Assignment1

Richard McCormick - RLM443

January 18th, 2022

ODE systems in R

We need to install the package *deSolve* - to help us numerically approximate systems of differential equations.

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

library(deSolve)
```

Now we can set up a differential equation function, which is used by the *deSolve* package to solve the ODE system. This is the system in LaTeX form:

$$\frac{dS}{dt} = -\beta SI$$
$$\frac{dI}{dt} = \beta SI - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

```
# ODE SYSTEM FUNCTION
SIR_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] = -beta * y[1] * y[2]                # S, susceptible hosts
    dydt[2] = beta * y[1] * y[2] - gamma * y[2] # I, infectious hosts
    dydt[3] = gamma * y[2]                       # R, recovered/removed hosts

    return(list(dydt))

  })
}
```

Now we can set up our parameters and initial conditions of the system to solve.

```
# Specify the parameter values
beta = 1.0
gamma = 1/10.0
R_naught = beta/gamma
print(paste0("R_naught = ", R_naught))
```

```
## [1] "R_naught = 10"
```

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

# Specify the initial conditions
# Remember, S+I+R = 1
S_t0 = 0.999
I_t0 = 1 - S_t0
R_t0 = 0

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

# Each time step that will be solved
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)
```

Plotting

Plot using base plotting:

```
# Set up a blank plot
plot(x=NA,
     y=NA,
     xlim = c(0, max(time_ode)),
     ylim = c(0, 1),
     xlab = "Time (days)",
     ylab = "Fraction of the Population")
# Plot one line at a time:
lines(out$S ~ out$time, lty = 1, col = "black") # S, suceptible hosts
lines(out$I ~ out$time, lty = 1, col = "red")   # I, infectious hosts
lines(out$R ~ out$time, lty = 1, col = "blue")  # R, removed hosts
```

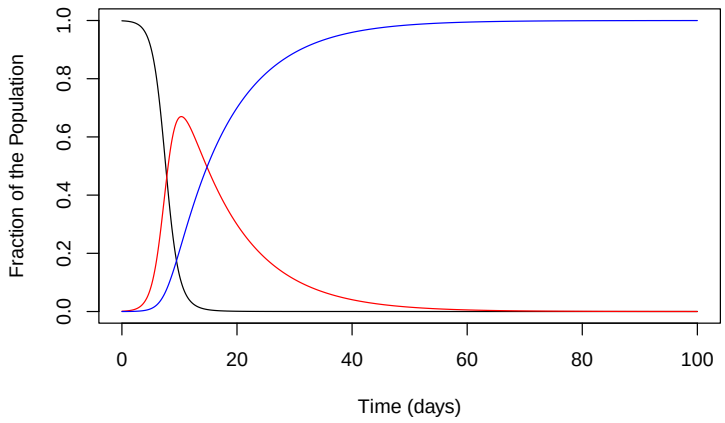


Figure 1: SIR model solution, plotted with base graphics.

Here's another way you could plot the lines (a little trickier). This type of method, however, might be useful later.

```
plot(x=NA,
     y=NA,
     xlim = c(0, max(time_ode)),
     ylim = c(0, 1),
     xlab = "Time (days)",
     ylab = "Fraction of the Population")

these_cols = c("black", "red", "blue")
# Loop through the lines
for(i in 1:3){
    lines(out[,(i+1)] ~ out$time, lty = 1, col = these_cols[i])
}
# I'm suppressing the output with 'eval=FALSE'
```

Task 1 (20 points)

The following figure shows the **phase diagram** of the SIR system with parameters $\beta = 1.0$ and $\gamma = 0.1$, and the initial value of $I(0) = 0.01$.

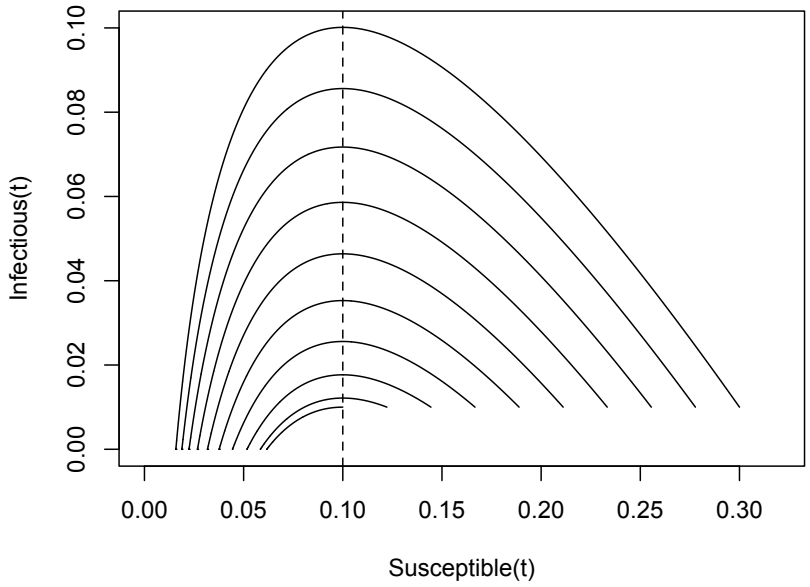


Figure 2: Phase diagram of the SIR system.

