5.2 Data augmentation

One of the great strengths of Bayesian methods is the way all unknowns are represented as random variables. The cancer status of an asymptomatic woman attending screening is unknown, and hence random. The mean income of all expatriates in Singapore is unknown, and hence random. The secret weapon of Bayesian statistics is that if there is something that you would like to know, and if you knew it, it would make the likelihood function for the data you *do* know easy to calculate, you can introduce it into the analysis by treating it as a random variable. This trick is also called *extending the conversation* to include the additional variable. In notation, if you wish to calculate $\mathrm{p}(D|\theta)$ and cannot, but you can calculate both $\mathrm{p}(D|\theta, A)$ and $\mathrm{p}(A|\theta)$, where A is the additional "data" you would need to calculate the likelihood (possibly high dimensional), then you can recast the problem as finding

$$\mathrm{p}(D|\theta) = \int_{\mathcal{A}} \mathrm{p}(D|\theta, A)\mathrm{p}(A|\theta)\,\mathrm{d}A.$$

This turns the problem of calculating an intractable likelihood into solving an integral. Well, this is something you are going to be doing anyway if you are taking a Bayesian approach, because MCMC and similar routines are, essentially, tools to solve difficult integrals, although the integration problem is dressed up as a sampling one.

Data augmention involves treating the unknown "data" as additional parameters to be estimated, i.e. involves switching from sampling θ to sampling (θ, A). You can then perform MCMC as usual, but if the interest is just on θ, you can ignore the samples from A—*integrating them out.*

There are several technical problems you may encounter when doing data augmentation. If the augmented data A are high dimensional, then you may have to propose values to a considerably larger number of parameters than you would had you known the information. This may increase the computational burden. It also makes it harder to assess convergence, since there may be too many additional, unwanted parameters to justify storing in memory. Another issue is that the augmented data are often correlated quite closely with each other and/or one or more parameters, so it can be difficult to design good proposal distributions.

5.2.1 Citrus canker revisited

Recall in our analysis of the citrus canker data from Miami that we were unable to get good mixing of the ABC routine unless we took an overly broad matching threshold. We are unable to do straightforward MCMC for this problem because:

- the disease status of two trees in the population are not independent—knowing one is infected makes it more likely the other is infected, as the former may have infected the latter;
- the infection time of each tree is interval censored—we know the 30d window in which it occurred, ignoring the delay between infection and onset, or assuming it to be unknown but constant.

The combination of the two of these makes it impossible to derive the likelihood directly[1]. But, if we happened to know the exact times of infection, we could calculate the likelihood exactly. If the time of infection of the ith tree is t_i, the likelihood under this hypothetical scenario would be

$$\begin{aligned}
& \mathrm{p}(t_1, t_2, t_3, \ldots, t_{1124}|b) \\
= \; & \mathrm{p}(t_2, t_3, \ldots, t_{1124}|t_1, b)\mathrm{p}(t_1|b) \\
= \; & \mathrm{p}(t_3, \ldots, t_{1124}|t_1, b)\mathrm{p}(t_2|t_1, b)\mathrm{p}(t_1|b) \\
= \; & \prod_{i=1}^{1124} \mathrm{p}(t_i|t_{i-1}, \ldots, t_1, b) \\
= \; & \prod_{i=1}^{1124} \mathrm{p}(t_i|t_{i-1}, b)
\end{aligned}$$

where, without loss of generality, we have ordered the individual indices by the infection order (so $t_i < t_j$ for $i < j$), and we set $t_0 = 0$ for the first new infection. (Actually, an additional term is required to account for no further infections between t_{1124} and 360. This takes the form of a survival function; see R code that follows for details.) Note, I will change the model slightly by starting with 0 infections and assuming the first infection has distribution $U(0, 30)$ as an "act of god," with subsequent infections following the stochastic SI model. The rationale for that is to make the approach easier to follow—it is not necessary for the approach used below.

Each term in this product is fairly easy to calculate. If the infection rate at time t is $\lambda(t)$, the likelihood contribution from $i(> 1)$ is:

$$\mathrm{p}(t_i|t_{i-1}, b) = \lambda(t_i) \exp\left\{-\int_{t_{i-1}}^{t_i} \lambda(t)\,\mathrm{d}t\right\}$$

[1]This is not strictly true: the likelihood *could* be derived if you could work out the matrix exponential of a matrix with dimensions roughly 1000×1000. As far as I'm aware, current state of the art software to calculate matrix exponentials is limited to matrices 1% of that size.

