Estimating R_0: Solutions

John M. Drake and Pejman Rohani

Exercise 1. Show how this result could have been obtained graphically without the rearranged equation.

Here we use the influenza data discussed in the handout. First we load the data. The total number infected was given to us as $Z(\infty) = 512$. Given a population size of $N = 764$ we have the final epidemic size, expressed as a fraction, as $Z(\infty)/N = 512/764 = 0.6701571$.

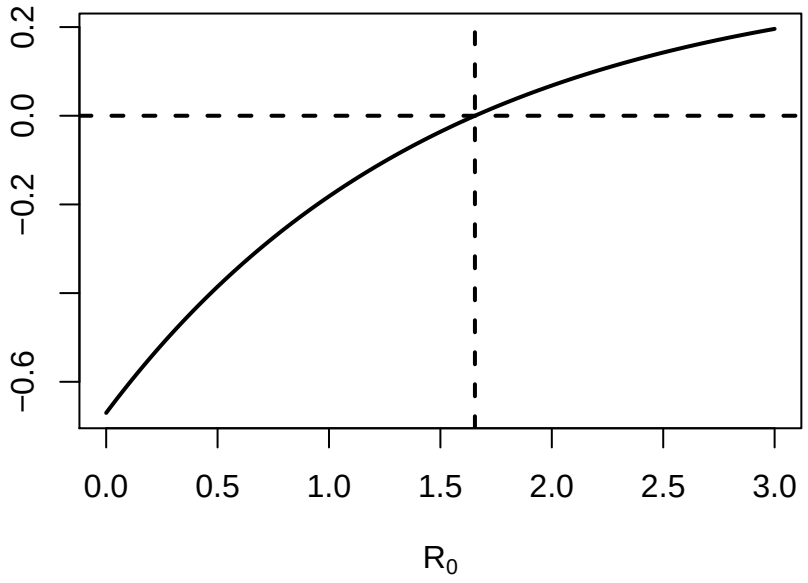
```
> load('data.RData')     #load the data and plot flu cases
> Z <- 512/764
```

We also need the formula for the final epidemic size.

```
> final.size.estimator<-function(R0,final.size) 1-final.size-exp(-final.size*R0)
```

To solve the problem graphically we plot the final size estimator as a function of R_0 (which ranges from 0 to 1). The R_0 for this epidemic is the value of the x-axis at which the y=axis is zero. We draw a horizontal line at $y = 0$ and a vertical line where this intersect the curve.

```
> x <- seq(0,3,by=0.01)
> y <- final.size.estimator(x,Z)
> plot(x, y, type='l', lwd=2, ylab='', xlab=expression(R[0]))
> abline(h=0, lty=2, lwd=2)
> abline(v=1.655, lty=2, lwd=2)
```



Evidently, R_0 is slightly greater than 1.5. This is consistent, of course, with our earlier calculation that $R_0 \approx 1.66$.

Exercise 2. This equation shows the important one-to-one relationship between R_0 and the final epidemic size. Plot the relationship between the total epidemic size and R_0 for the complete range of values between 0 and 1.

To solve this problem we first create a vector of total epidemic sizes from 0.001 to 0.999. (In the SIR epidemic, only with $R_0 = \infty$ can we get an epidemic with final size of 100%.)

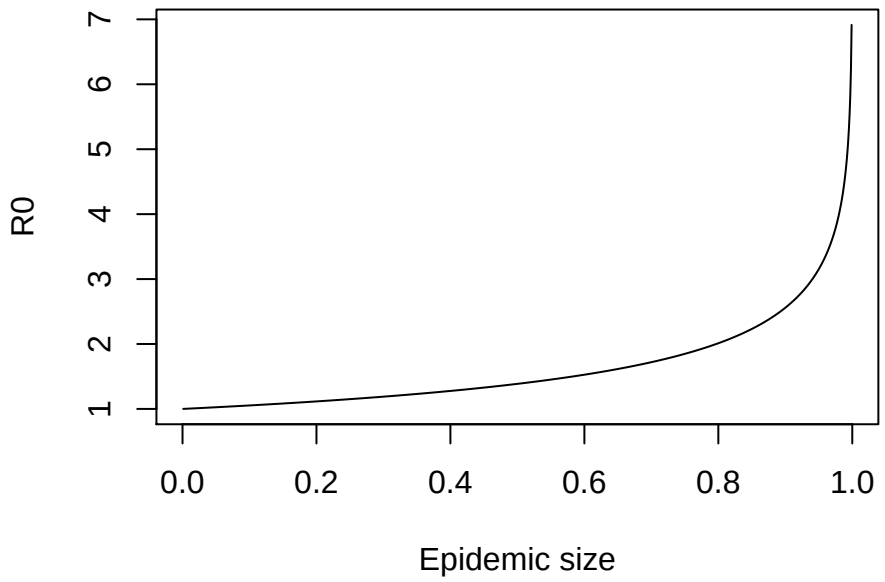
```
> epi.size <- seq(0.001, 0.999, by=0.001)
```