1.1 (10 points)

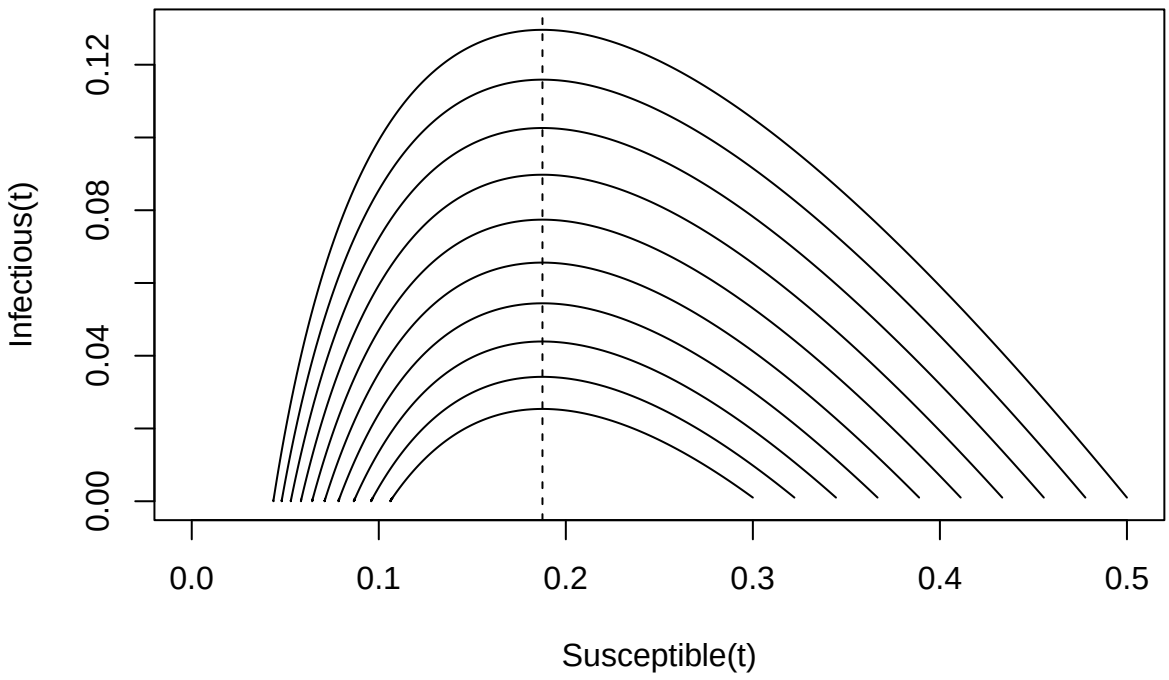
1. Describe in your own words what each solid line (trajectory) of the figure represents.

Each solid line of the figure represents the relationship between Susceptible Individuals and Infectious Individuals for an outbreak of disease with a given susceptible population. The larger lines represent populations with a higher initial fraction of susceptible people, and the smaller lines represent populations with lower initial fractions susceptible.

2. Develop your own script to re-create this figure, but with parameters $\beta = 0.8$ and $\gamma = 0.15$, and hold the initial value of $I(0) = 0.001$. Use 10 initial values of susceptible hosts $S(0)$. See hints below.

```
# Hint: Here's a sequence of S_vals to use
# i changed the s vals
S_vals = seq(0.3, 0.5, length.out = 10)

beta = 0.8
gamma = 0.15
params = c(beta = beta,
           gamma = gamma)

plot(x=NA,
   y=NA,
   xlim = c(min(out$S), max(S_vals)),
   ylim = c(0, 0.13),
```

```
   xlab = "Susceptible(t)",
   ylab = "Infectious(t)")

# Each time step that will be solved
time_ode = seq(0, 500, by = 0.1)

SIR_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] = -beta * y[1] * y[2]                # S, susceptible hosts
    dydt[2] = beta * y[1] * y[2] - gamma * y[2] # I, infectious hosts
    dydt[3] = gamma * y[2]                       # R, recovered/removed hosts

    return(list(dydt))

  })
}

for (i in 1:10){
  S_t0 = S_vals[i]
  I_t0 = 0.001
  R_t0 = 0

  inits = c(S_t0, I_t0, R_t0)

  # 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)

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

abline(v=(gamma/beta), lty=2)
```



1.2 (10 points)

1. In your own words, what does the vertical dashed line in the phase diagram represent, biologically speaking?

The vertical dashed line in the phase diagram represents the peak of the infection, or the point at which the highest number of people are getting infected at one time. It is also the point where the rate of change of new infections is equal to zero. Before this point, new infections are increasing; after this point, new infections will decrease. The herd immunity threshold.

