Chapter 5

Advanced topics

This chapter covers various advanced topics that are useful for some applications. There are a great many ways in which the methods described in previous chapters can be adapted for difficult applied problems, and the choice of methods to cover in this chapter is mostly due to my own interests. The first is approximate Bayesian computation, which is a simulation based method of inference that has become popular for fitting stochastic processes in genetics and infectious diseases, among other applications. The second is data augmentation, which uses the Bayesian secret weapon of extending the conversation to allow the likelihood function to be calculated for some difficult problems where ostensibly it could not be. The third is hidden Markov models, which are often used in time series analysis in finance and other areas. The fourth is Bayesian decision making, which drawns upon ideas from, primarily, econometrics, to deal with parametric uncertainty.

5.1 Approximate Bayesian computation

The use of simulations for inference is actually rather old, dating back at least to Diggle and Gratton (1984, *J Roy Stat Soc ser B* 46:193–227), but was more recently repopularised by an influential paper by Marjoram et al (2003, *Proc Natl Acad Sci USA* 100:15324–8). The basic idea is simple. Imagine you have observed a series of N Bernoulli trials with probability of success p, and the number of successes is X (which, you'll note, is a sufficient statistic for the model). Then, the likelihood is:

$$\mathrm{p}(\tilde{X} = X|p, N) = \binom{N}{X} p^X (1-p)^{N-X},$$

which can be simply calculated on a computer for any value of $p \in (0, 1)$. However, suppose for a minute that you knew nothing of statistics and hadn't the foggiest what the likelihood function was for a binomial sample, but you were very good at simulation. Then you might calculate p($\tilde{X} = X|p, N$) for a specific p and N by simulating a large number of Bernoulli trials and simply counting the proportion of times in which the simulated $\tilde{X}$ equalled the observed X. Code to do so for $p = 0.5$ follows.

```
X=8; N=10
p=0.5
real_likelihood=dbinom(X,N,p)
simulatedX=rbinom(100000,N,p)
sim_likelihood=mean(simulatedX==X)
```

The real likelihood is 0.044, while the simulated likelihood was 0.045, which is sufficiently close for most purposes. We can repeat this process for multiple values of p on a grid search like so:

```
pv=seq(0.01,0.99,0.01)
real_likelihoodv=dbinom(X,N,pv)
sim_likelihoodv=0*pv
for(i in 1:length(pv))sim_likelihoodv[i]=
                          mean(rbinom(100000,N,pv[i])==X)
```

The resulting estimates are plotted in figure 5.1, alongside the results of the same approach applied to the Thai AIDS vaccine trial vaccine arm data. The simulated estimates agree closely with the actual ones. Although this is an extremely artificial example, it indicates that there is a potential to use this same approach for other situations *including those in which the actual likelihood cannot readily be generated.*

In this example, the calculated likelihood is actually an estimate, i.e. a value that can be made arbitrarily precise by increasing the sample size. This is true because the data are discrete and so it is possible to simulate them exactly—i.e. to get a *match*—for any given parameter values. However, for continuous data, this is not possible. Instead, if simulations are to be used for continuous data, we must allow simulations that are 'close' but not equal to the data to count as a match. One obvious way to do this would be to allow a constant threshold of ϵ around each datum and a match is declared if the simulation falls within the threshold.

For instance, consider the leukaemia data of Freireich et al again. In the placebo arm, the times are all recorded to the nearest whole day. A natural ϵ would thus be 12h. So, we could fit a model (exponential in the code that follows, to more readily allow visualisation of the process) by estimating the likelihood over a grid, simulating data for each parameter value, and

