Fitting epidemics in R

John M. Drake

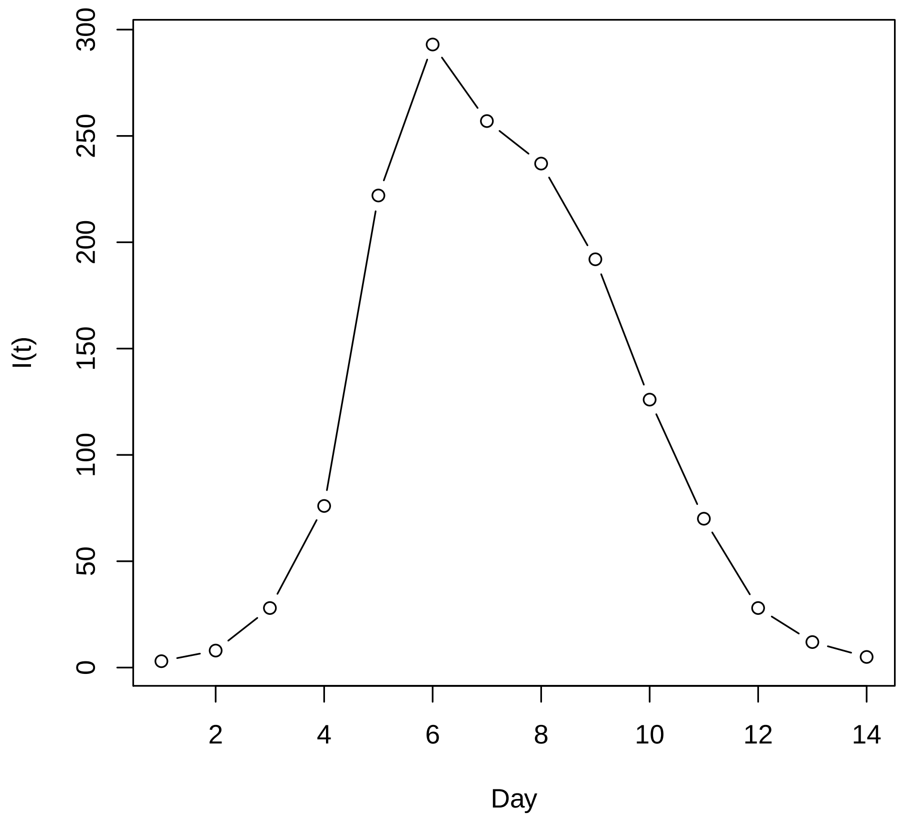
June 23, 2011

Contents

1 Introduction

In this section we bring together the two tools introduced so far in this course: statistical estimation and dynamical systems. As a demonstration, we fit the closed deterministic SIR model to data on an outbreak of flu in a British boarding school. The data are available in the workspace `flu.RData`.

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



2 Fitting continuous-time models to data: trajectory matching with least squares

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. If these conditions apply, then *least squares* is a statistically appropriate basis for estimation. Moreover, as mentioned earlier, in the case that errors are normally distributed with Gaussian distribution then least squares estimates are also the maximum likelihood estismates. Finally, if we're doing some exploratory work, least squares is often a simple approach to get things started, to find some initial values, to discover deviations (by inspeciting residuals), etc.

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. This is only very slightly different (and simpler) than the models we were simulating earlier.