Here we loop over the values in this vector, using the formula $\hat{R_0} = \frac{log(1-Z/N)}{-Z/N}$ to get our estimate of R_0, and store the result in a vector.

```
> R0<-c()
> for(z in epi.size)  R0 <- c(R0, log(1-z)/-z) #estimate R0
```

Now we plot the result.

```
> plot(epi.size, R0, type='l', xlab='Epidemic size')
```



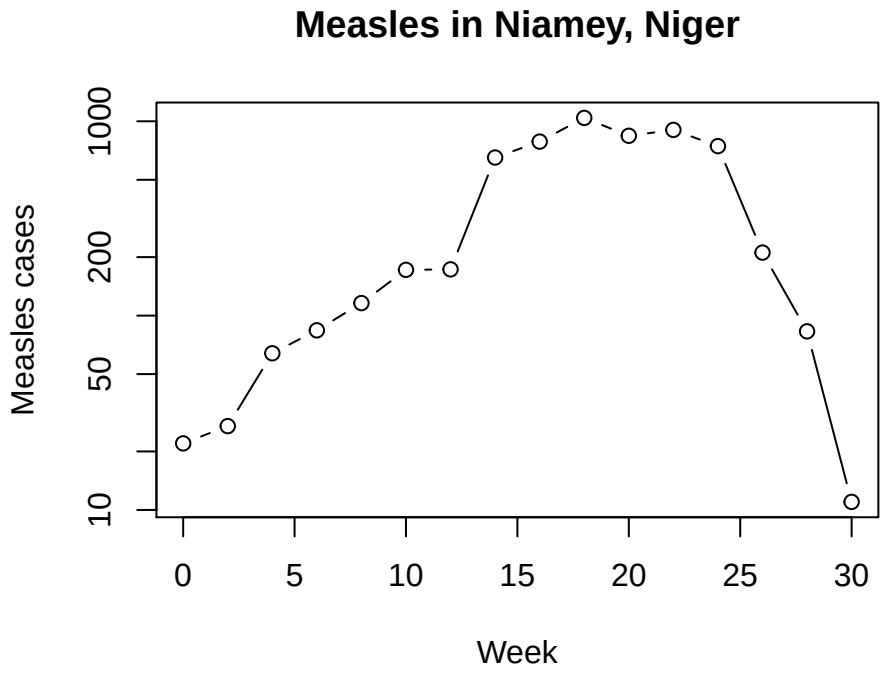
Exercise 3. Our estimate assumes that boys remained infectious during the natural course of infection. The original report on this epidemic indicates that boys found to have syptoms were immediately confined to bed in the infirmary. The report also indicates that only 1 out of 130 adults at the school exhibited any symptoms. It is reasonable, then, to suppose that transmission in each case ceased once he had been admitted to the infirmary. Supposing admission happened within 24 hours of the onset of symptoms. How does this affect our estimate of R_0? Twelve hours?

The formula to estimate R_0 is $\hat{R}_0 = \hat{\beta}_1/\gamma + 1$. If individuals are quarantined, the realized removal rate is greater than the natural recovery rate. Since the removal rate is in the denominator of the estimating equation, using the natural recovery rate positively biases our estimate (it is too big). For quarantine in 24 hours (or one day), we have $\gamma = 1$ yielding $\hat{R}_0 = 1.094913 + 1 \approx 2.09$. For qarantine in 12 hours (one half day), we have infectious period of $\gamma^{-1} = 0.5$ implying $\gamma = 2$, yielding $\hat{R}_0 = 1.094913/2 + 1 \approx 1.55$.

Exercise 4. Biweekly data for outbreaks of measles in three communities in Niamey, Niger are provided in the dataframe `niamey`. Use this method to obtain estimates of R_0 for measles from the first community assuming that the infectious period is approximately two weeks or $14/365 \approx 0.0384$ years.

