Basic SIR fitting

Ben Bolker, David Earn, Dora Rosati

April 21, 2015

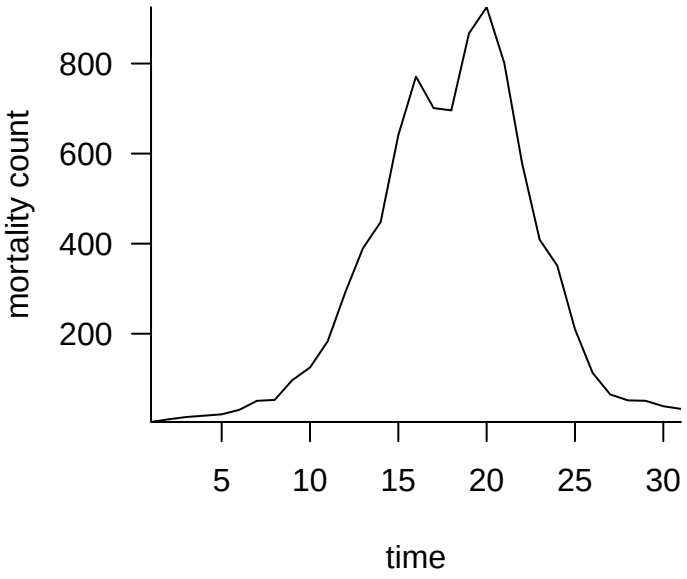
This has been done a million times, but let's try to do it in a reasonably systematic way that could be used in a pedagogical paper.

```
library("fitsir")
library("bbmle") ## need this for now, for coef()
library("plyr")
library("reshape2")
library("ggplot2"); theme_set(theme_bw())
library("RColorBrewer")
```

The current version of `fitsir` assumes that time and prevalence are stored as columns `tvec` and `count` within a data frame. Since the `bombay` data set instead has `week` (week of epidemic) and `mort` (mortality), we'll rename it for convenience. (We will for now resolutely ignore issues about fitting weekly mortality counts as prevalences ...)

```
bombay2 <- setNames(bombay,c("tvec","count"))
```

```
plot(count~tvec,data=bombay2,
     type="l",xaxs="i",yaxs="i",
     xlab="time",ylab="mortality count")
```



1 Fit the model to the data

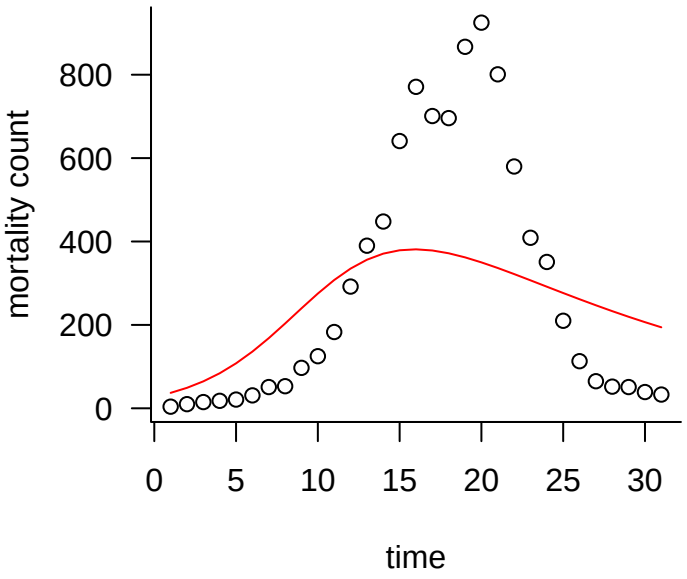
Basic fit:

```
m1 <- fitsir(data=bombay2)
```

```
summarize.pars(coef(m1))

##          R0           r      infper          i0
##  5.13774293  0.28629956 14.45249473  0.05235677
```

Seemingly reasonable answers, but ...

```
ss <- with(bombay2,SIR.detsim(tvec,trans.pars(coef(m1))))
plot(count~tvec,data=bombay2,
     xlab="time",ylab="mortality count")
lines(bombay2$tvec,ss,col=2)
```



2 Troubleshooting

We're obviously not getting a good answer here. When this happens there are a variety of possibilities.

