Ecological Models and Data: Estimation of R_0

Matthew Ferrari

1 The SIR Model

The classic model for microparasite dynamics is the flow of hosts between **S**usceptible, **I**nfectious, and **R**ecovered classes. This leads to the following standard formulation of the *SIR* model:

$$\begin{aligned} \frac{dS}{dt} &= -\beta IS \\ \frac{dI}{dt} &= \beta IS - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{aligned} \tag{1}$$

Here, $1/\gamma$ is the average infectious period (we'll gloss over the separation of the *generation time* into incubation and infectious periods). The parameter β is the transmission rate. The average infectious period (or technically, the *serial interval*) is generally known or estimable from clinical data, and the goal is to estimate the rate of transmission β.

2 Definition of R_0

The transmission rate parameter β can be difficult to interpret as the absolute value, and the impact on population dynamics, often depends on the size of host population size and rates of host mixing. As such, the common summary value used to describe epidemic dynamics is basic reproductive ratio, R_0, which is the expected number of secondary cases casued by a single infectious individual in a wholly susceptible population. For the above model, $R_0 = \frac{S_0\beta}{\gamma}$.

3 Chain binomial model

The classic formulation of the SIR model is a continuous time, deterministic model. In order use the likelihood framework we nee a stochastic version of the model. Commonly, monitoring data on infectious diseases reports only the number of cases and these reports are given as aggregates over some monitoring interval (i.e. weekly or monthly). For these data the *chain binomial* epidemic model is a useful stochastic epidemic model. We can write the chain binomial model as

$$
\begin{aligned}
P(I_{t+1} = n|\beta, I_t, S_t) &= \binom{S_t}{n}(1-\exp(-\beta I_t))^n \exp(-\beta I_t)^{S_t - n} \\
St+1 &= St - It + 1
\end{aligned}
\tag{2}
$$

Here the time step is taken as the average infectious period, $1/\gamma$. Then the number of cases, I_{t+1}, in time $t+1$, is a binomial draw from the number of susceptibles in the previous time step, S_t with probability $(1-\exp(-\beta I))$ determined by the trasmission rate and number of infected hosts in the population (NOTE: this probabity of infection comes from the formulation of the epidemic as a continuous time birth-and-death process, and $(\exp(-\beta I))$ gives the Poisson probability of 0 new infections occuring in 1 time step). All infectious individuals move into the recovered class with probability at the end of the infectious period with probability 1. As such, the number of susceptibles at time $t+1$ is simply the number at time t minus those that became infected.

4 Likelihood

To make a useful likelihood of this model we must consider that we usually do not see the timeseries of susceptibles. Noting that the expression for S_{t+1} is recurssive we can see that $S_1 = S_0 - I_1$

$$
\begin{aligned}
S_1 &= S_0 - I_1 \\
S_2 &= S_0 - (I_1 + I_2) \\
&\vdots \\
S_{t+1} &= S_0 - \sum_{i=1}^{t} I_i
\end{aligned}
\tag{3}
$$

Let's define $\hat{S}_t$ as $S_0 - \sum_{i=1}^{t-1} I_i$. Then we can treat the unknown initial number of susceptibles, S_0 as a parameter and write the likelihood for the Is in terms of β and S_0. Thus, conditional the number previous time steps, we can define I_{t+1} as a binomial random variable:

$$
nI_{t+1} \sim \text{Binomial}(\hat{S}_t, 1-\exp(-\beta I_t)). \tag{4}
$$

If we make the assumption that each epidemic generation depends only on the state of the sytem in the previous timestep ("conditional independence"), then we can write the joint likelihood for a time series of observed cases, I, as

$$
L(I|\beta, S_0) = \prod_{t=1}^{T} \binom{\hat{S}_t}{I_t}(1-\exp(-\beta I_t))^{I_t} \exp(-\beta I_t)^{\hat{S}_t - I_t} \tag{5}
$$

(the $\prod$ symbol is like the summation symbol $\sum$, but for multiplication instead).

