Simulating epidemics in R

John M. Drake

June 6, 2010

Contents

1 Introduction **1**

2 The SIR model **1**

3 Stochastic model **5**

4 Fitting continuous-time models to data: trajectory matching **12**

1 Introduction

This workshop will introduce techniques for estimating the parameters of epidemiological models. At the same time, we'll cover ways of evaluating models (e.g., overall model fit) and parts of models (e.g., hypothesis tests based on parameters or combinations of parameters). Typically, we assume (that is, we pretend we know to be true) that the model takes a certain structure. In this module, we introduce some of these structures. This introduction serves several purposes. First, by first looking at some specific models we will start the estimation part of the workshop with a shared conceptual baseline. Second, the models we look at here are fundamental and relatively general and therefore readily extended for your own purposes in the future. Third, we introduce a number of numerical tools that are useful for studying epidemiological systems. And, finally, by simulating these systems we produce some datasets in which the dynamical data-generating process is truly known. By trying out our estimation techniques on these known processes, we can study how well the various techniques perform under different circumstances.

2 The SIR model

The simplest place to start is with the classical SIR model. This model expands the SI model you studied yesterday to include a class of "recovered" individuals, which are assumed to be immune. The simply keeps track of how many individuals are in each class: individuals that leave one class must enter another class (this is the *conservation property*), with exceptions for births and deaths. As with the SI

model, the state variables change according to a system of differential equations:

$$\begin{aligned}\frac{dS}{dt} &= \mu\,N - \lambda(I,t)\,S - \mu\,S\\ \frac{dI}{dt} &= \lambda(I,t)\,S - \gamma\,I - \mu\,I\\ \frac{dR}{dt} &= \gamma\,I - \mu\,R\end{aligned}$$

Here, μ is the birth and death rates (which we assume to be equal), N is the host population size, and γ the recovery rate. The only interesting bit is the force of infection $\lambda(I,t)$. We'll assume that it has the so-called *frequency dependent* form

$$\lambda(I,t) = \beta(t)\,\frac{I}{N}$$

so that the risk of infection a susceptible faces is proportional to the fraction of the population that is infectious. Notice that we allow for the possibility of a contact rate, β, that varies in time. In this model, S, I, and R may be interpreted either as proportions of the population (if $N = 1$) or abundances (if $N > 1$).

Like many epidemiological models, one can't solve the SIR equations explicitly. Rather, to find the trajectory of a continuous-time model such as the SIR, we must integrate those ordinary differential equations (ODEs) numerically. What we mean by this is that we use a computer algorithm to approximate the solution. In general, this can be a tricky business. Fortunately, this is a well studied problem in numerical analysis and (when the equations are smooth, well-behaved functions of a relatively small number of variables) standard numerical integration schemes are available to approximate the integral with arbitrary precision. Particularly, R has a very sophisticated ODE solver facility which for many problems will give highly accurate solutions. To use the numerical integration package, we must load the package

```
> require(deSolve)
```

[Note: If you get a warning that the package was not loaded, check to make sure it is installed on your computer.]

The ODE solver needs to know the right-hand sides of the ODE. We give it this information as a function:

```
> sir.model <- function (t, x, params) {
+   S <- x[1]
+   I <- x[2]
+   R <- x[3]
+   with(
+        as.list(params),
+        {
+          dS <- mu*(N-S)-beta*S*I/N
+          dI <- beta*S*I/N-(mu+gamma)*I
+          dR <- gamma*I-mu*R
+          res <- c(dS,dI,dR)
+          list(res)
+        }
+        )
+ }
```

Notice that here, we've assumed β is constant.

[Note: In case the `with` function is unfamiliar, it serves here to make the parameters `params` available to the expressions in the brackets, *as if they were variables*. One could achieve the same effect by, for example, `dS <- params["mu"]*(params["N"]-S)-params["beta"]*S*I/params["N"]` and so on.]

We'll now define the times at which we want solutions, assign some values to the parameters, and specify the *initial conditions*, i.e., the values of the state variables S, I, and R at the beginning of the simulation:

