Assignment2

Richard McCormick

February 3rd, 2022

Task 1

The SIR model with demography was originally conceived to explain the cyclical dynamics of infectious disease outbreaks. For instance, before the measles vaccine was introduced, countries all over the world experienced large, yearly outbreaks of measles. The figure below shows the weekly number of cases of measles across England and Wales in the time before the vaccine.

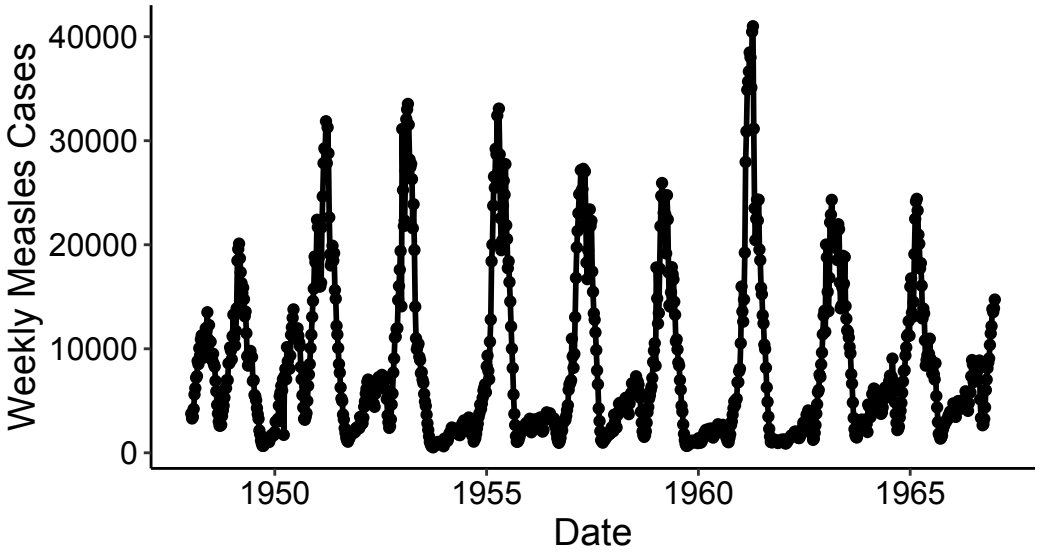


Figure 1: Weekly cases of measles reported across England and Wales. Data from Dr. Ben Bolker.

Task 1.1 (10 points)

First, verify that the SIR model with demography shows damped oscillations towards the endemic equilibrium when $R_0 > 1$.

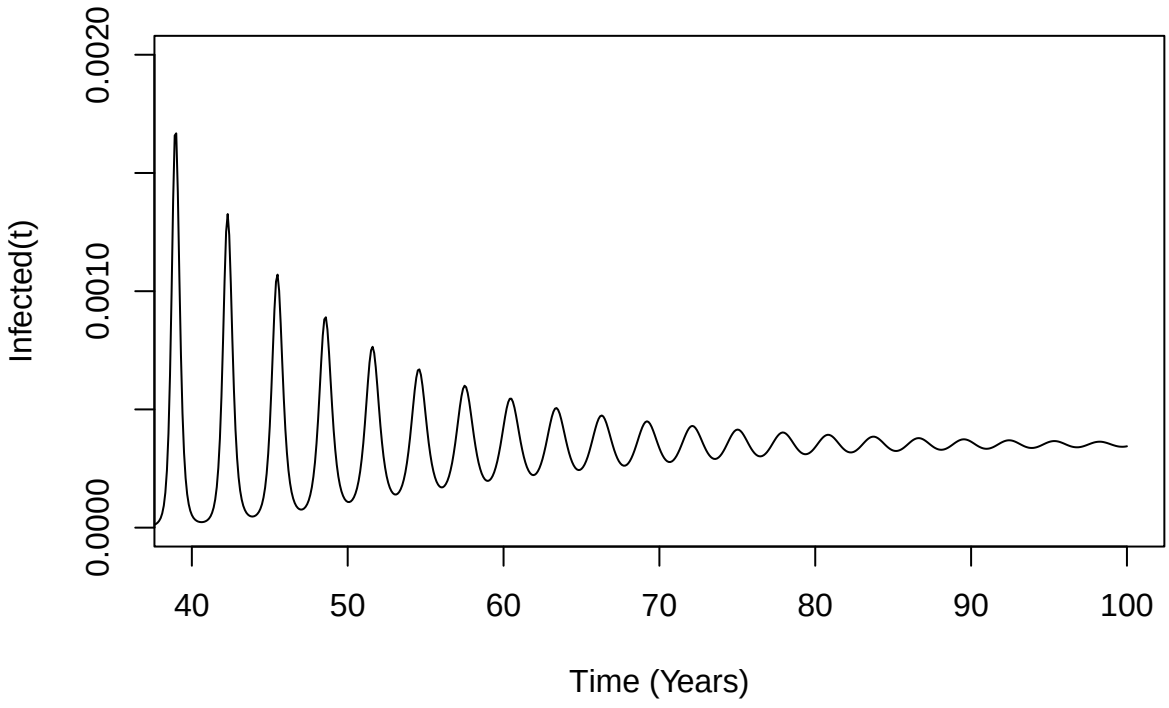
1. Adapt the code from Assignment 1 to solve the ODE system for the SIR model with demography.
2. Create a figure that shows $I(\hat{t})$ versus time. You must label the axes with the correct units.

