Simulating epidemics in R

John M. Drake

June 24, 2011

Contents

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 model simply keeps track of how many individuals are in each class: individuals that leave one class must enter another class. 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. A particularly interesting bit is the *force-of-infection*, represented by the function $\lambda(I,t)$. 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 faced by a susceptible individual 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 very sophisticated ODE solving capabilities in the package `deSolve`. To use these algorithms we first load the package:

```
> require(deSolve)
```

[Note: If you get a warning that the package was not loaded, check to be sure it is installed on your computer. It can be installed/re-installed by typing `install.packages('deSolve')` at the command line.]

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]     #susceptibles
+   I <- x[2]     #infected
+   R <- x[3]     #recovered
+   with(
+        as.list(params), #local environment to evaluate derivatives
+        {
+          dS <- mu*(N)-beta*S*I/N-mu*S
+          dI <- beta*S*I/N-(mu+gamma)*I
+          dR <- gamma*I-mu*R
+          dx <- c(dS,dI,dR)
+          list(dx)
+        }
+        )
+ }
```

Notice that in this case 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, with `dS <- params["mu"]*(params["N"]-S)-params["beta"]*S*I/params["N"]` and so on-but that's more complicated to write.]

We'll also write a function to calculate R_0.

```
> R0 <- function(params) with(as.list(params), beta/(mu+gamma))
```

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:

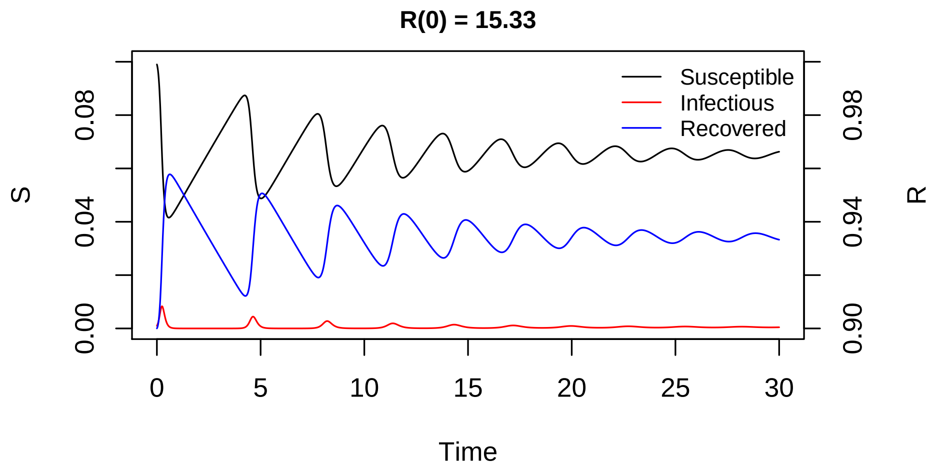
```
> times <- seq(0,30,by=1/120)
> params <- c(mu=1/70,N=1,beta=400,gamma=365/14)
> xstart <- c(S=1-0.001-0.9,I=0.001,R=0.9)
```

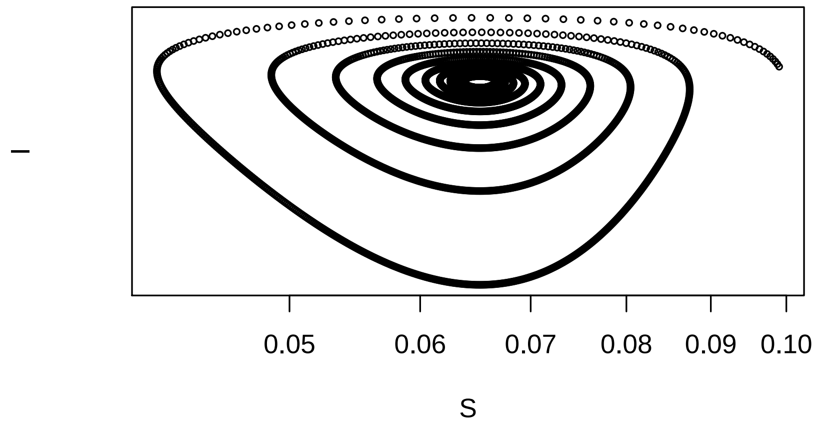
Now we can simulate a model trajectory with the `ode` command:

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

and plot the results