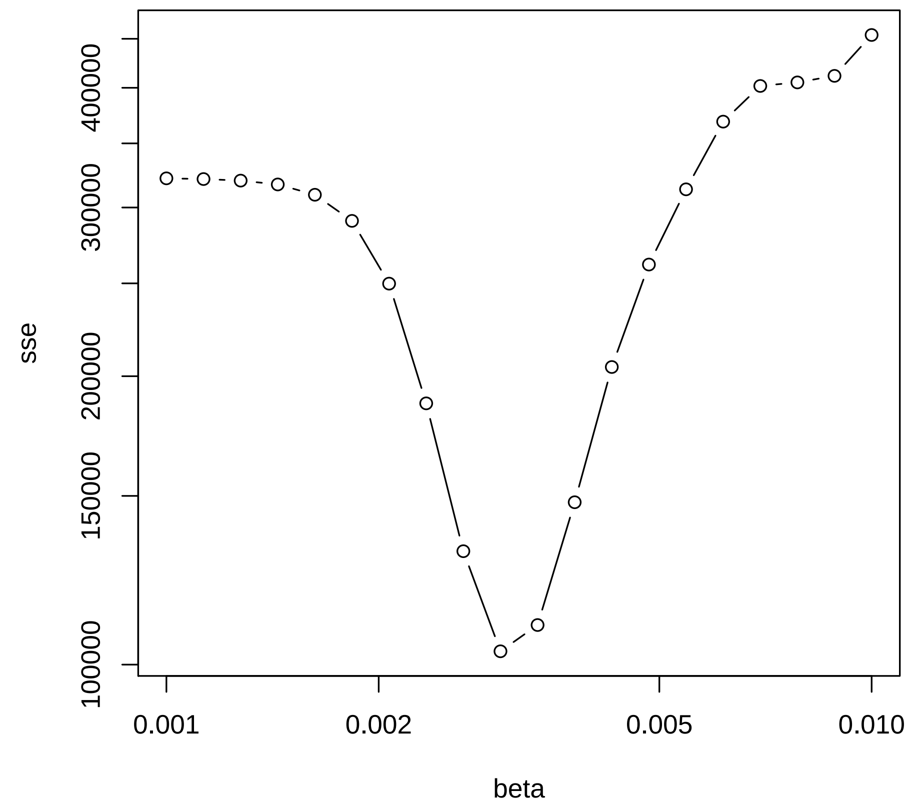
```
> closed.sir.model <- function (t, x, params) {
+   S <- x[1]
+   I <- x[2]
+   R <- x[3]
+   beta <- params[1]
+   gamma <- params[2]
+   dS <- -beta*S*I
+   dI <- beta*S*I-gamma*I
+   dR <- gamma*I
+   list(c(dS,dI,dR))
+ }
```

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").

```
> require(deSolve)
> sse.sir <- function(params0,data){
+   t <- data[,1]
+   cases <- data[,2]
+   beta <- params0[1]
+   gamma <- params0[2]
+   S0 <- 762
+   I0 <- 1
+   R0 <- 0
+   out <- as.data.frame(ode(y=c(S=S0,I=I0,R=R0),times=t,closed.sir.model,parms=c(beta,gamma),hmax=1/12
+   sse<-sum((out$I-cases)^2)
+ }
```

Now, let's see how this function works. To start, we'll fix γ. Individual infected with flu typically are infectious for roughly one day prior to presenting with symptoms. In this case, we may suspect that sick individuals were immediately removed from interacting with susceptible individuals as soon as they were detected. Therefore, for current purposes we assume $\gamma = \tilde{g} = 1$. 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 β values uniformly on a log scale and the function `rep` to assign to each value of β the given value of γ. 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 β.

```
> sse.example<-data.frame(beta=1*10^-seq(2,3,length.out=20), gamma=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~beta,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 β (the value that minimizes the sum of squared errors) is around $\hat{\beta} = 0.003$. 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 $\tilde{\gamma} = 1.0$ is valid, but this was only an approximation.

To solve these problems we need to (1) fit both β and γ 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 $(\hat{\beta}, \hat{\gamma})$ 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 function `optim`. We can use it to return the best fit values for β and γ as follows.

```
> params0<-c(0.001,0.5)
> fit0 <- optim(params0,sse.sir,data=flu); fit0$par
```