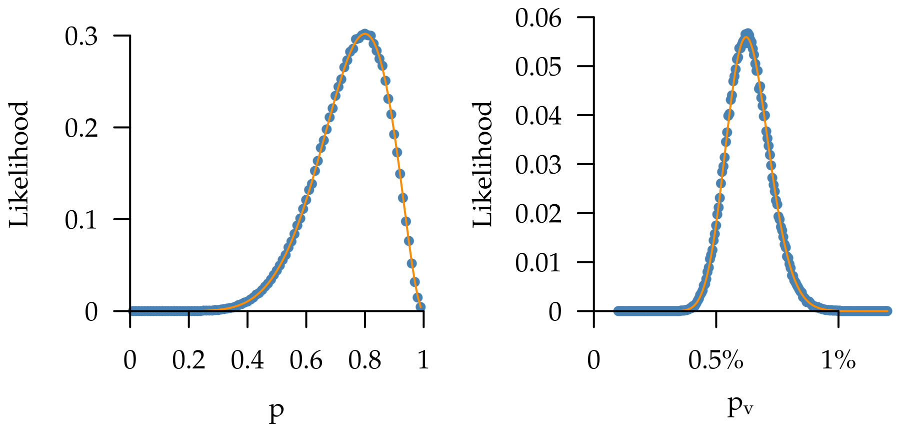

Figure 5.1: **Simple binomial examples**. Left: The actual likelihood for $X = 8$, $N = 10$ is indicated by the blue line, the simulated likelihood by orange circles. Right: same colour coding for the Thai AIDS vaccine trial vaccine arm data.

counting a match to each datum for every simulation that falls within 0.5 units. Since the likelihood is the product of terms from each patient—due to the assumption of independence in the model—we could separately estimate the likelihood contribution from each point, take logarithms, and sum to get an estimate of the log-likelihood, which could be combined readily with a prior and transformed to obtain a summary of the posterior. (In taking the logarith, some numerical hack would be required for parameter values in which for at least one datum there were no hits whatsoever. This could involve replacing the loglikelihood for that value to some small value such as -999999.) Code to do this follows. Note that this code does not run especially quickly.

```
simulator=function(lambda)
{
  tp=c(1,1,2,2,3,4,4,5,5,8,8,8,8,11,11,12,12,15,17,22,23)
  x=round(rexp(100000,lambda))
  fhat=0*tp;for(i in 1:length(tp))fhat[i]=mean(x==tp[i])
  if(min(fhat)==0)return(-999999)
  else return(sum(log(fhat)))
}
lambdav=seq(0.001,0.3,length.out=50)
simlogl=0*lambdav
for(j in 1:length(lambdav))simlogl[j]=simulator(lambdav[j])
```

The resulting log-likelihood and likelihood are plotted in figure 5.2 against the actual values. Although the values seem suitable on the log-likelihood scale, on the linear scale a worrying irregularity near the peak becomes noticable. Increasing the simulation size to one million, helps somewhat, though at considerable computational cost (on my old laptop at home). Inducing correlation between neighbouring points by drawing uniform variates on $[0, 1]$ and using these to seed all draws via the inverse CDF trick also improves matters somewhat, but it is clear that using simulations for inference is not ideal and should be avoided unless there are no alternatives.

