Bayesian Inference II

Course lecturer: Loukia Meligkotsidou

2014

Contents

Chapter 1

Introduction

1.1 Scope of course

In this course, we shall aim at understanding computer techniques which require innovative algorithms to apply Bayesian inference. An introduction to the basic tools of stochastic simulation and Monte Carlo integration methods will be given, before moving on to the advanced numerical techniques used for Bayesian inference, referred to as Markov chain Monte Carlo (MCMC) algorithms. These algorithms, largely developed during the nineties, have revolutionized the practice of statistics by allowing us to tackle problems of real complexity that were impossible (or extremely difficult) to handle before. This means that we can now be much more realistic in our modelling, which is extremely important when dealing with 'real-world' applications. These methods are particularly well suited to computing the posterior distribution (via simulation) in Bayesian inference, and are to a large extent responsible for the great upsurge in popularity of Bayesian statistics during the last decades. Thus, although we shall make some remarks about the use of MCMC algorithms in other contexts, this course will be focussed on the application of MCMC methods to Bayesian inference.

There are two roles of this type of course. The first is to gain some understanding and knowledge of the techniques and tools which are available. The second is that many of the techniques (if not all of them) are themselves clever applications or interpretations of the interplay between probability and statistics. So, understanding the principles behind the different algorithms can often lead to a better understanding of inference, probability and statistics in general. In short, the techniques are not just a means to an end. They have their own intrinsic value as statistical exercises.

This is not a course on computing itself. We won't get into the details of programming. However, we will need programming in R to handle the examples of the course and implement the algorithms.

1.2 Computers as inference machines

It is something of cliché to point out that computers have revolutionized all aspects of statistics. In the context of inference there have really been two substantial impacts: the first has been the freedom to make inferences without the catalogue of arbitrary (and often blatantly inappropriate) assumptions which standard techniques necessitate in order to obtain analytic solutions — Normality, linearity, independence etc. The second is the ability to apply standard type

models to situations of greater data complexity — missing data, censored data, latent variable structures.

Chapter 2

Simulation

2.1 Introduction

In this chapter we look at different techniques for simulating values from distributions. We will also describe the Monte Carlo method and its use in computing integrals.

2.2 Motivation

Many applications of simulation are based on the idea of using samples from a distribution in order to approximate characteristics of this distribution. As a trivial example, suppose that we are interested in the mean of a real random variable X with distribution function F and probability density function f. This expectation is given by

$$\int_{-\infty}^{\infty} xf(x)dx.$$

However, it might be difficult or impossible to perform the above integration. Suppose that a random sample $x_1, \ldots, x_n$ was available where each x_i is a realisation of X_i, and the $X_1, \ldots, X_n$ are i.i.d (independent and identically distributed) random variables with distribution F. Then we could approximate the mean of X by the observed sample mean

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

This result, which is routinely used in statistical estimation, will be shown to be related with the so-called Monte Carlo integration (Section 2.7).

As another simple example, suppose that we are interested in the median of the distribution of X. Notice now that unless F is a symmetric distribution, the median cannot be expressed as an expectation of some function of X. Instead, the median is the 0.5 quantile of F, where the αth quantile of F is $F^{-1}(\alpha)$, where F^{-1} denotes the inverse of F and $0 \leq \alpha \leq 1$. (You have encountered such quantiles when constructing confidence intervals.) As before, analytical calculation of the median (or of any other quantile) might be infeasible. Nevertheless, we can use the sample median to approximate the median of F. This technique is again related to bootstrap methods.

The validity of using samples to estimate characteristics of F is justified by some basic results of probability theory, summarised in the next section.

2.3 Some limit theorems

The *Law of Large Numbers* is the main result which validates using sample averages to estimate population means. Before stating (a very simple version of) the result, let $X_1, \ldots, X_n$ be a sequence of random variables, and $\theta_n = \theta_n(X_1, \ldots, X_n)$ be a real function of them (e.g $\theta_n = n^{-1}\sum_{i=1}^n X_i$). We say that θ_n converges in mean square sense to a fixed value θ if

$$E(\theta_n - \theta)^2 \to 0, \text{ as } n \to \infty.$$

Theorem 2.3.1. (A Law of Large Numbers). *Let $X_1, \ldots, X_n$ be a sequence of independent random variables with common means $E(X_i) = \theta$ and variances $Var(X_i) = \sigma^2 < \infty$. If $\theta_n = n^{-1}\sum_{i=1}^n X_i$ then*

$$E(\theta_n - \theta)^2 = \sigma^2/n \to 0, \text{ as } n \to \infty.$$

Although this theorem suggests that sample averages will converge to the population mean (with rate $1/n$), the following theorem gives more information about the (asymptotic) error of θ_n in estimating θ.

Theorem 2.3.2. (A Law of Large Numbers). *Let $X_1, \ldots, X_n$ be a sequence of independent random variables with common means $E(X_i) = \theta$ and variances $Var(X_i) = \sigma^2$. If $\theta_n = n^{-1}\sum_{i=1}^n X_i$ then*

$$\frac{\sqrt{n}}{\sigma}(\theta_n - \theta)$$

is approximately distributed as a $N(0,1)$ random variable, as $n \to \infty$.

Therefore, we wish to have a small σ in order to obtain accurate estimates of θ by θ_n. (This issue is related to the variance reduction techniques of Section 2.7.2.)

Similar results show that sample quantiles converge to population quantiles as the sample size increases.

2.4 Issues in simulation

The results of the previous section show that it is reasonable to use samples from a distribution in order to approximate certain characteristics of that distribution. Therefore, of key importance becomes how to generate samples from a specified distribution. In particular, this chapter using basic results from probability theory, will give some answers to the following two questions:

1. How to do it; and
2. How to do it efficiently.

To some extent, just doing it is the priority, since many applications are sufficiently fast for even inefficient routines to be acceptably quick. On the other hand, efficient design of simulation can add insight into the statistical model itself, in addition to CPU savings.

2.5 Raw ingredients

The raw material for any simulation exercise is uniformly distributed random numbers: $U \sim U[0,1]$. Transformation or other types of manipulation can then be applied to build simulations

of more complex univariate or multivariate distributions, as we show in the rest of this chapter. But, how can uniform random numbers be generated?

The standard approach is to use a **deterministic** algorithm which however produces a sequence of numbers which do not exhibit any obvious pattern: they look random in the sense that they can pass all the statistical tests of randomness despite their deterministic derivation. The most common technique is to use a *congruential generator*, which generates a sequence of integers via the algorithm

$$X_i = aX_{i-1}(modM) \tag{2.1}$$

for suitable choices of a and M. Dividing this sequence by M gives a sequence $U_i = X_i/M$ which can be regarded as realizations from the Uniform $U[0,1]$ distribution. Note that any statistical package has its random number generator checked pretty thoroughly. The point worth remembering though is that computer generated random numbers aren't random at all, but that (hopefully) they look random enough for that not to matter.

Nowadays, however, true random numbers can actually be obtained. Rather controversially, pseudo-random numbers are much more preferable than true random numbers for several reasons, for example for code-checking or variance reduction techniques. From this perspective, it is actually an advantage, rather than a limitation, to be able to produce random-looking pseudo-random numbers.

In subsequent sections we will assume that we can generate a sequence of numbers $U_1, U_2, \ldots, U_n$ which may be regarded as n independent realizations from the $U[0,1]$ distribution.

2.6 Simulating from specified distributions

In this section we look at ways of simulating data from a specified **univariate** distribution F, on the basis of a simulated sample $U_1, U_2, \ldots, U_n$ from the distribution $U[0,1]$. Some of the techniques of this section can be generalized to higher dimensions, although they become less efficient with increasing dimensions. The next chapters will introduce much more powerful techniques for simulation from multivariate distributions.

Notice however, that in principle we could simulate from a **multivariate** distribution using only univariate simulations. Assume that $(Y_1, \ldots, Y_m)$ is an m-dimensional vector with joint distribution function $F_{(Y_1,\ldots,Y_m)}$. This joint distribution has a well-known representation as a product of conditional distributions:

$$F_{(Y_1,\ldots,Y_m)} = F_{Y_1} F_{Y_2|Y_1} F_{Y_3|(Y_1,Y_2)} \cdots F_{Y_m|(Y_1,\ldots,Y_{m-1})}$$

where F_{Y_1} is the marginal distribution of Y_1, $F_{Y_2|Y_1}$ is the conditional distribution of Y_2 given Y_1, and so on. Therefore, we can simulate a realization of $(Y_1, \ldots, Y_m)$ by first simulating $Y_1 \sim F_{Y_1}$, then conditionally on that value we simulate $Y_2 \sim F_{Y_2|Y_1}$, etc, and finally $Y_m \sim F_{Y_m|Y_1,\ldots,Y_{m-1}}$. This sometimes is called the *composition* method. The problem with this approach is that it is difficult to derive the conditional distributions from the joint. Nevertheless, for small m, for example $m = 2$, it might be feasible. As a trivial special case, this method can be used to simulate a vector of independent random variables.

We now describe some of the established techniques for univariate simulation.

2.6.1 Inversion

This is the simplest of all procedures, and is nothing more than a straightforward application of the probability integral transform: if $X \sim F$, then $F(X) \sim U[0,1]$, so by inversion if $U \sim U[0,1]$,

then $F^{-1}(U) \sim F$. Thus, defining $x_i = F^{-1}(u_i)$, generates a sequence of independent realizations from F. Figure 2.1 illustrates how this works.

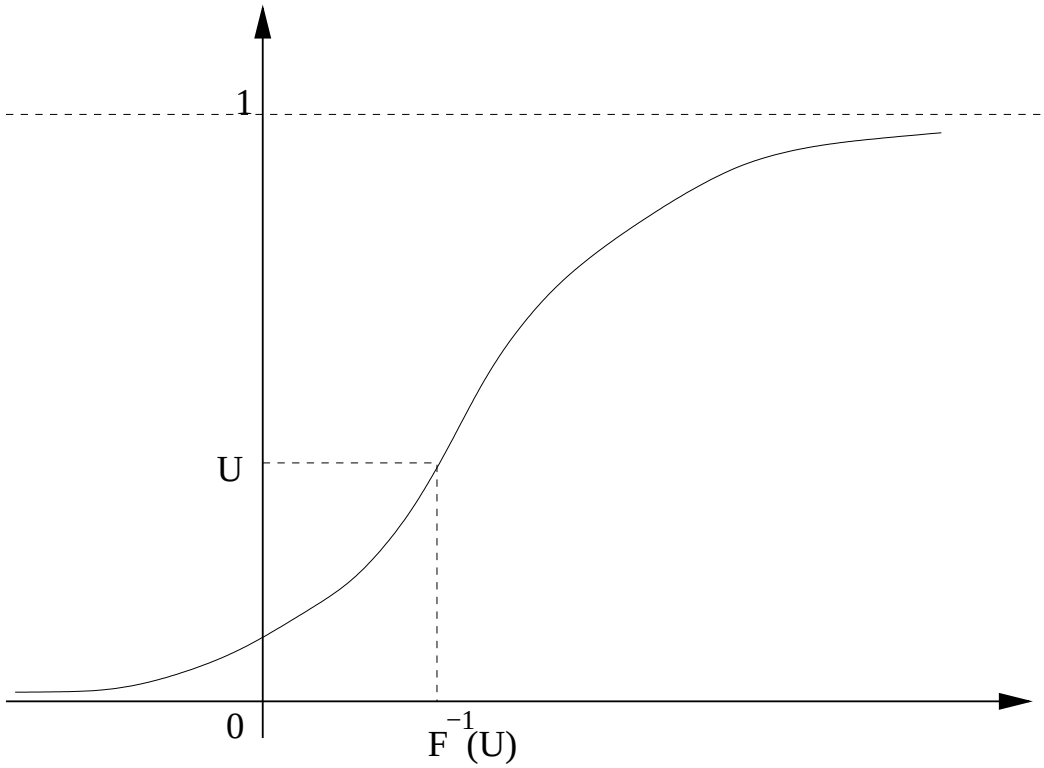


Figure 2.1: Simulation by inversion

This procedure is easily implemented in R with the following code:

```
dsim=function(n, inv.df) {
        u = runif(n)
        inv.df(u)
}
```

where `inv.df` is the user–supplied inverse distribution function F^{-1}. For example, to simulate from the exponential distribution we have $F(x) = 1 - \exp(-\lambda x)$, so $F^{-1}(u) = -\lambda^{-1}\log(1-u)$. Thus defining

```
inv.df=function(x,lam=1) -(log(1-x))/lam
```

we can simulate from the exponential distribution (unit exponential as default). Figure 2.2 shows a histogram of 1000 standard exponential variates simulated with this routine.

This procedure works equally well for discrete distributions, provided we interpret the inverse distribution function as

$$F^{-1}(u) = \min\{x|F(x) \geq u\} \tag{2.2}$$

The procedure then simply amounts to searching through a table of the distribution function. For example, the distribution function of the Poisson (2) distribution is

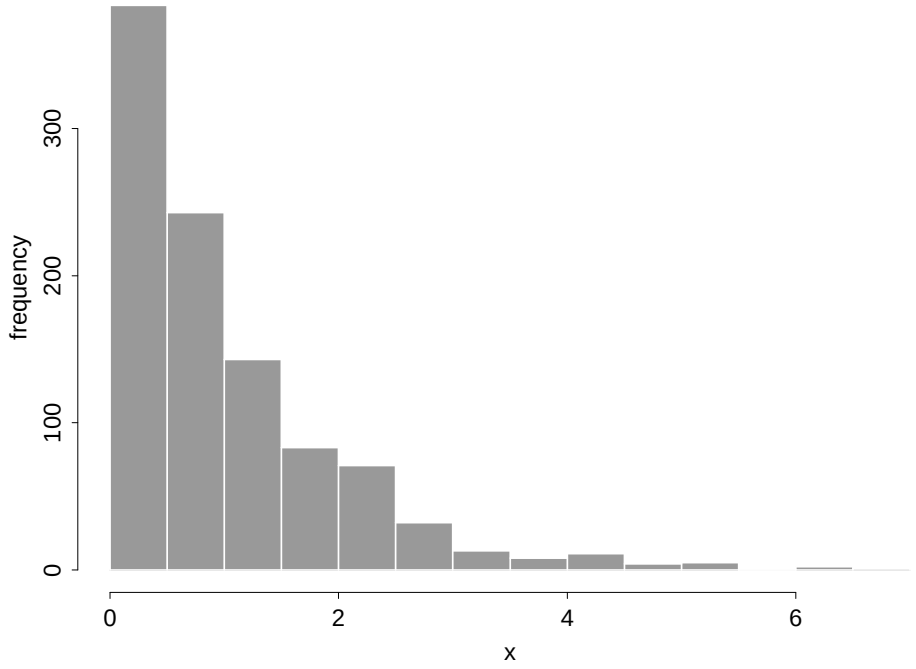


Figure 2.2: Histogram of 1000 simulated unit exponential variates

```
 x     F(x)
------------
 0  0.1353353
 1  0.4060058
 2  0.6766764
 3  0.8571235
 4  0.9473470
 5  0.9834364
 6  0.9954662
 7  0.9989033
 8  0.9997626
 9  0.9999535
10  0.9999917
```

so, we generate a sequence of standard uniforms $u_1, u_2, \ldots, u_n$ and for each u_i obtain a Poisson (2) variate x where $F(x-1) < u_i \leq F(x)$. So, for example, if $u_1 = 0.7352$ then $x_1 = 3$.

The limitation on the efficiency of this procedure is due to the necessity of searching through the table, and there are various schemes to optimize this aspect.

Returning to the continuous case, it may seem that the inversion method is sufficiently universal to be the only method required. In fact, there are many situations in which the inversion method is either (or both) complicated to program or excessively inefficient to run. The inversion method is only really useful if the inverse distribution function is easy to program and compute. This is not the case, for example, with the Normal distribution function for which the inverse distribution function, Φ^{-1}, is not available analytically and slow to evaluate numerically. To deal with such cases, we turn to a variety of alternative schemes.

The main limitation of the method however, is that it cannot be extended to multivariate simulation: it is by construction a method for simulating from univariate distributions.

2.6.2 Transformation of variables

The main idea is simple and is used extensively in simulation: suppose that we want to simulate a random variable $X \sim F$, and we know that it can be written as $X = t(Y)$, where $Y \sim G$ is another random variable and $t(\cdot)$ is some function. Then, defining $x_i = t(y_i)$ where y_i has been generated from G, generates a sequence of independent realizations from F. Notice that the inversion method is a special case of this technique.

As an example, suppose that $X \sim Exp(\theta)$, that is X has the exponential distribution with mean $1/\theta > 0$, then $X = \theta^{-1}Y$, where $Y \sim Exp(1)$. Therefore, we can simulate X by first simulating a standard exponential random variable, for example using the inversion method, and then dividing the drawn value by θ.

The function $t(\cdot)$ can be a many-to-one function. For example, suppose that $X \sim Gamma(n, \theta)$, that is X has the gamma distribution with mean n/θ and variance n/θ^2, where n is some integer. It is known that $X = t(Y_1, \ldots, Y_n) = \sum_{i=1}^{n} Y_i$, where the Y_is are independent and identically distributed $Exp(\theta)$ random variables. Thus, X can be simulated by simulating n independent Y_is and then summing them up.

2.6.3 Rejection sampling

The idea in rejection sampling is to simulate from one distribution which is easy to sample from, but then to only accept that simulated value with some probability p. By choosing p correctly,

we can ensure that the sequence of accepted simulated values are from the desired distribution. Rejection sampling has a very intuitive geometrical interpretation, illustrated in Figure 2.3. It is

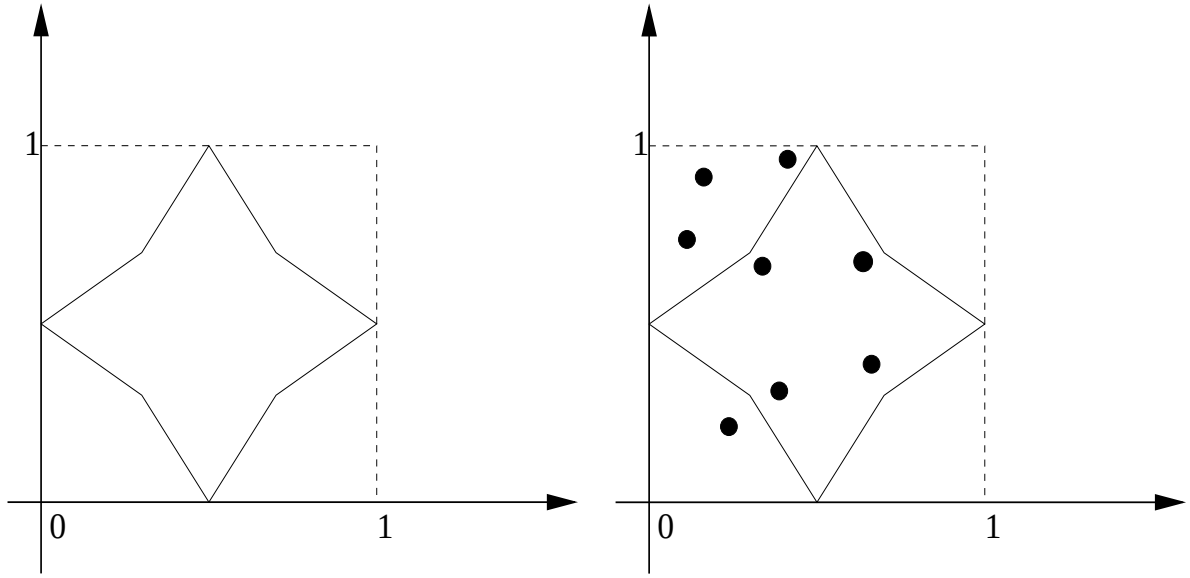


Figure 2.3: Simulation of uniformly distributed points inside the star-shaped set (left) by simulating uniformly points inside the unit square (formed by the axes and the dashed lines) and rejecting those who fall outside the star-shaped set (right)

not surprising (and it can be proved) that points can be simulated uniformly inside a set C (e.g the star-shaped set in Figure 2.3) by simulating uniformly points on a superset $C \subset B$ (e.g the unit square in Figure 2.3) and rejecting those which fall outside C. The hope is that it is easier to simulate points uniformly inside B than inside C. In our example, it is trivial to simulate a point uniformly inside B, simply take (U, V), where U, V are independent $U[0, 1]$ variables. Notice that if B is much larger than C then most of the simulated points will be rejected, thus it is desirable to find B so that to match closely C.

We now turn to the problem of simulating from a given density f, and link this problem with the rejection idea described above. Suppose that f is a probability density function on the real line. Moreover, let (X, Y) be a uniformly distributed point under the density f (the grey-shaded area in Figure 2.4). Therefore, if $h_{(X,Y)}(x, y)$ denotes the probability density function of (X, Y), then

$$h_{(X,Y)}(x, y) = 1, \ -\infty < x < \infty, 0 < y < f(x).$$

It is easy to show (check!) that the marginal density of X is exactly f. The rejection sampling idea described earlier can be used to generate samples uniformly under f. Suppose that $g(x)$ is another density function, which is easy to simulate from, and K a constant such that

$$f(x) \leq Kg(x), \text{ for all } x,$$

therefore the set of points under f lies entirely within the set of points under Kg (see Figure 2.5). A point (X^*, Y^*) can be simulated uniformly under Kg, by sampling X^* from g, and then simulate Y^* given X^* from a uniform distribution $U[0, Kg(X^*)]$ (composition method). Applying the rejection idea described earlier, if we keep all such points that fall under f as well, i.e all those (X^*, Y^*) such that $Y^* < f(X^*)$ we will obtain a sample of points uniformly distributed under f. The X^*s of this sample are an independent sample from f.

Thus, the procedure described above is equivalent to the following algorithm:

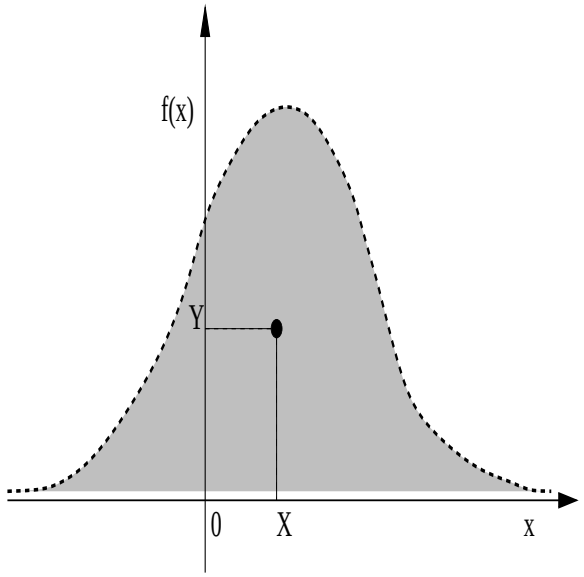


Figure 2.4: The density f (thick dashed line) and a point (X, Y) uniformly distributed under the area defined by f and the x-axis.

