500
501 for (i in id.i)
502 {
503   segments(x0 = g.f.data$vertices$x[g.f.data$edges$from[i]],
504            y0 = g.f.data$vertices$y[g.f.data$edges$from[i]],
505            x1 = g.f.data$vertices$x[g.f.data$edges$to[i]],
506            y1 = g.f.data$vertices$y[g.f.data$edges$to[i]],
```

```
            lwd = 0.5, col = "#FF0000")
}



points(V(temp.g)$x[S], V(temp.g)$y[S], pch = 21, col = '#9900FF', bg = '#9900FF', cex = 0.5) # Blue
points(V(temp.g)$x[I], V(temp.g)$y[I], pch = 21, col = '#FF0000', bg = '#FF0000', cex = 0.5) # Red
points(V(temp.g)$x[R], V(temp.g)$y[R], pch = 21, col = '#7D7D7D', bg = '#7D7D7D', cex = 0.5) # Grey

### This is the code to distinguish moving points for figure 3.2 # (\ref{fig:eg:move})
# points(V(temp.g)$x, V(temp.g)$y, pch = 21, col = '#9900FF', bg = '#9900FF', cex = 0.5)
# points(V(temp.g)$x[pt.move.rec], V(temp.g)$y[pt.move.rec], pch = 21, col = '#FF0000', bg = '#FF0000', cex = 0.5)

rect(-1, -1, 0, 2, col = "white", border = NA)
rect(-1, 1, 2, 2, col = "white", border = NA)
rect(1, -1, 2, 2, col = "white", border = NA)
rect(-1, -1, 2, 0, col = "white", border = NA)
axis(1)
axis(2)
box()


# dev.off()


### An example of RGG for Figure 2.2 # (\ref{fig:eg})
tiff("fig_eg.tiff", units="in", width=5, height=5, res=300)
set.seed(129)
g.test <- sample_grg(100, 0.15, coords = TRUE)
plot(g.test, vertex.size = 1, vertex.label = NA)
dev.off()


### Code for a plot showing points connected across the boundary
### Figure 2.3 #(\ref{fig:eg:pts})
tiff("connect_eg.tiff", units="in", width=5, height=5, res=300)
plot.new()
plot.window(xlim=c(0,1), ylim=c(0,1), xaxs="i", yaxs="i")
points(c(0.1, 0.9), c(0.3, 0.5) , pch = 21, col = '#9900FF', bg = '#9900FF', cex = 1.5)
segments(0.1, 0.3, -0.1, 0.5, lwd = 2)
segments(1.1, 0.3, 0.9, 0.5, lwd = 2)
segments(0.1, 0.3, 0.9, 0.5, lty = 2)
points(0.5, 0.4, pch = 4, cex = 2, col = "#FF0000", lwd = 2)
rect(-1, -1, 0, 2, col = "white", border = NA)
rect(-1, 1, 2, 2, col = "white", border = NA)
rect(1, -1, 2, 2, col = "white", border = NA)
rect(-1, -1, 2, 0, col = "white", border = NA)
axis(1)
axis(2)
box()
dev.off()




### Without Movements with tracking I in details
### Only used when analyzing what happened during an epidemic
{
  num = 100  # Number of nodes
  r = 0.13   # Infective distance

  beta = 6   # Infection rate
  delta = 2  # Recovery rate

  ###################################

  # set.seed(2020)
```

```r
set.seed(714)
g1 <- sample_grg(num, r, coords = TRUE)

