Introduction to inference: parameter estimation

ICTP Workshop on Mathematical Models of Climate Variability, Environmental Change and Infectious Diseases

Aaron A. King

8–19 May 2017

Important Note: These materials have been updated for use with version 2.8. As of version 2, **pomp** syntax has changed substantially. These changes are documented (http://kingaa.github.io/pomp/vignettes/upgrade_guide.html) on the **pomp** website.

Introduction

This course focuses on the use of models for understanding, predicting, and controlling ecological and epidemiological systems. Models play a central role in this work because they allow us to precisely and quantitatively express our ideas about mechanisms. To the extent our understanding of the important mechanisms is correct, such models are extremely useful in the design of policy. On the other hand, models are useful in uncovering the important mechanisms, inasmuch as we can compare model predictions to data. Specifically, if we can translate competing hypotheses into mathematical models, these can be compared in terms of their ability to correctly capture the patterns seen in data.

In order to fairly compare competing models, we must first try to find out what is the best each model can do. Practically speaking, this means that we have to find the values of the models' parameters that give the closest correspondence between model predictions and data. Parameter estimation can be challenging even when we are fairly confident in the ability of a single model to explain the dynamics.

The SIR model

As a simple example for use in this lesson, we'll focus on the classical SIR model (Kermack and McKendrick 1927). This model's simplicity allows us to postpone many of the complexities that will arise as we grapple with real data. However, it enjoys the feature that many complexities can be incorporated as modifications and extensions of the model. The model divides a population of hosts into three classes: susceptible, infected, recovered. The model describes how the portion of the population in each of these classes changes with time. Births are modeled as flows from "elsewhere" into the susceptible class; deaths are modeled as flows from the S, I, or R compartment into "elsewhere". If S, I, and R refer to the numbers of individuals in each compartment, then these **state variables** change according to the following system of differential equations:

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

Here, B is the crude birth rate (births per unit time), μ is the death rate and γ is the recovery rate. We'll assume that the force of infection, λ, has the form

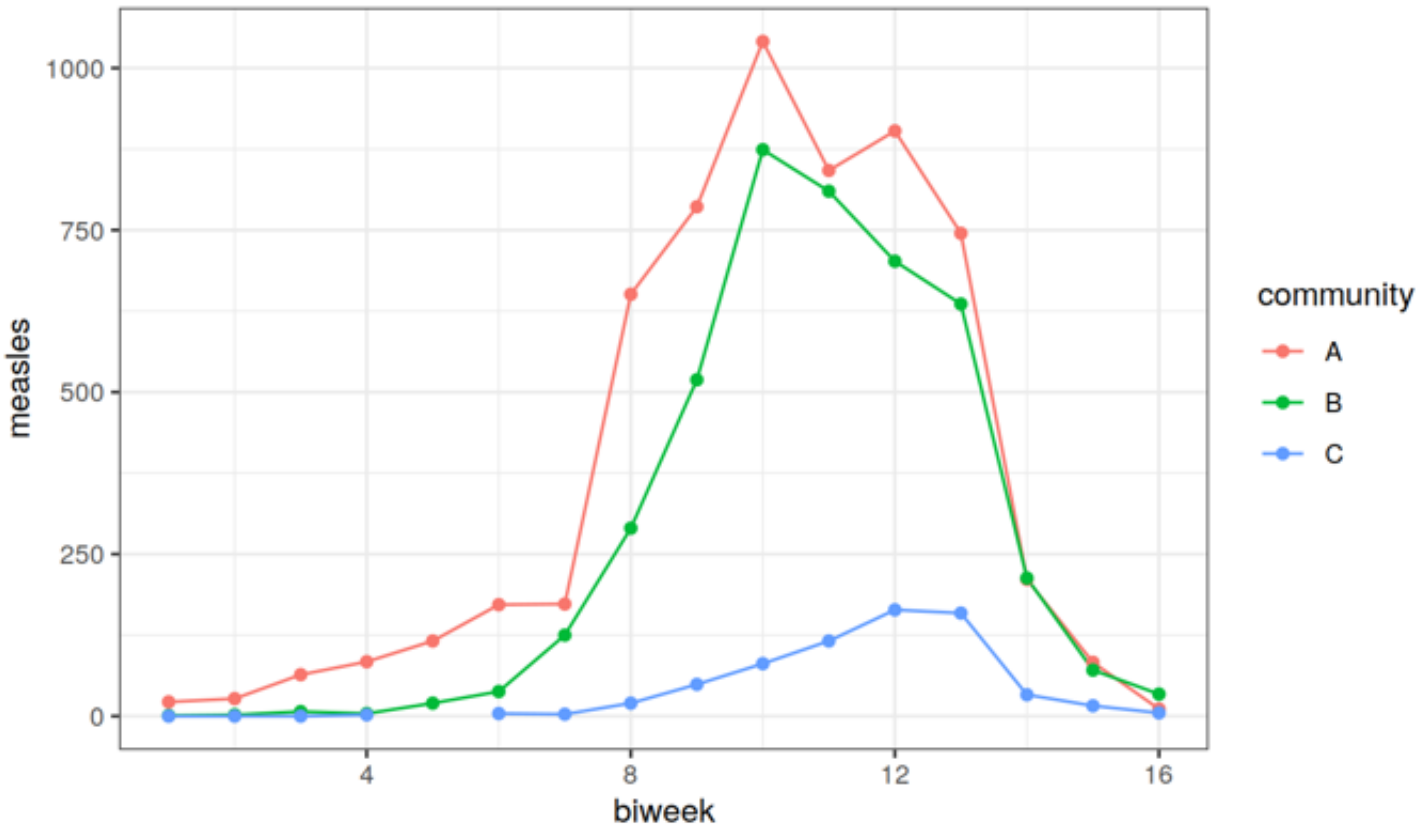
$$\lambda = \beta \frac{I}{N},$$

so that the risk of infection a susceptible faces is proportional to the *prevalence* (the fraction of the population that is infected). This is known as the assumption of frequency-dependent transmission.

Data from an outbreak of measles in Niamey

Biweekly data for outbreaks of measles in three communities within Niamey, Niger (Grais et al. 2006) are provided on the course website. To download and plot the data, do, e.g.,

```
niamey <- read.csv("http://kingaa.github.io/clim-dis/parest/niamey.csv")
ggplot(niamey,mapping=aes(x=biweek,y=measles,color=community))+
  geom_line()+geom_point()
```



Integrating differential equations with **pomp**

Although focused on stochastic models, **pomp** provides facilities for dealing with the important special case of deterministic dynamics. The “Working with ODE models” tutorial (./odes.html) shows how deterministic models are implemented and solved.

Feature-based parameter estimation

A classical approach to the estimation of parameters is to identify informative features of a dataset and then choose parameters in a model so as to match those features. This *feature matching* approach is sometimes known as the *generalized method of moments* and is experiencing something of a revival in recent years. Here, we give some examples of this approach in the specific context of a single epidemic curve, modeled as a closed SIR epidemic.

Estimating R_0 in an invasion

During the early stages of an SIR outbreak, the number of infected individuals I at time t is approximately

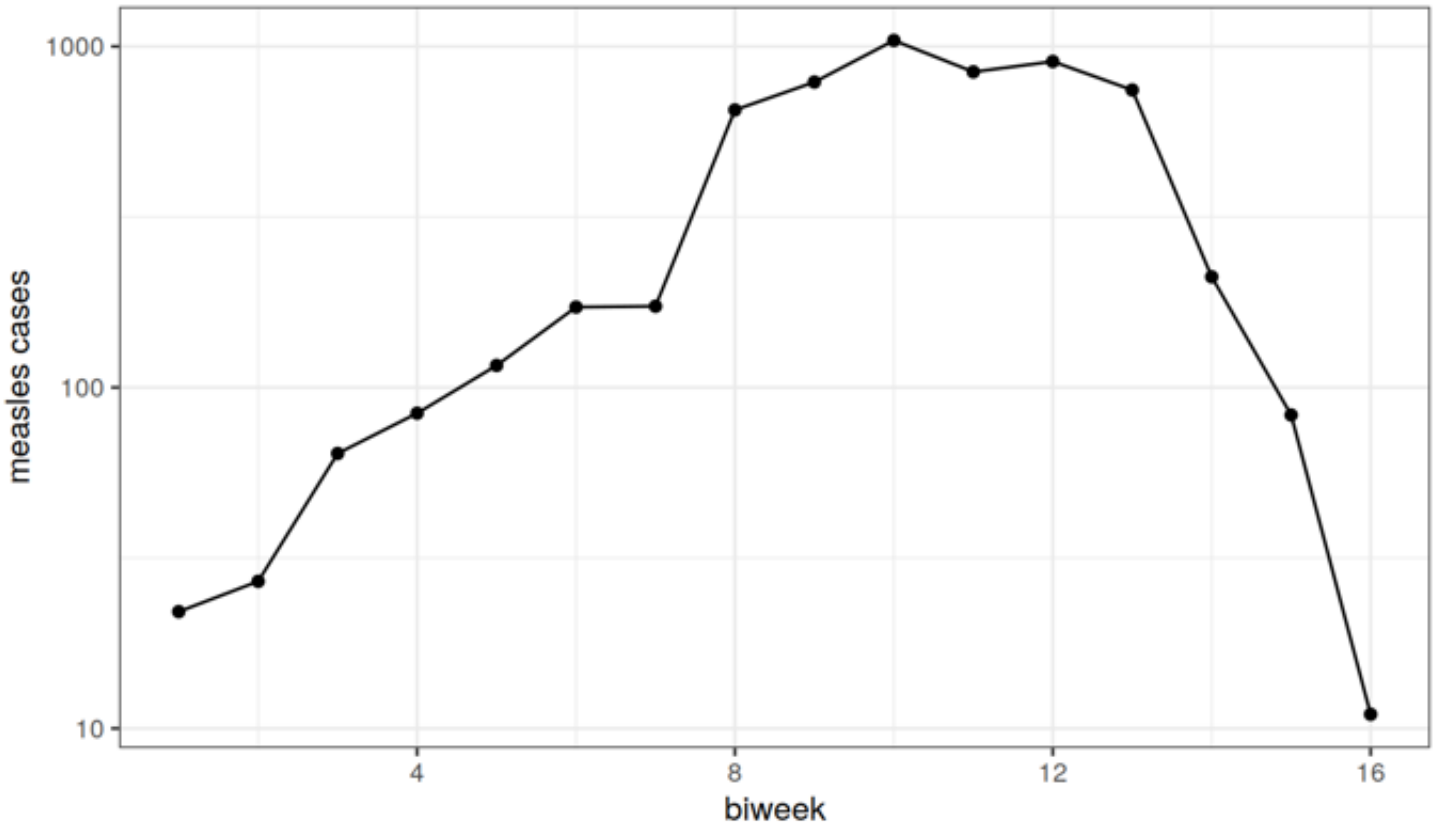
$$I(t) \approx I_0\, e^{(R_0-1)\,(\gamma+\mu)\,t}$$

where I_0 is the (small) number of infectives at time 0, $\frac{1}{\gamma}$ is the infectious period, and $\frac{1}{\mu}$ is the host lifespan. Taking logs of both sides, we get

$$\log I \approx \log I_0 + (R_0 - 1)(\gamma + \mu) t,$$

which implies that a semi-log plot of I vs t should be approximately linear with a slope proportional to $R_0 - 1$ and the recovery rate.

