Disease Spread Model

Modeling the Spread of Seasonal Influenza

Rita Kurban - 13 December 2017
Professor Drummond
CS111 Structure: Mathematical and Computational Models

Introduction

Seasonal influenza is an acute viral infection caused by influenza viruses which circulate all around the world. The droplets containing viruses are dispersed into the air when an infected person coughs or sneezes and spread to people nearby. The time from infection to illness, the incubation period, is around two days. The typical symptoms are a sudden onset of fever, cough, sore throat, headache, muscle and joint pain, and a runny nose. The cough can be severe and in some cases lasts more than two weeks which is the length of the infectious period.

In the United States, around 5-20% of people get infected every year. Moreover, 3,000 to 49,000 people are hospitalized because of the complications caused by the virus, such as sinus infections, bronchitis, or pneumonia. The death rate is 1.4 people per 100000. For these reasons, there is a desperate need for a reliable mathematical model that can predict the propagation of this disease to inform public health interventions.

Models

In this paper, I will use two modeling approaches (systems of differential equations and Markov chains) to analyze the spread of the influenza virus as well as identify their advantages and limitations. The first model will be based on a widely-known mathematical framework SIR — "susceptible, infectious, recovered" — which divides the population into three compartments and uses differential equations to model the change in the groups' proportions over time. I will expand upon the model by adding a fourth group of individuals and implement it in python to visualize the results. I will also use an optimization method to solve the system of linear equations numerically. The second model will be based on a different approach. I will use randomization to model the distribution of individuals in the population and Markov chains to construct a transition matrix to see how the proportions of compartments change over time. I will also create a dynamic simulation in python which will illustrate the change in the groups over time. After that, I will discuss the limitations and strengths of these models and quickly discuss how they can be used in real-life scenarios.

Compartmental model

Influenza has a latent period when a person doesn't have any symptoms of flu (they don't cough or sneeze), so they are not infectious at this stage and don't pose a risk for the rest of the population. Based on this observation, we can add a new group of

exposed individuals to the basic SIR framework. The compartments are presented in the table below:

Table 1

	The compartments of the SEIR model
S	The number of susceptible individuals who are healthy but not immune
E	The number of exposed individuals who are not infectious
I	The number of sick infectious individuals
R	The number of individuals removed from the simulation (either immune or dead)

The total number of individuals N is the sum of all the compartments.

Let's now look at every group of people separately.

- The number of susceptible individuals changes when they have contact with infected individuals. If c is per capita contact rate and p is the probability of getting infected after the contact, we can define our parameter $\alpha = pc$ — intensity of infection.
- Exposed individuals can only transfer to the infectious category after the latent period is over. The parameter that characterizes this transition is the infectious rate β. In this case, $1/\beta$ is the average time for an exposed individual to become infectious, or the incubation period.
- Infectious individuals eventually recover or die at a rate γ. The percentage of dead people is negligible compared to the number of people who recovered and got immune to this specific strain of the virus. In both scenarios, they are not participating in the simulation anymore. As in case of the exposed individuals, $1/\gamma$ is the average time it takes to recover, or infectious period.

The resulting model can be described by the following equations:

$$dS/dt = -\alpha SI/N$$
$$dE/dt = \alpha SI/N - \beta E$$
$$dI/dt = \beta E - \gamma I$$
$$dR/dt = \gamma I$$

This model is a good representation of the natural spread of the disease. However, it doesn't account for vaccination which is the primary preemptive measure in the US. Vaccination is extremely helpful, and it helps to transfer people from the susceptible group directly to the immune population. The parameter σ represents the vaccination rate:

$$dS/dt = -\alpha SI/N - \sigma S$$
$$dE/dt = \alpha SI/N - \beta E$$
$$dI/dt = \beta E - \gamma I$$
$$dR/dt = \gamma I + \sigma S$$

To apply the model in a real-life scenario and examine the effectiveness of vaccination, I modeled seasonal flu spread in NYC using Python. You can find the code in the appendix. To identify the parameters, I conducted a research and found out that the incubation period of influenza is around two days, which means that the infectious rate, β = 0.5 which is the inverse of the incubation period. On average, it takes seven days to recover from the major symptoms and up to 14 days to recover fully which leads us to ɣ=1/14. ɑ is probably the most tricky parameter, and it's one of the limitations of the model which I'll talk about in the next section. For the purpose of this simulation, we will assume that everybody has an equal probability to meet an infected individual and that the probability of getting sick is always 100% after a contact with an infectious person. According to the Centre for Disease Control and Prevention (2017), up to 20% of the population is infected with influenza during the flu season. That's why we can assume that ɑ is around 0.2.