```
[1] 0.002567216 0.473148936
```

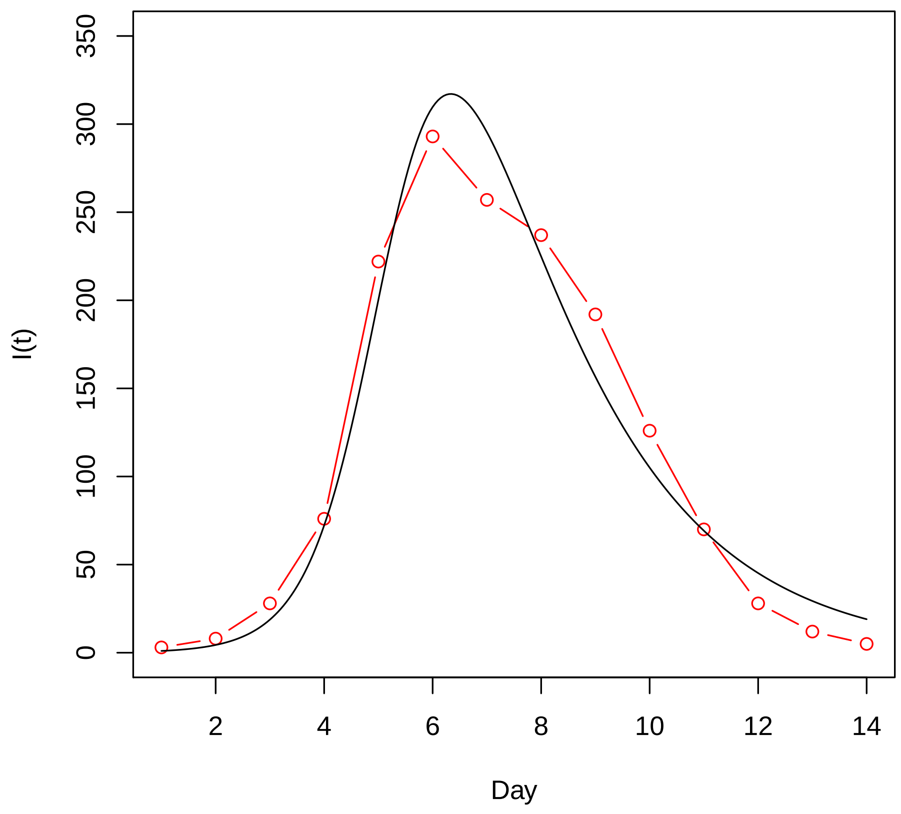
The first set of parameters returned by `optim` are not always the best. Therefore, we'll run it one more time just to be sure, starting at the values returned by the first optimization.

```
> fit1 <- optim(fit0$par,sse.sir,data=flu); fit1$par
```

```
[1] 0.002567234 0.473118744
```

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 1. 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 2. 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
```

3 Fitting continuous-time models to data: trajectory matching with Maximum Likelihood

Now we have a very generic tool that you can use to fit just about any complicated dynamical model you might invent. But, this morning Ben gave the hard sell for likelihood. Now we want to fit a dynamical model using maximum likelihood (and not just the Gaussian case, where MLE and least sqaures are equivalent). As before, we write a function for simulating the SIR model in a case where the removal rate is "hard-wired" (*i.e.*, not passed as a parameter) and with no demography.

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

Now we write a function to return the negative log-likelihood of the data, give some combination of parameters. Recalling that β and γ must (by definition) be greater than one, we will think of β and γ as simple transformations of two other parameters $\beta = e^b$ and $\gamma = e^g$. The parameters b and g are defined from negative infinity to positive infinity. This will help our numerical algorithms (e.g., `mle2`) to behave better. Further, we will assume te observations to be Poisson distributed. This seems appropriate, since we have discrete observations of cases and it is natural to expect the variance to scale with the number infected.

```
> sir.nll <- function(b){
+   times <- seq(0,14)
+   parms <- c(exp(b), 0.75)
+   S0 <- 762
+   I0 <- 1
+   R0 <- 0
+   out <- as.data.frame(ode(y=c(S=S0,I=I0,R=R0),times=times,closed.sir.model,parms=parms,hmax=1/120))
+   nll<--sum(dpois(x=flu$flu, lambda=tail(out$I,14), log=TRUE))
+ }
>
```

Using `mle2` we fit the model (according to the likelihood criterion).