5 Example

Here are some data on an outbreak of measles from three different reporting centers in Niamey, Niger from 2003. The data have been aggregated into 14-day intervals (the generation time (= incubation + infectious period) for measles).

```
> niamey_cases1 <- c(4, 1, 12, 17, 37, 55, 51, 331, 323, 370, 145,
+     224, 162, 26, 6)
> niamey_cases2 <- c(1, 2, 3, 6, 9, 24, 40, 55, 88, 158, 173, 155,
+     141, 36, 20, 2)
> niamey_cases3 <- c(2, 7, 4, 14, 21, 49, 73, 151, 279, 245, 185,
+     129, 49, 19, 2)

> ## basic plot
> plot(niamey_cases1,type="l",xlab="biweek",ylab="measles cases")
> ## add lines, one at a time, different line types
> lines(niamey_cases2, lty=2)
> lines(niamey_cases3, lty=3)
> legend("topleft",
+        legend=c("Center 1","Center 2","Center 3"),lty=1:3)
```

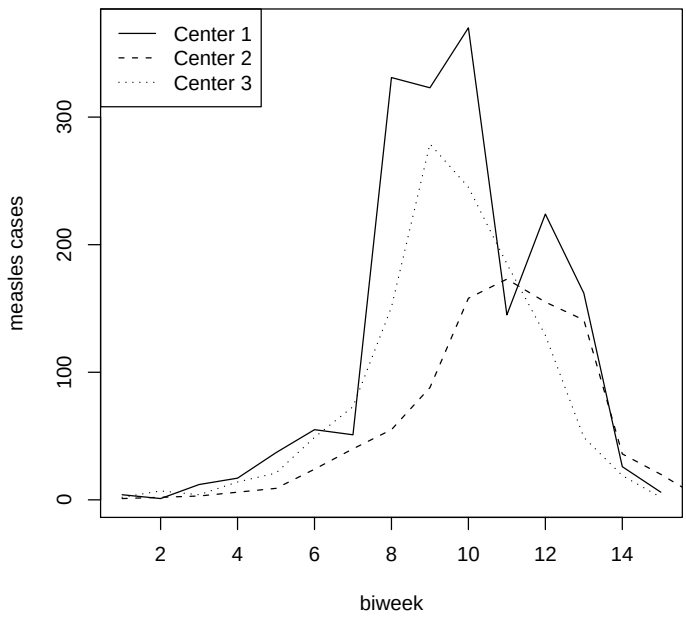


Figure 1: Time series of cases at three reporting centers in Niamey, Niger.

5.1 Basic Estimation

Following the standard recipe, the first step is to write a function for the likelihood. Note that I'm using observations 2 onwards as the data as the probability of observing any cases in the first time step, given that there were none previously is 0 in this model. Thus I am making an implicit assumption that the first cases are coming from elsewhere.

```
> likelihood <- function(S0, beta, I) {
+     n <- length(I)
+     S <- floor(S0 - cumsum(I[-n]))
+     p <- 1 - exp(-beta * (I[-n]))
+     L <- dbinom(I[-1], S, p, log = TRUE)
+     Lik <- sum(-L, na.rm = TRUE)
+ }
```

We've used the `floor()` function for the vector of Ss (line 4 in the function), because `dbinom()` expects integers.

We can visualize the likelihood surface by plotting the likelihood values over grid of potential values for S_0 and β. (For models with more parameters, corresponding to a higher-dimensional parameter space, this is unfortunately not practical.) For S_0 we'll choose the range from `sum(I)` to `3*sum(I)`, and for β we'll choose values that would correspond to R_0 equal to 1.5–5.

```
> tot1 <- sum(niamey_cases1)
> S0<-floor(seq(tot1+1,3*tot1,length=40))
> beta<-seq(1.5/S0[1],5/S0[1],length=40)
> lik<-matrix(NA,nrow=40,ncol=40)
> for(i in 1:40){
+     for(j in 1:40){
+         lik[i,j]<-likelihood(S0=S0[i],beta=beta[j],
+                              I=niamey_cases1)
+     }
+ }
```

Or, alternatively:

```
> library(emdbook)
> lik <- apply2d(likelihood, S0, beta, I = niamey_cases1)
```

