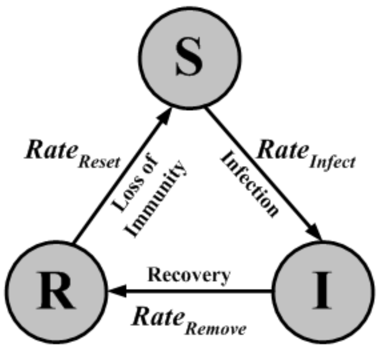
A Brief Look at the SIRS Epidemic Model (Math 450 Honors Option)

Simranjeet Singh

Herein, we consider a variant (the SIRS model) of the original compartmental model proposed by Kermack and McKendrick (1927). After analyzing the classical ODE dynamics associated with the model, we consider the spatial model implemented by Ballegooijen and Boerlijst (2004), and analyze their results in the context of classical theory.



Let S, I, and R denote the members of the population susceptible to infection, already infected, and the resistant to infection, respectively. Further, let β denote the rate of infection (that is, it is the rate at which an individual moves from the susceptible compartment to the infected compartment), γ be the rate at which an individual moves from the infected compartment to the resistant compartment, and ρ be the rate at which an individual moves from the resistant compartment back into the susceptible population. Then we can describe the classical model of the associated viral dynamics by

$$\begin{cases} \frac{\partial S}{\partial t} = \rho R - \beta SI \\ \frac{\partial I}{\partial t} = \beta SI - \gamma I \\ \frac{\partial R}{\partial t} = \gamma I - \rho R \end{cases}$$

where βSI is a mass action term (there is an assumed homogeneous mixture within the population) . From the equations above, we can derive a key quantity called the basic reproduction number R_0, which denotes the average number of individuals an infected person will go on to infect in a completely susceptible population. Logically, it makes sense that a virus will proceed to become an epidemic if $R_0 > 1$, else it will die out. In this model, we have that $R_0 = \frac{\beta}{\gamma}$. Using the *scipy* library we can visually observe the system of nonlinear ODEs above.

Fixing $\beta = 0.25$, $\gamma = 0.2$, and $\rho = 0.1$, we observe plots with various initial conditions.