You should use the following parameter values, but you need to think about the units of measure on these parameters.

```
{
  # Define the parameters
  if(!(require(deSolve))) install.packages("deSolve")

  library(deSolve)

  beta = 1.0 * 365
  gamma = 1/10 * 365
  mu = 1/70

  R_naught = beta/(gamma+mu)

  S_t0 = 0.999
  I_t0 = 1 - S_t0
  R_t0 = 0

  # Store in a vector
  inits = c(S_t0, I_t0, R_t0)

  # Store parameters in a named list
  params = c(beta = beta,
             gamma = gamma,
             mu = mu)

  # ODE SYSTEM FUNCTION
  SIR_ode = function(t, y, params){
    with(as.list(c(y, params)), {
      dydt = rep(0, 3)

      dydt[1] = mu -beta * y[1] * y[2] - mu *y[1]
      dydt[2] = beta * y[1] * y[2] - (gamma+mu) * y[2]
      dydt[3] = gamma * y[2] - mu*y[3]

      return(list(dydt))

    })
  }

  time_ode = seq(0, 100, by = 0.1)

  # Run ODE
  out = ode(y = inits,
            times = time_ode,
            func = SIR_ode,
            method="ode45", # Runge-Kutta 4-5 method
```

```
            parms = params)

  # Object 'out' is a matrix
  # Specify the column names, convert to data frame:
  colnames(out) = c("time", "S", "I", "R")
  out = data.frame(out)

  plot(x=NA,
     y=NA,
     xlim = c(40, max(time_ode)),
     ylim = c(0, 0.002),
     xlab = "Time (Years)",
     ylab = "Infected(t)")

  # Plot one line at a time:
  lines(out$I ~ out$time, lty = 1, col = "black")

  #Hint: you will need to adjust the y-axis limits to adequately visualize the damped oscillations.
}
```

```
## Loading required package: deSolve
```



