Assignment7

Richard McCormick

March 10th, 2022

In this assignment, we'll explore the effects of adding both types of stochasticity to the XYZ model without demography. In the deterministic case of the model, we expect that, when $R_0 > 1$, there will be an epidemic curve, and then the infection will eventually burn-out, $Y(\infty) = 0$.

Environmental stochasticity

Environmental stochasticity means that there are unpredictable environmental attributes that randomly change the processes in our model over time. If you remember from last assignment, we used a time-dependent notation for transmission rate, $\beta(t)$, such that transmission rate was a function of time. In the case of environmental stochasticity, we assume that the transmission rate is changing **randomly** over time.

When adding environmental stochasticity to our model, however, we need to think about how the effect of stochasticity changes with the number of individuals in the population. We will assume that the transmission process becomes less random (i.e., more deterministic) when we have lots of infected individuals. Specifically, this assumption means that, with fewer infected individuals in the population at a given time, we are less confident about how the dynamics will unfold.

Our model therefore becomes:

$$\begin{aligned}
\frac{dX}{dt} &= \frac{\beta(t)}{N} XY \\
\frac{dY}{dt} &= \frac{\beta(t)}{N} XY - \gamma Y \\
\frac{dZ}{dt} &= \gamma Y
\end{aligned}$$

Then, $\beta(t) = \left| \bar{\beta} \left(1 + \frac{\xi(t)}{\sqrt{Y(t)}} \right) \right|$, where $\bar{\beta}$ is the average transmission rate, and $\xi(t)$ is a random variate that varies per time step and is drawn from a normal distribution with mean zero and some assigned value of standard deviation. A larger standard deviation would increase the effect of stochasticity on the transmission process. Essentially, this is adding random noise to the value of the transmission rate. $Y(t)$ is the number of infectious individuals in the population at any given time. Overall, the value of $\beta(t)$ changes randomly at each time step, but the effect of randomness decreases with larger numbers of infectious individuals (i.e., the $\beta(t)$ is closer to the average $\bar{\beta}$). If you need a refresher as to why we are dividing transmission rate by the total population size N, look at the Week 1 lecture notes.

Task 1 (20 points)

Your goal is to implement **environmental stochasticity**, using code very similar to the code from the previous assignment (and assignment 4).

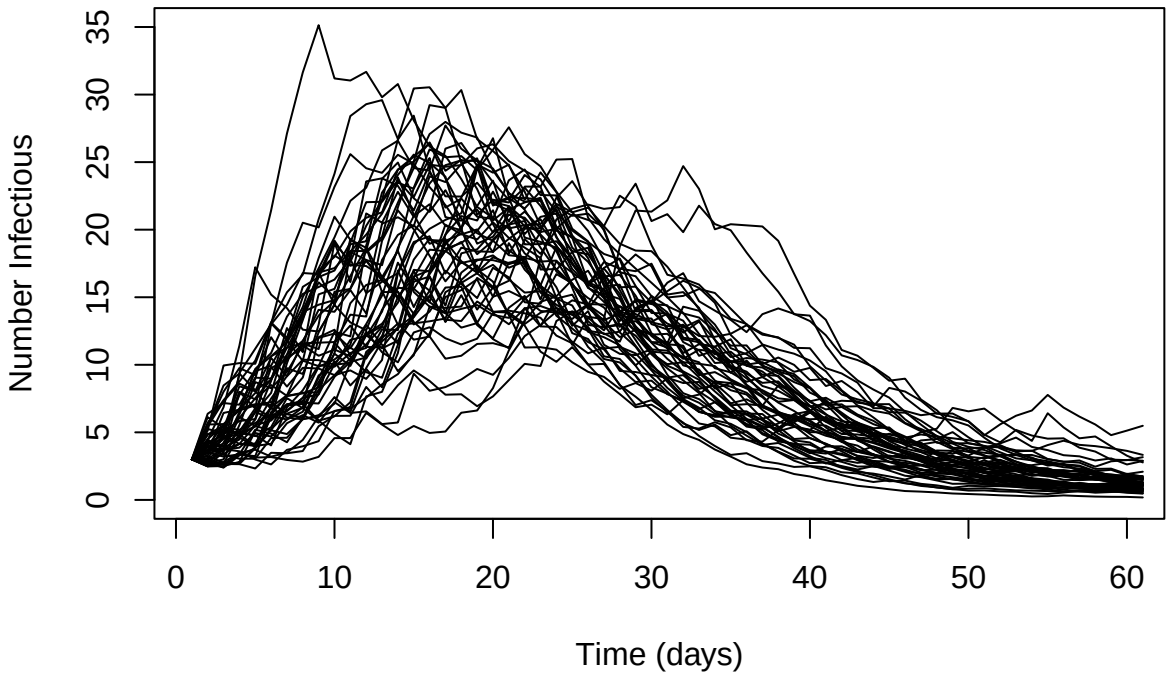
Use the code below. Create a for-loop to:

- Plot 50 realizations of your stochastic model on one figure.

```
#-------------------------------------------------------
#-------------------------------------------------------
library(deSolve)
library(tidyverse)

#-------------------------------------------------------
#-------------------------------------------------------
# Function to solve the ODE system

stoch_XYZ_ode = function(t, y, params){
  with(as.list(c(y, params)), {

    # storage
    ## We need to have a vector of size equal to the number of ODEs in the system
    dydt = rep(0, 3)

    # equations:
    dydt[1] = -abs(beta_bar * (1 + noise / sqrt(abs(y[2]))))) * y[1] * y[2] / pop_N
    dydt[2] = abs(beta_bar * (1 + noise / sqrt(abs(y[2])))) / pop_N * y[1] * y[2] - gamma * y[2]
    dydt[3] = gamma * y[2]

    return(list(dydt))

  })
}

#-------------------------------------------------------
#-------------------------------------------------------

# Define a function that will solve the ODE over one day
# (i.e. one "step") at a time

solve_ODE_1step = function(inits, params){

  # Run your ODE with the inputs to the function:
  out_temp = ode(y = inits,
                 times = time_ode,
                 func = stoch_XYZ_ode,
                 method="ode45", # Runge-Kutta 4-5 method
                 parms = params)

  # Object 'out' is a matrix
  # Specify the column names
  # Then convert to data.frame object
  colnames(out_temp) = c("time", "X", "Y", "Z")
  out_temp = data.frame(out_temp)
```

```
  # Return the data frame
  return(out_temp)

}

#-------------------------------------------------------
#-------------------------------------------------------

# Now we can create a loop to solve the ODE over many days
# Only one epidemic:
t_min = 1
t_max = 60

# population size
pop_N = 150

# static parameter values
gamma = 1 / 5
beta_bar = 0.35

# Plot the number infectious over time.
plot(NA,NA,
     xlim = c(t_min, t_max),
     ylim = c(0, 35),
     xlab = "Time (days)",
     ylab = "Number Infectious")

for (i in 1:50)
{
  # Time to solve ODE, one day at a time:
time_ode = seq(0, 1, by = 1)

# Create a loop:
for(i in t_min:t_max){

  if(i == t_min){ # If this is the first time looping...
    # Set your initial values manually:
    Y0 = 3
    X0 = pop_N - Y0
    Z0 = 0
  }else{ # otherwise...
    # What am I doing here??
    X0 = df_temp[nrow(df_temp), 2]
    Y0 = df_temp[nrow(df_temp), 3]
    Y0 = ifelse(Y0 < 0, 0, Y0) # Ensure positive values
    Z0 = df_temp[nrow(df_temp), 4]
  }

  inits_temp = c(X0, Y0, Z0)

  # Draw a value of noise that randomly alters transmission:
  noise_temp = rnorm(1, 0, 2)
```

```
  # Store your params:
  params_temp = c(beta_bar = beta_bar,
                  noise = noise_temp,
                  gamma = gamma,
                  pop_N = pop_N)

  # Solve over one day
  df_temp = solve_ODE_1step(inits = inits_temp,
                                params = params_temp)

  # What is the rest of this doing??
  time_ode_fix = seq(i, i+1, by = 1)
  df_temp$time = time_ode_fix

  if(i == t_min){
    out = df_temp
  }else{
    out = rbind(out, df_temp)
  }

}
lines(out$Y ~ out$time)
}
```



Demographic Stochasticity

The better way to include stochasticity in these models is to take an event-driven approach, which is referred to as **demographic stochasticity**. In lecture, we wrote pseudo-code for Gillespie's Direct Algorithm, which solves the probabilistic disease model. The code below implements this algorithm for one realization of the stochastic XYZ model. Carefully read the code I provide, and make sure you understand what is going on. It may help to compare this code to the psuedo-code from lecture.

Task 2 (20 points)