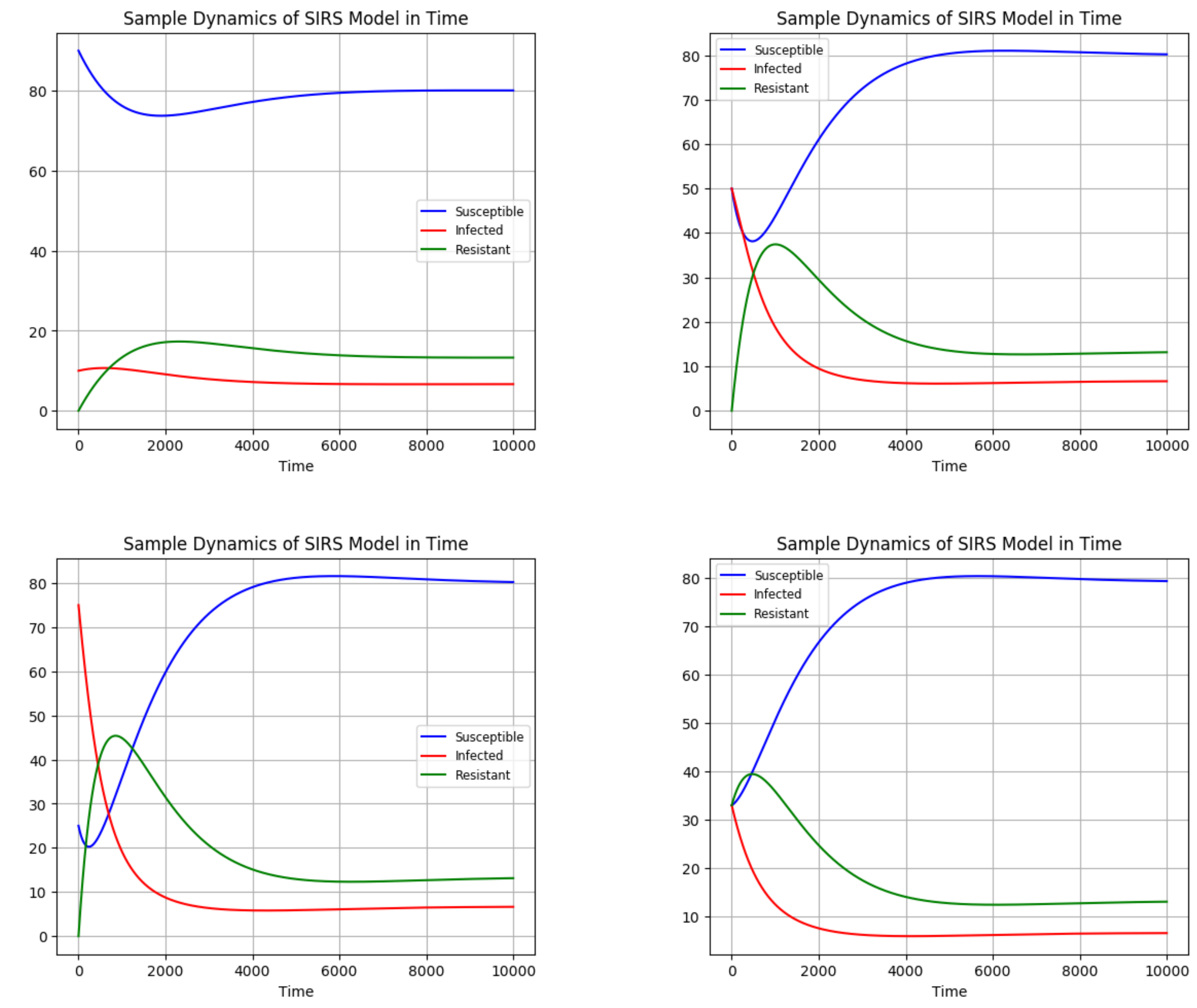


Figure 1: In each of these instances we let the total population N equal to 100. In the upper left, we use the initial condition $S(0) = 90$ and $I(0) = 10$, while in the upper right, we have $S(0) = I(0) = 50$. In the lower left plot, we use $S(0) = 25$ and $I(0) = 75$, while the lower right plot has $S(0) = I(0) = R(0) = 33.33$. It is clear that for any non-critical points, the overall system asymptotically approaches an equilibrium state, where $\lim_{t\to\infty}\{S(t), I(t), R(t)\} \approx \{80, 8, 12\}$. In the above example, we have $R_0 = 1.25$.

Classical theory would suggest that a virus would mutate in order to maximize R_0. However, the stochastic spatial model implemented by Ballegooijen and Boerlijst (B&B) does not follow this hypothesis. Instead, we see that although β mutates to increase over time, the mean infection period τ_I (which equals $1/\gamma$ in the classical model) decreases. We now consider their spatial model, and analyze our attempt to replicate their results.

The B&B model begins by considering an $n \times n$ square lattice structure, where each cell can either be susceptible, infected, or resistant. An infected cell can infect its neighboring cells at the infection rate β, where we use the von Neumann neighborhood to consider surrounding cells. Once a cell is infected, it remains so for a fixed period of time τ_I, before becoming resistant for a fixed period τ_R (scaled to one in the model). After its resistance period is completed, the cell becomes susceptible to infection once again. The update rules at each time step Δt are defined as follows:

1. The probability of a susceptible cell becoming infected by one of its eight neighbors is given by $p_{\text{inf}} = 1 - e^{-i\beta\Delta t}$ where i is the number of infected neighbors, and β is the sum of all of the infection rates belonging to the infected neighbors.

2. If a cell does become infected, it inherits the *genotype* (the β and τ_I) of the infecting cell. We assume this is done by taking a weighted random sample of the infection rates (Prob(Inherit β_j) $= \beta_j / \sum_i \beta_i$).

