Let’s code an individual based model in R

Type in the lines of code in the boxes sequentially into your R Studio script editor.

# Setting up initial conditions and parameters

Let’s start by creating a new R script. For simplicity, we’ll model only 20 individuals, starting with 19 susceptible people and 1 infected. We want to see how transmission is happening for 10 timesteps. This is how to set that up in R:

**initS <- 19**

**initI <- 1**

**no.of.timesteps <- 10**

# Build you model framework

Revisit the compartmental models described before and write down the diagram for an SEIR model:

Let’s consider a coding system to represent the possible state variables. For simplicity, let’s use the following:

Susceptible – 0

Exposed – 1

Infectious – 2

Recovered – 3

To initialize your population of 20 individuals, run the following code to store our simple population in a vector called ‘pop’.

**pop <- c(rep(0,initS),rep(2,initI))**

**pop**

> pop

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2

Now that you’ve created your population, you can simulate the system’s dynamics over 10 timesteps. But, wait! To help with visualization of the system behavior over time we need a place to store the sum of the individual state variables at each timestep.

# Creating a table to store the results of simulation

Let’s create a data frame called ‘sim.table’ that stores the sum of each state variable at each timestep:

**sim.table <- as.data.frame(matrix(NA, no.of.timesteps, 5))**

This is a table of 5 columns with ‘no.of.timesteps’ as the number of rows. Let’s change the column names to our state variable names:

**colnames(sim.table) <- c('WeekNo','S','E','I','R')**

**sim.table**

> sim.table

WeekNo S E I R

1 NA NA NA NA NA

2 NA NA NA NA NA

3 NA NA NA NA NA

4 NA NA NA NA NA

5 NA NA NA NA NA

6 NA NA NA NA NA

7 NA NA NA NA NA

8 NA NA NA NA NA

9 NA NA NA NA NA

10 NA NA NA NA NA

# Describe what can happen every timestep

We’ll try to describe all processes happening every timestep as they would in real life. Initially, our population has 20 (19 susceptible and 1 infectious) individuals. This is reflected in the initialization of pop.

To simulate the processes taking place in our system at each timestep we will use a ‘for’ loop:

**for(t in 1:no.of.timesteps)**

**{**

**#Simulate what happens at each timestep**

**}**

To efficiently store the updated summary of the state variables at each timestep:

**sim.table$WeekNo[t] <- t**

**sim.table$S[t] <- sum(pop==0)**

**sim.table$E[t] <- sum(pop==1)**

**sim.table$I[t] <- sum(pop==2)**

**sim.table$R[t] <- sum(pop==3)**

# Implementing transmission

For each individual, the probability of infection is a reflection of his/hers susceptibility and the number of infectious contacts he/she receives each timestep. Let’s say the number of potentially infectious contacts is given by parameter beta.

Create parameter beta and assign it a value between 2 and 5. If I have 1 contact per day, the chances that I will become infected is given by the probability that that 1 person I have contract with is presently infectious.

So, let’s create a variable lambda to represent the probability that a contact occurs with a presently infectious individual:

**beta <- 3**

**lambda <- beta\*sum(pop==2)/length(pop)**

Now let’s make it realistic by making it stochastic. In programming terms that means assigning some randomness to the contacts between individuals to assess whether or not an infection takes place. We start by casting a uniformly distributed random number for each individual.

**random.no <- runif(length(pop))**

**random.no**

> random.no

[1] 0.99474386 0.41812795 0.08883136 0.77035668 0.27272002 0.96196350 0.19896840 0.38292608 0.86681999 0.68108204

[11] 0.33380588 0.99075675 0.84305909 0.58943470 0.49186679 0.68111441 0.30561522 0.60752673 0.64994846 0.62527370

So, let’s implement the transmission process at the individual level. For this we need another ‘for’ loop.

# Let’s look at individuals one by one

**for(i in 1:length(pop))**

**{**

**#Simulate what happens to each individual**

**}**

Remember that these processes are all happening in a single timestep. So this second ‘for’ loop has to be within the first ‘for’ loop. Also note the difference in loop lengths.

If we take lambda to literally be the probability to become infectious in each timestep, we simply have to test whether the random number created for each individual is less than lambda. Obviously, you need to check the susceptibility condition. If an individual is susceptible (ie. state 0), we’ll compare its corresponding random no. with lambda. If it’s less than lambda, the individual gets infected and we change that individual’s state to 1 (first week of infection).

