Discussion 2: SIR model

Today we will be implementing the SIR model. But first, lets talk about discrete versus continuous time models. In discrete time models, time advances in discrete chunks of time (timesteps). These models are often appropriate for systems with a strong biological rhythm, such as populations with a single annual birth pulse, or very well defined life stages. Last week we did a quick implementation of a simple discrete model. Discrete time models are often easy to solve with for loops. Their equations typically return population size, rather than a derivative.

Continuous time ODE models use continuous rates of change for each state variable. Solving them analytically is often very hard or impossible, so we an ODE solver, which is computer code that approximates the analytical solution. You could still implement your own ODE solver using Euler's method, which is a relatively simple way to approximate differential equations, but using an ODE solver that has been implemented already is faster and more reliable.

The SIR model

Below is a traditional way to write the model.

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

1. Go over each equation and identify all of the state spaces and parameters.
2. In a new R script, install and load package deSolve. Then, use help(lsoda) to look at the documentation for the function lsoda and read carefully about the 4 parameters it takes: y, times, func, and parms. What are they?
3. Set up all of the starting values and parameters you will need, as well as the time points you will want to calculate values for.

```r
S0 =
I0 =
R0 =
state_vars = c(SS = S0, II = I0, RR = R0)

#Generate a  series of times at which you want the ODE solver to output population sizes
tseq <-

#Generate a vector of parameter values. This syntax gives the appropriate name to each entry in the vec
pars <- c(beta = , gamma = , mu = )
```

4. Write a function that computes the SIR model. This function will be your *func* parameter. It needs to get a time point, the state variables, and the parameters as arguments, and return a list with the derivative values. Use the template below if needed:

```r
SIR_system <- function(tseq, state_vars, pars){
    SS = state_vars[1]
    II =
    RR =
    beta = pars[1]
    gamma =
    mu =

    dS_dt =
    dI_dt =
    dR_dt =

    return(list(dS = dS_dt, dI = dI_dt, dR = dR_dt))
}
```

5. Use lsoda to calculate your population sizes, then plot S, I, and R versus time with continuous lines in distinct colors. Add a legend to your plot. You can use the code below as a guide.

```
output <- lsoda(c(S0, I0, R0), tseq, SIR_system, pars)

# Look at the format of the results using head()
head(output)
#The first column is time
#"1" is your first state variable (S), "2" is I, and "3" is R
```