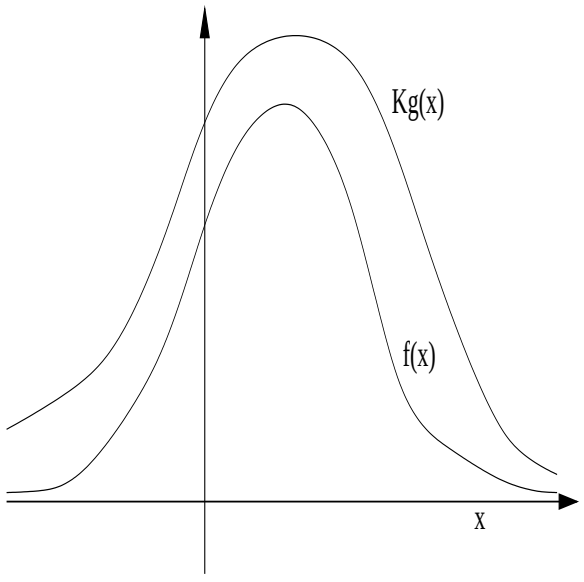


Figure 2.5: The density f and the envelope function $Kg(x)$.

Algorithm 2.1.

1. *Simulate* x^* *from* $g(x)$.
2. *Simulate* y^* *from* $U(0, Kg(x^*))$.
3. *Accept* x^* *if* $y^* \leq f(x^*)$.
4. *Continue.*

Furthermore, $\Pr(X \text{ accepted}) = 1/K$.

Formal Proof Here is given the formal proof that rejection sampling works. Generalizing slightly, let f be the density we want to simulate from up to a proportionality constant, i.e $\int f(x)dx = c_1$, and g be the proposal density also up to a proportionality constant, $\int g(x)dx = c_2$, where both c_1 and c_2 might be unknown. Let K be such that $f(x) \leq Kg(x)$, for all x, and I be an indicator variable, such that $\Pr(I = 1 \mid X^* = x^*) = f(x^*)/(Kg(x^*))$, and $\Pr(I = 0 \mid X^* = x^*) = 1 - f(x^*)/(Kg(x^*))$. The rejection sampling algorithm we have presented, can be implemented using the indicator variables, as:

1. Simulate x^* from $g(x)$.
2. Conditionally on $X^* = x^*$, simulate the indicator I, so that $\Pr(I = 1 \mid X^* = x^*) = f(x^*)/(Kg(x^*))$.
3. Accept x^* if $I = 1$.
4. Continue.

Therefore, all those x^*s which are paired with $I = 1$ are retained as sample from f. Since x^* is simulated from g, we have that:

$$P(I = 1) = \int P(I = 1 \mid X^* = x^*) \frac{g(x^*)}{c_2} dx^* = \frac{c_1}{c_2 K},$$

which gives the overall acceptance probability of the algorithm (which equals $1/K$ when f and g are proper densities). To show that rejection sampling ends up with samples from f we need to show that the conditional density of X^* given that $I = 1$ is proportional to f. Let $k(x^* \mid I = 1)$ denote this conditional density. Then, by Bayes theorem:

$$k(x^* \mid I = 1) = \frac{\Pr(I = 1 \mid X^* = x^*) g(x^*)/c_2}{\Pr(I = 1)} = f(x^*)/c_1,$$

which completes the proof.

□

Note that we only need to know f up to a normalizing constant in order for this technique to work. This is a big advantage of the method since in many statistical applications we need to simulate from densities for which we cannot compute the normalizing constant (typically, maximization is easier than integration). The efficiency of the procedure depends on the quality of the agreement between f and the bounding envelope Kg since if a large value of K is necessary, then the acceptance probability is low, so that large numbers of simulations are needed to achieve a required sample size.

As an example, consider the distribution with density

$$f(x) \propto x^2 e^{-x}; \quad 0 \leq x \leq 1 \tag{2.3}$$

a truncated gamma distribution. Then, since $f(x) \leq e^{-x}$ everywhere, we can set $g(x) = \exp(-x)$ and so simulate from an exponential distribution, rejecting according to the above algorithm. Figure 2.6 shows both $f(x)$ and $g(x)$. Clearly in this case the envelope is very poor so the routine is highly inefficient (though statistically correct).

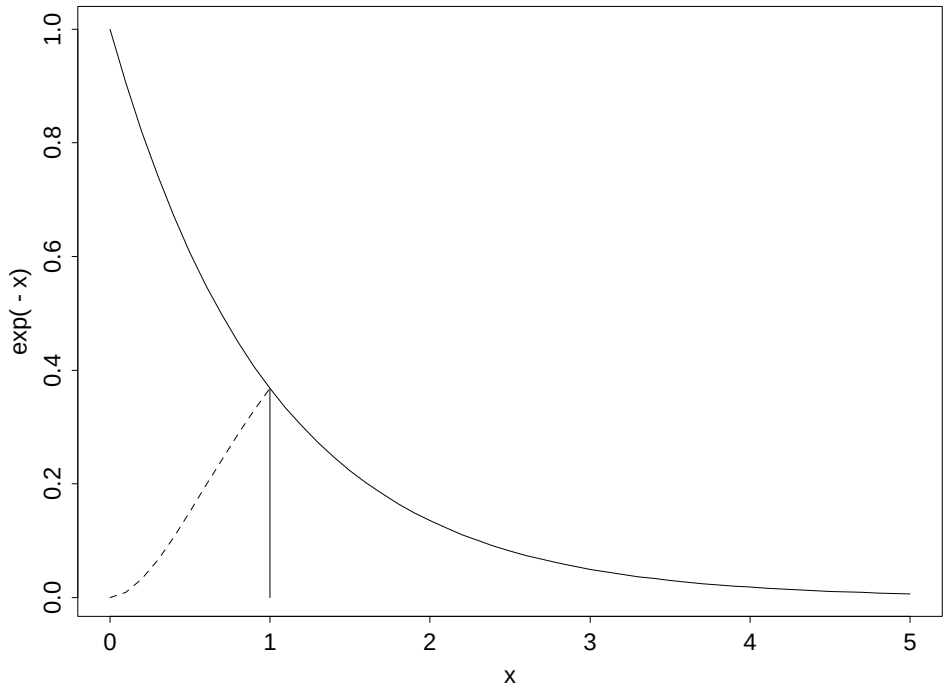


Figure 2.6: Scaled density and envelope

Applying this to generate a sample of 100 data using the following code

```
 rej.sim=function(n)
{
        r = NULL
        for(i in 1:n) {
                t = -1
                while(t < 0) {
                        x = rexp(1, 1)
                        y = runif(1, 0, exp( - x))
                        if(x > 1)
                                t =  - y
                        else t = x^2 * exp( - x) - y
                }
                r[i] = x
        }
```

```
        r
}
```

gave the histogram in Figure 2.7.

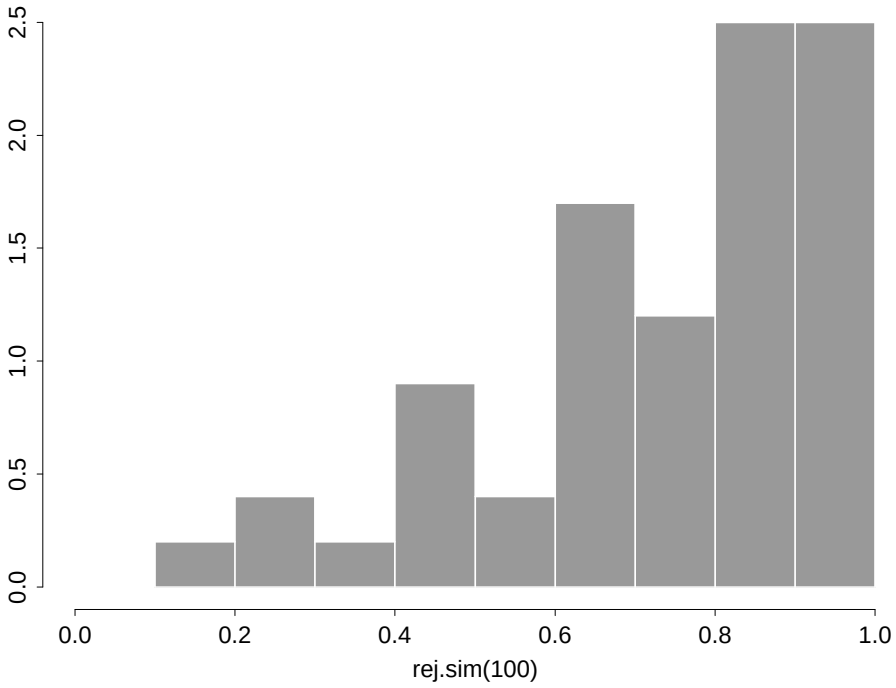


Figure 2.7: Histogram of simulated data

2.6.4 Ratio of uniforms

An adaptation of the rejection algorithm which works well for many distributions is the ratio of uniforms method. Here a pair of independent uniforms are simulated and the ratio accepted as a simulant from the required distribution according to a rejection scheme.

The basis of the technique is the following argument. Suppose h is a non–negative function such that $\int h < \infty$ and we let $C_h = \{(u, v) : 0 \leq u \leq \sqrt{h(v/u)}\}$. Then if (U, V) is uniformly distributed over C_h then $X = V/U$ has pdf $h/\int h$.

So, to simulate from a density proportional to h, we simulate uniformly over the region C_h, and take ratios of coordinates. In practice, C_h may be complicated in form, so the only practical solution is to bound it with a rectangle (if possible), simulate within the rectangle (by a pair of uniforms), and apply rejection: hence, *ratio of uniforms*.

The reason this works is as follows. Let Δ_h be the area of C_h. Then on changing variables $(u, v) \to (u, x)$, where $x = v/u$,

$$\Delta_h = \int\int_{C_h} dudv \quad = \int\int_0^{\sqrt{h(x)}} u \; dudx \quad = \int \frac{1}{2} h(x) dx \tag{2.4}$$

Because of the uniformity of (U, V) over C_h, (U, V) has pdf $1/\Delta_h$ so that on transformation, (U, X) has pdf u/Δ_h, and integrating out U gives the marginal pdf of X as:

$$\Delta_h^{-1} \int_0^{\sqrt{h(x)}} u \, du \quad = \quad h(x)/\{2\Delta_h\} \quad = \quad h(x)/\int h(x)dx \tag{2.5}$$

Thus V/U has pdf proportional to h. Again, a key property of this method is that h is only required to be specified up to proportionality.

As discussed above, this is only useful if we can generate uniformly over C_h, which is most likely to be achieved by simulating uniformly within a rectangle $[0, a] \times [b_-, b_+]$ which contains C_h (provided such a rectangle exists). If it does, we have the following algorithm.

Algorithm 2.2.

1. *Simulate independent* $U \sim U[0, a]$, $V \sim U[b_-, b_+]$.
2. *If* $(U, V) \in C_h$, *accept* $X = V/U$, *otherwise repeat.*
3. *Continue.*

As an example, consider the Cauchy distribution with density

$$h(x) \propto \frac{1}{1 + x^2} \tag{2.6}$$

Then $C_h = \{(u, v) : 0 \leq u \leq \sqrt{h(v/u)}\} = \{(u, v) : 0 \leq u, u^2 + v^2 \leq 1\}$, a semicircle. Hence we can take $[0, a] \times [b_-, b_+] = [0, 1] \times [-1, 1]$ and get the algorithm

Algorithm 2.3.

1. *Simulate independent* $U \sim U[0, 1]$, $V \sim U[-1, 1]$.
2. *If* $u^2 + v^2 \leq 1$ *accept* $x = v/u$, *otherwise repeat.*
3. *Continue.*

This can be implemented with the R code

```
ru.sim=function(n) {
        r = NULL
        for(i in 1:n) {
                t = 1
                while(t > 0) {
                        u = runif(1, 0, 1)
                        v = runif(1, -1, 1)
                        t = u^2 + v^2 - 1
                }
                r[i] = u/v
        }
        r
}
```

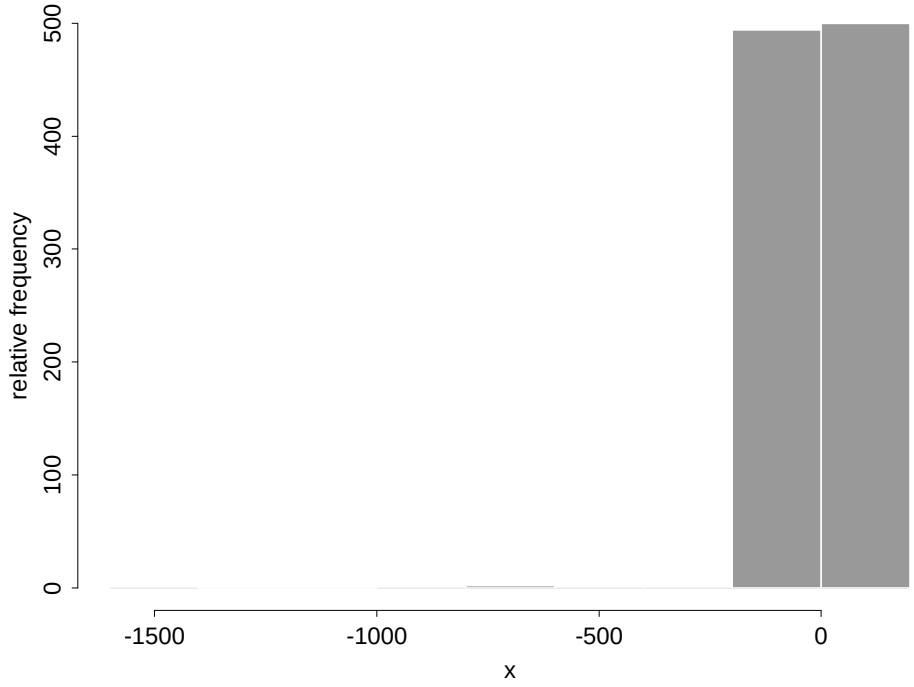


Figure 2.8: Histogram of simulated Cauchy

and a histogram of 1000 simulated values is given in Figure 2.8 (note the unusual scale (!) because the heaviness of the Cauchy tail causes one or two simulated values to be extreme relative to the rest of the data).

A number of modifications have been proposed to improve on the efficiency of this procedure, which amount to rescaling and locating distributions before applying the method.

Another method for improving the efficiency is by a process known as 'squeezing' or 'pre–testing'. This applies to both the rejection and ratio of uniform methods. The point is that, in the ratio of uniforms method for example, the slowest part of the algorithm can be the check of whether $(u, v) \in C_h$ or not. However, there may be simpler regions C_1 and C_2 such that $C_1 \subset C_h \subset C_2$, so that if (u, v) is found to lie inside C_1 or outside C_2 then we immediately know whether it lies inside C_h or not.

2.7 Monte–Carlo integration

In one form or another, the reason for simulation can often be evaluating an integral. Section 2.3 showed that we can use sample means of the form

$$\hat{\theta}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i), \tag{2.7}$$

where the x_is are a sample from the distribution of X with density f, in order to approximate expectations

$$\theta = \int \phi(x) f(x) dx = E_f(\phi(X)), \tag{2.8}$$

where $E_f(\phi(X))$ denotes the expectation of $\phi(X)$ with respect to the density f. The notation $\hat{\theta}_n(f)$ reflects that it is an estimate of θ (thus the "hat" notation) which depends both on the sample size n and the fact that the x_is have been simulated from f. The idea behind Monte Carlo integration is to express an integral we wish to compute as an expectation of a random variable and then simulate samples from that random variable to approximate the integral by the appropriate sample mean. This approach is remarkably easy to use, even in high dimensions. The cost for this simplicity is that the variance of the sample mean estimators might be high.

(The main advantage of Monte Carlo integration over numerical analysis methods for approximating integrals - such as Simpson's rule - is that the latter suffer in high dimensions. However, for small-dimensional integrals (say up to dimension 8) numerical analysis methods with the same computational cost as Monte Carlo can achieve a much better approximation.)

As an example of this, suppose we wish to calculate $P(X < 1, Y < 1)$ where (X, Y) are bivariate Standard Normal with correlation 0.5. This can be written as

$$\int I_A(x, y) f(x, y) dx dy \tag{2.9}$$

where f is the bivariate normal density, and I_A is the indicator function on $A = \{(x, y) : x < 1, y < 1\}$. Thus, provided we can simulate from the bivariate normal, we can estimate this probability as

$$n^{-1} \sum_{i=1}^{n} I_A(x_i, y_i) \tag{2.10}$$

which is simply the proportion of simulated points falling in A. There are various approaches to simulating bivariate Normals. In my experiment, on the basis of 1000 simulations I got 0.763 as an estimate of the probability.

2.7.1 Importance sampling

There are several applications (Bayesian inference and filtering are two classic examples) where, although we want to estimate (2.8), we cannot obtain samples from f by any of the available methods (for example inverse CDF, rejection sampling, etc) due to the complicated and/or high-dimensional form of the density. Moreover, we might be interested in estimating the expectation of several different functions with respect to f. Suppose, however, that instead we can sample from a density g defined on the same space as f, which dominates f, in the sense that $g(x) = 0 \implies f(x) = 0$. For a given function ϕ, we re-write (2.8) as,

$$\theta = \int \phi(x)\frac{f(x)}{g(x)}g(x)dx = E_g(\phi(X)w(X)), \quad \text{where } w(x) = \frac{f(x)}{g(x)};$$

$w(X)$ is known as the *importance weight* associated to the sampled point (*particle*) X, and g is called the *importance density*. The above expression suggests two different so-called *importance sampling* estimators:

$$\hat{\theta}_n(g) = \frac{1}{n}\left\{\sum_{i=1}^{n}\phi(x_i)w(x_i)\right\},$$

and

$$\hat{\theta}^b_n(g) = \frac{\sum_{i=1}^{n}\phi(x_i)w(x_i)}{\sum_{i=1}^{n}w(x_i)},$$

where the x_is are iid draws from g. As an exercise, you can show that $\hat{\theta}_n(g)$ is an unbiased estimator of θ. Moreover, you can also show that $E_g(w(X)) = 1$, therefore $\hat{\theta}^b_n(g)$ is a consistent estimator of θ (appealing to a stronger version of Theorem 2.3.1). Generally $\hat{\theta}^b_n(g)$ is a biased estimator of θ (thus the superscript "b"), but in many cases it might have a smaller mean square error than $\hat{\theta}_n(g)$. However, the main advantage of the biased estimator over the unbiased is that the former does not require knowledge of the normalizing constants of f and g in order to be computed.

A necessary requirement for the implementation of the method is that the importance density g is easy to sample from. An important question however, is which is the "optimal" choice of g among the class of densities which we can sample from and which dominate f. When interest lies in estimating the expectation of a specific function ϕ, the next section provides an answer (where it will also become clear why the method is called *importance sampling*). However, when the expectation of several different functions are to be estimated we can informally recommend that g should be chosen to be as "close" to f as possible. More formally, for importance sampling to be efficient it is crucial to ensure that the variance of the importance weights is finite:

$$Var(w(X)) = \int (f(x)/g(x) - 1)^2 g(x)dx < \infty.$$

It can be seen from this expression that this variance depends on the discrepancy between f and g and it would be 0 if we could indeed use f as the importance density. Check that when $f(x)/g(x) \leq K < \infty$, the variance of the importance weights will be finite. How does this relate to rejection sampling?

2.7.2 Variance Reduction

Although Monte Carlo methods (such as Monte Carlo integration) are very intuitive, the methods employed to reduce the variance of the resulting estimates are often rather involved. Nevertheless, in many modern applications (such as Monte Carlo maximum likelihood estimation,

particle filters etc) variance reduction techniques have a substantial role to play, since they improve the precision of the Monte Carlo estimates by many orders of magnitude. In this section we describe some of these methods. However, the applications given here are rather basic and do not represent the most exciting problems where variance reduction is relevant.

Importance sampling

Suppose that we want to estimate

$$\theta = \int \phi(x) f(x) dx, \tag{2.11}$$

for a specific function ϕ. It will be useful for the context of this section to take ϕ to be positive. The previous section introduced the importance sampling as a method for Monte Carlo integration, especially when it is infeasible to simulate from f. However, even if simulation from f is possible, the importance sampling estimator $\hat{\theta}_n(g)$ (or $\hat{\theta}_n^b(g)$) can give a much more efficient estimator than the plain-vanilla estimator based on samples from f, $\hat{\theta}_n(f)$, if g is appropriately chosen.

The variance of $\hat{\theta}_n(f)$ is large when $\phi(x)$ varies a lot in areas where f is very small. The example in Figure 2.9, where $\phi(x) = x^8$ and $f(x) = e^{-x^2/2}/\sqrt{2\pi}$ is revealing as to why $\hat{\theta}_n(f)$ might give poor estimates of θ.

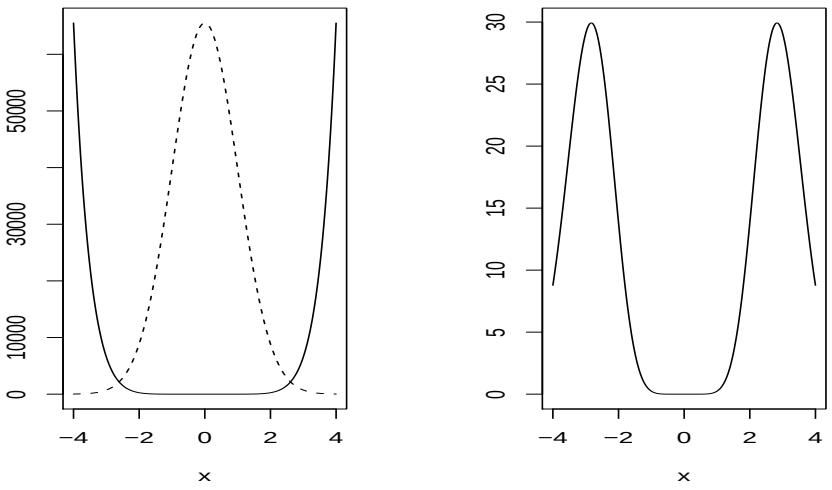


Figure 2.9: Left: $\phi(x) = x^8$ (solid line) and $f(x) \propto e^{-x^2/2}$ (dashed line). f has been scaled to have the same maximum as ϕ for visual purposes. Right: The quantity to be estimated is the area under the plotted curve $f(x)\phi(x)$.

Instead, it is preferable to simulate draws from the *important* areas, i.e the areas where ϕ varies a lot. More formally, the variance of $\hat{\theta}_n(g)$ is

$$Var(\hat{\theta}_n(g)) = n^{-1} \int \left\{ \frac{\phi(x) f(x)}{g(x)} - \theta \right\}^2 g(x) dx \tag{2.12}$$

This variance can be much lower than the variance of $\hat{\theta}_n(f)$, if g can be chosen so as to make the ratio $\phi(x)f(x)/g(x)$ nearly constant in x. Essentially what is happening is that the simulations are being concentrated in the areas where there is greatest variation in the integrand, so that the informativeness of each simulated value is greatest.

The following example illustrates the idea. Suppose we want to estimate the probability $P(X > 2)$, where X follows a Cauchy distribution with density function

$$f(x) = \frac{1}{\pi(1 + x^2)} \tag{2.13}$$

so we require the integral

$$\theta = \int I_A(x)f(x)dx \tag{2.14}$$

where $A = \{x : x > 2\}$. The most intuitive and straightforward Monte Carlo method would be to simulate values from the Cauchy distribution directly and apply (2.7), that is approximate $P(X > 2)$ by the proportion of the simulated values which are bigger than 2. Nevertheless, the variance of this estimator is substantial, since it is the empirical proportion of draws which exceed 2, but due to the importance density (f in this case) draws rarely exceed 2, so the variance is large compared to its mean.

