Durham University

MATH3421

Bayesian Computation and Modelling III

Michaelmas, 2023–24

Contents

Chapter 1

Preliminaries

1.1 Introduction and background

Markov chain Monte Carlo (MCMC) methods are a class of stochastic simulation algorithms that have been around since the 1940s, even though their impact on statistics was not truly felt until the very early 1990s, with the seminal paper Gelfand and Smith (1990) convincing the community that the methods are easy to understand and practical to implement. Since then, a considerable amount of research has been devoted to theory, methodological development and application of MCMC methods. MCMC utilises a Monte Carlo method based upon Markov chains (as the name suggests). MCMC is particularly powerful within the Bayesian paradigm, as a method to provide samples from a posterior distribution that might only be known up to a normalising constant.

We will cover three preliminary topics before discussing MCMC. These are:-

1. Bayesian Statistics (with a very brief overview – further details in MATH2711),
2. Monte Carlo methods,
3. Markov chains.

1.2 Bayes Theorem

For two events E and F, the *conditional probability* of E given F is:

$$\mathbb{P}(E|F) = \frac{\mathbb{P}(E \cap F)}{\mathbb{P}(F)}.$$

This gives us the simplest version of Bayes theorem:

$$\mathbb{P}(E|F) = \frac{\mathbb{P}(F|E)\mathbb{P}(E)}{\mathbb{P}(F)}.$$

If $E_1, E_2, \ldots, E_n$ is a partition i.e. the events E_i are mutually exclusive with $\mathbb{P}(E_1 \cup \ldots E_n) = 1$, then

$$\mathbb{P}(F) = \mathbb{P}\left((F \cap E_1) \cup (F \cap E_2) \cup \ldots \cup (F \cap E_n)\right) = \sum_{i=1}^{n} \mathbb{P}(F \cap E_i)$$
$$= \sum_{i=1}^{n} \mathbb{P}(F|E_i)\mathbb{P}(E_i).$$

This gives us the more commonly used version of Bayes theorem:

$$\mathbb{P}(E_i|F) = \frac{\mathbb{P}(F|E_i)\mathbb{P}(E_i)}{\sum_{j=1}^{n} \mathbb{P}(F|E_j)\mathbb{P}(E_j)}.$$

Bayes theorem is useful because it tells us how to turn probabilities around. Often we are able to understand the probability of outcomes F conditional on various hypotheses E_i. We can then compute probabilities of the form $\mathbb{P}(F|E_i)$. However, when we actually observe some outcome, we are interested in which hypothesis is most likely to be true, or in other words, we are interested in the probabilities of the hypotheses conditional on the outcome $\mathbb{P}(E_i|F)$. Bayes theorem tells us how to compute these posterior probabilities for the hypotheses, but we need to use our prior belief $\mathbb{P}(E_i)$ for each hypothesis. In this way, Bayes theorem gives us a coherent way to update our prior beliefs into posterior probabilities for the hypotheses following an outcome F.

1.2.1 Discrete parameters and data

When our data and parameters have discrete distributions, the hypotheses E_i are values for our parameter(s) θ, while the outcome F is a set of data $\boldsymbol{X}$. Bayes theorem is then

$$\mathbb{P}(\theta|\boldsymbol{X} = \boldsymbol{x}) = \frac{\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|\theta)\mathbb{P}(\theta)}{\mathbb{P}(\boldsymbol{X} = \boldsymbol{x})}$$

where the denominator can be calculated as above.

Example 1.2.1. Due to inaccuracies in drug testing procedures (e.g., false positives and false negatives), in the medical field the results of a drug test represent only one factor in a physician's diagnosis. Yet when Olympic athletes are tested for illegal drug use (i.e. doping), the results of a single test are used to ban the athlete from competition. In *Chance* (Spring 2004), University of Texas biostatisticians D. A. Berry and L. Chastain demonstrated the application of Bayes's Rule for making inferences about testosterone abuse among Olympic athletes. They used the following example. In a population of 1000 athletes, suppose 100 are illegally using testosterone. Of the users, suppose 50 would test positive for testosterone. Of the nonusers, suppose 9 would test positive.

If an athlete tests positive for testosterone, use Bayes theorem to find the probability that the athlete is really doping.

Let $\theta = 1$ be the event that the athlete is a user, and $\theta = 0$ be a nonuser; and let $x = 1$ be the event of a positive test result. We need to consider the following questions.

- What is the prior information for θ?
- What is the observation?
- What is the posterior distribution of θ after we get the data?

