Chapter 8
Trajectory Matching

8.1 Preamble: Prevalence *Versus* Incidence

When we fit mechanistic models to data, we have to consider carefully the relationship between the nature of the data *versus* the nature of the model state variables. For example, when we work with continuous-time S(E)IR models it is important to keep in mind that *incidence is not prevalence*; so if our data is incidence we will need to do something more than trying to match simulated prevalence with observed incidence. We therefore start with a toy example using simulated data.

When/if we can assume that dynamics is unaffected by *process noise* (demographic and environmental stochasticity), we can fit models to data using trajectory matching. The assumption is that discrepancies between the observations and the predictions from the dynamic model are due to observational errors. The upside of trajectory matching is that we can easily fit continuous-time models to variably spaced observations on any/all state-variable, the downside is that these assumptions are usually restrictive.

8.2 Event-Based Stochastic Simulation

To begin, we will consider how to stochastically simulate the continuous-time SIR model (Eqs. (2.1)–(2.3)). Previously we consider stochastic simulation using discrete-time models. An alternative is to do continuous-time stochastic simulation using an event-based approach: The Gillespie exact algorithm (Gillespie 1977) and

This chapter uses the following R-package: `deSolve`.

the τ-leap approximation (Gillespie 2001). As discussed in Sect. 2.7, the S(E)IR-model (and all simple ODEs) implies exponentially distributed waiting times between events. The Gillespie algorithms take advantage of this idea. If we for example consider how the SIR-states of the SIR flows ((2.1)–(2.3)) should change over time, we expect the following six possible changes:

- $S \to S+1$ at rate μN from births
- $S \to S-1$ at rate μS from deaths
- $S \to S-1$ and $I \to I+1$ at rate $\beta SI/N$ from infection
- $I \to I-1$ at rate μI from deaths
- $I \to I-1$ and $R \to R+1$ at rate γI from recovery
- $R \to R-1$ at rate μR from deaths

Thus, the system is expected to change by an overall summed rate of $r = \mu N + \mu S + \beta SI/N + \mu I + \gamma I + \mu R$. We can therefore draw a random exponential waiting-time with mean r to update a continuous-time clock, then draw a random event from a multinomial distribution with probabilities given by the relative rates, update the state variables accordingly, and repeat...

Because of the many versions of compartmental models used in studying disease dynamics, it is useful to write a general purpose stochastic simulator that can be applied to any set of rate equations. To this end we first define a `rlist`-list of equations corresponding to the rates for the six transitions of the SIR flows. The `quote`-formalism allows us to set up the list such that all equations can be evaluated in a single `sapply`-call as the simulation progress.

```
rlist=c(quote(mu * (S+I+R)), #Births
quote(mu * S), #Susceptible deaths
quote(beta * S * I /(S+I+R)), #Infection
quote(mu * I), #Infected death
quote(gamma * I), #Recovery
quote(mu*R)) #Recovered death
```

We next define a transition matrix associated with each SIR event. The three columns correspond to changes in S, I, and R, respectively; The rows correspond to the six possible events.

```
emat=matrix(c(1,0,0,
-1,0,0,
-1,1,0,
0,-1,0,
0,-1,1,
0,0,-1),
ncol=3, byrow=TRUE)
```

We finally write a general-purpose function to simulate a dynamical systems using the Gillespie algorithm. The idea is to write a function that is sufficiently robust and general that it can be applied to event-based stochastic simulation of any model that fits within a compartmental framework. The function takes five arguments to accomplish this:

- `rateqs`—a list of `E` rate equations corresponding to each of the `E` possible events using the `quote`-formalism
- `eventmatrix`—a `E`-by-`S` matrix of changes to each of the `S` state variables associated with each event
- `parameters`—a vector of parameter values
- `initialvals`—a vector of initial values for the `S` states
- `numevents`—number of events to be simulated

```
gillespie=function(rateqs, eventmatrix, parameters,
     initialvals, numevents){
   res=data.frame(matrix(NA, ncol=length(initialvals)+1,
      nrow=numevents+1))
   names(res)=c("time", names(inits))
   res[1,]=c(0, inits)
   for(i in 1:numevents){
   #evaluate rates
   rat=sapply(rateqs, eval,
      as.list(c(parameters, res[i,])))
   #update clock
   res[i+1,1]=res[i,1]+rexp(1, sum(rat))
   #draw event
   whichevent=sample(1:nrow(eventmatrix), 1, prob=rat)
   #updat states
   res[i+1,-1]=res[i,-1]+eventmatrix[whichevent,]
   }
return(res)
}
```

We provide parameters and initial conditions for a stochastic simulation assuming an infectious period of 20 days (Fig. 8.1):

```
paras=c(mu=1, beta=500, gamma=365/20)
inits=c(S=100, I=2, R=0)
sim=gillespie(rlist, emat, paras, inits, 1000)
matplot(sim[,1],sim[,2:4], type="l", ylab="Numbers",
     xlab="Time", log="y")
legend("topright", c("S", "I", "R"), lty=c(1,1,1),
     col=c(1,2,3))
```

The Gillespie algorithm provides an "exact" stochastic simulation in the sense that the time-evolution of the system is changing exactly according to the expo-

