Numerical solution of stochastic epidemiological models*

John M. Drake & Pejman Rohani

1 Introduction

He we expand our modeling toolkit to include methods for studying stochastic versions of the compartmental models studied in the last session. Particularly, we introduce the concept of birth-death processes and Gillespie's direct method for solving such processes.

2 Birth-death processes

First, we aim to understand the theory of birth-death processes in general. In the last session we studied a model that was deterministic, continuous in time, and continuous in the state variables S, I, and R. In this section, we retain the realistic assumption of continuous time, but also require that the number of susceptible, infected, and recovered individuals be integers (we are no longer modeling proportions). Of course, if the equation for dI/dt specified by the model should require that 1.8 individuals be infected in some interval of time, the only way to interpret such a model is that it is some average. If it is an average, then we are thinking that the process is probabilistic and evolves stochastically. This is, in fact, a perfectly reasonable assumption since not all infected individuals will give rise to exactly the same number of secondary infections nor recover after exactly the same interval of time. Since the random variation considered in this example is variation among individuals, such as scenario is referred to as *demographic stochsaticity*, which is a kind of *intrinsic noise*.

At this point, there are still a number of different directions we could go. We will make two assumptions that, together with the assumption of demographic stochasticity, uniquely determine a whole class of models. First, we assume that the epidemic is a *Markov chain*. A Markov chain is a stochastic process with the property that the future state of the system is dependent only on the present state of the system and conditionally independent of all past states. This is known as the *memoryless property*. Second, we assume that the changes in the state variables (increments and decrements) are independent. This implies that they occur one at a time. That is, two individuals cannot simultaneously undergo a "transition" (a change in state, i.e., birth, death, conversion between classes, etc.). Any time an individual undergoes such a transition, we will call it an "event". Accordingly, this kind of modeling is sometimes referred to as "event-driven simulation". For historical reasons, the continuous time Markov chain with increments and decrements of one is known as a birth-death process. (In general, a Markov chain with integer-valued increments and decrements is known as a *jump process*.)

[Note: An aside about birth-death processes is that notation varies considerably from author to author, even though the authors are referring to exactly the same stochastic process. Particularly, the simulational literature uses a notation borrowed from chemistry, *e.g.*, $S \xrightarrow{\mu S} S-1$, probably because the algorithms that are commonly used to simulate birth-death processes were developed in the context of

chemical kinetics. Probabilists, by contrast, often use the generating function notation and scientists that come to birth-death processes from a background in statistical mechanics represent the process using the *Forward Kolmogorov Equation* or *Fokker-Planck Equation* (a partial differential equation). Textbooks in epidemiology differ, too. The point is that once you learn to "read" the different notations, they are all saying the same thing, *i.e.*, that changes in the state variables occur according to such and such rates. It's these rates that are the basis of simulation modeling.]

Simple stochastic SI epidemic

To illustrate the approach, we'll start with the simple closed stochastic SI epidemic. Because the population is closed (no births, deaths, or migration) we represent total population size as a constant N. Because we are considering abundances, not proportions, we adopt the convention used in the lecture portion of respresenting susceptibles, infecteds, and recovereds by X, Y, and Z and denote the initial number of infected individuals by Y_0 accordingly. This gives the initial number of susceptible individuals $X_0 = N - Y_0$. By analogy to our deterministic model, we want the average rate at which susceptibles individually become infectious (the force of infection) to be $\beta \frac{Y}{N}$ and the average rate at which the population as a whole converts from susceptible to infectious to be $\beta \frac{Y}{N} S$. That is, at average rate $\beta \frac{Y}{N} X$ the value of X is decremented by one and the value of Y is incremented by one. But when do these increments and decrements occur? To answer this, we turn to our assumption that the epidemic process is Markovian.

