Assignment6

Richard McCormick

March 8th, 2022

Here we will implement seasonal forcing in the SIR model with demography. First, we will assume transmission rate is forced with a sinusoidal forcing function, and we will examine the implications. The model becomes:

$$\frac{dS}{dt} = \mu - \beta(t)SI - \mu S$$
$$\frac{dI}{dt} = \beta(t)SI - (\gamma + \mu)I$$
$$\frac{dR}{dt} = \gamma I - \mu R$$

For the forcing function, we will assume $\beta(t) = \beta_0(1 + cos(\frac{2\pi t}{t_{\text{mode}}}))$.

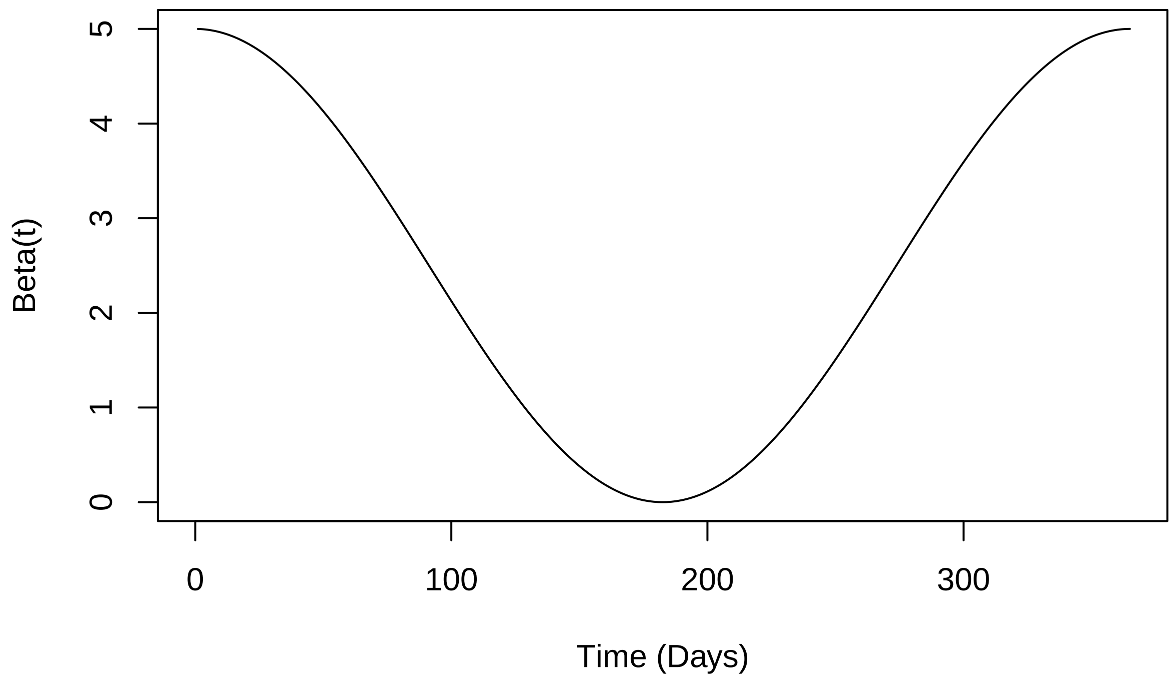
In Assignment 4, we learned how to solve a system of ODEs one day at a time. In that case, we were assigning a new value of $\beta(t)$ each day, based on the value of R_t per day. With seasonal forcing, we are simply replacing this for a new function of what the value of $\beta(t)$ should be each day.

Task 1 (5 points)

Create a figure that shows β as a function of time, using the sinusoidal forcing function specified above. Set $\beta_0 = 2.5$ and $t_{\text{mode}} = 365$. Have your time range from 0 to 365 days. Label your axes appropriately.

```
library(deSolve)

beta_calc = function(beta_zero, t, t_mode){

    beta_t = beta_zero * (1 + cos((2 * pi * t)/(t_mode)))

    return(beta_t)
}

beta_zero = 2.5
t_mode = 365
t = seq(1, 365, by = 1)

beta_storage = vector(mode="numeric", length = 365)

for (i in t){
  beta_storage[i] = beta_calc(beta_zero, t[i], t_mode)
}

plot(x=NA,
   y=NA,
   xlim = c(0, 365),
```

```r
   ylim = c(0, max(beta_storage)),
   xlab = "Time (Days)",
   ylab = "Beta(t)")

lines(beta_storage ~ t, lty = 1, col = "black")
```



Task 2 (25 points)