The previously suggested method, although inefficient it still is very intuitive and beautifully simple, since it approximates a probability by a sample frequency. On the other hand, the method we will now describe is not intuitive but it can be very efficient! The key in understanding this method is to think of θ not as a probability, but as an area under a function. Specifically, in our example θ is the area under $f(x)$, for $x \geq 2$ (see Figure 2.10a),

$$\theta = \int_2^\infty f(x)dx.$$

Observe that for large x, $f(x)$ is approximately equal to $1/(\pi x^2)$, since the term πx^2 dominates

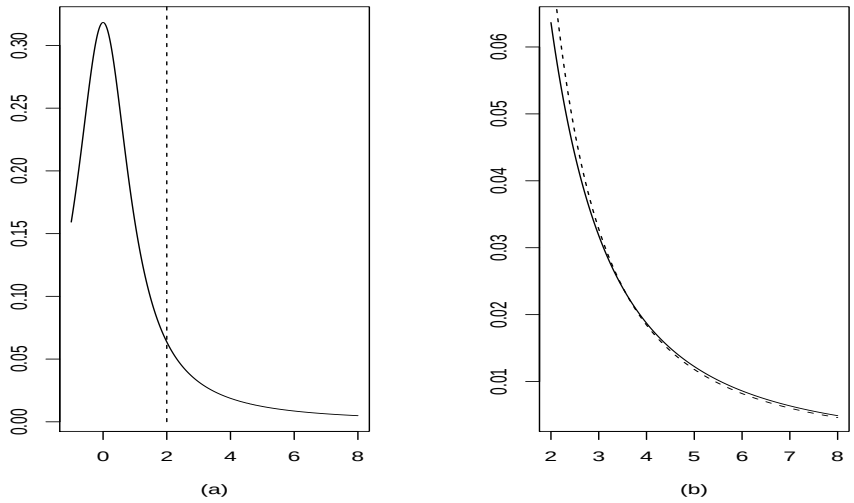


Figure 2.10: (a): The Cauchy density, (b) the Cauchy density $f(x), x \geq 2$ with solid line and the function $0.1476g(x)$, $g(x) = 2/x^2$ with a dashed line superimposed.

over π in the denominator of f. If we normalise this function so that it is a density function with support on $[2, \infty)$ we get $g(x) = 2/x^2, x \geq 2$, and we re-write θ as

$$\theta = \int_2^\infty \frac{f(x)}{g(x)} g(x) dx.$$

It is easy to simulate from g using the inversion method: take $x_i = 2/u_i$ where $u_i \sim U[0, 1]$. Thus, our estimator becomes:

$$\hat{\theta}_n(g) = n^{-1} \sum_{i=1}^{n} \frac{x_i^2}{2\pi(1 + x_i^2)} \tag{2.15}$$

where $x_i = 2/u_i$. Notice that $f(x)/g(x)$ is close to being a constant (see Figure 2.10b) therefore $\hat{\theta}_n(g)$ has much smaller variance than $\hat{\theta}_n(f)$. Implementing this with the R function

```
i.s=function(n) {
        x = 2/runif(n)
        psi = x^2/(2 * pi * (1 + x^2))
        mean(psi)
}
```

gave the estimate $\hat{\theta} = .1478$. The exact value is $.5 - \pi^{-1} \tan 2 = .1476$.

Figure 2.11 compares the estimators $\hat{\theta}_n(f)$ and $\hat{\theta}_n(g)$ using a simulation. We plot the convergence of the sample means $\hat{\theta}_n(f)$ (left), and $\hat{\theta}_n(g)$ (right) for $n = 1, \ldots, 1000$, and we have repeated this experiment 10 different times. Notice that $\hat{\theta}_n(g)$ converges very quickly to the true value of θ for all 10 independent realizations of the experiment. On the other hand, some of the paths corresponding to $\hat{\theta}_n(f)$ are far from the true value of θ even for $n = 1000$. The spread of the values $\hat{\theta}_n(f)$ for any n among the 10 different simulated paths, shows the high variance of the estimator. However, the average values of the estimator is θ, since the estimator is unbiased.

Control variates

In general, the idea of control variates is to modify an estimator according to a correlated variable whose mean is known. To fix ideas, suppose we wish to estimate $\theta = E(Z)$ where $Z = \phi(X)$. Then $\hat{\theta} = Z$ is an unbiased estimator of θ. Suppose now that W is another random variable which however has known mean $E(W)$. Then, we can construct an alternative unbiased estimator

$$\hat{\theta} = Z - W + E(W). \tag{2.16}$$

In this context W is called a *control variate*. The variance of the estimator is given by

$$\text{Var}(\hat{\theta}) = \text{Var}(Z) - 2 \text{ Cov}(W, Z) + \text{Var}(W). \tag{2.17}$$

Therefore it is desirable to choose as control variate a random variable W which we believe is correlated with Z so that Cov (W, Z) is large and the variance of the above estimator is smaller than the variance of the basic estimator $\hat{\theta} = Z$. Notice that

$$\hat{\theta} = Z - \beta(W - E(W)) \tag{2.18}$$

is also an unbiased estimator of θ, for any β, where we now have the added degree of freedom to choose β in order to minimize the variance of the estimator. Generally, we could come up

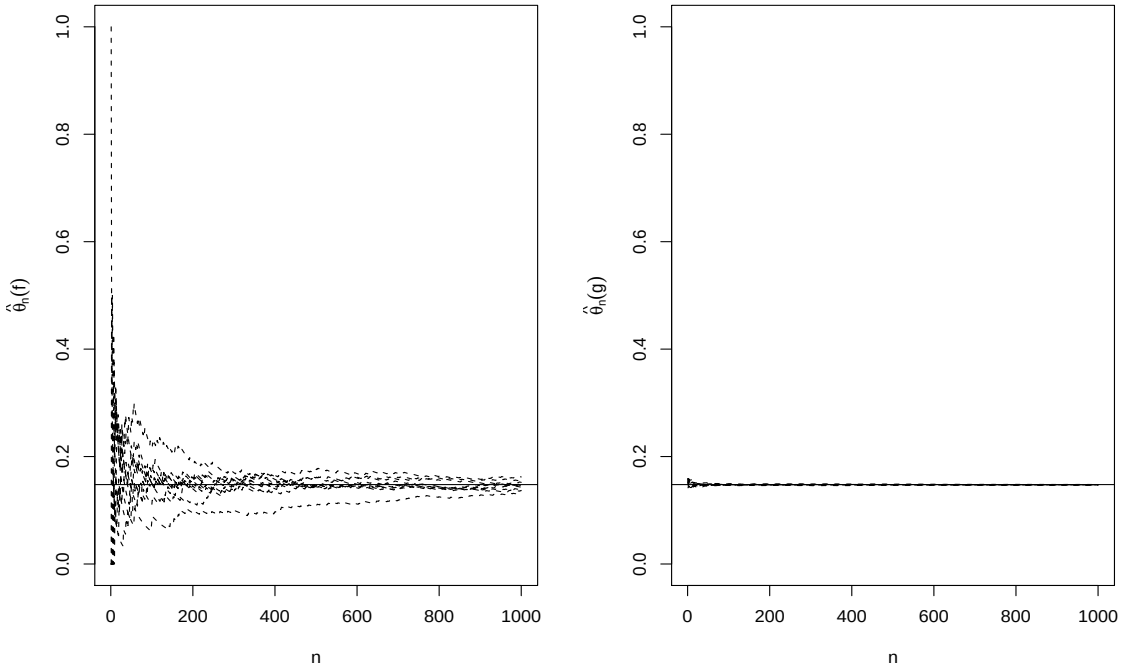


Figure 2.11: Convergence of the standard $\hat{\theta}_n(f)$ (left) and the importance $\hat{\theta}_n(g)$ (right) sampled mean.

with several variables, say $W^{(1)}, \ldots, W^{(p)}$, which could serve as candidate control variates and construct the following unbiased estimator:

$$\hat{\theta} = Z - \beta_1(W^{(1)} - E(W^{(1)})) - \cdots \beta_p(W^{(p)} - E(W^{(p)})), \tag{2.19}$$

where the coefficients $\beta = (\beta_1, \ldots \beta_p)$ are chosen to minimize the variance of the estimator. Linear regression theory suggests to choose the coefficients by regressing observations of Z on observations of the control variates. To avoid biases, one way to implement this idea is to run a small preliminary simulation experiment, where independent realizations from Z and from $W^{(j)}$s are simulated. Based on these realizations the coefficients are estimated using linear regression yielding an estimate $\hat{\beta}$ say. Subsequently n independent realizations of Z and $W^{(j)}$s are simulated and θ is estimated by

$$\frac{1}{n}\sum_{i=1}^{n}\left\{Z_i - \hat{\beta}_1(W_i^{(1)} - E(W^{(1)})) - \cdots \hat{\beta}_p(W_i^{(p)} - E(W^{(p)}))\right\}. \tag{2.20}$$

It turns out (and it is quite easy to see) that the variance of the estimator becomes approximately $n^{-2}RSS$ where RSS is the residual sum of squares of the regression fit.

This is easily applied in the context of Monte–Carlo integration. Again developing the Cauchy example, we require an estimate of

$$\theta = \frac{1}{2} - \int_0^2 f(x)dx \tag{2.21}$$

where $f(x) = \frac{1}{\pi(1+x^2)}$. We can estimate this as

$$\hat{\theta} = \frac{1}{2} - 2 \times \frac{1}{n}\sum_{i=1}^{n} f(x_i), \tag{2.22}$$

where the $x_i \sim U[0, 2]$.

To estimate the integral using a control variate we seek a function (or functions) with known mean which varies with $f(x)$ over $[0, 2]$. A Taylor series expansion of $f(x)$ suggests control variates of x^2 and x^4, whose means, with respect to the $U[0, 2]$ distribution, are easily evaluated over $[0, 2]$ as $8/6$ and $32/10$. Now, in principle, any function of the form $\beta_1 x^2 + \beta_2 x^4$ suffices as a control variate. To minimize variance however, we need to choose the β's so as to optimize agreement between $f(x)$ and $\beta_1 x^2 + \beta_2 x^4$. This is achieved through simple regression. I chose to simulate 10 observations uniformly over $[0, 2]$, leading to the regression equation

$$W(x) = .288 - .143x^2 + .024x^4 \tag{2.23}$$

and hence,

$$\hat{\theta} = \frac{1}{2} - 2\left[\frac{1}{n}\sum_{i=1}^{n}\{f(X_i) - W(x_i)\} + .288 \times 2 - .143 \times 8/6 + .024 \times 32/10\right] \tag{2.24}$$

With $n = 1000$ I got an estimate of $\theta = .1474$.

Antithetic variates

Antithetic variates are almost the converse of control variates: we obtain a variate Z^* which has the same distribution as Z, but is negatively correlated with Z. Then,

$$\hat{\theta} = \frac{1}{2}(Z + Z^*) \tag{2.25}$$

is an unbiased estimator of θ, with variance:

$$\mathrm{Var}(\hat{\theta}) = \frac{1}{2}\mathrm{Var}(Z)\{1 + \mathrm{corr}(Z, Z^*)] \tag{2.26}$$

which constitutes a reduction in variance provided the correlation is indeed negative. For simple problems, antithetic variates are easily achieved by inversion, since if $Z = F^{-1}(U)$ then $Z^* = F^{-1}(1-U)$ has the same distribution as Z and can be shown to be negatively correlated with Z for all choices of F. Applying this to the estimation of $\theta = \frac{1}{2} - \int_0^2 f(x)dx$ in the Cauchy example leads to the estimator

$$\frac{1}{2} - \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{1}{\pi(1+u_i^2)} + \frac{1}{\pi(1+(2-u_i)^2)}\right\} \tag{2.27}$$

where $u_i \sim U[0,2]$.

Chapter 3

Gibbs Sampling

In this chapter we study the first of the MCMC algorithms, namely the Gibbs sampler.

3.1 Brief overview of Bayesian inference and notation

Here we give a brief reminder of key aspects of Bayesian statistics and introduce notation that will be used throughout this course.

Let $\mathbf{y} = (y_1, \ldots, y_n)$ be a vector of observations with *sampling density* (*likelihood*) given by $f(\mathbf{y}|\theta)$, and let $\pi(\theta)$ be the *prior distribution* of the unknown parameter θ.

Inference on the parameter is based on the *posterior distribution*, which is computed via *Bayes' theorem*:

$$\pi(\theta|\mathbf{y}) = \frac{f(\mathbf{y}|\theta)\pi(\theta)}{f(\mathbf{y})} \propto f(\mathbf{y}|\theta)\pi(\theta), \tag{3.1}$$

where

$$f(\mathbf{y}) = \int f(\mathbf{y}|\theta)\pi(\theta)d\theta \quad \text{is called } \textit{the evidence or marginal likelihood}. \tag{3.2}$$

Forecasting a future value of an observation y_f is based on the *predictive distribution*, whose density is given by

$$f(y_f|\mathbf{y}) = \int f(y_f|\mathbf{y}, \theta)\pi(\theta|\mathbf{y})d\theta. \tag{3.3}$$

In the common case of independent sampling (that is, when the future observation y_f is independent of $\mathbf{y}$), we have $f(y_f|\mathbf{y}, \theta) = f(y_f|\theta)$. Hence, the predictive density becomes

$$f(y_f|\mathbf{y}) = \int f(y_f|\theta)\pi(\theta|\mathbf{y})d\theta. \tag{3.4}$$

Often we are not interested in the entire posterior distribution $\pi(\theta|\mathbf{y})$ but, rather, in some feature of it. The latter feature often takes the form of the posterior expectation of some function $t(\theta)$, that is

$$E[t(\theta)|\mathbf{y}] = \int t(\theta)\pi(\theta|\mathbf{y})d\theta.$$

For example,

• if $t(\theta) = \theta$, we obtain the posterior expectation of the parameter θ;

• if $t(\theta) = \theta^r$ for some $r > 1$, we obtain higher order posterior moments of θ;

• if $t(\theta) = I[\theta \in A]$ (the indicator function of a set A), then $E[t(\theta)|\mathbf{y}]$ is the posterior probability that the parameter θ lies in the set A;

• if $t(\theta) = f(y_f|\theta)$, then $E[t(\theta)|\mathbf{y}]$ is the predictive density $f(y_f|\mathbf{y})$ (assuming independent sampling as explained above).

Other features of interest might be quantiles (*e.g.* median or certain percentiles) of the posterior distribution. Furthermore, if the parameter $\theta = (\theta_1, \ldots, \theta_d)$ is multi-dimensional, we are likely to be also interested in the *marginal posterior distributions* of the components: $\pi(\theta_j|\mathbf{y})$ for $j = 1, \ldots, d$.

3.2 Bayesian Statistics and conditional conjugacy

More than any other technique, Markov Chain Monte Carlo (MCMC) has been responsible for a current resurgence in Bayesian statistics, since its application allows a vast range of Bayesian models, previously thought to be completely intractable, to be easily estimated. We will look at a number of applications later. Although MCMC is intrinsically just a device for simulating from a multivariate distribution, its application in statistics using the Gibbs sampler is greatly aided by the statistical concept known as *conditional conjugacy*.

Recall the notion of *conjugacy* used (normally) in Bayesian problems where there is a single unknown parameter θ, say. The concept is best illustrated using an example. Suppose that our data $y_1, \ldots, y_n$ are independent observations from an exponential distribution with parameter θ. Thus the likelihood is

$$f(\mathbf{y}|\theta) = \theta^n e^{-\theta \sum_{i=1}^n y_i} \ . \tag{3.5}$$

Suppose our prior for θ, $\pi(\theta)$, is Gamma$(2, 1)$. Then the posterior distribution for θ given $\mathbf{y}$ is Gamma$(2 + n, 1 + \sum_{i=1}^n y_i)$. Thus the posterior distribution of θ remains in the Gamma family whatever data is observed. The Gamma family is called a *conjugate* family for this problem. The major advantage of the conjugate family is that it allows the effect of the data on our posterior beliefs to be summarised in terms of two parameters, which in turn describe a simple well known and easy to handle distribution.

In many problems with a single unknown parametert it is usually possible to find useful conjugate priors. However, even though in theory this is also possible in higher dimensional situations, in practice the conjugate families end up being highly complex and difficult to summarise.

On the other hand, many multi-parameter Bayesian problems exhibit *conditional conjugacy*. Consider the following example to illustrate the concept.

Example 3.1. *Suppose $Y_i|\mu, \omega \sim N(\mu, 1/\omega)$ independently for $i = 1, \ldots, n$. Hence the sampling density is*

$$f(\mathbf{y}|\mu, \omega) = \prod_{i=1}^n \left(\frac{\omega}{2\pi}\right)^{1/2} \exp\left\{-\frac{\omega}{2}(y_i - \mu)^2\right\} \propto \omega^{n/2} \exp\left\{-\frac{\omega}{2}\sum_{i=1}^n (y_i - \mu)^2\right\}.$$

Suppose also we have prior distributions for μ and ω:

$$\mu \sim N(\mu_0, 1/\kappa_0) \quad \text{and} \quad \omega \sim \text{Gamma}\,(\alpha_0, \lambda_0) \tag{3.6}$$

where μ and ω are considered to be a priori independent and μ_0, κ_0, α_0 and λ_0 are considered to be known hyperparameters. The posterior density of (μ, ω) is:

$$\pi(\mu, \omega|\mathbf{y}) \propto e^{-\frac{\omega}{2}\sum_{i=1}^n (y_i - \mu)^2} e^{-\frac{\kappa_0}{2}(\mu - \mu_0)^2} \omega^{\frac{n}{2} + \alpha_0 - 1} e^{-\lambda_0 \omega}. \tag{3.7}$$

Note that although the form of the prior distribution is tractable, the posterior distribution is a complex two-dimensional distribution. This is a typical characteristic of Bayesian analyses of multi-dimensional problems.

Note, however, that the conditional posterior distributions belong to the same families as the priors:

$$\pi(\mu|\omega,\mathbf{y}) \propto e^{-\frac{\omega}{2}\sum_{i=1}^{n}(y_i-\mu)^2} e^{-\frac{\kappa_0}{2}(\mu-\mu_0)^2}, \textit{ which is } N\left(\frac{\kappa_0\mu_0+\omega\sum_{i=1}^{n} y_i}{\kappa_0+n\omega}, \frac{1}{\kappa_0+n\omega}\right), \tag{3.8}$$

while

$$\pi(\omega|\mu,\mathbf{y}) \textit{ is } \text{Gamma}\left(\alpha_0+\frac{n}{2}, \lambda_0+\frac{\sum_{i=1}^{n}(y_i-\mu)^2}{2}\right). \tag{3.9}$$

This phenomenon, which is called conditional conjugacy, will motivate the introduction of the Gibbs sampler in the next section.

In multi-dimensional Bayesian problems it is rarely possible to compute analytically summary statistics such as posterior means and variances, or even posterior probabilities. Therefore it is necessary to estimate the quantities of interest using a Monte Carlo approach. However simulating from an arbitrary high dimensional distribution is usually difficult and often impossible to do directly. Instead, Markov chain Monte Carlo (MCMC) methods are used to simulate a Markov chain, whose stationary or limiting distribution is the posterior distribution of interest.

The concept of conditional conjugacy is crucial in the construction of one of the basic forms of MCMC: the Gibbs sampler.

3.3 The Gibbs sampler

We wish to obtain a sample from the multivariate distribution $\pi(\theta_1,\ldots,\theta_d)$. We shall call this distribution the *target* distribution. In Bayesian statistics, the target distribution is the joint posterior distribution. The Gibbs sampler obtains a sample from $\pi(\theta_1,\ldots,\theta_d)$ by succesively and repeatedly simulating from the conditional distributions of each component given the other components. Under conditional conjugacy, this simulation step is usually straightforward. The algorithm is as follows:

Algorithm 3.1.

1. *Initialize with* $\theta = (\theta_1^{(0)},\ldots,\theta_d^{(0)})$.
2. *Simulate* $\theta_1^{(1)}$ *from the conditional distribution* $\pi(\theta_1|\theta_2^{(0)},\theta_3^{(0)}\ldots,\theta_d^{(0)})$.
3. *Simulate* $\theta_2^{(1)}$ *from the conditional distribution* $\pi(\theta_2|\theta_1^{(1)},\theta_3^{(0)},\ldots,\theta_d^{(0)})$.
4. ...
5. *Simulate* $\theta_d^{(1)}$ *from the conditional distribution* $\pi(\theta_d|\theta_1^{(1)},\theta_2^{(1)},\ldots,\theta_{d-1}^{(1)})$.
6. *Iterate this procedure.*

Under mild regularity conditions, convergence of the Markov chain to the stationary distribution $\pi(\theta_1,\ldots,\theta_d)$ is guaranteed, so after a burn–in period (that is, a number of iterations for which the

draws are discarded), the subsequent draws $(\theta_1^{(1)}, \ldots, \theta_d^{(1)})$, $\ldots$, $(\theta_1^{(J)}, \ldots, \theta_d^{(J)})$ can be regarded as realizations from this distribution.

As stressed, this procedure is valid in any situation where the requirement is a sample from a multivariate distribution. Applications have been concentrated in Bayesian statistics because the technique gives a sample based approach to posterior inference in situations where most other approaches are either difficult or impossible. As we have seen, Bayes' theorem has the form

$$\pi(\theta|\mathbf{y}) = \frac{f(\mathbf{y}|\theta)\pi(\theta)}{\int f(\mathbf{y}|\theta)\pi(\theta)d\theta} \tag{3.10}$$

and it is often impossible to solve the normalizing integrals (that is, to obtain the denominator in Bayes' theorem in closed form). With the Gibbs sampler, as long as we are able to simulate from each of the conditional posterior distributions, we obtain a sample from the joint posterior distribution without worrying about computing any integrals.

Once we obtain a sample $\theta^{(1)}, \ldots, \theta^{(J)}$ from $\pi(\theta|\mathbf{y})$ we can approximate any feature of the posterior distribution using its empirical counterpart from the simulated draws. For example, we can approximately compute

$$E[t(\theta)|\mathbf{y}] \approx \frac{1}{J}\sum_{j=1}^{J} t(\theta^{(j)}),$$

for any function $t(\theta)$ whose posterior expectation exists. We can also approximately compute the median of $\pi(\theta|\mathbf{y})$ as the median of the draws $\theta^{(1)}, \ldots, \theta^{(J)}$, and similarly for other quantiles. Furthermore, if $\theta = (\theta_1, \ldots, \theta_d)$, then the ith component of each of the draws $\theta^{(1)}, \ldots, \theta^{(J)}$ constitutes a sample from $\pi(\theta_i|\mathbf{y})$.