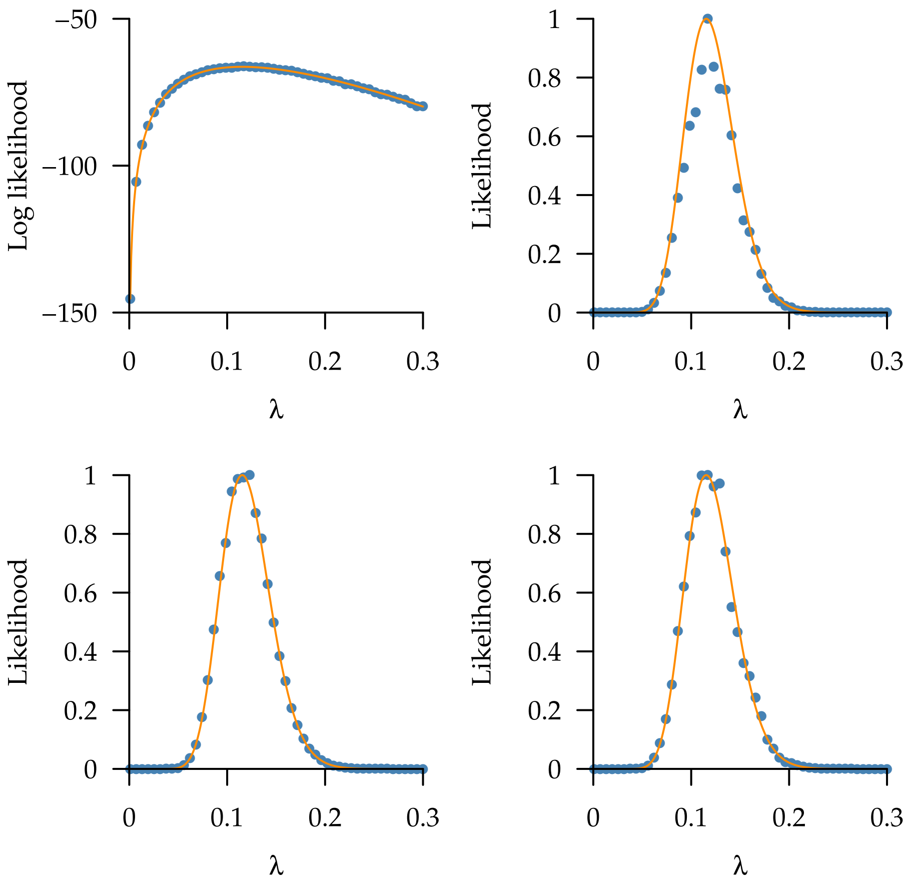

Figure 5.2: **Simulations for the leukaemia chemotherapy trial, placebo arm**. Top left: The actual log-likelihood is indicated by the blue line, the simulated log-likelihood by orange circles. Top right: ditto, but likelihood estimates using 100 000 simulations. Bottom left: ditto, but using 1 000 000 simulations. Bottom right: ditto, but using 100 000 simulations using inverse CDF method and common uniform variates.

On reflexion, the task we have set ourselves—to match an entire data set—is harder than it needed to be, since the model we postulated for the data—the exponential—only requires one summary statistic to calculate the likelihood exactly, the mean of the sample (a sufficient statistic). We therefore could perform simulations of the same sample size and attempt to match the mean of the dataset, within a threshold. Now there is no natural threshold to choose based on rounding in the data, and so we may have to try several to judge the sensitivity of the results to this choice. Code follows, for the leukaemia data once more, again using the inverse CDF trick. The figure (fig. 5.3) shows good agreement between the real likelihood and both approximations.

```
U=array(runif(100000*length(tp)),c(100000,length(tp)))
simulator=function(lambda,u=U,epsilon=1)
{
  meantp=8.667
  x=round(qexp(u,lambda))
  simmeans=rowMeans(x)
  hits=mean(abs(simmeans-meantp)<epsilon)
  if(hits==0)return(-999999)
  else return(log(hits))
}
```

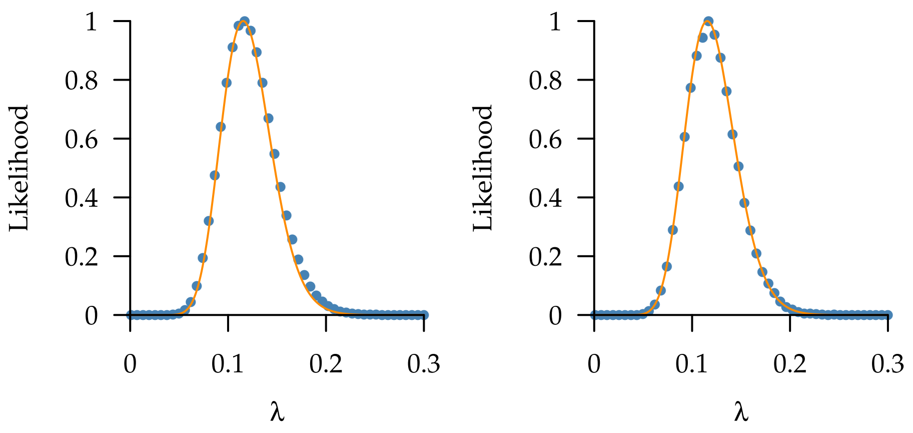

Figure 5.3: **Simulations for the leukaemia chemotherapy trial, placebo arm, using sufficient statistics (the mean)**. Left: the threshold is means within 1 unit of 8.667, the sample average. Right: a lower threshold of 0.1.

5.1.1 The ABC algorithm

The methods we have discussed so far in this section involve simulation-based variants of the grid search. They therefore suffer the same problems that deterministic grid searches have, in particular, the curse of dimensionality. The Marjoram et al paper introduced the idea of embedding simulations within an MCMC algorithm, which allows the approach to be used on higher dimensional data. The algorithms they develop are breathtakingly simple.