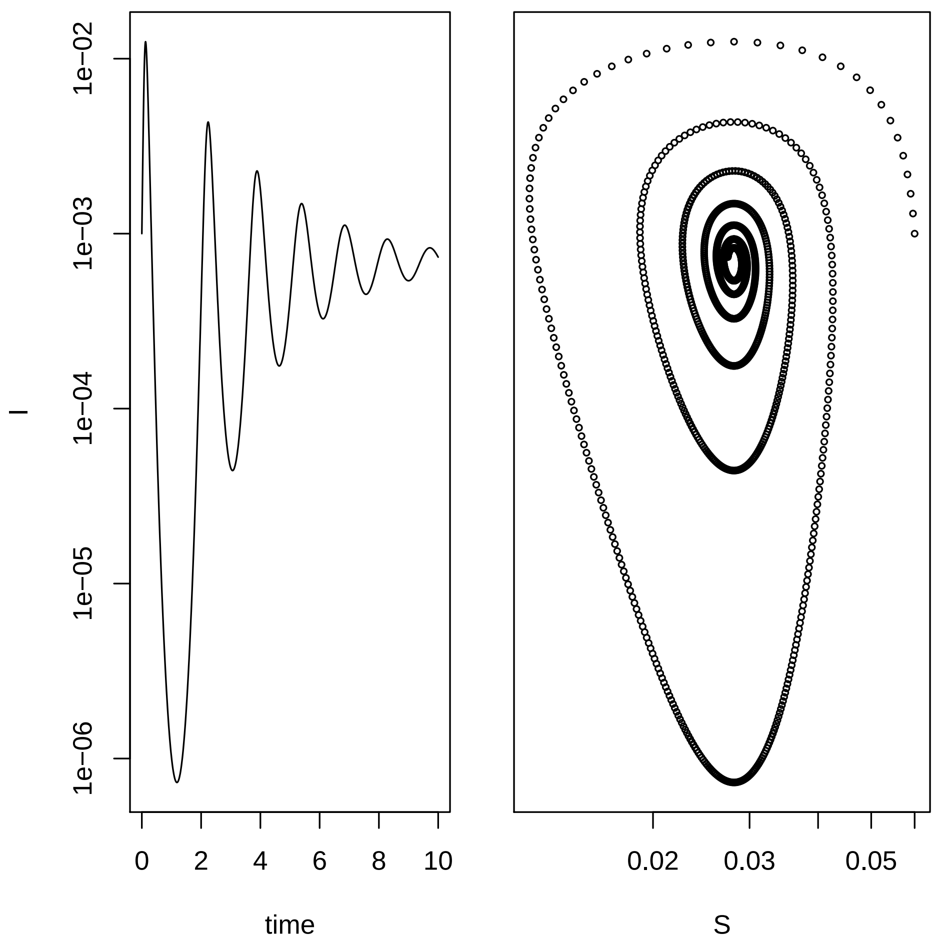
```
> times <- seq(0,10,by=1/120)
> params <- c(mu=1/50,N=1,beta=1000,gamma=365/13)
> xstart <- c(S=0.06,I=0.001,R=0.939)
```

Now we can simulate a model trajectory with the `lsoda` command:

```
> out <- as.data.frame(lsoda(xstart,times,sir.model,params))
```

and plot the results

```
> op <- par(fig=c(0,0.5,0,1),mar=c(4,4,1,1))
> plot(I~time,data=out,type='l',log='y')
> par(fig=c(0.5,1,0,1),mar=c(4,1,1,1),new=T)
> plot(I~S,data=out,type='p',log='xy',yaxt='n',xlab='S',cex=0.5)
> par(op)
```



Exercise 1. Explore the dynamics of the system for different values of the β and μ parameters by simulating and plotting trajectories as time series and in phase space (e.g., I vs. S).