	Prior dist.		*Posterior dist.*
θ	$\mathbb{P}(\theta)$	$\mathbb{P}(x=1\|\theta)$	$\mathbb{P}(\theta\|x=1)$
1			
0			

1.2.2 Continuous parameters and data

We need to be slightly more careful when we work with continuous parameters and data, because strictly speaking, the probabilities in the equations above become zero. The basic rule is to replace probabilities with density functions and summation with integration.

For continuous problems, the hypotheses are represented by a continuous parameter (or parameter vector) θ, and the outcomes are observed data $\boldsymbol{X}$. (Note that both data and parameters are random variables.) Our prior beliefs about the parameters define the prior distribution for θ, specified by a *density* $\pi(\theta)$. We formulate a model that defines the distribution of $\boldsymbol{X}$ given the parameters, i.e. we specify a density $f(\boldsymbol{x}|\theta)$. This can be regarded as a function of θ when we have got some fixed observed data $\boldsymbol{x}$, called the *likelihood*:

$$\ell(\theta|\boldsymbol{x}) = f(\boldsymbol{x}|\theta).$$

The prior and likelihood determine the full *joint density* over data and parameters:

$$f(\theta, \boldsymbol{x}) = \pi(\theta)\ell(\theta|\boldsymbol{x}).$$

(It is easy to check that this integrates to 1, so it really is a well-defined density). Given the joint density we are then able to compute its marginals and conditionals:

$$f(\boldsymbol{x}) = \int f(\theta, \boldsymbol{x})\ d\theta = \int \pi(\theta)\ell(\theta|\boldsymbol{x})\ d\theta$$

and

$$f(\theta|\boldsymbol{x}) = \frac{f(\theta, \boldsymbol{x})}{f(\boldsymbol{x})} = \frac{\pi(\theta)\ell(\theta|\boldsymbol{x})}{\int \pi(\theta)\ell(\theta|\boldsymbol{x})\ d\theta}.$$

$f(\theta|\boldsymbol{x})$ is known as *the posterior density*, and is usually denoted $\pi(\theta|\boldsymbol{x})$, leading to the continuous version of Bayes theorem:

$$\boxed{\pi(\theta|\boldsymbol{x}) = \frac{\pi(\theta)\ell(\theta|\boldsymbol{x})}{\int \pi(\theta)\ell(\theta|\boldsymbol{x})\ d\theta}}$$

Now, the denominator is not a function of θ, so we can in fact write this as

$$\boxed{\pi(\theta|\boldsymbol{x}) \propto \pi(\theta)\ell(\theta|\boldsymbol{x})}$$

where the constant of proportionality is chosen to ensure that the density integrates to one. Hence, *the posterior is proportional to the prior times the likelihood*.

1.2.3 Bayesian computation

In principal, the previous sections cover the basics about Bayesian inference, specifying the posterior distribution of the parameters given the data. However, the posterior distribution may be far from trivial to work with. Typically, we are able to write down in closed form the numerator of Bayes theorem:

$$\text{numerator} = \pi(\theta)\ell(\theta|\boldsymbol{x}).$$

The numerator is *not* itself a density function: it does not integrate to 1 with respect to θ. A function like this, i.e. one that is an unknown multiple of a true density function, is called a *kernel*. So while we can usually write down a kernel of the posterior distribution (i.e. we can write down the numerator in closed form), working out the denominator can be impossible analytically. The denominator has the form

$$\text{denominator} = \int \pi(\theta)\ell(\theta|\boldsymbol{x})\ d\theta$$

and so is an integral (or summation in the discrete case) over the space of parameters. We can try to evaluate this numerically, but even that may be very hard to do because

- the integral may be very high dimensional, i.e. there might be many parameters, and
- the support of θ may be very complicated.

The support of a random variable is the set of values on which its density is non-zero. Similar problems arise when we want to marginalize or work out an expectation:

$$\mathbb{E}(\theta|\boldsymbol{x}) = \int \theta\pi(\theta|\boldsymbol{x})\ d\theta.$$

Both types of integral – the denominator in Bayes theorem, and the expectation – can effectively be evaluated via stochastic simulation, as we will see. First, however, let's look at the kind of situation where these integrals do not have closed analytic forms.