If we can determine what the sequence of "inter-event times" is, then we have fully specified the trajectory of the epidemic, for we know that at each of those times the number of susceptibles decreases by one and the number of infecteds increases by one. So, what are the inter-event times? The memoryless property of the continuous time Markov chain requires that the time between events is independent of the time between any other set of events, and, moreover, that if we were to investigate the process at any point in time between events the time to the next event would be independent of the time elapsed since the previous event. This defines the birth-death process as a kind of *Poisson process*. There is only one distribution for the inter-event times that has this property, the exponential distribution. Since we know how to simulate random variables (in this case we use the function `rexp`) we just simulate the sequence of event times and make our increments and decrements accordingly. This approach is known as *Gillespie's direct method*.

All that remains, then, is to relate the rate at which our process is happening to the generation of exponential random numbers. An exponential distribution is defined by a single parameter, although the formula may be written in different ways. The parameterization assumed by R conveniently assumes the distribution is expressed as a Poisson process, *i.e.* the argument is itself expected to be the rate. Thus, we can simulate the inter-event time and update the state variables using one function.

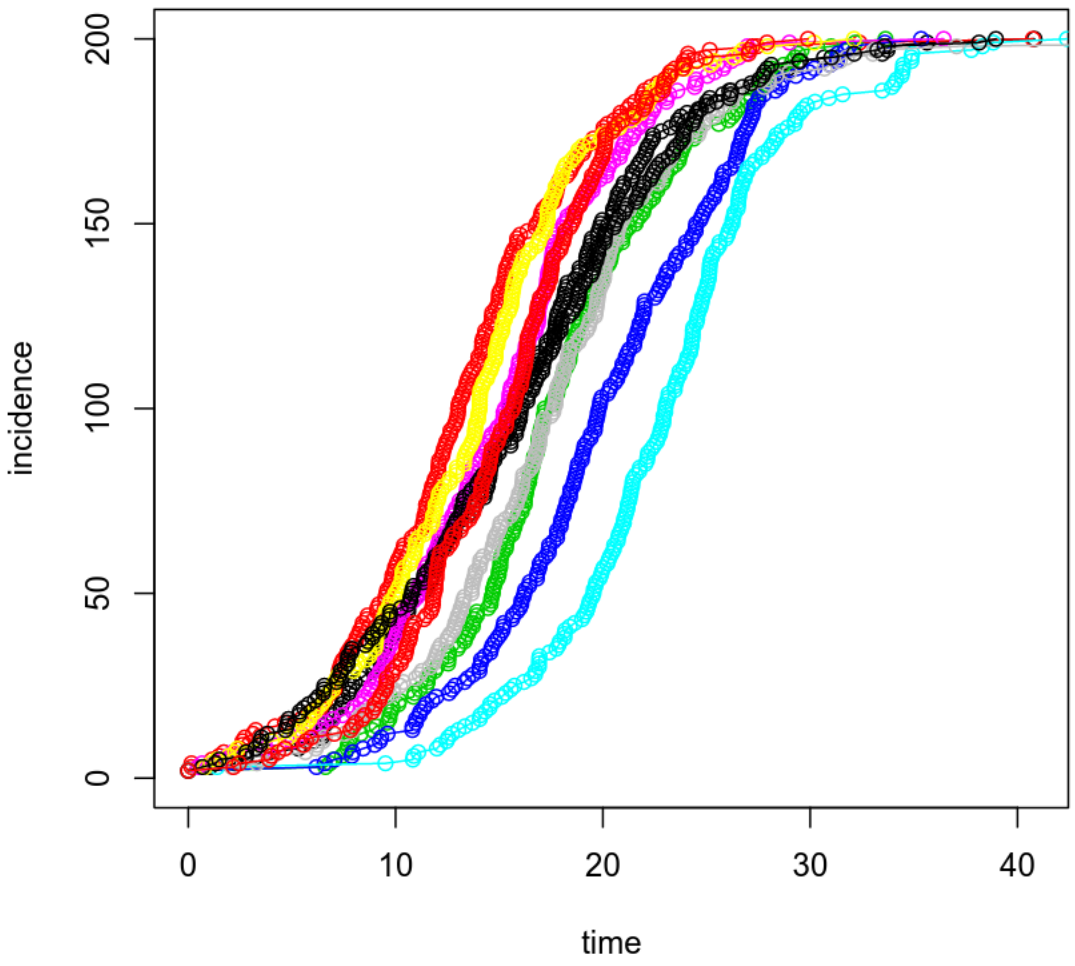
```
> SI.onestep <- function (x, params) {     #function for one step of the stochastic SI epidemic
+    X <- x[2]                             #the second element of x is number of susceptibles X
+    Y <- x[3]                             #the third element of x is number of infecteds Y
+    with(
+      as.list(params),
+      {
+        new.Y <- Y+1                      #whenever an event occurs we increase infecteds by 1...
+        new.X <- X-1                      #and descrease susceptibles by 1
+        tau <- rexp(n=1,rate=params$beta*X*Y/(X+Y))  #exponential random time to next event
+        c(tau=tau,X=new.X,Y=new.Y)        #store result
+        }
+      )
+ }
```