***Exercise 2.** Modify the codes given to study the dynamics of an SEIR model.

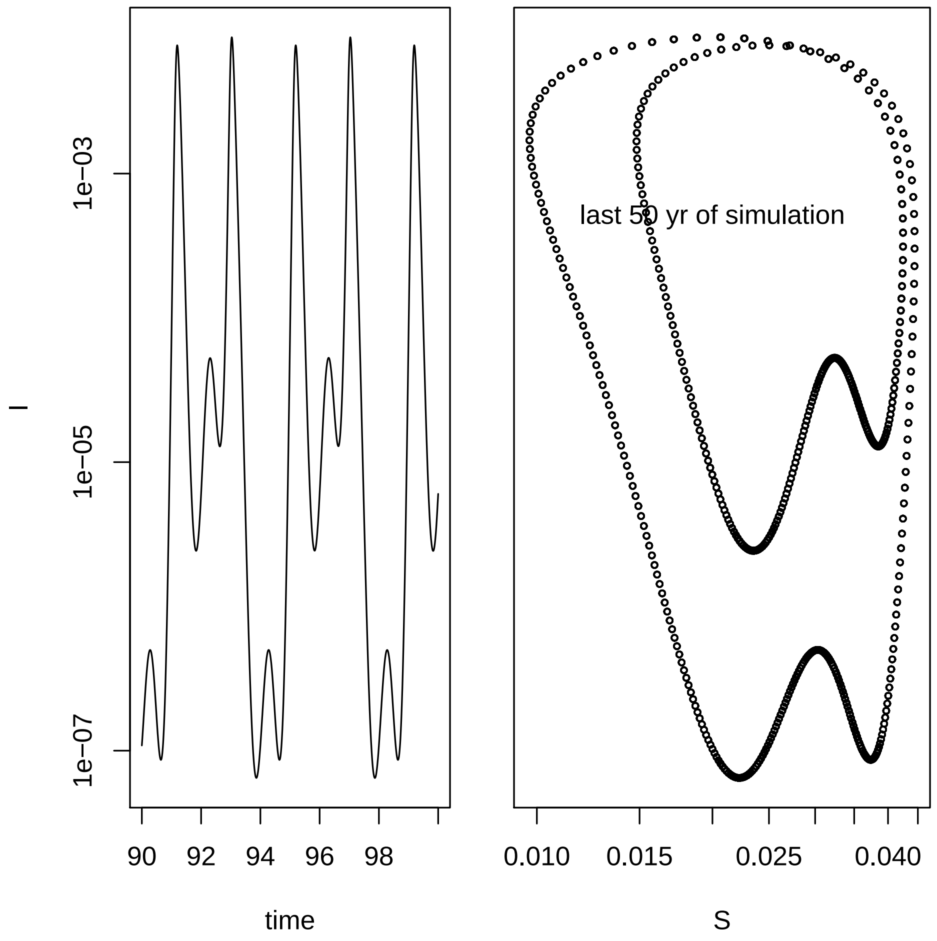
Seasonality

The simple SIR model always predicts damped oscillations towards an equilibrium (or pathogen extinction if R_0 is too small). This is at odds with the recurrent outbreaks seen in many real pathogens. Sustained oscillations require some additional drivers in the model. An important driver in childhood infections of humans (e.g., measles) is seasonality in contact rates because of aggregation of children the during school term. We can analyze the consequences of this by assuming sinusoidal forcing on β according to $\beta(t) = \beta_0\,(1 + \beta_1 \cos(2\,\pi\,t))$. Translating this into R:

```
> seasonal.sir.model <- function (t, x, params) {
+   with(
+        as.list(c(x,params)),
+        {
+          beta <- beta0*(1+beta1*cos(2*pi*t))
+          dS <- mu*(N-S)-beta*S*I/N
+          dI <- beta*S*I/N-(mu+gamma)*I
+          dR <- gamma*I-mu*R
+          res <- c(dS,dI,dR)
+          list(res)
+        }
+        )
+ }
```

We'll simulate as before, with the same mean contact rate, β_0 as before, but now with a fairly strong amplitude of seasonality, β_1.

```
> times <- seq(0,100,by=1/120)
> params <- c(mu=1/50,N=1,beta0=1000,beta1=0.4,gamma=365/13)
> xstart <- c(S=0.06,I=0.001,R=0.939)
> out <- as.data.frame(lsoda(xstart,times,seasonal.sir.model,params,rtol=1e-12,hmax=1/120))
> op <- par(fig=c(0,0.5,0,1),mar=c(4,4,1,1))
> plot(I~time,data=out,type='l',log='y',subset=time>=90)
> par(fig=c(0.5,1,0,1),mar=c(4,1,1,1),new=T)
> plot(I~S,data=out,type='p',log='xy',subset=time>=50,yaxt='n',xlab='S',cex=0.5)
> text(0.02,0.0005,"last 50 yr of simulation")
> par(op)
```



Exercise 3. Explore the effects of changing amplitude of seasonality, β_1 on the dynamics of this model. Be careful to distinguish between transient and asymptotic dynamics.

So far today we have used one model (the SIR model) to introduce two concepts (frequency-dependent transmission and seasonal forcing) and one technique (numerical solution of ODEs).

3 Stochastic model

In this next section we extend our toolbox to include a stochastic model. Why do we need a stochastic model? The models introduced in the preceding section were both systems of deterministic differential equations. Although useful for some puposes, these models makes two very restrictive assumptions. First, they assume that the change in the number of susceptible and infectious individuals in the population happens continuously. In fact, since the population is finite and the class values (S and I) are categorical, changes in the state variables in reality occur in jumps. This is the problem of assuming a continuous state space. A more realistic model is one that is restricted to the integers. To relax this assumption we could easily integerize this model by asserting that each infected individual gives rise to β new infected individuals in each time step and numerically iterating the model forward in time. But, this just reinforces another unrealistic assumption, which is that each infected individual gives rise to precisely the same number of secondary infections, which arise simultaneously after precisely the same amount of time. This deterministic assumption clearly is not biologically realistic. A solution that solves both problems is to use a model that is both naturally restricted to the integers and in which the number of secondary infections is a random variable. Such a model is said to exhibit *demographic stochasticity*. Demographic stochasticity is a kind of *process noise* (to be contrasted with *sampling error*.

In the first section, we discussed a model that was deterministic, continuous in time, and continuous in the state variables S, I, and R. Here, we relax the assumptions of determinism and continuous state-space.

At this point, there are still a number of different directions we could go. We will make two additional 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 defined as 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) occur one at a time. That is, we cannot have two individuals simultaneously undergoing a transition, where "transition" refers to any change in the state variables (birth, death, conversion between classes, etc.). 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*.)

Since this terminology is well entrenched, we'll continue using it. Thus, when we refer to the "birth of a susceptible" we really mean a demographic birth. But, when we refer to the "death of a susceptible", this could be a demographic death or it could be the transition of a individual from the susceptible class to the infected class. In this general terminology, "birth" means "add one to the state variable" and "death" means "subtract one from the state variable". As with the deterministic SIR model, transitions will be conserved, but (demographic) births and (demographic) deaths need not be conserved.