Here we follow the influnza example. First we plot the data with a logarithmic y-axis.

```
> plot(seq(0,15*2,by=2),niamey[,1],type='b',log='y',
+      main='Measles in Niamey, Niger', xlab='Week', ylab='Measles cases')
```



Eye-balling this figure, the trajectory appears approximately linear on a log scale up to week 18. Since the data are reported in two-week intervals, this is the tenth observation. Here we fit the linear model.

```
> model<-lm(log(niamey[1:10,1])~seq(0,18,by=2))
> summary(model)         #summary statistics for fit model


Call:
lm(formula = log(niamey[1:10, 1]) ~ seq(0, 18, by = 2))

Residuals:
     Min       1Q   Median       3Q      Max
-0.51796 -0.07364  0.00790  0.10727  0.36805

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.03600    0.15111   20.09 3.93e-08 ***
seq(0, 18, by = 2)  0.21960    0.01415   15.52 2.96e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2571 on 8 degrees of freedom
Multiple R-squared:  0.9678,        Adjusted R-squared:  0.9638
F-statistic: 240.8 on 1 and 8 DF,  p-value: 2.963e-07


> slope<-coef(model)[2]  #extract slope parameter
> slope                  #print to screen
```

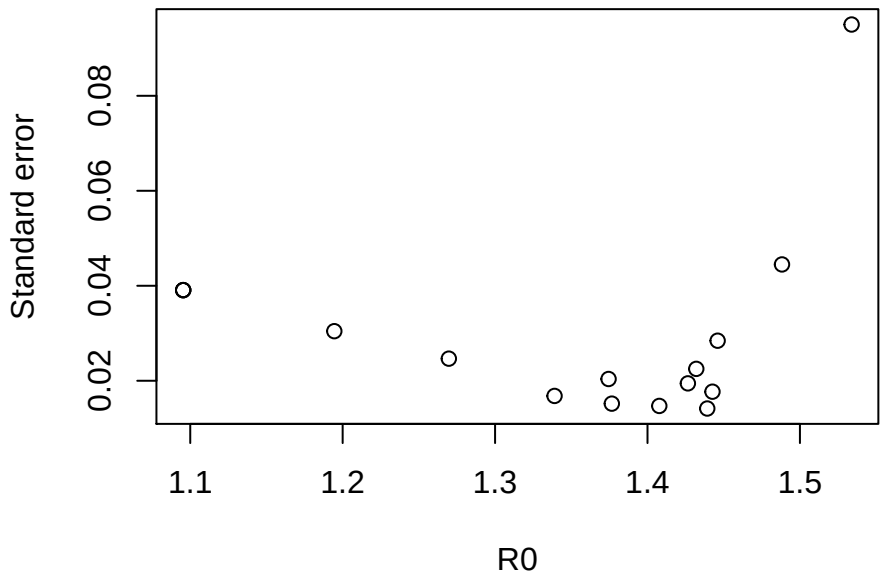
```
seq(0, 18, by = 2)
         0.2196041
```

Expressing the removal rate in units of weeks, we have $\gamma = 0.5$. Sustituting into our estimating formula, we have $\hat{R}_0 = 0.2196/0.5 + 1 \approx 1.44$

Exercise 5. A defect with this method is that it uses only a small fraction of the information that might be available, *i.e.*, the first few data points. Indeed, there is nothing in the method that tells one how many data points to use–this is a matter of judgment. Further, there is a tradeoff in that as more and more data points are used the precision of the estimate increases, but this comes at a cost of additional bias. Plot the estimate of R_0 obtained from $n = 3, 4, 5, ...$ data points against the standard error of the slope from the regression analysis to show this tradeoff.

To solve this problem we embed the regression approach used in the previous example in a for loop and extract the slope parameter and standard error. We calculate the desired estimate of R_0 after the loop as in the exercise above and plot.

```
> slope <- NULL
> se <- NULL
> for(i in 3:18){
+   model<-lm(log(niamey[1:i,1])~seq(0,(i-1)*2,by=2))
+   slope<-c(slope,as.numeric(coef(model)[2]))
+   se <- c(se, summary(model)$coefficients[4])
+ }
> R0 <- slope/0.5 + 1
> plot(R0, se, ylab='Standard error')
```



Exercise 6. To make things easier, we have assumed the infectious period is known to be 13 days. In terms of years, $\gamma = (365/13)^{-1} \approx 0.0357$. Now, modify the code above to estimate γ and β simultaneously.