Let us plot the Niamey measles data (Grais et al. 2006) from community "A" in this way to see if this is the case.



Plotted on a log scale, the linearity of the first several data points is evident. This suggests that we can obtain a cheap and cheerful estimate of R_0 by a simple linear regression:

```
fit1 <- lm(log(measles)~biweek,data=subset(niamey,biweek<=8&community=="A"))
summary(fit1)
```

```
##
## Call:
## lm(formula = log(measles) ~ biweek, data = subset(niamey, biweek <=
##     8 & community == "A"))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.49040 -0.09510  0.00664  0.12315  0.40282
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.61968    0.22727  11.527 2.56e-05 ***
## biweek       0.43200    0.04501   9.599 7.31e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2917 on 6 degrees of freedom
## Multiple R-squared:  0.9389, Adjusted R-squared:  0.9287
## F-statistic: 92.13 on 1 and 6 DF,  p-value: 7.312e-05
```

```
coef(fit1)
```

```
## (Intercept)      biweek
##   2.6196750   0.4320018
```

```
slope <- coef(fit1)[2]; slope
```

```
##    biweek
## 0.4320018
```

Now, we know that measles' infectious period is about 2 wk. Moreover, since this is far shorter than an average human life ($\mu \ll \gamma$), we can neglect μ in our estimating equation. Thus our estimate of R_0 is

$$\hat{R_0} \;=\; \text{slope}/\gamma + 1 \;\approx\; 0.43 \times 1 + 1 \;\approx\; 1.4.$$

Our strategy in this case has been to redefine the problem so that it fits a standard form, i.e., we showed how to rearrange the model so that the relevant quantity (R_0) could be obtained from linear regression. This means that all the usual diagnostics associated with linear regression are also available to check the fit of the model. We can get a rough estimate of the uncertainty in our estimate by looking at the standard errors in our estimator.

```
coef(summary(fit1))
```

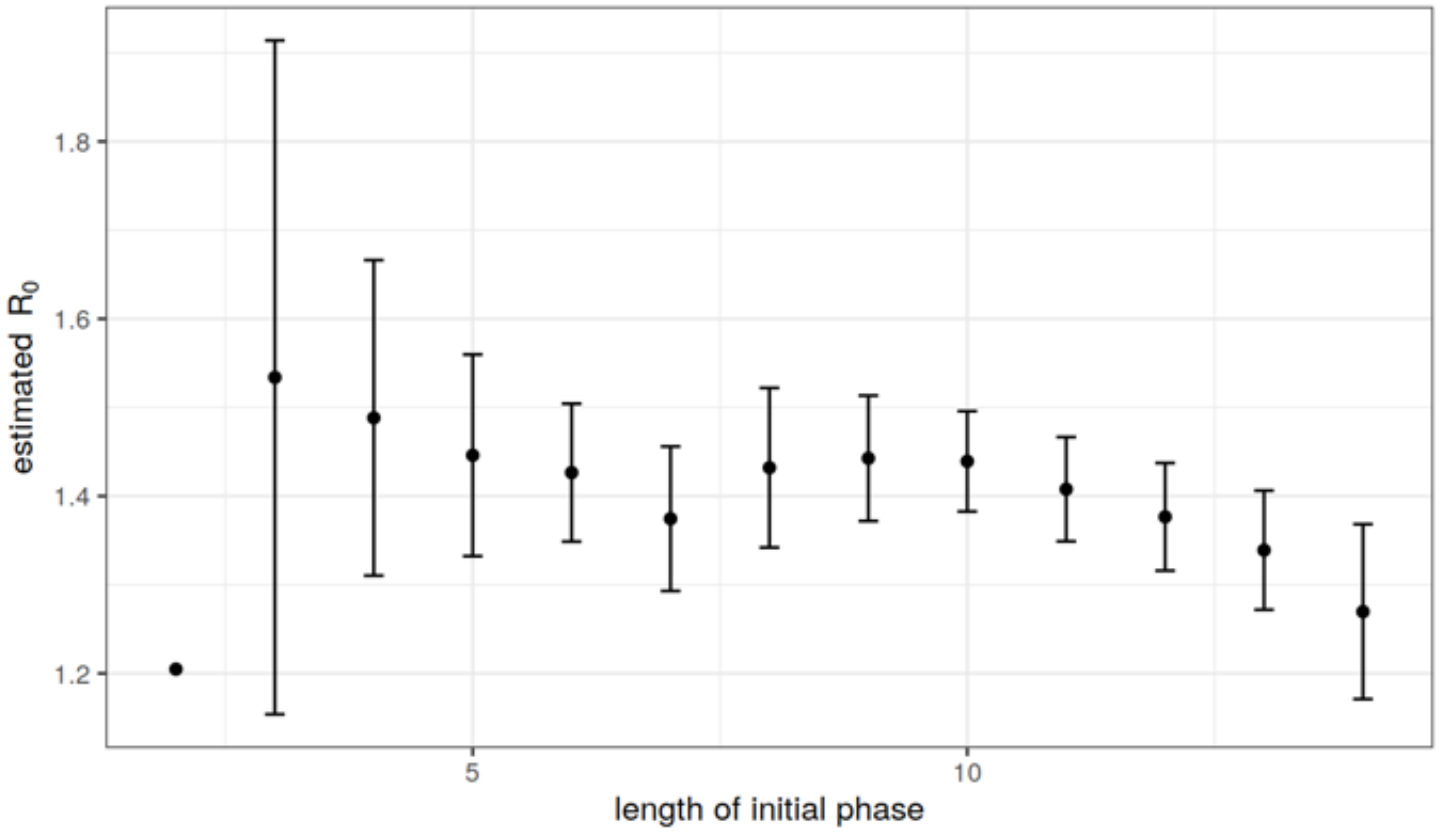
```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2.6196750 0.22727156 11.526629 2.562760e-05
## biweek      0.4320018 0.04500648  9.598659 7.312397e-05
```

```
slope.se <- coef(summary(fit1))[2,2]
1*slope.se
```

```
## [1] 0.04500648
```

So we reckon we've got an error of ± 0.05 in our estimate of R_0, i.e., we feel pretty confident that $1.34 < R_0 < 1.52$.

A defect of this method is that it uses only a small amount of the data to compute an important quantity. Moreover, we have to make a subjective judgement as to how much of the data to use. Further, as we use more data, and presumably obtain more precise estimates, we simulataneously get further from the realm where our approximation is valid, which introduces greater bias. Let's see how our estimates of R_0 depend on what we choose to be the "initial phase" of the outbreak. Below, we estimate R_0 and its standard error using the first 2, 3, 4, …, 10 data points. We then plot these estimates with their uncertainties to show this precision-accuracy tradeoff.



Estimating R_0 from the final size

If we know the final size of an outbreak and the total population size, we can use the final size equation (./odes.html#the-epidemic-final-size) to estimate R_0.

Exercise: estimating R_0 for measles in Niamey

Combine the final-size and the invasion rate methods to obtain estimates of R_0 and N using the data from each of the communities of Niamey, assuming that the infectious period is approximately two weeks.

Fitting deterministic dynamical epidemiological models to data

Least squares

We now move on to a much more general but considerably more complicated technique for estimating R_0. The method of *least squares* gives us a way to quantify the discrepancy between the data and a model's predictions. We can then search over all possible values of a model's parameters to find the parameters that minimize this discrepancy.

We'll illustrate this method using the Niamey data. Since this is a single outbreak, and the number of births and deaths into the population over the course of the outbreak is small relative to the size of the population, we can treat this outbreak as if it were occurring in a closed population.

Thus far, we have only considered deterministic models. In the next lesson, we will begin to think about more realistic models that begin to take into account some aspects of the stochastic nature of real epidemics. For now, under the assumption that the epidemic is deterministic, parameter estimation is a matter of finding the model trajectory that gives the best fit to the data. The first thing we need is a `pomp` object encoding the model and the data.

```
pomp(
  data=subset(niamey,community=="A",select=-community),
  times="biweek",t0=0,
  skeleton=vectorfield(
    Csnippet("
      DS = -Beta*S*I/N;
      DI = Beta*S*I/N-gamma*I;
      DR = gamma*I;")),
  rinit=Csnippet("
      S = S_0;
      I = I_0;
      R = N-S_0-I_0;"),
  statenames=c("S","I","R"),
  paramnames=c("Beta","gamma","N","S_0","I_0")) -> niameyA
```

Now we set up a function that will calculate the sum of the squared errors (SSE), or discrepancies between the data and the model predictions.

```
sse <- function (params) {
  x <- trajectory(niameyA,params=params)
  discrep <- x["I",,]-obs(niameyA)
  sum(discrep^2)
}
```

To get a sense of what this gives us, let's vary some parameters and computing the SSE for community "A" of the Niamey data set. To begin with, we'll assume we know that $\gamma = 1$ and that the initial numbers of susceptibles, infectives, and recovereds, $S(0)$, $I(0)$, $R(0)$ are known to be 10000, 10, and 20000, respectively. We'll write a little function that will plug a value of β into the parameter vector and compute the SSE.

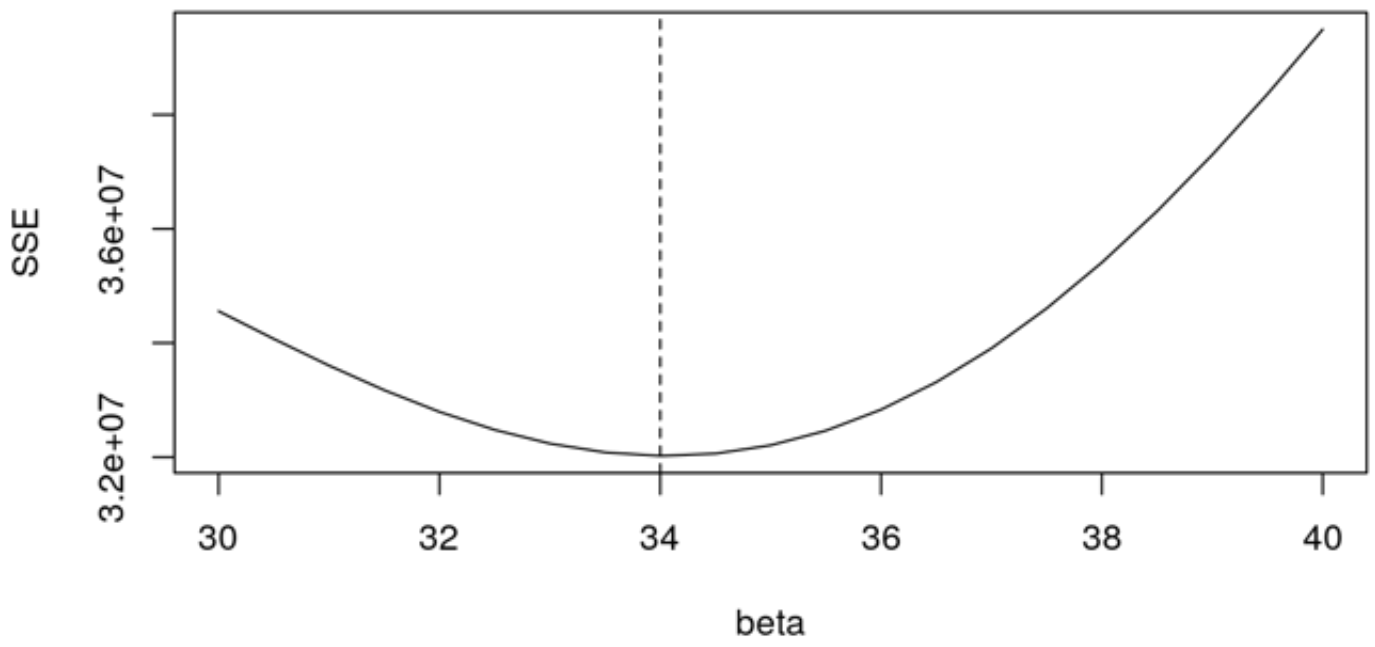
```
f1 <- function (beta) {
  params <- c(Beta=beta,gamma=1,N=50000,S_0=10000,I_0=10)
  sse(params)
}
beta <- seq(from=30,to=40,by=0.5)
SSE <- sapply(beta,f1)
```