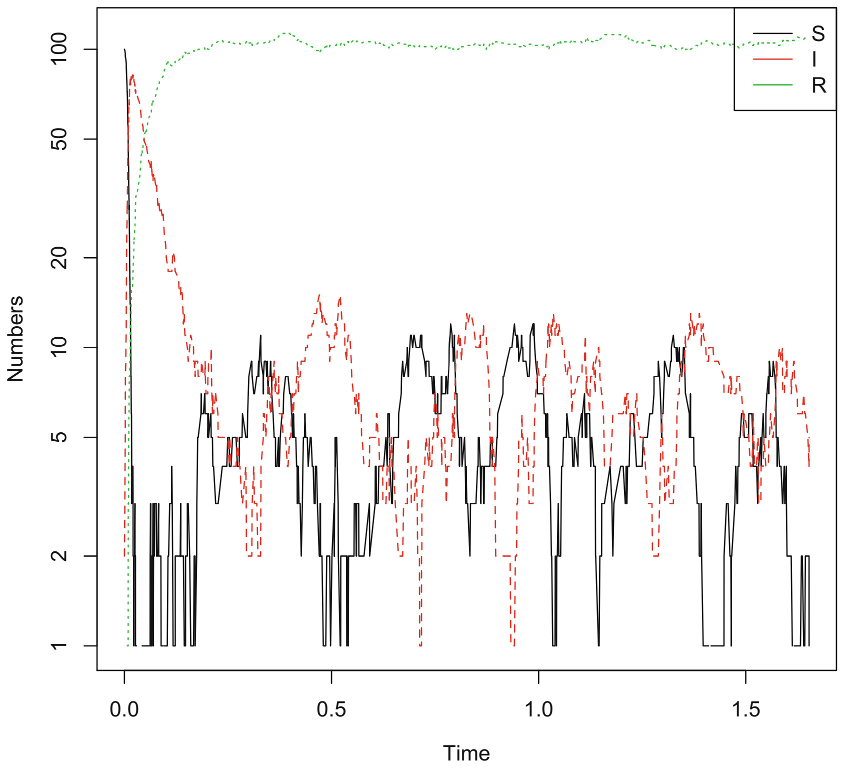

Fig. 8.1 A Gillespie exact simulation of the stochastic SIR model with $\mu = 1$, $\beta = 500$, and $\gamma = 365/20$

nential waiting-time distributions of the stochastic differential system. It is, however, computationally expensive as every event is recorded separately. Gillespie's τ-leap method uses the Poisson approximation corresponding to the discussion of Sect. 7.1; If we assume that the interval, Δt, is sufficiently short that any change in the rates are negligible, the number of events should be Poisson-distributed with mean $overallrate * \Delta t$ and multinomially divided among the events according to their relative rates.

We write a general τ-leap simulator and then apply it to the SEIR model. The SEIR model has eight possible events:

- $S \to S+1$ at rate μN from births
- $S \to S-1$ at rate μS from deaths
- $S \to S-1$ and $E \to E+1$ at rate $\beta SI/N$ from infection
- $E \to E-1$ at rate μE from deaths
- $E \to E-1$ and $I \to I+1$ at rate σE from becoming infectious
- $I \to I-1$ at rate μI from deaths
- $I \to I-1$ and $R \to R+1$ at rate γI from recovery
- $R \to R-1$ at rate μR from deaths

We thus have the following event matrix:

```
emat2=matrix(c(1,0,0,0,
-1,0,0,0,
-1,1,0,0,
0,-1,0,0,
0,-1,1,0,
0,0,-1,0,
0,0,-1,1,
0,0,0,-1),
ncol=4, byrow=TRUE)
```

The SEIR equations associated with each event are:

```
rlist2=c(quote(mu * (S+E+I+R)), quote(mu * S),
     quote(beta * S * I/(S+E+I+R)), quote(mu*E),
     quote(sigma * E), quote(mu * I),
     quote(gamma * I), quote(mu*R))
```

A general-purpose τ-leap simulator is:

```
tau=function(rateqs, eventmatrix, parameters,
     initialvals, deltaT, endT){
time=seq(0, endT, by=deltaT)
res=data.frame(matrix(NA, ncol=length(initialvals)+1,
     nrow=length(time)))
res[,1]=time
names(res)=c("time", names(inits))
res[1,]=c(0, inits)
for(i in 1:(length(time)-1)){
     #calculate overall rates
     rat=sapply(rateqs, eval, as.list(c(parameters,
          res[i,])))
     evts=rpois(1,  sum(rat)*deltaT)
     if(evts>0){
     #draw events
     whichevent=sample(1:nrow(eventmatrix), evts,
          prob=rat, replace=TRUE)
     mt=rbind(eventmatrix[whichevent,],
          t(matrix(res[i,-1])))
     mt=matrix(as.numeric(mt), ncol=ncol(mt))
     #update states
     res[i+1,-1]=apply(mt,2,sum)
     res[i+1, ][res[i+1,]<0]=0
     }
     else{ #if no events in delaT
     res[i+1,-1]=res[i,-1]
     }}
return(res)
}
```

We assume an initial population comprised of 1000 individuals and 1 initial infected and simulate daily incidence for 2 years and assume measles-like parameters:

```
paras  = c(mu = 1, beta =  1000,
     sigma = 365/8, gamma = 365/5)
inits = c(S=999, E=0, I=1, R = 0)
sim2=tau(rlist2, emat2, paras, inits, 1/365, 2)
matplot(sim2[,1],sim2[,2:5], type="l", log="y",
     ylab="Numbers", xlab="Time")
legend("bottomright", c("S", "E", "I", "R"),
     lty=c(1,1,1,1), col=c(1,2,3,4))
```