3. An infected cell will remain infected for a fixed period τ_I before becoming resistant. While infected, a cell can mutate its genotype with probability μ, which then determines the changes in genotype in small fixed steps $(\pm\Delta\beta, \pm\Delta\tau_I)$.

4. A resistant cell will remain so for a fixed time τ_R, after which it becomes susceptible once more.

We use the rules above, and apply the following parameter values: $n = 75$, $\Delta t = 0.01$, (initial) $\beta = 4.15$, (initial) $\tau_I = 0.72$, $\mu = 0.01$, and $\Delta\beta = \Delta\tau_I = 0.01$. With this, we were able to recreate the following wave-like behavior:

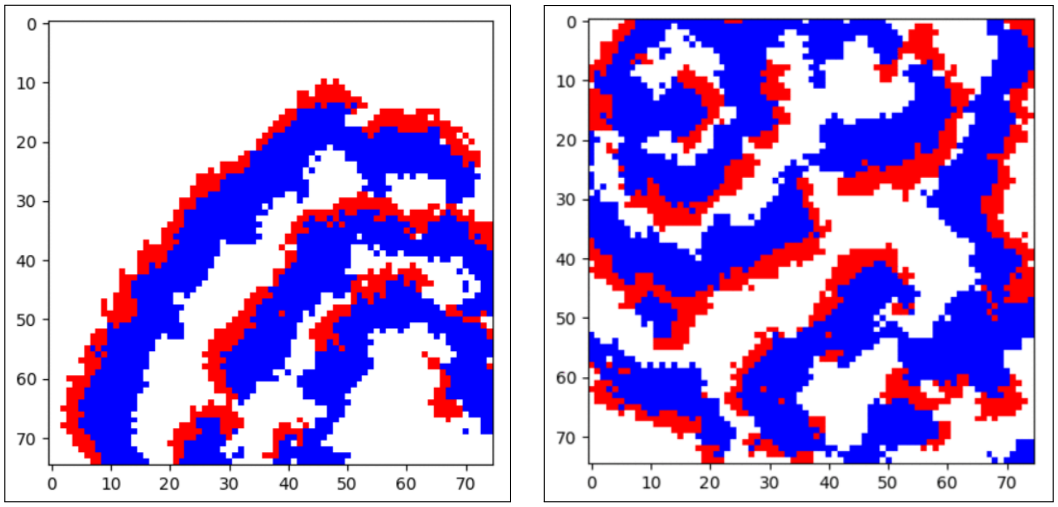


Figure 2: It is clear that we observe wave-like behavior, where the white-colored cells are susceptible, the red-colored cells indicate the infection front, and the blue-colored cells indicate the resistant front.

We can also see how the mean infection rate, infection period, and basic reproduction number evolve over time.