```
> op <- par(fig=c(0,1,0,0.5),mar=c(4,4,2,5))
> plot(I~S,data=out,type='b',log='xy',yaxt='n',xlab='S',cex=0.5)
> par(fig=c(0,1,0.5,1),mar=c(4,4,2,5), new=TRUE)
> plot(S~time,data=out,type='l', ylim=c(0,0.1), xlab='Time')
> lines(I~time,data=out,type='l',col='red'); par(new=TRUE)
> plot(R~time,data=out,type='l', ylim=c(0.9,1), col='blue', axes=FALSE,
+   xlab='', ylab='', main=paste('R(0) =',round(R0(params),2)), cex.main=0.9)
> axis(4)
> mtext('R', side=4, line=3)
> legend('topright', legend=c('Susceptible', 'Infectious', 'Recovered'), col=c('black','red','blue'), l
> 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.

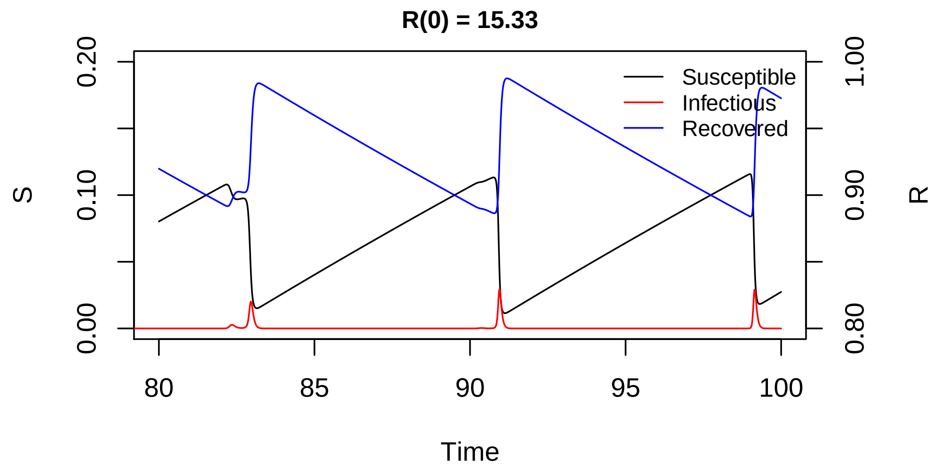
Seasonality

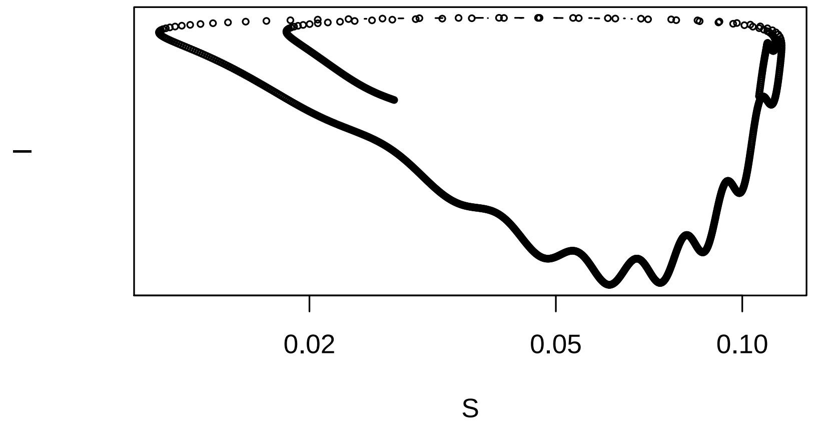
The simple SIR model always predicts damped oscillations towards an equilibrium (or pathogen extinction if R_0 is too small). This is at odds with the recurrent outbreaks seen in many real pathogens. Sustained oscillations require some additional drivers in the model. An important driver in childhood 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 <- beta*(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=1/70,N=1,beta=400, beta1=0.6, gamma=365/14)
> xstart <- c(S=1-0.001-0.9,I=0.001,R=0.9)
> out <- as.data.frame(ode(xstart,times,seasonal.sir.model,params,rtol=1e-12,hmax=1/120))
> op <- par(fig=c(0,1,0,0.5),mar=c(4,4,2,5))
> plot(I~S,data=out,type='b',log='xy',yaxt='n',xlab='S',cex=0.5, subset=time>=90)
> par(fig=c(0,1,0.5,1),mar=c(4,4,2,5), new=TRUE)
> plot(S~time,data=out,type='l', subset=time>=80, ylim=c(0,0.2), xlab='Time')
> lines(I~time,data=out,type='l',col='red'); par(new=TRUE)
> plot(R~time,data=out,type='l', subset=time>=80, ylim=c(0.8,1), col='blue', axes=FALSE,
+   xlab='', ylab='', main=paste('R(0) =',round(R0(params),2)), cex.main=0.9)
> axis(4)
> mtext('R', side=4, line=3)
> legend('topright', legend=c('Susceptible', 'Infectious', 'Recovered'), col=c('black','red','blue'), l
> 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.