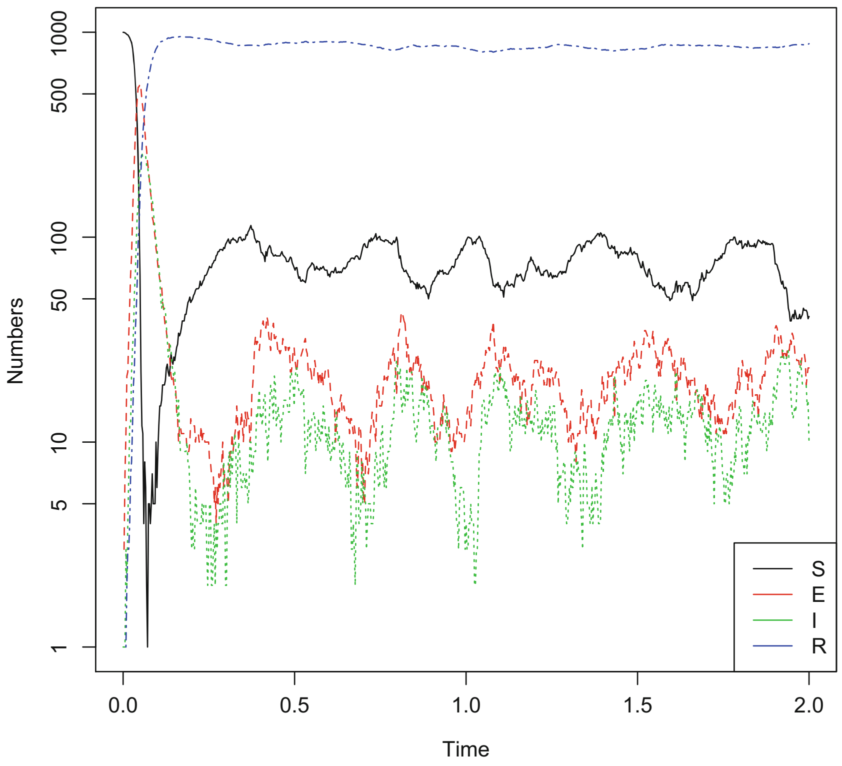

Fig. 8.2 A τ-leap simulation of the SEIR model using a daily time step for 2 years assuming $\mu = 1$, $\beta = 1000$, an infectious period of 8 days, a latent period of 5 days, and an initial population comprised of 1000 individuals one of which is infected

Following the virgin epidemic, the inherent birth/death stochasticity leads to low-amplitude oscillations (Fig. 8.2) according to the resonant periodicity of the SEIR model (see Chap. 9).

8.3 Trajectory Matching

Trajectory matching assumes that the discrepancies between models and data are due to error of observation. The event-based, stochastic simulation breaks with this assumption as model discrepancies are due to demographic stochasticity; Let us nevertheless see if we can fit the SEIR model to the event-based simulation. We first recall the gradient-function for the system:

```
require(deSolve)
seirmod = function(t, y, parms) {
    S = y[1]
    E = y[2]
    I = y[3]
    R = y[4]

    with(as.list(parms), {
        dS = mu * (N - S) - beta * S * I/N
        dE = beta * S * I/N - (mu + sigma) * E
        dI = sigma * E - (mu + gamma) * I
        dR = gamma * I - mu * R
        res = c(dS, dE, dI, dR)
        list(res)
    })
}
```

Following the ideas introduced in Sect. 3.4, we define a likelihood function to estimate parameters. The Gaussian log-likelihood is $= \text{const} - \frac{n}{2}\log(\text{RSS})$, where n is the length of the time series, `RSS` is the residual sum-of-squares, and the constant is $n(log(n) - log(2\pi) - 1)/2$ (Aitkin et al. 2005).[1]

```
lfn=function(p){
     times = seq(0, 2, by=1/365)
     start = c(S=999, E=0, I=1, R = 0)
     paras=exp(c(mu=p[1], N=p[2], beta=p[3],
```

[1] If in a hurry we can ignore the constant and minimize $\frac{n}{2}\log(RSS)$ because it is the relative likelihood that matters.

```
        sigma=p[4], gamma=p[5]))
     out = as.data.frame(ode(start, times=times,
        seirmod, paras))
     n=length(sim2$I)
     rss=sum((sim2$I-out$I)^2)
     return(log(rss)*(n/2)-n*(log(n)-log(2*pi)-1)/2)
}
```

We next estimate parameters:

```
# initial values for mu, N, beta, sigma, gamma
paras0 = log(c(2, 500, 500, 365/7, 365/7))
fit = optim(paras0, lfn, hessian = TRUE)
```

and plot the deterministic prediction:

```
times = seq(0, 2, by=1/365)
paras  = exp(c(mu = fit$par[1], N = fit$par[2],
     beta = fit$par[3], sigma = fit$par[4],
     gamma = fit$par[5]))
start = c(S=999, E=0, I=1, R = 0)
out = as.data.frame(ode(start, times, seirmod, paras))
plot(out$time, out$I, xlab="Time", ylab="Prevalence",
     type="l")
lines(sim2$time, sim2$I, col=2, type="l")
legend("topright", c("Gillespie simulation",
     "SEIR fit"), lty=c(1,1), col=c(2,1))
```

The trajectory-match'ed fit predicts the virgin epidemic and the next dampened epidemic well, but not—as expected—the subsequent stochastically excited low-amplitude cycles (Fig. 8.3). In addition to finding parameter estimates we are usually interested in uncertainty and trade-offs among parameters in producing a fit to the data.

8.4 Likelihood Theory 101

We have used maximum likelihood principles in several of our previous analysis of, for example, the chain-binomial, the catalytic and the TSIR models. We have, however, not discussed likelihood theory in a formal fashion.[2] For our purposes it

[2] Bolker (2008) is an excellent broad discussion on estimation for ecologically realistic models using a variety of methods.