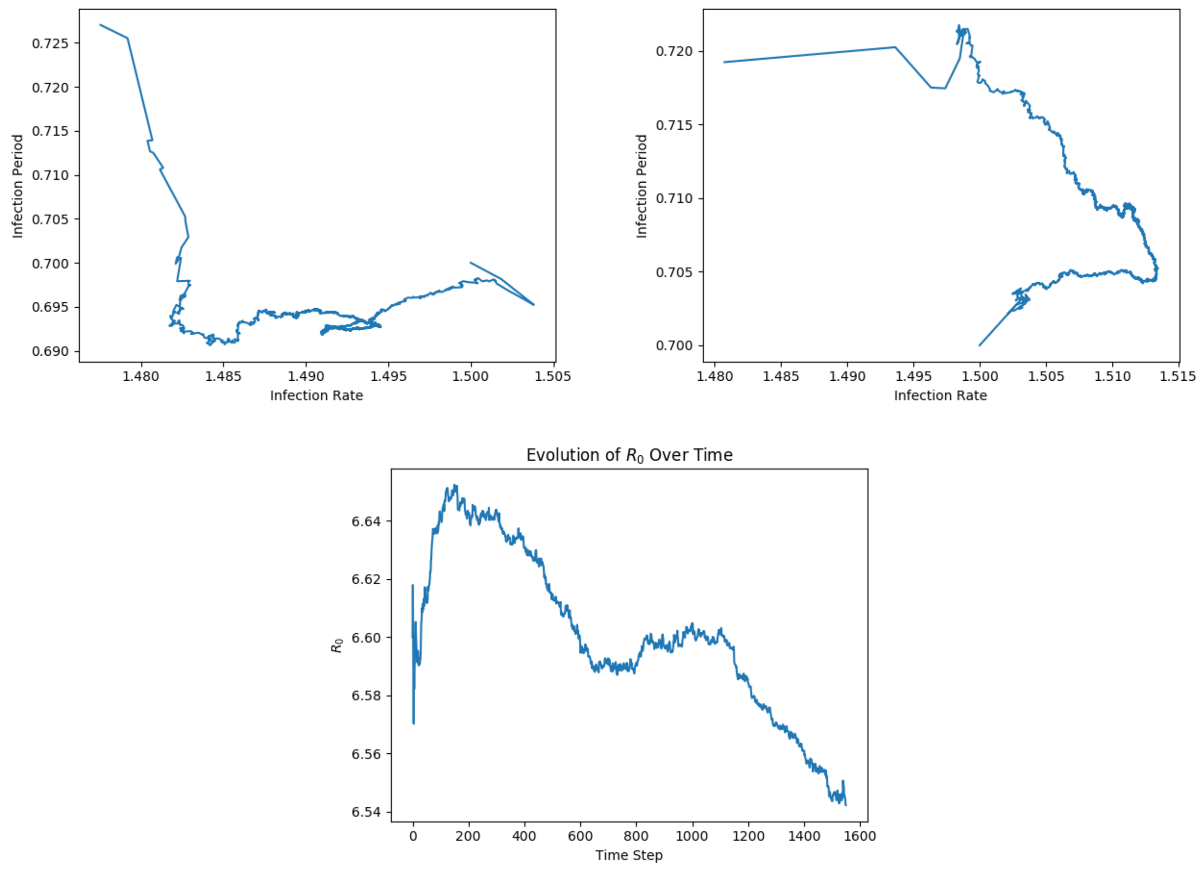


Figure 3: The overall trend in both sample plots suggests that β increases and τ_I decreases over time ($R_0 = 8\beta\tau_I$ decreases), which agrees with the findings of the B&B model, but disagrees with classical theory.

Possible reasons why the B&B model may contradict classical theory is partially dependent on the fact that classical theory assumes a homogeneous continuous mixture, while the spatial model lacks such a mixing assumption. Further, the dimensionality of the lattice may possibly change the dynamics we observe. Possible extensions that could potentially impact the results would be having a time-varying mutation rate, as well as a time-dependent resistance period.

References

[1] Ballegooijen, W. M. Van, and M. C. Boerlijst. "Emergent Trade-offs and Selection for Outbreak Frequency in Spatial Epidemics." Proceedings of the National Academy of Sciences 101.52 (2004): 18246-8250. Web.

[2] Brauer, F., van den Driessche, P., Wu, J.: Mathematical Epidemiology. Springer, Heidelberg (2008)