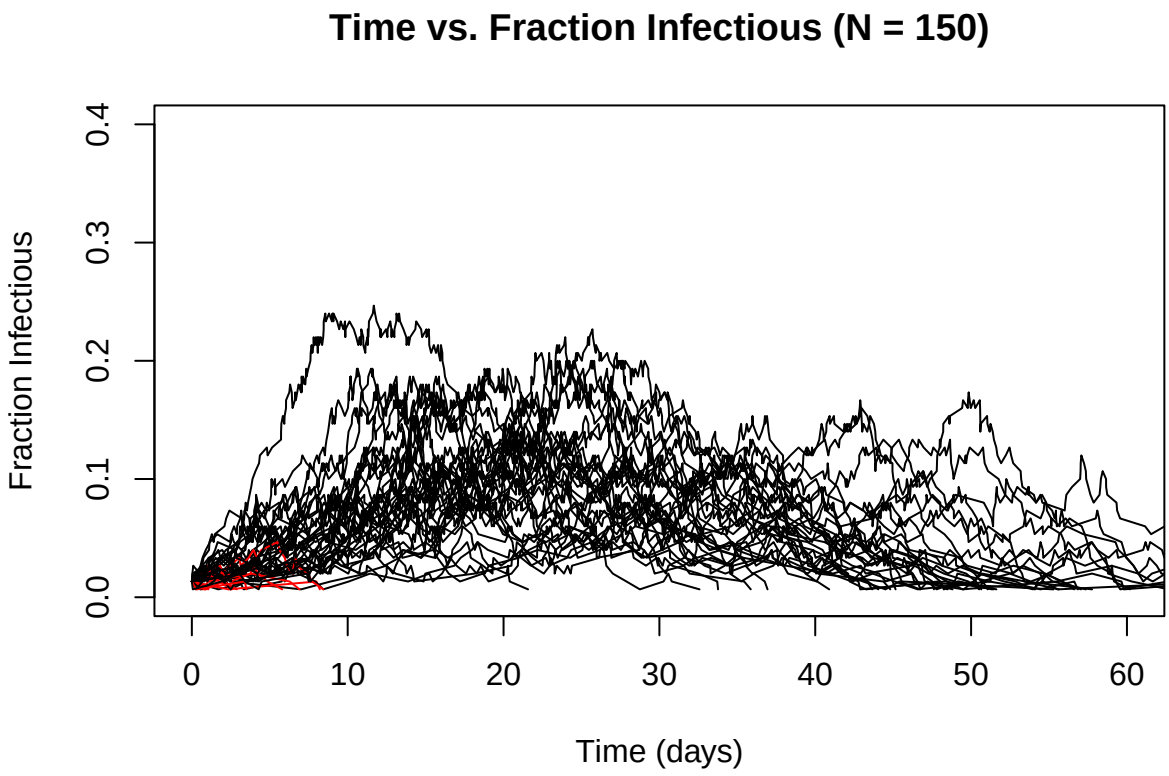
Create two figures:

- In the first figure, generate 50 realizations of the model below, using `pop_N = 150`.
- In the second figure, plot 50 realizations of the model, with the same parameters, but `pop_N = 5000`.
- In both figures, for each realization, if the infection burns out before 20 days, make the line red.

The axes of these plots should be the *fraction* infectious versus time. Plotting the fraction infectious helps us to better compare the scenarios with different population sizes, because it puts the output on the same scale.