(If `likelihood()` were *vectorized* we could use `outer()` instead — this will come up later.) Or:

```
> lik <- curve3d(likelihood(x, y, I = niamey_cases1), from = c(tot1 +
+     1, 1.5/S0[1]), to = c(3 * tot1, 5/S0[1]), sys3d = "contour",
+     n = c(40, 40), levels = 200 * (2:11))$z
```

Now calculate the grid point that has the smallest negative log-likelihood: save this value as a starting point for optimization.

```
> symbols(rep(S0, 40), rep(beta, each = 40), circles = 50 * as.vector(lik)/max(lik),
+     xlab = ~S[0], ylab = ~beta, inches = FALSE)
> points(S0[mle.ind[1]], beta[mle.ind[2]], col = 2, pch = 16)
> contour(S0, beta, lik, levels = (200 * (2:11)), add = TRUE)
```

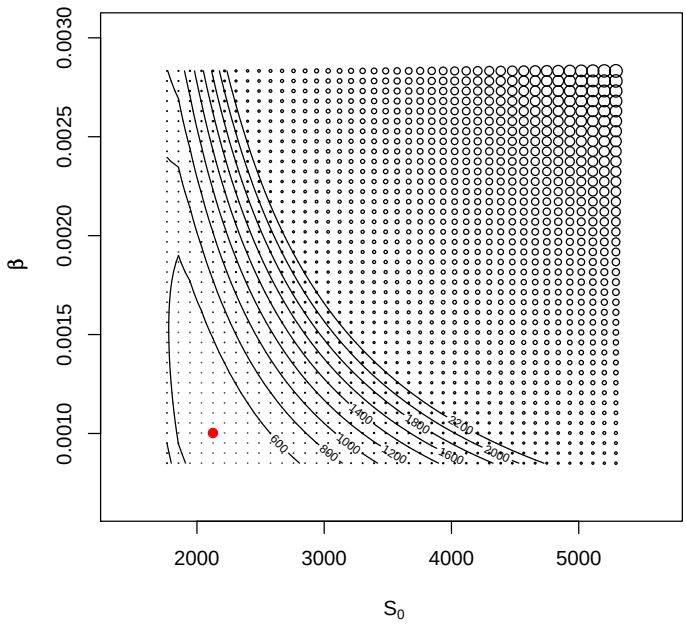


Figure 2: Negative log Likelihood over a grid of parameter values. Black circles give the values of the negative log likelihood, and the lines give contours. The red point indicates the best parameter combination.

```
> mle.ind <- which(lik == min(lik), arr.ind = TRUE)
> init <- list(S0 = S0[mle.ind[1]], beta = beta[mle.ind[2]])
```

We can then `mle2()` (or `optim()`), to get MLEs.

```
> library(bbmle)
> mle.fit2 <- mle2(start = init, likelihood, method = "L-BFGS-B",
+     lower = c(tot1, 1e-06), upper = c(Inf, 0.1), data = list(I = niamey_cases1),
+     control = list(ndeps = c(1, 1e-05)))
```

I set lower and upper bounds for `S0` and `beta` to stop the optimization routines from getting into trouble later on (thus I had to use `method="L-BFGS-B"`); similarly, `ndeps` controls the size of the step for numerical approximation of the derivatives (it has to be bigger than the default of 0.001 for `S0` and smaller for `beta`). Unfortunately, this kind of tweaking is often necessary. (`parscale` is another useful optimization control; it sets the overall parameter scale, and sometimes works better than `ndeps`.)

While this gave me an answer, I actually ran into a problem later on, so I'm going to cheat a little bit here and use a different set of starting conditions:

```
> newstart <- list(S0 = 2153.976, beta = 0.0009697629)
> mle.fit2 <- mle2(start = newstart, likelihood, method = "L-BFGS-B",
+     lower = c(tot1, 1e-06), upper = c(Inf, 0.1), data = list(I = niamey_cases1),
+     control = list(ndeps = c(1, 1e-05)))
```

This gives us estimates:

```
> coef(mle.fit2)
```

```
          S0         beta
2.153976e+03 9.697650e-04
```