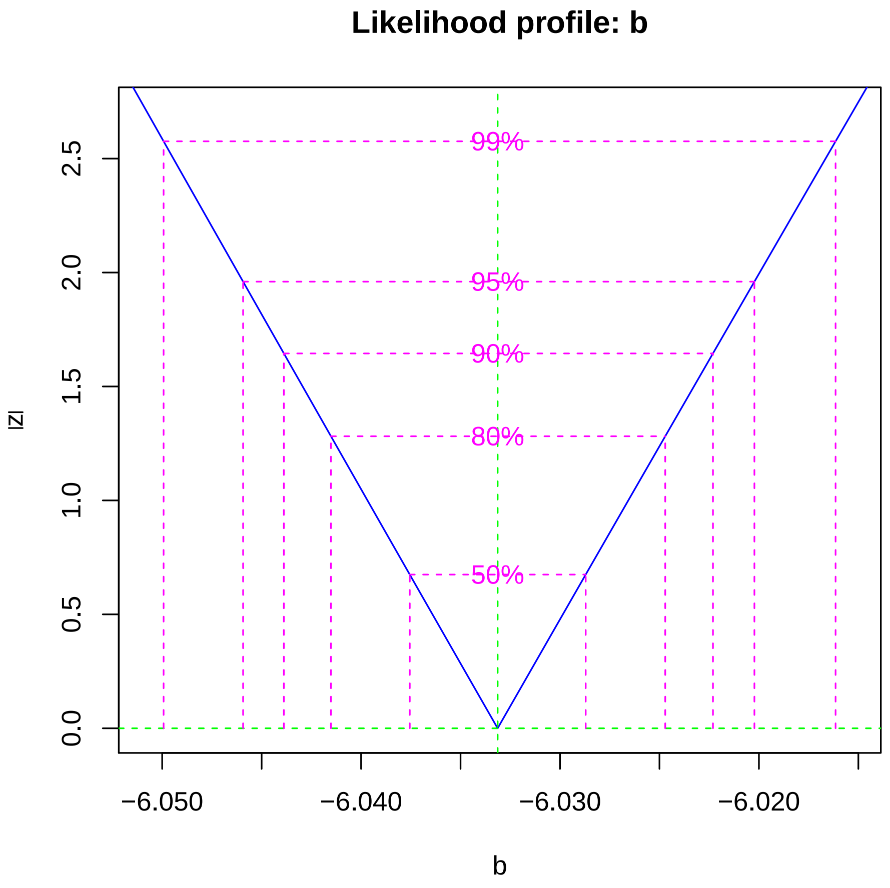
```
> require(bbmle)
> params0 <-list(b=-6)
> fit0 <- mle2(sir.nll, start=params0); fit0

Call:
mle2(minuslogl = sir.nll, start = params0)

Coefficients:
       b
-6.03314

Log-likelihood: -271.05

> fit <-  mle2(sir.nll, start=as.list(coef(fit0))); fit

Call:
mle2(minuslogl = sir.nll, start = as.list(coef(fit0)))

Coefficients:
        b
-6.033135

Log-likelihood: -271.05
```

This makes available all the tools introduced this morning, such as the likelihood profile on b,

```
> p<-profile(fit)
```