Now, we write a loop to iterate this simulation routine.

```
> SI.model <- function (x, params, nstep) { #function to iterate the stochastic SI for nstep events
+   output <- array(dim=c(nstep+1,3))       #set up an array to store all the results
+   colnames(output) <- c("time","X","Y")   #name the variables in the array
+   output[1,] <- x                         #the first record of the array is the initial condition
+   for (k in 1:nstep) {                    #iterate the model for nstep events
+     output[k+1,] <- x <- SI.onestep(x,params) #update x and store result
+   }
+   output                                  #return output
+ }
```

Using the same value of β as before (but remember there is no recovery here), we run some simulations and plot. Note that because time is continuous, we don't actually know how many events will occur in some specified period of time. Instead, we save some pre-set number of "events". Additionally, notice how the function `cumsum` is used to add up the inter-event times to give a time series.

```
> set.seed(38499583)                        #set the random seed so results are repeatable
> nsims <- 10                               #number of simulations to run
> pop.size <- 200                           #total size of the population
> Y0 <- 2                                   #initial number infected
> nstep <- pop.size-Y0                      #how many steps to run? until everyone infected
> xstart <- c(time=0,X=(pop.size-Y0),Y=Y0)  #initial conditions
> params <- list(beta=0.3) #parameters
> data <- vector(mode='list',length=nsims)  #create a list called ``data'' to store all runs
> for (k in 1:nsims) {                      #simulate k different runs
+   data[[k]] <- as.data.frame(SI.model(xstart,params,nstep))  #main simulation step
+   data[[k]]$cum.time <- cumsum(data[[k]]$time) #calculates the running sum of inter-event intervals
+ }
> max.y<-max(data[[1]]$cum.time)               #find the maximum time any simulation ran (to set x axis)
> plot(c(0,pop.size),c(0,pop.size),type='n',xlab='time',ylab='incidence',xlim=c(0,max.y)) #set up plot
> for (k in 1:nsims) {                      #loop over each simulation...
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')  #to plot
+ }
```



[Note: This program uses a few R techniques, such as listing with the command `data[[k]]`, that were not covered previously. In general, lists can be created with the function `list` in place of the combine function `c` that we used previously. The referencing is similar to referencing with vectors and matrices, but uses double square brackets. If you're curious, play around with this facility.]

Exercise 1. Simulate the stochastic SI model using Gillespie's direct method. Experiment with the initial number of infecteds (Y_0) and with the total population size (N). What effects do these have on the predictability of the epidemic? What effects do these have on the variability of the final outbreak size?