Now we can visualize the confidence regions for S_0 and β. First we can add the bivariate 95% confidence interval from the likelihood ratio test:

```
> contour(S0, beta, lik, levels = -logLik(mle.fit2) + qchisq(0.95,
+     2)/2, col = "red", add = TRUE)
```

We can also add the confidence region based on the Fisher information matrix. Recall that the Fisher information matrix is the variance-covariance matrix for the estimates, which we get by inverting the matrix of second derivatives (Hessian) estimated at the maximum likelihood estimate. The `vcov()` command does this automatically.

```
> (covmat = vcov(mle.fit2))
```

```
                S0          beta
S0    2.256644e+03 -2.129620e-03
beta -2.129620e-03  2.546555e-09
```

The diagonal elements of this matrix are the (approximate) variances for S_0 and β. Thus we can use either `sqrt(diag(covmat))` or `coef(summary(mle.fit2))` to get standard errors for the parameters[1]

The off-diagonal elements are the covariance between S_0 and β. We can translate this into a correlation using `cov2cor()`.

```
> (cor.mat <- cov2cor(covmat))
```

```
             S0       beta
S0    1.0000000 -0.8883703
beta -0.8883703  1.0000000
```

The correlation between the estimates of β and S_0 is large (-0.888). In this setting that means that the likelihood that there were many initial susceptible hosts and relatively low transmission rate is similar to the likelihood that there were few initial susceptibles and a relatively high transmission rate.

We can plot the confidence region based on the Fisher information matrix using the `ellipse()` function from the `ellipse` package[2]

We could zoom in on this

```
> c2 = curve3d(likelihood(x, y, I = niamey_cases1), from = c(tot1 +
+     1, 1.5/S0[1]), to = c(2500, 0.00125), sys3d = "contour",
+     n = c(40, 40))
> contour(c2$x, c2$y, c2$z, levels = -logLik(mle.fit2) + qchisq(0.95,
+     2)/2, col = "red", add = TRUE)
> lines(ellipse(vcov(mle.fit2), centre = coef(mle.fit2)), col = 4)
```

6 Profile confidence limits

Now we want to calculate the *likelihood profiles* and confidence limits for β and S_0.

```
> p1 = profile(mle.fit2)
```

```
> confint(mle.fit2)
```

```
            2.5 %       97.5 %
S0   2.074504e+03 2.269373e+03
beta 8.629068e-04 1.015379e-03
```

(it would be more efficient here to say `confint(p2)`, which would use the already-computed profile).

Compare with Fisher-information limits:

```
> confint(mle.fit2, method = "quad")
```

```
            2.5 %       97.5 %
S0   2.060870e+03 2.247082e+03
beta 8.708585e-04 1.068671e-03
```

[1] The Z- and p-values that appear in the `coef(summary(...))` output (which use the null hypotheses $\beta = 0$ and $S_0 = 0$) make very little sense in this particular case ...

[2] Because the `ellipse` package was written by Canadians and Irish, the middle of the ellipse is denoted by `centre` (!)

```
> symbols(rep(S0, 40), rep(beta, each = 40), circles = 50 * as.vector(lik)/max(lik),
+     xlab = ~S[0], ylab = ~beta, inches = FALSE)
> points(S0[mle.ind[1]], beta[mle.ind[2]], col = 2, pch = 16)
> contour(S0, beta, lik, levels = (200 * (2:11)), add = TRUE)
> contour(S0, beta, lik, levels = -logLik(mle.fit2) + qchisq(0.95,
+     2)/2, col = "red", add = TRUE)
> library(ellipse)
> lines(ellipse(vcov(mle.fit2), centre = coef(mle.fit2)), col = 4)
```