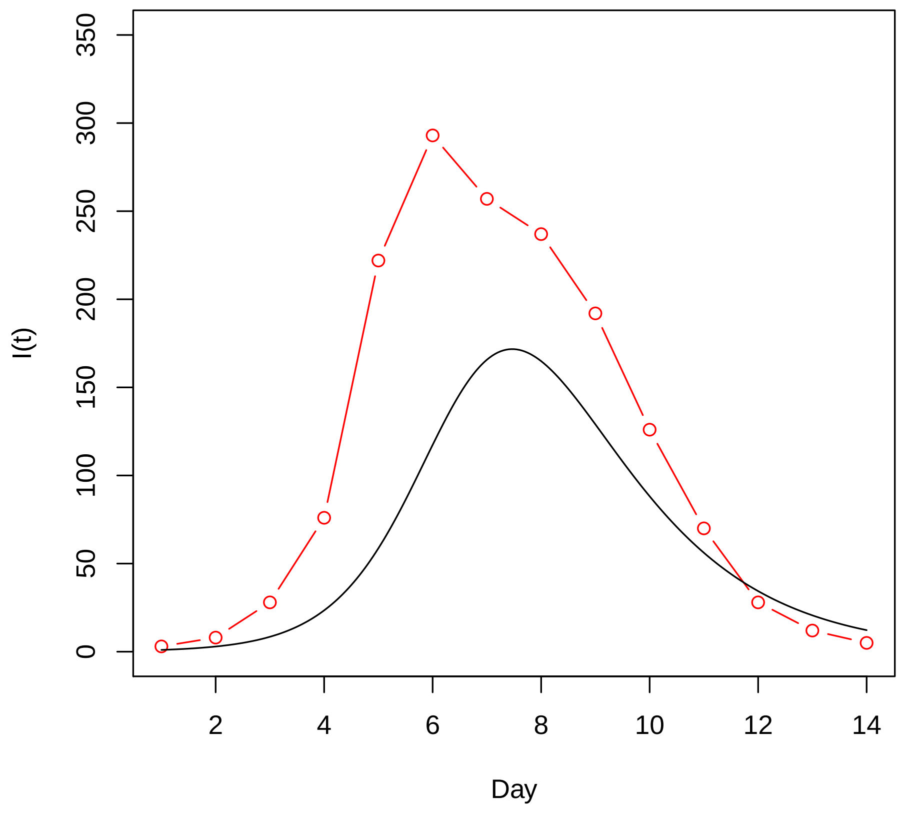
and plot.

```
> plot(p, absVal=TRUE)
```



How does this model look when plotted 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$day),by=0.05)
> mod.pred<-as.data.frame(ode(c(S=762,I=1,R=0),times=t, closed.sir.model,c(exp(coef(fit)), 0.75),hmax=1/
> lines(mod.pred$I~t)
```



Hmmm... Not so good.

Recalling that we obtained our parameter b (and therefore the transmission rate β) assuming a particular value of γ, we wonder now how good this assumption was. Indeed, it is not hard to extend our approach to estimate b and g simultaneously, as follows.

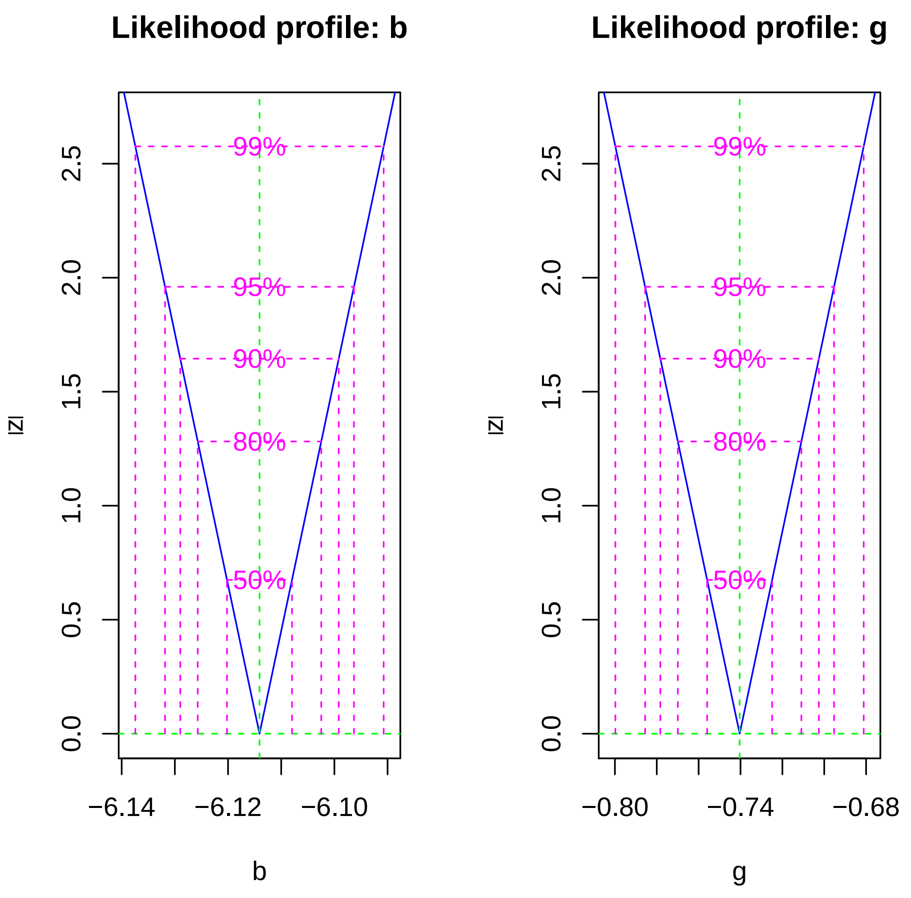
```
> closed.sir.model <- function (t, x, params) {
+   S <- x[1]
+   I <- x[2]
+   R <- x[3]
+   beta <- params[1]
+   gamma <- params[2]   #now gamma is a parameter
+   dS <- -beta*S*I
+   dI <- beta*S*I-gamma*I
+   dR <- gamma*I
+   list(c(dS,dI,dR))
+ }
> sir.nll <- function(b, g){       #read in  b and g
+   times <- seq(0,14)
+   parms <- c(exp(b),exp(g))      #anti-log to get beta and gamma
+   S0 <- 762
+   I0 <- 1
+   R0 <- 0
+   out <- as.data.frame(ode(y=c(S=S0,I=I0,R=R0),times=times,closed.sir.model,parms=parms,hmax=1/120))
+   nll<--sum(dpois(x=flu$flu, lambda=tail(out$I,14), log=TRUE))
+ }
>
```

As before, we pick some initial values and fit.