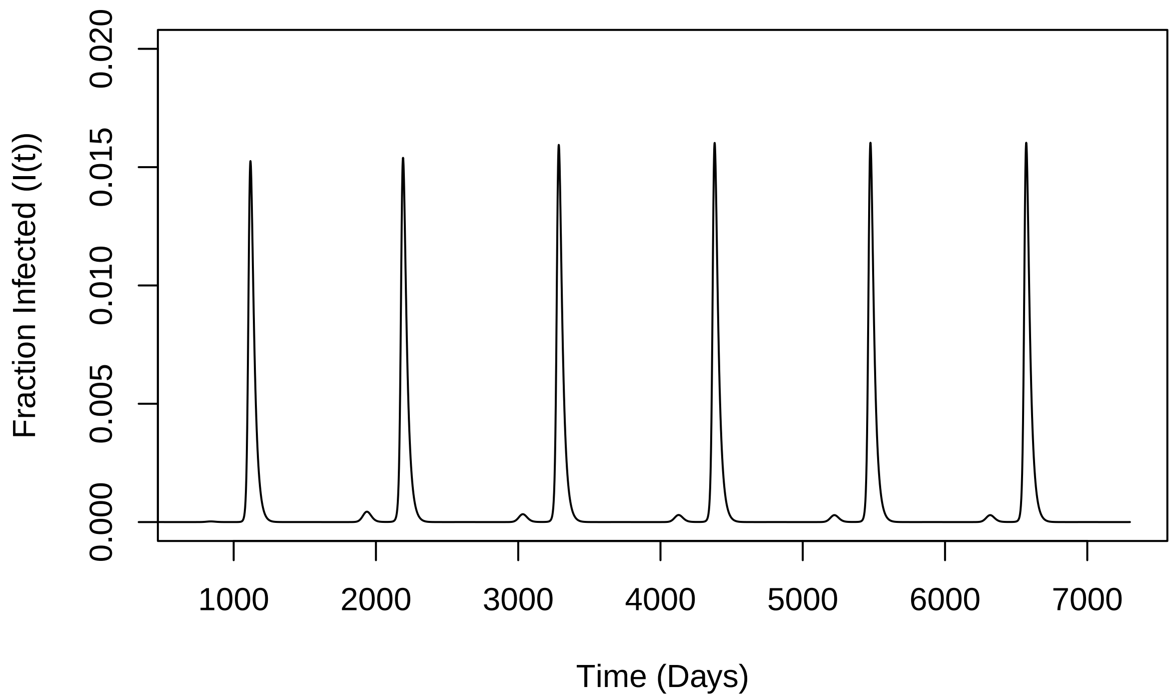
1. Adapt your code from Assignment 4 to solve the SIR model *with demography* using the forcing function $\beta(t) = \beta_0(1 + cos(\frac{2\pi t}{t_{\text{mode}}}))$ to assign $\beta(t)$. *Hints*: Re-code the function for daily beta to use the sinusoidal function. Also, we are no longer interested in simulating daily incidence, so remove those parts of the code.

2. Create a plot of the seasonally-forced dynamics across 2 decades, but remember that you are solving the equations one *day* at a time. Plot the Infectious class, $I(t)$ versus time. Use the following parameters and intial conditions:

```r
# static parameter values
mu = 1/(70*365)
gamma = 1 / 20
# To calculate the $\beta(t)$:
beta_0 = 2.5
t_hat = 365

# Intial conditions to use:
I0 = 0.001
```

```r
S0 = 1 - I0
R0 = 0
inits = c(S0, I0, R0)

t = seq(1, 365*20, by = 1)

# Another hint:
# To see the dynamics better, toss out the first 2 years
# This removes the "transient" behavior before it stabilizes
# Example: sub_out = out %>% filter(time >= 365 * 2)
# Then plot using the 'sub_out' data frame

SIR_ode = function(t, y, params){
    with(as.list(c(y, params)), {
        beta = beta_calc(beta_0, t, t_hat)
        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))

    })
}

# Function 2
# One step function to solve given ODE, specifically SIR. Pump out dataframe
out_func = function(inits, params){

    out_temp = ode(y = inits,
                   times = t,
                   func = SIR_ode,
                   method="ode45",
                   parms = params)

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

    return(out_temp)
}

params = c(gamma = gamma)

out = out_func(inits, params)


plot(x=NA,
   y=NA,
   xlim = c(365*2, max(t)),
   ylim = c(0, 0.02),
   xlab = "Time (Days)",
   ylab = "Fraction Infected (I(t))")
```

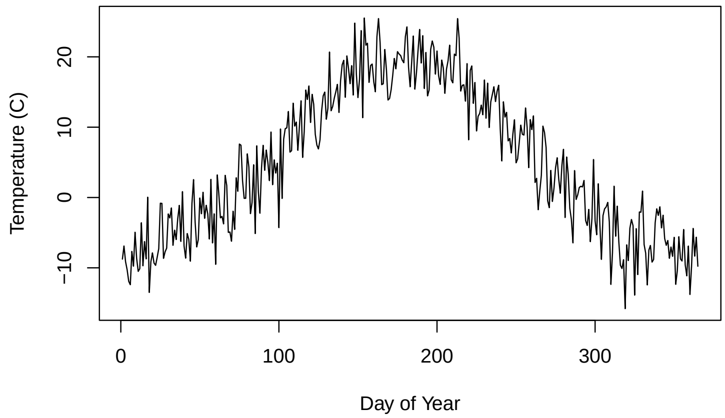
```r
lines(out$I ~ out$time, lty = 1, col = "black")
```