[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 computational 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 ecology and 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 a simple closed stochastic SI epidemic. Because the population is closed (no births, deaths, or migration) we represent total population size as a constant N. We denote the initial number of infected individuals by I_0 and have the initial number of susceptible individuals $S_0 = N - I_0$. By analogy to our deterministic model, we want the average rate at susceptibles individually become infectious (the force of infection) to be $\beta \frac{I}{N}$ and the average rate at which the population as a whole converts from susceptible to infectious to be $\beta \frac{I}{N} S$. That is, at average rate $\beta \frac{I}{N} S$ the value of S is decremented by one and the value of I 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 entails that the time between events is independent of the time between any other set of events, and, moreover, if we were to investigate the process at any point in time between events that 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 in terms of a Poisson process, such that the argument is itself already the rate. Thus, we simulate the inter-event time and update the state variables using one function

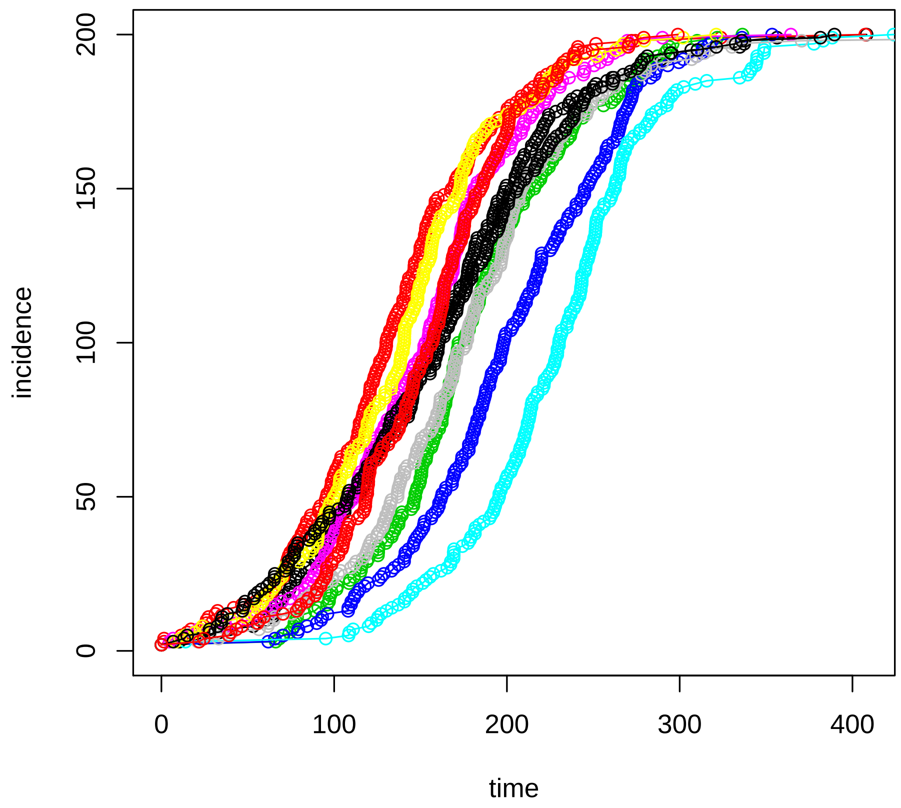
```
> birth.death.onestep <- function (x, params) {
+    S <- x[2]
+    I <- x[3]
+    beta <- params['beta']
+    new.I <- I+1
+    new.S <- S-1
+    new.t <- rexp(n=1,rate=beta*S*I/(S+I))
+    c(tau=new.t,S=new.S,I=new.I)
+ }
```

As in the examples above, we write a loop to iterate this simulation routine. 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".

```
> birth.death.model <- function (x, params, nstep) {
+   X <- array(dim=c(nstep+1,3))
+   colnames(X) <- c("time","S","I")
+   X[1,] <- x
+   for (k in 1:nstep) {
+     X[k+1,] <- x <- birth.death.onestep(x,params)
+   }
+   X
+ }
```

Using the same parameters as before, we run some simulations and plot. Notice the function `cumsum` is used to add up the inter-event times to give a time series.

```
> set.seed(38499583)
> nsims <- 10
> pop.size <- 200
> I0 <- 2
> nstep <- pop.size-I0
> xstart <- c(time=0,S=(pop.size-I0),I=I0)
> params <- c(beta=3e-2)
> x <- vector(mode='list',length=nsims)
> for (k in 1:nsims) {
+   x[[k]] <- as.data.frame(birth.death.model(xstart,params,nstep))
+   x[[k]]$cum.time <- cumsum(x[[k]]$time)
+ }
> max.y<-max(x[[1]]$cum.time)
> plot(c(0,pop.size),c(0,pop.size),type='n',xlab='time',ylab='incidence',xlim=c(0,max.y))
> for (k in 1:nsims) {
+   lines(I~cum.time,data=x[[k]],col=k,type='o')
+ }
```



Exercise 4. Simulate the stochastic SI model using Gillespie's direct method. Experiment with the initial number of infecteds (I_0) and with the total population size (N). What effects do these have on the predictability of the epidemic?

Extending the SI model

In this section we 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. (Actually, 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 $\beta \frac{I}{N} S$), births (at rate μN), and deaths (at rate μS). Thus, the "total rate" for the susceptible class is $\beta \frac{I}{N} S + \mu N + \mu S$. Our total rate for the whole proceess will include the rates for the I and R 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.