We take our estimate, $\hat{\beta}$ to be the value of β that gives the smallest SSE,

```
beta.hat <- beta[which.min(SSE)]
```

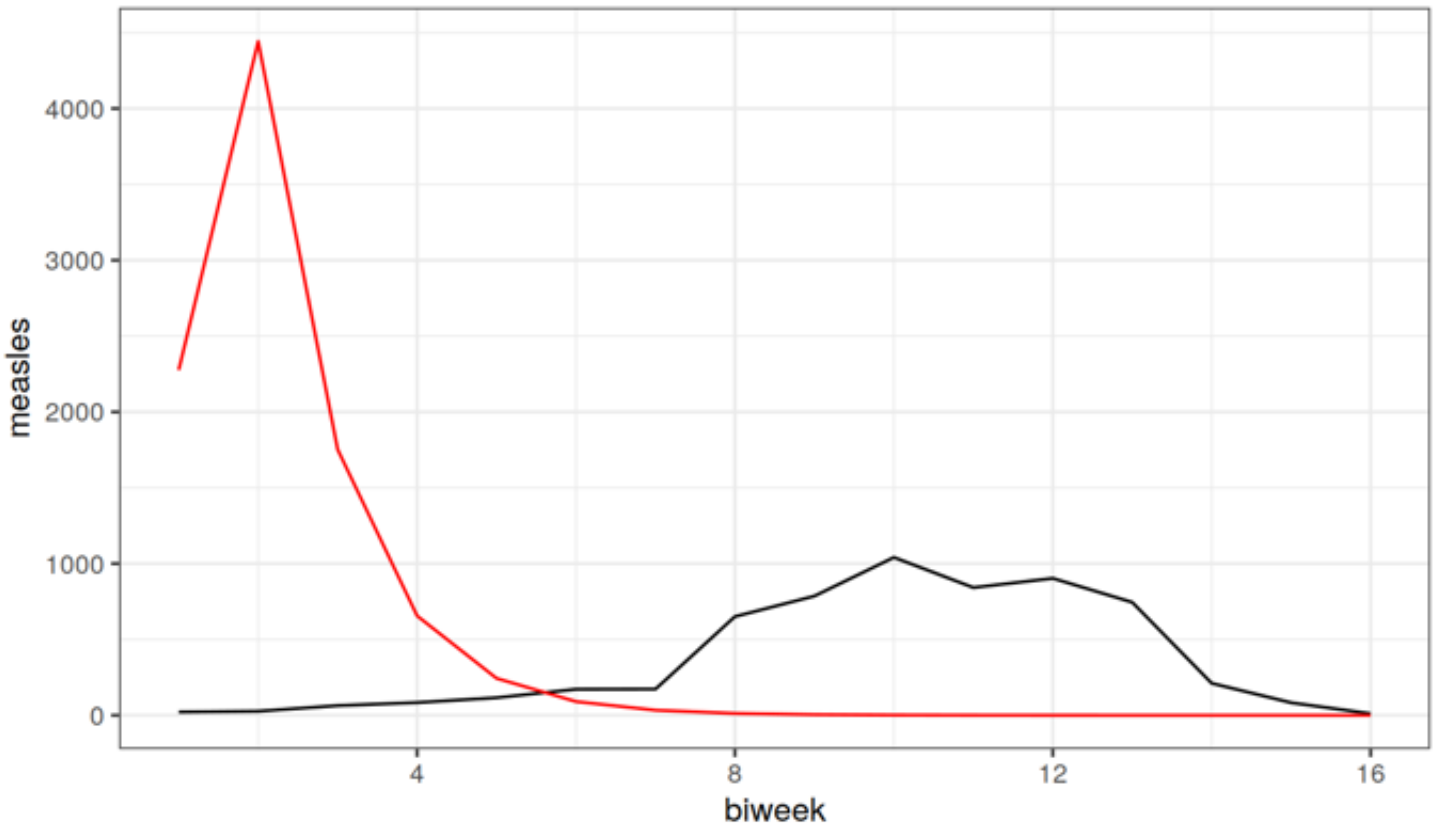
and plot SSE vs. β:

```
plot(beta,SSE,type='l')
abline(v=beta.hat,lty=2)
```

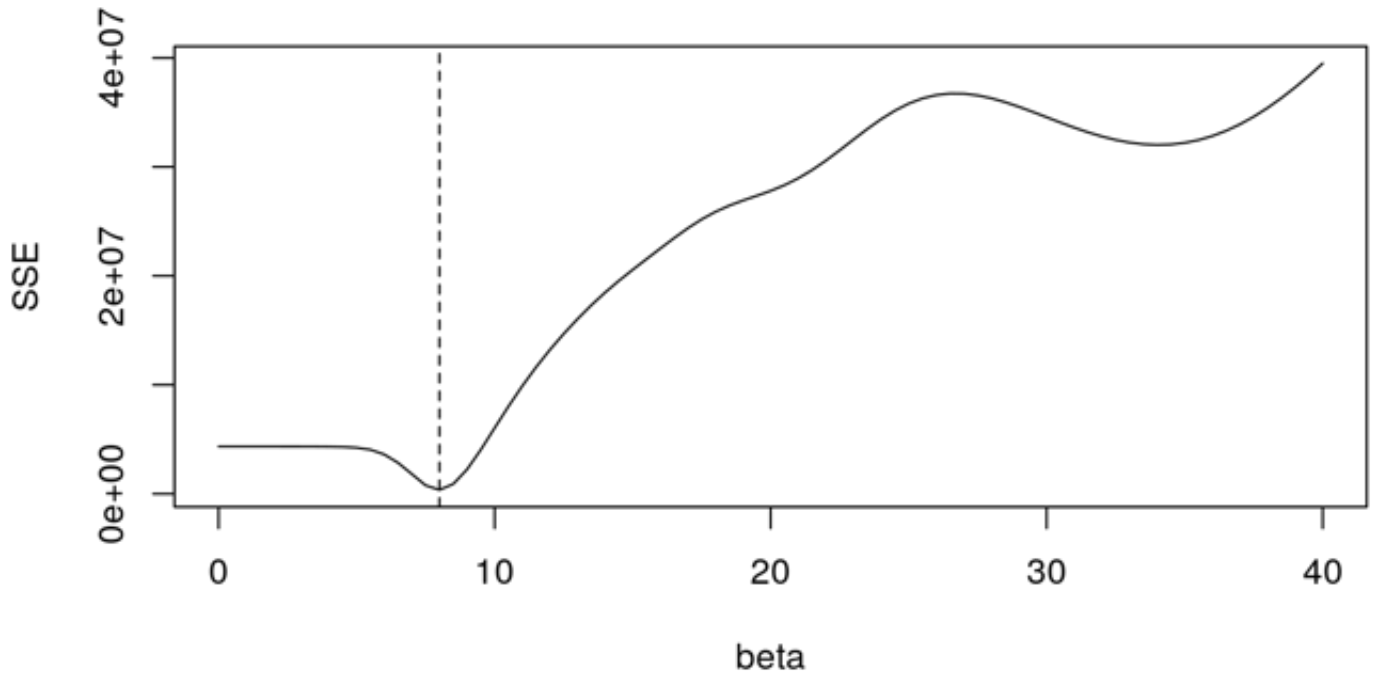


What does the SIR model predict at $\beta = \hat{\beta}$? We compute the model's trajectory to find out:

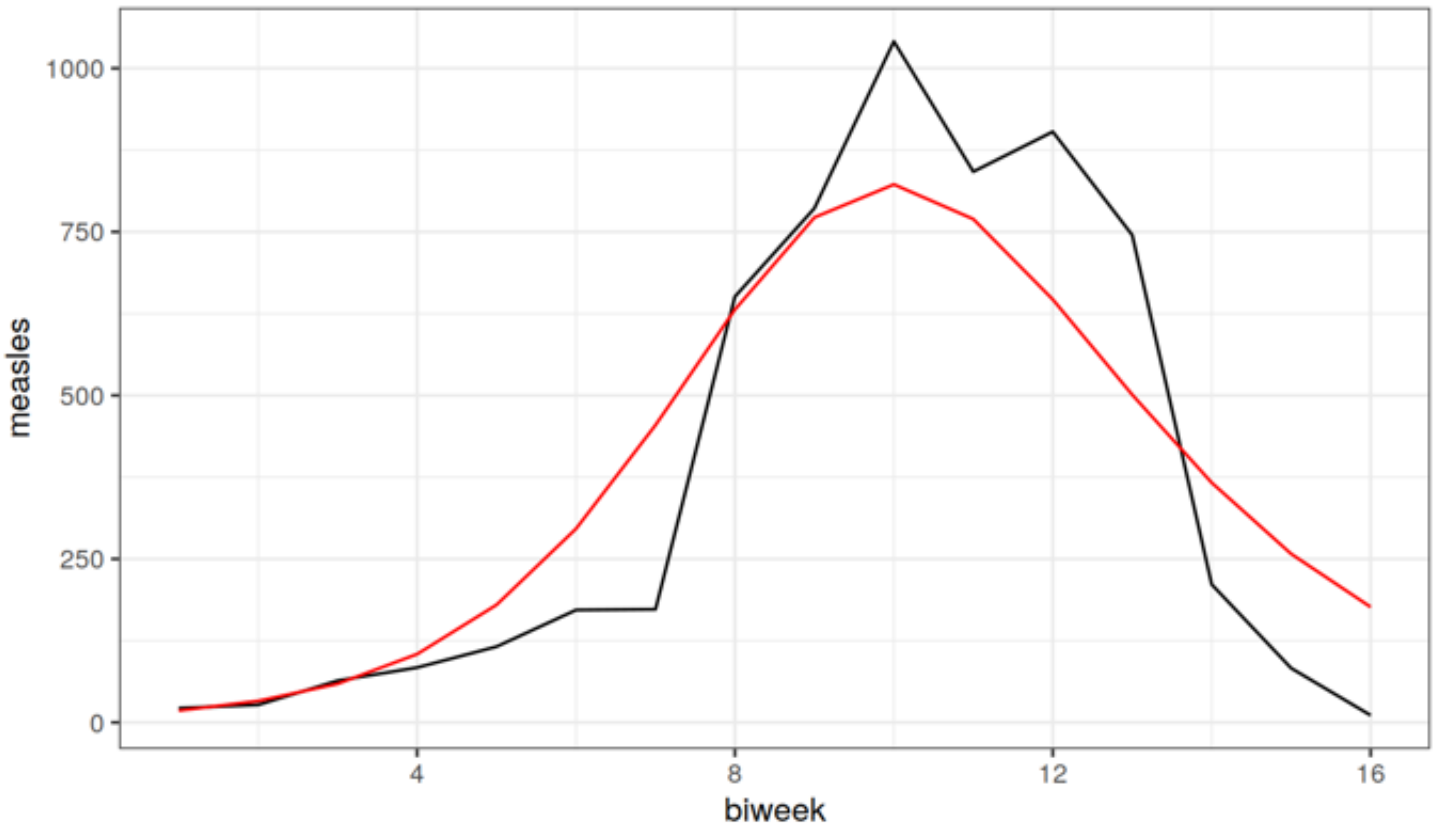
```
coef(niameyA) <- c(Beta=beta.hat,gamma=1,N=50000,S_0=10000,I_0=10)
x <- trajectory(niameyA,format="data.frame")
ggplot(data=join(as.data.frame(niameyA),x,by='biweek'),
       mapping=aes(x=biweek))+
  geom_line(aes(y=measles),color='black')+
  geom_line(aes(y=I),color='red')
```




```
beta <- seq(from=0,to=40,by=0.5)
SSE <- sapply(beta,f1)

plot(beta,SSE,type='l')
beta.hat <- beta[which.min(SSE)]
abline(v=beta.hat,lty=2)
```




```
coef(niameyA,"Beta") <- beta.hat
x <- trajectory(niameyA,format="data.frame")
dat <- join(as.data.frame(niameyA),x,by='biweek')

ggplot(dat,aes(x=biweek))+
  geom_line(aes(y=measles),color='black')+
  geom_line(aes(y=I),color='red')
```



