Basic SIR fitting - Details

June 12, 2017

This vignette provides technical details of the `fitsir` package.

Contents

1 Sensitivity equations

Let $x_i(t, \theta)$ be the states of the SIR model and $\theta_{j,u}$ and $\theta_{j,c}$ be unconstrained and constrained parameters of the model. In order to employ gradient-based optimization algorithms (e.g. `BFGS`), we must solve for $dx_i(t, \theta)/d\theta_{j,u}$. To find the sensitivity equations, `fitsir` integrates the following set of differential equations along with the basic SIR model:

$$\begin{aligned}\frac{d}{dt}\frac{dx_i(t;\theta)}{d\theta_{j,u}} &= \left(\frac{d}{dt}\frac{dx_i(t;\theta)}{d\theta_{j,c}}\right)\frac{d\theta_{j,c}}{d\theta_{j,u}} \\ &= \left(\frac{\partial f_x}{\partial \theta_{j,c}} + \sum_i \frac{\partial f_x}{\partial x_i}\frac{dx_i(t;\theta)}{d\theta_{j,c}}\right)\frac{d\theta_{j,c}}{d\theta_{j,u}},\end{aligned}$$

where $f_x = dx/dt$. Essentially, we integrate sensitivity equations with respect to constrained parameters for simplicity but multiply $d\theta_{j,c}/d\theta_{j,u}$ after to obtain sensitiviy equations with respect to unconstrained parameters because optimization is done using unconstrained parameters.

For clarity, we write ν_{x_i,θ_j} to represent sensitivity equations with respect to constrained parameters. Then, we write

$$\nu_{x,\theta}(t; x; \theta) = \begin{bmatrix}\nu_{S,\beta} & \nu_{S,N} & \nu_{S,\gamma} & \nu_{S,i_0} \\ \nu_{I,\beta} & \nu_{I,N} & \nu_{I,\gamma} & \nu_{I,i_0}\end{bmatrix}.$$

So the sensitivity equations of the SIR model are given by

$$\frac{d}{dt}\nu_{x,\theta}(t;\cdot) = \begin{bmatrix}-\beta I/N & -\beta S/N \\ \beta I/N & \beta S/N - \gamma\end{bmatrix}\nu_{x,\theta}(t;\cdot) + \begin{bmatrix}-SI/N & \beta SI/N^2 & 0 & 0 \\ SI/N & -\beta SI/N^2 & -I & 0\end{bmatrix}.$$

The following additional equations complete the sensitivity equations:

$$\begin{aligned}\nu_{x,\theta}(0; x(0); \theta) &= \begin{bmatrix}0 & 1-i_0 & 0 & -N \\ 0 & i_0 & 0 & N\end{bmatrix} \\ \left[\frac{d\theta_{j,c}}{d\theta_{j,u}}\right]_{j=1,2,3,4} &= \begin{bmatrix}\beta & \gamma & N & (1-i_0)i_0\end{bmatrix}.\end{aligned}$$

Using the sensitivity equations of the SIR model, we can now compute the sensitivity equations of negative log-likelihood functions with respect to the unconstrained parameters by applying the chain rule. Given a log likelihood function $l(X; \mu; \theta)$, we have:

$$\frac{dl(X;\mu;\theta)}{d\theta_{j,u}} = \frac{dl}{d\mu}\frac{d\mu}{d\theta_{j,u}},$$

where X is the observed counts and μ is the expected trajectory. Precisely, μ is equal to I for prevalence fits, $S(\tau_n) - S(\tau_{n+1})$ for incidence fits, and $R(\tau_{n+1}) - R(\tau_n)$ for death fits.

2 Starting function

To avoid falling into local minima, it is important to use a good starting parameter. `fitsir` provides a function (`startfun`) that estimates a reasonable starting parameter using a built-in algorithm. This section describes the method used in findiing starting parameters.

2.1 Prevalence

Let's consider the Philadelphia 1918 Flu data. First, we start by fitting a spline to the data. This allows us to estimate the information on the curvature more easily.