3.4 Implementing Gibbs sampling with R

To illustrate how to implement Gibbs sampling with the software R , we shall use Example 2.1. To simulate from the posterior distribution corresponding to equation (3.7) using Gibbs sampling, we need to generate draws from the distribution of μ given $(\omega, \mathbf{y})$ in equation (3.8), and from the distribution of ω given $(\mu, \mathbf{y})$ in equation (3.9). Thus, the Gibbs sampling *algorithm* for this problem is as follows:

Algorithm 3.2. *Gibbs sampling algorithm for Example 2.1:*

1. *Start the chain with some values* $(\mu^{(0)}, \omega^{(0)})$.
2. *Simulate* $\mu^{(1)}$ *from the distribution* $N\left(\frac{\kappa_0\mu_0+\omega^{(0)}\sum_{i=1}^n y_i}{\kappa_0+n\omega^{(0)}}, \frac{1}{\kappa_0+n\omega^{(0)}}\right)$.
3. *Simulate* $\omega^{(1)}$ *from the distribution* $\text{Gamma}\left(\alpha_0+\frac{n}{2}, \lambda_0+\frac{\sum_{i=1}^n (y_i-\mu^{(1)})^2}{2}\right)$.
4. *Iterate this procedure.*

The R function `gibbs`, printed below, implements this algorithm in R . I have stored this function in an external file called "gibbs.r". To be able to use this function, make sure you store the file "gibbs.r" in the same directory where you are going to start R . Once you start an R session, simply do

```
> source("gibbs.r")
```

and you will have the function `gibbs` available for use. Note that when calling the function `gibbs`, the argument `data` (the vector of observations $\mathbf{y}$) always needs to be supplied, whereas the function has as default the values $\mu_0 = 0$, $\kappa_0 = \alpha_0 = \lambda_0 = 0.1$, and a sampler with burn-in period of 0 iterations and with 5,000 iterations after burn-in. All these values can be changed when calling the function.

```
> gibbs
function (data,mu0=0,kappa0=0.1,alpha0=0.1,lambda0=0.1,nburn=0,ndraw=5000)
{

#Gibbs sampling for model:
#indep observ from N(mu,1/omega)
#pi(mu,omega)=N(mu|mu0,1/kappa0)Gamma(omega|alpha0,lambda0)

n <- length(data)
sumy <- sum(data)
alpha1 <- alpha0 + (n/2)

#initial values:drawn from prior

mu <- rnorm(1,mu0,sqrt(1/kappa0))
omega <- rgamma(1,alpha0,1)/lambda0

# matrix that will contain recorded draws:

draws <- matrix(ncol=2,nrow=ndraw)

# MCMC LOOP FOLLOWS:

it <- -nburn
while(it < ndraw){ it <- it+1;

# draw mu:
var <- 1/(kappa0+omega*n)
mu1 <- (kappa0*mu0 + omega*sumy)*var
mu <- rnorm(1,mu1,sqrt(var))

# draw omega:
omega <- rgamma(1,alpha1,1)/(lambda0 + (sum((data - mu)^2))/2 )

# after burn-in record mu and omega:
if(it>0){
draws[it,1] <- mu
draws[it,2] <- omega
}

}

# END MCMC LOOP

return(draws)
```

```
}
```

Now let us generate a vector of observations $\mathbf{y}$ consisting of 50 independent realizations from a Normal distribution with $\mu = 5$ and $\omega = 0.2$ (so the variance is $(1/\omega) = 5$).

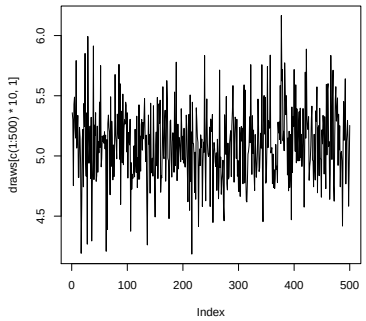
```
> y <- rnorm(50,5,sqrt(5))
> y
 [1]  5.469907  3.436438  3.557772  2.786530  7.440229  9.278197  7.318239
 [8]  4.424508  4.603065  3.582908 10.083979  5.700496  9.654838  5.849463
[15]  2.961325  4.230260  3.224884  5.713982  7.784721  5.669146  6.424668
[22]  4.950342  3.886849  5.883722  3.067205  8.180000  1.616655  5.338903
[29]  3.750824  5.132919  4.637062  1.907974  2.902405  5.441778  7.955641
[36]  5.739371  6.757808  6.482835  7.756410  4.883539  6.851959  7.179380
[43]  5.566471  4.132660  8.046997  5.156700  4.908252  6.898696  2.857113
[50] -2.252454
```

Now apply the function `gibbs` to the data-set $\mathbf{y}$:

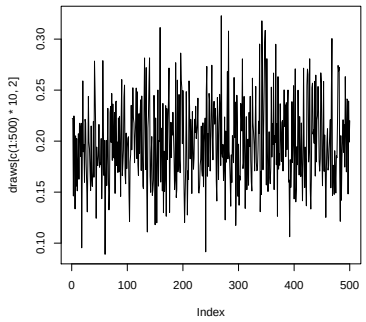
```
> draws <- gibbs(data=y,mu0=3,kappa0=1,alpha0=0.1,lambda0=0.1,nburn=0,ndraw=5000)
```

The draws from $\pi(\mu, \omega|\mathbf{y})$ have been stored in a $5,000 \times 2$ matrix called `draws`. The first column contains the draws of μ and the second column the draws of ω. To visualize these draws, we can plot, for example, 500 equally spaced draws (out of the 5,000) for μ and similarly for ω.

```
> plot(draws[c(1:500)*10,1],type="l")
```

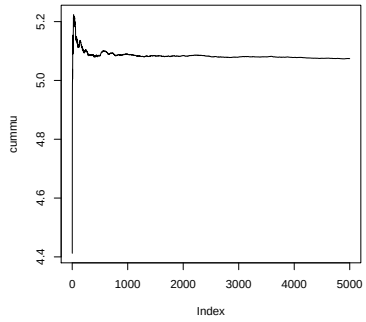



```
> plot(draws[c(1:500)*10,2],type="l")
```



An informal way to get an idea of the convergence of the sampler is to plot the posterior mean of the parameters against the number of iterations of the sampler. If convergence is achieved, the mean should settle around a certain value (of course, this is provided the mean exists, which is not guaranteed in general). We do this for μ and ω:

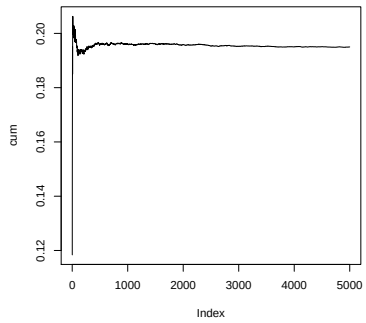
```
> cum <- cumsum(draws[,1])/c(1:5000)
> plot(cum,type="l")
```



Note that the posterior mean for μ has settled at a value that is in between its prior mean $\mu_0 = 3$ and the sample mean, which is:

```
> mean(y)
[1] 5.296271
```

```
> cum <- cumsum(draws[,2])/c(1:5000)
> plot(cum,type="l")
```

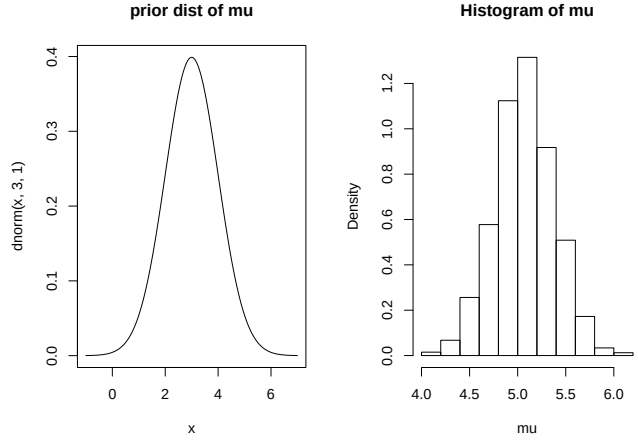


All these plots suggest that convergence has been achieved after about 1,000 iterations of the sampler. So we remove the first 1,000 draws (the burn-in) and keep the remaining 4,000 to conduct inference and prediction.

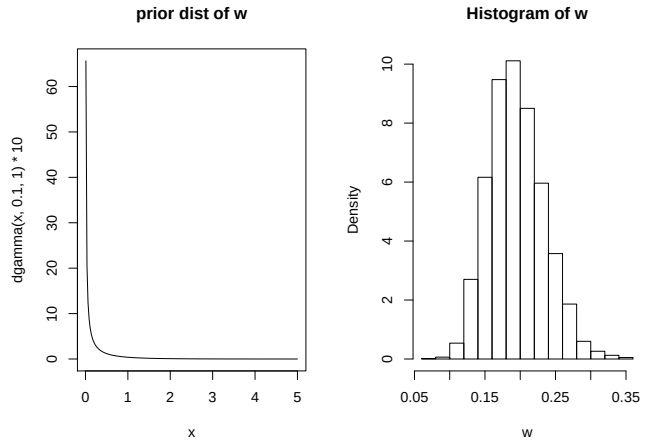
```
> mu <- draws[c(1001:5000),1]
> omega <- draws[c(1001:5000),2]
```

The following plots display the prior and posterior distributions of μ (first 2 plots) and ω (last 2 plots):

```
> par(mfrow=c(1,2))
> x <- seq(-1,7,length=200)
> plot(x,dnorm(x,3,1),type="l")
> title("prior dist of mu")
> hist(mu,probab=T)
```




```
> x <- seq(0.01,5,length=200)
> plot(x,dgamma(x,0.1,rate=0.1),type="l")
> title("prior dist of w")
> hist(omega,probab=T)
```

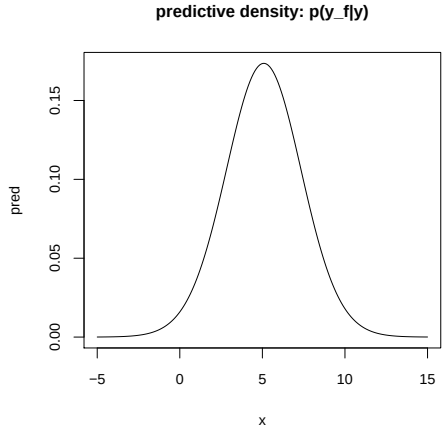


Now suppose we want to forecast an observable y_f. This is done according to the out-of-sample predictive distribution, which has density

$$f(y_f|\mathbf{y}) = \int f(y_f|\mu,\omega)\pi(\mu,\omega|\mathbf{y})d\mu d\omega \approx \frac{1}{J}\sum_{j=1}^{J}\left(\frac{\omega^{(j)}}{2\pi}\right)^{1/2}\exp\left\{-\frac{\omega^{(j)}}{2}(y_f-\mu^{(j)})^2\right\}$$

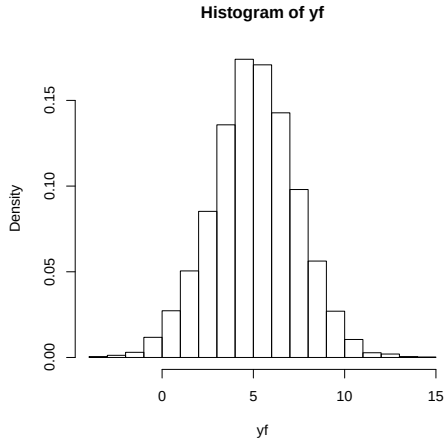
where $(\mu^{(j)},\omega^{(j)})$ are the draws obtained through the Gibbs sampler. To do this with R :

```
> x <- seq(-5,15,length=200)
> pred <- numeric(200)
> for(j in 1:200) pred[j] <- mean(dnorm(x[j],mu,1/sqrt(omega)))
> plot(x,pred,type="l")
> title("predictive density: p(y_f|y)")
```



If we want to compute some other characteristic of the predictive distribution, *e.g.* the probability that a future observable lies in the interval $(0,5)$, we can take a sample of draws from the predictive distribution. So, for each value $(\mu^{(j)},\omega^{(j)})$ drawn in the Gibbs sampler, we draw a value of y_f from a $N(\mu^{(j)},1/\omega^{(j)})$ distribution. In R :

```
> yf <- rnorm(4000,mu,1/sqrt(omega))
> pos <- yf[yf>0]
> length(pos[pos<5])/length(yf)
[1] 0.47275
> hist(yf,probab=T)
```

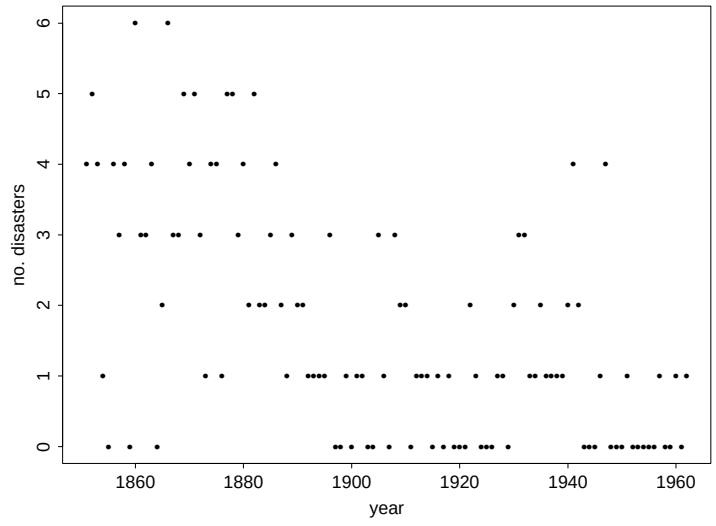


Clearly, the histogram based on 4000 draws from the predictive distribution looks very similar to its density function (plotted in previous figure).

3.5 Another example: a Poisson count change point problem

To illustrate further the use of Gibbs sampling in Bayesian statistics we shall look at an example concerning the change point in a Poisson process. The time series stored in the file "coal.dat" gives the number of British coal mining disasters per year, over the period 1851 – 1962. The following extract from an R session, reads the data into the session and plots them:

```
> coal <- scan("coal.dat")
> plot(c(1851:1962),coal,xlab="year",ylab="no. disasters")
```



From this plot it does seem to be the case that there has been a reduction in the rate of disasters over the period. Let Y_i = denote the number of disasters in year i, $i = 1, \ldots, 112$ (relabelling the years by numbers from 1 to $n = 112$). A model that has been proposed in the literature has the form

$$Y_i \sim \text{Poisson } (\theta); \quad i = 1, \ldots k; \tag{3.11}$$

$$Y_i \sim \text{Poisson } (\lambda); \quad i = k+1, \ldots n. \tag{3.12}$$

Thus, this model assumes a Poisson distribution with rate θ for the number of disasters per year up to the kth year (where k is unknown), and a Poisson distribution with rate λ thereafter.

The Bayesian specification of the model is completed in a hierarchical framework: $\theta \sim \text{Gamma } (a_1, b_1)$, $\lambda \sim \text{Gamma } (a_2, b_2)$, $k \sim$ discrete uniform over $\{1, \ldots, n\}$, each independent of one another, and then $b_1 \sim \text{Gamma } (c_1, d_1)$ and $b_2 \sim \text{Gamma } (c_2, d_2)$.

As usual, the posterior density is obtained (up to a constant of proportionality) as the product of the likelihood and the prior. This leads to

$$\begin{aligned} \pi(\theta, \lambda, k, b_1, b_2 | \mathbf{y}) &\propto e^{-\theta k} \theta^{\sum_{i=1}^{k} y_i} e^{-\lambda(n-k)} \lambda^{\sum_{i=k+1}^{n} y_i} \\ &\times b_1^{a_1} \theta^{a_1 - 1} e^{-b_1 \theta} b_1^{c_1 - 1} e^{-d_1 b_1} b_2^{a_2} \lambda^{a_2 - 1} e^{-b_2 \lambda} b_2^{c_2 - 1} e^{-d_2 b_2} I[k \in \{1, 2, \ldots, n\}] \end{aligned}$$

The conditional posterior distribution of each parameter is obtained from the last expression by picking those factors which involve that parameter. Therefore, the conditionals are as follows:

$$\theta | \mathbf{y}, \lambda, b_1, b_2, k \sim \text{Gamma } \left(a_1 + \sum_{i=1}^{k} y_i, b_1 + k \right) \tag{3.13}$$

$$\lambda | \mathbf{y}, \theta, b_1, b_2, k \sim \text{Gamma } \left(a_2 + \sum_{i=1}^{n} y_i - \sum_{i=1}^{k} y_i, b_2 + n - k \right) \tag{3.14}$$

$$b_1 | \mathbf{y}, \theta, \lambda, b_2, k \sim \text{Gamma } (c_1 + a_1, d_1 + \theta) \tag{3.15}$$

$$b_2 | \mathbf{y}, \theta, \lambda, b_1, k \sim \text{Gamma } (c_2 + a_2, d_2 + \lambda) \tag{3.16}$$

and

$$P(k | \mathbf{y}, \theta, \lambda, b_1, b_2) = \frac{e^{(\lambda - \theta)k} \ (\theta / \lambda)^{\sum_{i=1}^{k} y_i}}{\sum_{j=1}^{n} \left\{ e^{(\lambda - \theta) j} \ \left(\frac{\theta}{\lambda} \right)^{\sum_{i=1}^{j} y_i} \right\}} \ I[k \in \{1, 2, \ldots, n\}]. \tag{3.17}$$

Note that the conditional distribution of k is discrete, and is therefore characterized by a probability mass function. The corresponding Gibbs sampler is implemented in R with the following code:

```
> gibbs2
function (data,a1=0.5,a2=0.5,c1=1,d1=1,c2=1,d2=1,nburn=0,ndraw=1000)
{

#Gibbs sampling for Poisson change point problem

n <- length(data)
sum1 <- c1 + a1
sum2 <- c2 + a2
sumy <- sum(data)

p <- numeric(n)

#initial values:drawn from prior

b1 <- rgamma(1,c1,1)/d1
b2 <- rgamma(1,c2,1)/d2
theta <- rgamma(1,a1,1)/b1
```

```
lambda <- rgamma(1,a2,1)/b2
k <- sample(c(1:n),size=1)

# create matrix to record draws:

draws <- matrix(ncol=5,nrow=ndraw)


# MCMC LOOP FOLLOWS:

it <- -nburn
while(it < ndraw){ it <- it+1;

sumyk <- sum(data[c(1:k)])

# draw theta:
theta <- rgamma(1,a1+sumyk,1)/(b1+k)

# draw lambda:
lambda <- rgamma(1,a2+sumy-sumyk,1)/(b2+n-k)

# draw b1:
b1 <- rgamma(1,sum1,1)/(d1+theta)

# draw b2:
b2 <- rgamma(1,sum2,1)/(d2+lambda)

# draw k:
# first create the vector of probabilities:
aux1 <- lambda-theta
aux2 <- theta/lambda
for(j in 1:n){
sumyj <- sum(data[c(1:j)])
p[j] <- (exp(aux1*j))*(aux2**sumyj)
}
p <- p/sum(p)
# now sample k according to probabilities p:
k <- sample(c(1:n),size=1,prob=p)

# after bun-in record draws:
if (it>0){
draws[it,1] <- theta
draws[it,2] <- lambda
draws[it,3] <- b1
draws[it,4] <- b2
draws[it,5] <- k
}

}

# END MCMC LOOP

return(draws)
```

```
}
```

This function is stored in the external file "gibbs2.r" (which you can source into an R session as explained before). To run it with 1,100 iterations and no burn-in I did

```
> draws <- gibbs2(data=coal,ndraw=1100)
```

and this led to the output in Figure 3.1. The code to obtain this figure is

```
> par(mfrow=c(3,2))
> plot(draws[,1],type="l",main="draws theta")
> plot(draws[,2],type="l",main="draws lambda")
> plot(draws[,3],type="l",main="draws b1")
> plot(draws[,4],type="l",main="draws b2")
> plot(draws[,5],type="l",main="draws k")
```

Convergence of the algorithm seems rapid, so I deleted the first 100 values and based the subsequent analysis on the remaining 1000 points:

```
> theta <- draws[c(101:1100),1]
> lambda <- draws[c(101:1100),2]
> k <- draws[c(101:1100),5]
```

A histogram of the posterior distribution of k is given in the top left panel of Figure 3.2. On the basis of this estimate, it is almost certain that a changepoint has occurred, with the posterior mode estimate being $k = 41$, corresponding to a changepoint at the year 1891.

Histograms of the posterior distributions of θ and λ are also given in Figure 3.2. It is clear from this figure that λ is (almost certainly) less than θ. By looking at the sequence of realisations of $\theta - \lambda$, we obtain a sample from the marginal posterior distribution of $\theta - \lambda$. No complicated transformation theory is needed because we are working directly with MCMC the sample. The bottom-right panel of Figure 3.2 presents a histogram of the marginal posterior distribution of $\theta - \lambda$.