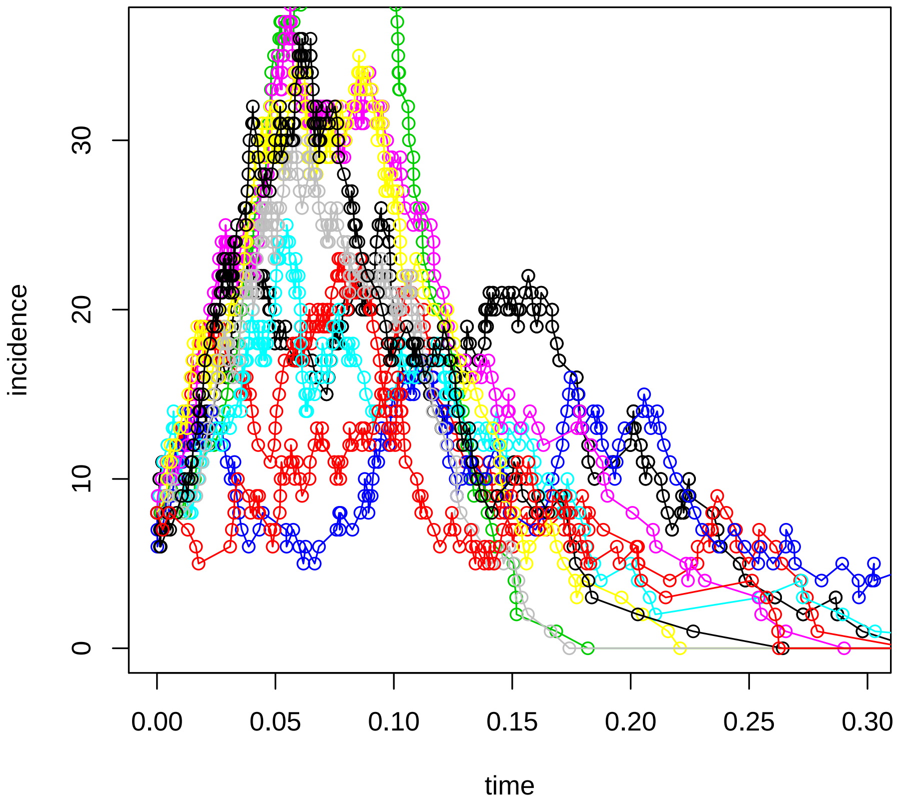
```
> sir.birth.death.onestep <- function (x, params) {
+   S <- x[2]
+   I <- x[3]
+   R <- x[4]
+   N <- S+I+R
+
+   with(
+        as.list(params),
+        {
+          total.rate <- mu*N+beta*S*I/N+mu*S+mu*I+gamma*I+mu*R
+          new.t <- rexp(n=1,rate=total.rate)
+          new.sir <- c(S,I,R)
+
+          U <- runif(1)
+          new.sir<-c(S,I,R-1) #death of recovered
+          if (U<=(mu*N+beta*S*I/N+mu*S+gamma*I+mu*I)/total.rate) new.sir<-c(S,I-1,R)
+                         #death of infected
+          if (U<=(mu*N+beta*S*I/N+mu*S+gamma*I)/total.rate) new.sir<-c(S,I-1,R+1)
+                         #recovery of infected
+          if (U<=(mu*N+beta*S*I/N+mu*S)/total.rate) new.sir<-c(S-1,I,R)    #death of a susceptible
+          if (U<=(mu*N+beta*S*I/N)/total.rate) new.sir<-c(S-1,I+1,R)       #transmission event
+          if (U<=(mu*N/total.rate)) new.sir<-c(S+1, I, R)                  #birth of susceptible
+          c(new.t,new.sir)
+        }
+        )
+ }
```

As before, we set parameters and loop through the process

```
> sir.birth.death.model <- function (x, params, nstep) {
+   X <- array(dim=c(nstep+1,4))
+   colnames(X) <- c("time","S","I","R")
+   X[1,] <- x
+   for (k in 1:nstep) {
+     X[k+1,] <- x <- sir.birth.death.onestep(x,params)
+   }
+   X
+ }
```

Now let's repeat for 10 runs and plot.

```
> set.seed(38499583)
> nsims <- 10
> pop.size <- 100
> I0 <- 8
> S0 <- round(0.98*pop.size)
> nstep <- 1600
> xstart <- c(time=0,S=S0,I=I0,R=pop.size-I0-S0)
> params <- c(mu=0.00001,beta=60,gamma=365/13)
> x <- vector(mode='list',length=nsims)
> for (k in 1:nsims) {
+   x[[k]] <- as.data.frame(sir.birth.death.model(xstart,params,nstep))
+   x[[k]]$cum.time <- cumsum(x[[k]]$time)
+ }
> max.time<-x[[1]]$cum.time[max(which(x[[1]]$I>0))]
> max.y<-1.4*max(x[[1]]$I)
> plot(I~cum.time,data=x[[1]],xlab='time',ylab='incidence',col=1,
+      xlim=c(0,max.time),ylim=c(0,max.y))
> for (k in 1:nsims) {
+   lines(I~cum.time,data=x[[k]],col=k,type='o')
+ }
```



Exercise 5. Simulate the stochastic SIR model using Gillespie's direct method. As before, experiment

with the initial number of infecteds (I_0) and with the total population size (N). What effects do these have on the predictability of the epidemic? How would you adapt the model to include seasonality?