- Propose to move from θ_i to $\theta^\star$ using a transition density q, as in regular MCMC;
- Simulate a single dataset, $D^\star$, using the proposed parameter values.
- If $\delta(D^\star, D)$, the distance according to some metric between the simulated data and the actual data, is greater than a threshold ϵ, set $\alpha = 0$. Otherwise, calculate

$$\alpha = \min\left(1, \frac{\mathrm{p}(\theta^\star)q(\theta^\star \to \theta_i)}{\mathrm{p}(\theta_i)q(\theta_i \to \theta^\star)}\right).$$

- Accept $\theta^\star$ and set $\theta_{i+1} = \theta^\star$ with probability α. Otherwise set $\theta_{i+1} = \theta_i$.
- Increment i by one.

In summary, the algorithm is the same as a regular Metropolis–Hastings algorithm, except that the likelihood is not calculated, and instead a dataset is simulated, and if it "matches" the observed data, a modified Metropolis–Hastings step is performed using only the prior and proposal densities.

How to determine a match in this algorithm? If only exact matches are allowed, this algorithm yields an exact sample from the posterior, clearly the ideal scenario. However, as mentioned earlier, the probability of an exact match may be very small, or zero, for many problems. Instead, we might define a distance metric based on sufficient statistics—or what we hope are sufficient statistics—or what we hope are almost sufficient statistics—and a threshold for a match and accept based on that, trying several plausible choices of threshold to assure ourselves of good performance.

R-code to do ABC for the leukaemia chemotherapy trial data follows. The traceplots for this and some other configurations are presented in figure 5.4. It can be seen that more iterations are necessary than for a regular MCMC algorithm, and that the results are not extremely sensitive to the choice of threshold. Kernel density estimates are presented in figure 5.6.

```
xbar=mean(tp)
n=length(tp)
epsilon=0.1
lambda=0.1
ABCiterations=10000
store=rep(0,ABCiterations)
for(iteration in 1:ABCiterations)
{
  if(iteration%%100==0)print(iteration)
  lambdaold=lambda
  lambda=rnorm(1,lambda,0.01)
  reject=FALSE
  if(lambda<0)reject=TRUE
  if(lambda>100)reject=TRUE
  if(!reject)
  {
    simdata=rexp(n,lambda)
    if(abs(xbar-mean(simdata))>epsilon)reject=TRUE
  }
  #if(!reject)alpha=1 for this proposal and prior so can skip
  if(reject)lambda=lambdaold
  store[iteration]=lambda
}
```

5.1.2 Example: stochastic exponential growth model for citrus canker

In tutorial 7 we introduced data on citrus canker in Florida, a disease of citrus trees that leaves lesions on the fruit making them impossible to sell—except as juice—and which is therefore economically devastating to citrus growing areas. In the late 1990s, citrus canker was found to be spreading in Florida again, after having previously been eradicated, and a chief source of inoculum seemed to be trees in the gardens of private dwellings, which could not be controlled as effectively as on commercial orchards. Gottwald and colleagues collected data on several thousand such ‘backyard’ citrus trees, which were later analysed by Cook et al (2008, *J Roy Soc Interface* 5:1203–13). The cumulative number of diseased trees at 30d intervals are presented in table 5.1. Diseased trees were not removed (as this was a control population meant to allow the quantification of disease spread in an undisturbed population).