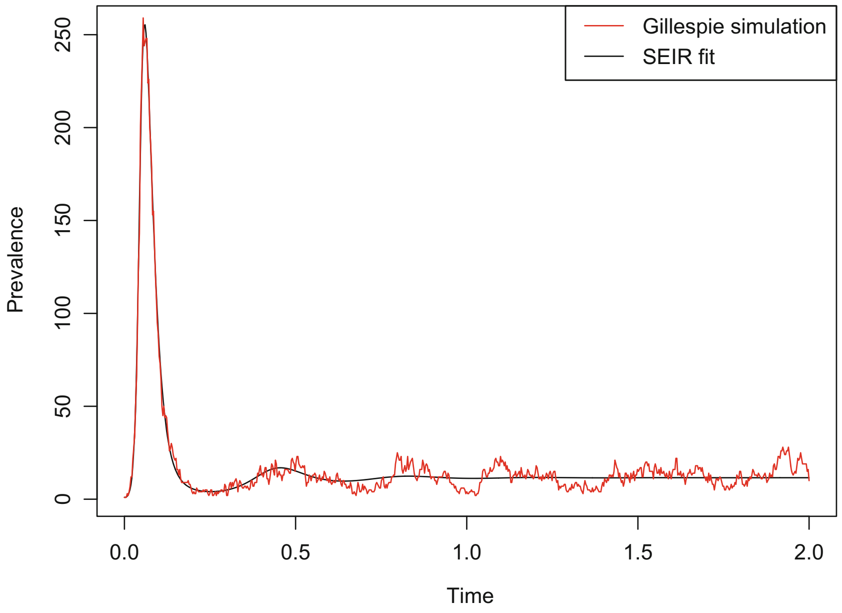

Fig. 8.3 SEIR fitted predicted trajectory superimposed on the τ-leap simulation of the SEIR model

is useful to summarize the key results with respect to inference from "elementary" likelihood theory with maximum brevity (see, for example, appendix A of McCullagh and Nelder 1989):

- Let $L(D|\theta)$ be the function that calculates the likelihood for a set of data, D; i.e., the probability of observing the data given some values for the parameters, θ. The values that maximize this probability are the maximum likelihood estimates (MLEs) of the parameters, $\hat{\theta}$.
- If $\ell(\theta)$ is the negative log-likelihood (i.e., $-\log L$), then $\hat{\theta}$ are the values that minimizes ℓ. If data points are independent, then the joint log-likelihood is simply the sum of the log-likelihoods of the data points.
- The MLE is a minimum of ℓ, so the *score function* $U(\theta) = \partial\ell/\partial\theta$ is zero at the MLE.
- The likelihood profile graphs how $\ell(\theta)$ changes with θ. The 95% confidence interval is the set of values of θ for which $\ell(\theta)$ is within $\chi^2(0.95, p)/2$ of the minimum, where p is the number of parameters. The quantity $2\ell(\theta)$ is referred to as the *deviance*, so if we work with the deviance we would use $\chi^2(0.95, p)$ as the cut-off.
- The second derivative of $\ell(\theta)$ with respect to θ is called the *Fisher information*, $\iota(\theta) = \partial^2\ell/\partial\theta^2$. The inverted information matrix is an approximation to the variance-covariance matrix of the parameters, so we can obtain approximate standard errors as the square-root of the diagonal of the inverted information ma-

trix. The approximate correlation matrix is the standardized inverted information matrix.
- A matrix of second derivatives is generally referred to as a Hessian matrix. If we call `optim(..., hessian=TRUE)`, R will numerically estimate the Hessian at the minimum, so if the function to be minimized is the negative log-likelihood, we can obtain approximate SEs and the approximate correlation matrix from this Hessian.
- If we have two alternative models that are *nested*—meaning that the more complex model contains all the parameters of the simpler—then we can test for significant model improvement; the difference in the log-likelihood is $\chi^2(df = \Delta p)/2$-distributed, where Δp is the number of extra parameters in the complex model.[3]

We apply these ideas to our model fit:

```
# MLEs:
round(exp(fit$par), 4)

  ## [1]   1.5165 642.3009 563.3407  50.3001  69.6786

# Approximate SEs:
round(exp(sqrt(diag(solve(fit$hessian)))), 4)

  ## [1] 1.2744 1.2441 1.2561 1.0224 1.0240

# Correlation matrix:
round(cov2cor(solve(fit$hessian)), 4)

  ##         [,1]    [,2]    [,3]    [,4]    [,5]
  ## [1,]  1.0000 -0.9976 -0.9972  0.6107 -0.9546
  ## [2,] -0.9976  1.0000  0.9974 -0.5872  0.9649
  ## [3,] -0.9972  0.9974  1.0000 -0.6421  0.9543
  ## [4,]  0.6107 -0.5872 -0.6421  1.0000 -0.4653
  ## [5,] -0.9546  0.9649  0.9543 -0.4653  1.0000
```

The true parameter values used in the simulation were $\mu = 1$, $N = 1000$, $\beta = 1000$, $\sigma = 45.6$, and $\gamma = 73$. So while the model prediction gives a good fit, the parameter estimates are not particularly accurate. This is where it is useful to use likelihood theory more extensively. From the normalized inverted Hessian we see that several of the parameters are highly (positively or negatively) correlated, and several with correlations more extreme than ± 0.9. That means that different parameter combination may provide a very similar fit to the data. This is an illustration of *identifiability problems*; With observations only on the infectious stage, for instance, a relatively short infectious period and high transmission rate will predict a