```
> params0 <-list(b=-6, g=-0.7)
> fit0 <- mle2(sir.nll, start=params0); fit0

Call:
mle2(minuslogl = sir.nll, start = params0)

Coefficients:
         b          g
-6.1140864 -0.7404035

Log-likelihood: -76.57

> fit <-  mle2(sir.nll, start=as.list(coef(fit0))); fit

Call:
mle2(minuslogl = sir.nll, start = as.list(coef(fit0)))

Coefficients:
         b          g
-6.1140809 -0.7404055

Log-likelihood: -76.57

>
+
```

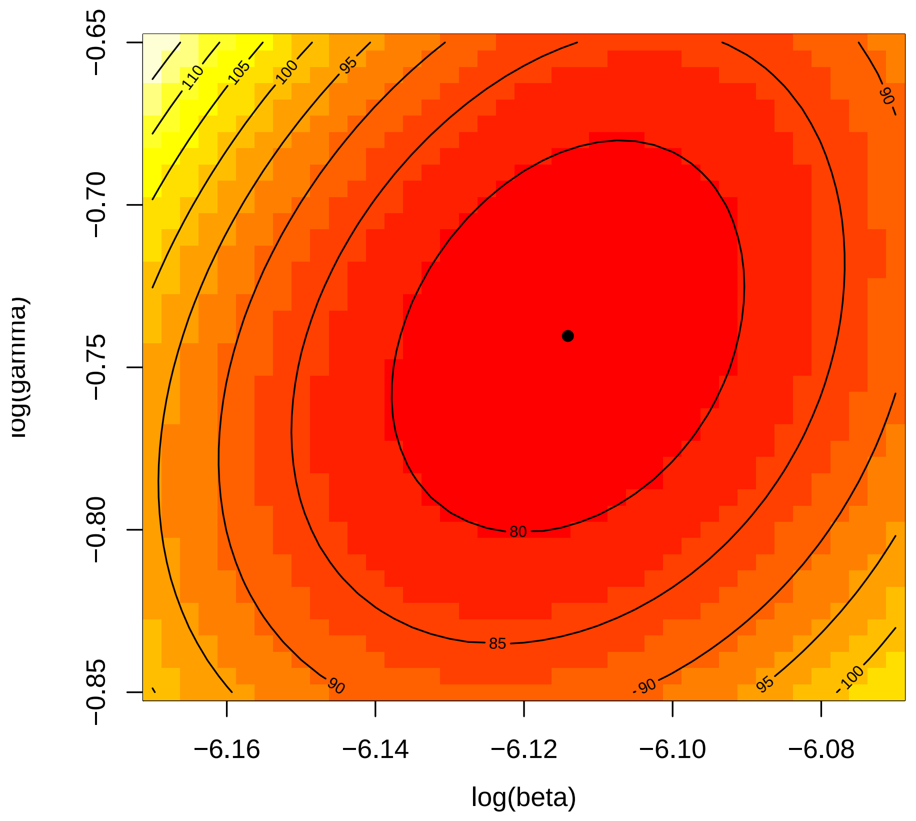
As before, we ask for likelihood profiles.