(take a course in survival analysis or stochastic processes to see why). Since the infection rate does not change in between infection events, according to this model, the integral becomes the area of a rectangle with height $b(i-1)(N-i+1)$ for the ith infection and length $t_i - t_{i-1}$ and so the likelihood contribution simplifies to

$$\mathrm{p}(t_i|t_{i-1}, b) = b(i-1)(N-i+1)\exp\left\{-b(i-1)(N-i+1)(t_i - t_{i-1})\right\}.$$

So, if we take a continuous uniform prior on $(0, 100)$ for b, the posterior given the actual data D is:

$$\begin{aligned}
\mathrm{p}(b|D) &\propto \mathrm{p}(D|b)\mathrm{p}(b) \\
&= \int \mathcal{T}\mathrm{p}(D|t, b)\mathrm{p}(t|b)\,\mathrm{d}t \\
&= \int \mathcal{T}\mathrm{p}(t|b)\,\mathrm{d}t
\end{aligned}$$

where t is the unobserved vector of infection times and $\mathcal{T}$ the space of times that are consistent with the data. We can then run an MCMC sampler on the times and parameter b, as long as the initial set of times fall in $\mathcal{T}$ and we never propose values outside $\mathcal{T}$. This can be done in R as follows.

The following function calculates the log-posterior for a parameter value and combination of infection times:

```
logposterior=function(pars,d=data)
{
  ninf   = length(pars$t)
  no_s   = d$N:1
  no_i   = 0:(d$N-1)
  rates  = (pars$b/d$N)*no_i*no_s
  deltat = pars$t[-1]-pars$t[-ninf]
  lastt  = 360-pars$t[ninf]
  loglikelihood = sum(dexp(deltat,rates[2:ninf],log=TRUE))+
                    pexp(lastt,rates[ninf+1])
  logprior = dunif(pars$b,0,100,log=TRUE)
  pars$logposterior = loglikelihood+logprior
  pars
}
```

Infection rates are calculated for all possible infections, but only (i) those rates corresponding to actual infections (via the `dexp` term) or (ii) the rate for the first infection that did not occur (the `pexp` term) enter the likelihood.

The Metropolis-Hastings step is standard, and the only thing to note is how the consistency of the event times with the data is checked (lines 1 and 2).

```
mh=function(current,old,data)
{
  reject=FALSE
  if(current$b<0)reject=TRUE
  if(current$b>100)reject=TRUE
  if(min(current$t-data$tmin)<0)reject=TRUE #1
  if(min(data$tmax-current$t)<0)reject=TRUE #2
  if(!reject)
  {
    current=logposterior(current)
    logaccprob=current$logposterior-old$logposterior
    lu=-rexp(1)
    if(lu>logaccprob)reject=TRUE
  }
  if(reject)current=old
  current
}
```

The data are set up as follows:

```
data      = list(t=seq(30,360,30),n=c(4,14,23,40,71,122,264,480,
                                      535,603,872,1124),N=6056)
data$tmin = rep(0,data$n[1])
for(k in 2:12)
  data$tmin=c(data$tmin,rep(data$t[k-1],data$n[k]-data$n[k-1]))
data$tmax = data$tmin+30
```

The sampler itself is also rather standard. Note that to ameliorate the computational burden, I prefer not propose changes to each infection time at each iteration, and when I propose a change, I propose it uniformly between the last and next infection times. Traceplots are provided in figure 5.8.

```
current=list(b=0.004,t=sort(runif(length(data$tmin),data$tmin,data$tmax)))
current=logposterior(current)
MCMCiterations=10000
store=data.frame(b=rep(0,MCMCiterations),t100=rep(0,MCMCiterations))
for(iteration in 1:MCMCiterations)
{
  if(iteration%%100==0)print(iteration)
  old=current
  current$b=rnorm(1,current$b,0.0005)
  current=mh(current,old,data)
  for(k in sample(1:length(data$tmin),100))
  {
    old=current
    mint=c(0,current$t[-length(current$t)])
    maxt=c(current$t[-1],360)
    current$t[k]=runif(1,mint[k],maxt[k])
    current=mh(current,old,data)
  }
  store[iteration,1]=current$b
  store[iteration,2]=current$t[100]
}
```