[3] If the models are *non-nested*, formal tests are not available but information theoretical rankings of models using AIC, BIC, AIC-weights, etc. are useful.

similar trajectory to a relatively short latent period and a lower transmission rate. Furthermore, a smaller population size and higher birth rate can result in identical susceptible recruitment rate than a larger population with lower birth rate. For inference it is therefore normally best to inform the analysis with any known biological quantities; For example if the latent and infectious periods are known from household or clinical studies, it may be best not to attempt to infer these from the time series alone (though, as King et al. (2008) point out for Cholera dynamics, conventional wisdom may not always be consistent with dynamical patterns). Moreover, if there are strong correlations, the individual SEs (and CIs derived there from) may be a poor representation of parametric uncertainty. It may then be better to look at pairwise confidence ellipses (e.g., Bolker 2008).

8.5 SEIR with Error

We can use `ode` to integrate the SEIR model and add noise, to generate a data set that exactly adheres to the assumption that the dynamics is only affected by observational noise. Let us simulate 10 years of weekly data assuming measles'ish parameters and that 6% of the initial population is susceptible:

```r
times  = seq(0, 10, by=1/52)
paras  = c(mu = 1/50, N = 1, beta =  1000,
     sigma = 365/8, gamma = 365/5)
     start = c(S=0.06, E=0, I=0.001, R = 0.939)
out = as.data.frame(ode(start, times, seirmod, paras))
```

We add noise to the data using the `jitter`-function (Fig. 8.4),

```r
datay = jitter(out$I, amount = 1e-04)
```

```r
plot(times, datay, ylab = "Infected", xlab = "Time")
lines(times, out$I, col = 2)
```

define a Gaussian likelihood function,

```r
lfn=function(p, data){
     times = seq(0, 10, by=1/52)
     start = c(S=0.06, E=0, I=0.001, R = 0.939)
     paras=c(mu=p[1], N=p[2], beta=p[3],
        sigma=p[4], gamma=p[5])
     out = as.data.frame(ode(start, times=times,
        seirmod, paras))
     n=length(data)
```

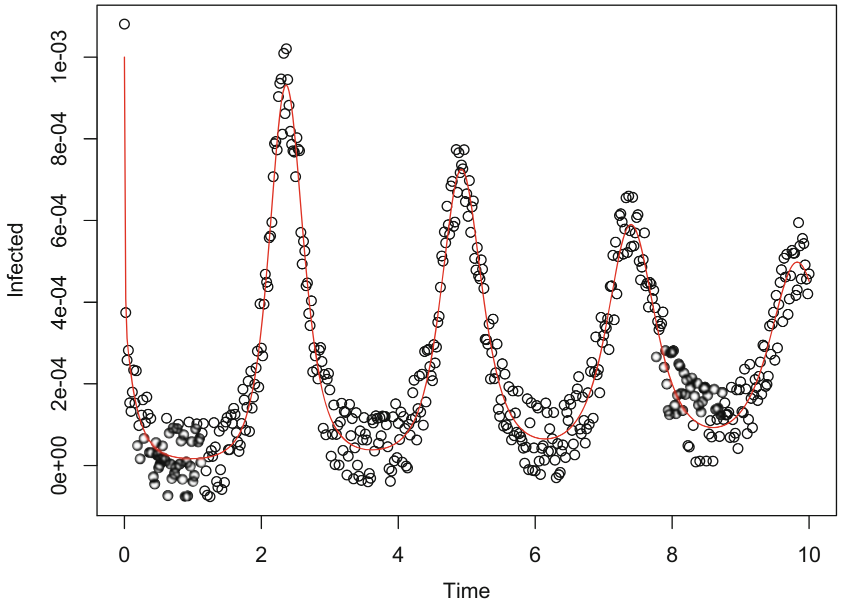

Fig. 8.4 Fraction infectious and jittered data from the SEIR model assuming $\mu = 0.02$, $\beta = 1000$, $\sigma = 45.6$, and $\gamma = 73/\text{year}$

```
     rss=sum((data-out$I)^2)
     return(log(rss)*(n/2)-n*(log(n)-log(2*pi)-1)/2)
 }
```

and estimate parameters using the jittered observations.

```
# mu, N, beta, sigma, gamma
paras0 = c(1/30, 1, 1500, 365/4, 365/10)
fit = optim(paras0, lfn, data = datay, hessian = TRUE)
```

The estimates are

```
# MLEs:
round(fit$par, 3)

  ## [1]    0.036    1.031 2179.946   71.197  138.851

# Approximate SEs:
round(sqrt(diag(solve(fit$hessian))), 3)

  ## [1]   0.003   0.066 200.261   7.584   8.718

# Correlation matrix:
round(cov2cor(solve(fit$hessian)), 3)
```

```
##         [,1]   [,2]   [,3]   [,4]   [,5]
## [1,]  1.000 -0.743 -0.330 -0.407  0.374
## [2,] -0.743  1.000  0.820 -0.258  0.204
## [3,] -0.330  0.820  1.000 -0.658  0.714
## [4,] -0.407 -0.258 -0.658  1.000 -0.821
## [5,]  0.374  0.204  0.714 -0.821  1.000
```

8.6 Boarding School Flu Data

The boarding school flu data set introduced in Sect. 3.6.1 has an approximate match between *observation* and *prevalence* because the data represents the number of children confined to bed each day, and while the average stay in bed (3–7 days) is maybe a bit different than the infectious period, the durations are comparable.

```
data(flu)
```