In the tutorial, I proposed using *deterministic* exponential or logistic

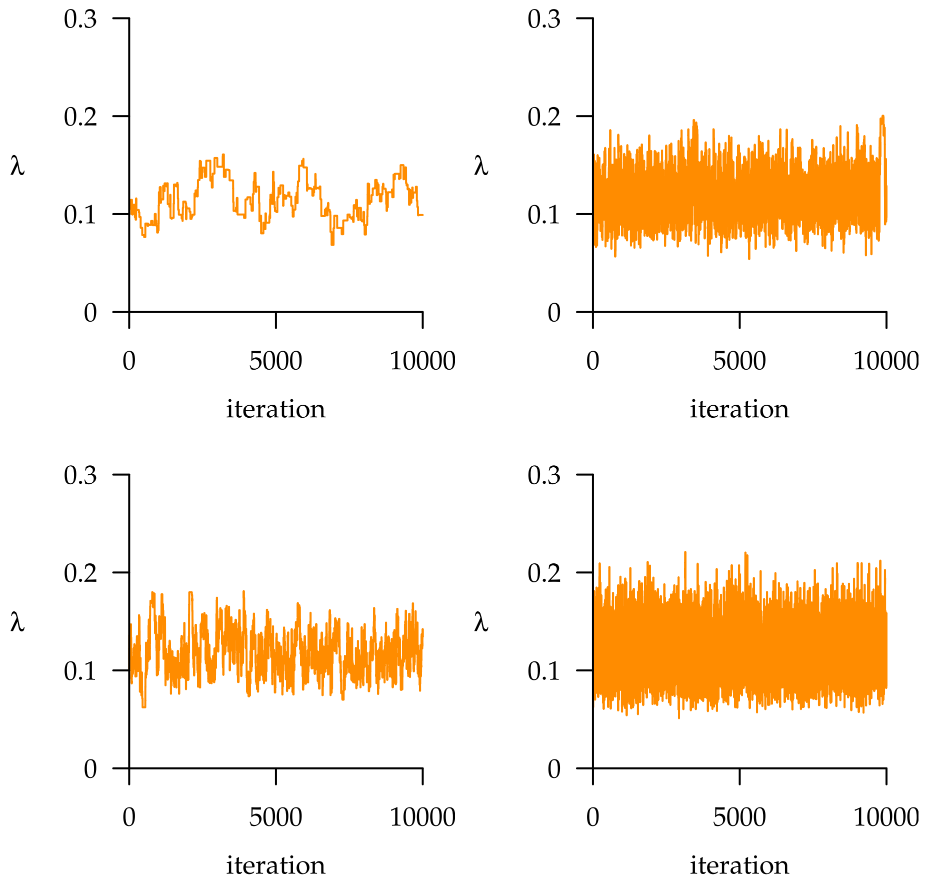

Figure 5.4: **ABC for the leukaemia chemotherapy trial, placebo arm, using sufficient statistics (the mean)**. Top left: the threshold is means within 0.1 unit of 8.667, the sample average. Top right: same threshold, but thinning of 100. Bottom left: a larger threshold of 1. Bottom right: larger threshold and thinning of 100.

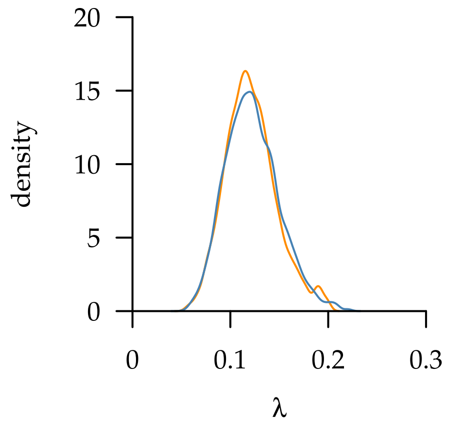

Figure 5.5: **ABC for the leukaemia chemotherapy trial, placebo arm, using sufficient statistics (the mean)**. Density plots for narrow (0.1, orange) and broad (1, blue) thresholds.

growth models for the data, in which case an appropriate choice of error—representing biological variability, not observation error—must be selected. Note that if these really are variability and not measurement errors, then the model should be restarted from each observation to each next observation, rather than simply being run once from an initial condition at time 0. A more satisfying alternative is to fit a *stochastic* model, in which the variability is implicit to the model description. Here we shall focus on a stochastic SI model, which is one way to formulate a logistic growth for an infectious disease.

If there are currently $n(t)$ trees infected out of $N = 6056$, then the SI model says that:

$$\mathrm{p}\left\{n(t+h) = n(t) + 1 | n(t), \beta\right\} = \beta n(t)\{N - n(t)\}h + \mathrm{o}(h),$$

that is, $\beta n(t)\{N - n(t)\}$ is the instantaneous rate of infection. Since this rate does not vary between infection times, we can simulate "data" from this model for a given $n(0)$ and β by generating a series of Exp(1) variates and scaling them by a simple function of infections so far. It is convenient to rescale the parameter by the population size $b = \beta/N$.

Here we have a vector of the number of cases over time, and no sufficient statistic, and no obvious summary statistics that would encapsulate most of the information in the data. We shall therefore try to match the counts of diseased trees for all times. Let us define a match to be when $S_t \in (D_t/\delta, D_t\delta)$ for all times $t \in \{30, 60, 360\}$, where S_t is the simulated and D_t the observed

Table 5.1: Data on citrus canker in Miami, Florida, from Cook et al (2008, *J Roy Soc Interface* 5:1203–13). The total number of hosts in this geographic area of Miami was 6056 and there were no diseased trees at the start of data collection.