Sample Classical Dynamics of SIRS Model

```
from scipy.integrate import odeint
from numpy import arange
import matplotlib.pyplot as plt

def SIRS(state,t):

  # Initialization of States
  S = state[0]
  I = state[1]
  R = state[2]

  # Appropriate Constants
  beta = 0.25 # Rate of infection; needs to be divided by N in the system of ODEs
  gamma = 0.2 # Proportion leaving infected to become resistant
  rho = 0.1 # Proportion leaving resistant to become susceptible.
  N = S + I + R
  # compute state derivatives
  dS = (rho*R) - ((beta/N)*S*I)
  dI = ((beta/N)*S*I) - (gamma*I)
  dR = (gamma*I) - (rho*R)

  # return the state derivatives
  return [dS, dI, dR]

state0 = [33, 33, 33]
t = arange(0.0, 100, 0.01)
state = odeint(SIRS, state0, t)

# Plots
susceptible = state[:, 0]
infected = state[:, 1]
resistant = state[:, 2]

fig = plt.figure(figsize=(15,5))
fig.subplots_adjust(wspace = 0.5, hspace = 0.3)
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)
ax1.plot(susceptible, 'b-', label = "Susceptible")
ax1.plot(infected, 'r-', label = "Infected")
ax1.plot(resistant, 'g-', label = "Resistant")
ax1.set_title("Sample Dynamics of SIRS Model in Time")
ax1.set_xlabel("Time")
ax1.grid()
ax1.legend(loc = 'best', fontsize = 'small')
ax2.plot(susceptible, infected, color = "purple", label = 'S-I Phase')
ax2.plot(susceptible, resistant, color = "cyan", label = 'S-R Phase')
ax2.plot(infected, resistant, color = "yellow", label = 'I-R Phase')
ax2.set_title("SIRS Sample Phase Space")
ax2.legend(loc = 'best', fontsize = 'small')
ax2.grid()

extent = ax1.get_window_extent().transformed(fig.dpi_scale_trans.inverted())
fig.savefig('timedynamics5.png', bbox_inches=extent.expanded(1.3, 1.3))
extent = ax2.get_window_extent().transformed(fig.dpi_scale_trans.inverted())
fig.savefig('phase5.png', bbox_inches=extent.expanded(1.2, 1.2))

plt.show()
```

