SIR Exercise

Alex Grimaudo

2023-04-25

SIR Model

In this exercise, we are going to construct a Susceptible-Infected-Recovered (SIR) model of infectious disease. The SIR model we construct is the basis of many models used in epidemiology to understand the spread of pathogens through a host population. There are many variations of the SIR model, each used for host-pathogen systems with different characteristics (later on, you will have the opportunity to construct an SIR model with a vaccination intervention). We are going to use several packages today, including *deSolve*, *tidyr*, and *ggplot2*. First, if you have not already done so, install and load the packages:

```
#install.packages('deSolve')
#install.packages('tidyr')
#install.packages('ggplot2')
library(deSolve)
library(tidyr)
library(ggplot2)
```

deSolve is a package for "solving" ordinary differential equations. We will use it to "solve" our SIR models. *ggplot2* is a package for visualizing data. We will use it to plot the results of our SIR models. *tidyr* is a package for data cleaning and organizing. We will use its *pivot_longer()* function to re-shape our simulated data for plotting.

Now that our package are loaded in, we can construct our basic SIR model by defining it as a function. Start by filling in the differential equations below:

$dS =$

$dI =$

$dR =$

Using the differential equations you wrote above, enter the chunk of code below into your script and fill in the missing spaces:

```
SIR <- function(time, state, parameters) {
  with(as.list(c(state, parameters)), {
    dS <- #Need an equation!
    dI <- #Need an equation!
    dR <- #Need an equation!
    return(list(c(dS, dI, dR)))
  })
}
```

The code above creates a new function, which we have called *SIR()*, that will calculate values of *dS*, *dI*, and *dR* based on the "state" and "parameters" that we give it. "State" refers to the classes of our SIR model: Susceptible, Infected, and Recovered. "Parameters" refers to a list of parameters that we incorporate into the model, which are currently β (the transmission rate) and γ (the recovery rate).

Of course, we need to tell our model how long to run for, which will be referred to as the "times." Here, we are going to arbitrarily define the length of the simulation as 400 time steps, which is unit-less (not days, weeks, hours, etc.). In practice, however, you will always need to make sure that your model is running over reasonable and meaningful durations and that **your parameters are scaled correctly to the length of a time step.** For example, the conclusions you draw from your model will be wrong if you accidentally define the transmission rate, β, in terms of hours but your time step on the order of days. **Always make sure the scale of your parameters match the units of time you are interested in.**

The below chunk of code defines the time duration of our model:

```
times <- seq(0, 400, by = 0.01)
```

Next, we need to give our model some starting conditions. Let's assume that at the very beginning of the epidemic, only 0.01% of our population is infected and nobody has any immunity from recovery. We can define these starting conditions using the code below (**notice that the initial size of the Susceptible class needs to be filled in**):

```
initial.I <- 0.0001
initial.R <- 0
initial.S <- # What will this be?

initial.values <- c(S = initial.S, I = initial.I, R = initial.R)
```

It is important to note that the above starting values are proportions of the population, not numbers of individuals. We could, however, easily modify our code to be in units of individuals, but today we will be working with proportions.

We will also need to define our values for β and γ. For now, let's start with a value of 0.2 for β and 0.1 for γ. **What is the basic reproduction number (R_0) of our model?**

```
parameters <- c(beta = 0.2, gamma = 0.1)
```

Now that we have defined the structure of our SIR model, provided initial class sizes, and given a value to β and γ, we can run our model! To do so, we are going to use the *ode()* function from the *deSolve* package, which "solves" differential equations, given a set of starting conditions. We will save the output of the model in a new dataframe called 'output', which we will then plot using the package *ggplot2*.

The below chunk of code runs our model (using the starting conditions and times we have defined), saves it as a new dataframe, and re-shapes it into long format using the *pivot_longer()* function from the *tidyr* package. You can run this code without the *pivot_longer()* to understand what the difference between wide and long format is. Long format is considered 'tidy' and is easier to work with when plotting and analyzing data.

```
output<- as.data.frame(ode(func = SIR, y = initial.values, parms = parameters, times = t
imes)) %>%
  pivot_longer(!time, names_to="state",values_to="proportion")

head(output)
# Use the function head() to view the first rows of your dataframe
```

Take a look at *output* datframe using function *head()* or *View()*. As you will see, the *ode()* function has solved the series of differential equations we defined in the *SIR()* function and they are now stored in a dataframe of three columns: time, state, and proportion.