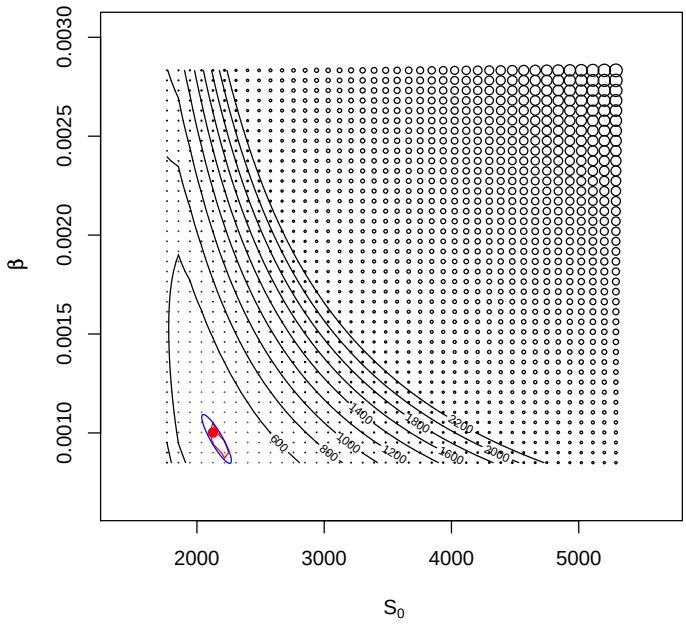


Figure 3: Negative log Likelihood surface with confidence regions. Red is the likelihood interval based on the χ^2 approximation. Blue is the confidence ellipse based on the Fisher information matrix.

```
> plot(p1)
```

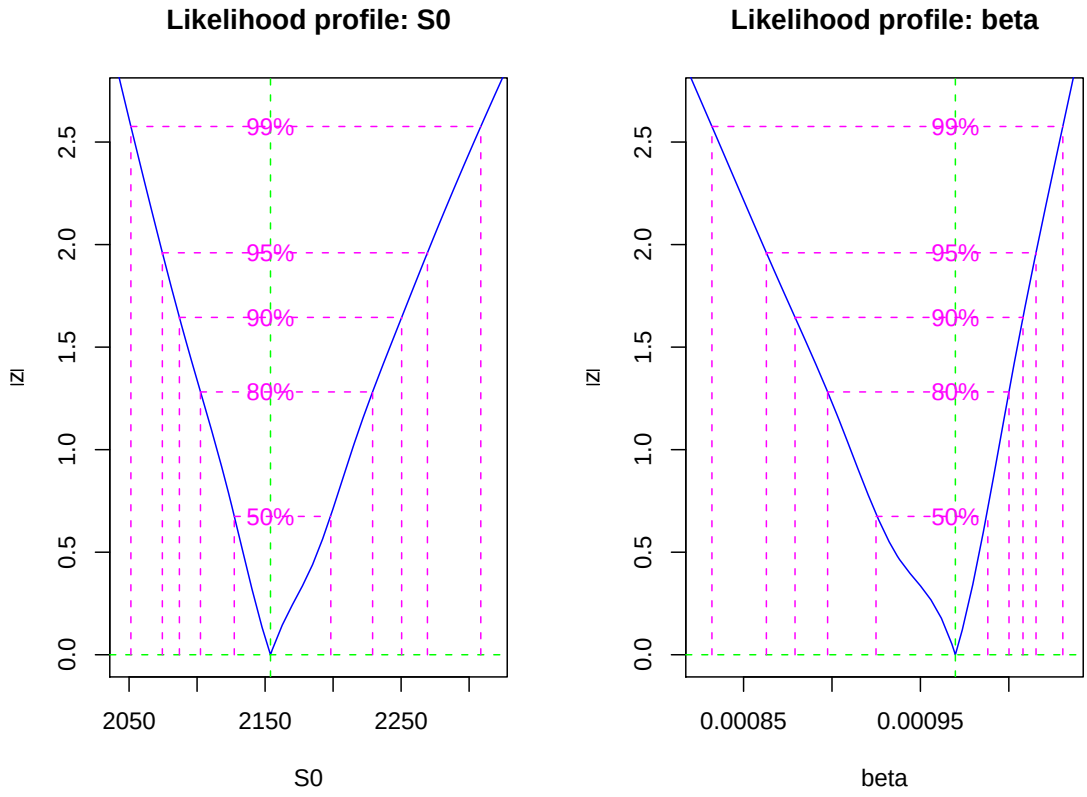


Figure 4: Likelihood profiles. These are expressed in terms of the absolute value of the square root of the deviance difference from the best fit (this makes "well-behaved" profiles V-shaped and symmetric).