Extending the SI model

We can extend the simple SI model to an arbitrary number of compartments. For concreteness, we study a stochastic version of the SIR model from the first section. The main difference between the SI model and the SIR model is that in the SIR model there's more than one kind of event that can

occur. (This is true of the SI model with births and deaths, too, but we didn't look at that). That means we need to account for two things: (1) Our determination of the next event time has to take into consideration the multiple processes that are occurring simultaneously, and (2) Once we determine what time the event occurs we have to determine what type of event it is. Since the transition processes are independent we can calculate a "total rate" as the sum of the individual rates. That is, the rate at which the whole system is evolving is the sum of the rates of the individual processes which are the absolute values of the different transition terms in the model. For example, the transitions associated with the susceptible class are transition to the infected class (at rate $\beta\frac{Y}{N}X$), births (at rate μN), and deaths (at rate μX). Thus, the "total rate" for the susceptible class is $\beta\frac{Y}{N}X + \mu N + \mu X$. Our total rate for the whole proceess will include the rates for the Y and Z classes as well. The next step in the multi-dimensional Gillespie simulation is to determine which event occurs. In the long run each event much occur at its specific rate. This means that we can just randomly choose which event occurs so long as we do it in a weighted way such that each transition is selected in proportion to its contribution to the total rate.

First we define our one-step function. Notice the ordering of the `if` statements.

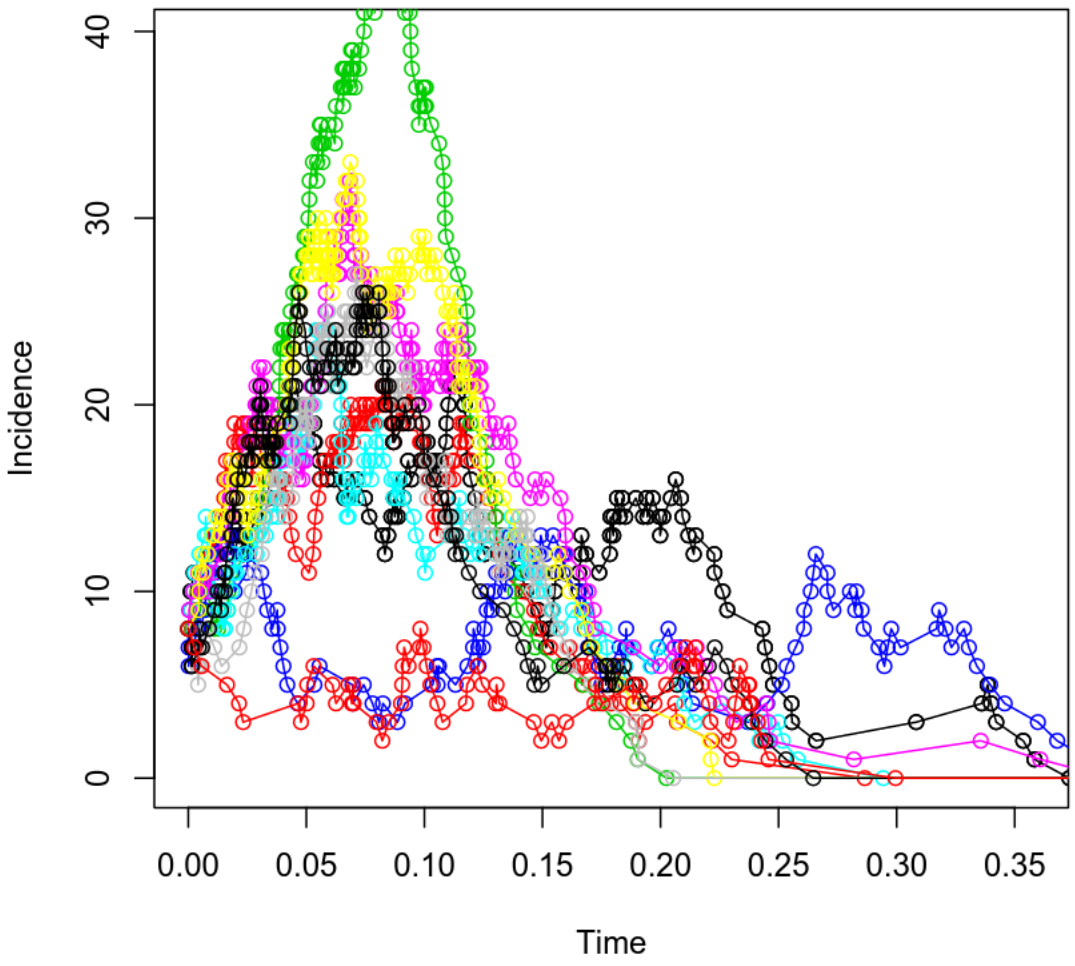
```
> SIR.onestep <- function (x, params) {  #function to calculate one step of stochastic SIR
+   X <- x[2]                            #local variable for susceptibles
+   Y <- x[3]                            #local variable for infecteds
+   Z <- x[4]                            #local variable for recovereds
+   N <- X+Y+Z                           #total population size (subject to demographic change)
+   with(                                #use with as in deterministic model to simplify code
+        as.list(params),
+        {
+          rates <- c(mu*N, beta*X*Y/N, mu*X, mu*Y, gamma*Y, mu*Z)
+          changes <- matrix(c( 1, 0, 0,
+                              -1, 1, 0,
+                              -1, 0, 0,
+                               0,-1, 0,
+                               0,-1, 1,
+                               0, 0,-1),
+                            ncol=3, byrow=TRUE)
+          tau <- rexp(n=1,rate=sum(rates))   # exponential waiting time
+          U <- runif(1)       #uniform random deviate
+          m <- min(which(cumsum(rates)>=U*sum(rates)))
+          x <- x[2:4] + changes[m,]
+          return(out <- c(tau, x))
+        }
+        )
+ }
```