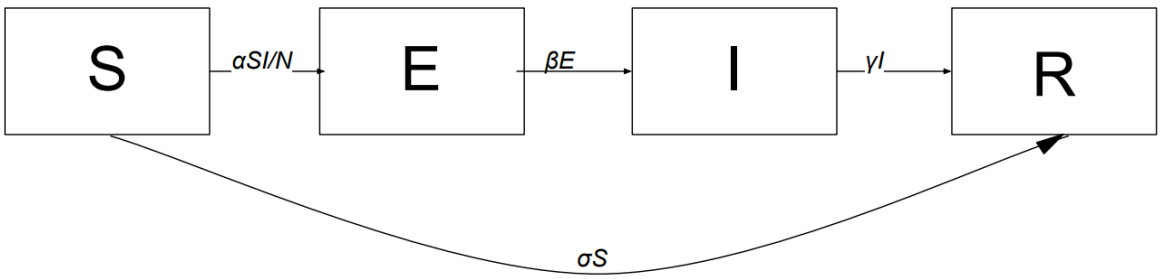


Figure 1 represents the logic of the final model with all the parameters.

Optimization

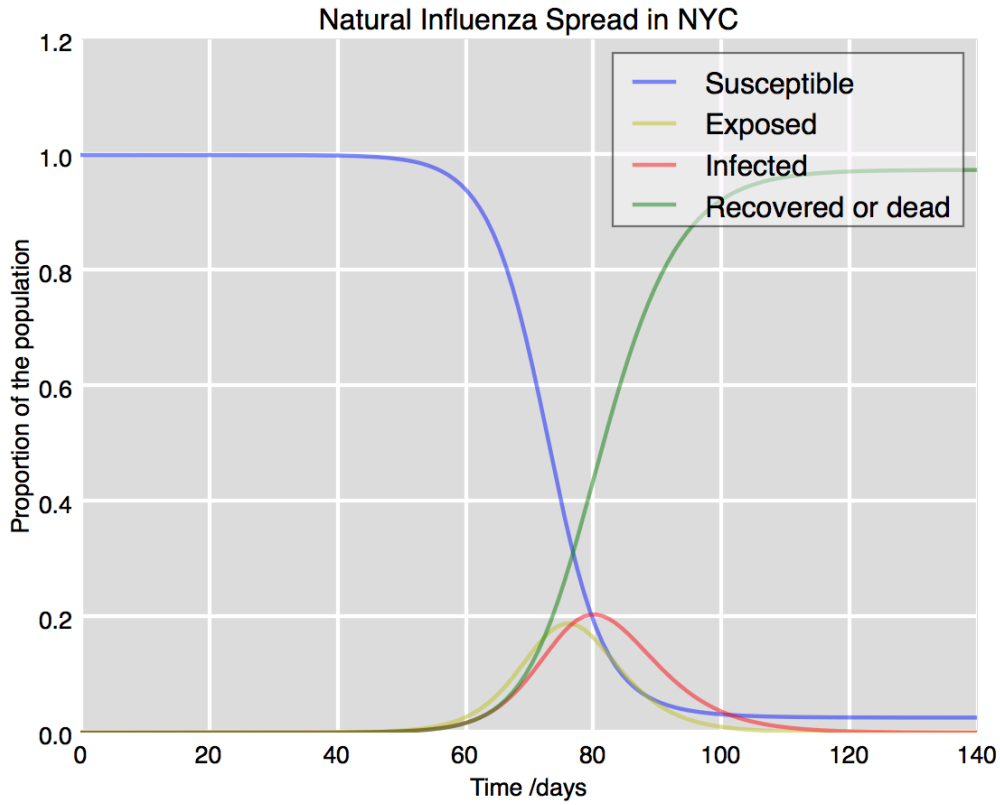
Many differential equations cannot be solved using symbolic computation. Fortunately, there are multiple numeric methods available. Numeric methods solve the systems of ODEs by finding numerical approximations to the solutions. For the purpose of this model, a numeric approximation is sufficient as we are only interested in the change of the proportions in the compartments which doesn't have to be very precise. The system of differential equations devised above involves several components that are decaying at widely differing rates (infectious vs. recovery rate). It means that this system is potentially stiff as the numerical solution has its step size limited more by the stability of the numerical technique than by its accuracy. It is hard to identify whether the system is stiff or not but it is important to account for this in the model to get reasonable results.