2. Show how to derive this value mathematically. I would prefer you to show your work in LaTeX-style equations (e.g., see above). I will instead accept a photo of your work on scratch paper, with this attached to your BBLearn submission.

$$\begin{aligned}
\frac{dI}{dt} &= \beta S_0 I - \gamma I \\
\frac{dI}{dt} &= I(\beta S_0 - \gamma) \\
0 &> I(\beta S_0 - \gamma) \\
\gamma &> (\beta S_0) \\
\frac{\gamma}{\beta} &> S_0
\end{aligned}$$

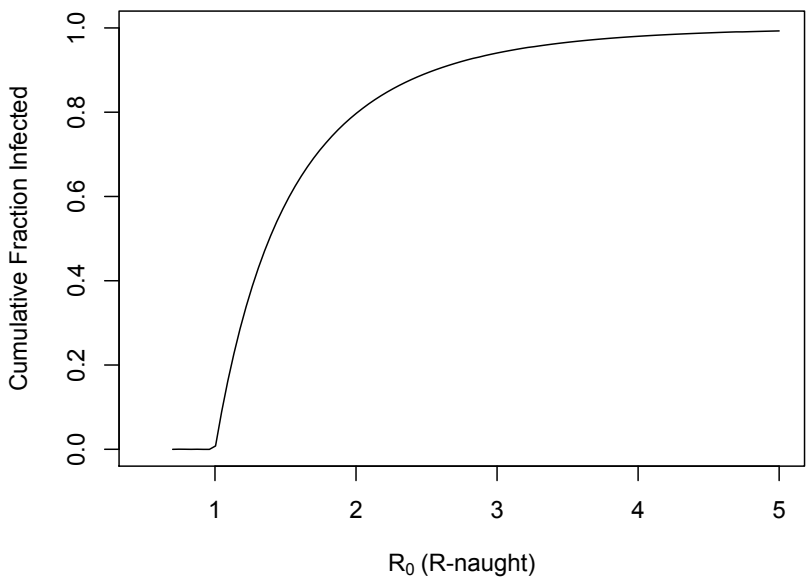


Figure 3: Figure for you to explain.

Task 2 (25 points)

2.1 (5 points)

What are the units on R_0? Show this mathematically. Again, use LaTeX or attach a picture of your work on scratch paper.

$$R_0 = \frac{\beta}{\gamma}$$

$$\beta = host * time$$
$$\gamma = \frac{time}{host}$$

$$R_0 = host * host$$

(?)

2.2 (10 points)

1. What does this figure represent?

This figure represents the total cumulative fraction of the population that will become infected over the course of an epidemic for a pathogen with the given R-Naught value. This fraction represents what percentage of the population will become infected before the virus burns out and is no longer able to spread.

2. Describe, in your own words what the following code is doing. Also, what do R_inf and R_naught represent biologically. What are the dynamics of the curve in the figure (i.e., why is it shaped that way)?

```
burn_out = function(R_inf, R_naught){
  1 - exp(-R_naught * R_inf) - R_inf
}

temp_R_naught = 2.0

temp_root =  uniroot(burn_out, interval = c(0.01, 1), extendInt = "yes",
                     R_naught = temp_R_naught, tol = 0.001)

root = temp_root$root
paste("The root of the equation is:", round(root,3))
```

```
## [1] "The root of the equation is: 0.797"
```

The above code finds the total cumulative fraction of the population that will be infected for a virus with the given R_0 value. It produces one point on the above graph from one point of given data.

Mathematically, R_∞ represents the total number of people which will become infected and eventually removed from the susceptible population over the entire course of the epidemic. R_0 represents the average number of immunilogically naive people that a single infectious host will transmit the pathogen to.

\textbf{When R_0 is less than one, the virus is not spreading fast enough to infect a growing number of people, so the total fraction infected will stay near zero. When R_0 is greater than or equal to one, it will be able to infect people faster than those people recover, and the cumulative fraction infected will greatly increase, plateauing at 100% of the population.}

2.2 (10 points)

Develop code to replicate the above figure. Vary R_0 from 0.7 to 5.0. Hints below.

```
# HINTS:
# Values to test:
n_R_naught = 100
temp_R_naught = seq(0.7, 5.0, length.out = n_R_naught)
# Allocate storage:
roots = vector(mode = "numeric", length = n_R_naught)

burn_out = function(R_inf, R_naught){
  1 - exp(-R_naught * R_inf) - R_inf
}

plot(x=NA,
   y=NA,
   xlim = c(0, max(temp_R_naught)),
   ylim = c(0, 1),
   xlab = "R-Naught",
   ylab = "Total Fraction Infected")

for (i in 1:100)
{
```

```
  temp_root =  uniroot(burn_out, interval = c(0.01, 1), extendInt = "yes",
                       R_naught = temp_R_naught[i], tol = 0.001)

  roots[i] = temp_root$root
}

lines(roots ~ temp_R_naught, lty = 1, col = "black")
```