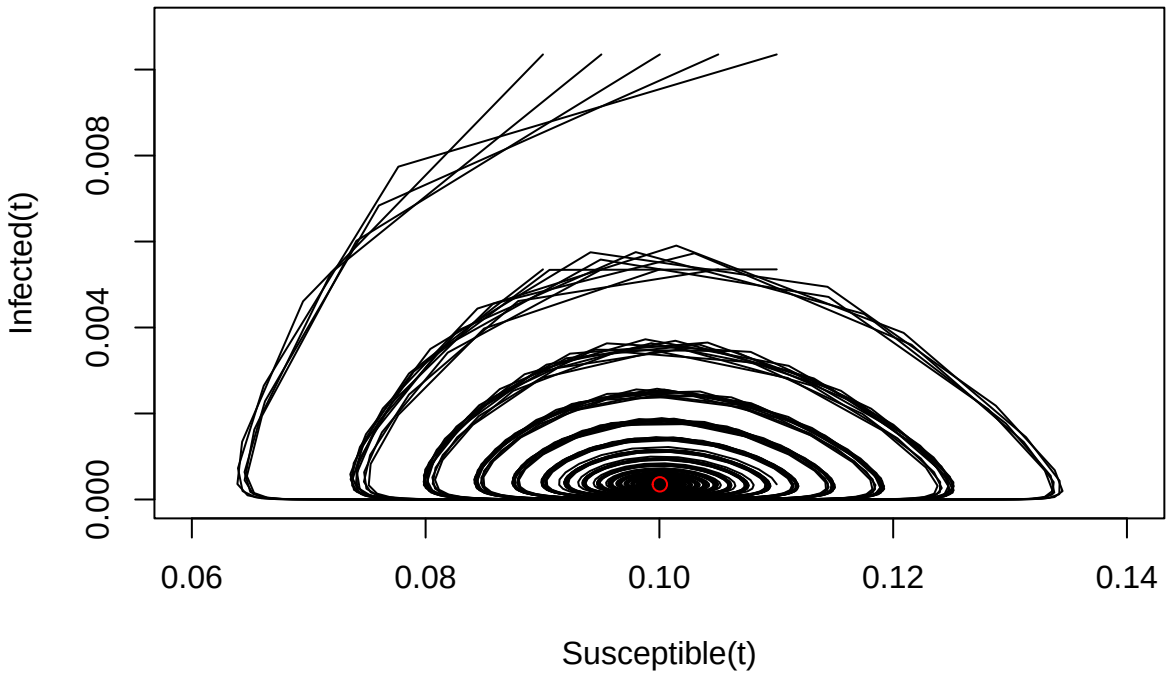
Task 1.2 (10 points)

Create a phase diagram for the SIR model with host demography. Use the parameter values given above. For this phase diagram, however, vary both $I(0)$ and $S(0)$ across five values each, in a pairwise fashion. This

will result in 25 total trajectories on your plot. Put a red point onto the graph that represents the endemic equilibrium.

```
# Hint: Set the ranges of $I(0)$ and $S(0)$ close to their equilibrium values $I^*$ and $S^*$.
#       You want a slight perturbation away from the equilibrium values
beta = 1.0 * 365
gamma = 1/10 * 365
mu = 1/70

R_naught = beta/(gamma+mu)

S_Equil = (1/R_naught)
I_Equil = (mu/beta)*(R_naught-1)

I_t0_temp = seq(I_Equil-0.01, I_Equil+0.01, length.out=5)
S_t0_temp = seq(S_Equil-0.01, S_Equil+0.01, length.out=5)

S_t0 = 0.999
I_t0 = 1 - S_t0
R_t0 = 0

# Store in a vector
inits = c(S_t0, I_t0, R_t0)

# Store parameters in a named list
params = c(beta = beta,
           gamma = gamma,
           mu = mu)

# ODE SYSTEM FUNCTION
SIR_ode = function(t, y, params){
  with(as.list(c(y, params)), {
    dydt = rep(0, 3)

    dydt[1] = mu -beta * y[1] * y[2] - mu *y[1]
    dydt[2] = beta * y[1] * y[2] - (gamma+mu) * y[2]
    dydt[3] = gamma * y[2] - mu*y[3]

    return(list(dydt))

  })
}

time_ode = seq(0, 100, by = 0.1)


plot(x=NA,
   y=NA,
   xlim = c(0.06, 0.14),
   ylim = c(0, 0.011),
   xlab = "Susceptible(t)",
   ylab = "Infected(t)")

# You can use a nested for loop to get a combinatorial of
```

```r
# I_t0_temp and S_t0_temp
for (i in 1:5)
{
  for (j in 1:5)
  {
    S_t0 = S_t0_temp[i]
    I_t0 = I_t0_temp[j]

    inits = c(S_t0, I_t0, R_t0)

    out = ode(y = inits,
          times = time_ode,
          func = SIR_ode,
          method="ode45", # Runge-Kutta 4-5 method
          parms = params)

    colnames(out) = c("time", "S", "I", "R")
    out = data.frame(out)

    lines(out$I ~ out$S, lty = 1, col = "black")
  }
}

points(S_Equil, I_Equil, col="red")
```



```
# Plot the equilibrium point using
# points(...)
```

