Simulating epidemics in R

Helen J. Wearing, John M. Drake & Aaron A. King

May 20, 2012

Contents

1 Introduction

This workshop will introduce techniques for fitting models to different types of epidemiological data. 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 the structures that are useful for modeling the *temporal dynamics* of disease transmission. This introduction serves several purposes. First, by 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 some 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 Chain binomial model

We begin by developing an intuitive understanding of the mechanics of the transmission process by considering a simple stochastic model of an epidemic: the *chain binomial* model.

This model stipulates that the epidemic evolves according to discrete generations. In each generation, new infections are binomially distributed with the number of trials equal to the number of susceptibles, S_t, and probability of infection, $p = 1 - \exp(-\beta\, I_t)$. In probability notation:

$$I_{t+1} \sim \mathrm{binom}(S_t, 1 - \exp(-\beta\, I_t))$$

Susceptibles are then depleted by the number of these infections

$$S_{t+1} = S_t - I_{t+1}$$

Recall from probability and statistics that the binomial random variable is the number of independent "successes" in a sequence of weighted coin tosses. The analogy here is that we toss a weighted coin (with probability of heads $p = 1 - \exp(-\beta\, I_t)$) for each susceptible individual in the population. If the weighted coin does in fact come up heads then the susceptible individual becomes infected. Otherwise, it stays susceptible and we move on to the next susceptible individual.

Simulating this simple model is easy in R. First, we write a function that simulates a single generation.

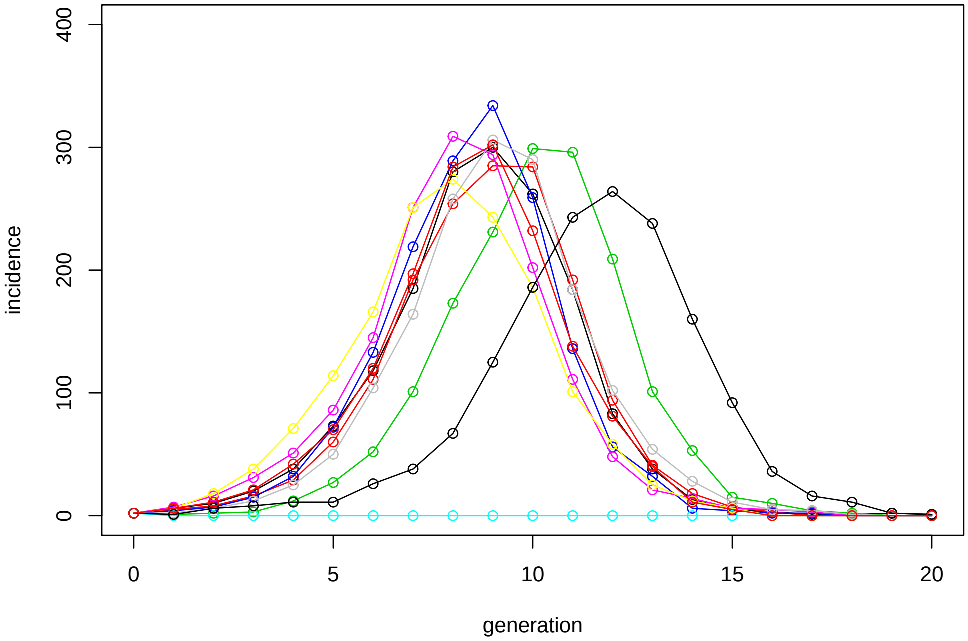
```
chain.binomial.onestep <- function(x, params) {
    S <- x[1]
    I <- x[2]
    beta <- params["beta"]
    new.I <- rbinom(n = 1, size = S, prob = 1 - exp(-beta * I))
    new.S <- S - new.I
    c(S = new.S, I = new.I)
}
```

Then we put these together in sequence to simulate an entire epidemic:

```
chain.binomial.model <- function(x, params, nstep) {
    X <- array(dim = c(nstep + 1, 3))
    colnames(X) <- c("time", "S", "I")
    X[1, 1] <- 0
    X[1, -1] <- x
    for (k in 1:nstep) {
        X[k + 1, 1] <- k
        X[k + 1, -1] <- x <- chain.binomial.onestep(x, params)
    }
    X
}
```

We'll now specify some parameters, simulate the model a few times, and plot the results.

```
set.seed(38499583)
nsims <- 10
nstep <- 20
xstart <- c(S = 2000, I = 2)
params <- c(beta = 0.001)
x <- vector(mode = "list", length = nsims)
for (k in 1:nsims) {
    x[[k]] <- as.data.frame(chain.binomial.model(xstart, params, nstep))
}
plot(c(0, 20), c(0, 400), type = "n", xlab = "generation", ylab = "incidence")
for (k in 1:nsims) {
    lines(I ~ time, data = x[[k]], col = k, type = "o")
}
```



Exercise 1. Explore the dynamics of the system for different values of β, as well as different initial values of S and I.