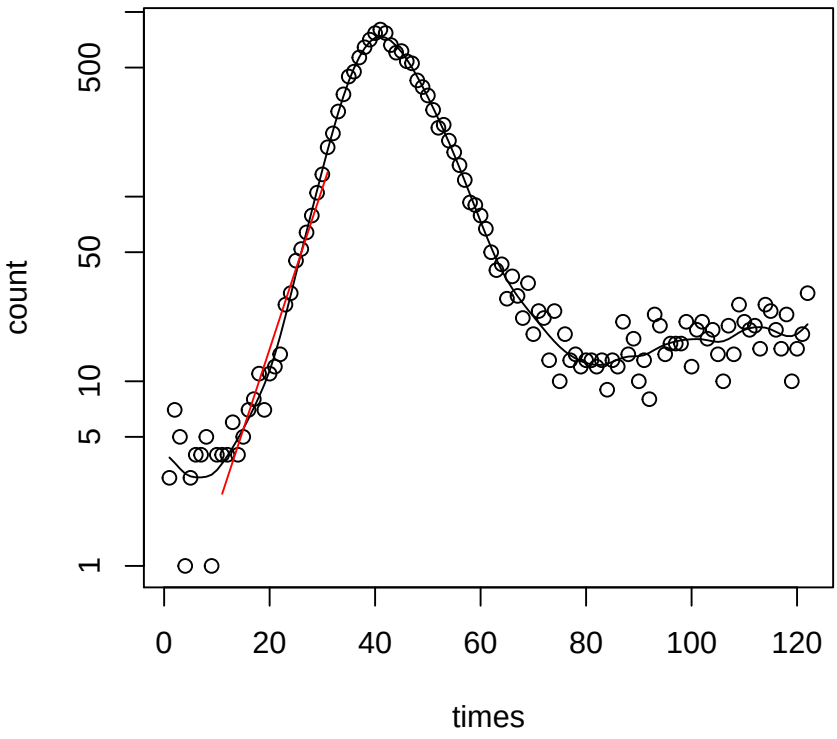
```
library(fitsir)
times <- seq_along(phila1918$date)
count <- phila1918$pim
ss <- fitsir:::smooth.spline2(times, count, itmax = 100, relpeakcrit = 0.1)
```

Then, we estimate the little r $(\beta - \gamma)$ by finding appropriate fitting window using spline fit and fitting a linear model to the real data.

```
ss.data <- data.frame(times = times, count = exp(predict(ss)$y))
ss.tmax <- ss.data$times[which.max(ss.data$count)]
ss.t1 <- min(times)+0.25*(ss.tmax-min(times))
ss.t2 <- min(times)+0.75*(ss.tmax-min(times))

m <- lm(log(count)~times,data=subset(ss.data,times<=ss.t2 & times>=ss.t1))
(r <- unname(coef(m)[2])) ##beta - gamma

## [1] 0.2005722
```



Notice that this data has trails in the beginning of the epidemic and does not follow the typical SIR shape. So taking the initial value from the data or from the spline fit will not correctly estimate the initial epidemic size. To avoid this issue, we take the y-intercept of the linear fit.

```
(iniI <- unname(exp(coef(m)[1])))

## [1] 0.2704097
```

Then, since

$$d^2 \log I/dt^2 = -\beta S'/N = \beta(-\beta SI/N)/N = -\beta^2/N^2(\gamma N/\beta)I = -\beta\gamma I/N,$$

knowing the second derivative at the peak gives us another equation. Rearranging, we get

$$c = -d^2 \log I/dt^2/I_{\text{peak}} = \beta\gamma/N$$

We obtain this quanity by using the spline fit.

```
Qp.alt <- predict(ss,ss.tmax,deriv=2)$y
Ip <- exp(max(predict(ss,times)$y))
(c <- -Qp.alt/Ip)

## [1] 4.360793e-05
```

Finally, we can obtain the fourth equation as follows:

$$\begin{aligned} dI/dt &= \beta SI/N - \gamma I \\ &= \beta(N - I - R)I/N - \gamma I \\ &= \beta\left(N - I - \int_0^t \gamma I(s)ds\right) I/N - \gamma I \\ 0 &= (\beta - \gamma)I_{\text{peak}} - \frac{\beta}{N} I_{\text{peak}}^2 - \frac{\beta\gamma}{N} I_{\text{peak}} \int_0^{t_{\text{peak}}} I(s)ds \end{aligned}$$

Integral is calculated numerically:

```
t.diff <- diff(times)
t.diff <- c(t.diff[1], t.diff)
ss.int <- transform(ss.data, int = count * t.diff)
ss.int <- ss.int[times<ss.tmax,]

d0 <- sum(ss.int[,3])
```

After rearranging the equations, we find that $r - c\int_0^{t_{\text{peak}}}$ must be negative in order to get a reasonable parameter. So we use a loop to avoid this value becoming positive:

```
while(r - c * d0 < 0){
    ss.int <- ss.int[-nrow(ss.int),]
    d0 <- sum(ss.int[,3])
}
```