Example 1.2.2. Consider the case where we have a collection of observations, $X_i \sim N(\mu, \tau^{-1})$, which we believe to be conditionally independent (given the parameters) and identically distributed (iid). We write

$$X_i|\mu,\tau \sim N(\mu, 1/\tau).$$

The likelihood for a single observation is

$$\ell(\mu,\tau|x_i) = f(x_i|\mu,\tau) = \sqrt{\frac{\tau}{2\pi}}\exp\left\{-\frac{\tau}{2}(x_i-\mu)^2\right\}$$

and so for n independent observations, $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ is

$$\begin{aligned}
\ell(\mu, \tau|\boldsymbol{x}) = f(\boldsymbol{x}|\mu, \tau) &= \prod_{i=1}^{n} \sqrt{\frac{\tau}{2\pi}} \exp\left\{-\frac{\tau}{2}(x_i - \mu)^2\right\} \\
&= \left(\frac{\tau}{2\pi}\right)^{\frac{n}{2}} \exp\left\{-\frac{\tau}{2}\sum_{i=1}^{n}(x_i - \bar{x} + \bar{x} - \mu)^2\right\} \\
&\propto \tau^{\frac{n}{2}} \exp\left\{-\frac{\tau}{2}\left[\sum_{i=1}^{n}(x_i - \bar{x})^2 + n(\bar{x} - \mu)^2\right]\right\} \\
&\propto \tau^{\frac{n}{2}} \exp\left\{-\frac{n\tau}{2}\left[s^2 + (\bar{x} - \mu)^2\right]\right\}
\end{aligned}$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

For a Bayesian analysis, we need also to specify prior distributions for the parameters, μ and τ. There is a conjugate analysis for this problem based on the specifications:

$$\begin{aligned}
\tau &\sim Gamma(g, h) \\
\mu|\tau &\sim N\left(b, \frac{1}{c\tau}\right).
\end{aligned}$$

However, this specification is rather unsatisfactory — μ and τ are not independent, and in many cases our prior beliefs for μ and τ will separate into independent specifications. For example, we may prefer to specify independent priors for the parameters:

$$\begin{aligned}
\tau &\sim Gamma(g, h) \\
\mu &\sim N\left(b, \frac{1}{c}\right).
\end{aligned}$$

However, this specification is no longer conjugate, making analytic analysis intractable. Let us see why:

$$\begin{aligned}
\pi(\mu) &= \sqrt{\frac{c}{2\pi}} \exp\left\{-\frac{c}{2}(\mu - b)^2\right\} \\
&\propto \exp\left\{-\frac{c}{2}(\mu - b)^2\right\}
\end{aligned}$$

and

$$\begin{aligned}
\pi(\tau) &= \frac{h^g}{\Gamma(g)}\tau^{g-1}\exp\{-h\tau\} \\
&\propto \tau^{g-1}\exp\{-h\tau\}
\end{aligned}$$

so

$$\pi(\mu, \tau) \propto \tau^{g-1} \exp\left\{-\frac{c}{2}(\mu - b)^2 - h\tau\right\}$$

giving

$$\pi(\mu,\tau|\boldsymbol{x}) \propto \tau^{g-1}\exp\left\{-\frac{c}{2}(\mu-b)^2 - h\tau\right\} \times \tau^{\frac{n}{2}}\exp\left\{-\frac{n\tau}{2}\left[s^2+(\bar{x}-\mu)^2\right]\right\}$$
$$= \tau^{g+\frac{n}{2}-1}\exp\left\{-\frac{n\tau}{2}\left[s^2+(\bar{x}-\mu)^2\right] - \frac{c}{2}(\mu-b)^2 - h\tau\right\}.$$

The posterior density for μ and τ certainly won't factorise (μ and τ are not independent *a posteriori*), and will not even separate into the form of the conditional Normal-Gamma conjugate form mentioned earlier.

So, we have the kernel of the posterior for μ and τ, but it is not in a standard form. We can gain some idea of the likely values of (μ,τ) by plotting the bivariate surface (the integration constant isn't necessary for that), but we cannot work out the posterior mean or variance, or the forms of the marginal posterior distributions for μ or τ, since we cannot integrate out the other variable. We need a way of understanding posterior densities which does not rely on being able to analytically integrate the posterior density.

In fact, there is nothing particularly special about the fact that the density represents a Bayesian posterior. Given any complex non-standard probability distribution, we need ways to understand it, to calculate its moments, to compute its conditional and marginal distributions and their moments. Stochastic simulation is one possible solution.

1.2.4 Notation recap

The notation used above for the continuous parameters / data case can easily be used for the discrete (or mixed) case, leading to the notation:

- $\boldsymbol{x}$ data,
- θ model parameter (or parameters $\boldsymbol{\theta}$),
- $f(\boldsymbol{x}|\theta)$ probability model for the data,
- $\pi(\theta)$ prior: pdf/pmf representing beliefs before examining the data,
- $\pi(\theta|\boldsymbol{x})$ posterior: pdf/pmf of distribution after examining the data.