- optimizer getting stuck
- a small number of local optima
- a large number of local optima, on many different scales (fractal-like or rugged surface)
- a large number of local optima, all similar in scale/height ("fakir's bed" geometry)

Some solutions:

- center/scale parameters and/or reparameterize the model to remove correlation and equalize scales of variation in different parameters
- try to come up with a rule for finding better starting values ("self-starting" fits)

- use a better/more robust local optimizer
- use lots of starting values, randomly or regularly distributed
- use a stochastic global optimizer

```
confint(m1,method="quad")

## Warning in sqrt(diag(object@vcov)):  NaNs produced

##                2.5 %     97.5 %
## log.beta  -1.453650 -0.6148568
## log.gamma -2.748828 -2.5929057
## log.N            NaN        NaN
## logit.i   -4.049810 -1.7419839
```

Suggests *some* sort of unidentifiability ...

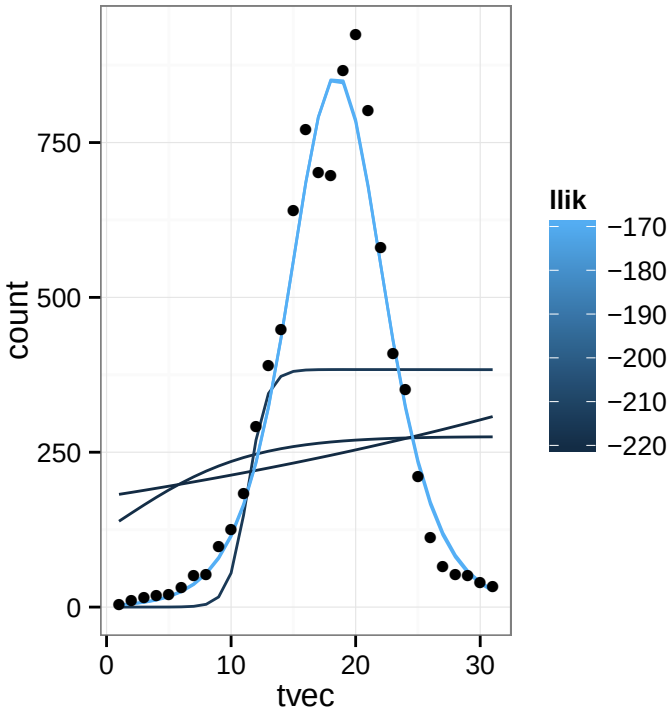
What if we try a bunch of starting values?

A crude Latin-hypercube-like strategy: pick evenly spaced values on sensible log scales, then permute to get random (but even) coverage of the space.

```
qlhcfun <- function(n=5,seed=NULL) {
    require("plyr")
    if (!is.null(seed)) set.seed(seed)
    R0vec <- 1+10^seq(-1,1.5,length=n)
    infpervec <- sample(10^seq(-1,2,length=n))
    Nvec <- sample(10^seq(2,5,length=n))
    i0vec <- sample(10^seq(-3,-1,length=n))
    startlist <- alply(cbind(R0=R0vec,infper=infpervec,N=Nvec,i0=i0vec),1,
                       function(x) {
                           with(as.list(x), {
                               beta <- R0/infper
                               gamma <- 1/infper
                               c(log.beta=log(beta),log.gamma=log(gamma),
                                 log.N=log(N),logit.i=qlogis(i0))
                           })
                       })
    return(startlist)
}
startlist <- qlhcfun(n=5,seed=101)
```

```
fitlist <- llply(startlist,fitsir,data=bombay2,
      method="Nelder-Mead",control=list(maxit=1e5))
```