Although simple, the chain binomial model captures some key properties of the real biological process:

- demographic stochasticity - a type of process noise (to be contrasted with *measurement error*)
- categorical class variables (S and I are integer-valued)

However, the chain binomial, like all models, is an approximation. One large assumption that it makes is that the generations are perfectly synchronized. For some diseases, this may not be such a bad approximation; for others, it might very well be. Let's have a look at what can be done with models that don't make this assumption, i.e. generations of infection are not synchronized. In fact, for now, we'll take it one step further and assume that the change in the number of susceptible and infectious individuals in the population happens continuously.

3 The SIR model

The simplest place to start is with the classical SIR model. This model divides the host population into three classes with respect to their infection status: individuals are either Susceptible, Infected (and Infectious), or Recovered. The model simply keeps track of how many individuals are in each class: individuals that leave one class must enter another. The only exceptions, of course, are births and deaths.

We could formulate a stochastic model that is continuous in time (see later if time permits) but here we're going to assume we have a large (technically infinitely large) population in which the effects of demographic stochasticity become negligible. Therefore, in the estimation module, we'll be thinking about measurement error when fitting this model to data.

The state variables change according to a system of differential equations:

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

Here, B is the crude birth rate, μ is the per capita death rate, N is the host population size, and γ the recovery rate. The term that makes this model interesting (and nonlinear) 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)


## Loading required package: 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) {
    ## first extract the state variables
    S <- x[1]
    I <- x[2]
    R <- x[3]
    N <- S + I + R
    ## now extract the parameters
    beta <- params["beta"]
    gamma <- params["gamma"]
    mu <- params["mu"]
    B <- params["B"]
```

```r
    ## now code the model equations
    dSdt <- B - beta * S * I/N - mu * S
    dIdt <- beta * S * I/N - (mu + gamma) * I
    dRdt <- gamma * I - mu * R
    ## combine results into a single vector
    dxdt <- c(dSdt, dIdt, dRdt)
    ## return result as a list!
    list(dxdt)
}
```

Notice that in this case we've assumed β is constant.

We'll also write a function to calculate R_0.

```r
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:

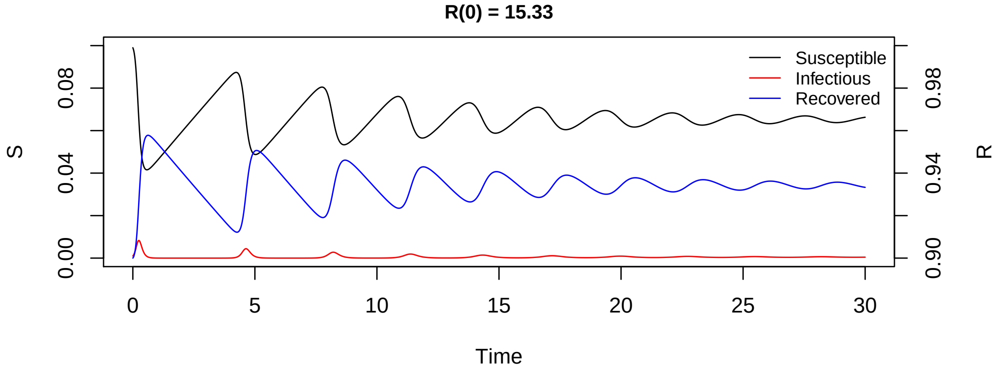
```r
times <- seq(0, 30, by = 1/120)
params <- c(B = 1/70, mu = 1/70, N = 1, beta = 400, gamma = 365/14)
xstart <- c(S = 1 - 0.001 - 0.9, I = 0.001, R = 0.9)
```