**if(pop[i]==0)**

**{**

**if(random.no[i]<lambda)**

**{**

**pop[i]<- 1**

**}**

**}**

Let’s say each timestep is equivalent to one week and that both latent and infectious periods are also one week. If individuals are latent or infectious, we’ll just move them on to the next state each timestep. Since we’ve coded the states S, E, I and R numerically as 0, 1, 2, and 3 respective, we just have to add 1 to move them to the next state.

**else if(pop[i]==1 | pop[i]==2)**

**{**

**pop[i] <- pop[i] + 1**

**}**

That’s it. You’ve just written your first individual-based model in R!

The full code:

**initS <- 19**

**initI <- 1**

**beta <- 1.5**

**gamma<-1**

**tau<-1**

**no.of.timesteps <- 10**

**pop <- c(rep(0,initS),rep(1,initI))**

**sim.table <- as.data.frame(matrix(NA, no.of.timesteps, 5))**

**colnames(sim.table) <- c('WeekNo','S','E','I','R')**

**for(t in 1:no.of.timesteps)**

**{**

**#Simulate what happens at each timestep**

**sim.table$WeekNo[t] <- t**

**sim.table$S[t] <- sum(pop==0)**

**sim.table$E[t] <- sum(pop==1)**

**sim.table$I[t] <- sum(pop==2)**

**sim.table$R[t] <- sum(pop==3)**

**lambda <- beta\*sum(pop==1| pop==2)/length(pop)**

**random.no <- runif(length(pop))**

**for(i in 1:length(pop))**

**{**

**#Simulate what happens to each individual**

**if(pop[i]==0)**

**{**

**if(random.no[i]<lambda)**

**{**

**pop[i] <- 1**

**}**

**}**

**else if(pop[i]==1 | pop[i]==2)**

**{**

**pop[i] <- pop[i] + 1**

**}**

**}**

**}**

**sim.table**

Exercises

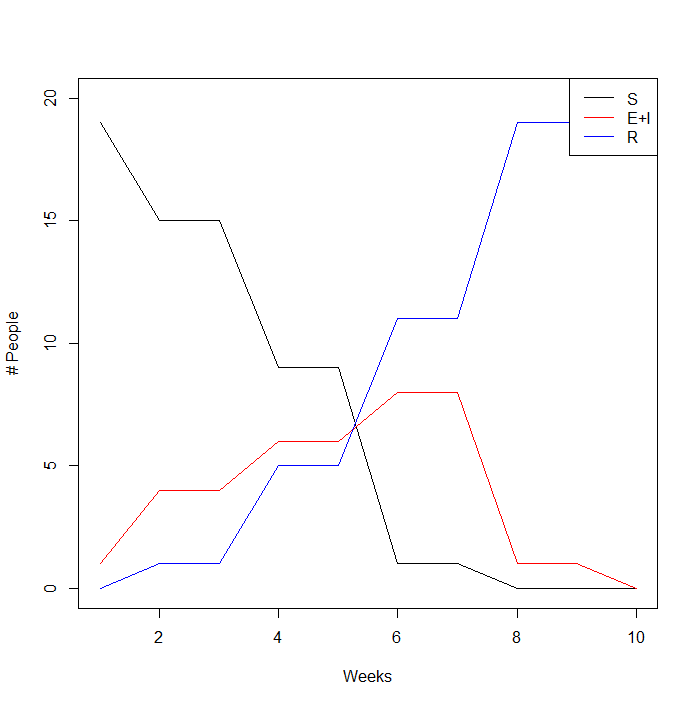
1. Create a graph from ‘sim.table’ similar to one below.

**plot(sim.table$WeekNo,sim.table$S,type='l',ylim=c(0,20),col='black',xlab="Weeks",ylab=("# People"))**

**lines(sim.table$WeekNo,sim.table$E+sim.table$I,col='red')**

**lines(sim.table$WeekNo,sim.table$R,col='blue')**

**legend("topright", legend = c("S","E+I","R"), lwd = 1, col=c("black","red","blue"))**



1. Add a column to your ‘sim.table’ which shows the value of lambda each week.
2. Run the script a few times. Are the outputs always same? How would you capture the variance observed in your runs?

…………………………………………………………………………………………..

1. Even for the simplest models, Individual-based simulations could be computationally expensive when you have a large population or when you run them for many timesteps. Experience it by changing parameters **no.of.timesteps** and **initS**.

…………………………………………………………………………………….

# A more general framework including recovery and immunity loss

Let’s implement latency, recovery from infection and no immunity as stochastic processes to better reflect malaria epidemiology.