Exercise: least-squares estimation of R_0

Use this method to obtain estimates of R_0 for measles from each of the three communities in Niamey. You may again assume that the infectious period is two weeks.

Exercise: interpreting local optima

The above plot of SSE against β shows a second local minimum of the SSE at a much higher value of β. Why is this?

Exercise: local vs global optima

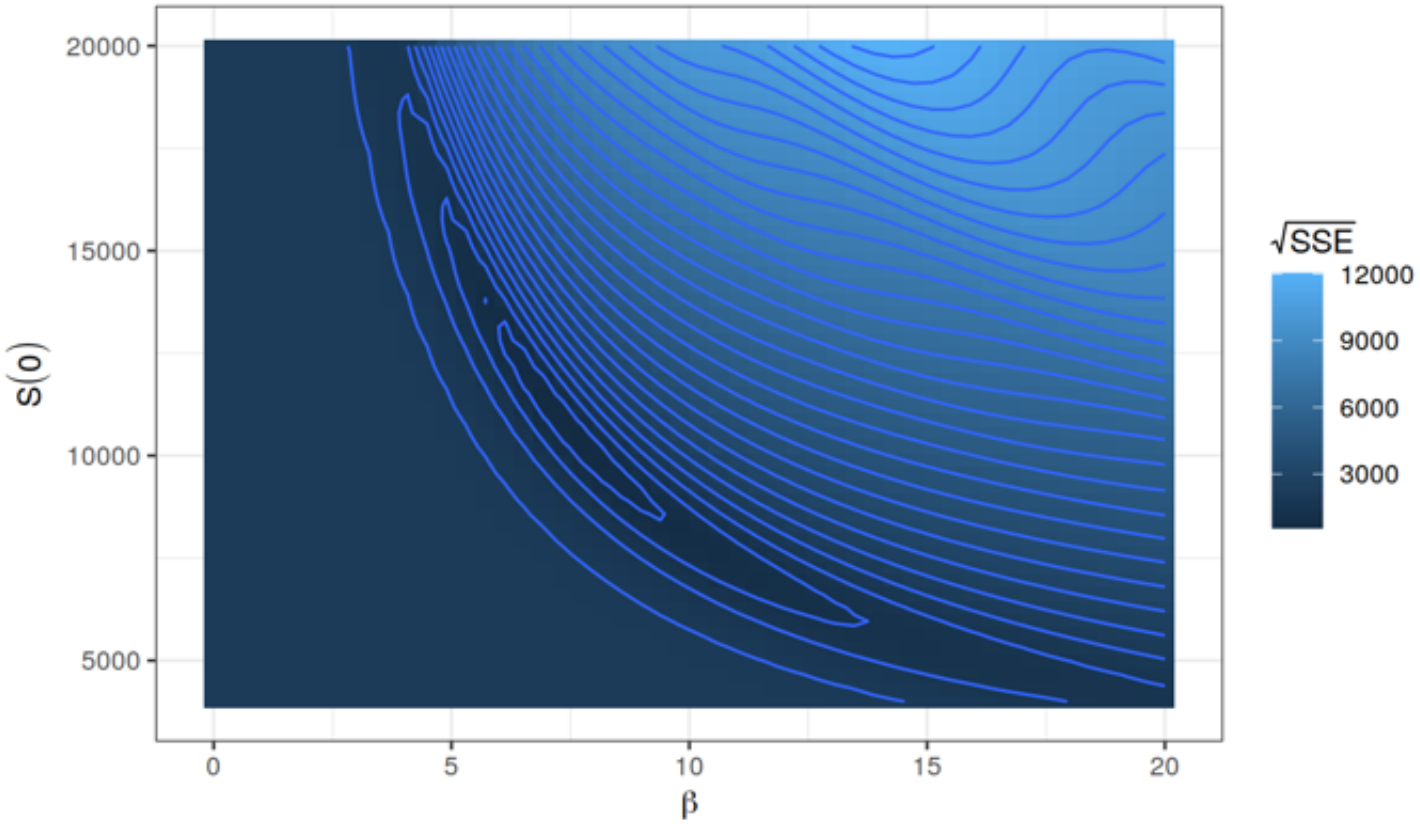
Discuss the following:

- How do we interpret the existence of a local (but not global) optimum?
- Under what circumstances can we be certain a unique global optimum exists?
- What are strategies for increasing the chance that we find the global optimum?
- When do we expect to find multiple local optima?

Clearly, this fit leaves much to be desired, but recall that we've here assumed that we know the correct values for all parameters but β. In particular, we've assumed we know the infectious period, and the initial conditions. [NB: the initial value of R is entirely irrelevant. Why?] Let's see what happens when we try to estimate two parameters at once.

```
grid <- expand.grid(Beta=seq(from=0,to=20,length=50),
                    S_0=seq(from=4000,to=20000,length=50),
                    N=50000,gamma=1,I_0=10)
x <- trajectory(niameyA,params=t(grid),format="data.frame")
library(plyr)
join(x,as.data.frame(niameyA),by="biweek") -> x
ddply(x,~.id,summarize,sse=sum((measles-I)^2)) -> x
cbind(grid,x) -> grid
```

We can visualize this as a surface:



Exercise: interpreting the shape of the likelihood surface

Discuss the shape of this surface: what does it tell us about the uncertainty in the model's parameters?

Exercise: SEIR model estimation

Repeat the estimation using a closed SEIR model. Assume that the infectious period is 5 da and the latent period is 8 da. How and why does your estimate of R_0 differ from that you obtained using the SIR model?

Optimization algorithms

When we have more than two parameters to estimate (as we usually will), we cannot rely on grid searches or simple graphical techniques to find the region of parameter space with the best parameters. We need more systematic ways of searching through the parameter space. Mathematicians have devoted much study to *optimization algorithms*, and there are many of these. Many of them are implemented in **R**.

The first place to go is the function `optim`, which implements several common, well-studied, generally-useful optimization algorithms.

```
?optim
```

To use it, we have to specify the function we want to *minimize* and a starting value for the parameters. Starting from this point, `optim`'s algorithms will search the parameter space for the value that minimizes the value of our *objective function*.

We'll write an objective function to try to estimate β, S_0, and I_0 simultaneously. For the moment, we'll continue to assume that the recovery rate γ is known.

```
f2 <- function (par) {
  params <- c(Beta=par[3],gamma=1,N=50000,S_0=par[1],I_0=par[2])
  sse(params)
}
optim(fn=f2,par=c(10000,10,8)) -> fit2
fit2
```

Exercise: four-parameter optimization

Try to estimate all four parameters at once. Start your algorithm from several places to check that they all converge to the same place. You may find it useful to restart the optimizer to verify its convergence.

Exercise: rescaling the data

In the foregoing, we've estimated parameters by minimizing the sum of squared differences between model-predicted number of cases and the data. What would happen if we tried to minimize the squared error on the log scale, i.e., to minimize $(\log(\text{model}) - \log(\text{data}))^2$? What would happen if we minimized the squared error on the square-root scale, i.e., $(\sqrt{\text{model}} - \sqrt{\text{data}})^2$? What's the "correct" scale to choose?

Many other optimization algorithms exist. `optim` implements several of these (see `?optim`). Other options worth looking into include `constrOptim`, `nlm`, and `nlminb`, and the **subplex** and **nloptr** packages.

Exercise: alternative optimization algorithms

1. Change the optimization algorithm used by `optim` via the `method` argument. Investigate the effect on your parameter estimates.
2. Try using one of the other optimizers mentioned above.

Likelihood

The “brief introduction to the likelihood” (./likelihood.html) lesson introduces this extremely important concept.