Day	30	60	90	120	150	180
Trees	4	14	23	40	71	122
Day	210	240	270	300	330	360
Trees	264	480	535	603	872	1124

number of cases, and δ characterises the threshold for a match. As a first attempt, let us use $\delta = 2$ (i.e. the rather low bar of getting simulations to lie between 50% and 200% of the data). The code follows.

```
simulator=function(pars,d=data)
{
  no_s   = (d$N-pars$n0):1
  no_i   = pars$n0:(d$N-1)
  Z      = rexp(no_s[1])
  rates  = (pars$b/d$N)*no_i*no_s
  deltat = Z/rates
  t   = c(rep(0,pars$n0),cumsum(deltat))
  sim = list(t=d$t,n=rep(0,length(d$t)),N=d$N)
  for(i in 1:length(sim$t))sim$n[i] = sum(t<sim$t[i])
  sim
}
```

This creates vectors of the number of susceptible and infected trees just before each infection event, calculates the infection rates using these, and applies them to some standard exponential variates to generate inter-event times. The event times are then obtained by cumulating these. The ABC code using this follows.

```
data=list(t=seq(30,360,30), N=6056,
          n=c(4,14,23,40,71,122,264,480,535,603,872,1124))
current=data.frame(b=0.018,n0=3)
epsilon=2
ABCiterations=10000
thinning=10
store=data.frame(b=rep(0,ABCiterations),n0=rep(0,ABCiterations))
```

```
for(iteration in 1:ABCiterations)
{
  if(iteration%%100==0)print(iteration)
  for(thin in 1:thinning)
  {
    old=current
    current$b=rnorm(1,current$b,0.002)
    current$n0=round(rnorm(1,current$n0,0.5))
    reject=FALSE
    if(current$b<0)reject=TRUE
    if(current$n0<1)reject=TRUE
    if(current$n0>data$n[1])reject=TRUE
    if(!reject)
    {
      simdata=simulator(current)
      cond1=simdata$n>(data$n/epsilon)
      cond2=simdata$n<(data$n*epsilon)
      conds=cond1*cond2
      if(sum(conds==0)>0)reject=TRUE
    }
    #if(!reject)alpha=1 so can skip
    if(reject)current=old
  }
  store[iteration,]=current
}
```

Traceplots (figure 5.6) indicate good mixing and a narrow posterior that may be good enough for practical purposes (such as forecasting). However, we would ideally like a smaller threshold. Rerunning the model with a threshold of 1.75 yields, again, good samples. However, with a threshold of 1.5, mixing becomes very poor and we would therefore not be satisfied using those output. The posterior predictive distribution of the number of diseased trees over time (figure 5.7) indicates that the reason for the lack of fit may be that the model does not reflect the shape of the data very well—with less disease later on than expected, and more randomness in the data than predicted by the model. To remedy this would require a more sophisticated, more highly parameterised model that relaxes the assumptions about the infection rate being proportional to the number of infected and susceptible trees, for instance, by introducing a non-linear function of the number infected, or by making β time varying, either deterministically or randomly, to account for climatic factors or other environmental forcing.