7 Estimating R_0

Now we can begin to approach the problem of estimating R_0. From the definition above, $R_0 = \frac{S_0\beta}{\gamma}$, where γ is the recovery rate. Because of the way we set up the model and the likelihood, we assume that all infected hosts recover at the end of every two week period, so we can simplify the expression to $R_0 = S_0\beta$, which is in terms only of our two estimated parameters. Thus the point estimate for R_0 is simply $\hat{R_0} = \hat{S_0}\hat{\beta}$. Getting an appropriate confidence interval on that estimate, however, requires a bit more work.

```
> (R0.mle <- prod(coef(mle.fit2)))
```

```
[1] 2.088850
```

First we can look at the contours for R_0 superimposed on the likelihood surface for the parameters. Since $R_0 = S_0\beta$ it is simple to get the value for all the parameter combinations on the grid.

```
> R0.fun <- function(S0, beta) {
+     S0 * beta
+ }
> mmat <- outer(S0, beta, R0.fun)
```

The `outer()` command here is a quick way to generate matrices over the parameter combinations in `S0` and `beta`[3]

```
> likelihood_R0 <- function(S0, R0, I) {
+     n <- length(I)
+     beta <- R0/S0
+     S <- floor(S0 - cumsum(I[-n]))
+     p <- 1 - exp(-beta * (I[-n]))
+     L <- dbinom(I[-1], S, p, log = TRUE)
+     Lik <- sum(-L, na.rm = TRUE)
+ }
> R0start <- list(S0 = coef(mle.fit2)["S0"], R0 = coef(mle.fit2)["S0"]/coef(mle.fit2)["beta"
> mle.fit3 = mle2(start = R0start, likelihood_R0, method = "L-BFGS-B",
+     lower = c(tot1, 0.5), upper = c(10 * tot1, 10), data = list(I = niamey_cases1),
+     control = list(ndeps = c(1, 0.001)))
> (c1 = confint(mle.fit3))
```

```
         2.5 %      97.5 %
S0 2074.979964 2268.878996
R0    1.993813    2.189592
```

There are other ways to solve this problem, which are useful if you're either in more of a hurry or

- the delta method

[3]Since the default function that `outer()` uses is `"*"` (multiplication), we could just say `outer(S0,beta)`.

```
> symbols(rep(S0, 40), rep(beta, each = 40), circles = 50 * as.vector(lik)/max(lik),
+     xlab = ~S[0], ylab = ~beta, inches = FALSE)
> contour(S0, beta, mmat, add = TRUE, col = "blue", labcex = 2)
> contour(S0, beta, mmat, add = TRUE, col = "blue", lty = 2, drawlabels = FALSE,
+     levels = seq(1.5, 12.5))
> lines(ellipse(vcov(mle.fit2), centre = coef(mle.fit2)), col = 5)
```