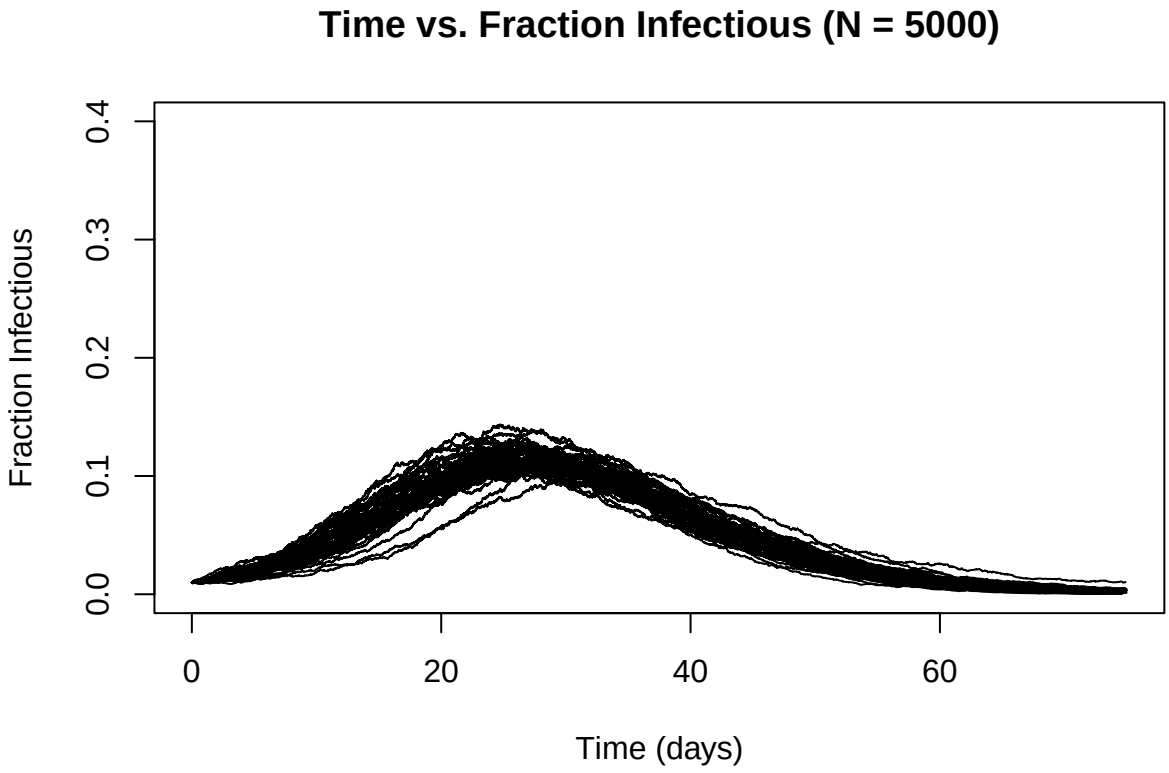
```
plot(NA, NA,
     xlim = c(0, t_max), ylim = c(0, 0.4),
     main = "Time vs. Fraction Infectious (N = 150)",
     xlab = "Time (days)", ylab = "Fraction Infectious")

for (i in 1:50)
{
  # Time
  t_now = 0
  next_event = 0
  t_max = 75
  t_vec = NULL

  # Population size
  pop_N = 150
  y_now = ceiling(pop_N * 0.01) # what is this doing?
  x_now = pop_N - y_now
  z_now = 0

  y_vec = NULL
  x_vec = NULL
  z_vec = NULL

  # Parameters
  beta = 0.35
  beta_scaled = beta / pop_N
  gamma = 1/5

  # counter
  count = 1

  # What does the while() function do?
  while(x_now > 0 & y_now > 0 & t_now <= t_max){

    # Store your variables
    t_vec[count] = t_now
```

```
    x_vec[count] = x_now
    y_vec[count] = y_now
    z_vec[count] = z_now

    # Draw the next event time
    SIR_rate = (beta_scaled * x_now * y_now) + (gamma * y_now)
    next_event = rexp(n=1, rate=SIR_rate)
    t_now = t_now + next_event

    # Decide the event
    # And update the variables
    rand = runif(n=1, min=0, max=1) # uniform random variate
    trans_prob = (beta_scaled * x_now * y_now) / SIR_rate

    if(rand < trans_prob){
      # Transmission event
      x_now = x_now - 1
      y_now = y_now + 1
      z_now = z_now
    }else{
      # Removal event
      x_now = x_now
      y_now = y_now - 1
      z_now = z_now + 1
    }

    count = count + 1

  }

  # Plot Fraction Infectious over time

  FractI = y_vec / (y_vec + x_vec + z_vec)
  if (max(t_vec) <= 20)
  {
    lines(FractI ~ t_vec, col = "red")
  } else {
    lines(FractI ~ t_vec, col = "black")
  }
}
```




```
plot(NA, NA,
     xlim = c(0, t_max), ylim = c(0, 0.4),
     main = "Time vs. Fraction Infectious (N = 5000)",
     xlab = "Time (days)", ylab = "Fraction Infectious")

for (i in 1:50)
{
  # Time
  t_now = 0
  next_event = 0
  t_max = 75
  t_vec = NULL

  # Population size
  pop_N = 5000
  y_now = ceiling(pop_N * 0.01) # what is this doing?
  x_now = pop_N - y_now
  z_now = 0

  y_vec = NULL
  x_vec = NULL
  z_vec = NULL

  # Parameters
  beta = 0.35
  beta_scaled = beta / pop_N
```

```
  gamma = 1/5

  # counter
  count = 1

  # What does the while() function do?
  while(x_now > 0 & y_now > 0 & t_now <= t_max){

    # Store your variables
    t_vec[count] = t_now
    x_vec[count] = x_now
    y_vec[count] = y_now
    z_vec[count] = z_now

    # Draw the next event time
    SIR_rate = (beta_scaled * x_now * y_now) + (gamma * y_now)
    next_event = rexp(n=1, rate=SIR_rate)
    t_now = t_now + next_event

    # Decide the event
    # And update the variables
    rand = runif(n=1, min=0, max=1) # uniform random variate
    trans_prob = (beta_scaled * x_now * y_now) / SIR_rate

    if(rand < trans_prob){
      # Transmission event
      x_now = x_now - 1
      y_now = y_now + 1
      z_now = z_now
    }else{
      # Removal event
      x_now = x_now
      y_now = y_now - 1
      z_now = z_now + 1
    }

    count = count + 1

  }

  # Plot Fraction Infectious over time

  FractI = y_vec / (y_vec + x_vec + z_vec)
  if (max(t_vec) <= 20)
  {
    lines(FractI ~ t_vec, col = "red")
  } else {
    lines(FractI ~ t_vec, col = "black")
  }
}
```


Time vs. Fraction Infectious (N = 5000)
Fraction Infectious
0.0
0.1
0.2
0.3
0.4
0
20
40
60
Time (days)