To solve this problem we first plot the data.

```
> niamey[5,3]<-0  #replace a "NA"
> niamey<-data.frame(biweek=rep(seq(1,16),3),site=c(rep(1,16),rep(2,16),rep(3,16)), cases=c(niamey[,1],n
> plot(niamey$biweek,niamey$cases,type='p',col=niamey$site,xlab='Biweek',ylab='Cases')
> lines(niamey$biweek[niamey$site==1],niamey$cases[niamey$site==1])
> lines(niamey$biweek[niamey$site==2],niamey$cases[niamey$site==2],col=2)
> lines(niamey$biweek[niamey$site==3],niamey$cases[niamey$site==3],col=3)
```

Next we set up a version of the closed SIR model in which both β and γ are passed as parameters and a function to calculate the sum of squared errors obtained by comparing a model solution to the observed data.

```
> closed.sir.model <- function (t, x, params) {  #SIR model equations
+   S <- x[1]
+   I <- x[2]
+   b <- params[1]
+   gamma <- params[2]
+ #  gamma = log(13/365)
+   dS <- -b*S*I
+   dI <- b*S*I-gamma*I
+   list(c(dS,dI))
+ }
> sse.sir <- function(params0,data,site){  #function to calculate squared errors
+   data<-data[data$site==site,]    #working dataset, based on site
+   t <- data[,1]*14/365            #time in biweeks
+   cases <- data[,3]               #number of cases
+   b <- exp(params0[1])            #parameter beta
+   S0 <- exp(params0[2])           #initial susceptibles
+   I0 <- exp(params0[3])           #initial infected
+   gamma <- exp(params0[4])
+   out <- as.data.frame(lsoda(c(S=S0,I=I0),times=t,closed.sir.model,parms=c(b=b, gamma=gamma),hmax=1/12
+   sse<-sum((out$I-cases)^2)       #sum of squared errors
+ }
```

As in our earlier exercise, solving the differential equations will require the package `deSolve`.

```
> require(deSolve)   #differential equation library
```

Now we imitate the exercise with a guess at the initial parmeters, fit, and optimization using `optim`. Our question is answered by returning the estimates of β and γ.

```
> params0<-c(-3.2,7.3,-2.6, log(13/365))  #initial guess
> fit1 <- optim(params0,sse.sir,data=niamey,site=1, hessian=TRUE) #fit
> exp(fit1$par)  #back-transform parameters
```

```
[1] 4.270174e-03 1.306079e+04 3.109724e+00 3.514039e+01
```

```
> fit2 <- optim(params0,sse.sir,data=niamey,site=2) #fit
> exp(fit2$par)  #back-transform parameters
```

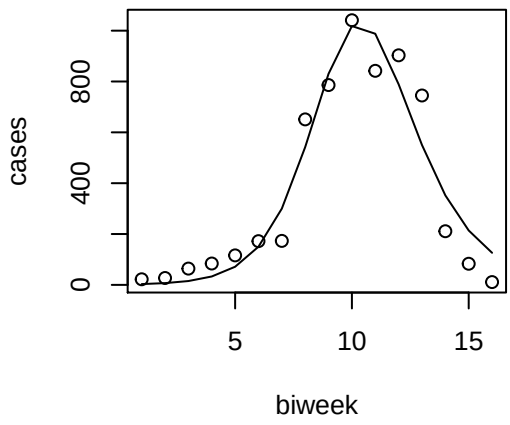
```
[1] 6.079919e-02 1.392372e+03 1.418267e-08 8.256003e+00
```

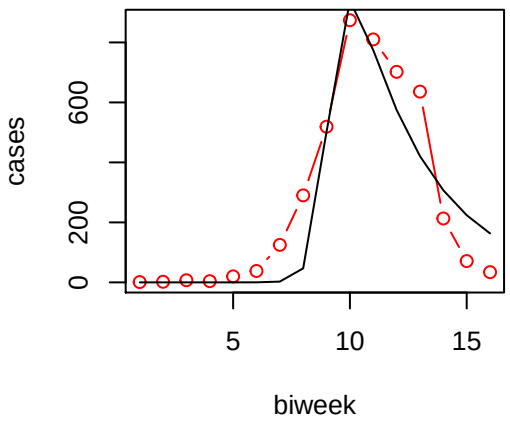
```
> fit3 <- optim(params0,sse.sir,data=niamey,site=3) #fit
> exp(fit3$par)  #back-transform parameters
```