We define the gradient functions for a closed SIR epidemic:

```
sirmod = function(t, y, params) {
    S = y[1]
    I = y[2]
    R = y[3]
    with(as.list(params), {
        dS = -beta * S * I/N
        dI = beta * S * I/N - gamma * I
        dR = gamma * I
        res = c(dS, dI, dR)
        list(res)
    })
}
```

and define the likelihood function assuming normally distributed errors

```
lfn2 = function(p, I, N) {
     times = seq(1, 14, by = 1)
     start = c(S = N, I = 1, R = 0)
     paras = c(beta = p[1], gamma = p[2], N = N)
     out = as.data.frame(ode(start, times = times,
        sirmod, paras))
     n = length(I)
     rss = sum((I - out$I)^2)
     return(log(rss) * (n/2) - n * (log(n) -
        log(2 * pi) - 1)/2)
}
```

There are two parameters to estimate: β and γ. The time-scale is daily so we set reasonable initial conditions and maximize the likelihood:

```
#beta, gamma
paras0 = c(1.5, 1/2)
flufit = optim(paras0, lfn2, I = flu$cases, N = 763,
     hessian = TRUE)
```

The estimated parameters and basic reproductive ratio, R_0, are:

```
# parameters
flufit$par

  ## [1] 1.9566375 0.4738335

# R0:
flufit$par[1]/flufit$par[2]

  ## [1] 4.129377
```

The R_0 estimate is comparable to the estimate we made in Chap. 3. The observed and predicted outbreaks are seemingly a good match (Fig. 8.5):

```
times = seq(1, 20, by=.1)
start = c(S=762, I=1, R = 0)
paras=c(beta=flufit$par[1], gamma=flufit$par[2], N=763)
out = as.data.frame(ode(start, times=times,
     sirmod, paras))
plot(out$time, out$I, ylab="Prevalence",
     xlab="Day", type="l")
points(flu$day, flu$cases)
```

8.7 Measles

We consider, again, the measles incidence data collected by Doctors Without Borders (MSF) during the 03/04 outbreak in Niamey, Niger (Fig. 8.6; Sect. 3.4) but using data at a daily resolution. We can compile the daily incidence in a vector `y`.

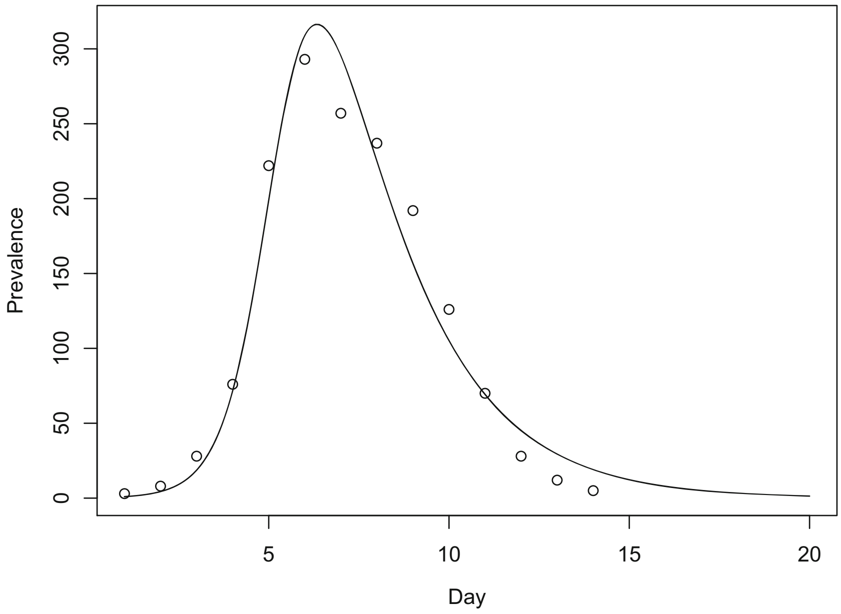

Fig. 8.5 Predicted and observed influenza prevalence for the 1978 boarding school data

```r
y = as.vector(table(niamey_daily))
```

The challenge with this data is that we need to make the SEIR-formulation relevant to the data on incidence. The complication is that `I` represents *prevalence* (i.e., current number of infected individuals), while incidence, `y`, represents appearance of new cases (i.e., *flux*) into the infected class. If we recast the SEIR model to also keep track of cumulative incidence, K, we can *difference* the `K` time series at time-steps corresponding to that of the observations to predict incidence (`y`). We define the SEIRK-model assuming known latent and infectious periods of 8 and 5 days, respectively.

```r
times= unique(niamey_daily$day)
paras  = c(mu = 0, N = 1, beta =  5, sigma = 1/8,
     gamma =1/5)
start = c(S=0.999, E=0, I=0.001, R = 0, K  = 0)
```

The resultant gradient function is:

```r
seirkmod = function(t, x, params) {
    S = x[1]
    E = x[2]
    I = x[3]
    R = x[4]
    K = x[5]

    with(as.list(params), {
        dS = mu * (N - S) - beta * S * I/N
        dE = beta * S * I/N - (mu + sigma) * E
        dI = sigma * E - (mu + gamma) * I
        dR = gamma * I - mu * R
        dK = sigma * E
        res = c(dS, dE, dI, dR, dK)
        list(res)
    })
}
```