Use the below chunk of code to plot the results of the model using the *ggplot2* package for data visualization:

```
SIR.plot <- ggplot(aes(x=time, y=proportion, color=state), data=output)+
  geom_line()+
  scale_y_continuous(limits=c(0,1), breaks=c(0,0.25,0.5,0.75,1))+
  scale_color_manual(values=c("red2","blue3","green4"));SIR.plot
```

In the space below, draw the graph that you see and label the graph with the values of β, γ, and R_0. **Try to preserve the values on the x- and y-axes. Leave some space to draw two more graphs.**

On the graph you drew, note somewhere what the maximum size of the epidemic was and at what time step it

occurred.

Now let's see what happens if we vary the transmission rate, β. Change β to 0.3 and leave γ the same and re-run your model. Draw the new plot to the right of the first one you drew, and again label it with the β value, γ value, and R_0. Note what the maximum epidemic size was and when it occurred.

Repeat the process with a β of 1.0.

Now, leaving β at a value of 1.0, change the value of γ to 0.33 and re-run the model. Draw the graph in the space below, again labeling with β, γ, and R_0, **leaving space for one additional graph.**

Holding β at 1.0, repeat the process again with a γ of 0.5.

Looking at all the graphs you've drawn, describe what happens as you increase or decrease each parameter. What happens when we increase β? What about when we increase γ? Keeping R_0 constant, what happens when we increase/decrease β and γ?

Set β equal to 0.1 and γ equal to 0.2. What happens? Why?

SIR Model with Vaccination

Now we will implement vaccinating our population as an intervention strategy. Let's re-visit our series of differntial equations and add a new class for vaccinated individuals. We are going to assume a few things: first, once an individual is vaccinated, they have **perfect** immunity, meaning there is a probability of 0 that they will become infected if they contact an infected individual. Second, once an individual is vaccinated, they are **immune for life**. This is similar to an assumption about our previous SIR model in which we assumed that once an individual recovered from infection, they had total, life-long immunity. If we wanted to, we could extend our model to include **leaky protection** (vaccines don't offer 100% protection) or **waning immunity** (the loss of immunity over time). Third, let's assume that we are only going to vaccinate susceptible individuals.

Re-write the differential equations, extending them for a new vaccination class (if you add a new parameter, let's make it theta (θ)):

dS =

dI =

dR =

dV =

Now let's define a new function for our revised SIR model with vaccination. Let's call the function *SIRv()*. Extend the code you've already written to incorporate vaccination:

```
SIRv <- function(time, state, parameters) {
  with(as.list(c(state, parameters)), {
    dS <- (-beta * S * I)
    dI <- (beta * S * I) - (gamma * I)
    dR <-(gamma * I)
    dV <-
    return(list(c(dS, dI, dR, dV)))
  })
}
```

We've already defined the amount of time we want the model to run, which is stored in the 'times' object. Now, we need to update our starting conditions to take into account the new vaccination class and θ (vaccination rate):

```
initial.I <- 0.0001
initial.R <- 0
initial.V <- 0 #Here, we are assuming none of our population is vaccinated at the start
of the simulation.
initial.S <- 1-initial.I-initial.R-initial.V

initial.values <- c(S = initial.S, I = initial.I, R = initial.R, V = initial.V)
parameters <- c(beta = 0.5, gamma = 0.1, theta = 0.001)
#Let's start with these parameters.
```

Now, exactly as before, we can run our model and plot its results. Notice that we have changed the 'func' argument in the *ode()* function to use our newly-defined model. We have also re-named our plot and assigned the color purple to the vaccinated class:

```
output<- as.data.frame(ode(func = SIRv, y = initial.values, parms = parameters, times =
times)) %>%
  pivot_longer(!time, names_to="state",values_to="proportion")
#View(output)

SIRv.plot <- ggplot(aes(x=time, y=proportion, color=state), data=output)+
  geom_line()+
  scale_y_continuous(limits=c(0,1), breaks=c(0,0.25,0.5,0.75,1))+
  scale_color_manual(values=c("red2","blue3","green4","purple"));SIRv.plot
```

Let's leave β at 0.5 and γ at 0.1 and vary the value of θ to be 0.001, 0.01, and 0.03. **Note that the R_0 is equal across all of these scenarios.** Again, draw each of the graphs below, labeling each with the value of θ. What pattern do you observe?

Now, set θ back to 0.001 (a pretty low rate of vaccination uptake), but vary the starting size of the vaccinated class to be 0.1, 0.4, and 0.8. Again, draw each graph below, labeling with the initial size of the vaccinated population. What pattern do you observe? What is this reduction in pathogen transmission due to the reduction of the susceptible class called? Hint: think about cows.

Get vaccinated!