Introduction to practical disease modeling: Exercise 2

Exercises with infection dynamics

---

1. Introducing herd dynamics

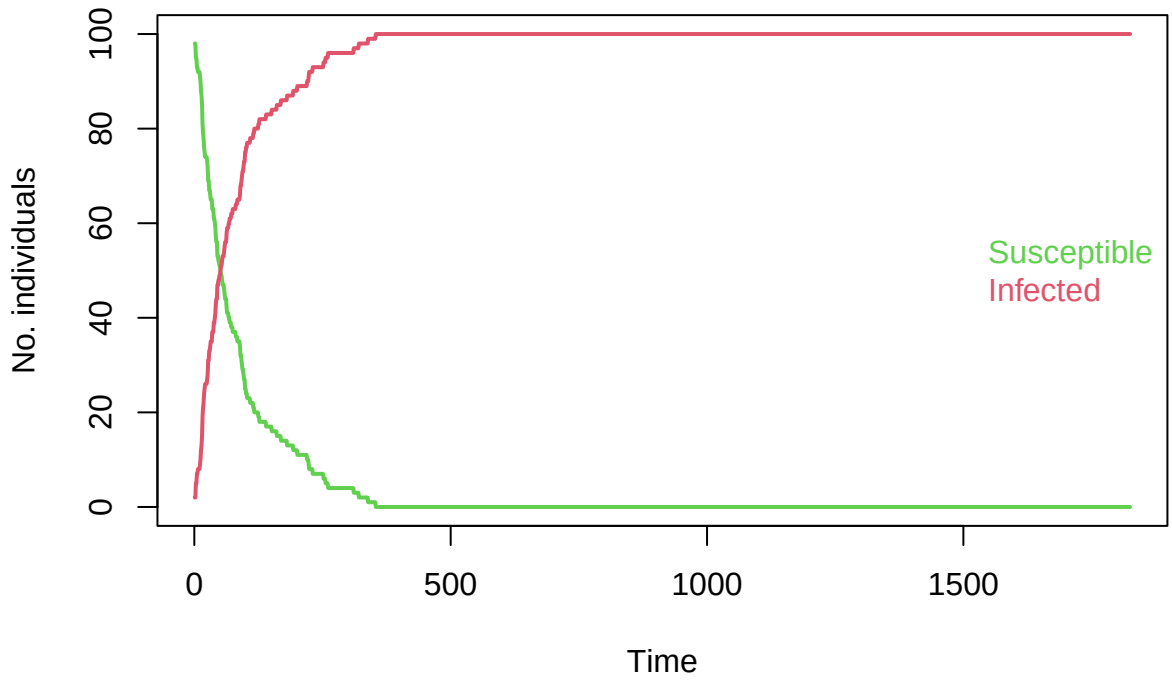
We use the same herd model as introduced earlier. This time we simulate a disease, that infects the cows at random. The disease is chronic, meaning that all susceptible cows can be infected, and will stay there. Thus, it is an *SI* model.

```
set.seed(250)

n_cows <- 100

# Create the farm:
farm <- data.frame(id =1:n_cows,
                   age=round(runif(n_cows,730,1642)),
                   infected=0)

# We start by initially infected one cow - the first one:
farm$infected[1] <- 1

# We set the probability of infection:
ProbInfection <- 0.01

# We want to simulate 5 years:
end_time <- 5 * 365

# Collect the mean age of the cows in herd over time:
age_collect <- numeric(end_time)

# Collect the number of susceptible and infected animals at each time point:
inf_collect <- data.frame(susceptible=rep(0,end_time), infected=rep(0, end_time))

for (t in seq_len(end_time))
{

    # Identify the cows that are not infected (susceptible):
    PotInf       <- which(farm$infected==0)
    # Draw randomly which of the cows that should be (newly) infected:
    NewInf       <- PotInf[rbinom(length(PotInf),1,prob=ProbInfection)==1]
    # If there are new infections, then we infect them:
    if(length(NewInf)>0) farm$infected[NewInf] <- 1

   # Add one day to the age of all the animals, for each simulated:
   farm$age <- farm$age + 1
```

```
   # Save the daily mean age of all cows:
   age_collect[t] <- mean(farm$age)

   # Save the number of susceptible and infected:
   sus <- length(farm$infected[farm$infected==0])
   inf <- length(farm$infected[farm$infected==1])
   inf_collect[t,1] <- sus
   inf_collect[t,2] <- inf
}
```

And we plot the course of infection dynamics:

```
plot(inf_collect$susceptible, type="l", lwd=2, col=3, xlab="Time", ylab="No. individuals", ylim=c(0,n_c
lines(inf_collect$infected, type="l", lwd=2, col=2)
legend("right", text.col=c(3,2), legend=c("Susceptible", "Infected"),
       bty="n")
```



The number of susceptible individuals decrease over time and the number of infected individuals increase.

2. Exercises

Hint: Modify the above code to solve the exercises.

A.

ProbInf is the equivalent of the beta parameter commonly used in simulation models. It is the probability that a new infection will occur.

Try to adjust the ProbInf up and down. Can you make the number of susceptible and infected cross each other right in the middle of the simulation? How does the stochastic element come into play here?

B.

Try to make the probability of infection dependent on the number of infected individuals, for example like: Beta * Number of infected / total number of individuals.

C.

Modify the model so that there is a third compartment, Recovered (R) that infected individuals can enter with a certain probability.

D.

A more realistic scenario is that the individuals recover after a fixed number of days, for example 10 days. Try to include that in the model. Hint: You will need to set a counter to keep track of how many days the infected individuals have before they can recover. Plot the output.

---

3. Bonus exercise

Modify the R6 model (bonus exercise from exercise 1) to include these infection dynamics. Include the Beta and Gamma parameters as arguments to the initialize method. Create a new method for infecting an animal, and another method for extracting the current total number of susceptible, infected and recovered animals.

You can also consider encapsulation of independent elements within the class itself i.e. separation of code to update the ages (and implement replacements) from code to update the disease statuses. Try to do this by having private methods called update_ages() and update_disease() where this code is separated, and then calling private$update_ages()$ and $private$update_disease() from within the public update() method. What is the benefit of using encapsulation like this?