We next define the likelihood function (assuming Poisson distributed errors) for the unknown transmission rate, β, and initial susceptible number, N. According to the MSF outbreak response protocol, an outbreak is declared once five cases have been confirmed. The unknown infectious fraction is thus $5/N$.

```
lfn4=function(p, I){
times=unique(niamey_daily$day)
xstart=c(S=(p[1]-5)/p[1], E=0, I=5/p[1], R = 0,
     K=0)
paras=c(mu=0, N=p[1], beta=p[2], sigma=1/8,
     gamma=1/5)
out=as.data.frame(ode(xstart, times=times, seirkmod,
     paras))
predinci=c(xstart["I"], diff(out$K))*p[1]
ll=-sum(dpois(I, predinci, log=TRUE))
return(ll)
}
```

For starting values we assume initial susceptible numbers $N = 11{,}000$ and $\beta = 5$ and optimize:

```
#N, beta
paras0  = c(11000, 5)
measfit=optim(paras0,lfn4,I=y, hessian=TRUE)
day = 1:230
xstart = c(S=(measfit$par[1]-5)/measfit$par[1], E=0,
     I=5/measfit$par[1], R = 0, K  = 0)
paras=c(mu=0, N=measfit$par[1], beta=measfit$par[2],
     sigma=1/8, gamma=1/5)
out = as.data.frame(ode(xstart, times=day,
     seirkmod, paras))
```

```
plot(table(niamey_daily), xlab="Day", ylab="Incidence")
lines(out$time, c(xstart["I"], diff(out$K))*
     measfit$par[1], col=2, lwd=2)
```

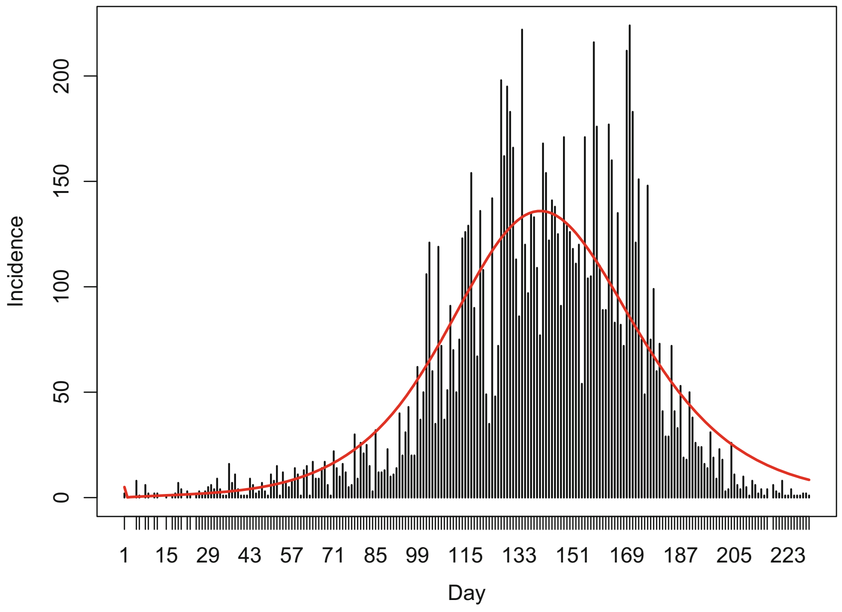

Fig. 8.6 Predicted and observed measles incidence using the MLEs from the Poisson-likelihood

The estimated effective reproductive ratio, R_E, is comparable to the estimates obtained in Chap. 3:

```
with(as.list(paras),
     sigma/(sigma+mu)*1/(gamma+mu)*beta /N)

## [1] 1.761133
```

8.8 Outbreak-Response Vaccination

Grais et al.'s (2008) objective in fitting a model to the Niamey outbreak data was to evaluate the effectiveness of outbreak-response vaccination (ORV) in reducing the burden of disease during an on-going outbreak. The ORV campaign began on day 161 after the beginning of the epidemic with a goal of vaccinating 50% of all children of ages between 9 months and 5 years. After 10 days, almost 85,000 (57%) of this at-risk group was vaccinated (without knowledge of previous disease or vaccination status). Assuming vaccination was at random with respect to immune status, we can write a modified SEIR function to study the problem. The vaccine cover is a fraction—effectively a *probability*—so we need to translate it to a rate using the relation discussed in Sect. 3.2: $r = -\log(1-p)/D$, where D is the length of the campaign. We define two functions to carry out the efficacy calculations. The `sivmod`-function integrates the SI-model with outbreak-response vaccination and the `retrospec`-function compares predicted epidemic trajectories with and without the ORV.

```
sivmod=function(t,x,parms){
    S=x[1]
    E=x[2]
    I=x[3]
    R=x[4]
    K=x[5]
    with(as.list(parms),{
      Q= ifelse(t<T | t>T+Dt,0,(-log(1-P)/Dt))
      dS= -B*S*I-q*Q*S
      dE= B*S*I-r*E
      dI= r*E - g*I
      dR= g*I+q*Q*S
      dK=r*E
      res=c(dS,dE,dI,dR,dK)
      list(res)
    })
  }

retrospec=function(R, day, vaccine_efficacy,
  target_vaccination,intervention_length,
  mtime, LP=7, IP=7, N=10000){
  steps=1:mtime
        out=matrix(NA,nrow=mtime, ncol=3)
  #starting values
  xstrt=c(S=1-1/N,E=0,I=1/N,R=0,K=0)
  beta= R/IP         #transmission rate
  #Without ORV
  par=c(B=beta, r=1/LP, g = 1/IP, q = vaccine_efficacy,
       P = 0, Dt = 0, T = Inf, R=R)
  outv=as.data.frame(ode(xstrt,steps,sivmod,par))
  fsv=max(outv$K)
  #With ORV
  par=c(B=beta, r=1/LP, g = 1/IP, q = vaccine_efficacy,
              P = target_vaccination, Dt =
              intervention_length, T = day)
  outi=as.data.frame(ode(xstrt,steps,sivmod,par))
  fsi=max(outi$K)
  res=list(redn=fsi/fsv, out=outv, orv=outi, B=par["B"],
     r=par["r"], g=par["g"], q=par["q"], P=par["P"],
     Dt=par["Dt"], T=par["T"], R=R)
  class(res)="retro"
  return(res)
}
```