g9 = g8 = g7 = g6 = g5 = g4 = g3 = g2 = g1

V(g2)$x <- V(g1)$x - 1
V(g2)$y <- V(g1)$y + 1

V(g3)$x <- V(g1)$x
V(g3)$y <- V(g1)$y + 1

V(g4)$x <- V(g1)$x + 1
V(g4)$y <- V(g1)$y + 1

V(g5)$x <- V(g1)$x - 1
V(g5)$y <- V(g1)$y

V(g6)$x <- V(g1)$x + 1
V(g6)$y <- V(g1)$y

V(g7)$x <- V(g1)$x - 1
V(g7)$y <- V(g1)$y - 1

V(g8)$x <- V(g1)$x
V(g8)$y <- V(g1)$y - 1

V(g9)$x <- V(g1)$x + 1
V(g9)$y <- V(g1)$y - 1

g <- g1 + g2 + g3 + g4 + g5 + g6 + g7 + g8 + g9


### Connect the points on the boundary
id <- which((((V(g)$x > -r & V(g)$x < r)|(V(g)$x > 1-r & V(g)$x < 1+r))
             & (V(g)$y > -r & V(g)$y < 1+r))
            | (((V(g)$y > -r & V(g)$y < r)|(V(g)$y > 1-r & V(g)$y < 1+r)))
            & (V(g)$x > -r & V(g)$x < 1+r))
id_1 <- sum(V(g)[id] <= num)

for (i in 1:id_1)
{
  for (j in (id_1+1):length(id))
  {
    if (get.dist(V(g)[id[i]]$x - V(g)[id[j]]$x,
                 V(g)[id[i]]$y - V(g)[id[j]]$y) <= r)
    {
      g <- g %>% add_edges(c(id[i], id[j]))
    }
  }
}


temp.data <- as_data_frame(g, what = "both")
temp.edges <- unique(t(apply((temp.data$edges-1)%%num+1, 1, sort)))
temp.data$edges <- list(from = temp.edges[, 1], to = temp.edges[, 2])

g0 <- g1

for (i in (length(E(g0))+1) : length(temp.data$edges$from))
{
  g0 <- g0 %>% add_edges(c(temp.data$edges$from[i], temp.data$edges$to[i]))
}



### Simulation

iter.num = 20

num.l <- matrix(nrow = iter.num, ncol = 2*num)

set.seed(2021)
```

```r
  for (iter in 1:iter.num) {
    temp.g <- g0

    S <- as.numeric(V(g0))
    I <- sample(1:num, 1)
    S <- S[S != I]
    R <- numeric(length = 0)

    num.I[iter, 1] <- 1

    times = 0

    while (is.null(I) == FALSE)
    {
      a <- beta * length(E(g0)[S %--% I])
      b <- delta * length(I)

      if(a+b == 0) break

      event <- sample(c(0, 1), size = 1, prob = c( a/(a+b), b/(a+b)))

      ### Infection
      if(event == 0)
      {
        temp.infect.list <- ends(g0, E(g0)[.inc(I)])[!ends(g0, E(g0)[.inc(I)]) %in% I]
        temp.infect.list <- temp.infect.list[!temp.infect.list %in% R]

        if (length(temp.infect.list) == 1){
          temp.infect <- temp.infect.list
        }
        else {
          temp.infect <- sample(temp.infect.list, 1)
        }

        I <- append(I, temp.infect)
        S <- S[S != temp.infect]
      }

      ### Recovery
      if(event == 1)
      {
        if (length(I) == 1) temp.recover = I
        else temp.recover <- sample(I, 1)
        I <- I[I != temp.recover]
        R <- append(R, temp.recover)
      }
      # break
      times = times + 1

      num.I[iter, times+1] <- length(I)

    }
    cat("End of iteration", iter,"\n")
  }

  # sheet.name <- paste("Ratio", beta/delta, "Dist", r, sep = "")
  #
  # write.xlsx(num.I, 'Data_Infect_Nomove_temp.xlsx', sheetName = sheet.name, col.names
     = TRUE,
  #            row.names = TRUE, append = TRUE, showNA = FALSE)

}




### Without Movements with tracking ONLY average proportions
### Used in Section 4.2, which only requires the average
{
  r.start <- 0.08
  r.end <- 0.25
  r.step <- 0.005
```