Task 3 (5 points)

Here, instead of using a sinusoidal forcing function, we'll link transmission rate directly to (simulated) data on daily temperature fluctuations in a single year.

```r
# Range of time (one year)
t_range = 1:365
# Gaussian-type temperature gradient, with a peak at day 180
temperature = 30 * exp(-0.5 * (t_range - 180)^2 / 5000) - 10
# Add some noise:
temperature = temperature + rnorm(length(temperature), 0, 3.5)

plot(temperature ~ t_range, type = "l",
     xlab = "Day of Year",
     ylab = "Temperature (C)")
```

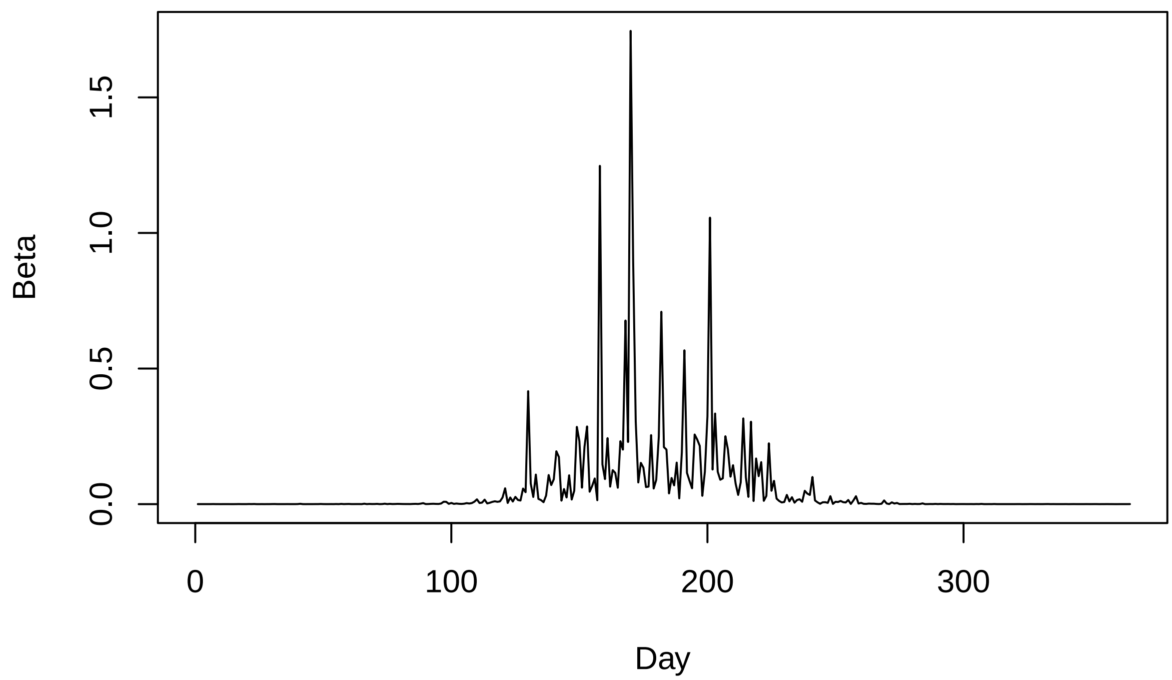


Assume that the transmission rate β is an exponential function of temperature, of the form $\beta(T) = \beta_0 e^{\alpha T}$, where T is the time-specific value of temperature. Set $\beta_0 = 0.0005$, such that when the temperature is zero, transmission is very low. And set $\alpha = 0.3$.

Using this covariate-based forcing equation, plot the relationship between β and day of the year. Label your axes appropriately.

```
beta_0 = 0.0005
alpha = 0.3
t_mode = 365
t = seq(1, 365, by = 1)

t_range = 1:365
# Gaussian-type temperature gradient, with a peak at day 180
temperature = 30 * exp(-0.5 * (t_range - 180)^2 / 5000) - 10

beta_storage = vector(mode="numeric", length = 365)

beta_calc = function(beta_zero, t, temp){

    beta_t = beta_zero * exp(alpha * temp)

    return(beta_t)
}

temperature = 30 * exp(-0.5 * (t_range - 180)^2 / 5000) - 10 + rnorm(length(temperature), 0, 3.5)

for (i in 1:365){
  beta_storage[i] = beta_calc(beta_0, t[i], temperature[i])
}

plot(x=NA,
   y=NA,
   xlim = c(0, max(t)),
   ylim = c(0, max(beta_storage)),
   xlab = "Day",
   ylab = "Beta")
```

```r
lines(beta_storage ~ t, lty = 1, col = "black")
```