We will discuss S3-class programming more formally in Sect. 12.1. However, as a preview we define a `plot.retro`-function for objects of class `retro` as the `list` returned by the `retrospec`-function is labeled:

```
plot.retro=function(x){
   plot(x$out[,1], x$out[,"I"], type="l", ylim=c(0,
       max(x$out[,"I"])), xlab='Day', ylab='Prevalence')
   polygon(c(x$T, x$T, x$T+x$Dt,
      x$T+x$Dt), c(-0.1,1,1,-.1), col="gray")
   lines(x$out[,1], x$out[,"I"])
   lines(x$orv[,1], x$orv[,"I"], col="red")
   title(paste("Final size: ", round(100*(x$redn),1),
      "% (R=",x$R,", target=", 100*x$P, "%)", sep=""))
   legend(x="topleft", legend=c("Natural epidemic",
      "With ORV"), col=c("black", "red"), lty=c(1,1))
   text(x=x$T+x$Dt, y=0, pos=4,
      labels=paste(x$intervention_length,
      "ORV from ", x$T))
}
```

If we assume our model is correct and that the vaccine either elicits instantaneous protection or after 2 or 4 weeks (for the antibody response to mature), the ORV is predicted to have reduce the epidemic by 25%, 15%, or 8% respectively:

```
red1=retrospec(R=1.8, 161, vaccine_efficacy=0.85,
     target_vaccination=0.5, intervention_length=10,
     mtime=250, LP=8, IP=5, N=16000)
red2=retrospec(R=1.8, 161+14, vaccine_efficacy=0.85,
     target_vaccination=0.5, intervention_length=10,
     mtime=250, LP=8, IP=5, N=16000)
red3=retrospec(R=1.8, 161+28, vaccine_efficacy=0.85,
     target_vaccination=0.5, intervention_length=10,
     mtime=250, LP=8, IP=5, N=16000)
1-red1$redn

  ## [1] 0.2612989

1-red2$redn

  ## [1] 0.1509867

1-red3$redn

  ## [1] 0.07827277
```

We can plot the `red1`-object to inspect the predicted epidemic curve with and without outbreak-response vaccination (Fig. 8.7). The key insight is that for ORVs to work it needs to be implemented early (Grais et al. 2008).

```
plot(red1)
```

8.9 ShinyApp

The `epimdr`-package contains the `orv.app` with a more detailed sensitivity analyses of outbreak response vaccine scenarios. The app can be launched from R through:

```
require(shiny)
orv.app
```

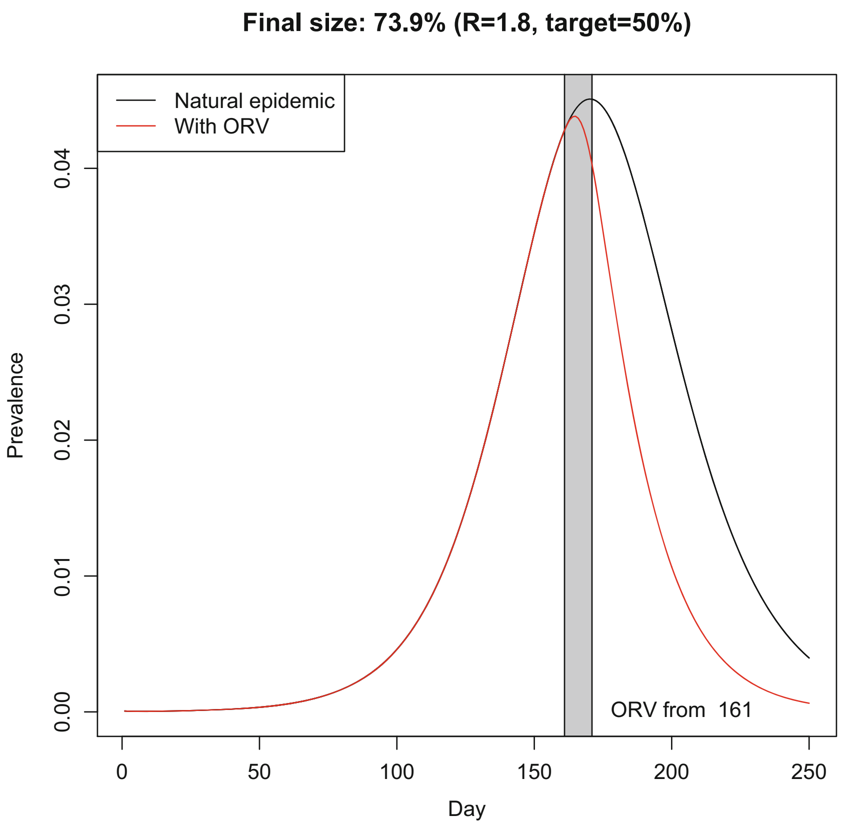

Fig. 8.7 Epidemic curve with and without outbreak-response vaccination starting on day 161 with a target of 50%, vaccine efficacy of 85%, campaign duration of 10 days, and an effective reproductive ratio of 1.8