```
720
721 R0.start <- 0.5
722 R0.end <- 3.0
723 R0.step <- 0.05
724
725 r.range <- seq(r.start, r.end, r.step)
726 R0.range <- seq(R0.start, R0.end, R0.step)
727
728 AverI <- matrix(nrow = length(r.range), ncol = length(R0.range))
729
730 time.begin <- Sys.time()
731
732 for (r in r.range)
733 # foreach (r = r.range) %dopar%
734 {
735   for (R0 in R0.range)
736   {
737     ###################################
738
739     num = 100  # Number of nodes
740
741     beta = 2 * R0   # Infection rate
742     delta = 2  # Recovery rate
743
744     ###################################
745
746     # set.seed(2020)
747     set.seed(714)
748     # iter.graph.num = 3
749     iter.graph.num = 5
750     tempAverI <- numeric(iter.graph.num)
751
752     for (i.g in 1:iter.graph.num)
753     {
754       g1 <- sample_grg(num, r, coords = TRUE)
755
756       g9 = g8 = g7 = g6 = g5 = g4 = g3 = g2 = g1
757
758       V(g2)$x <- V(g1)$x - 1
759       V(g2)$y <- V(g1)$y + 1
760
761       V(g3)$x <- V(g1)$x
762       V(g3)$y <- V(g1)$y + 1
763
764       V(g4)$x <- V(g1)$x + 1
765       V(g4)$y <- V(g1)$y + 1
766
767       V(g5)$x <- V(g1)$x - 1
768       V(g5)$y <- V(g1)$y
769
770       V(g6)$x <- V(g1)$x + 1
771       V(g6)$y <- V(g1)$y
772
773       V(g7)$x <- V(g1)$x - 1
774       V(g7)$y <- V(g1)$y - 1
775
776       V(g8)$x <- V(g1)$x
777       V(g8)$y <- V(g1)$y - 1
778
779       V(g9)$x <- V(g1)$x + 1
780       V(g9)$y <- V(g1)$y - 1
781
782       g <- g1 + g2 + g3 + g4 + g5 + g6 + g7 + g8 + g9
783
784
785       ### Connect the points on the boundary
786       id <- which((((V(g)$x > -r & V(g)$x < r)|(V(g)$x > 1-r & V(g)$x < 1+r))
787                    & (V(g)$y > -r & V(g)$y < 1+r))
788                   | (((V(g)$y > -r & V(g)$y < r)|(V(g)$y > 1-r & V(g)$y < 1+r)))
789                   & (V(g)$x > -r & V(g)$x < 1+r))
790       id_1 <- sum(V(g)[id] <= num)
791
792       for (i in 1:id_1)
```

```
793         {
794           for (j in (id_1+1):length(id))
795           {
796             if (get.dist(V(g)[id[i]]$x - V(g)[id[j]]$x,
797                          V(g)[id[i]]$y - V(g)[id[j]]$y) <= r)
798             {
799               g <- g %>% add_edges(c(id[i], id[j]))
800             }
801           }
802         }
803 
804         temp.data <- as_data_frame(g, what = "both")
805         temp.edges <- unique(t(apply((temp.data$edges-1)%%num+1, 1, sort)))
806         temp.data$edges <- list(from = temp.edges[, 1], to = temp.edges[, 2])
807 
808         g0 <- g1
809 
810         for (i in (length(E(g0))+1) : length(temp.data$edges$from))
811         {
812           g0 <- g0 %>% add_edges(c(temp.data$edges$from[i], temp.data$edges$to[i]))
813         }
814 
815 
816         ### Simulation
817 
818         # iter.num = 50
819         iter.num = 100
820 
821         num.I.overall <- numeric(iter.num)
822 
823         # set.seed(2021)
824         set.seed(714)
825 
826         for (iter in 1:iter.num) {
827           temp.g <- g0
828 
829           S <- as.numeric(V(g0))
830           I <- sample(1:num, 1)
831           S <- S[S != I]
832           R <- numeric(length = 0)
833 
834           times = 0
835 
836           while (is.null(I) == FALSE)
837           {
838             a <- beta * length(E(g0)[S %--% I])
839             b <- delta * length(I)
840 
841             if(a+b == 0) break
842 
843             event <- sample(c(0, 1), size = 1, prob = c( a/(a+b), b/(a+b)))
844 
845             ### Infection
846             if(event == 0)
847             {
848               temp.infect.list <- ends(g0, E(g0)[.inc(I)])[!ends(g0, E(g0)[.inc(I)]) %
    in% I]
849               temp.infect.list <- temp.infect.list[!temp.infect.list %in% R]
850 
851               if (length(temp.infect.list) == 1){
852                 temp.infect <- temp.infect.list
853               }
854               else {
855                 temp.infect <- sample(temp.infect.list , 1)
856               }
857 
858               I <- append(I, temp.infect)
859               S <- S[S != temp.infect]
860             }
861 
862             ### Recovery
863             if(event == 1)
864             {
```