```
> p<-profile(fit)
> plot(p, absVal=TRUE)
```



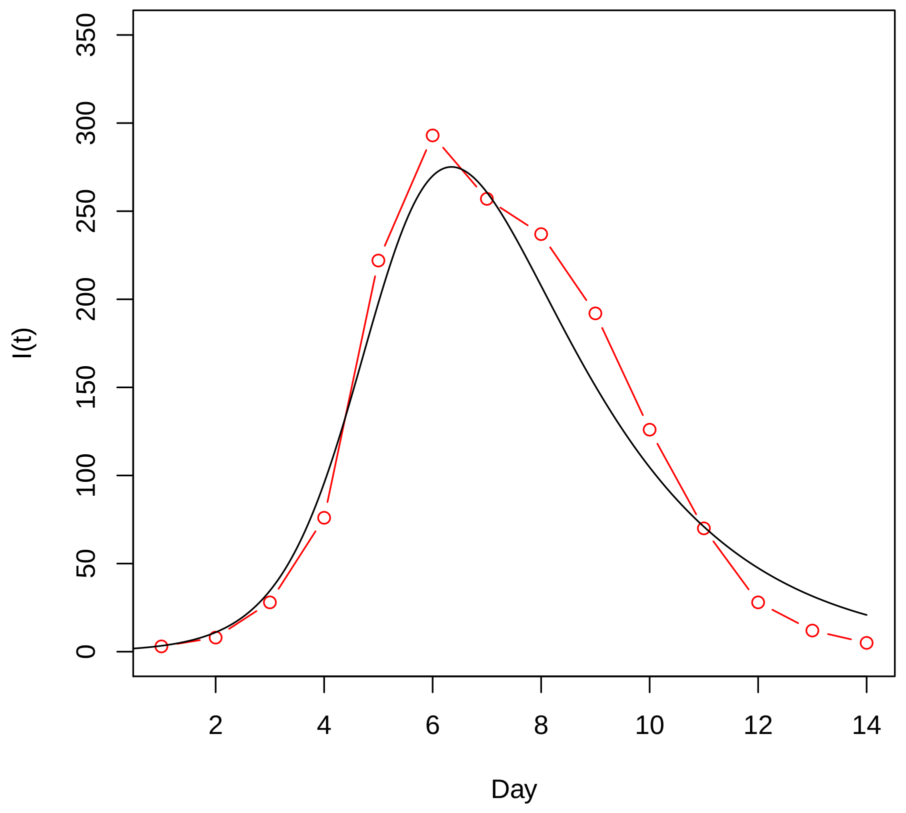
Now that we have two parameters, it's also sensible to look at the contour plot.

```
> require(emdbook)
> cc <- curve3d(sir.nll, xlim=c(-6.17,-6.07),ylim=c(-0.85,-0.65),
+               sys3d="image",ann=FALSE, useRaster=TRUE)
> mtext(side=1,line=2.5,"log(beta)")
> mtext(side=2,line=3.5,"log(gamma)",las=0)
> contour(cc$x,cc$y,cc$z,add=TRUE)
> points(coef(fit)[1],coef(fit)[2],pch=16)
```



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(0,max(flu$day),by=0.05)
> mod.pred<-as.data.frame(ode(c(S=762,I=1,R=0),times=t,
+                               closed.sir.model,exp(coef(fit)),hmax=1/120))
> lines(mod.pred$I~t)
>
```



Much better!

***Exercise 3.** Propose a model for the plague data fit earlier with least squares and fit instead using likelihood.

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.