The R code for Figure 3.2 is as follows:

```
> par(mfrow=c(2,2))
> hist(k,breaks=seq(min(k)-0.5,max(k)+0.5,by=1),probab=T)
> hist(theta,probab=T)
> hist(lambda,probab=T)
> hist(theta-lambda,probab=T)
```

3.6 Prediction

Here we return to the precision issue. As already discussed, the predictive distribution of a future observation y_f given the current observations $\mathbf{y}$ is defined as

$$f(y_f|\mathbf{y}) = \int f(y_f|\theta)\pi(\theta|\mathbf{y})d\theta \tag{3.18}$$

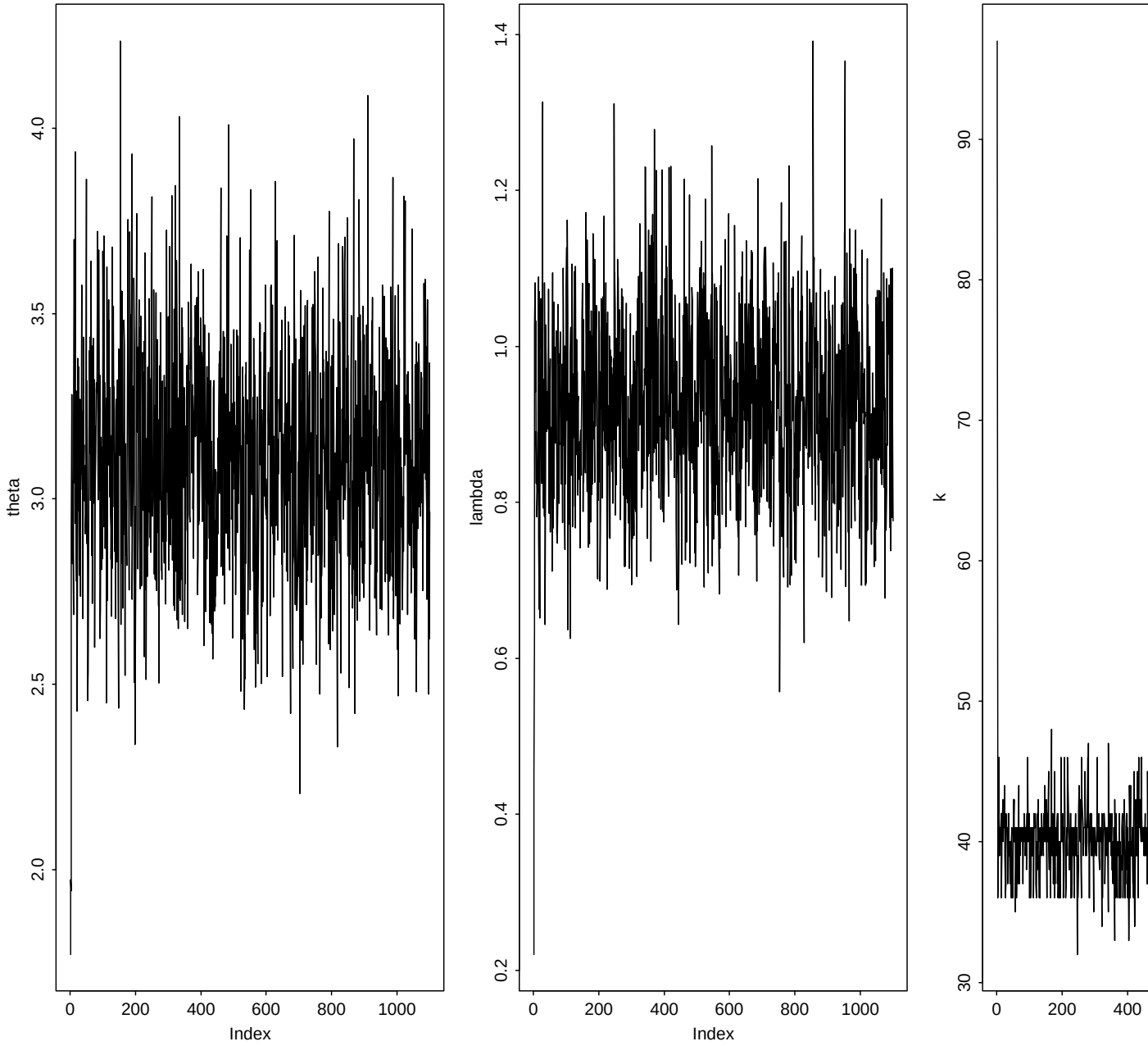


Figure 3.1: Gibbs sample of Poisson changepoint model parameters

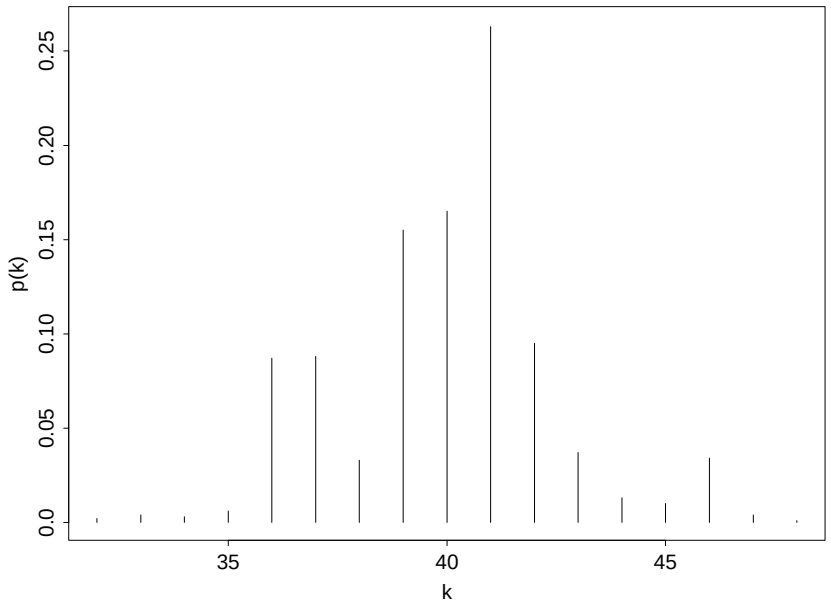


Figure 3.2: Histograms of marginal posterior distributions

(under independent sampling), so that the likelihood, $f(y_f|\theta)$ is averaged across the uncertainty in θ contained in the posterior distribution $\pi(\theta|\mathbf{y})$. Hence, given a sampled sequence of realizations $\theta^{(1)}, \ldots, \theta^{(J)}$ from this posterior, we can estimate

$$f(y_f|\mathbf{y}) \approx \frac{1}{J}\sum_{j=1}^{J} f(y_f|\theta^{(j)}) \tag{3.19}$$

For the coal mining example, we can use this estimator to estimate the predictive distribution of the number of disasters in a future year for which the Poisson rate is λ. A graph of the predictive distribution is given in Figure 3.3, together with an estimate which is simply based on the posterior mean. In this case, the fact that there is not much variation in the posterior distribution of λ means that the two distributions are almost identical. However, in general, the predictive distribution is likely to be flatter as it takes into account the uncertainty in the posterior distribution of λ.

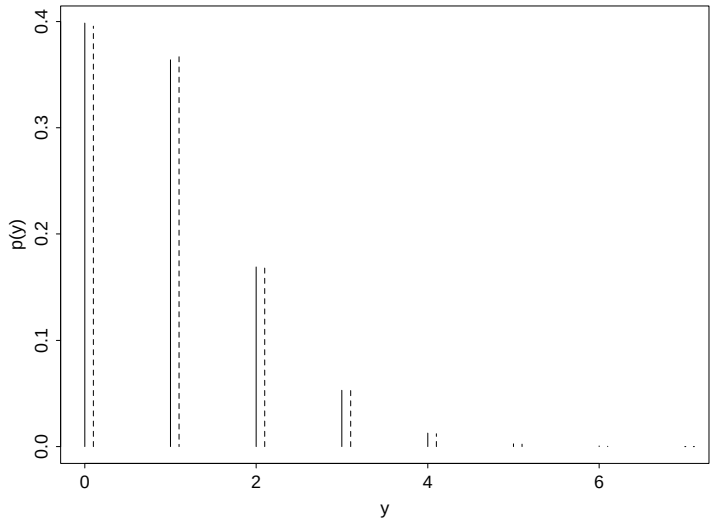


Figure 3.3: Predictive and estimative distributions of number of coal mining disasters in a future year (solid line is predictive)

Chapter 4

Data Augmentation

Data augmentation is a technique that can be helpful in making problems amenable to Gibbs sampling. Basically, it consists of augmenting the parameters with additional random variables in a way that leads to a model that is easier to handle computationally (despite the increase in the number of variables that need to be sampled a posteriori). So, in some sense, it can be thought of as a "trick" that can be applied in certain cases. Implementing data augmentation requires skill and experience, part of the skill being in recognizing when and how it can be used successfully. It can be applied equally well to the following two situations:

1. One of the main problems that afflicts statistical computing is that in the real world, some of our data could be missing. In theory, the distribution of the observed data can be obtained by integrating out the missing data. However this is frequently difficult if not impossible to do.
2. The likelihood function is not tractable for one reason or another (for instance there is no simple conjugate prior) but conditional on a collection of unobserved random variables, the likelihood becomes easy to handle.

These two situations are best illustrated using simple examples.

1. Suppose $Y_0, Y_1, \ldots, Y_n$ is a time series of random variables defined by $Y_0 = 0$ and for each $i = 1, \ldots, n$, $Y_i = Y_{i-1} + S_i$, where $S_i \sim Beta(\theta, \theta)$, $\theta > 0$. Therefore,

$$Y_i | Y_0, Y_1, \ldots Y_{i-1} \sim Y_{i-1} + S_i. \tag{4.1}$$

   Then, given the observations $(y_0, y_1, \ldots, y_n)$, the likelihood of θ can be easily obtained as

$$\begin{aligned} f(y_0, \ldots, y_n|\theta) &= f(y_0|\theta) \prod_{i=1}^{n} f(y_i|y_0, \ldots, y_{i-1}, \theta) = \prod_{i=1}^{n} f(y_i|y_{i-1}, \theta) \\ &= \prod_{i=1}^{n} \frac{\Gamma(2\theta)}{\{\Gamma(\theta)\}^2} (y_i - y_{i-1})^{\theta-1} \{1 - (y_i - y_{i-1})\}^{\theta-1} I[0 < y_i - y_{i-1} < 1]. \end{aligned}$$

   However suppose that an observation y_{i^*} is missing, then the likelihood is no longer available in closed form. How can we handle this situation?

2. Suppose that $Y_1, \ldots Y_n$ are i.i.d. data from the *mixture* density

$$f(y_i|\theta) = \frac{1}{2} \left(\frac{1}{(2\pi)^{1/2}} e^{-y_i^2/2} + \frac{1}{(2\pi)^{1/2}} e^{-(y_i - \theta)^2/2} \right) . \tag{4.2}$$

This mixture density can be thought of as the density of a random variable whose values are obtained by the following procedure. Toss a fair coin. Given a head, sample y_i from a $N(0,1)$ distribution. Given a tail, sample y_i from a $N(\theta,1)$ distribution.

With n independent observations, the likelihood of θ is given by:

$$f(\mathbf{y}|\theta) = \prod_{i=1}^{n} f(y_i|\theta) \propto \prod_{i=1}^{n} \left(e^{-y_i^2/2} + e^{-(y_i-\theta)^2/2} \right) . \tag{4.3}$$

If we want to compute the posterior distribution of θ, this likelihood function is difficult to handle (note that this function is also difficult to maximise for classical inference). If we multiply the n factors term by term, we end up with a sum of 2^n terms, so that computation soon becomes almost impossible as n increases.

Now, suppose I know the sequence of heads and tails that generated the data. In other words, suppose I know which observations came from the $N(0,1)$ distribution (and therefore are irrelevant to inference about θ) and which observations came from the $N(\theta,1)$ distribution (and, thus, contain information about θ). Then the inference problem becomes straightforward (both in a Bayesian and in a classical framework).

In these examples exact inference is not possible, both in a Bayesian and in a classical framework. Instead, we need to use Computationally intensive methods to deal with these problems. It turns out that the numerical techniques we use within the two different approaches to inference are highly related. You have already studied the EM algorithm for maximum likelihood inference in the course "Computationally Intensive Methods I". Now we shall explore related ideas within a Bayesian context.

First we remark that the distinction between data and parameters is somewhat blurred in a Bayesian setting: both are treated as random variables. When doing data augmentation, we add further random variables to the model. These can be viewed as data or parameters depending on the context, but the interpretation is not really relevant for the workings of the method. We shall denote by z the additional variables to be included in the model (z may be just a single variable or a vector containing several variables). As we have done throughout the course, θ denotes the (original) parameters in the model with prior $\pi(\theta)$, and $\mathbf{y}$ is the vector of observations. Then the above examples demonstrate the situation where $f(\mathbf{y}|\theta)$ is not tractable, but $f(\mathbf{y}, z|\theta)$ is easy to handle. As a result, the posterior distribution of (θ, z) is proportional to

$$\pi(\theta, z|\mathbf{y}) \propto f(\mathbf{y}, z|\theta)\pi(\theta). \tag{4.4}$$

Data augmentation proceeds by carrying out Gibbs sampling to sample successively from θ and z to produce a sample from this joint distribution. The marginal distribution of θ is therefore the posterior distribution of interest (in some cases, the variable z may also be of interest).

How does this work in the two examples above?

1. In this case, $\mathbf{y}$ is the vector $(y_0, \ldots, y_n)$ excluding y_{i^*}, the missing observation, and $z = y_{i^*}$. Thus, $(\mathbf{y}, z)$ reduces to the complete data vector and we obtain

$$f(\mathbf{y}, y_{i^*}|\theta) = f(y_0, \ldots, y_n|\theta) \propto \prod_{i=1}^{n} \frac{\Gamma(2\theta)}{\{\Gamma(\theta)\}^2} (y_i - y_{i-1})^{\theta-1} \{1-(y_i-y_{i-1})\}^{\theta-1} I[0 < y_i - y_{i-1} < 1]$$

Therefore, the posterior density of θ given $(\mathbf{y}, y_{i^*})$ is explicit

$$\pi(\theta|\mathbf{y}, y_{i^*}) \propto f(\mathbf{y}, y_{i^*}|\theta)\pi(\theta)$$

and can be sampled easily. To complete the Gibbs sampler, we also need to sample from the conditional posterior distribution of y_{i^*}. This has density

$$f(y_{i^*}|\mathbf{y},\theta) \propto f(\mathbf{y}, y_{i^*}|\theta) \propto \Big[(y_{i^*} - y_{i^*-1})\{1-(y_{i^*} - y_{i^*-1})\}(y_{i^*+1} - y_{i^*})\{1-(y_{i^*+1} - y_{i^*})\}\Big]^{\theta-1}$$

on the region $y_{i^*} \in (y_{i^*-1}, y_{i^*-1}+1) \cap (y_{i^*+1}-1, y_{i^*+1})$. This sampling can be carried out (among other ways) by rejection sampling.

2. Here z is a sequence of n heads or tails, with one element per observation. Hence, $z = (z_1, \ldots, z_n)$ and z_i equals 1 if observation i corresponds to head and 2 if it corresponds to tail. Suppose that the prior for θ is $N(0,1)$. Then we have

$$f(y_i, z_i|\theta) = f(y_i|z_i,\theta)P(z_i),$$

where

$$f(y_i|z_i,\theta) = \begin{cases} \frac{1}{(2\pi)^{1/2}} \exp\left(-\frac{y_i^2}{2}\right) & \text{if } z_i = 1 \\ \frac{1}{(2\pi)^{1/2}} \exp\left(-\frac{(y_i-\theta)^2}{2}\right) & \text{if } z_i = 2 \end{cases} \quad \text{and} \quad P(z_i = 1) = P(z_i = 2) = \frac{1}{2}.$$

Using that

$$\pi(\theta, z|\mathbf{y}) \propto \left\{\prod_{i=1}^{n} f(y_i|z_i,\theta)P(z_i)\right\}\pi(\theta),$$

it follows that

$$\pi(\theta|\mathbf{y}, z) \propto \exp\Big\{-\frac{1}{2}\sum_{i:z_i=2}(y_i-\theta)^2\Big\}\exp\Big(-\frac{\theta^2}{2}\Big) \equiv N\left(\frac{\sum_{i:z_i=2} y_i}{1+n_2}, \frac{1}{1+n_2}\right),$$

where n_2 is the number of observations for which $z_i = 2$, and that, for each i,

$$P(z_i = 2|\theta,\mathbf{y}) = \frac{e^{-(y_i-\theta)^2/2}}{e^{-(y_i-\theta)^2/2} + e^{-y_i^2/2}}, \quad P(z_i = 1|\theta,\mathbf{y}) = \frac{e^{-y_i^2/2}}{e^{-(y_i-\theta)^2/2} + e^{-y_i^2/2}}.$$

Hence, it is easy to implement a Gibbs sampler to simulate the posterior distribution of $(\theta, z_1, \ldots, z_n)$.

4.1 A genetic linkage example

We will now look at a genetic linkage example. This example is well–worn, but is useful because it illustrates very simply all the necessary ideas for data augmentation algorithms. The example concerns genetic linkage of 197 animals. The animals are distributed into 4 categories:

$$\mathbf{y} = (y_1, y_2, y_3, y_4) = (125, 18, 20, 34) \tag{4.5}$$

with cell probabilities

$$\left(\frac{2+\theta}{4}, \frac{1}{4}(1-\theta), \frac{1}{4}(1-\theta), \frac{\theta}{4}\right). \tag{4.6}$$

Since the cell probabilities must all be non-negative, the above implies $0 \le \theta \le 1$. Suppose that the prior distribution of θ is Uniform$(0,1)$. The posterior density of θ is

$$\pi(\theta|\mathbf{y}) \propto f(\mathbf{y}|\theta)\pi(\theta) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4} I[\theta \in (0,1)].$$

Though it is possible to sample the posterior distribution of θ directly (e.g. via rejection sampling), data augmentation brings about a substantial simplification:

Augment the observed data (y_1, y_2, y_3, y_4) by dividing the first cell into two portions, with respective probabilities proportional to θ and 2. In other words, consider

$$z|\mathbf{y}, \theta \sim \text{Binomial}\left(y_1, \frac{\theta}{2+\theta}\right). \tag{4.7}$$

This gives the augmented data set $(\mathbf{y}, z)$, for which we have that

$$f(\mathbf{y}, z|\theta) = f(\mathbf{y}|\theta)\pi(z|\mathbf{y}, \theta) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}\binom{y_1}{z}\left(\frac{\theta}{2+\theta}\right)^z\left(\frac{2}{2+\theta}\right)^{y_1-z} \tag{4.8}$$

Now it is immediate that

$$\pi(\theta|\mathbf{y}, z) \propto \theta^{z+y_4}(1-\theta)^{y_2+y_3}I[\theta \in (0,1)] \equiv Beta(z+y_4+1, y_2+y_3+1). \tag{4.9}$$

To complete the Gibbs sampler we also need to generate draws from the conditional posterior distribution of z, which is given in (3.7).

This Gibbs sampler is easily coded in R as

```
> gibbs3
function (data,nburn=0,ndraw=1000)
{

#Gibbs sampling for genetic linkage model:

#initial values:

z <- 20
theta <- 0.5

par2 <- data[2]+data[3]+1

# matrix to record draws:

draws <- matrix(ncol=2,nrow=ndraw)

# MCMC LOOP FOLLOWS:

it <- -nburn
while(it < ndraw){ it <- it+1;

# draw theta:
theta <- rbeta(1,z+data[4]+1,par2)

# draw z:
z <- rbinom(1,data[1],theta/(theta+2))

# after burn-in record theta and z:
if(it>0){
draws[it,1] <- theta
draws[it,2] <- z
```

```
}

}

# END MCMC LOOP

return(draws)

}
```

Note that the initial values $z = 20$ and $\theta = 0.5$ are chosen arbitrarly, with the restriction that z is an integer between 0 and $y_1 = 125$ and that $0 \leq \theta \leq 1$. Applying this algorithm with `ndraw=600` iterations and no burn-in gave the output in Figure 4.1.

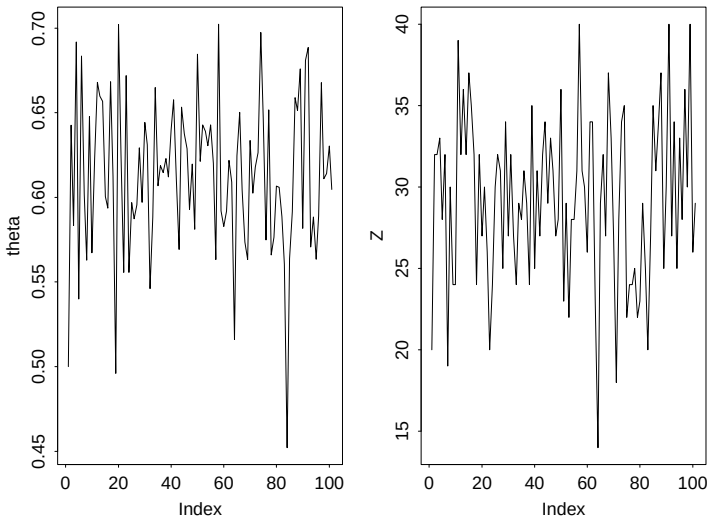


Figure 4.1: Realizations from data augmentation in genetic example

It seems that convergence of the algorithm is very fast, though there is some evidence that the chain behaves slightly differently at the start. Consequently, I have elected to disregard the first 100 realisations (this is the burn–in). The remaining 500 realisations are regarded as a sample from the required posterior distribution $\pi(\theta, z|\mathbf{y})$. In particular, histograms of the separate θ and z components (Figure 4.2) are estimates of the posterior distributions $\pi(\theta|\mathbf{y})$ and $\pi(z|\mathbf{y})$. Similarly, a plot of the (θ, z) pairs (Figure 4.3) gives us an idea of the joint posterior $\pi(\theta, z|\mathbf{y})$.

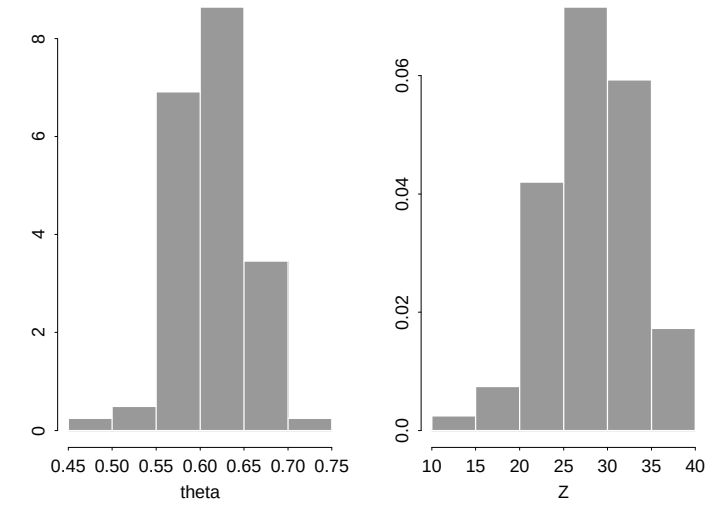


Figure 4.2: Marginal posteriors in genetic example

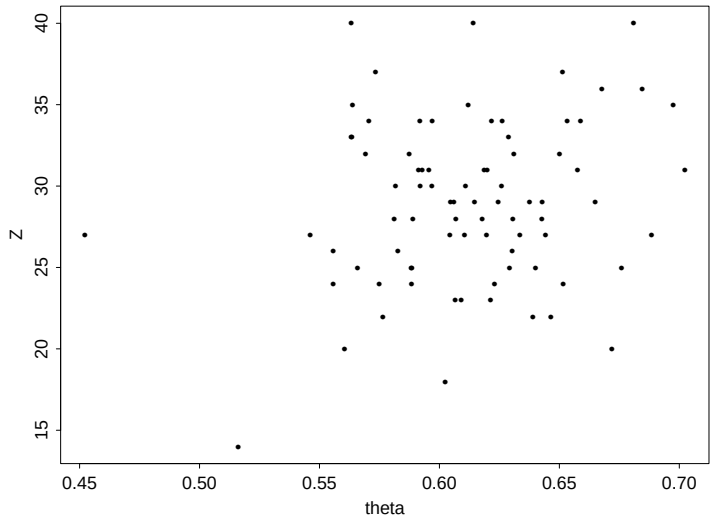


Figure 4.3: Scatterplot of (θ, z) pairs in genetic example

Interpretation of these results does need care, since the sequence of realisations are not (by definition) independent. There are two alternative strategies:

1. Run a single chain for sufficiently long, so that dependence between successive realisations does not diminish the precision of the sample information; and
2. Run several chains, and average across them.

I'm going to stick with the first of these approaches, though it should also be recognised that approaches to assessing convergence of such chains are often based on comparisons of within–chain and across–chain variability.