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.
- The Markovian assumption entails that the inter-event times of the birth-death process are exponentially 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.

We will briefly address the first of these issues today. Specifically, one of the parameter estimation methods that will be introduced tomorrow (*particle filtering*) requires the simulation of a large number of realizations of hypothesized trajectories–too many, in fact, to be simulated with Gillespie's direct method. When we get to particle filtering we'll want to use an alternative algorithm that approximates trajectories of the directly simulated process sampled at regular time intervals. The algorithm we will be using is one of the so-called "tau-leap" methods. Specifically, rather than considering our process to be directly observed at every event, we will sample the process at times $0, \tau, 2\tau, 3\tau, \ldots$. It turns out that the number of events of each time within an interval of duration τ is approximately Poisson distributed with mean equal to the rate of the transition at the start of the interval divided by the duration of the interval. For instance, between times t and $t+\tau$ the number of new infections is given by a Poisson random variate with mean $\beta\frac{IS}{N\tau}$. We can change our update function to use the tau-leap method as follows. Notice that the tau-leap method can occasionally give rise to a greater number of events than actually possible (*e.g.*, a greater number of infections than there are susceptible individuals to be infected). This is related to assumptions that are made in the derivation of the approximation. For our purposes, it will suffice in such cases to set the number of events to the maximum possible. In our function below, this is accomplished using the function `min` at the time the rates are calculated.

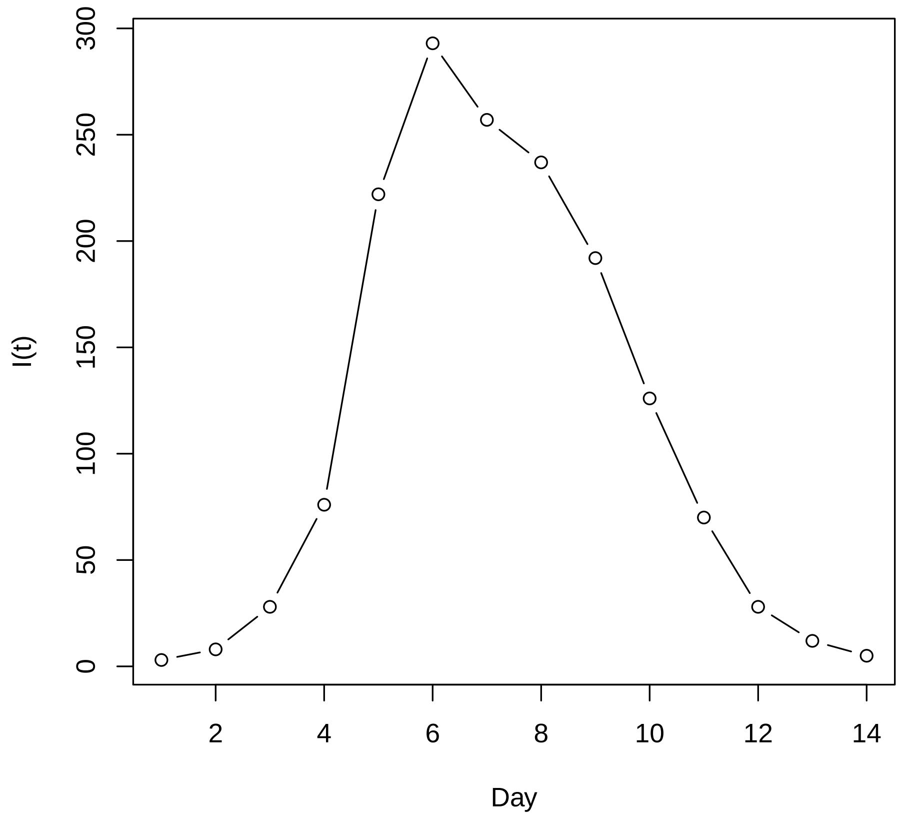
```
> sir.birth.death.onestep.tauleap <- function (x, params, tau) {
+ S <- x[2]
+ I <- x[3]
+ R <- x[4]
+ N <- S+I+R
+ with(
+      as.list(params),
+      {
+        dSI <- min(S,rpois(1,beta*S*I/N * tau))
+        dIR <- min(I,rpois(1,gamma*I * tau))
+        new.sir<-cbind(S - dSI , I + dSI - dIR, R + dIR)
+        cbind(timestep,new.sir)
+        }
+      )
+ }
```

This section on the simple stochastic SI and SIR epidemics has introduced two concepts (continuous time Markov chains and the birth death process) and two techniques (Gillespie's direct method, tau-leap method).