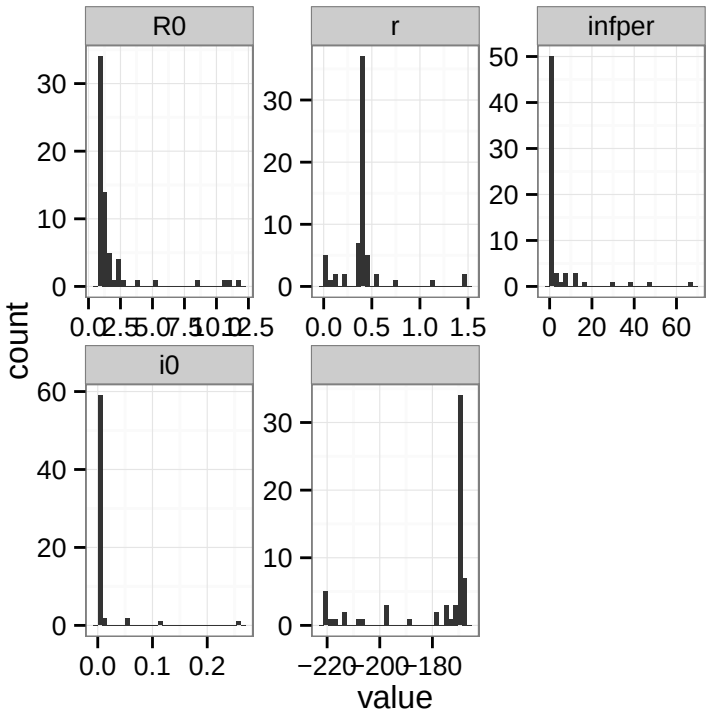
```
## extract log-likelihoods
likframe <- data.frame(.id=1:5,llik=unlist(llply(fitlist,logLik)))
## compute trajectories
gettraj <- function(x,tvec=bombay2$tvec) {
    data.frame(tvec=tvec,
               count=SIR.detsim(tvec,trans.pars(coef(x))))
}
fittraj <- ldply(fitlist,gettraj)
fittraj <- merge(fittraj,likframe)
## plot together
ggplot(fittraj,aes(tvec,count,colour=llik,group=.id))+geom_line()+
    geom_point(data=bombay2,colour="black",aes(group=NA))
```



Now try a much larger sample:

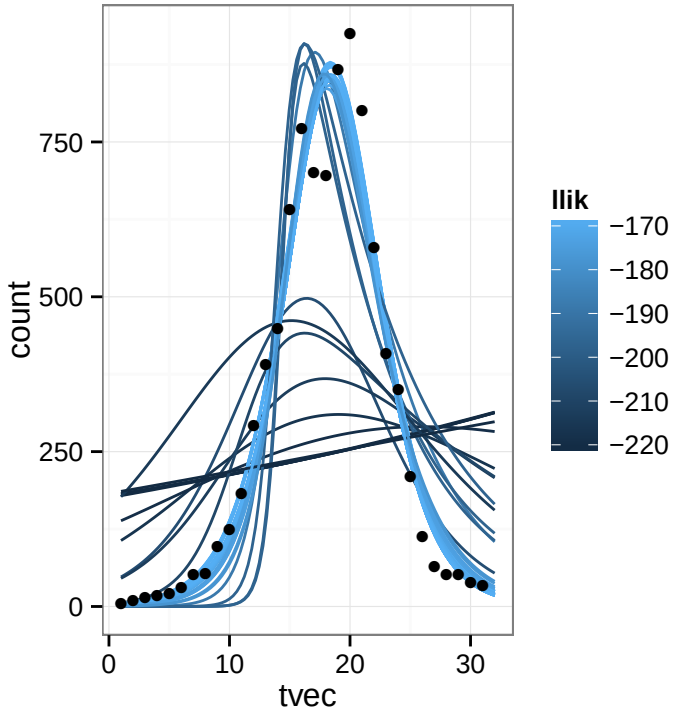
```
startlist100 <- qlhcfun(n=100,seed=101)
fitlist100 <- llply(startlist100,
                    function(x) {
                        r <- try(fitsir(start=x,data=bombay2),silent=TRUE)
                        if (is(r,"try-error")) NULL else r
                    })
```

```
testOK  <- function(x,max.R0=100,max.r=1000,max.infper=400) {
    if (is.null(x)) return(FALSE)
    ss <- summarize.pars(coef(x))
    return(ss["R0"]<max.R0 & ss["r"]<max.r & ss["infper"] < max.infper)
}
fitlist100.OK <- fitlist100[sapply(fitlist100,testOK)]
length(fitlist100.OK)

## [1] 65

fittab <- laply(fitlist100.OK,function(x) c(summarize.pars(coef(x)),logLik(x)))
ggplot(melt(fittab),aes(x=value))+geom_histogram()+facet_wrap(~Var2,scale="free")

## Warning:  position_stack requires constant width:  output may be
incorrect
```