Delays

In this section we introduce a new possibility for our epidemic: delays. One of the later objectives of this workshop and an active area of research is to understand role that individual heterogeneity plays in determining the shape of epidemics. It is sometimes said that models such as the SIR model above assume *exponential waiting times*. This is because the only stochastic model that is uniquely determined by the above mean rates of change entails that the period of time an individual is infectious is exponentially distributed. This is awkward because it further implies that the mode of the distribution of time individuals are infectious is 0! At the other extreme, we could assume that all individuals have a fixed infectious period of exactly $1/\gamma$.

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

$$
\begin{aligned}
\frac{dS}{dt} &= \mu\, N - \lambda(I,t)\, S - \mu\, S \\
\frac{dI}{dt} &= \lambda(I,t)\, S - \lambda(I(t-1/\gamma)), t-1/\gamma)\, S(t-1/\gamma) e^{\mu/\gamma} - \mu\, I \\
\frac{dR}{dt} &= \lambda(I(t-1/\gamma), t-1/\gamma)\, S(t-1/\gamma) e^{\mu/\gamma} - \mu\, R
\end{aligned}
$$

The only difference between this model and the earlier one is that we want the loss of infected individuals due to recovery to be exactly the incidence of infection at time $t-1/\gamma$ (*i.e.*, $\lambda(I(t-1/\gamma), t-1/\gamma)\, S(t-1/\gamma)$), less the proportion of those individuals that died during the infectious period $(1-e^{\mu/\gamma})$. Ultimately, we will be solving our model using a new function (`dede`). As for `ode`, `dede` requires that we provide the solver with our equations in a very specific format that deviates from the `ode` format only slightly in that our new function allows us to retrieve former values of the process.

Translating into R:

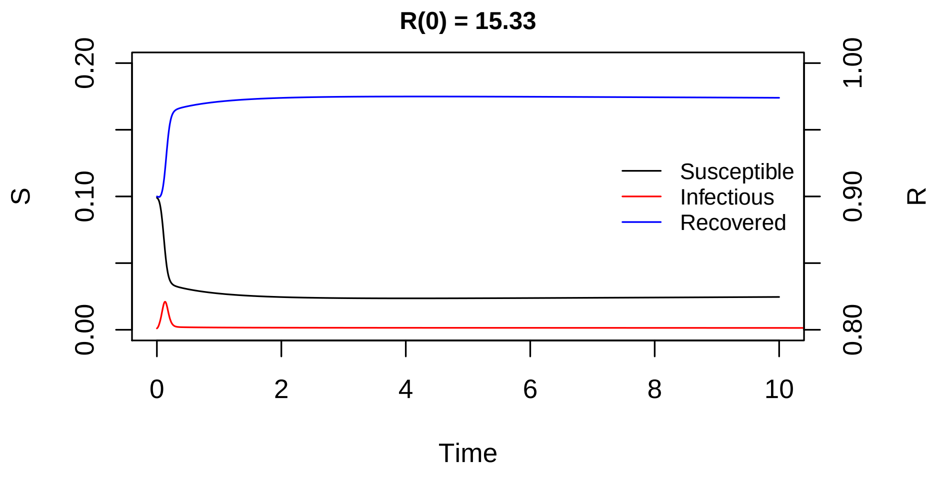
```
> delay.sir.model <- function (t, x, params) {
+   with(
+        as.list(c(x,params)),
+        {
+
+         if (t < 1/gamma){  #if time is less than one infectious period then lagged I is 0
+          I.lag <- 0
+          S.lag <- 0
+          }
+         else{
+         I.lag <- lagvalue(t - 1/gamma, 2)  #get I at lag of infectious period
+         S.lag <- lagvalue(t - 1/gamma, 1)  #get S at lag of infectious period
+         }
+          dS <- mu*(N-S)-beta*S*I/N
+          dI <- beta*S*I/N-mu*I-beta*I.lag*S.lag*exp(-mu/gamma)
+          dR <- beta*I.lag*S.lag*exp(-mu/gamma)-mu*R
+          dx <- c(dS,dI,dR)
+          list(dx)
+        }
+        )
+ }
```