Note that in other sources (e.g. textbooks) you will see different notations for the prior, posterior and model. For example, the prior might be $f(\theta)$ or $p(\theta)$; the posterior might be $f(\theta|\boldsymbol{x})$ or $p(\theta|\boldsymbol{x})$ and the model might be $p(\boldsymbol{x}|\theta)$ or $\pi(\boldsymbol{x}|\theta)$. Usually the easiest way to translate the notation is to examine the arguments.

1.3 Monte Carlo methods

The rationale for stochastic simulation can be summarised very easily: to understand a statistical model simulate many realisations from it and study them. For example, consider a bivariate density $f_{X,Y}(x,y)$ and suppose that the marginals $f_X(x)$ and $f_Y(y)$ are difficult to compute. If we can simulate lots of realisations of X and Y then we can look at histograms of the X and Y

values, to get an idea of the marginals. We can also look at the sample mean and variance of the X values (for example) to find out the mean and variance of the marginal for X.

In statistics, the term *Monte Carlo* means *simulation*.

1.3.1 Monte Carlo integration

Suppose we want to evaluate an integral of the form

$$\mu_h = \int_{\mathcal{X}} h(x) f(x) dx$$

for some function $h(\cdot)$ and some pdf $f(\cdot)$ with support $\mathcal{X}$. It should be clear that $\mu_h = \mathbb{E}[h(X)]$. By the law of large numbers, we can estimate μ_h by taking a sample from the distribution of X and using the sample mean as an estimate of the theoretical mean. In particular,

1. Let $X_1, X_2, \ldots, X_N$ be iid draws from f.
2. Then, $\hat{\mu}_h = \frac{1}{N} \sum_{i=1}^{N} h(X_i)$ is an *unbiased Monte Carlo estimator* of μ_h.

Note that the unbiasedness property follows directly by linearity. Moreover, provided $\sigma_h^2 = \mathrm{Var}[h(X)] < \infty$ exists,

$$\begin{aligned} \mathrm{Var}(\hat{\mu}_h) &= \frac{1}{N^2} \mathrm{Var}\left[\sum_{i=1}^{N} h(X_i) \right] \\ &= \frac{\sigma_h^2}{N}. \end{aligned}$$

Hence, via the *Central Limit Theorem*

$$\frac{\sqrt{N}(\hat{\mu}_h - \mu_h)}{\sigma_h} \xrightarrow{D} N(0,1)$$

that is,

$$\hat{\mu}_h \approx N\left(\mu_h, \sigma_h^2/N\right).$$

The above properties hold provided the $\{X_i\}$ are independent. Note that the size of the error in $\hat{\mu}_h$ is proportional to the standard deviation of the estimator. Hence, this error (or equivalently, speed of convergence of the estimator) is $\mathcal{O}(N^{-1/2})$[1], rather than $\mathcal{O}(N^{-2/d})$, as would be obtained using simple numerical integration (and note that d is the dimension of X).

Further motivation

Almost everything we may wish to calculate about the distribution of X can be represented by an expectation, for example

- expectation $\mathbb{E}(X)$,
- variance $\mathrm{Var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$
- cdf $F(X) = \mathbb{P}(X \leq x) = \mathbb{E}[I(X \leq x)]$ where $I(\cdot)$ is the indicator function,
- ...

[1] A sequence x_n is $\mathcal{O}(g_n)$ as $n \to \infty$ if for all sufficiently large n, $|x_n| \leq b g_n$ for some $b < \infty$.

1.3.2 Importance sampling

Consider $\mu_h = \mathbb{E}[h(X)]$ as above but suppose that $f(x)$ is difficult to sample from. Suppose we can find a *proposal density* $g(x)$ such that $g(x) > 0$ for all x with $f(x) \geq 0$, and that is easy to sample from. Then consider

$$\mu_h = \int_{\mathcal{X}} h(x)f(x)dx = \int_{\mathcal{X}} \frac{h(x)f(x)}{g(x)} g(x)dx$$

which we recognise as $\mathbb{E}_g[h(X)f(X)/g(X)]$ where the subscript makes clear that the expectation is with respect to $g(x)$. It is convenient to write this expectation as $\mathbb{E}_g[h(X)w(X)]$ where $w(X) = f(X)/g(X)$ is known as the *weight function*. Hence, given iid draws $X_1, \ldots, X_N$ from g, an unbiased and consistent *importance sampling estimator* of μ_h can be constructed as

$$\hat{\mu}_h = \frac{1}{N} \sum_{i=1}^{N} h(X_i)w(X_i).$$

Example 1.3.1. Suppose we wish to compute $\mathbb{E}[X^2]$ where $X \sim N(0, 1)$. We take $g(x)$ as the density associated with a Student's t-distribution.

```
## Example: f=N(0,1), g=Student-t_df, h(x)=x^2
set.seed(3421) ## for reproducibility
N=100; df=5
xs=rt(N,df=df)
fdens=dnorm(xs); gdens=dt(xs,df=df)
hs=xs^2 ## E[hs]=1
mean(hs * fdens/gdens) ## true expectation is 1

## [1] 0.9191474
```