Fitting SIR to an epidemic curve using likelihood

Let us revisit the the case of measles in Niamey. We’ll simplify the model slightly to eliminate some unnecessary and wasteful elements. Our frequency-dependent SIR model, again, is

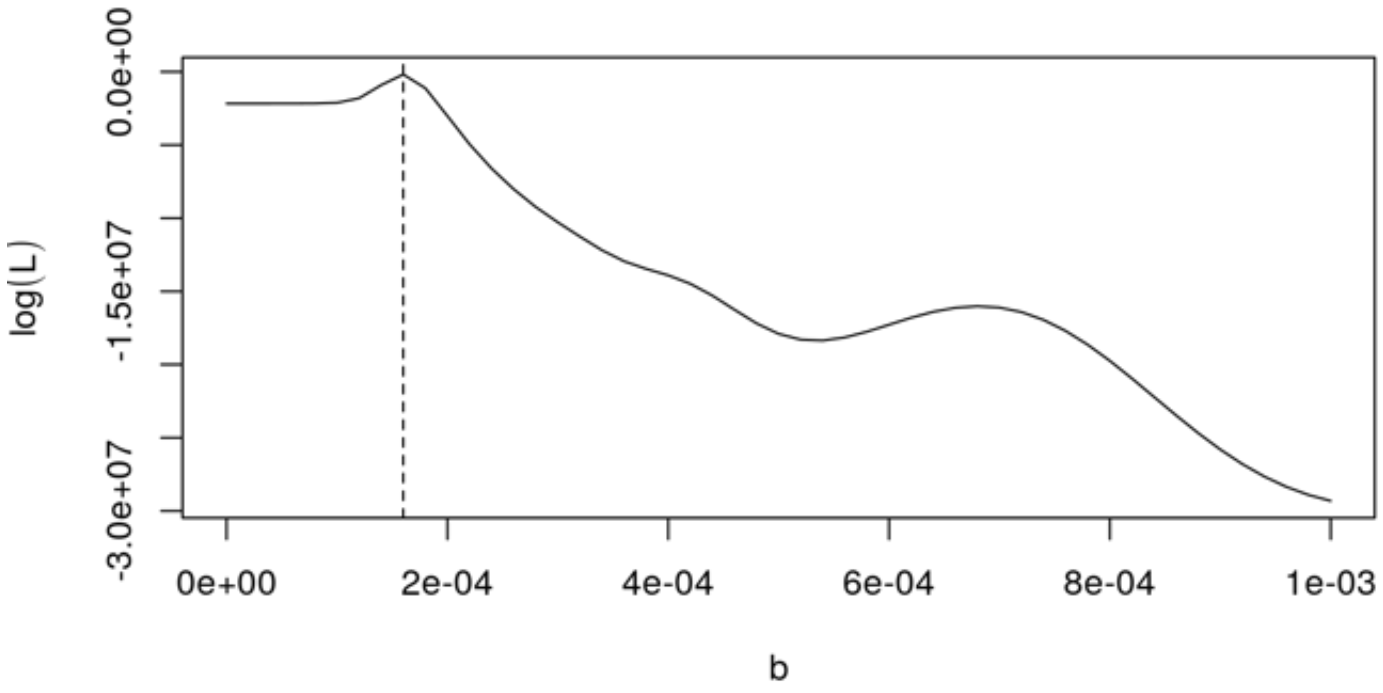
$$\begin{aligned}\frac{dS}{dt} &= -\frac{\beta S I}{N}\\ \frac{dI}{dt} &= \frac{\beta S I}{N} - \gamma I\\ \frac{dR}{dt} &= \gamma I\end{aligned}$$

Notice that 1. the R equation is irrelevant for the dynamics of the epidemic and we can drop it entirely, and 1. β only ever occurs in combination with N, so we can combine these two into a single parameter by defining $b = \beta/N$. We can modify the **R** codes we used before to take account of this.

```
pomp(
  data=subset(niamey,community=="A",select=-community),
  times="biweek",t0=0,
  skeleton=vectorfield(
    Csnippet("
      double incidence;
      incidence = b*S*I;
      DS = -incidence;
      DI = incidence-gamma*I;")),
  rinit=Csnippet("
      S = S_0;
      I = I_0;"),
  paramnames=c("b","gamma","S_0","I_0"),
  statenames=c("S","I")) -> niameyA2
```

Earlier, we used sum of squared error (SSE) as a measure of the discrepancy between model predictions and data. Now let’s use likelihood instead. Let’s suppose that, when we record cases, we make errors that are normally distributed. Here’s how we can compute the log likelihood of the data given the model and its parameters:

```
loglik.normal <- function (params) {
  x <- trajectory(niameyA2,params=params)
  sum(dnorm(x=obs(niameyA2),mean=x["I",,],
            sd=params["sigma"],log=TRUE))
}

f3 <- function (b) {
  params <- c(S_0=10000,I_0=10,gamma=1,b=b,sigma=1)
  loglik.normal(params)
}

b <- seq(from=0,to=0.001,by=0.00002)
ll <- sapply(b,f3)

plot(b,ll,type='l',ylab=expression(log(L)))
b.hat <- b[which.max(ll)]
abline(v=b.hat,lty=2)
```



The great similarity in the likelihood estimate to our first least-squares estimate is no accident. Why is this? Let y_t^* be the observed number of infectives at time t and Y_t be the model's prediction. Then the log likelihood is

$$\log \mathbb{P}\left[y_t^* | Y_t\right] = \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_t^* - Y_t)^2}{2\sigma^2}\right)\right) = -\tfrac{1}{2}\log 2\pi\sigma^2 - \tfrac{1}{2}\frac{(y_t^* - Y_t)^2}{\sigma^2}$$

and

$$\log \mathcal{L} = -\frac{1}{2}\left(\frac{1}{\sigma^2}\sum_t (y_t^* - Y_t)^2 + \log(\sigma^2) + \log(2\pi)\right)$$

So MLE and least-squares are equivalent if the errors are normal with constant variance!

Exercise: log-normal errors

Suppose, alternatively, that the errors are log-normal with constant variance. Under what definition of SSE will least-squares and maximum likelihood give the same parameter estimates?

Modeling the noise

Poisson errors

All this raises the question of what the best model for the errors really is. Of course, the answer will certainly depend on the nature of the data. The philosophy of likelihood encourages us to think about the question mechanistically. When the data, y_t, are the result of a sampling process, for example, we can think of them as binomial samples

$$y_t \sim \text{Binomial}\left(I_t, \frac{n}{N}\right)$$

where n is the sample size, N the population size, and I_t is the true number of infections at time t. Alternatively, we might think of y_t as Poisson samples

$$y_t \sim \text{Poisson}(p\, I_t)$$

where the parameter p reflects a combination of sampling efficiency and the detectability of infections. The latter leads to the following log-likelihood function

```
poisson.loglik <- function (params) {
  x <- trajectory(niameyA2,params=params)
  sum(dpois(x=obs(niameyA2),lambda=params["p"]*x["I",,],log=TRUE))
}
```

Estimating one parameter: point estimate and confidence interval

Let's see what the MLE parameters are for this model. We'll start by estimating just one parameter. Note that the objective function should return the negative log likelihood!

```
f4 <- function (par) {
  params <- c(S_0=20000,I_0=1,gamma=1,b=par,p=0.2)
  -poisson.loglik(params)
}
```

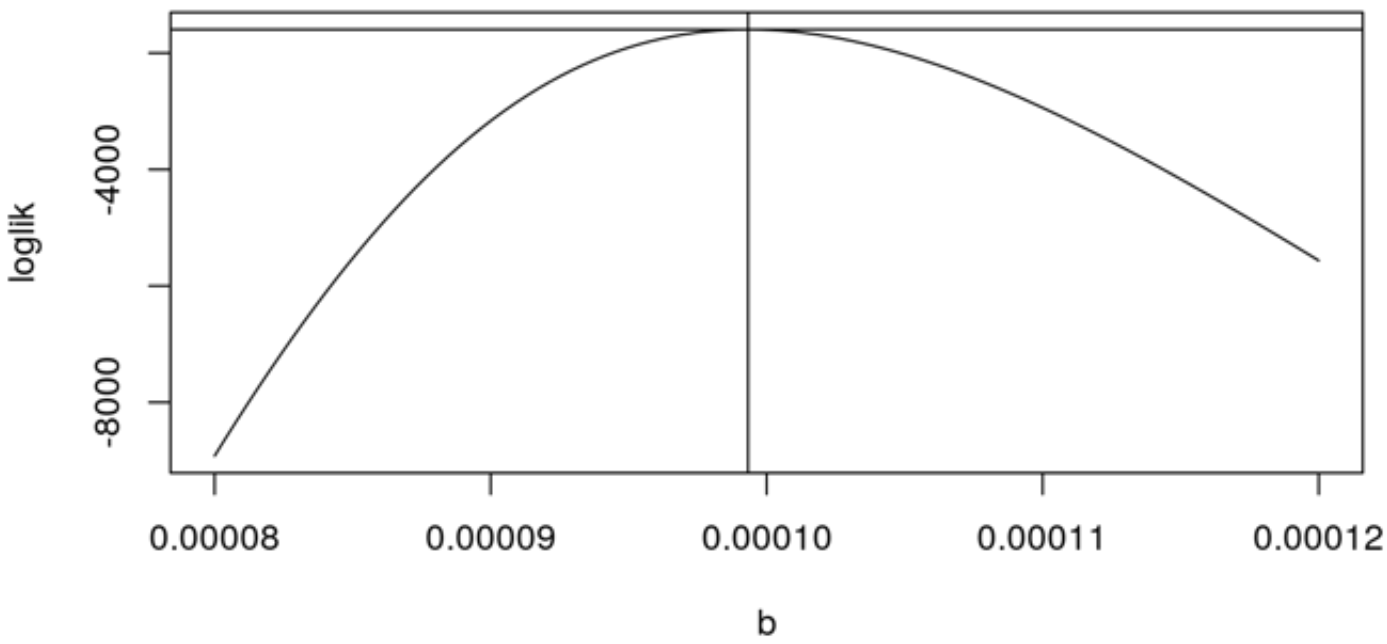
Let's again use `optim` to minimize the objective function. To get started, it needs an initial guess. It's always a good idea to put a bit of thought into this. Since $b = \beta/N = R_0/(\mathrm{IP}\, N)$, where IP is the infectious period, and using guesses $R_0 \approx 15$, $\mathrm{IP} \approx 14$ da, and $N \approx 50000$, we get $b \approx 15/(14{\times}50000)\,\mathrm{da}^{-1} \approx 3 \times 10^{-4}\,\mathrm{biweek}^{-1}$.

```
fit4 <- optim(f4,par=c(0.0003),method="Brent",
              lower=0,upper=1); fit4
```

```
## $par
## [1] 9.93286e-05
##
## $value
## [1] 1595.173
##
## $counts
## function gradient
##       NA       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

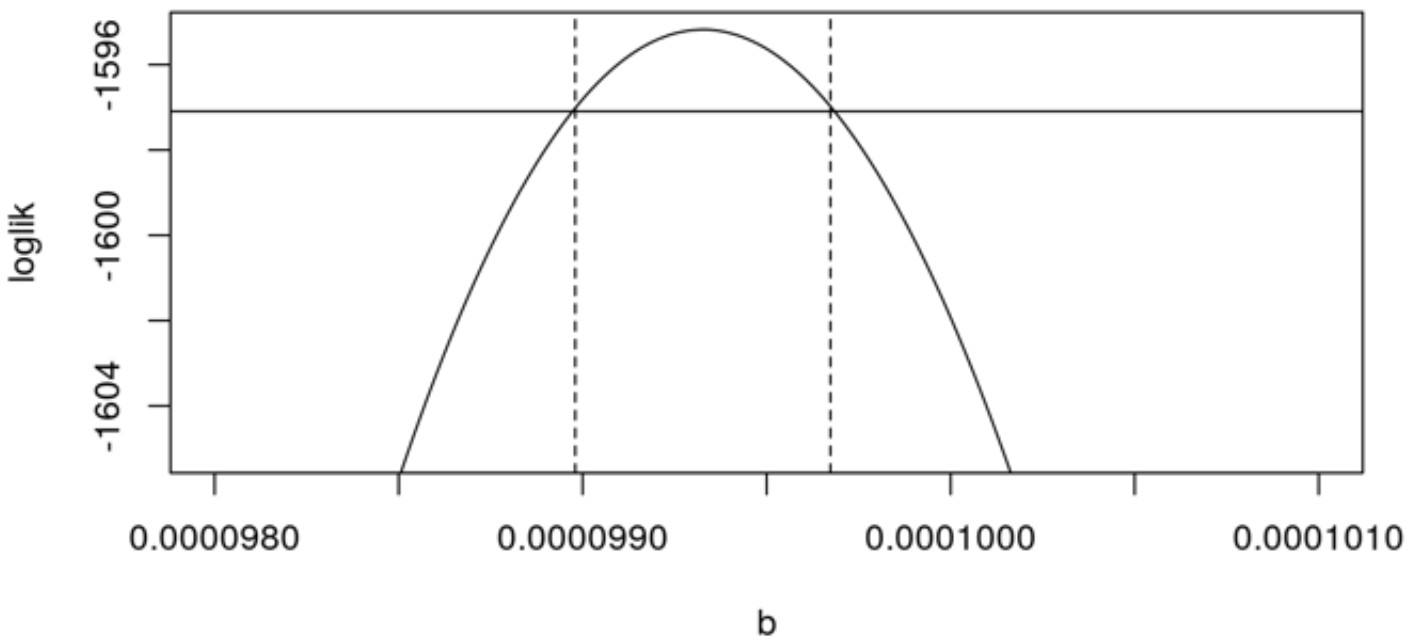
We can get an idea about the uncertainty and in particular obtain confidence intervals by comparing the likelihoods we get at different values of the parameter.