Chapter 5

The Metropolis–Hastings Algorithm

5.1 The general MCMC algorithm

The Gibbs sampler is the best publicised MCMC algorithm, but there are many others. It is not possible to use Gibbs sampling in all problems since sometimes the conditional posterior distributions can not be calculated. To illustrate this, consider again Example 3.1 but instead of Normal sampling suppose that the sampling model is a Cauchy distribution (note: the Cauchy distribution is a generalization of the Normal distribution that allows for thicker tails). Here are the details:

Example 5.1. *Suppose* $Y_i|\mu,\omega \sim \text{Cauchy}(\mu, 1/\omega)$ *independently for* $i = 1,\ldots,n$. *Hence, the sampling density is*

$$f(\mathbf{y}|\mu,\omega) = \prod_{i=1}^{n} f(y_i|\mu,\omega) = \prod_{i=1}^{n} \frac{\omega^{1/2}}{\pi} \frac{1}{1+\omega(y_i-\mu)^2}$$

Suppose also we have prior distributions for μ *and* ω*:*

$$\mu \sim N(\mu_0, 1/\kappa_0) \quad \text{and} \quad \omega \sim \text{Gamma}\,(\alpha_0, \lambda_0), \tag{5.1}$$

where μ *and* ω *are considered to be a priori independent and* μ_0, κ_0, α_0 *and* λ_0 *are considered to be known hyperparameters. The posterior density of* (μ,ω) *is:*

$$\pi(\mu,\omega|\mathbf{y}) \propto \left\{ \prod_{i=1}^{n} \frac{1}{1+\omega(y_i-\mu)^2} \right\} e^{-\frac{\kappa_0}{2}(\mu-\mu_0)^2} \omega^{\frac{n}{2}+\alpha_0-1} e^{-\lambda_0\omega} I[\omega>0]. \tag{5.2}$$

Clearly, the posterior distribution is a complex two-dimensional distribution. If we now look at the conditional posterior densities of μ *and* ω *we obtain:*

$$\pi(\mu|\omega,\mathbf{y}) \propto \left\{ \prod_{i=1}^{n} \frac{1}{1+\omega(y_i-\mu)^2} \right\} e^{-\frac{\kappa_0}{2}(\mu-\mu_0)^2}, \tag{5.3}$$

and

$$\pi(\omega|\mu,\mathbf{y}) \propto \left\{ \prod_{i=1}^{n} \frac{1}{1+\omega(y_i-\mu)^2} \right\} \omega^{\frac{n}{2}+\alpha_0-1} e^{-\lambda_0\omega}\, I[\omega>0]. \tag{5.4}$$

None of these distributions has a well-known form, so Gibbs sampling is precluded. In order to compute the posterior distribution in this model, we need to use more general MCMC algorithms.

The previous example illustrates the need for MCMC algorithms that are more general than Gibbs sampling. Later in this chapter, we shall study the most important of these algorithms, namely the Metropolis-Hastings algorithm. But before doing that, I want to stress that **all** MCMC algorithms to sample from a target distribution $\pi(\theta)$ follow the general scheme:

Algorithm 5.1. *General MCMC Algorithm:*

1. *Break the components of θ into d groups $\theta_1, \ldots, \theta_d$, where each group θ_j has dimension ≥ 1.*
2. *Initialize with $\theta_1^{(0)}, \ldots, \theta_d^{(0)}$.*
3. *Update $\theta_1^{(0)}$ to $\theta_1^{(1)}$ "according to" the conditional distribution $\pi(\theta_1|\theta_2^{(0)}, \theta_3^{(0)} \ldots, \theta_d^{(0)})$.*
4. *Update $\theta_2^{(0)}$ to $\theta_2^{(1)}$ "according to" the conditional distribution $\pi(\theta_2|\theta_1^{(1)}, \theta_3^{(0)}, \ldots, \theta_d^{(0)})$.*
5. *...*
6. *Update $\theta_d^{(0)}$ to $\theta_d^{(1)}$ "according to" the conditional distribution $\pi(\theta_d|\theta_1^{(1)}, \theta_2^{(1)}, \ldots, \theta_{d-1}^{(1)})$.*
7. *Iterate this updating procedure.*

After discarding an initial number of draws (the burn-in), the remaining draws can be regarded as a sample from the target distribution $\pi(\theta)$ (under suitable regularity conditions as explained in the Appendix).

Note that in the updating steps of Algorithm 4.1 we *"update according to"* the appropriate conditional distribution. For Gibbs sampling, we have seen that this means *simulating from* the conditional distribution, but for other MCMC algorithms *"update according to"* means something else. The most general algorithm in this context is the Metropolis–Hastings algorithm, which we describe in the following section.

5.2 The Metropolis–Hastings algorithm

Suppose that we are running a general MCMC algorithm as described in the previous section. Further, suppose that the current value of the chain is $\theta_1^{(j)}, \ldots, \theta_d^{(j)}$ and that we now want to simulate $\theta_1^{(j+1)}$, the next value of θ_1. From the general scheme, this means that we now need to update $\theta_1^{(j)}$ to $\theta_1^{(j+1)}$ *"according to"* the conditional distribution $\pi(\theta_1|\theta_2^{(j)}, \ldots, \theta_d^{(j)})$. In the Metropolis-Hastings algorithm this is done as follows:

Metropolis-Hastings updating mechanism:

- *Propose a candidate value θ_1^{can}, which is a draw from an arbitrary distribution with density $q(\theta_1^{can}|\theta_1^{(j)}, \theta_2^{(j)}, \ldots, \theta_d^{(j)})$.*
- *Take as the next value of θ_1 in the chain*

$$\theta_1^{(j+1)} = \begin{cases} \theta_1^{can} & \text{with probability } p \\ \theta_1^{(j)} & \text{with probability } 1-p \end{cases}$$

where

$$p = \min\left\{1,\ \frac{\pi(\theta_1^{can}|\theta_2^{(j)},\ldots,\theta_d^{(j)})}{\pi(\theta_1^{(j)}|\theta_2^{(j)},\ldots,\theta_d^{(j)})}\ \frac{q(\theta_1^{(j)}|\theta_1^{can},\theta_2^{(j)},\ldots,\theta_d^{(j)})}{q(\theta_1^{can}|\theta_1^{(j)},\theta_2^{(j)},\ldots,\theta_d^{(j)})}\right\}, \tag{5.5}$$

with $\pi(\theta_1^{can}|\theta_2^{(j)},\ldots,\theta_d^{(j)})$ *denoting the density corresponding to the conditional posterior distribution of* θ_1 *evaluated at* $\theta_1 = \theta_1^{can}$ *and similarly for* $\pi(\theta_1^{(j)}|\theta_2^{(j)},\ldots,\theta_d^{(j)})$.

Some comments:

• In practice, the way to implement the second part of the Metropolis-Hastings updating mechanism above is by drawing a value u from a Uniform$(0,1)$ distribution, and taking $\theta_1^{(j+1)} = \theta_1^{can}$ if $u < p$ and $\theta_1^{(j+1)} = \theta_1^{(j)}$ otherwise.

• The *candidate generator* $q(\theta_1^{can}|\theta_1^{(j)},\theta_2^{(j)},\ldots,\theta_d^{(j)})$ is arbitrary so, in principle, any choice should work. However, in terms of the convergence properties (or *mixing* properties) of the algorithm, not all candidate generators are equally good. Sometimes an obvious candidate generator will suggest itself but, generally, there are no clear guidelines for the choice of this distribution. Note that the candidate generator can depend on the *current* value of the chain, although this is not a requirement. Making good choices of candidate generators is a skill in the implementation of the Metropolis-Hastings algorithm which improves with practice and experience!

• The Metropolis–Hastings algorithm has the major advantage over the Gibbs sampler that it is not necessary to be able to simulate from all the conditional posterior distributions — we only need to simulate from the candidate generator $q(\cdot)$ which we can choose arbitrarily. Moreover, and this can be of crucial importance, we only need to know the conditional posterior densities up to proportionality, since any constants of proportionality cancel in the numerator and denominator of the calculation of p in (5.5). The price for the simplicity is that if $q(\cdot)$ is poorly chosen, then the number of rejections can be high, so that the efficiency of the procedure can be very low.

• Gibbs sampling is a special case of the Metropolis-Hastings algorithm where the candidate generator is chosen as $q(\theta_1^{can}|\theta_1^{(j)},\theta_2^{(j)},\ldots,\theta_d^{(j)}) = \pi(\theta_1^{can}|\theta_2^{(j)},\ldots,\theta_d^{(j)})$ (in other words, we draw directly from the corresponding conditional distribution). In this case, the acceptance probability p in equation (5.5) is equal to 1 so that $\theta_1^{(j+1)} = \theta_1^{can}$ always.

• Another common choice of the candidate generator is to take $q(\theta_1^{can}|\theta_1^{(j)},\theta_2^{(j)},\ldots,\theta_d^{(j)})$ to be the density of a Normal distribution for θ_1^{can} with mean$= \theta_1^{(j)}$ and some suitably chosen variance$= v$. This is the so-called *Random walk Metropolis algorithm with Normal increments* and it is very popular due to its simplicity. Note that the symmetry of the candidate generator means that the terms involving $q(\cdot)$ cancel in equation (5.5), so that the acceptance probability simply becomes

$$p = \min\left\{1,\ \frac{\pi(\theta_1^{can}|\theta_2^{(j)},\ldots,\theta_d^{(j)})}{\pi(\theta_1^{(j)}|\theta_2^{(j)},\ldots,\theta_d^{(j)})}\right\}. \tag{5.6}$$

The variance of the candidate generator v plays an important role in the mixing properties of the algorithm: if v is chosen to be too large, the moves we propose can be too bold leading to very low acceptance probabilities (and, thus, slow mixing); if, on the other hand, v is too small, the acceptance probability will be very high but at the expense of moving in very little steps (again resulting in slow convergence of the algorithm). v is typically chosen by trial and error, aiming at an acceptance probability (very) roughly around 30%.

• We have described the Metropolis-Hastings algorithm for updating θ_1 but, of course, it works in the same way for updating any other parameter component. For example, to obtain $\theta_2^{(j+1)}$ we propose a candidate θ_2^{can} drawn from an arbitrary distribution with density

$q(\theta_2^{can}|\theta_1^{(j+1)}, \theta_2^{(j)}, \ldots, \theta_d^{(j)})$ and accept this candidate as $\theta_2^{(j+1)}$ with probability

$$p = \min\left\{1,\ \frac{\pi(\theta_2^{can}|\theta_1^{(j+1)}, \theta_3^{(j)}, \ldots, \theta_d^{(j)})}{\pi(\theta_2^{(j)}|\theta_1^{(j+1)}, \theta_3^{(j)}, \ldots, \theta_d^{(j)})}\ \frac{q(\theta_2^{(j)}|\theta_1^{(j+1)}, \theta_2^{can}, \ldots, \theta_d^{(j)})}{q(\theta_2^{can}|\theta_1^{(j+1)}, \theta_2^{(j)}, \ldots, \theta_d^{(j)})}\right\}.$$

If θ_2^{can} is rejected, then $\theta_2^{(j+1)} = \theta_2^{(j)}$.

• The most commonly used MCMC algorithms in practice consist of *hybrid chains*. In other words, we use the general Algorithm 4.1, where the various simulation steps are conducted using a combination of Gibbs sampling steps, where possible, and more general Metropolis-Hastings schemes when Gibbs sampling is not possible.

5.3 The genetic linkage example revisited

We can illustrate the procedure with the genetic linkage example of the previous chapter, for which we had

$$\pi(\theta|\mathbf{y}) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4} I[0 \le \theta \le 1].$$

As candidate generator, we can take a Uniform$[0,1]$ distribution. Thus, given that the chain is currently at a certain value θ, we propose $\theta^{can} \sim$Uniform$[0,1]$, and the acceptance probability p in (5.5) reduces to

$$p = \min\left\{1,\ \frac{\pi(\theta^{can}|\mathbf{y})}{\pi(\theta|\mathbf{y})}\right\} \ = \ \min\left\{1,\ \left(\frac{2+\theta^{can}}{2+\theta}\right)^{y_1}\left(\frac{1-\theta^{can}}{1-\theta}\right)^{y_2+y_3}\left(\frac{\theta^{can}}{\theta}\right)^{y_4}\right\}.$$

So the corresponding algorithm works as follows

Algorithm 5.2. *Metropolis-Hastings algorithm for genetic linkage example*

1. *Start the chain at some value* $\theta^{(0)}$
2. *Propose a candidate value* $\theta^{can} \sim$*Uniform*$[0,1]$. *Take as the new value of the chain*

$$\theta^{(1)} = \begin{cases} \theta^{can} & \text{with probability } p \\ \theta^{(0)} & \text{with probability } 1-p \end{cases}$$

where

$$p = \min\left\{1,\ \left(\frac{2+\theta^{can}}{2+\theta^{(0)}}\right)^{y_1}\left(\frac{1-\theta^{can}}{1-\theta^{(0)}}\right)^{y_2+y_3}\left(\frac{\theta^{can}}{\theta^{(0)}}\right)^{y_4}\right\}.$$

(the latter is carried out by sampling $u \sim$Uniform$(0,1)$, *and taking* $\theta^{(1)} = \theta^{can}$ *if and only if* $u < p$*).*

3. *Iterate this procedure*

The R code follows:

```
> mh
function (data,nburn=0,ndraw=1000)
{
```

```
#Metropolis-Hastings for genetic linkage model:

#initial value: drawn from prior

theta <- runif(1,0,1)

# vector of recorded draws:

draws <- numeric(ndraw)

# counter for acceptance probability:

accept <- 0

# MCMC LOOP FOLLOWS:

it <- -nburn
while(it < ndraw){ it <- it+1;

# draw thetacan from Uniform(0,1):

thetacan <- runif(1,0,1)

# compute log(acceptance probability):

logp <- data[1]*log((2+thetacan)/(2+theta)) +
(data[2]+data[3])*log((1-thetacan)/(1-theta)) +
data[4]*log(thetacan/theta)

# draw u ~ Uniform (0,1):

u <- runif(1,0,1)

# if u<p, take thetacan as next value of chain:

if (log(u) < logp){accept <- accept + 1; theta <- thetacan}

# after burn-in record theta:
if(it>0){draws[it] <- theta}

}

# END MCMC

print(c("percentage accepted draws:",100*accept/(nburn+ndraw)))

return(draws)

}
```

As with the Gibbs sampler, it is necessary to monitor the output to ensure convergence. Running the function with 1,100 iterations and no burn-in, and plotting the output I obtained

```
> theta <- mh(data=c(125,18,20,34),ndraw=1100)
[1] "percentage accepted draws:" "15.6363636363636"
> plot(theta,type="l")
```

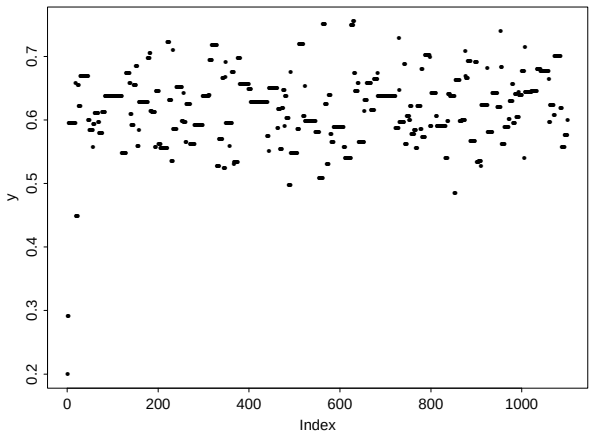


Figure 5.1: 1,100 draws of θ in Metropolis–Hastings algorithm

Thus, the acceptance rate is 16%, a bit low for good mixing as is evident from Figure 5.1. Hence, I decided to run a longer chain of 5,500 iterations. Results are in Figure 5.2. Convergence seems

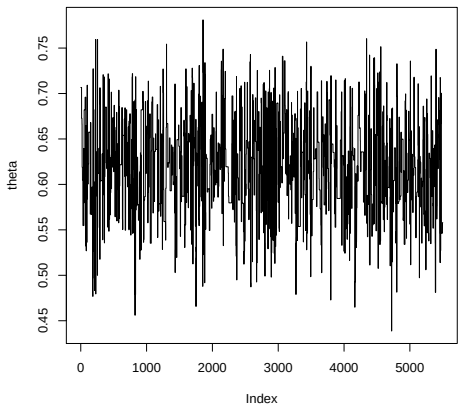


Figure 5.2: 5,500 draws of θ in Metropolis–Hastings algorithm

ok after the first few iterations, so I shall delete the first 500 draws. In addition, to counteract for the slow mixing, I shall keep only every 5^{th} draw:

```
> theta <- theta[-c(1:500)]
> theta <- theta[c(1:1000)*5]
```

The corresponding posterior density estimate of θ is given as a histogram and as a kernel estimate in Figure 5.3. The R code for this figure is

```
> par(mfrow=c(1,2))
> hist(theta,probab=T,xlim=c(0,1))
> plot(density(theta),type="l",xlim=c(0,1),
+ main="posterior density of theta")
```

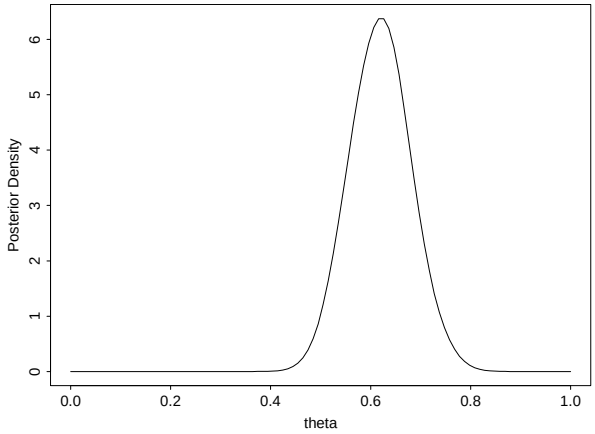


Figure 5.3: Posterior distribution of θ based on Metropolis–Hastings algorithm

5.4 A more involved example

Here we consider again Example 4.1 and construct a Metropolis–Hastings algorithm to generate draws from the posterior distribution in (5.2). Using the general Algorithm 4.1, the parameter is divided into 2 groups, namely μ and ω, and we need to update μ *"according to"* $\pi(\mu|\omega, \mathbf{y})$ and ω *"according to"* $\pi(\omega|\mu, \mathbf{y})$. The required conditional posterior densities are given in (5.3) and (5.4) and, clearly, do not have a form from which we can draw directly. Hence, Gibbs sampling is precluded and we need to resort to the more general Metropolis–Hastings algorithm.

Both for drawing μ and for drawing ω we are going to use Random Walk Metropolis with Normal increments. This leads to the following MCMC algorithm:

Algorithm 5.3. *Random Walk Metropolis-Hastings algorithm for Example 4.1*

1. *Choose initial values* $(\mu^{(0)}, \omega^{(0)})$.
2. *Given that the chain is currently at* $(\mu^{(j)}, \omega^{(j)})$*:*
   - *Draw* $\mu^{can} \sim N(\mu^{(j)}, v_\mu)$ *for some suitably chosen variance* v_μ*, and take*

$$\mu^{(j+1)} = \begin{cases} \mu^{can} & \text{with probability } p \\ \mu^{(j)} & \text{with probability } 1-p \end{cases}$$

where

$$p = \min\left(1, e^{\frac{\kappa_0}{2}\left\{(\mu^{(j)}-\mu_0)^2-(\mu^{can}-\mu_0)^2\right\}} \prod_{i=1}^{n}\left\{\frac{1+\omega^{(j)}(y_i-\mu^{(j)})^2}{1+\omega^{(j)}(y_i-\mu^{can})^2}\right\}\right)$$

(this is implemented by drawing $u \sim$Uniform$(0,1)$ *and taking* $\mu^{(j+1)} = \mu^{can}$ *if and only if* $u < p$*).*

- *Draw* $\omega^{can} \sim N(\omega^{(j)}, v_\omega)$ *for some suitably chosen variance* v_ω*, and take*

$$\omega^{(j+1)} = \begin{cases} \omega^{can} & \text{with probability } p \\ \omega^{(j)} & \text{with probability } 1-p \end{cases}$$

where

$$p = \min\left(1, \left(\frac{\omega^{can}}{\omega^{(j)}}\right)^{\frac{n}{2}+\alpha_0-1} e^{\lambda_0(\omega^{(j)}-\omega^{can})} \prod_{i=1}^{n} \left\{\frac{1+\omega^{(j)}(y_i-\mu^{(j+1)})^2}{1+\omega^{can}(y_i-\mu^{(j+1)})^2}\right\} I[\omega^{can}>0]\right)$$

(this is implemented by drawing $u \sim$Uniform$(0,1)$ *and taking* $\omega^{(j+1)} = \omega^{can}$ *if and only if* $u < p$*).*

3. *Iterate Step 2 a large number of times. Discard an initial number of draws (burn-in) and base inference on subsequent draws.*

Important remark: The acceptance probability p for ω^{can} in *Step 2* can only be positive if the drawn value $\omega^{can} > 0$. In other words, if we draw a value $\omega^{can} < 0$, then $p = 0$ and $\omega^{(j+1)} = \omega^{(j)}$.

The R code to implement this algorithm is:

```
> rwmetrop
function (data,mu0=0,kappa0=1,alpha0=1,lambda0=1,nburn=0,ndraw=5000,
vmu=1,vomega=1)
{

#Random walk Metropolis for model:
#indep observ from Cauchy(mu,1/omega)
#pi(mu,omega)=N(mu|mu0,1/kappa0)Gamma(omega|alpha0,lambda0)

n <- length(data)

alpha1 <- (n/2) + alpha0 - 1

# standard deviations of Normal candidate generators:

stdvmu <- sqrt(vmu)
stdvomega <- sqrt(vomega)

#initial values:drawn from prior

mu <- rnorm(1,mu0,sqrt(1/kappa0))
omega <- rgamma(1,alpha0,1)/lambda0

# create matrix for recorded draws:

draws <- matrix(ncol=2,nrow=ndraw)

# counters for acceptance probabilities:

acceptmu <- 0
```

```
acceptomega <- 0

# MCMC LOOP FOLLOWS:

it <- -nburn
while(it < ndraw){ it <- it+1;

# draw mucan from N(mu,vmu):
mucan <- rnorm(1,mu,stdvmu)

# calculate log(acceptance probability):
logp <- (kappa0/2)*((mu-mu0)^2-(mucan-mu0)^2) +
sum( log( 1+omega*((data-mu)^2)) -
     log( 1+omega*((data-mucan)^2)) )

# draw u ~ Uniform(0,1):
u <- runif(1,0,1)


# if u<p, accept mucan:
if (log(u) < logp){acceptmu <- acceptmu + 1; mu <- mucan}

# draw omegacan from N(omega,vomega):
omegacan <- rnorm(1,omega,stdvomega)

# only compute acceptance probability if omegacan>0:
if(omegacan > 0){

# log(acceptance probability):
logp <- alpha1*(log(omegacan)-log(omega)) + lambda0*(omega-omegacan) +
sum(log(1+omega*((data-mu)^2))-
    log(1+omegacan*((data-mu)^2)) )

# draw u ~ Uniform(0,1):
u <- runif(1,0,1)

# if u<p, accept omegacan:

if (log(u) < logp){acceptomega <- acceptomega + 1; omega <- omegacan}

}

# after bun-in record (mu,omega):
if(it>0){
draws[it,1] <- mu
draws[it,2] <- omega
}

}

# END MCMC

print(c("percentage accepted draws for mu:",100*acceptmu/(nburn+ndraw)))
print(c("percentage accepted draws for omega:",100*acceptomega/(nburn+ndraw)))
```

```
return(draws)

}
```

5.5 Uses of MCMC in classical statisics and beyond...

MCMC has certainly had a fundamental effect on statistics in the last 10 years, influencing not only the way quantities for inference are computed, but also the type of models the statistician tries to fit. This is because the flexibility of the technique expands immeasurably the classes of statistical models for which computation is considered possible. This effect is most pronounced in Bayesian statistics. However MCMC also has many important applications in classical statistics.