```
            if (length(I) == 1) temp.recover = I
            else temp.recover <- sample(I, 1)
            I <- I[I != temp.recover]
            R <- append(R, temp.recover)
          }
          # break
          times = times + 1
        }
        num.I.overall[iter] = (times + 1) / 2

        # cat("End of iteration", iter ,"\n")
      }

      tempAverI[i.g] = mean(num.I.overall)
    }

    AverI[round((r-r.start)/r.step)+1, round((R0-R0.start)/R0.step)+1] = mean(tempAverI)

    cat("End of","r", r, "R0", R0, "\n")
  }
}

time.end <- Sys.time()

time.end - time.begin

}

write.xlsx(AverI, 'Data_Infect_Nomove_temp.xlsx', sheetName = "AverI", col.names = TRUE,
           row.names = TRUE, append = TRUE, showNA = FALSE)




### With Movements with tracking ONLY average proportions
### Used in Section 4.1 heatmap, which only requires the average
{

  r.start <- 0.2
  r.end <- 0.2
  r.step <- 0.01

  R0.start <- 0.6
  R0.end <- 0.8
  R0.step <- 0.2

  r.range <- seq(r.start, r.end, r.step)
  R0.range <- seq(R0.start, R0.end, R0.step)

  AverI <- matrix(nrow = length(r.range), ncol = length(R0.range))

  time.begin <- Sys.time()

  for (r in r.range)
    # foreach (r = r.range) %dopar%
  {
    for (R0 in R0.range)
    {
      ###################################

      #r = 0.088
      # R0 = 1

      num = 100  # Number of nodes

      beta = 2 * R0   # Infection rate
      delta = 2  # Recovery rate

      ###################################

```

```r
set.seed(2020)
# set.seed(714)
iter.graph.num = 1
tempAverl <- numeric(iter.graph.num)

for (i.g in 1:iter.graph.num)
{
  g1 <- sample_grg(num, r, coords = TRUE)