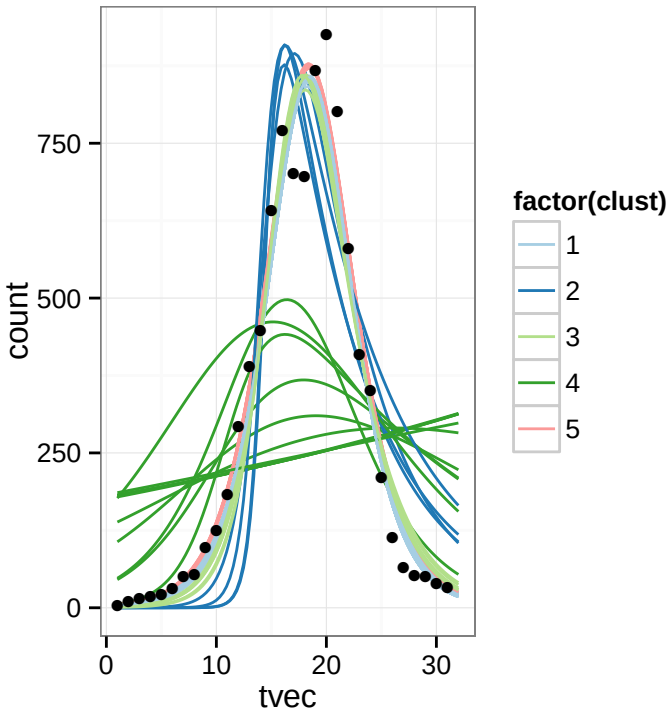

```
likframe100 <- setNames(ldply(fitlist100.OK,logLik),c(".id","llik"))
fittraj100 <- ldply(fitlist100.OK,gettraj,tvec=seq(1,32,length=101))
fitmat100 <- acast(fittraj100,tvec~.id,value.var="count")
fittraj100 <- merge(fittraj100,likframe100)
## plot together
```

```
ggplot(fittraj100,aes(tvec,count,colour=llik,group=.id))+geom_line()+
    geom_point(data=bombay2,colour="black",aes(group=NA))
```



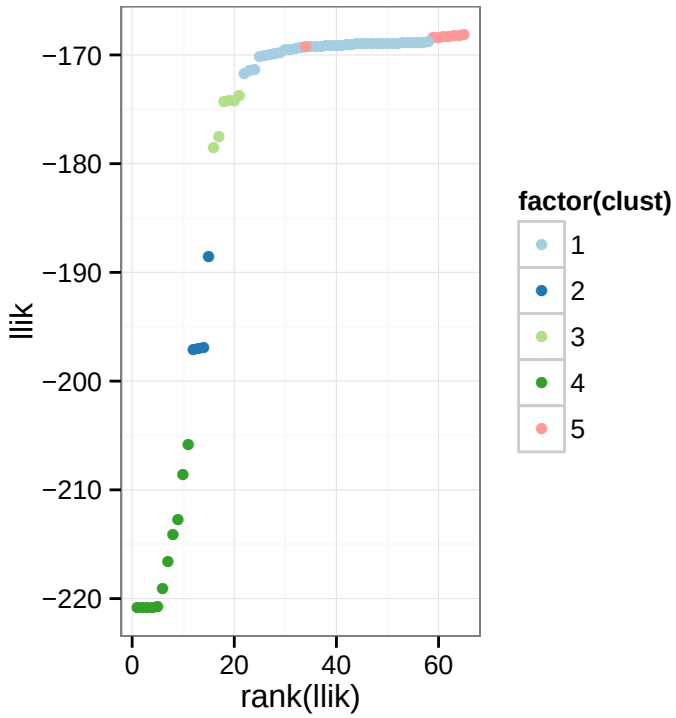
We can identify clusters ...

```
clust <- kmeans(t(fitmat100),5)
cframe <- data.frame(.id=names(clust$cluster),clust=clust$cluster)
fittraj100B <- transform(fittraj100,llikcat=cut_number(llik,5))
fittraj100B <- merge(fittraj100B,cframe)
ggplot(fittraj100B,aes(tvec,count,colour=factor(clust),group=.id))+geom_line()+
    geom_point(data=bombay2,colour="black",aes(group=NA))+
        scale_colour_brewer(palette="Paired")
```