Task 1.3 (5 points)

Think about the dynamics of the SIR model with demography and how they apply to childhood diseases or diseases like influenza that show yearly outbreaks. The model predicts damped oscillations to an endemic state, assuming $R_0 > 1$. However, we know that many diseases often show cyclical patterns that do not necessarily dampen (i.e., oscillations seem sustained through time). What are some biological reasons why the model and data do not match well? In other words, what are some assumptions that the model makes that might not be valid for some recurrent diseases? Or, what might we need to add to the model to be more realistic?

The model makes certain assumptions that are not true in all cases. For example, it assumes that susceptible, infected, and recovered individuals can all continue to contribute to new births, and that all new births are into the susceptible class. It also assumes that transmission requires contact between the susceptible and the infected, and that all recoveries and removals occur at an equal rate in the population.

In order to make the model more realistic, we should add seperate risk classes into the model to represent people who are more or less likely to become infected. These risk classes also allow a more accurate representation of the recovered/removed group.

Task 2

Your second task is to apply the techniques of linear stability to a new system. A well-studied concept in ecology is the dynamics between consumers (e.g., predators) and their resources (e.g., prey). In Canada, a classic predator-prey system is that of lynx and hares. The data below show how lynx and hare populations cycle through time. When prey populations become large enough, this causes a numerical response from their predators, allowing the predator populations to rise. Then, the increase in predator populations drives the decline of prey, but this decline causes a concomitant decline in predators, as their resource runs out. This leads to a classic, slightly out-of-sync cycling of predators and prey.

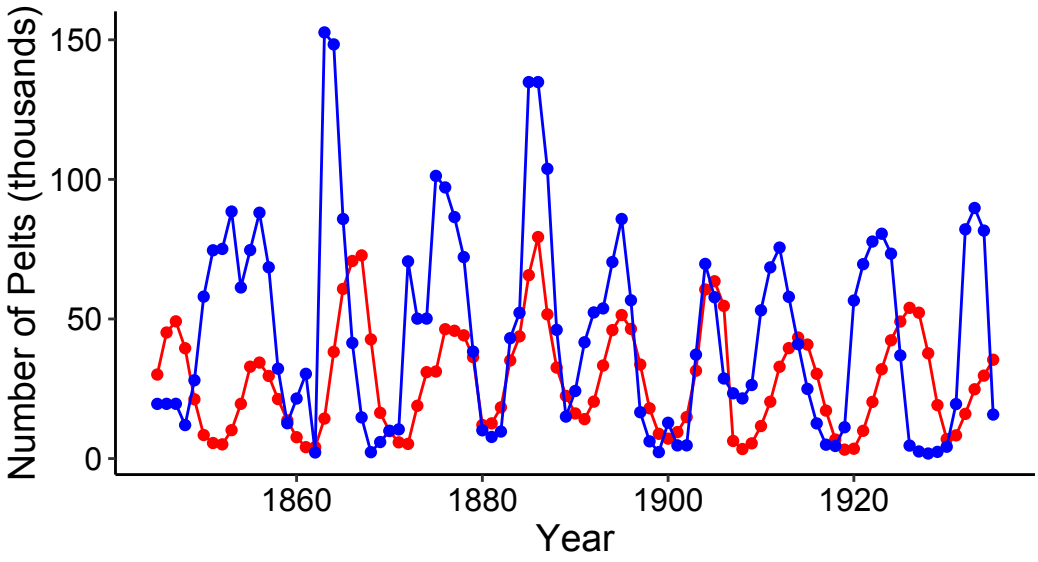


Figure 2: The number of pelts reported by trappers for lynx (red) and hare (blue) in Canada. Data from Hudson's Bay Company of Canada

Robert MacArthur, a well-known ecologist, formulated a generalized model to describe the dynamics of consumers (X) and their resources (R):

$$\frac{dR}{dt} = rR - \rho XR$$
$$\frac{dX}{dt} = bX(\rho R - m)$$

In this model, resources R grow exponentially at per-capita rate, r. Consumption of resources is mediated by contact with the consumers, where ρ is the rate that consumption occurs, given contact, similar to transmission in the SIR model. The consumer population grows by converting energy received from consuming the resource into energy for reproduction. Thus, b represents the conversion factor, whereby consumers convert any excess energy into reproductive output, while m represents the consumer's maintenance requirement (e.g., death and physiological maintenance energy). In other words, m represents the rate that the consumer loses the energy gained from consuming the resource to all processes other than reproduction.