We briefly mention three examples.

1. Firstly note that given a flat prior, the likelihood and the posterior density of a parameter θ are equal. Therefore it is feasible to estimate the maximum of the likelihood as the mode of a density estimate of the posterior distribution. This involves the additional problem of obtaining an estimate of the density from a sample of points from that density. This is usually done in terms of a *kernel density estimate*. For example given a sample $\theta^{(j)}$ $1 \leq j \leq J$ (obtained via MCMC) we could estimate the posterior density (or likelihood function) as
$$\frac{1}{J}\sum_{j=1}^{J}\frac{1}{(2\pi\sigma^2)^{1/2}}\exp\{(\theta-\theta^{(j)})^2/(2\sigma^2)\}$$
The 'smoothness' (or band-width) parameter σ^2 needs to be chosen. As a general rule, the larger J, the smaller we would normally like to choose σ^2. Note: this procedure led to the density estimate in Figure 5.3.

2. A rather different application is to the (common) situation where the sampling density is known only up to a normalisation constant, that is
$$f(y|\theta)=\frac{f_\theta(y)}{c(\theta)},$$
where $f_\theta(y)$ is known but $c(\theta)$ is unknown. The problem here is that the normalisation constant $c(\theta)$ is typically a function of θ, in which case it will affect the maximisation problem for the likelihood. Hence, it is necessary to estimate $c(\theta)$. The following idea to find the MLE in such cases is due to Charlie Geyer: Suppose θ_0 is a 'first guess' of the MLE, and notice that the expectation of $f_\theta(Y)/f_{\theta_0}(Y)$ with respect to the density $f(y|\theta_0)$ is
$$E\left[\frac{f_\theta(Y)}{f_{\theta_0}(Y)}\right]=\int\frac{f_\theta(y)}{f_{\theta_0}(y)}f(y|\theta_0)dy=\frac{c(\theta)}{c(\theta_0)}$$
Therefore, if we draw a sample $y_1,\ldots y_I$ from $f(\cdot|\theta_0)$ we can use
$$\frac{1}{I}\sum_{i=1}^{I}\frac{f_\theta(y_i)}{f_{\theta_0}(y_i)}\ .$$
as an estimator of $c(\theta)/c(\theta_0)$. Now given a data vector $\mathbf{y}$ containing n independent observations, it is immediate to estimate the likelihood ratio $f(\mathbf{y}|\theta)/f(\mathbf{y}|\theta_0)$ since
$$\frac{f(\mathbf{y}|\theta)}{f(\mathbf{y}|\theta_0)}=\frac{f_\theta(\mathbf{y})}{f_{\theta_0}(\mathbf{y})}\left(\frac{c(\theta_0)}{c(\theta)}\right)^n\ .$$

In this way, we can improve our estimate of the MLE. This procedure is usually iterated a number of times until the estimated terms stabilize.

3. *Simulated annealing* is a stochastic algorithm for maximising functions. It is very closely related to MCMC. The only difference is that the target density changes as the algorithm proceeds. First notice that if we want to maximise a non-negative function $h(\theta)$ where $\theta \in \Theta$, the maximising value also maximises $h(\theta)^{1/T}$ for arbitrary $T > 0$ known as the temperature. However for small T, assuming that $h(\theta)^{1/T}$ is an integrable function, the distribution with density proportional to $h(\theta)^{1/T}$ concentrates most of its probability mass in the vicinity of the mode of the function. The idea behind simulated annealing is, at each iteration j, to carry out MCMC on a density proportional to $h(\theta)^{1/T_j}$, where T_j is a sequence of temperatures that converges to zero. The algorithm converges to the maximum value of $h(\theta)$ under suitable regularity conditions. It is important that T_j does not converge to zero too rapidly, since this could then lead to the algorithm getting stuck in a minor mode of the function $h(\theta)$.

5.6 Further recommendations about MCMC

As you have probably gathered by now, MCMC is an extremely powerful tool that allows us to handle highly complex models. However, it is not straightforward to implement, and doing this successfully requires care and skill. There are a couple of issues that I would like to stress regarding implementation of MCMC algorithms:

• Assessing convergence of the algorithm is extremely important, but can be problematic in high dimensional situations. Therefore it is rarely possible to be absolutely certain that our algorithm has converged sufficiently well. At the minimum, *always run the chain several times using different starting values* and check that the output from the various chains is very similar. Furthermore, since computers these days tend to be very fast *run long chains and have long burn-in periods*. It is much better to have a very long chain than to have one that is not long enough.

• Be extremely cautious if you use improper prior distributions. Improper prior distributions do not always lead to proper posteriors, but a lack of propriety of the posterior distribution may not be detected when implementing an MCMC algorithm. This is because it may well happen that all the conditional posterior distributions are proper and, yet, the joint posterior (the one that needs to be proper for posterior inference to be valid) is not proper. It may well happen that the MCMC output looks very nice and, yet, posterior inference is invalid. Thus, if you use improper priors *always check analytically* that the joint posterior distribution is proper, otherwise you can not have any faith in the results obtained via computation. This problem does not arise if you use proper prior distributions.

Chapter 6

MCMC in Generalised Linear Models

In this chapter we will discuss various approaches to implementing MCMC algorithms in generalised linear models. In particular, we will consider logistic and probit regression models for binary data and log-linear Poisson regression models for count data. These models have in common that there are no conjugate priors available for regression parameters and, therefore, conditional posterior distributions have non-standard forms. We will review different algorithms proposed in the literature and discuss their implementation. Those will mostly involve Metropolis-Hastings proposals.

We will first discuss models which are essentially Bayesian versions of *generalised linear models*. Here we can perform a comparison with results obtained from a maximum likelihood approach.

We will then extend this class of models by introducing additional *random effects*, i.e. we will consider the so-called class of *generalised linear mixed models*. This class of models have a wide range of applications, but we will discuss only one: to account for overdispersion.

6.1 Logistic regression

Consider the following data taken from Fahrmeir and Tutz (2001). The response variable is the occurrence or non-occurrence of infection following birth by Caesarian section. The following covariate information was available:

$x_1 = 1$ if Caesarian section was not planned and $x_1 = 0$ otherwise;

$x_2 = 1$ if there is presence of one of more risk factors, such as diabetes or excessive weight, and $x_2 = 0$ otherwise;

$x_3 = 1$ if antibiotics were given as prophylaxis and $x_3 = 0$ otherwise.

There is a total of 251 births, and the data are given in Table 6.1.

Note that we have aggregated all binary responses in each covariate category obtaining binomial responses. Thus,

$$y_i \sim \text{Binomial}(n_i, p_i)$$

where i is an index for each covariate combination, y_i is the number of infections, n_i the total number of observations, and p_i the probability of infection for each individual with this covariate

Covariates			Infection	
x_1	x_2	x_3	yes	no
0	0	0	8	32
0	0	1	0	2
0	1	0	28	30
0	1	1	1	17
1	0	0	0	9
1	0	1	0	0
1	1	0	23	3
1	1	1	11	87

Table 6.1: Data on Caesarian birth study

combination. Suppose we assume the binary logistic regression model,

$$\text{logit } p_i = \log\left(\frac{p_i}{1-p_i}\right) = \eta_i = \boldsymbol{x}_i^T\boldsymbol{\beta}$$

where $\boldsymbol{x}_i^T = (1, x_{1i}, x_{2i}, x_{3i})$ denotes the vector of covariates. Then, the likelihood is given by

$$f(\boldsymbol{y}|\boldsymbol{\beta}) = \prod_{i=1}^{n} f(y_i|\boldsymbol{\beta}) \propto \prod_{i=1}^{n} \frac{\exp(\boldsymbol{x_i}^T\boldsymbol{\beta}\cdot y_i)}{(1+\exp(\boldsymbol{x_i}^T\boldsymbol{\beta}))^{n_i}}.$$

A Maximum likelihood analysis in R using

```
> data <- matrix(c(0,0,0,8,32,
+ 0,0,1,0,2,
+ 0,1,0,28,30,
+ 0,1,1,1,17,
+ 1,0,0,0,9,
+ 1,0,1,0,0,
+ 1,1,0,23,3,
+ 1,1,1,11,87),ncol=5,byrow=T)

> noplan <- data[,1]
> factor <- data[,2]
> antib <- data[,3]
> yes <- data[,4]
> no <- data[,5]

> caesarianglm <- glm(cbind(yes, no) ~ noplan + factor + antib,
family=binomial)
```

gives the following results

```
> summary(caesarianglm)

Call:
glm(formula = cbind(yes, no) ~ noplan + factor + antib, family = binomial)
```

```
Deviance Residuals:
[1]   1.21563  -0.15231  -0.78520   0.26470  -2.56229   0.00000   1.49623
[8]  -0.07162

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.8926     0.4124  -4.590 4.44e-06 ***
noplan        1.0720     0.4253   2.520   0.0117 *
factor        2.0299     0.4552   4.459 8.23e-06 ***
antib        -3.2544     0.4813  -6.762 1.37e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 83.491  on 6  degrees of freedom
Residual deviance: 10.997  on 3  degrees of freedom
AIC: 36.178

Number of Fisher Scoring iterations: 4
```

The 10.997 value for the residual deviance is not obviously inconsistent with a χ^2_3 distribution, since we can compute $P(\chi^2_3 > 10.997)$ as

```
> 1-pchisq(10.997,3)
[1] 0.01174211
```

so there is no strong warning sign that the model may not fit adequately.

A Bayesian analysis places a prior on $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)^T$, usually a multivariate normal prior $\boldsymbol{\beta} \sim N_4(\boldsymbol{\mu}_0, \boldsymbol{C}_0)$. The posterior distribution for $\boldsymbol{\beta}$ is therefore

$$\pi(\boldsymbol{\beta}|\boldsymbol{y}) \propto f(\boldsymbol{y}|\boldsymbol{\beta})\pi(\boldsymbol{\beta}) \propto \left\{ \prod_{i=1}^{n} \frac{\exp(\boldsymbol{x_i}^T\boldsymbol{\beta} \cdot y_i)}{(1+\exp(\boldsymbol{x_i}^T\boldsymbol{\beta}))^{n_i}} \right\} \exp\left(-\frac{(\boldsymbol{\beta}-\boldsymbol{\mu}_0)^T \boldsymbol{C}_0^{-1} (\boldsymbol{\beta}-\boldsymbol{\mu}_0)}{2} \right). \quad (6.1)$$

The posterior density is a complicated, non-linear function of $\boldsymbol{\beta}$. How can we construct an MCMC algorithm to sample from this distribution? There are various approaches discussed in the literature:

1. **Update $\boldsymbol{\beta}$ component-wise using Gibbs sampling**: Generate draws from the conditional posterior distribution of each component of $\boldsymbol{\beta}$,

   $$\pi(\beta_j|\boldsymbol{y}, \boldsymbol{\beta}_{-j}) \propto \pi(\boldsymbol{\beta}|\boldsymbol{y})$$

   in turn.
   Problems: • It is rather tedious to sample from $\pi(\beta_j|\boldsymbol{y}, \boldsymbol{\beta}_{-j})$ because it is non-standard; • the algorithm may converge slowly if there are strong correlations between parameters.

2. **Update $\boldsymbol{\beta}$ component-wise via Metropolis-Hastings**: For example, use a simple random walk proposal.
   Problems: • Each random walk proposal needs tuning; • problems with collinearity may occur.

3. **Update $\boldsymbol{\beta}$ jointly**: Use a Multivariate Normal candidate generator with mean equal to posterior mode and covariance matrix equal to inverse curvature at mode. This is an *independence sampler*.
   Problems: • Need to implement Newton-Raphson-type algorithm for $\pi(\boldsymbol{\beta}|\boldsymbol{y})$; • approach relies on asymptotic normality of posterior distribution; • independence candidate generator may rarely propose values in the tails of the posterior.

4. **Update $\boldsymbol{\beta}$ jointly**: Use a multivariate Gaussian random walk candidate generator, with covariance matrix equal to the inverse curvature at posterior mode times a (scalar) tuning factor, or some other pre-estimated covariance matrix
   Problems: • Needs tuning and pre-calculation of covariance estimate.

The code that implements algorithms 2, 3 and 4 (file "logistic.r") can be found in one of the appendices at the end of these notes and on the course web page. Note that in algorithms 2 and 4 random walk proposals are being used, which implies that the proposal ratio in the Metropolis-Hastings-acceptance probability equals one.

The performance of these algorithms will be illustrated in class (see files "logistic2.pdf", "logistic3.pdf" and "logistic4.pdf" on the course web page). Estimates (posterior mean, standard deviation and posterior probabilities of exceeding 0) using the third algorithm (5,000 samples after burn-in of 500) are given in the following table:

Coefficients:	posterior mean	posterior Std	posterior probability
(Intercept)	-1.9544	0.4228	0
noplan	1.1071	0.4229	0.9968
factor	2.0955	0.467	1
antib	-3.3322	0.4867	0

The overall acceptance rate of this run was 87.6%.

Generally we would prefer an algorithm which is automatic (no tuning required) and updates $\boldsymbol{\beta}$ in one block, in order to avoid strong autocorrelations in the samples. A slightly more involved, but generally better algorithm for inference in logistic regression models, has been proposed by Gamerman (1997). His method is based on the weighted least squares (WLS) algorithm, known from the class on Generalised Linear Models. We will review the WLS algorithm first and comment on connections between likelihood and Bayesian inference.

The ML estimator in a GLM and its asymptotic covariance matrix are obtained by iterative use of WLS on transformed observations. The algorithm takes a simple form in the case of canonical link functions $g(\mu_i) = \eta_i = \boldsymbol{x}_i^T\boldsymbol{\beta}$, where $\mu_i = E(y_i)$, such as the logit link for binary regression, or the logarithmic link for Poisson regression. For logistic regression with binomial response (as in the example here), the canonical link function is

$$g(\mu_i) = \log\left(\frac{p_i}{1-p_i}\right) = \log\left(\frac{n_i p_i}{n_i - n_i p_i}\right) = \log\left(\frac{\mu_i}{n_i - \mu_i}\right).$$

Define a vector of transformed observations $\tilde{\boldsymbol{y}}(\boldsymbol{\beta})$ and a diagonal matrix of weights $\boldsymbol{W}(\boldsymbol{\beta})$ as

$$\begin{aligned} \tilde{y}_i(\boldsymbol{\beta}) &= \eta_i + (y_i - \mu_i)g'(\mu_i) \\ W_i(\boldsymbol{\beta}) &= 1/g'(\mu_i), \end{aligned}$$

where $g'(\mu_i)$ is the first derivative of the link function.

The Iterative Weighted Least Squares (IWLS) algorithm starts with some arbitrary value $\boldsymbol{\beta}^{(0)}$, and iteratively obtains $\boldsymbol{\beta}^{(t)}$, $t = 1, \ldots$ as the least squares estimator in the general (with weights)

linear model

$$\tilde{\boldsymbol{y}}(\boldsymbol{\beta}^{(t-1)}) \sim N(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{W}^{-1}(\boldsymbol{\beta}^{(t-1)}))$$

i.e.

$$\boldsymbol{\beta}^{(t)} = (\boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{\beta}^{(t-1)})\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{\beta}^{(t-1)})\tilde{\boldsymbol{y}}(\boldsymbol{\beta}^{(t-1)}). \tag{6.2}$$

After convergence, the final estimate $\hat{\boldsymbol{\beta}}$ is the ML estimate and the matrix $(\boldsymbol{X}^T\boldsymbol{W}(\hat{\boldsymbol{\beta}})\boldsymbol{X})^{-1}$ is the associated asymptotic covariance matrix.

Bayesian context:

The latter estimates also have a Bayesian interpretation: they can be shown to correspond to the mode and inverse curvature matrix at the mode of the posterior $\pi(\boldsymbol{\beta}|\boldsymbol{y})$ under a flat prior on the regression coefficients, i.e. under $\pi(\boldsymbol{\beta}) \propto 1$.

With a proper normal prior on $\boldsymbol{\beta}$, $\boldsymbol{\beta} \sim N(\boldsymbol{\mu}_0, \boldsymbol{C}_0)$, the IWLS algorithm can be modified as follows in order to find the mode and curvature at the mode of $\pi(\boldsymbol{\beta}|\boldsymbol{y})$:

Combining

$$\tilde{\boldsymbol{y}}(\boldsymbol{\beta}^{(t-1)}) \sim N(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{W}^{-1}(\boldsymbol{\beta}^{(t-1)}))$$

with the prior

$$\boldsymbol{\beta} \sim N(\boldsymbol{\mu}_0, \boldsymbol{C}_0),$$

we obtain

$$\boldsymbol{\beta}|\tilde{\boldsymbol{y}}(\boldsymbol{\beta}^{(t-1)}) \sim N(\boldsymbol{\beta}^{(t)}, (\boldsymbol{C}_0^{-1} + \boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{\beta}^{(t-1)})\boldsymbol{X})^{-1})$$

where

$$\boldsymbol{\beta}^{(t)} = (\boldsymbol{C}_0^{-1} + \boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{\beta}^{(t-1)})\boldsymbol{X})^{-1}(\boldsymbol{C}_0^{-1}\boldsymbol{\mu}_0 + \boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{\beta}^{(t-1)})\tilde{\boldsymbol{y}}(\boldsymbol{\beta}^{(t-1)})). \tag{6.3}$$

Note that this involves only the prior precision matrix $\boldsymbol{P}_0 = \boldsymbol{C}_0^{-1}$.

In order to find the posterior mode (let us call it $\hat{\boldsymbol{\beta}}$), we simply need to iterate (6.3) until convergence. The inverse of the curvature at the posterior mode is then given by:

$$(\boldsymbol{C}_0^{-1} + \boldsymbol{X}^T\boldsymbol{W}(\hat{\boldsymbol{\beta}})\boldsymbol{X})^{-1}$$

Note: this algorithm is implemented in MCMC samplers 3 and 4 in order to compute the posterior mode and inverse curvature at the mode.

Asymptotically (as $n \to \infty$), the posterior distribution is Normal

$$\boldsymbol{\beta}|\boldsymbol{y} \sim N(\hat{\boldsymbol{\beta}}, (\boldsymbol{C}_0^{-1} + \boldsymbol{X}^T\boldsymbol{W}(\hat{\boldsymbol{\beta}})\boldsymbol{X})^{-1})$$

Gamerman's Metropolis-Hastings IWLS algorithm:

The idea of Gamerman's IWLS proposal is now to perform only one step of this iteration, starting at the current value $\boldsymbol{\beta}$, and use the resulting multivariate Gaussian as the proposal distribution in a Metropolis-Hastings setting. In other words, take as candidate generator the density

$$q(\boldsymbol{\beta}^{\boldsymbol{can}}|\boldsymbol{\beta}, \boldsymbol{y}) \sim N(f(\boldsymbol{\beta}), (\boldsymbol{C}_0^{-1} + \boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X})^{-1}) \tag{6.4}$$

where

$$f(\boldsymbol{\beta}) = (\boldsymbol{C}_0^{-1} + \boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X})^{-1}(\boldsymbol{C}_0^{-1}\boldsymbol{\mu}_0 + \boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{\beta})\tilde{\boldsymbol{y}}(\boldsymbol{\beta})). \tag{6.5}$$

The resulting MCMC algorithm has several advantages:

- It does not require any tuning constants;
- The proposal obtained is reasonably close to the posterior, and therefore the algorithm leads to high acceptance probabilities;

• It avoids problems with independence proposals at the posterior mode;

• A similar approach can also be used in more complicated models, such as generalized linear mixed models (GLMM).

Acceptance probability:

$$p = \min\left\{1, \frac{\pi(\boldsymbol{\beta}^{\boldsymbol{can}}|\boldsymbol{y})}{\pi(\boldsymbol{\beta}|\boldsymbol{y})}\frac{q(\boldsymbol{\beta}|\boldsymbol{\beta}^{\boldsymbol{can}},\boldsymbol{y})}{q(\boldsymbol{\beta}^{\boldsymbol{can}}|\boldsymbol{\beta},\boldsymbol{y})}\right\}$$

Note that evaluation of the proposal ratio involves not only the term $q(\boldsymbol{\beta}^{can}|\boldsymbol{\beta},\boldsymbol{y})$ (the denominator in the acceptance probability formula), but also the term $q(\boldsymbol{\beta}|\boldsymbol{\beta}^{can},\boldsymbol{y})$, the numerator of the proposal ratio. So, to evaluate the proposal ratio, we also have to perform the inverse IWLS step, starting at the proposed value $\boldsymbol{\beta}^{can}$, calculating mean and covariance matrix of the associated proposal, and finally evaluating the normal density with these parameters at the point $\boldsymbol{\beta}$, the current state of the chain.

On our test data, consisting of logistic regression with binary responses, we have

$$g(\mu_i) = \log\left(\frac{\mu_i}{n_i - \mu_i}\right) \implies g'(\mu_i) = \frac{n_i}{\mu_i(n_i - \mu_i)},$$

from which the matrix of weights $\boldsymbol{W}(\boldsymbol{\beta})$ is constructed. The results using Gamerman's algorithm (also in file "logistic.r") are as follows (using 5,000 samples after burn-in of 500):

Coefficients:	posterior mean	posterior Std	posterior probability
(Intercept)	-1.9717	0.4328	0
noplan	1.092	0.4206	0.995
factor	2.1148	0.4823	1
antib	-3.3148	0.4922	0

See also the file "logisticgamer.pdf". Note the general good agreement with the results obtained earlier. Nevertheless, for both algorithms there is still some Monte Carlo error left in the estimates. The overall acceptance rate of this run was 74.1%. More details on this algorithm are given in class.

6.2 Data augmentation in binary probit regression

We have seen in the previous paragraph that inference in Bayesian logistic regression models via MCMC is non-trivial and requires quite clever algorithms, because of non-linearities in the likelihood. This is the case for nearly all non-normal GLM's and the corresponding extensions, but there is an exception: binary regression with probit rather than logit link.

We will show that, in this case, by clever introduction of auxiliary variables (that is, via data augmentation), we can reduce the algorithm essentially to a normal linear model. This algorithm goes back to the work by Albert and Chib (1993).

Consider the Bayesian probit regression model,

$$\begin{aligned} y_i &\sim \text{Bernoulli}\left(\Phi(\eta_i)\right) \\ \eta_i &= \boldsymbol{x}_i^T\boldsymbol{\beta} \\ \boldsymbol{\beta} &\sim N(\boldsymbol{\mu}_0, \boldsymbol{C}_0) \end{aligned} \tag{6.6}$$

where $y_i \in \{0,1\}, i = 1,\ldots,n$ is a binary response variable for a collection of n objects with associated covariate measurement $\boldsymbol{x}_i$, $\Phi(\cdot)$ is the standard normal distribution function, η_i is

the linear predictor and $\boldsymbol{\beta}$ represents a $(p \times 1)$ column vector of regression coefficients which *a priori* are from a normal distribution. In the probit model, the mean is given by $\mu_i = \Phi(\eta_i)$, hence, it corresponds to the probit link function: $g(\mu_i) = \Phi^{-1}(\mu_i)$.