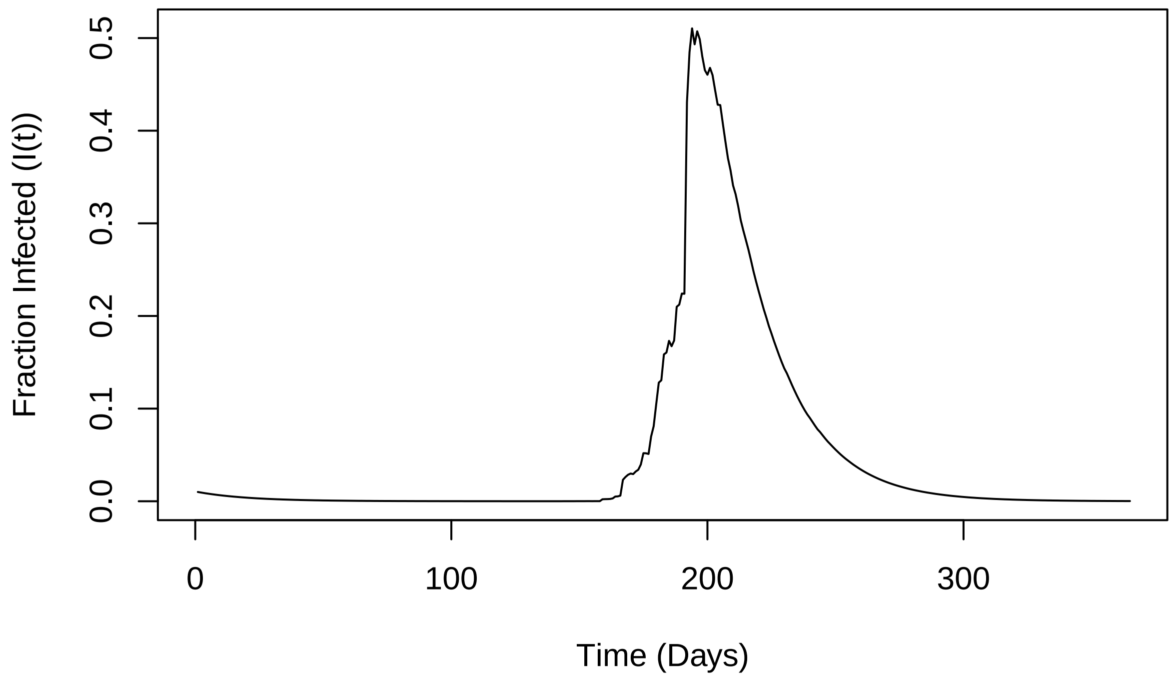


Task 4 (15 points)

Your final task is to alter the parameter assignments and other appropriate elements in your code to plot the SIR dynamics *for a single year*, using the covariate-based forcing (equation in Task 3). HINT: Set $I(0) = 0.01$ and $S(0) = 1 - I(0)$ Plot the fraction infectious over time for one year using the covariate-based forcing.

```r
# static parameter values
mu = 1/(70*365)
gamma = 1 / 20
# To calculate the $\beta(t)$:
beta_0 = 0.0005
t_hat = 365

# Intial conditions to use:
I0 = 0.01
S0 = 1 - I0
R0 = 0
inits = c(S0, I0, R0)

beta_0 = 0.0005
alpha = 0.3
t_mode = 365
t = seq(1, 365, by = 1)
```

```r
t_range = 1:365
# Gaussian-type temperature gradient, with a peak at day 180
temperature = 30 * exp(-0.5 * (t_range - 180)^2 / 5000) - 10

beta_calc = function(beta_zero, t, temp){

    beta_t = beta_zero * exp(alpha * temp)

    return(beta_t)
}

temperature = 30 * exp(-0.5 * (t_range - 180)^2 / 5000) - 10 + rnorm(length(temperature), 0, 3.5)

time_ode = seq(1, 365, by = 1)

SIR_ODE = function(t, y, params){
    with(as.list(c(y, params)), {
        dydt = rep(0, 3)

        beta = beta_calc(beta_0, t[t], temperature[t])

        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))
    })
}
params = c(gamma = gamma,
           mu = mu)

# 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(0, 365),
   ylim = c(0, max(out$I)),
   xlab = "Time (Days)",
   ylab = "Fraction Infected (I(t))")

lines(out$I ~ out$time, lty = 1, col = "black")
```


Fraction Infected (I(t))
0.0
0.1
0.2
0.3
0.4
0.5
0
100
200
300
Time (Days)