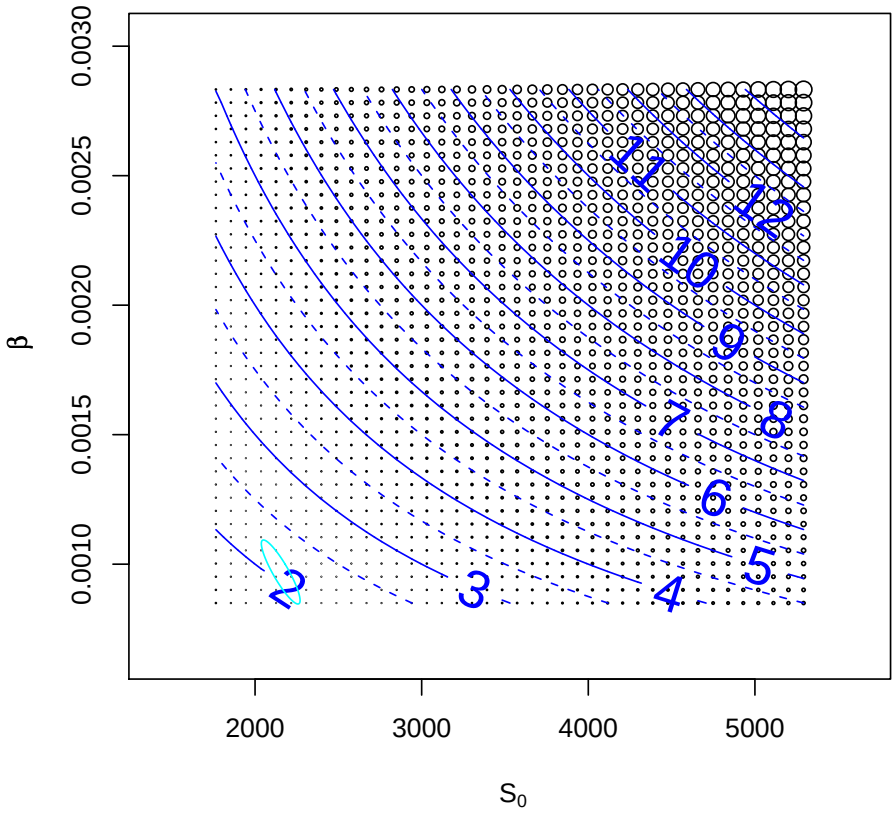
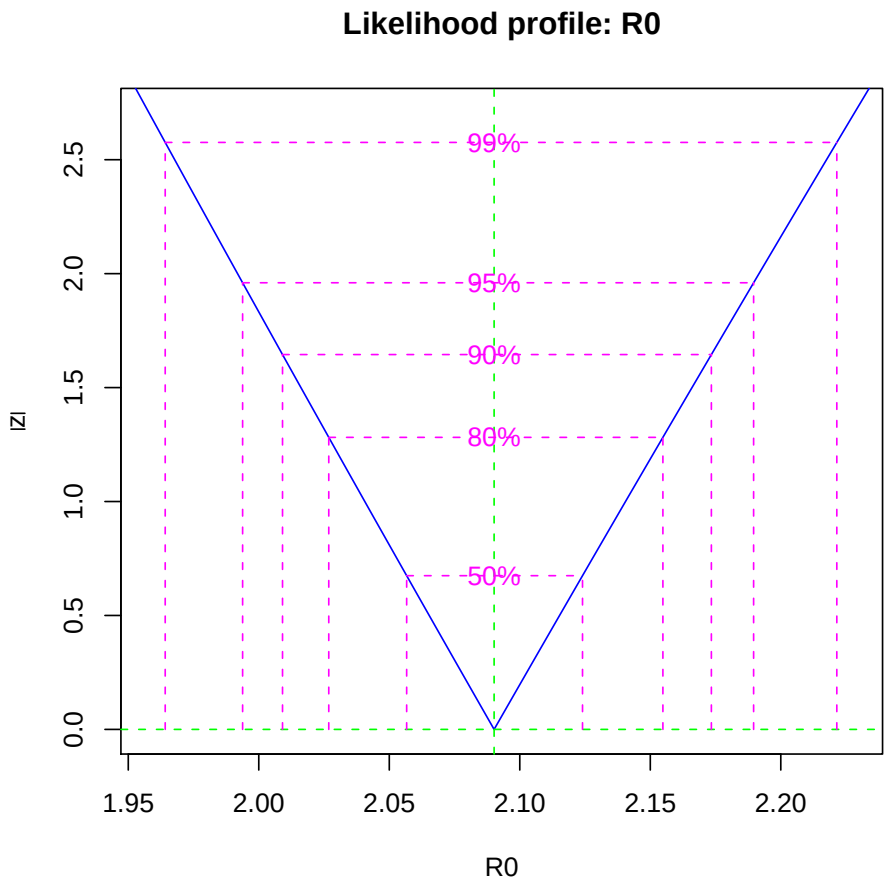


Figure 5: R_0 contours superimposed on the likelihood surface for S_0 and β. Cyan line is the Fisher information confidence ellipse — showing that R_0 is approximately in the range 2–2.5.

```
> p3 = plot(profile(mle.fit3, which = "R0"))
```



- resampling from the sampling distribution of the parameters, estimated from the Fisher information matrix
- creating a likelihood function with a penalty term for the distance from a "target" R_0

8 R_0 for other sites

Let's do the same for the other two data sets and compare the estimates of R_0.