As before, we set parameters and loop through the process

```
> SIR.model <- function (x, params, nstep) {  #function to simulate stochastic SIR
+   output <- array(dim=c(nstep+1,4))         #set up array to store results
+   colnames(output) <- c("time","X","Y","Z") #name variables
+   output[1,] <- x                           #first record of output is initial condition
+   for (k in 1:nstep) {                      #iterate for nstep steps
+     output[k+1,] <- x <- SIR.onestep(x,params)
+   }
+   output                                    #return output
+ }
```

Now let's repeat for 10 runs and plot.

```
> set.seed(38499583)                #set seed
> nsims <- 10                       #number of simulations
> pop.size <- 100                   #total population size
> Y0 <- 8                           #initial number infected
> X0 <- round(0.9*pop.size)        #initial number suscepitlble (~90% of population)
> nstep <- 1600                     #number of events to simulate
> xstart <- c(time=0,X=X0,Y=Y0,Z=pop.size-X0-Y0) #initial conditions
> params <- list(mu=0.00001,beta=60,gamma=365/13) #parameters
> data <- vector(mode='list',length=nsims) #initialize list to store the output
> for (k in 1:nsims) {              #simulate nsims times
+   data[[k]] <- as.data.frame(SIR.model(xstart,params,nstep))
+   data[[k]]$cum.time <- cumsum(data[[k]]$time)
+ }
> max.time<-data[[1]]$cum.time[max(which(data[[1]]$Y>0))] #maximum time in first simulation
> max.y<-1.8*max(data[[1]]$Y)       #find max infected in run 1 and increase by 80% for plot
> plot(Y~cum.time,data=data[[1]],xlab='Time',ylab='Incidence',col=1,xlim=c(0,max.time),ylim=c(0,max.y))
> for (k in 1:nsims) {              #add multiple epidemics to plot
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')
+ }
```



Exercise 2. Simulate the stochastic SIR model using Gillespie's direct method. As before, experiment with the initial number of infecteds (Y_0) and with the total population size (N). What effects do these have on the predictability of the epidemic?

Additional issues

The birth-death framework is a popular approach to stochastic epidemic modeling. It has some important limitations, however. Overcoming these limitations is an area of active research:

- For even moderately large systems, Gillespie's direct method is very slow. Approximations are available (e.g., the so-called "tau-leap method") to speed up the calculations, though none is as easy to program as the direct method. Some of these may be called directly using the R package `GillespieSSA`.
- The Markovian assumption entails that the inter-event times of the birth-death process are expo-

nentially distributed. This is a biologically unrealistic assumption that has considerable impact on the variance of the process. Non-Markovian (i.e., non-memoryless) processes are the solution, but these come at the cost of additional conceptual and computational complexity.