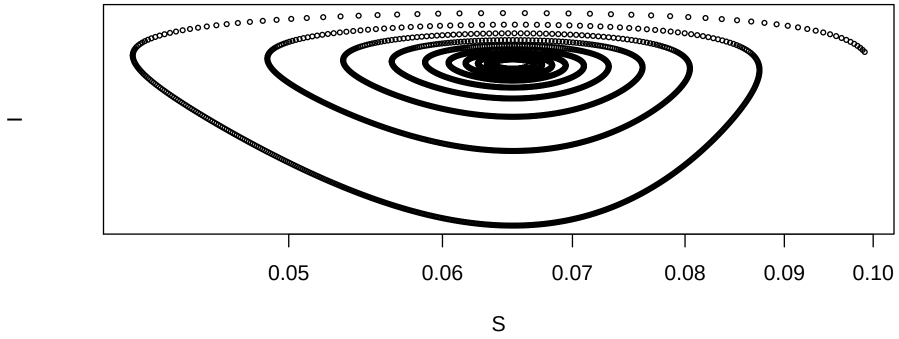
Now we can simulate a model trajectory with the `ode` command:

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

and plot the results

```r
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"), lty = 1, bty = "n", cex = 0.8)
par(op)
```





Exercise 2. Explore the dynamics of the system for different values of the β and B parameters by simulating and plotting trajectories as time series and in phase space (e.g., I vs. S).

***Exercise 3.** Modify the codes given to study the dynamics of an SEIR model.

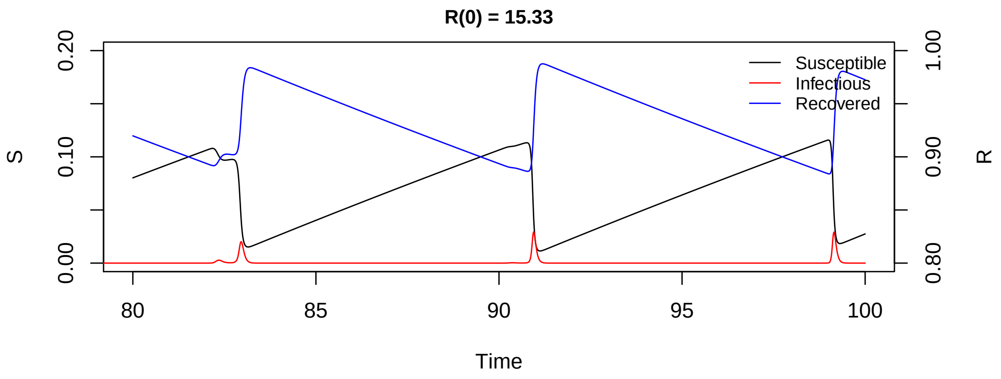
Seasonality

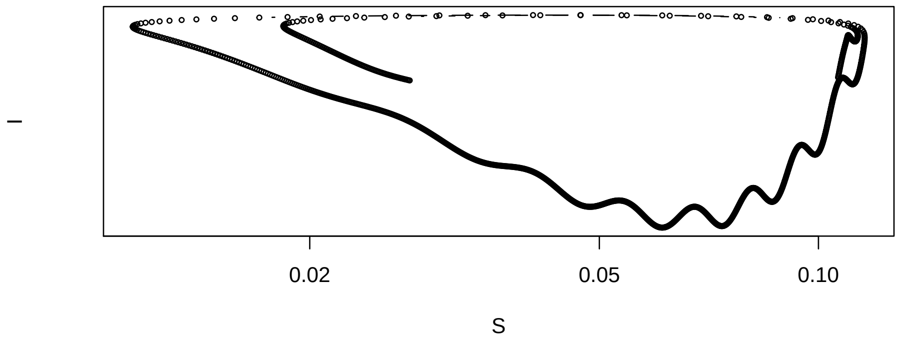
The simple SIR model always predicts damped oscillations towards an equilibrium (or pathogen extinction if R_0 is too small). This is at odds with the recurrent outbreaks seen in many real pathogens. Sustained oscillations require some additional drivers in the model. An important driver in childhood infections of humans (e.g., measles) is seasonality in contact rates because of aggregation of children during the 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) {
    S <- x[1]
    I <- x[2]
    R <- x[3]
    N <- S + I + R
    #
    beta <- params["beta"]
    beta1 <- params["beta1"]
    gamma <- params["gamma"]
    mu <- params["mu"]
    B <- params["B"]
    #
    beta <- beta * (1 + beta1 * cos(2 * pi * t))
    dS <- B - mu * S - beta * S * I/N
    dI <- beta * S * I/N - (mu + gamma) * I
    dR <- gamma * I - mu * R
    dxdt <- c(dS, dI, dR)
    list(dxdt)
}
```

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(B = 1/70, 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"), lty = 1, bty = "n", cex = 0.8)
par(op)
```





Exercise 4. 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.