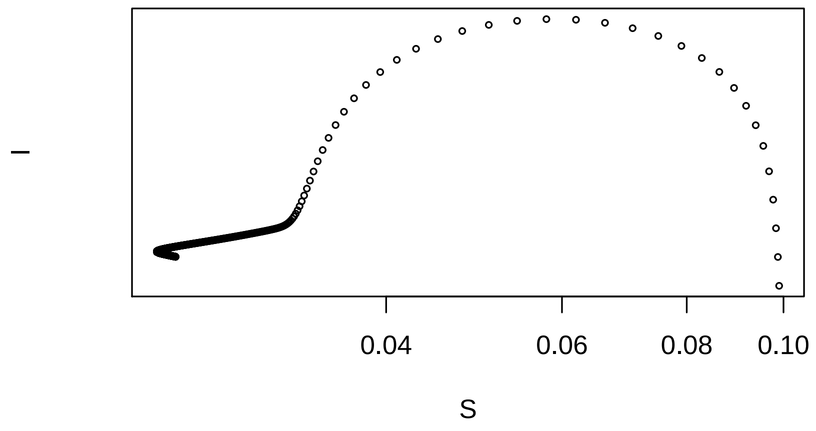
We'll simulate as before

```
> times <- seq(0,100,by=1/120)  #lagged model onl runs to time=25
> params <- c(mu=1/70,N=1,beta=400,gamma=365/14)
> xstart <- c(S=1-0.001-0.9,I=0.001,R=0.9)
> out <- as.data.frame(dede(xstart,times,delay.sir.model,params,rtol=1e-12,hmax=1/120))
```

and plot:

```
> op <- par(fig=c(0,1,0,0.5),mar=c(4,4,2,5))
> plot(I~S,data=out,type='b',log='xy',yaxt='n',xlab='S',cex=0.5, subset=time<=10)
> par(fig=c(0,1,0.5,1),mar=c(4,4,2,5), new=TRUE)
> plot(S~time,data=out,type='l', ylim=c(0,0.2), xlab='Time', subset=time<=10)
> lines(I~time,data=out,type='l',col='red'); par(new=TRUE)
> plot(R~time,data=out,type='l', ylim=c(0.8,1), col='blue', axes=FALSE, subset=time<=10, xlab='', ylab=
> axis(4)
> mtext('R', side=4, line=3)
> legend('right', legend=c('Susceptible', 'Infectious', 'Recovered'), col=c('black','red','blue'), lty=
> par(op)
```





What we see is that delays strongly synchronize epidemics. The epidemiological consequences are substantial. Epidemic fadeouts are much more rapid and transient osciallations damp much more quickly.

Exercise 4. Explore the effects of changing the infectious period in the delay-SIR model.

***Exercise 5.** Modify the codes given to study the dynamics of an SEIR model with fixed incubation period (E-stage) and infectious period (I-stage).

***Exercise 6.** Study a model with seasonality and fixed incubation period.

3 Summary

So far today we have used one model (the SIR model) to introduce three concepts (frequency-dependent transmission, seasonal forcing, and delay differential equations) and one technique (numerical solution with `ode` and `dede`).