I'm cheating a little bit (again), using `curve3d` to explore the parameter space and find decent starting conditions. This will be harder with a more complex model ...

```
> curve3d(likelihood_R0(x, y, I = niamey_cases2), from = c(900,
+     1), to = c(2000, 3), sys3d = "contour", n = c(40, 40), levels = 60:100)
```

```
> mle.fit.s2 = mle2(start = list(S0 = 1100, R0 = 2), likelihood_R0,
+     method = "L-BFGS-B", lower = c(tot1, 0.5), upper = c(10 *
+         tot1, 10), data = list(I = niamey_cases2), control = list(ndeps = c(1,
+         0.001)))
> c2 = confint(mle.fit.s2)
> mle.fit.s3 = mle2(start = list(S0 = 1450, R0 = 2.2), likelihood_R0,
+     method = "L-BFGS-B", lower = c(tot1, 0.5), upper = c(10 *
+         tot1, 10), data = list(I = niamey_cases3), control = list(ndeps = c(1,
+         0.001)))
> c3 = confint(mle.fit.s3)
```

- Likelihood ratio test???
- Simulated annealing for more reliable estimation ???
- Fit on log scales, ditto???
- parscale??

```
> library(plotrix)
> R0vals = c(coef(mle.fit3)[2], coef(mle.fit.s2)[2], coef(mle.fit.s3)[2])
> cint = cbind(c1[2, ], c2[2, ], c3[2, ])
> suppressWarnings(plotCI(1:3, R0vals, li = cint[1, ], ui = cint[2,
+     ], axes = FALSE, ylab = ~R[0], xlab = "Site"))
> box()
> axis(side = 1, at = 1:3)
> axis(side = 2)
```

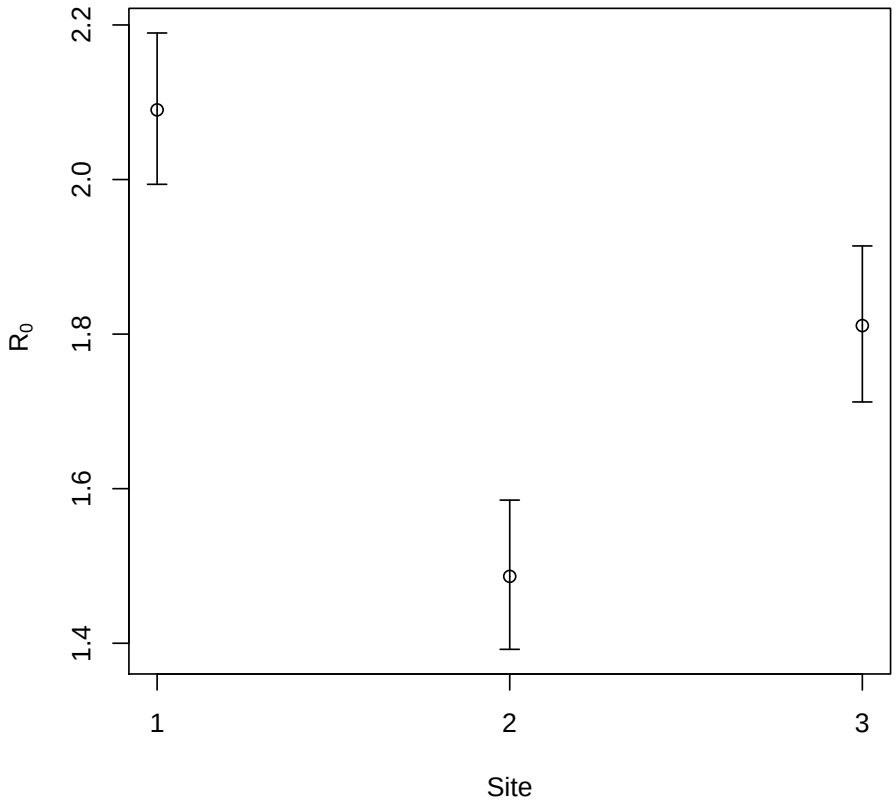


Figure 6: Comparison of confidence intervals for reporting centers in Niamey.