Check out clustering on log-likelihood cumulative distribution curve:

```
dd2 <- merge(cframe,likframe100)
(g1 <- ggplot(dd2,aes(rank(llik),llik,colour=factor(clust)))+geom_point(size=2)+
              scale_colour_brewer(palette="Paired"))
```



I'm not 100% sure (yet) what this tells us. The clusters aren't so well separated that I necessarily believe that they are distinct modes.

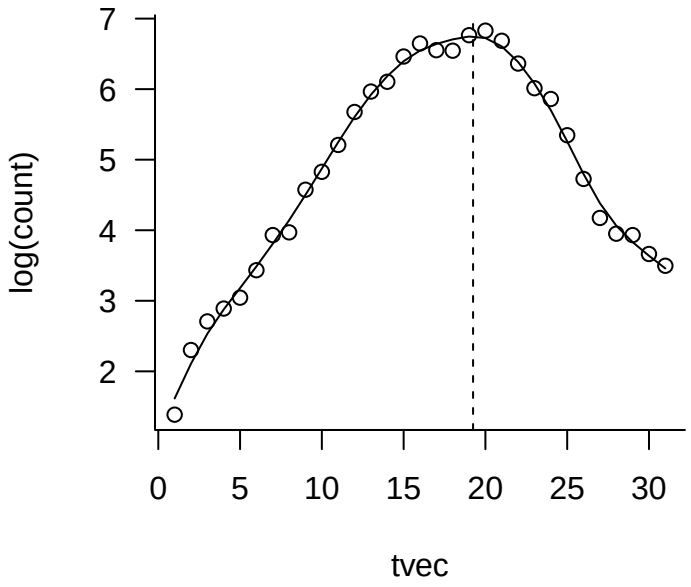
```
startmat100 <- do.call(rbind,startlist100)
```

to do: characterize starting value sets by cluster (or "bad"), plot, look for regularities. Suspect that large i_0 is a problem?