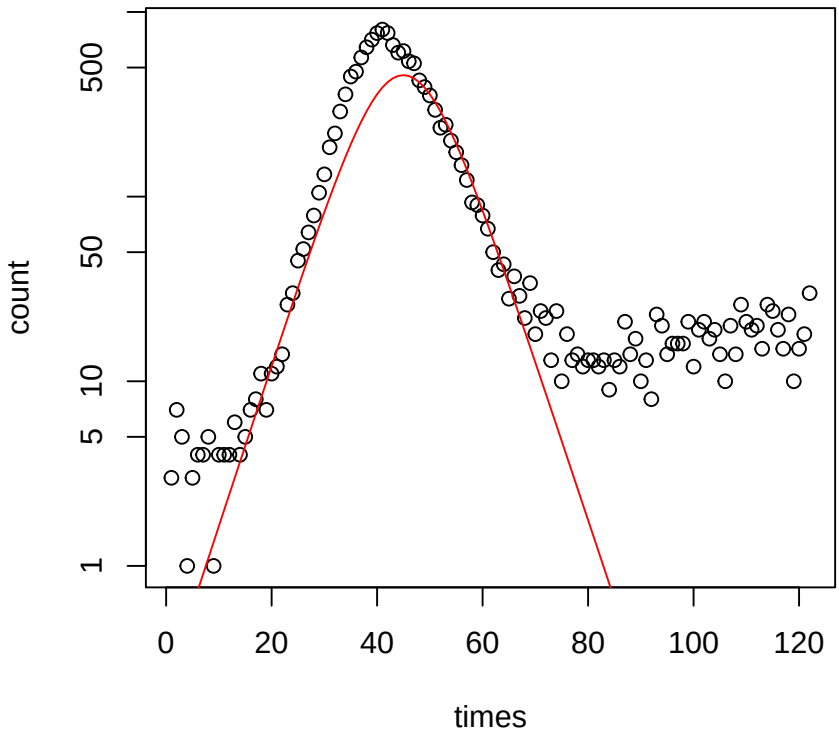
Therefore, we get

```
prev.pars <- list()
(prev.pars <- within(prev.pars,{
    gamma <- c * Ip/(r - c * d0)
    beta <- gamma + r
    N <- beta*gamma/c
    i0 <- iniI/N
}))

## $i0
## [1] 1.434909e-07
```

```
##
## $N
## [1] 1884507
##
## $beta
## [1] 9.166129
##
## $gamma
## [1] 8.965557
```

```
plot(times, count, log="y")
lines(times, SIR.detsim(times, unlist(prev.pars)[c(3, 4, 2, 1)]), col="red")
```



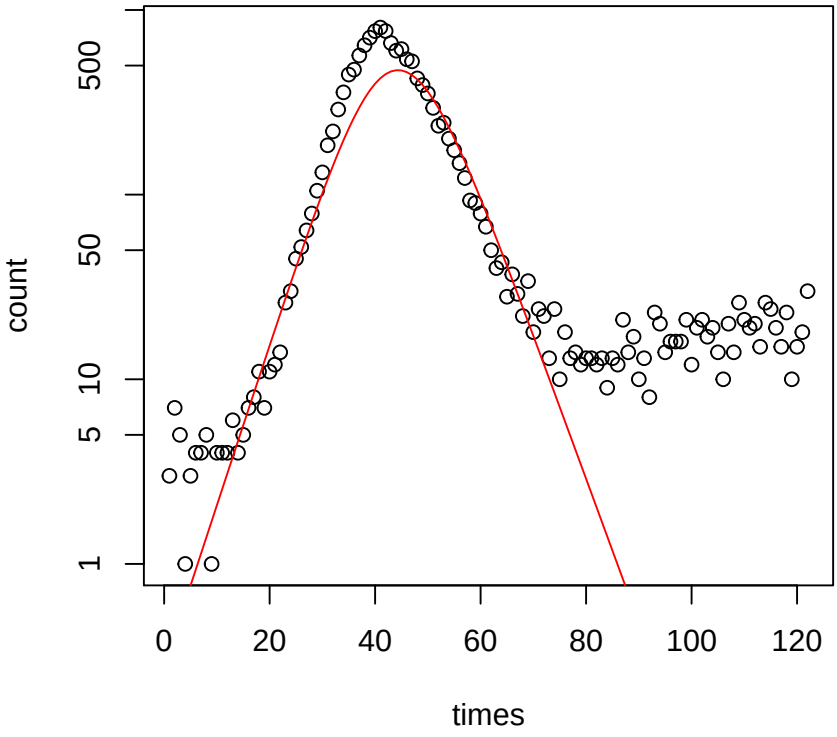
2.2 Death

Very similar to what we did so far...

```r
death.pars <- list()
(death.pars <- within(prev.pars,{
    gamma <-  c*(d0 + Ip)/r
    beta <- gamma + r
    N <- beta*gamma/c
    i0 <- iniI/N
}))

## $i0
## [1] 7.687394e-06
##
## $N
## [1] 35175.73
##
## $beta
## [1] 1.342863
##
## $gamma
## [1] 1.142291

plot(times, count, log="y")
lines(times, SIR.detsim(times, unlist(death.pars)[c(3, 4, 2, 1)], type="death"), col="red")
```



2.3 Incidence

3 Confidence intervals