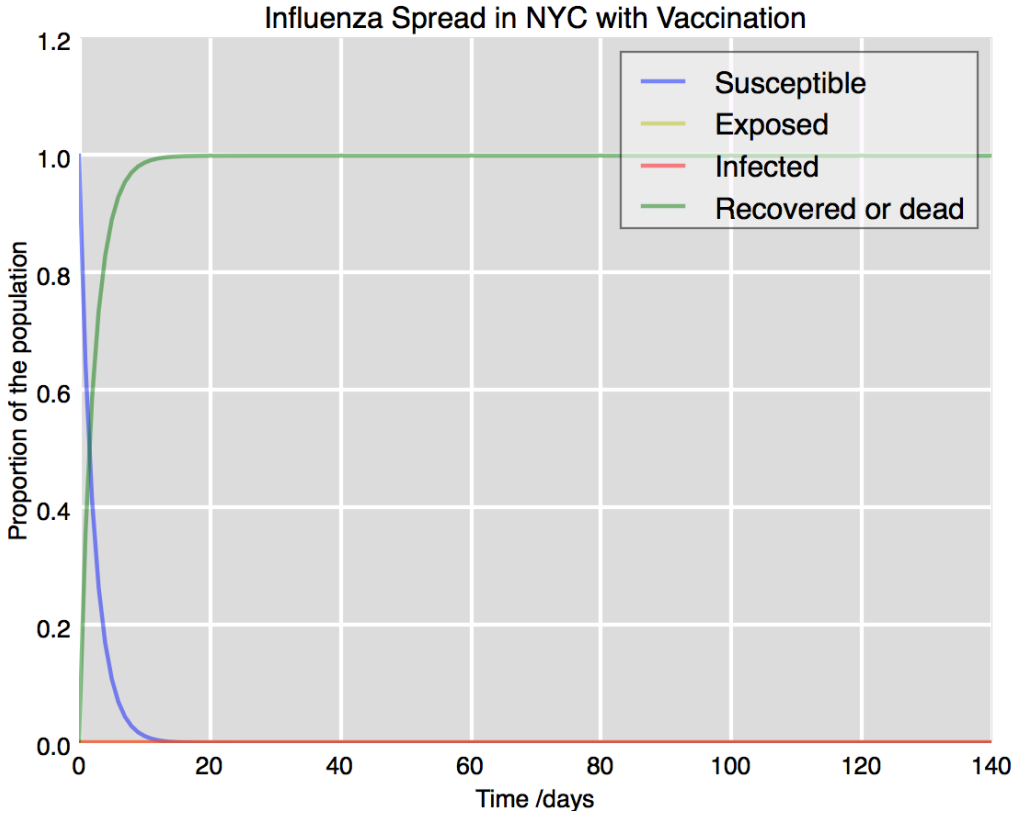
To do so, I chose a python built-in function "odeint" from the scipy package which is very robust and deals with both stiff and non-stiff systems successfully. To solve stiff equations, this function uses backward differentiation formula (BDF) which belongs to a family of multistep methods. Single-step methods (such as the Euler's method that we've discussed in class) refer to only one previous point and its derivative to calculate the current value, that's why they can be far off from the reality. In contrast, BDF approximates the derivative of a function using information from already computed steps, by finding a linear combination of the previous points and their derivative values. I chose this approach as it dramatically increases the accuracy of the approximation compared to the methods we studied in class, has a good running time even on average computers, and can be easily implemented in python. *(See the code for the reference).*

Results

My simulation produced two graphs presented below. The first one represents the natural spread of influenza in a community as described by the first set of differential equations. The second model assumes a vaccination rate of 44% (the percentage of vaccinated people as reported by the NYC Health).