4 Fitting continuous-time models to data: trajectory matching

Thie final section of this module is a segue into the rest of the workshop. To get started, we focus on one of the simplest approaches there is to estimation. If we assume that the only source of variability in the data is measurement error, and that this is symmetrically distributed with a constant variance, then *least squares* is a statistically appropriate basis for estimation. As a demonstration, we fit the deterministic SIR model to data on an outbreak of flu in a British boarding school.

```
> load('flu.RData')
> plot(flu~day,data=flu, type='b', xlab='Day', ylab='I(t)')
```



The first thing we do is write a specialized function for simulating the SIR model in a case where the removal rate is hard-wired in and with no demography.

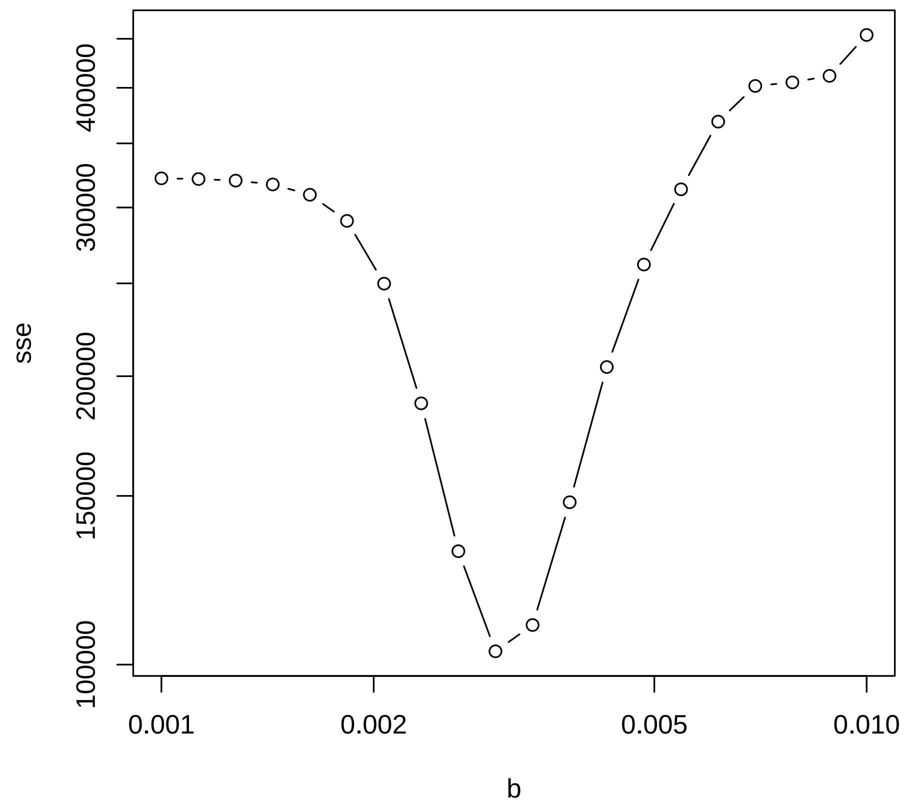
```
> closed.sir.model <- function (t, x, params) {
+   S <- x[1]
+   I <- x[2]
+   R <- x[3]
+   b <- params[1]
+   g <- params[2]
+   dS <- -b*S*I
+   dI <- b*S*I-g*I
+   dR <- g*I
+   list(c(dS,dI,dR))
+ }
```

Now we set up a function that will calculate the sum of the squared differences between the observations and the model at any parameterization (more commonly known as "sum of squared errors").

```
> sse.sir <- function(params0,data){
+   t <- data[,1]
+   cases <- data[,2]
+   b <- params0[1]
+   g <- params0[2]
+   S0 <- 762
+   I0 <- 1
+   R0 <- 0
+   out <- as.data.frame(lsoda(y=c(S=S0,I=I0,R=R0),times=t,closed.sir.model,parms=c(b,g),hmax=1/120))
+   sse<-sum((out$I-cases)^2)
+ }
```

Now, let's see how this function works. To get things started, we'll fix g. Individual infected with flu typically are infectious for roughly one day prior to presenting with symptoms. In this case, we know that sick individuals were immediately removed from interacting with susceptible individuals as soon as they were detected. Therefore, for current purposes we assume $g = \tilde{n} = 1$. In what follows, we first create a dataframe to contain different parameter combinations and our sum of squared errors. We use the function `seq` to generate a sequence of b values uniformly on a log scale and the function `rep` to assign to each value of b the given value of g. Then we use a for loop to evaluate our function `sse.sir` at each combination of parameters. Finally, we plot the resulting values as a function of b.

```
> sse.example<-data.frame(b=1*10^-seq(2,3,length.out=20), g=rep(1,20),sse=NA)
> for(i in 1:dim(sse.example)[1]){
+   sse.example$sse[i]<-sse.sir(as.numeric(sse.example[i,]),data=flu)
+   }
> plot(sse~b,data=sse.example,type='b',log='xy')
```



So far, so good. We can look at our plot and see that there is indeed a local minimum in the sum of squared errors. Specifically, it looks as if the best fit value of b (the value that minimizes the sum of squared errors) is around $\hat{b} = 0.003$. Just two problems remain: (1) The location of this minimum isn't exactly fixed. It looks like it falls somewhere between the 9th and 11th points, but where exactly is uncertain. (2) The minimum of this function is only correct if our assumption $tildeg = 1.0$ is valid, but this was only an approximation.