  g9 = g8 = g7 = g6 = g5 = g4 = g3 = g2 = g1

  V(g2)$x <- V(g1)$x - 1
  V(g2)$y <- V(g1)$y + 1

  V(g3)$x <- V(g1)$x
  V(g3)$y <- V(g1)$y + 1

  V(g4)$x <- V(g1)$x + 1
  V(g4)$y <- V(g1)$y + 1

  V(g5)$x <- V(g1)$x - 1
  V(g5)$y <- V(g1)$y

  V(g6)$x <- V(g1)$x + 1
  V(g6)$y <- V(g1)$y

  V(g7)$x <- V(g1)$x - 1
  V(g7)$y <- V(g1)$y - 1

  V(g8)$x <- V(g1)$x
  V(g8)$y <- V(g1)$y - 1

  V(g9)$x <- V(g1)$x + 1
  V(g9)$y <- V(g1)$y - 1

  g <- g1 + g2 + g3 + g4 + g5 + g6 + g7 + g8 + g9


  ### Connect the points on the boundary
  id <- which((((V(g)$x > -r & V(g)$x < r)|(V(g)$x > 1-r & V(g)$x < 1+r))
               & (V(g)$y > -r & V(g)$y < 1+r))
              | (((V(g)$y > -r & V(g)$y < r)|(V(g)$y > 1-r & V(g)$y < 1+r)))
              & (V(g)$x > -r & V(g)$x < 1+r))
  id_1 <- sum(V(g)[id] <= num)

  for (i in 1:id_1)
  {
    for (j in (id_1+1):length(id))
    {
      if (get.dist(V(g)[id[i]]$x - V(g)[id[j]]$x,
                   V(g)[id[i]]$y - V(g)[id[j]]$y) <= r)
      {
        g <- g %>% add_edges(c(id[i], id[j]))
      }
    }
  }

  temp.data <- as_data_frame(g, what = "both")
  temp.edges <- unique(t(apply((temp.data$edges-1)%%num+1, 1, sort)))
  temp.data$edges <- list(from = temp.edges[, 1], to = temp.edges[, 2])

  g0 <- g1

  for (i in (length(E(g0))+1) : length(temp.data$edges$from))
  {
    g0 <- g0 %>% add_edges(c(temp.data$edges$from[i], temp.data$edges$to[i]))
  }

  ### Simulation

  # iter.num = 50
  iter.num = 10

```

```
        num.I.overall <- numeric(iter.num)

        # set.seed(2021)
        set.seed(714)

        for (iter in 1:iter.num) {
          temp.g <- g0

          p1 <- 0.01
          p2 <- 0.9

          # number of points moving
          pt.n <- round(p1/(1+p1-p2)*num)
          pt.move <- sample(num, pt.n)
          pt.still <- 1:num
          pt.still <- pt.still[!pt.still %in% pt.move]


          S <- as.numeric(V(g0))
          I <- sample(1:num, 1)
          S <- S[S != I]
          R <- numeric(length = 0)


          times = 0

          while (is.null(I) == FALSE)
          {
            a <- beta * length(E(g0)[S %--% I])
            b <- delta * length(I)

            if(a+b == 0) break

            time.int <- rexp(1, a+b) * 160

            while (time.int > 0)
            {
              pt.mtos <- choosep(pt.move, 1-p2)
              pt.stom <- choosep(pt.still, p1)

              pt.move <- pt.move[!pt.move %in% pt.mtos]
              pt.move <- append(pt.move, pt.stom)

              pt.still <- pt.still[!pt.still %in% pt.stom]
              pt.still <- append(pt.still, pt.mtos)

              # pt.move.rec <- append(pt.move.rec, pt.move)

              for (j in pt.move) {
                temp.g <- move.noedge(temp.g, j)
              }

              time.int <- time.int - runif(1, 0.8, 1.2)
            }

            pt.mtos <- choosep(pt.move, 1-p2)
            pt.stom <- choosep(pt.still, p1)

            pt.move <- pt.move[!pt.move %in% pt.mtos]
            pt.move <- append(pt.move, pt.stom)

            temp.g <- move(temp.g, pt.move, R)

            ### If only edges between S and I are needed
            ### Comment the above line and use the line below
            # temp.g <- move.simple(temp.g, pt.move, S, I)


            ### New graph after points moving has been obtained ###

            event <- sample(c(0, 1), size = 1, prob = c( a/(a+b), b/(a+b)))

            ### Infection
```

```
            if(event == 0)
            {
              temp.infect.list <- ends(g0, E(g0)[.inc(I)])[!ends(g0, E(g0)[.inc(I)]) %in% I]
              temp.infect.list <- temp.infect.list[!temp.infect.list %in% R]

              if (length(temp.infect.list) == 1){
                temp.infect <- temp.infect.list
              }
              else {
                temp.infect <- sample(temp.infect.list, 1)
              }