As we can see from the graph, without vaccination, the rate of infected people will be around 20% and will highly depend on the number of contacts between the infected and susceptible individuals as the other parameters are constant and defined by the nature of the influenza virus. In case we add the vaccination rate of 44%, the probability of getting infected decreases dramatically which proves the effectiveness of such intervention:





Figures 2 and 3 illustrate the flu propagation in the presence and without vaccination

Limitations of SEIR Model

It is hard to deny that this model is over-simplified. For instance, it assumes that individual characteristics of immunity, susceptibility, and ability to recover are the same for all members of the population.

It also doesn't account for the spatial distribution and mobility of the individuals and assumes that the probability of getting infected is the same for everyone. It is not the case in real life as some people make more contacts to other individuals so their probability of getting infected is higher. At the same time, infected individuals have reduced mobility and are often isolated from the community.

As a result, this model is deterministic and will lead to the same final distribution of the compartments given specific parameters.

Despite these limitations, SEIR model grasps the general trend and helps people learn more about the qualitative dynamics of a disease on the population level. It can be used to inform public health interventions such as vaccination campaigns and test different hypotheses about the effectiveness of such campaigns.

Dynamic Model

The second model will address the limitations of the first one by representing the population using a graph. Similar to the *pi* approximation model that uses Monte Carlo method, I generated random points on the plane to represent the distribution of individuals. After that, I created the edges between the points based on their proximity to each other as it is obvious that individuals from the upper right corner won't be able to get infected in case the source of infection is on the opposite side of the graph. The number of individuals as well as the rate of interaction can be changed by the user dependent on the population.

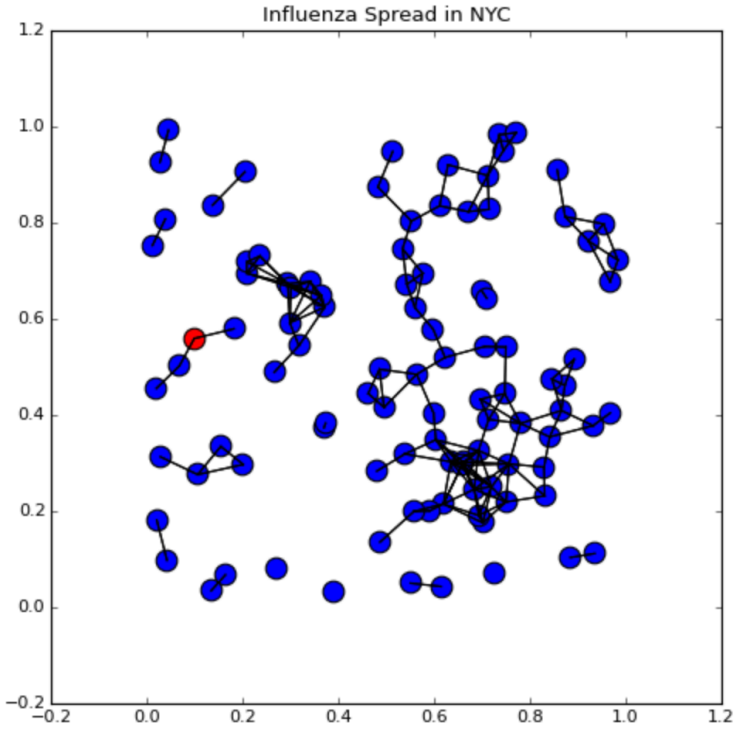
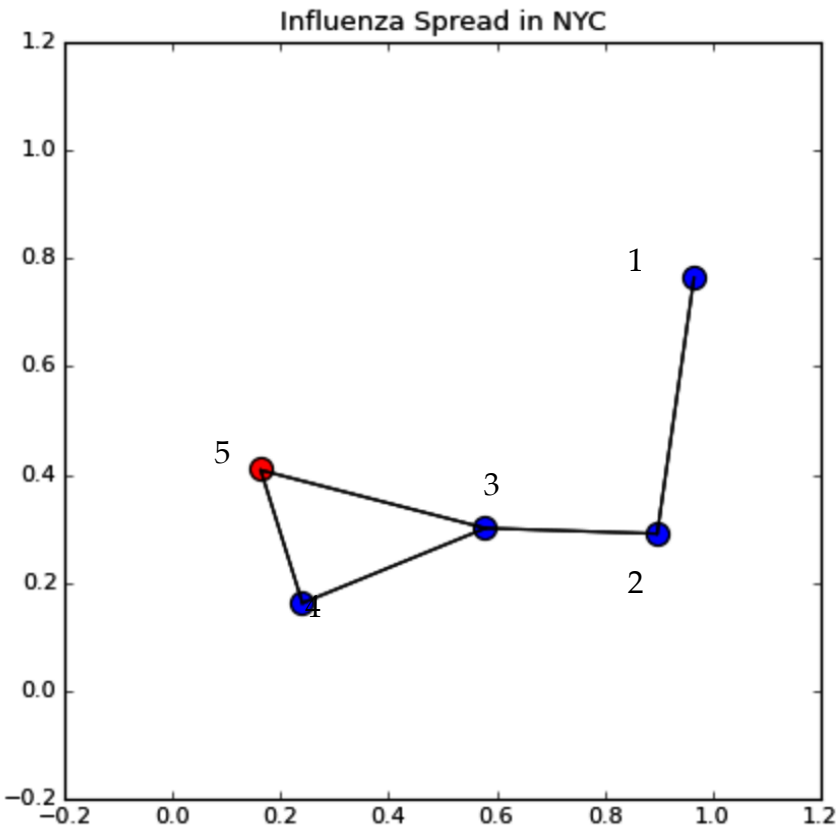