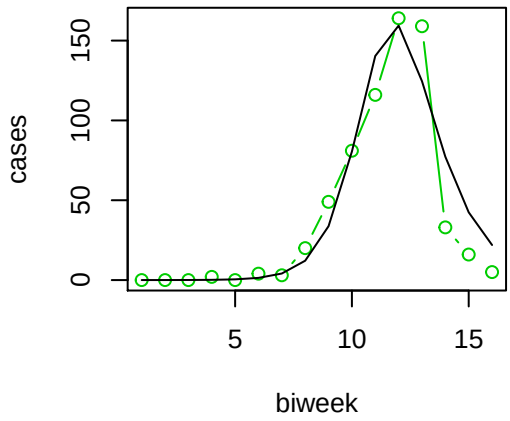
```
[1] 4.857679e-02 1.329556e+03 5.226157e-03 3.555167e+01
```

Finally, we plot the fit model against the data.

```
> par(mfrow=c(2,2))   #set up plotting area for multiple panels
> plot(cases~biweek,data=subset(niamey,site==1),type='p',pch=21) #plot site 1
> t <- subset(niamey,site==1)[,1]*14/365
> mod.pred<-as.data.frame(lsoda(c(S=exp(fit1$par[2]),I=exp(fit1$par[3])),times=t,
+                               closed.sir.model,c(exp(fit1$par[1]),exp(fit1$par[4])),hmax=1/120))
>                               #obtain model predictions
> lines(mod.pred$I~subset(niamey,site==1)[,1]) #and plot as a line
> plot(cases~biweek,data=subset(niamey,site==2),type='b',col=site) #site 2
> t <- subset(niamey,site==2)[,1]*14/365
> mod.pred<-as.data.frame(lsoda(c(S=exp(fit2$par[2]),I=exp(fit2$par[3])),times=t,
+                               closed.sir.model,c(exp(fit2$par[1]),exp(fit2$par[4])),hmax=1/120))
> lines(mod.pred$I~subset(niamey,site==2)[,1])
> plot(cases~biweek,data=subset(niamey,site==3),type='b',col=site) #site 3
> t <- subset(niamey,site==3)[,1]*14/365
> mod.pred<-as.data.frame(lsoda(c(S=exp(fit3$par[2]),I=exp(fit3$par[3])),times=t,
+                               closed.sir.model,c(exp(fit3$par[1]),exp(fit3$par[4])),hmax=1/120))
> lines(mod.pred$I~subset(niamey,site==3)[,1])
```







Exercise 7. What happens if one or both of the other unknowns (I_0 and S_0) is fixed instead of γ?

Here we let I_0 be an unknown parameter and we fit it instead of γ.

As before, we write a function to return the derivatives of the closed SIR epidemic. Notice that `closed.sir.model` treats I as a state variable, but `sse.sir` treats I_0 as an unknown parameter to be fitted.

```
> closed.sir.model <- function (t, x, params) {  #SIR model equations
+   S <- x[1]
+   I <- x[2]
+   b <- params[1]
+   gamma <- params[2]
+ #  gamma = log(13/365)
+   dS <- -b*S*I
+   dI <- b*S*I-gamma*I
```

```
+   list(c(dS,dI))
+ }
> sse.sir <- function(params0,data,site){  #function to calculate squared errors
+   data<-data[data$site==site,]    #working dataset, based on site
+   t <- data[,1]*14/365            #time in biweeks
+   cases <- data[,3]               #number of cases
+   b <- exp(params0[1])            #parameter beta
+   S0 <- 12500           #initial susceptibles
+   I0 <- exp(params0[2])           #initial infected
+   gamma <- exp(params0[3])
+   out <- as.data.frame(lsoda(c(S=S0,I=I0),
+                              times=t,
+                              closed.sir.model,
+                              parms=c(b=b, gamma=gamma),
+                              hmax=1/120))
+   sse<-sum((out$I-cases)^2)       #sum of squared errors
+ }
```

As above, we proceed with an initial guess at parameters and use `optim` to fit the model.

```
> params0<-c(-3.2,-2.6, log(13/365))  #initial guess
> fit1 <- optim(params0,sse.sir,data=niamey,site=1) #fit
> exp(fit1$par)  #back-transform parameters

[1]  0.004440909  2.802295901 34.568270916

> fit2 <- optim(params0,sse.sir,data=niamey,site=2) #fit
> exp(fit2$par)  #back-transform parameters

[1] 1.932654e-01 1.176771e-04 5.053424e+02

> fit3 <- optim(params0,sse.sir,data=niamey,site=3) #fit
> exp(fit3$par)  #back-transform parameters

[1] 3.628534e-01 6.102647e-07 3.073515e+05
```