3 Self-starting strategies

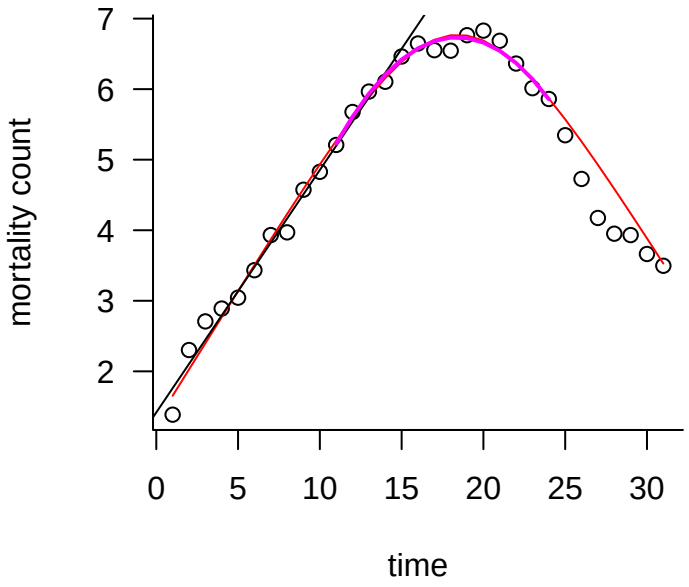
Try `smooth.spline` with `spar=0.5` to identify max.; linear regression through times up to 1/2 tmax to identify $i(0)N$ and r; then try a range of other parameters?

```
tvec <- bombay2$tvec
ss <- with(bombay2,smooth.spline(tvec,log(count),spar=0.5))
ss.tmax <- uniroot(function(x) predict(ss,x,deriv=1)$y,c(0,40))$root
plot(log(count)~tvec,data=bombay2)
lines(predict(ss,tvec))
abline(v=ss.tmax,lty=2)
```




```
ss.thalf <- min(tvec)+(ss.tmax-min(tvec))/2
```

```
plot(log(count)~tvec,data=bombay2,
     xlab="time",ylab="mortality count")
bestFit <- fitlist100.OK[[which.max(likframe100$llik)]]
bestTraj <- gettraj(bestFit)
m1 <- lm(log(count)~tvec,data=subset(bombay2,tvec<ss.thalf))
c(linfit=coef(m1)[2],
  sirfit=with(as.list(coef(bestFit)),exp(log.beta)-exp(log.gamma)))

## linfit.tvec      sirfit
##   0.3436854   0.3723578

with(bestTraj,lines(log(count)~tvec,col=2))
abline(m1)
m4 <- lm(log(count)~poly(tvec,2,raw=TRUE),data=subset(bombay2,tvec>10 & tvec<25))
lines(11:24,predict(m4),col=6,lwd=2)
```



Looks like this works. Is there a way to get a crude starting guess for R_0 and N?

Quadratic fit to peak of log trajectory is very good: what do these parameters tell us about the epidemic?

We want $d^2(\log I)/dt^2$ at the peak ... $d\log I/dt = \beta S/N - \gamma$ and $\hat{S} = \gamma N/\beta$ so we have (at the peak where $d\log I/dt = 0$)

$$d^2 \log I/dt^2 = \beta S'/N = \beta(-\beta SI/N)/N = -\beta^2/N^2(\gamma N/\beta)I = -\beta\gamma I/N \quad (1)$$

Second derivative of $a + bt + ct^2 = 2c$

```
Qp <- unname(2*coef(m4)[3])
Qp.alt <- predict(ss,ss.tmax,deriv=2)$y
Ip <- max(predict(m4))  ## peak log(I) by smoothing
(Qfits <- c(quadfit=-Qp,ssderiv2=-Qp.alt,
  sirfit=with(as.list(coef(bestFit)),exp(log.beta+log.gamma+Ip-log.N))))

##    quadfit   ssderiv2     sirfit
## 0.05537391 0.07264227 0.06565816
```

OK, I guess, although I expected a little better? Second derivative at the smoothing spline peak is a little bit easier (we don't have to decide on a range

over which to fit the quadratic), and in this case is actually closer to the theoretical value (proportional error 0.11 vs. -0.16) although it's possibly more sensitive to weird shapes at the peak.

This should get us one more parameter.

With Q, a_0, and b_0, this gives me so far:

$$\begin{aligned} a_0 &= \log i_0 + \log N \quad \text{(initial number infected, log scale)} \\ b_0 &= \log \beta - \log \gamma \quad (r) \\ \log(-Q) - \log(I_{\max}) &= \log \beta + \log \gamma + \log N \end{aligned} \tag{2}$$

$S = \gamma N/\beta$ at the peak time is approximately $N - I_0 - \sum_{t=0}^{\hat{t}} S(t)$ (we have to be careful to decide whether we're counting incidence or prevalence, and correct for γ accordingly: prevalence = incidence/γ).

Number of counts up to peak:

```
sumcount.tmax <- with(subset(bombay2,tvec<ss.tmax),sum(count))
```

Should be able to do something with this??

Should use $N(1 - i_0)$, not N, as starting condition for S.