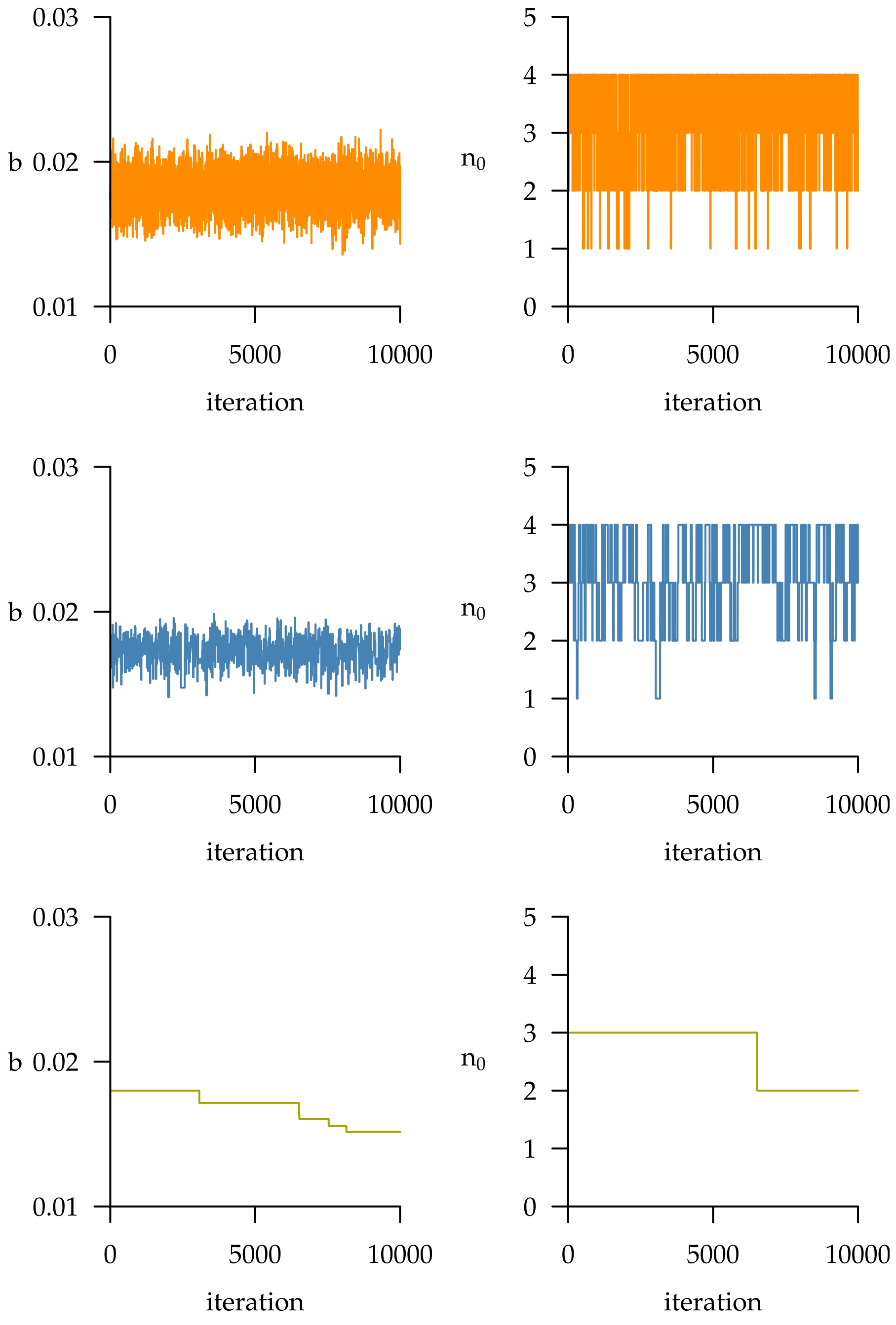

Figure 5.6: **ABC for the canker data**. Top: $\epsilon = 2$. Middle: $\epsilon = 1.75$. Bottom: $\epsilon = 1.5$.

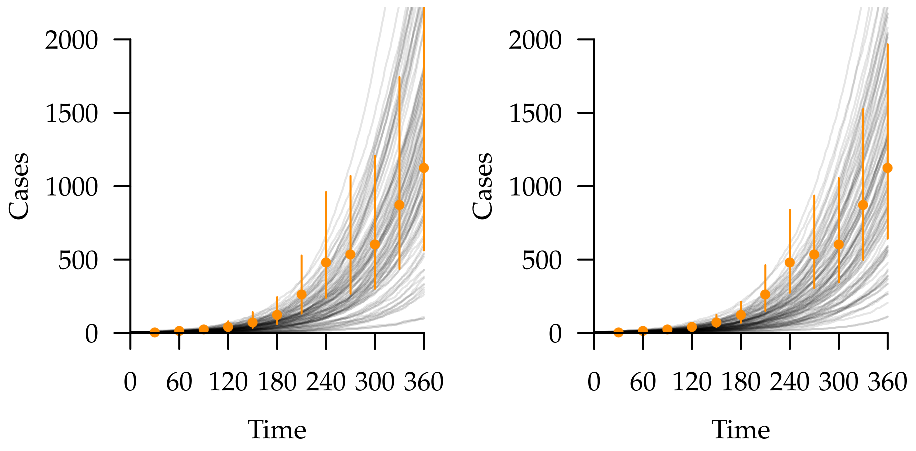

Figure 5.7: **Posterior predictive distribution of the number of diseased trees using ABC**. Left: $\epsilon = 2$. Right: $\epsilon = 1.75$. Simulations are displayed in grey, data in orange (points, with lines representing the threshold for acceptance).