The probit model in (6.6) has an equivalent representation using auxiliary variables z_i (for $i = 1, \ldots, n$):

$$\begin{array}{rcl} y_i & = & \left\{ \begin{array}{ll} 1 & \text{if } z_i > 0 \\ 0 & \text{otherwise} \end{array} \right. \\ z_i & \sim & N(\boldsymbol{x}_i^T \boldsymbol{\beta}, 1) \\ \boldsymbol{\beta} & \sim & N(\boldsymbol{\mu}_0, \boldsymbol{C}_0) \end{array} \tag{6.7}$$

where y_i is now deterministic conditional on the sign of the stochastic auxiliary variable z_i, which follows a normal distribution. The distribution of y_i in model (6.7), having integrated out z_i, is the same as in (6.6):

$$P(y_i = 1) = P(z_i > 0) = P(N(\boldsymbol{x}_i^T \boldsymbol{\beta}, 1) > 0) = P(N(0,1) > -\boldsymbol{x}_i^T \boldsymbol{\beta}) = \Phi(\boldsymbol{x}_i^T \boldsymbol{\beta})$$

The advantage of working with representation (6.7) is that it lends itself to efficient simulation using the block Gibbs sampler. The joint posterior density of $(\boldsymbol{\beta}, z_1, \ldots, z_n)$ is given by

$$\begin{aligned} \pi(\boldsymbol{\beta}, z_1, \ldots, z_n | \boldsymbol{y}) \quad \propto \quad & \prod_{i:y_i=1} \exp\Big[-\frac{1}{2}(z_i - \boldsymbol{x_i}^T \boldsymbol{\beta})^2\Big] I[z_i > 0] \prod_{i:y_i=0} \exp\Big[-\frac{1}{2}(z_i - \boldsymbol{x_i}^T \boldsymbol{\beta})^2\Big] I[z_i < 0] \\ & \exp\Big[-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\mu}_0)^T \boldsymbol{C}_0^{-1} (\boldsymbol{\beta} - \boldsymbol{\mu}_0)\Big]. \end{aligned}$$

It is now immediate that the conditional posterior distribution of $\boldsymbol{\beta}$ is normal:

$$\boldsymbol{\beta} | \boldsymbol{z}, \boldsymbol{y} \sim N\Big((\boldsymbol{C}_0^{-1} + \boldsymbol{X}^T \boldsymbol{X})^{-1} (\boldsymbol{C}_0^{-1} \boldsymbol{\mu_0} + \boldsymbol{X}^T \boldsymbol{z}),\ (\boldsymbol{C}_0^{-1} + \boldsymbol{X}^T \boldsymbol{X})^{-1}\Big), \tag{6.8}$$

where $\boldsymbol{z} = (z_1, \ldots, z_n)^T$, whereas the conditional posterior distribution for each z_i is *truncated normal*,

$$z_i | \boldsymbol{\beta}, \boldsymbol{y} \quad \propto \quad \left\{ \begin{array}{ll} N(\boldsymbol{x}_i^T \boldsymbol{\beta}, 1)\ I[z_i > 0] & \text{if } y_i = 1 \\ N(\boldsymbol{x}_i^T \boldsymbol{\beta}, 1)\ I[z_i \leq 0] & \text{otherwise,} \end{array} \right. \tag{6.9}$$

which is simple to sample from, see for example Robert (1995). This algorithm is simple Gibbs sampling, hence quite easy to implement (see file "probit.r" in an appendix at the end of these notes and in the course web page).

A slight disadvantage of the auxiliary variables approach, however, is that we can not aggregate to binomial responses (as we have done when considering logistic regression). Instead, we have to consider the individual $n = 251$ binary responses. This causes the algorithm to be slightly slower than the ones used for logistic regression. However, this can be different for other data.

The results obtained using the probit approach can be found in the following table (see file "probit.pdf" on course web page for additional details on the results):

Coefficients:	posterior mean	posterior Std	posterior probability
(Intercept)	-1.115	0.2211	0
noplan	0.6092	0.2501	0.9954
factor	1.2204	0.2608	1
antib	-1.9115	0.2634	0

Those are quite different from the logistic model because of the different link function. However, the relationship between the Probit and the Logit link function is mainly a change in scale, so if we adjust our estimates of posterior mean and standard deviation by a factor of $\pi/\sqrt{3} \cdot 15/16 \approx 1.7$, the results look now very similar to the ones obtained using logistic regression:

Coefficients:	posterior mean	posterior Std	posterior probability
(Intercept)	-1.8955	0.3759	0
noplan	1.0356	0.4251	0.9954
factor	2.0747	0.4434	1
antib	-3.2496	0.4478	0

6.3 Poisson regression

In log-linear Poisson regression, it is assumed that

$$y_i \sim \text{Poisson}(\mu_i), \quad \text{where } g(\mu_i) = \log(\mu_i) = \eta_i = \boldsymbol{x}_i^T \boldsymbol{\beta},$$

independently for $i = 1, \ldots, n$. As before, we consider a normal prior for the regression vector:

$$\boldsymbol{\beta} \sim N(\boldsymbol{\mu}_0, \boldsymbol{C}_0).$$

For these models, there exists no data augmentation approach. However, we can implement methods similar to the ones discussed for logistic regression. In particular, Gamerman's Metropolis-Hastings IWLS algorithm can be adapted to the Poisson case and will be useful for block updates of $\boldsymbol{\beta}$. This algorithm will be implemented in the practical. It now takes a slightly different form from the one seen for logistic regression, because we have to replace the logit link function by the logarithmic link function, and hence $g'(\mu_i) = 1/\mu_i$.

Note that often *offsets* are used in Poisson regression. For example, in geographical epidemiology, a common model for observed disease counts in area i is

$$y_i \sim \text{Poisson}(\mu_i = e_i \lambda_i)$$

where e_i are known expected cases. Assume now we want to investigate the effect of further covariates $\boldsymbol{x}_i$ in this model. We could then assume

$$\lambda_i = \exp(\boldsymbol{x}_i^T \boldsymbol{\beta}),$$

obtaining that

$$\log(\mu_i) = \eta_i = \log(e_i) + \boldsymbol{x}_i^T \boldsymbol{\beta}$$

is the linear predictor. Therefore, the log expected cases $\log(e_i)$ appear as an *offset* and have to be incorporated in the IWLS proposal. In order to do this, the only change required is to replace the transformed variables $\tilde{y}_i(\boldsymbol{\beta})$ by $\tilde{y}_i(\boldsymbol{\beta}) - \log(e_i)$ (in other words, subtract the offset).

MCMC in Poisson regression will be discussed in detail in the practical.

6.4 A case study: Logistic regression with random effects

We will now consider logistic regression models with additional random effects. For simplicity, we will only consider the case where, for each observation, there is an additional random effect. Such models are often used to adjust for *overdispersion*. The most common model is to assume

that the random effects are independent realizations from a Gaussian distribution with mean zero and unknown variance. We will restrict our attention to this case.

As an example we consider data given by Crowder (1978) consisting of the proportion of seeds that germinated in $n = 21$ plates. Plate i ($i = 1, \ldots, n$) contains n_i seeds of which y_i germinated. Relevant covariates in $\boldsymbol{x}_i$ are seed type (2 types), root extract (2 types) and an interaction term; the vector $\boldsymbol{x}_i^T$ also contains a unit element (i.e. the model also has an intercept). A standard logistic regression model would assume

$$y_i \sim \text{Binomial}(n_i, p_i),$$

where

$$\text{logit } p_i = \log\left(\frac{p_i}{1-p_i}\right) = \boldsymbol{x}_i^T\boldsymbol{\beta}.$$

If we want to adjust for overdispersion, we could extend this model by including additional random effects b_i ($i = 1, \ldots, n$), as follows:

$$\text{logit } p_i = \eta_i = \boldsymbol{x}_i^T\boldsymbol{\beta} + b_i, \tag{6.10}$$

where

$$b_i \sim N(0, \omega^{-1})$$

are assumed to be independent. The precision ω is treated as unknown and is typically assigned a Gamma prior, say

$$\omega \sim \text{Gamma}(c, d).$$

Such an extension increases the number of parameters considerably: For the data above there are $n = 21 + 1$ additional parameters in the model.

The posterior distribution is now given as

$$\pi(\boldsymbol{\beta}|\boldsymbol{y}) \propto f(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{b})\pi(\boldsymbol{\beta})\pi(\boldsymbol{b}|\omega)\pi(\omega),$$

where $f(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{b})$ is the binomial likelihood and $\pi(\boldsymbol{b}|\omega)$ is the normal random effects prior. As usual, we will take a Normal prior for $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} \sim N(\boldsymbol{\mu}_0, \boldsymbol{C}_0).$$

How can we construct an efficient MCMC algorithm to sample from this distribution, preferably without any need for tuning? We will discuss two approaches, the first is based on Gamerman's IWLS proposals, the other one is based on a clever *reparametrization* of the model. Note that sampling from the conditional posterior distribution of ω is straightforward, because it is still Gamma with parameters $c + n/2$ and $d + 0.5\sum_i b_i^2$.

6.5 An algorithm based on IWLS proposals

First we note that, for updating $\boldsymbol{\beta}$ given $\boldsymbol{b}$, the b_i's serve as an offset. This means that updating can be done similarly to how it was done in the model without random effects:

$$\begin{aligned} \tilde{y}_i(\boldsymbol{\beta}) &= \eta_i + (y_i - \mu_i)g'(\mu_i) - b_i = \boldsymbol{x}_i^T\boldsymbol{\beta} + (y_i - \mu_i)g'(\mu_i) \\ W_i(\boldsymbol{\beta}) &= 1/g'(\mu_i), \end{aligned}$$

and then using equations (6.4) and (6.5).

Similarly, for updating b_i, the term $\boldsymbol{x}_i^T\boldsymbol{\beta}$ will serve as an offset and hence the IWLS machinery can be used again:

$$\begin{aligned}\tilde{y}_i(b_i) &= \eta_i + (y_i - \mu_i)g'(\mu_i) - \boldsymbol{x}_i^T\boldsymbol{\beta} = b_i + (y_i - \mu_i)g'(\mu_i)\\ W_i(b_i) &= 1/g'(\mu_i).\end{aligned}$$

Note that only one likelihood term is relevant for each random effect. For updating b_i, $i = 1, \ldots, n$, the proposal distribution takes the form

$$b_i^{can} \sim N\left(\frac{W_i(b_i)\tilde{y}_i(b_i)}{\omega + W_i(b_i)}, \frac{1}{\omega + W_i(b_i)}\right).$$

6.6 Reparametrising the model

A slight simplification of the algorithm can be obtained by a reparametrisation of the model as follows. Instead of considering the parameters $b_1, \ldots, b_n$, where $b_i \sim N(0, \omega^{-1})$, we can consider $\eta_1, \ldots, \eta_n$, where $\eta_i|\boldsymbol{\beta} \sim N(\boldsymbol{x}_i^T\boldsymbol{\beta}, \omega^{-1})$. With this parametrisation, the model can be written as

$$\begin{aligned}y_i &\sim \text{Binomial}(n_i, p_i), \quad \text{where logit } p_i = \eta_i\\ \eta_i &\sim N(\boldsymbol{x}_i^T\boldsymbol{\beta}, \omega^{-1})\\ \boldsymbol{\beta} &\sim N(\boldsymbol{\mu}_0, \boldsymbol{C}_0)\\ \omega &\sim \text{Gamma}(c, d)\end{aligned}$$

This procedure is termed *hierarchical centering* and has the advantage that the full conditional for $\boldsymbol{\beta}$ is now multivariate normal:

$$\boldsymbol{\beta}|\boldsymbol{\eta} \sim N((\boldsymbol{C}_0^{-1} + \omega\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{C}_0^{-1}\boldsymbol{\mu}_0 + \omega\boldsymbol{X}^T\boldsymbol{\eta}), (\boldsymbol{C}_0^{-1} + \omega\boldsymbol{X}^T\boldsymbol{X})^{-1}).$$

The updating of η_i can be done along similar lines as that of b_i in the algorithm discussed earlier. If fact, the only two differences are that (a) we have to set the offset equal to zero (in other words, there is no longer an offset) and (b) we have to replace the prior mean (which was $\mu_i = 0$ for b_i) by $\mu_i = \boldsymbol{x_i}^T\boldsymbol{\beta}$. The proposal is then

$$\eta_i^{can} \sim N\left(\frac{\omega\boldsymbol{x}_i^T\boldsymbol{\beta} + W_i(\eta_i)\tilde{y}_i(\eta_i)}{\omega + W_i(\eta_i)}, \frac{1}{\omega + W_i(\eta_i)}\right),$$

where

$$\begin{aligned}\tilde{y}_i(\eta_i) &= \eta_i + (y_i - \mu_i)g'(\mu_i)\\ W_i(\eta_i) &= 1/g'(\mu_i).\end{aligned}$$

Finally, the full conditional for ω is Gamma with parameters $c + n/2$ and $d + \sum_i(\eta_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2$.

Note: Both algorithms (with and without reparametrisation) can be found in the file "logisticrandom.r" (appendix at the end of these notes and course web page), whereas the files "crowderrand1.pdf" and "crowderrand2.pdf" (on web page) display results from running them.

The files "logisticrand1.pdf" and "logisticrand2.pdf" (on web page) display results from running these algorithms on the Ceasarian data.

6.7 Binary regression with random effects and probit link

Just as for the ordinary probit regression model, things are much simpler here. Due to the introduction of the auxillary variables, the full conditional distributions for $\boldsymbol{\beta}$ and b_i, $i = 1, \ldots, n$ are all normal. Similarly, in the reparametrised version discussed in the previous section, not only the full conditional for $\boldsymbol{\beta}$, but also for η_i is normal. Hence a simple Gibbs samples can be implemented. We leave it to the reader to derive the exact formulae of these normal distributions.

Chapter 7

Appendix 1: Basic MCMC Theory

Here we present a brief account of the theory behind MCMC algorithms. In other words, we try to explain why they work and why we can treat the draws from the conditional posterior distributions as if they were draws from the joint posterior. The summary we present here is far from complete and is not very rigorous, the aim being merely to provide some intuition as to why things work. For more detail, see the article by Peter Green in the book *Complex Stochastic Systems* and references therein.

The general idea behind MCMC is that instead of simulating independent draws from the posterior (or target) distribution $\pi(\theta|\mathbf{y})$, we construct a Markov chain whose state space is the parameter space (the space Θ to which θ belongs) and whose limiting distribution is the required posterior distribution $\pi(\theta|\mathbf{y})$. Now we present the theory under which we can treat the draws from this Markov chain essentially as if they were draws from $\pi(\theta|\mathbf{y})$.

Let $\pi(\theta|\mathbf{y})$ be the posterior distribution of interest and $P(\theta, \theta')$ the *transition kernel* of the Markov chain (that is, $P(\theta, \theta')$ gives the probability that the chain moves to θ' given that it is at θ).

Definition 1. $\pi(\theta|\mathbf{y})$ *is* **invariant** *for the transition kernel* $P(\theta, \theta')$ *if*

$$\int_{\Theta} P(\theta, A)\pi(\theta|\mathbf{y})d\theta = \pi(A|\mathbf{y}), \quad \text{for any set } A \subset \Theta.$$

Intuitively, this means that iterating via the Markov chain does not alter the target distribution $\pi(\theta|\mathbf{y})$: if we draw θ from $\pi(\theta|\mathbf{y})$ and then iterate once through the transition kernel $P(\theta, \cdot)$, the probability that we end up in a set A is the posterior probability of that set. It seems, intuitively, that this is a sensible property to require if we are to use the draws from the Markov chain as if they were draws from $\pi(\theta|\mathbf{y})$.

Definition 2. *The transition kernel* $P(\theta, \theta')$ *is* **irreducible** *for* $\pi(\theta|\mathbf{y})$ *if for any set* $A \subset \Theta$ *that has positive posterior probability, there exists a positive probability that the chain visits* A *after a finite number of iterations regardless of the chain's starting value.*

This simply means that the chain is able to visit all regions that have positive posterior probability. For example, if the parameter θ is real-valued, the chain should be able to visit the entire real line (a chain that only visits the positive real line, say, is not valid). Again, this seems like a sensible property to require from the Markov chain.

Definitions 1 and 2 capture all the conditions that are required to be able to use the draws from the chain to compute posterior expectations. The result is given in the following theorem:

Theorem 1. *Consider a Markov chain whose transition kernel $P(\theta, \theta')$ has $\pi(\theta|\mathbf{y})$ as invariant distribution and is irreducible for $\pi(\theta|\mathbf{y})$. Then, for each real-valued function $t(\theta)$ whose posterior expectation exists we have that*

$$\frac{1}{J}\sum_{j=1}^{J} t(\theta^{(j)}) \longrightarrow \int t(\theta)\pi(\theta|\mathbf{y})d\theta,$$

where $\theta^{(1)}, \ldots, \theta^{(J)}$ is a sequence of draws from the Markov chain, for almost all starting values $\theta^{(0)}$ [with respect to $\pi(\theta|\mathbf{y})$].

Thus, provided that invariance and irreducibility hold, empirical averages converge to the corresponding posterior expectations. As we explained in Chapter 1, in Bayesian inference many of the quantities of interest can be expressed as posterior expectations of suitably-chosen functions $t(\theta)$. Thus, Theorem 1 covers the vast majority of practical situations.

So the question now is: when constructing a Markov chain, how can we make sure that invariance and irreducibility hold? Invariance is not that easy to check but, fortunately, Gibbs sampling and Metropolis-Hastings algorithms always fulfill this property, so we do not need to check it. Irreducibility is something that we need to check for each particular Markov chain we construct. Remember that this boils down to checking that the chain is able to visit all areas of the parameter space that have positive posterior probability. So, basically, look at the range of values that each parameter can take and make sure that the chain you have constructed is able to visit all. This will happen naturally for the vast majority of chains that one would think of constructing, it is only pathological cases that could lead to problems of lack of irreducibility.

Finally, stronger distributional results are possible if cyclic behaviour of the chain is ruled out:

Definition 3. *An m-***cycle** *for a Markov chain with transition kernel $P(\theta, \theta')$ is a collection of sets $\{A_0, \ldots, A_{m-1}\}$ such that $P(\theta, A_{(i+1)mod(m)}) = 1$ for all $\theta \in A_i$. The* **period** *of the chain is defined as $d = \max\{m :$ an $m-$cycle exists$\}$. The chain is* **aperiodic** *if $d = 1$.*

Thus, a Markov chain is periodic if we can find a collection of sets such that the chain visits them in a periodic way: from A_0 it always moves to A_1, from A_1 to A_2,..., from A_{m-1} back to A_0, and so on... If this periodic behaviour is ruled out, then we obtain a stronger form of convergence than in Theorem 1.

Theorem 2. *Consider a Markov chain whose transition kernel $P(\theta, \theta')$ has $\pi(\theta|\mathbf{y})$ as invariant distribution, is irreducible for $\pi(\theta|\mathbf{y})$ and aperiodic. Then, $P^t(\theta^{(0)}, \cdot)$, the transition kernel corresponding to t consecutive iterations starting from $\theta^{(0)}$, converges in total variation distance to the posterior distribution when $t \to \infty$, for almost all $\theta^{(0)}$ [with respect to $\pi(\theta|\mathbf{y})$].*

Chapter 8

Appendix 2: Some Common Probability Distributions

$*$ A **univariate Normal** distribution on $Y \in \mathbb{R}$ with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 > 0$ is denoted by $N(\mu, \sigma^2)$, and the corresponding density function is:

$$f\left(y \,|\, \mu, \sigma^2\right) = \frac{1}{\sigma} \frac{1}{(2\pi)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(y-\mu)^2\right\}$$

$E[Y] = \mu$, $V[Y] = \sigma^2$.

Therefore, $N(\mu, 1/\omega)$ denotes a Normal distribution with mean μ, variance $1/\omega$ and density

$$f\left(y \,|\, \mu, \omega\right) = \omega^{1/2} \frac{1}{(2\pi)^{1/2}} \exp\left\{-\frac{\omega}{2}(y-\mu)^2\right\}.$$

$*$ A **Gamma distribution** on $Y > 0$ with shape parameter $\alpha > 0$ and rate parameter $\lambda > 0$ is denoted by Gamma(α, λ), and the corresponding density is:

$$f(y \,|\, \alpha, \lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)} y^{\alpha-1} \exp(-\lambda y) \quad \text{for } y > 0$$

$E[Y] = \frac{\alpha}{\lambda}$; $V[Y] = \frac{\alpha}{\lambda^2}$.

We note that Exponential(λ) = Gamma$(1, \lambda)$.

$*$ A **Poisson distribution** on $Y \in \{0, 1, 2, \dots\}$ with parameter $\theta > 0$ is denoted by Poisson(θ), and the corresponding probability function is:

$$f(y|\theta) = \frac{\exp(-\theta)\, \theta^y}{y!} \quad \text{for } y = 0, 1, 2, \dots$$

$E[Y] = V[Y] = \theta$.

$*$ A **Binomial distribution** on $Y \in \{0, 1, \ldots, n\}$ with parameters $n \in \{1, 2, 3, \ldots\}$ and $p \in (0, 1)$ is denoted by Binomial(n, p), and the corresponding probability function is:

$$f(y|n, p) = \binom{n}{y} p^y (1-p)^{n-y} \quad \text{for } y = 0, 1, \ldots, n$$

$E[Y] = np$; $V[Y] = np(1-p)$.

$*$ A **Beta distribution** on $Y \in (0, 1)$ with parameters $a, b > 0$ is denoted by Beta(a, b), and the corresponding density function is:

$$f(y|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} y^{a-1}(1-y)^{b-1} \quad \text{for } y \in (0, 1)$$

$E[Y] = \frac{a}{a+b}$; $V[Y] = \frac{ab}{(a+b)^2(a+b+1)}$.

$*$ A **Uniform distribution** on $Y \in (l, r)$ where $-\infty < l < r < \infty$ is denoted by Uniform(l, r), and the corresponding p.d.f. is:

$$f(y|l, r) = \frac{1}{r-l} \quad \text{for } y \in (l, r)$$

$E[Y] = \frac{l+r}{2}$; $V[Y] = \frac{(r-l)^2}{12}$.

$*$ A **k-variate ($k > 1$) Normal distribution** on $\mathbf{Y} \in \mathbb{R}^k$ with mean vector $\mathbf{b} \in \mathbb{R}^k$ and positive definite symmetric (PDS) $k \times k$ covariance matrix $\mathbf{C}$ is denoted by $N_k(\mathbf{b}, \mathbf{C})$, and the corresponding density function is:

$$f(\mathbf{y}|\mathbf{b}, \mathbf{C}) = \frac{1}{\{det(\mathbf{C})\}^{1/2}} \frac{1}{(2\pi)^{k/2}} \exp\left\{-\frac{1}{2}(\mathbf{y}-\mathbf{b})^T \mathbf{C}^{-1}(\mathbf{y}-\mathbf{b})\right\}$$

$E[\mathbf{Y}] = \mathbf{b}$; $Cov[\mathbf{Y}] = \mathbf{C}$, where $Cov[\mathbf{Y}]$ denotes the covariance matrix of $\mathbf{Y}$.