```
prof.b <- expand.grid(b=seq(8e-5,1.2e-4,length=100))
prof.b$loglik <- -sapply(prof.b$b,f4)
maxloglik <- -fit4$value
plot(loglik~b,data=prof.b,type="l")
abline(v=fit4$par)
abline(h=maxloglik)
```



An important and extremely useful property of the log likelihood is that differences in log likelihood have a natural scale that, to a first approximation, does not depend on the model or the data. In particular, log likelihood differences on the order of 1 units are statistically meaningful. The scale above, of 1000s of log likelihood units, is far too gross to be meaningful.

```
prof.b <- expand.grid(b=seq(9.8e-5,1.01e-4,length=200))
prof.b$loglik <- -sapply(prof.b$b,f4)
plot(loglik~b,data=prof.b,type="l",ylim=maxloglik+c(-10,0))
cutoff <- maxloglik-qchisq(p=0.95,df=1)/2
abline(h=c(0,cutoff))
abline(v=range(subset(prof.b,loglik>cutoff)$b),lty=2)
```



In the above, we've computed an approximate 95% confidence interval for the b parameter, subject to the assumptions we've made about the fixed values of the other parameters.

Estimating multiple parameters

Now let's try to estimate both b and the reporting probability p. Since we have constraints on both b and p ($b > 0, 0 \leq p \leq 1$), we'll transform both parameters and estimate them on the transformed scale. For parameters like p that vary on the unit interval, the *logit* function and its inverse are useful:

$$\text{logit}(p) = \log \frac{p}{1-p}, \qquad \text{expit}(x) = \frac{1}{1+e^{-x}}.$$

Now, we must have $b > 0$. This is a *constraint* on the parameter. One way to enforce this constraint is by transforming the parameter so that it cannot ever be negative. We'll log-transform b in writing our objective function, f.

```
logit <- function (p) log(p/(1-p))    # the logit transform
expit <- function (x) 1/(1+exp(-x))   # inverse logit

f5 <- function (par) {
  params <- c(S_0=20000,I_0=1,gamma=1,b=exp(par[1]),p=expit(par[2]))
  -poisson.loglik(params)
}

fit5 <- optim(f5,par=c(log(0.0001),logit(0.2)))
fit5
```

```
## $par
## [1] -9.2266816 -0.4712741
##
## $value
## [1] 579.7906
##
## $counts
## function gradient
##       63       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

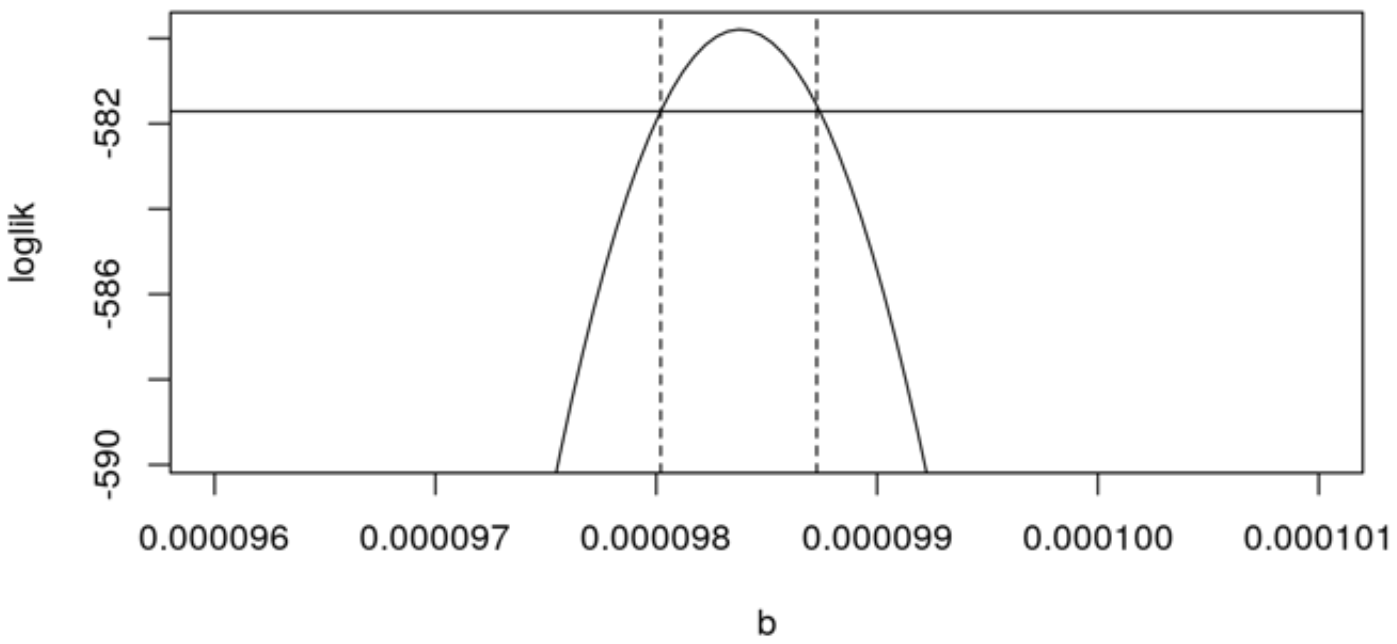
Note that `optim` estimates these parameters on the transformed scale. To get our estimates on the original scale, we must back-transform:

```
mle1 <- c(b=exp(fit5$par[1]),p=expit(fit5$par[2]))
signif(mle1,3)
```

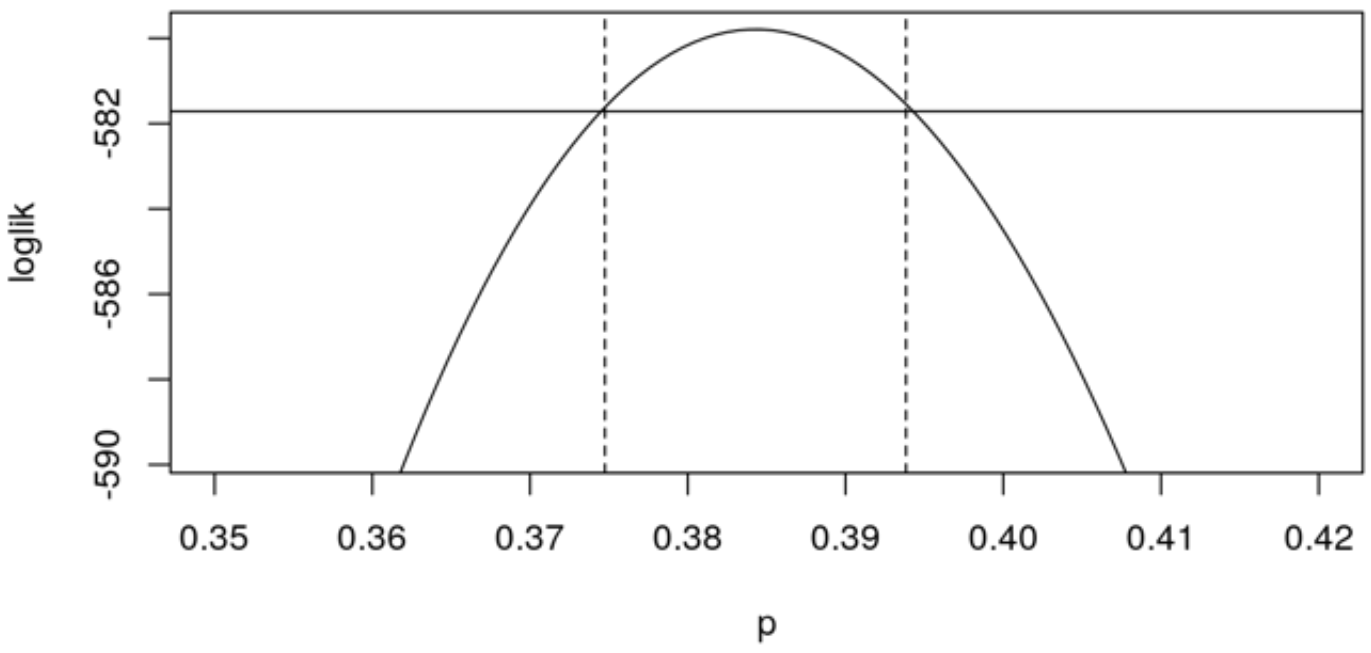
```
##        b        p
## 9.84e-05 3.84e-01
```

To construct confidence intervals—and more generally to visualize the portion of the likelihood surface most relevant to the data—we can construct a *profile likelihood* for each parameter in turn. To profile over a parameter, we fix the value of that parameter at each of several values, then maximize the likelihood over the remaining unknown parameters.

```
prof2.b <- expand.grid(b=seq(9.6e-5,1.01e-4,length=100))
fitfn <- function (dat) {
  fit <- optim(fn=function(p)f5(c(log(dat$b),logit(p))),
               par=mle1[2],method="Brent",lower=0,upper=1)
  c(p=expit(fit$par),loglik=-fit$value)
}

library(plyr)
ddply(prof2.b,~b,fitfn) -> prof2.b
maxloglik <- max(prof2.b$loglik)
plot(loglik~b,data=prof2.b,type="l",ylim=maxloglik+c(-10,0))
cutoff <- maxloglik-qchisq(p=0.95,df=1)/2
abline(h=c(0,cutoff))
abline(v=range(subset(prof2.b,loglik>cutoff)$b),lty=2)
```