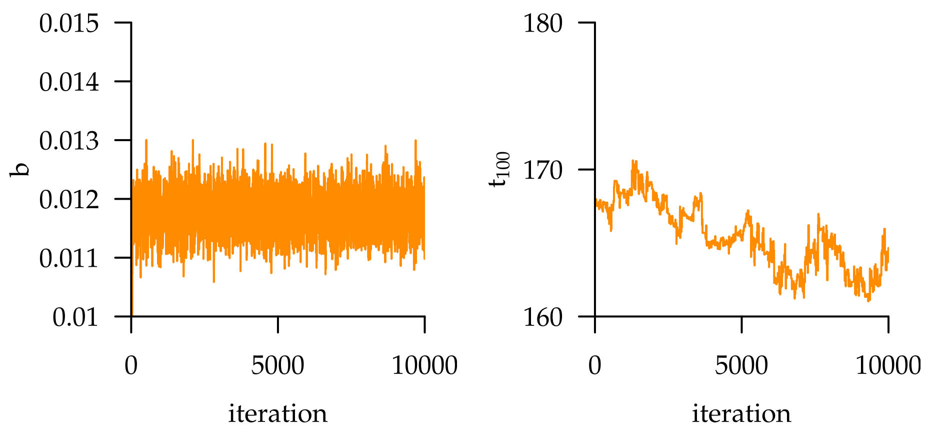

Figure 5.8: **Traceplots of infection parameter (left) and an arbitrary infection time (right) using data augmentation**. Proposals used are uniform for infection times between last and next infection times.

Although the traceplot for the unknown of interest, b, appears to have converged, the mixing of the one time point we choose to monitor is poor. It is not clear, then, whether the sample of b really represents the marginal posterior. We might therefore try to improve the proposal distribution for the infection time. Instead of a proposal to change time t_i that is $U(t_{i-1}, t_{i+1})$, we might propose from $N(t_i, \sigma)$ and then relabel the times to ensure $t_i < t_j$ for all $i < j$. With the following change

```
for(k in sample(1:length(data$tmin),100))
{
  old=current
  current$t[k]=rnorm(1,current$t[k],5)
  mint=c(0,current$t[-length(current$t)])
  maxt=c(current$t[-1],360)
  current$t=sort(current$t)
  current=mh(current,old,data)
}
```

mixing is much improved (see figure 5.9) and with more iterations we would be satisfied. This code could then act as the basis for an improved model of how infection rates change with time.

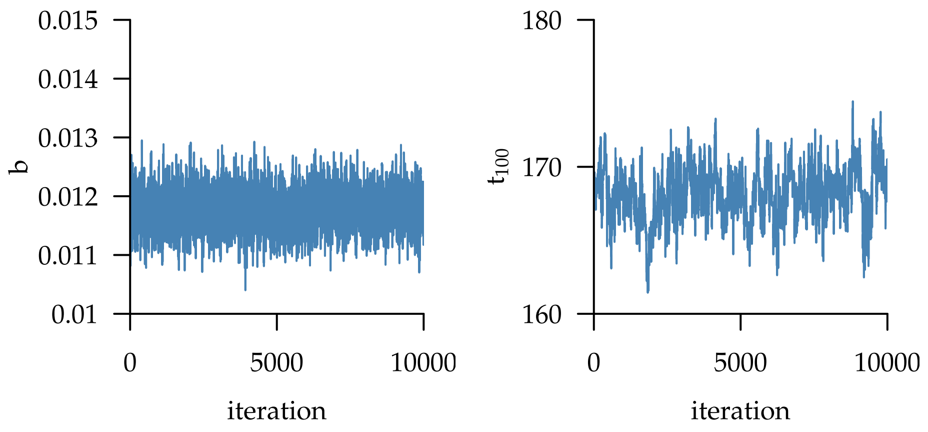

Figure 5.9: **Traceplots of infection parameter (left) and an arbitrary infection time (right) using data augmentation**. Proposals for infection times are normal with re-sorting to ensure they are properly ordered.

Data augmentation is particularly valuable when dealing with discrete space, continuous time stochastic processes, as the likelihood often becomes tractable once you condition on unobserved event times and natures. However, for large numbers of events, simulation based methods may be more feasible. Data augmentation is also valuable for discrete time stochastic processes, as we shall see in the next section.