Task 2.1 (20 points)

1. Calculate the coexistence equilibrium of this consumer-resource system (i.e., the equilibrium where both X^* and R^* are greater than one). Show your work.

$$
\begin{aligned}
\frac{dR}{dt} = 0 &= rR - \rho XR \\
\rho XR &= rR \\
rhoX &= r \\
X &= \frac{r}{\rho} \\
\frac{dX}{dt} = 0 &= bX(\rho R - m) \\
0 &= bX\rho R - mbX \\
mbX &= bX\rho R \\
m &= \rho R \\
\frac{m}{\rho} &= R
\end{aligned}
$$

2. Use the parameters below to generate the Jacobian matrix for this coexistence equilibrium.
3. Calculate the eigenvalues, using the supplied parameter values. Interpret the eigenvalues and make a prediction for how the system behaves around the equilibrium point.

```
# Params:
# b: conversion factor (converts excess energy into reproductive output)
# r: per-capita growth rate of the resource
# rho: capture and consumption rate of resource
# m: maintenance requirement (death and physiological maintenance energy)

b = 0.1
r = 1.2
rho = 0.5
m = 0.2

r_star = m / rho
x_star = r / rho

# Build the jacobian matrix:
J = matrix(0, nrow = 2, ncol = 2)

# Row 1
J[1,1] = b * ((rho * r_star) - m)
J[1,2] = b * x_star * (rho)

# Row 2
J[2,1] = -rho * r_star
J[2,2] = r - (rho * x_star)

# Calculate the eigenvalues:
eigen_vals = eigen(J)

eigen_vals$values[1]
```

```
## [1] 0+0.1549193i
```

```
eigen_vals$values[2]
```

```
## [1] 0-0.1549193i
```

```
#####
# Real parts: 0
# Imaginary parts: One negative, one positive
# Conclusion: Neutrally stable, sustained ocilations
```

Task 2.2 (15 points)

Adapt the code that you've created for the SIR model to numerically solve this consumer-resource model with the given parameters. Use initial conditions $X(0) = 2.0$ and $R(0) = 0.4$, which represent densities of organisms per hectare. Assume the time units is weeks. Show that the prediction of the system dynamics from your stability analysis is accurate by plotting the dynamics of both the consumer and resource on the same figure, with their densities as functions of time.

```
X_t0 = 2.0
R_t0 = 0.4

if(!(require(deSolve))) install.packages("deSolve")

library(deSolve)

inits = (c(R_t0, X_t0))

b = 0.1
r = 1.2
rho = 0.5
m = 0.2

params = c(b = b,
           r = r,
           m = m,
           rho = rho)

# ODE SYSTEM FUNCTION
XR_ode = function(t, y, params){
  with(as.list(c(y, params)), {
    dydt = rep(0, 2)

    dydt[1] = (r * y[1]) - (rho * y[2] * y[1])
    dydt[2] = (b * y[2] * (rho * y[1] - m))

    return(list(dydt))

  })
}

time_ode = seq(0, 500, by = 0.1)

out = ode(y = inits,
      times = time_ode,
      func = XR_ode,
      method="ode45", # Runge-Kutta 4-5 method
```

```
      parms = params)

colnames(out) = c("time", "R", "X")
out = data.frame(out)

plot(x=NA,
   y=NA,
   xlim = c(0, max(out$time)),
   ylim = c(0, max(out$X)),
   xlab = "Time",
   ylab = "X(t) & R(t)")

lines(out$X ~ out$time, lty = 1, col = "black")
lines(out$R ~ out$time, lty = 1, col = "red")
```