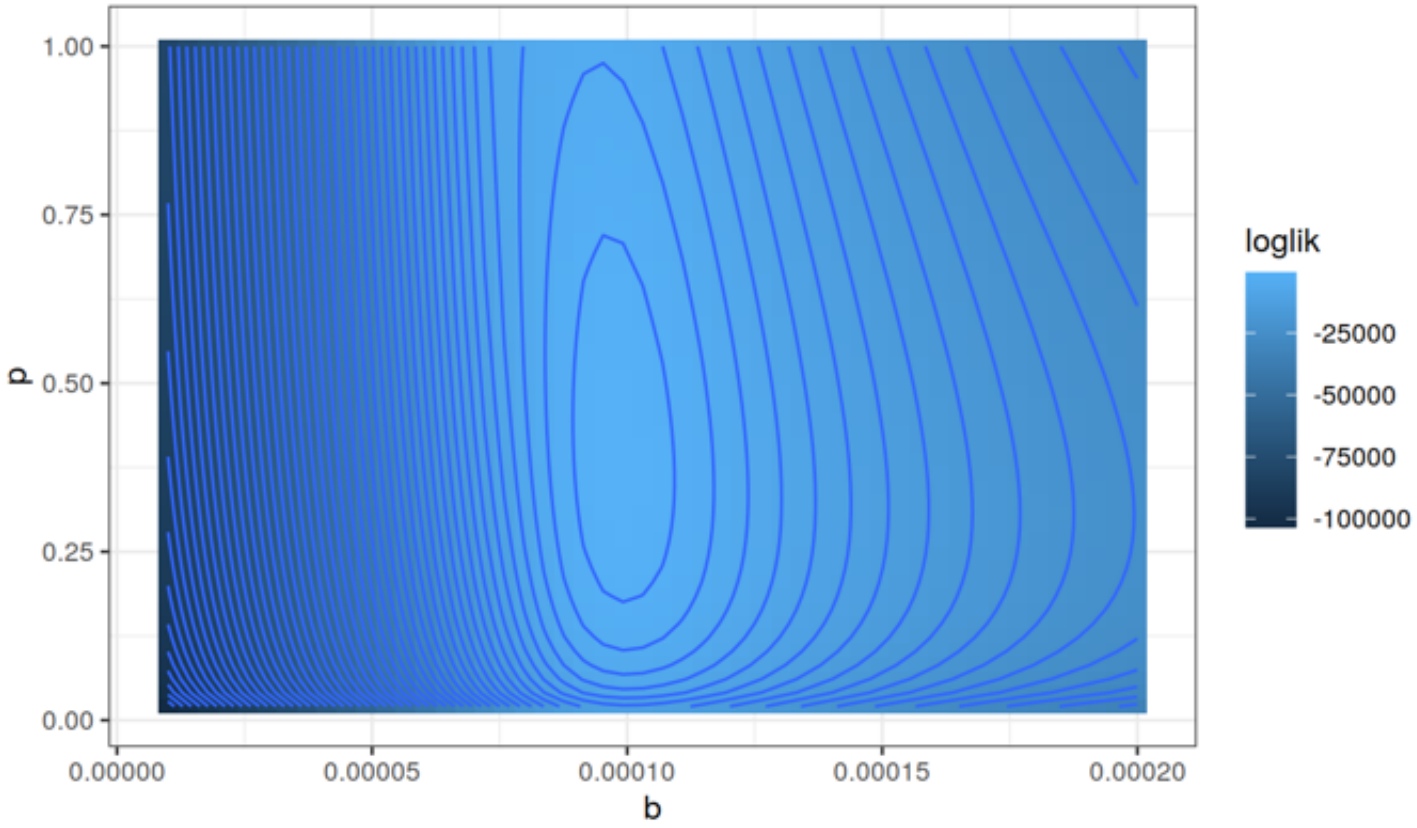



```
prof2.p <- expand.grid(p=seq(0.35,0.42,length=100))
fitfn <- function (dat) {
  fit <- optim(fn=function(b)f5(c(log(b),logit(dat$p))),
               par=mle1[1],method="Brent",lower=9.5e-5,upper=1e-4)
  c(b=expit(fit$par),loglik=-fit$value)
}

library(plyr)
ddply(prof2.p,~p,fitfn) -> prof2.p
maxloglik <- max(prof2.p$loglik)
plot(loglik~p,data=prof2.p,type="l",ylim=maxloglik+c(-10,0))
cutoff <- maxloglik-qchisq(p=0.95,df=1)/2
abline(h=c(0,cutoff))
abline(v=range(subset(prof2.p,loglik>cutoff)$p),lty=2)
```



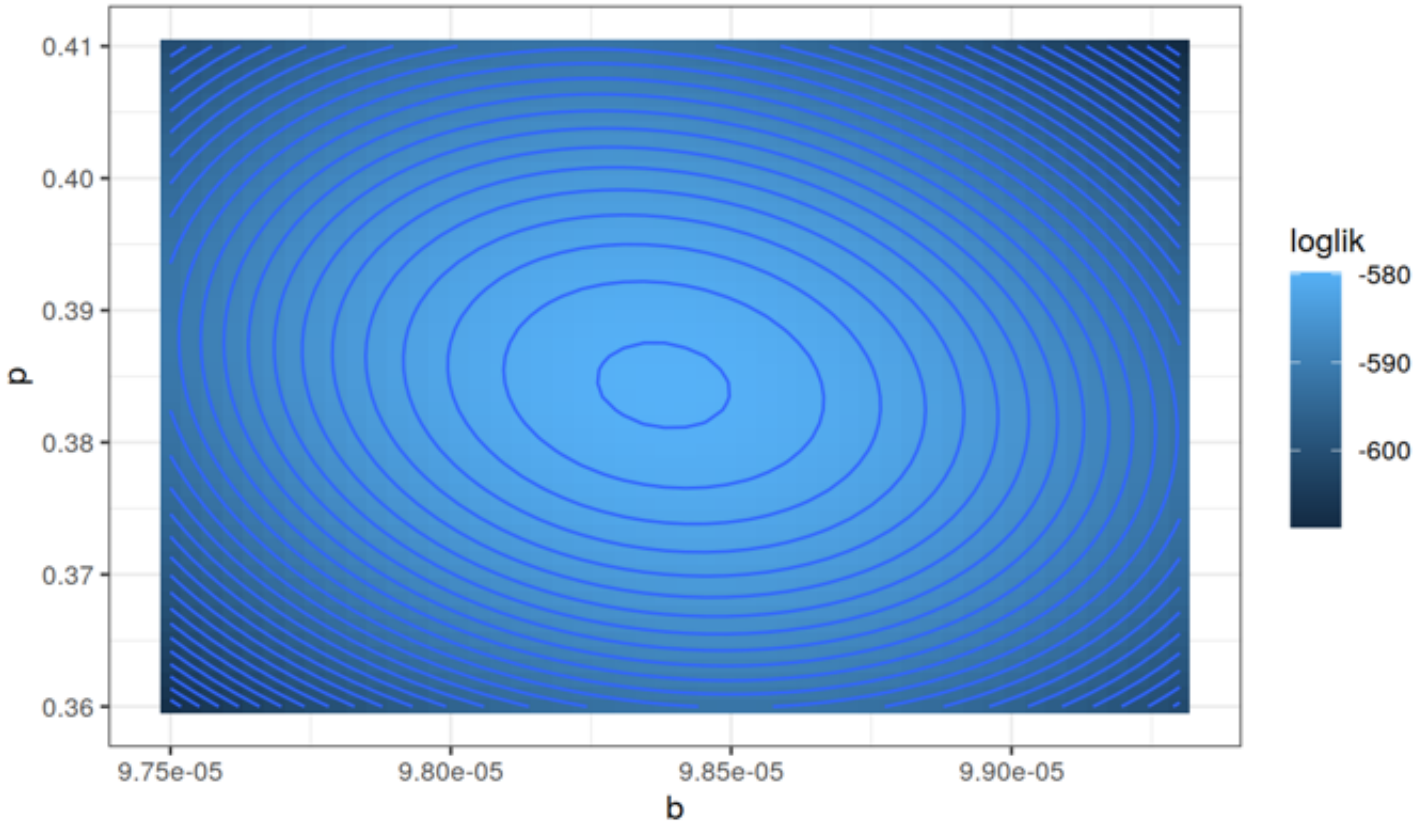
Let's make a contour plot to visualize the likelihood surface.

```
f6 <- function (b, p) {
  params <- c(S_0=20000,I_0=1,gamma=1,b=b,p=p)
  poisson.loglik(params)
}

b <- seq(from=0.00001,to=0.0002,length=50)
p <- seq(0,1,length=50)

library(plyr)
grid <- expand.grid(b=b,p=p)
grid <- ddply(grid,~b+p,mutate,loglik=f6(b,p))
grid <- subset(grid,is.finite(loglik))
ggplot(grid,aes(x=b,y=p,z=loglik,fill=loglik))+
  geom_tile()+geom_contour(binwidth=2000)
```



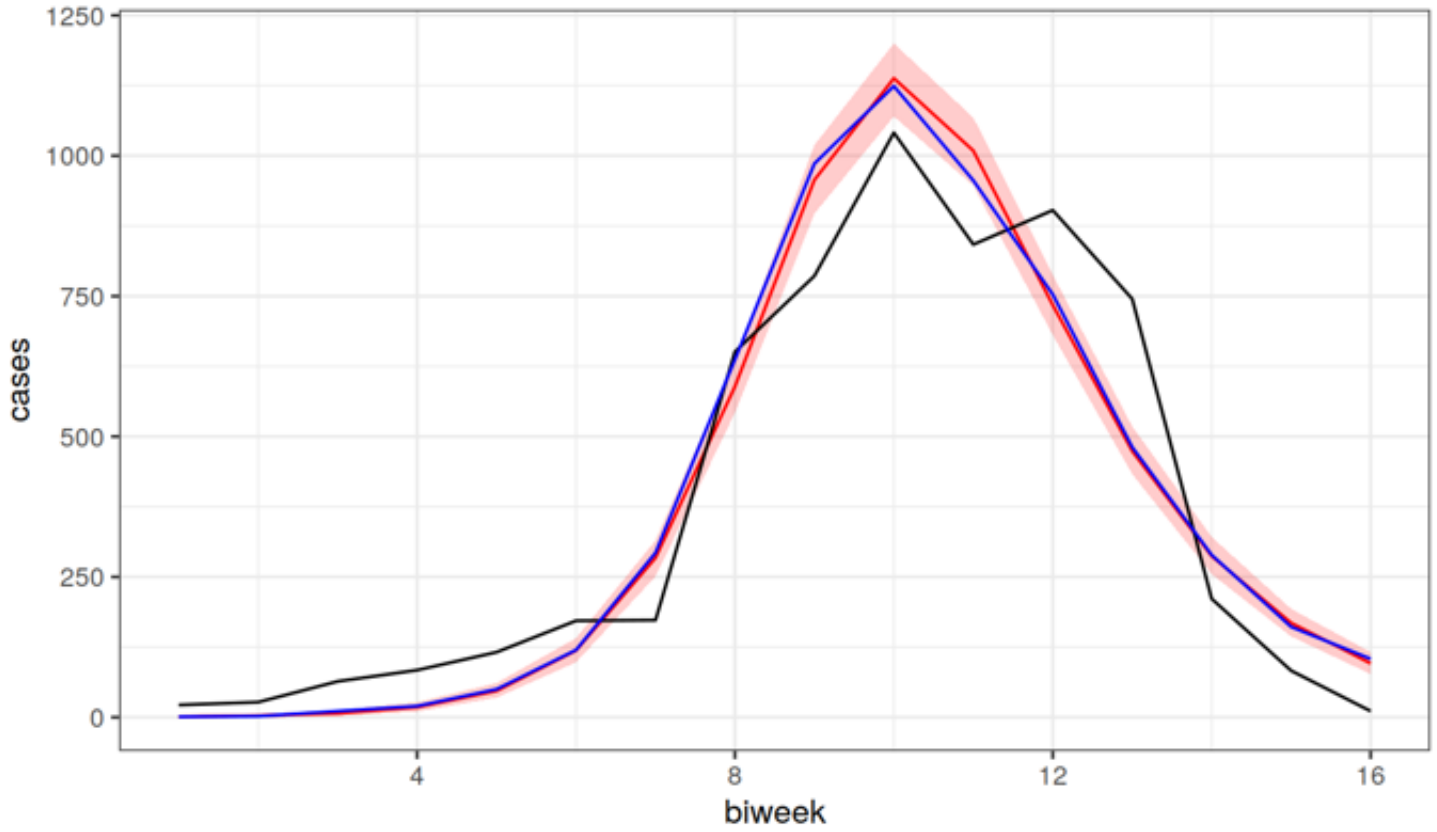
The scale over which the log likelihood is varying is clearly huge relative to what is meaningful. Let's focus in on the region around the MLE.

```
b <- seq(from=0.0000975,to=0.0000993,length=50)
p <- seq(0.36,0.41,length=50)
grid <- expand.grid(b=b,p=p)
grid <- ddply(grid,~b+p,mutate,loglik=f6(b,p))
grid <- subset(grid,is.finite(loglik))
ggplot(grid,aes(x=b,y=p,z=loglik,fill=loglik))+
  geom_tile()+geom_contour(binwidth=1)
```