As before, let’s start by generating a vector pop of individuals and an output table sim.table.

**initS <- 19**

**initI <- 1**

**beta <- 0.5**

**no.of.timesteps <- 100**

**pop <- c(rep(0,initS),rep(1,initI))**

**sim.table <- as.data.frame(matrix(NA, no.of.timesteps, 5))**

**colnames(sim.table) <- c('Day','S','E','I','R')**

Notice the change in number of timesteps. We are now using a daily timestep. Write the new function for the daily probability of infection lambda:

**for(t in 1:no.of.timesteps)**

**{**

**#Simulate what happens at each timestep**

**sim.table$Day[t] <- t**

**sim.table$S[t] <- sum(pop==0)**

**sim.table$E[t] <- sum(pop==1)**

**sim.table$I[t] <- sum(pop==2)**

**sim.table$R[t] <- sum(pop==3)**

**lambda <- beta\*sum(pop==2)/length(pop)**

**}**

We now have 3 state transitions: Infection, Latency and Recovery. Define parameters for the daily probabilities of becoming infectious (gamma) and recovering from infection (tau), assuming latency lasts an average of 5 days and people are usually infectious for 10 days.

**random.lambda <- runif(length(pop))**

**random.gamma <- runif(length(pop))**

**random.tau <- runif(length(pop))**

**for(i in 1:length(pop))**

**{**

**#Simulate what happens to each individual**

**## INFECTION ##**

**if(pop[i]==0)**

**{**

**if(random.lambda[i]<lambda)**

**{**

**pop[i] <- 1**

**}**

**}**

**## LATENCY ##**

**if(pop[i]==1)**

**{**

**if(random.gamma[i]<gamma)**

**{**

**pop[i] <- 2**

**}**

**}**

**## RECOVERY ##**

**if(pop[i]==2)**

**{**

**if(random.tau[i]<tau)**

**{**

**pop[i] <- 3**

**}**

**}**

**}**

Exercises:

1. You should now be able to easily include a transition from R to S to represent waning immunity. Write the Immunity loss code below the RECOVERY SECTION. Explore different values for the loss of immunity parameter *alpha.*
2. Plot all variables over time. Run the model several times to observe its stochasticity. What happens if you change the initial conditions?
3. Change the values of gamma and tau and elaborate on how the dynamics of the system depend on those parameters. Can you generate an outbreak even if R0 <1?

Full code for the last example:

**initS <- 19**

**initI <- 1**

**beta <- 0.5**

**gamma<-1/5**

**tau<-1/10**

**alpha<-1**

**no.of.timesteps <- 100**

**pop <- c(rep(0,initS),rep(1,initI))**

**sim.table <- as.data.frame(matrix(NA, no.of.timesteps, 5))**

**colnames(sim.table) <- c('Day','S','E','I','R')**

**for(t in 1:no.of.timesteps)**

**{**

**#Simulate what happens at each timestep**

**sim.table$Day[t] <- t**

**sim.table$S[t] <- sum(pop==0)**

**sim.table$E[t] <- sum(pop==1)**

**sim.table$I[t] <- sum(pop==2)**

**sim.table$R[t] <- sum(pop==3)**

**lambda <- beta\*sum(pop==2)/length(pop)**

**random.lambda <- runif(length(pop))**

**random.gamma <- runif(length(pop))**

**random.tau <- runif(length(pop))**

**random.alpha <- runif(length(pop))**

**for(i in 1:length(pop)){**

**#Simulate what happens to each individual**

**## INFECTION ##**

**if(pop[i]==0){**

**if(random.lambda[i]<lambda){**

**pop[i] <- 1**

**}**

**}**

**## LATENCY ##**

**if(pop[i]==1){**

**if(random.gamma[i]<gamma){**

**pop[i] <- 2**

**}**

**}**

**## RECOVERY ##**

**if(pop[i]==2){**

**if(random.tau[i]<tau){**

**pop[i] <- 3**

**}**

**}**

**## LOSS OF IMMUNITY ##**

**if(pop[i]==3){**

**if(random.alpha[i]<alpha){**

**pop[i] <- 0**

**}**

**}**

**}**

**}**

**sim.table**

**plot(sim.table$Day,sim.table$S,type='l',ylim=c(0,20),col='black',xlab="Weeks",ylab=("# People"))**

**lines(sim.table$Day,sim.table$I,col='red')**

**lines(sim.table$Day,sim.table$R,col='blue')**

**legend("topright", legend = c("S","I","R"), lwd = 1, col=c("black","red","blue"))**