Figure 3. The population represented by a graph

Now, I will explain the underlaying logic of the model. The model is not deterministic as it uses stochastic modelling approach - Markov chains. The computer calculates the probability of transitioning to a different state for every single individual and moves it to a different state using the "random walk" principle. To explain how it works, I will create a transition matrix for just five data points.

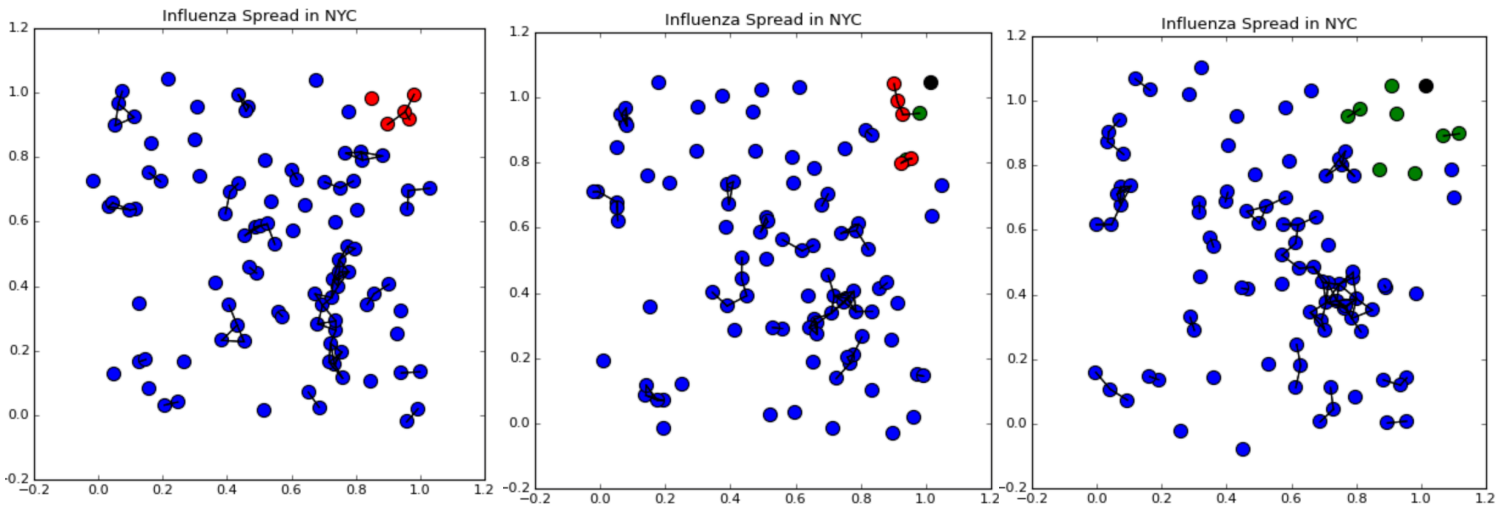


From the graph on the left, we can see that this simplified population consists of just five dots. Not all dots are connected to each other, and only two dots are connected to the infected individual who is colored in red. Let's look at the probability of each dot to become either blue or red. The simulation assumes that each susceptible dot changes after it makes contact to another dot, so it either stays blue or gets infected and changes its color to red. The first and the second dots don't have contact to any infected individuals, that's why they have a 100% probability to stay blue. The third individual is connected to two susceptible and one infected individual, their probability of getting infected in the next iteration is 1/3. Individual number 4 is connected to two other individuals which means there is 50% chance they will be infected. As for the infected dot, it stays red for the infectious period (two weeks or 14 days) and changes its color to green which corresponds to "recovered". During this two weeks, they have a 0.0014% chance of dying. To demonstrate this, I introduced the forth group of dead people. The red dot changes its color to black, doesn't make contact to any other dots and stops moving, which I'll talk more about in the next section.

	1	2	3	4	5
1	0	1	0	0	0
2	0.5	0	0.5	0	0
3	0	1/3	0	1/3	1/3
4	0	0	0.5	0	0.5
5	0	0	0	0	1

Table 2. A transition matrix from one state to another. The rows and columns represent the individuals from the population above. All the rows in the table sum up to one. For example, it shows that the probability of becoming red, which corresponds to contacting the fifth dot, is 1/3 for the third dot. In the simulation, computer automatically calculates such matrices which have hundreds of individuals.

To address the second limitation of the deterministic model, which is a lack of mobility, I added movement to the stochastic simulation so the dots can meet other individuals and potentially get infected or infect someone. To explore this interactive simulation, please watch the video from the zip-file. In the simulation, you can see how the first focus of infection appears in the bottom left corner and starts to spread across the population until all the infected people recover. You can also run the code from the appendix ("Dynamic Model"). Here are some stages of different run of the simulation:



These figures represent the main stages of the simulation: the outbreak of the disease, the death of one individual + the first recoveries, and the final stage of total recovery. Note that only some individuals get infected which is realistic as we can't expect the entire city of NY to become sick.

This model gives us a much better understanding of the disease propagation compared to the first model. Given that the epidemic has started in a specific district of the NYC, it is logical that it first spreads in this district and then, in some cases, propagates through the entire population. My computer is not powerful enough to run this simulation on more than a few hundreds dots, but, potentially, it can be used to model the behavior of thousands of individuals. The model can also be adjusted to study other diseases by simply changing the infectious and death rate, as well as the contagiousness by adjusting respective parameters. The simulation can also be stopped at any specific point of time to see the distributions of different groups.

Limitations and Possible Improvements

This model addresses all the major limitations of the compartmental model. It assigns different probabilities of getting infected to different individuals, accounts for the spatial distribution and mobility, and is not deterministic. However, it is still a simplified version of the reality as it is impossible to account for all the factors. This dynamic model can be further improved by adding some other groups, such as exposed individuals, or the vaccination rate. These two factors are advantages of the first model which is also much easier to implement. Potentially, it is possible to divide individuals into households where the interaction rate is much higher or decrease the mobility of sick individuals by introducing quarantine zones.

Conclusion

To sum up, a combination of these two methods that use differential equations and stochastic models help us understand how influenza spreads in NYC and can be used to model multiple scenarios by putting the focus of infection into a specific district and see what happens using the dynamic model. The compartmental model can be used to inform public health intervention, for example, calculate the minimum vaccination rate required to prevent an epidemic.

Technical Appendix:

Compartmental Model:

```
#Installing all the necessary packages
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import pylab as pl
%matplotlib inline

# Total population of NYC
N = 8538000

# Initial number of exposed, infected and recovered individuals.
E0, I0, R0 = 0, 1, 0
# Everyone else, S0, is susceptible to infection.
S0 = N - E0 - I0 - R0
# Contact rate (alpha), infectious rate (beta), recovery rate
#(gamma), and vaccination rate (sigma).
alpha, beta, gamma, sigma = 0.75, 0.25,1/5., 0
# A grid of time points (in days)
t = np.linspace(0, 140, 140)

# The SEIR model differential equations.
def deriv(y, t, N, alpha, beta, gamma, sigma):
    S, E, I, R = y
    dSdt = - alpha * S * I/N - sigma * S
    dEdt = alpha * S * I/N - beta * E
    dIdt = beta * E - gamma * I
    dRdt = gamma * I + sigma * S
    return dSdt, dEdt, dIdt, dRdt

# Initial conditions vector
y0 = S0, E0, I0, R0

# Solving differential equations numerically
ret = odeint(deriv, y0, t, args=(N, alpha, beta, gamma, sigma))
S, E, I, R = ret.T

# Plotting the graph
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111, axis_bgcolor='#dddddd', axisbelow=True)
ax.plot(t, S/N, 'b', alpha=0.5, lw=2, label='Susceptible')
ax.plot(t, E/N, 'y', alpha=0.5, lw=2, label='Exposed')
ax.plot(t, I/N, 'r', alpha=0.5, lw=2, label='Infected')
ax.plot(t, R/N, 'g', alpha=0.5, lw=2, label='Recovered or dead')
ax.set_xlabel('Time /days')
ax.set_ylabel('Proportion of the population')
ax.set_ylim(0,1.2)
```

```
ax.grid(b=True, which='major', c='w', lw=2, ls='-')
pl.title('Natural Influenza Spread in NYC')
legend = ax.legend()
legend.get_frame().set_alpha(0.5)
plt.show()
```

Dynamic Model:

```
import random
import math
from scipy import stats
import numpy as np
import time
import pylab as pl
import matplotlib.pyplot as plt
%matplotlib notebook

# Draws the dots and edges
def draw(ax, fig, coords, edges, group):
    ax.cla()
    color = {0: "blue", 1: "red", 2: "green", 3: "black"}
    ax.set_xlim(-0.2, 1.2)
    ax.set_ylim(-0.2, 1.2)
    for v in range(len(edges)):
        for u in edges[v]:
            x = []
            y = []
            x.append(coords[v][0])
            x.append(coords[u][0])
            y.append(coords[v][1])
            y.append(coords[u][1])
            ax.plot(x,y, "black")
    for i in range(len(coords)):
        ax.scatter(coords[i][0], coords[i][1], c =
color[group[i]], s = 100)
    pl.title('Influenza Spread in NYC')
    fig.canvas.draw()

# Creates dots
num = 100
coords = [(random.random(),random.random()) for i in range(num)]
group = [0 for i in range(num)]

# Creates edges
def make_edges(coords, group):
    edges = [[] for i in range(num)]
    for i in range(len(coords)):
        for j in range(len(coords)):
            if i == j:
                continue
```

```
dist = math.sqrt((coords[i][0]-coords[j][0])**2+(coords[i][1]-
coords[j][1])**2)
            if dist < 0.07 and group[i] != 3 and group[j] != 3:
                edges[i].append(j)
    return edges

# Creates the initial plot
fig, ax = plt.subplots(figsize=(6, 6))
illness_start = [1e9 for i in range(num)]
group[0] = 1
illness_start[0] = 0

# Enabling the movement
for day in range(100):
    for i in range(num):
        if group[i] == 3:
            continue
        x, y = coords[i]
        shift_x, shift_y = ((random.random() - 0.5) / 20,
(random.random() - 0.5) / 20)
        coords[i] = (x + shift_x, y + shift_y)

# Recovery or death phases
    for i in range(num):
        if group[i] == 1 and day - illness_start[i] >= 14:
            group[i] = 2
        if group[i] == 1:
            death = random.random()
            if death <= 0.000014:
                group[i] = 3
    edges = make_edges(coords, group)

# Updating the groups based on their interaction
    new_group = [i for i in group]
    for v in range(len(edges)):
        infectious_neighbors = 0
        for u in edges[v]:
            if group[u] == 1:
                infectious_neighbors += 1
        if group[v] == 0 and infectious_neighbors >= 1:
            prob = float(infectious_neighbors)/len(edges[v])
            if random.random() <= prob:
                new_group[v] = 1
                illness_start[v] = day
        group = new_group

    draw(ax, fig, coords, edges, group)
    time.sleep(0.2)
    if 1 not in set(group):
        break
```

Bibliography:

1. FastStats. (2017). *Cdc.gov.*
2. Gear, B. (2007). Backward differentiation formulas. *Scholarpedia*, *2*(8), 3162.
3. Influenza (Seasonal). (2017). *World Health Organization.*
4. Petzold, L. R., & Hindmarsh, A. C. (1997). Lsoda. *Computing and Mathematics Research Division, I-316 Lawrence Livermore National Laboratory, Livermore, CA, 94550.*