Let's look at the model's predictions at the MLE. The model is a probability distribution, so we should look at a number of simulations. An important question is: are the data a plausible sample from the predicted probability distribution?

```
coef(niameyA2) <- c(S_0=20000,I_0=1,gamma=1,mle1)
model.pred <- trajectory(niameyA2)["I",,]

library(plyr)
raply(2000,rpois(n=length(model.pred),lambda=coef(niameyA2,"p")*model.pred)) -> simda
t
aaply(simdat,2,quantile,probs=c(0.025,0.5,0.975)) -> quantiles

typ <- sample(nrow(simdat),1)

ggplot(data=cbind(as.data.frame(niameyA2),
                  quantiles,
                  typical=simdat[typ,]),
       mapping=aes(x=biweek))+
  geom_line(aes(y=`50%`),color='red')+
  geom_ribbon(aes(ymin=`2.5%`,ymax=`97.5%`),fill='red',alpha=0.2)+
  geom_line(aes(y=measles),color='black')+
  geom_line(aes(y=typical),color='blue')+
  labs(y="cases",x="biweek")
```



Exercise: three-parameter estimation

Try to estimate p, b, and $S(0)$ simultaneously.

Exercise: binomial errors

Reformulate the problem using the binomial error model. Modify the parameter estimation codes appropriately, estimate the parameters, and comment on the results.

Overdispersion: a negative binomial model

Clearly the Poisson model is not doing a very good job of capturing the pattern in the data. Recall that, under the Poisson assumption, the variance of the error is equal to the mean. It appears that we will need an error model that has the potential for more variability than does the Poisson. The negative binomial distribution is such a distribution. Let's explore the alternative assumption that y_t is negative-binomially distributed with mean $p\,I_t$, as before, but larger variance, $p\,I_t\,(1+\theta\,p\,I_t)$, i.e.,

$$y_t \sim \text{Negbin}\left(\text{mean} = p\,I_t,\ \text{size} = \frac{1}{\theta}\right)$$

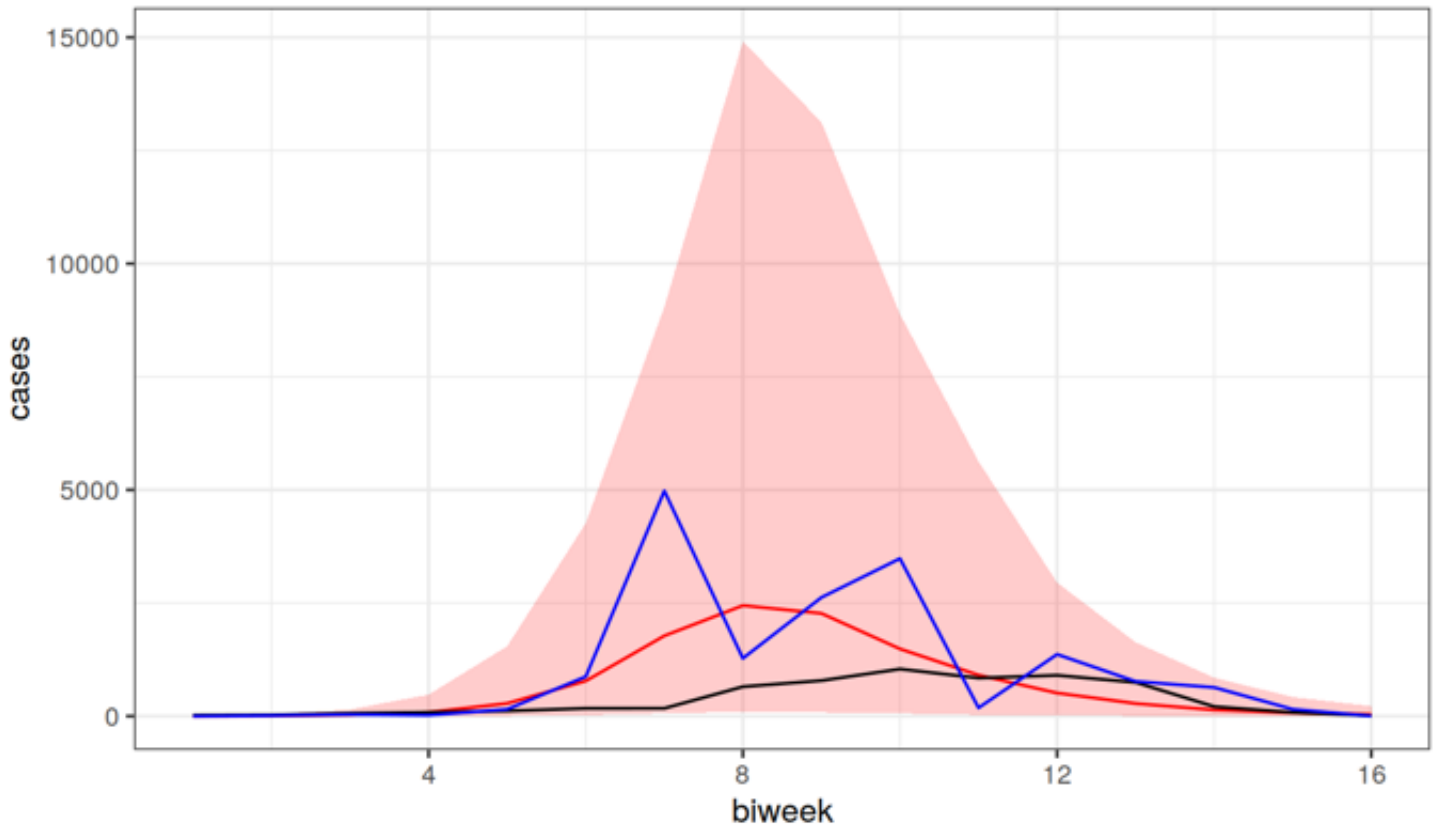
```
negbin.loglik <- function (params) {
  x <- trajectory(niameyA2,params=params)
  prediction <- x["I",,]
  sum(dnbinom(x=obs(niameyA2),
              mu=params["p"]*prediction,size=1/params["theta"],
              log=TRUE))
}

f7 <- function (par) {
  params <- c(S_0=20000,I_0=1,gamma=1,
              b=exp(par[1]),p=expit(par[2]),theta=exp(par[3]))
  -negbin.loglik(params)
}

guess <- c(log(0.0001),logit(0.4),log(1))
fit7 <- optim(fn=f7,par=guess); fit7
```

```
## $par
## [1] -9.10757276 13.58531257  0.06162341
##
## $value
## [1] 108.8148
##
## $counts
## function gradient
##      172       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
mle3 <- c(b=exp(fit7$par[1]),p=expit(fit7$par[2]),theta=exp(fit7$par[3]))
signif(mle3,3)
```

```
##        b        p    theta
## 0.000111 1.000000 1.060000
```

```
coef(niameyA2) <- c(S_0=20000,I_0=1,gamma=1,mle3)
model.pred <- trajectory(niameyA2)["I",,]

library(plyr)
raply(2000,rnbinom(n=length(model.pred),
                   mu=coef(niameyA2,"p")*model.pred,
                   size=1/coef(niameyA2,"theta"))) -> simdat
aaply(simdat,2,quantile,probs=c(0.025,0.5,0.975)) -> quantiles

typ <- sample(nrow(simdat),1)

ggplot(data=cbind(as.data.frame(niameyA2),
                  quantiles,
                  typical=simdat[typ,]),
       mapping=aes(x=biweek))+
  geom_line(aes(y=`50%`),color='red')+
  geom_ribbon(aes(ymin=`2.5%`,ymax=`97.5%`),fill='red',alpha=0.2)+
  geom_line(aes(y=measles),color='black')+
  geom_line(aes(y=typical),color='blue')+
  labs(y="cases",x="biweek")
```



What does this plot tell us? Essentially, the deterministic SIR model, as we've written it, cannot capture the shape of the epidemic. In order to fit the data, the optimization algorithm has expanded the error variance, to the point of absurdity. The typical model realization (in blue) does not much resemble the data.

Exercise: modifying the mean-variance relationship

Reformulate the problem using a normal error model in which the variance is proportional to the mean:

$$y_t \sim \text{Normal}\left(p\, I_t, \sigma \sqrt{I_t}\right).$$

Modify the parameter estimation codes appropriately, estimate the parameters (including both p and σ), and comment on the results.

Exercise: prevalence vs incidence

We've been treating the Niamey data as if they were direct—though inaccurate—measurements of the prevalence. Actually, these are incidence data: they are measures of unique infections. It would be more appropriate to model these data by adding another equation

$$\frac{dH}{dt} = \frac{\beta\, S\, I}{N}$$

to accumulate new infections and assuming the data are distributed according to, for example,

$$y_t \sim \text{Poisson}\left(p\,(H_t - H_{t-1})\right).$$

Modify the codes above to reflect these more appropriate assumptions, estimate the parameters, and comment on the results.

Back to course homepage (../)

R codes for this document (http://raw.githubusercontent.com/kingaa/clim-dis/master/parest/parest.R)

References

Grais, R. F., M. J. Ferrari, C. Dubray, O. N. Bjørnstad, B. T. Grenfell, A. Djibo, F. Fermon, and P. J. Guerin. 2006. Estimating transmission intensity for a measles epidemic in Niamey, Niger: Lessons for intervention. Transactions of the Royal Society of Tropical Medicine and Hygiene. 100:867–873.

Kermack, W. O., and A. G. McKendrick. 1927. A contribution to the mathematical theory of epidemics. Proceedings of the Royal Society of London, Series A 115:700–721.