To solve these problems we need to (1) fit both b and g simultaneously, in which case the sum of squared errors is a surface in two dimenions and we are looking for the minimum of this surface, and (2) examine the shape of this minimum over smaller and smaller ranges until we've zeroed in on a pair of values $(\hat{b}, \hat{g})$ that are known with sufficient precision for whatever purposes we might hope to use them for.

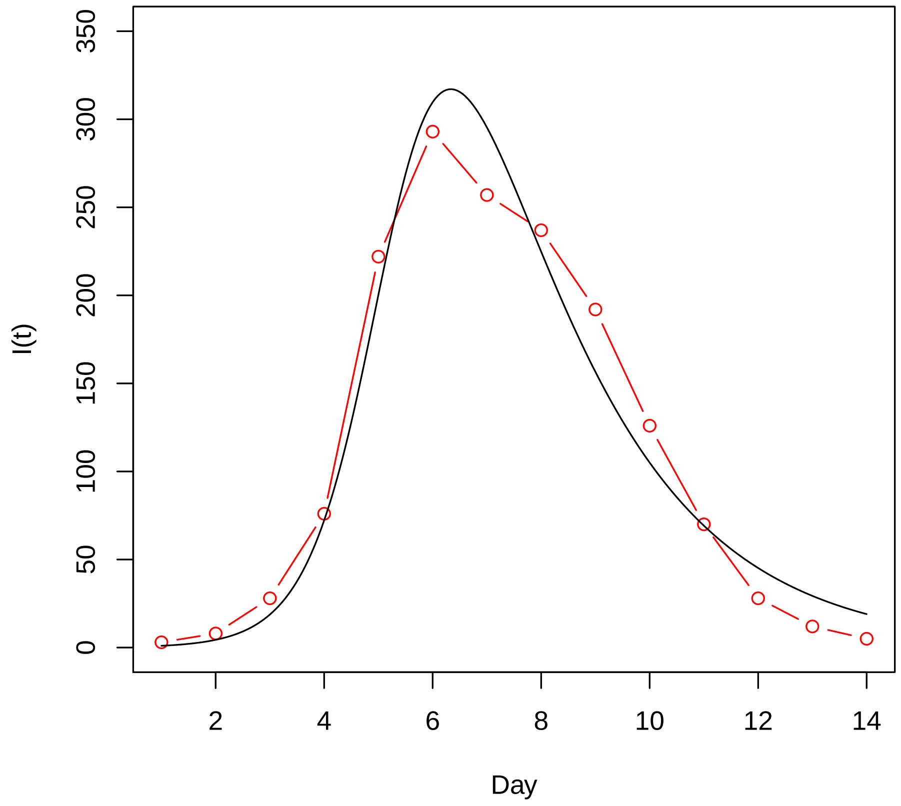
This process is referred to as *optimization* and, fortunately for us, there are many robust algorithms available for this purpose. One of them, the *Nelder-Mead algorithm*, is the default in the Rfunction `optim`. We can use it to return the best fit values for b and g as follows.

```
> params0<-c(0.001,0.5)
> fit1 <- optim(params0,sse.sir,data=flu); fit1$par

[1] 0.002567216 0.473148936
```

Finally, we plot these fits against the data.

```
> plot(flu~day,data=flu, type='b', xlab='Day', ylab='I(t)',ylim=c(0,350),col='red')
> t <- seq(1,max(flu$day),by=0.05)
> mod.pred<-as.data.frame(lsoda(c(S=762,I=1,R=0),times=t,
+                                 closed.sir.model,fit1$par,hmax=1/120))
> lines(mod.pred$I~t)
```



Exercise 6. The file `plague.RData` gives weekly mortality for the plague outbreak in Mumbai, December 1905 to July 1906. We assume that human mortality X is proportional to the number of infectious rats $X(t) = \mu I(t)$ and that the epidemic in the rat population can be represented by a simple closed SIR model where S, I, and R are proportions of the rate population. Use least squares to estimate the parameters of this model, assuming that one in a million rats are infected at time $t = 0$. Compare model output with data using a plot.

Exercise 7. Repeat the fit you performed in the previous exercise, treating $I(0)$ as an unknown parameter that must be estimated.

Hint. All parameters are necessarily positive. It sometimes helps the optimization algorithm if negative parameter values are impossible. Thus, it might be useful to re-parameterize the model with a new parameter $b = log(\beta)$, i.e., $dS/dt = -e^b SI$, and so on.

```
[1]   2.350065   2.313281 -12.917853  14.083983
```

```
[1]   2.357647   2.320961 -12.956612  14.091217
```

Acknowledgements. This tutorial is the extant type of a lineage that has an unknown evolutionary history. Thanks to Aaron King, Ben Bolker, Ottar Bjørnstad, and Dave Smith for the use of source materials from recent common ancestors.