Model Implementation:

```python
import matplotlib.pyplot as plt
from matplotlib import colors
import numpy as np

nn = 75 # Grid size
dt = 0.01 # time incrementation
beta0 = 2.75 # initial infection rate
tauI0 = 0.3 # initial infection period
tauR0 = 1.0 # initial (constant) resistance period
mu = 0.01 # infection rate / probability
dbeta = 0.01
dtauI = 0.01 # change in infection rate/period upon mutation
t = 0.0 # start time
T = 600. # end time; increase to observe convergence

state = np.zeros((nn, nn))# Initializie state with infectious cells
state[2, 2] = 1
state[73, 73] = 1

t_infect = np.zeros((nn, nn)) # How long the cell has been infected
t_resist = np.zeros((nn, nn)) # How long the cell has been resistant
beta_geno = np.zeros((nn, nn)) # Genotype of infection rate
tauI_geno = np.zeros((nn, nn)) # Genotype of infection period

# We first give all infected cells the same genotype
for rr in range(0, nn):
    for cc in range(0, nn):
        if state[rr, cc] == 1:
            beta_geno[rr, cc] = beta0
            tauI_geno[rr, cc] = tauI0

# The following function finds all the neighbors of a cell for a given matrix
# and given indices (x, y)
def nbrs(mat, x, y ):
    results = []
    for dx in [-1,0,1]:
        for dy in [-1,0,1]:
            newx = x+dx
            newy = y+dy
            if (dx == 0 and dy == 0):
                continue
            if (newx >= 0 and newx < len(mat) and newy >= 0 and newy < len(mat)):
                results.append( mat[newx, newy] )
    return results

# Make a color map for the state transitions
# White -> Susceptible
# Red -> Infected
# Blue -> Resistant
cmap = colors.ListedColormap(['white', 'red', 'blue'])
bounds=[0, 1, 2, 3]
norm = colors.BoundaryNorm(bounds, cmap.N)
img = plt.imshow(state, interpolation='nearest', origin='upper',
                  cmap=cmap, norm=norm)
plt.savefig('tmp_0001.png')
plt.close()
```

```
pic_counter = 1
fileName = 'tmp_'

infection_rates = [beta0] # List of average infection rates
infection_times = [tauI0] # List of average infection periods

# Evolution
while t <= T:

    t += dt
    pic_counter += 1
    newstate = np.copy(state) # Temporary newstate matrix to allow update at each time
        step

    for rr in range(0, nn):
        for cc in range(0, nn):

            # 1) Check if cell is susceptible
            if state[rr, cc] == 0:
                i = nbrs(state, rr, cc).count(1) # If susceptible, observe nbrs

                if i >= 1: # If one or more nbrs is infected
                    beta = sum(nbrs(beta_geno, rr, cc)) # Add up infection rate of all nbrs
                    p_inf = 1 - np.exp(-i * beta * dt) # Calculate "exponential"
                        probability of infection
                    dec_inf = np.random.binomial(1, p_inf) # Determine if cell will be
                        infected

                    if dec_inf == 1:
                    # If infected, determine genotype of newly infected cell by weighing
                        the
                    # infection rates and infection periods.
                        newstate[rr, cc] = 1
                        b = nbrs(beta_geno, rr, cc)
                        tau = nbrs(tauI_geno, rr, cc)
                        wts = b / sum(b)

                        choice = np.random.choice(b, 1, p = wts)
                        beta_geno[rr, cc] = choice
                        tauI_geno[rr, cc] = tau[b.index(choice)]

            # 2) Check if cell is infected
            elif state[rr, cc] == 1:

                if t_infect[rr, cc] <= tauI_geno[rr, cc]: # If infection period is not over
                    t_infect[rr, cc] += dt # Update how long cell has been infected
                    mut_flip1 = np.random.binomial(1, mu) # Flip to determine if mutation
                        occurs

                    if mut_flip1 == 1:
                    # If mutation occurs, flip to determine direction of change
                        mut_flip2 = np.random.binomial(1, p = 0.50, size = (1, 2))

                        if (mut_flip2[:, 0] == 0) and (mut_flip2[:, 1] == 0):
                            # (rate goes down, period goes down)
                            beta_geno[rr, cc] -= dbeta
                            tauI_geno[rr, cc] -= dtauI

                        elif (mut_flip2[:, 0] == 0) and (mut_flip2[:, 1] == 1):
```

```
                        # (rate goes down, period goes up)
                        beta_geno[rr, cc] -= dbeta
                        tauI_geno[rr, cc] += dtauI

                    elif (mut_flip2[:, 0] == 1) and (mut_flip2[:, 1] == 0) :
                        # (rate goes up, period goes down)
                        beta_geno[rr, cc] += dbeta
                        tauI_geno[rr, cc] -= dtauI

                    elif (mut_flip2[:, 0] == 1) and (mut_flip2[:, 1] == 1):
                        # (rate goes up, period goes up)
                        beta_geno[rr, cc] += dbeta
                        tauI_geno[rr, cc] += dtauI

                    infection_rates.append(np.true_divide(beta_geno.sum(), (beta_geno
                        != 0).sum()))
                    infection_times.append(np.true_divide(tauI_geno.sum(), (tauI_geno
                        != 0).sum()))

            elif t_infect[rr, cc] > tauI_geno[rr, cc]: # If infection period is over
                newstate[rr, cc] = 2 # Update state to resistant
                t_infect[rr, cc] = 0 # Reset infected time
                # Reset infection rate and infection period genotype
                beta_geno[rr, cc] = 0
                tauI_geno[rr, cc] = 0

        # 3) Check if cell is resistant
        elif state[rr, cc] == 2:

            if t_resist[rr, cc] <= tauR0: # If resistance period is not over
                t_resist[rr, cc] += dt

            else: # If resistance period is over
                newstate[rr, cc] = 0 # Make cell susceptible again
                t_resist[rr, cc] = 0

    state = np.copy(newstate) # State update


    # Save each transition plot; leave as comment when analyzing mutation behavior alone.
    # img = plt.imshow(state, interpolation='nearest', origin='upper',
    #                     cmap=cmap, norm=norm)
    #plt.savefig(fileName + '%04d.png' %pic_counter)

#plt.plot(infection_rates, infection_times)
#plt.xlabel('Infection Rate')
#plt.ylabel('Infection Period')
R = [8*a*b for a,b in zip(infection_rates, infection_times)]
plt.plot(R)
plt.show()
```