              I <- append(I, temp.infect)
              S <- S[S != temp.infect]
            }

            ### Recovery
            if(event == 1)
            {
              if (length(I) == 1) temp.recover = I
              else temp.recover <- sample(I, 1)
              I <- I[I != temp.recover]
              R <- append(R, temp.recover)
            }
            # break
            times = times + 1
          }
          num.I.overall[iter] = (times + 1) / 2

          cat("End of iteration", iter,"\n")
        }

        tempAverI[i.g] = mean(num.I.overall)
      }

      AverI[round((r-r.start)/r.step)+1, round((R0-R0.start)/R0.step)+1] = mean(tempAverI)

      cat("End of","r", r, "R0", R0, "\n")

      time.end <- Sys.time()
      cat("Accumulative time used is" , time.end - time.begin, "\n")
      cat(mean(tempAverI), "\n")
    }

    sheet.name <- paste("r", r, sep = "_")

    write.xlsx(AverI, 'Data_Infect_Move.xlsx', sheetName = sheet.name, col.names = TRUE,
               row.names = TRUE, append = TRUE, showNA = FALSE)
  }

}



### Read the saved data, unnecessary if 'AverI' are not polluted, just continue using it
AverI <- read.xlsx("Data_Infect_Nomove_temp.xlsx", 1, header = TRUE)[, -1]
AverI <- as.matrix(AverI)

r.start <- 0.08
r.end <- 0.21
r.step <- 0.005*2

R0.start <- 0.1
R0.end <- 3.0
R0.step <- 0.05*2

### Generate heatmap using the computed averages
```

```r
fig <- plot_ly(
  type = "contour",
  x = R0.range,
  y = r.range,
  z = Averl,
  contours = list(
    end = 90,
    size = 5,
    start = 5,
    coloring = 'heatmap',
    showlabels = TRUE),
  line = list(smoothing = 0.9)
)

fig <- fig %>% colorbar(title = "Average \nProportion \n(%)") # Add a legend title

fig




### Line plot of average infection proportions against r for varied R_0
### For Section 4.3, Figure 4.6

r1 <- read.xlsx("Data_Infect_Nomove_1.xlsx", 5, header = TRUE)
r2 <- read.xlsx("Data_Infect_Nomove_1.xlsx", 6, header = TRUE)
r3 <- read.xlsx("Data_Infect_Nomove_1.xlsx", 7, header = TRUE)
r4 <- read.xlsx("Data_Infect_Nomove_1.xlsx", 8, header = TRUE)
r5 <- read.xlsx("Data_Infect_Nomove_1.xlsx", 9, header = TRUE)


par(xpd=FALSE)
plot(r1, type = "l", xlim = c(0.08, 0.3), col = "#000000")
lines(r2, col = "#FF0000")
lines(r3, col = "#00FF00")
lines(r4, col = "#0000FF")
lines(r5, col = "#777777")

abline(h = 5, lty = 2, lwd = 0.5, col = "#BBBBBB")
abline(h = 50, lty = 2, lwd = 0.5, col = "#BBBBBB")
abline(h = 90, lty = 2, lwd = 0.5, col = "#BBBBBB")
text(0.2, 5, "5%", col = "#999999", adj = c(0, -.1), cex = 0.8)
text(0.2, 50, "50%", col = "#999999", adj = c(0, -.1), cex = 0.8)
text(0.08, 90, "90%", col = "#999999", adj = c(0, -.1), cex = 0.8)


legend("bottomright",
       legend = c("R0 = 0.5", "R0 = 1.0", "R0 = 1.5", "R0 = 2.0", "R0 = 3.0"),
       col = c("#000000", "#FF0000", "#00FF00", "#0000FF", "#777777"),
       lty = 1, cex = 0.8, lwd = 2)
```