Note that in general, the *importance weights* $w(X_1), \ldots, w(X_N)$ do not add up to one. We may therefore wish to consider the *self-normalised estimator*

$$\tilde{\mu}_h = \frac{1}{\sum_{i=1}^{N} w(X_i)} \sum_{i=1}^{N} h(X_i)w(X_i).$$

Remarks:

- The estimator is *biased* (proof omitted) but *consistent*. To see consistency, note that

$$\begin{aligned}\tilde{\mu}_h &= \frac{\frac{1}{N}\sum_{i=1}^N h(X_i)w(X_i)}{\frac{1}{N}\sum_{i=1}^N w(X_i)} \\ &= \frac{\frac{1}{N}\sum_{i=1}^N h(X_i)\frac{f(X_i)}{g(X_i)}}{\frac{1}{N}\sum_{i=1}^N \frac{f(X_i)}{g(X_i)}} \\ &\to \frac{\mathbb{E}_g[h(X)f(X)/g(X)]}{\mathbb{E}_g[f(X)/g(X)]} \qquad \text{as } N\to\infty, \text{ using the SLLN} \\ &= \frac{\int_{\mathcal{X}} h(x)f(x)dx}{\int_{\mathcal{X}} f(x)dx} \\ &= \mathbb{E}_f[h(X)]\end{aligned}$$

  as required.

- An advantage of using the self-normalised estimator is that it can be applied when $f(\cdot)$ is only known up to a multiplicative constant (as is often the case for a Bayesian posterior). To see this, replace $f(x)$ with $\tilde{f}(x)/k$ (where $\tilde{f}$ is an unnormalised kernel) in the expression for $\tilde{\mu}_h$ and note that the unknown normalising constant k cancels. In this case, we may use the *unnormalised* weight function $w(X) = \tilde{f}(X)/g(X)$ directly in the expression for $\tilde{\mu}_h$.

Example 1.3.2. Suppose we wish to compute $\mathbb{E}[X^2]$ where $X \sim N(0,1)$ but we can only evaluate the kernel $\exp(-x^2/2)$. We take $g(x)$ as the density associated with a Student's t-distribution.

```
## Example: f=unnormalised N(0,1), g=Student-t_df, h(x)=x^2
set.seed(3421) ## for reproducibility
N=100
xs=rt(N,df=df)
ws=exp(-xs^2/2)/dt(xs,df=df)
wsn=ws/sum(ws)  ## normalise the weights
hs=xs^2
sum(hs * wsn) ## true expectation is 1

## [1] 0.9020084
```

1.3.3 Weighted resampling

Since the choice of $h(\cdot)$ used above was arbitrary, we can use the sample and normalised weights to approximate any (suitable) expectation with respect to $f(x)$. Let $\tilde{w}(x_i) = w(x_i)/\sum_{j=1}^N w(x_j)$ denote the normalised weight. We can then view $\{x_i, \tilde{w}(x_i)\}_{i=1}^N$ as an *empirical approximation* of $f(x)$ via

$$\hat{f}(x) = \sum_{i=1}^N \tilde{w}(x_i)\delta(x - x_i)$$

where δ denotes the Dirac mass function, and is a convenient way of representing a discrete distribution as a pdf. The important point to note here is that we have approximated a continuous distribution $f(x)$ via a discrete distribution taking values $\{x_1, \ldots, x_N\}$ with associated probabilities $\{\tilde{w}(x_1), \ldots, \tilde{w}(x_N)\}$.

Remark: we may wish to use the weighted sample to construct summaries such as the mean, variance etc. This is most easily achieved by resampling (with replacement) from the discrete distribution on $\{x_1, \ldots, x_N\}$ with associated probabilities $\{\tilde{w}(x_1), \ldots, \tilde{w}(x_N)\}$. The result is an *equally weighted sample, approximately distributed according to* f. This technique is known as *weighted resampling*.

Example 1.3.3. Suppose that we wish to use weighted resampling to generate draws of a random variable X that follows a $N(0,1)$ distribution, truncated between -2 and 2. Furthermore, suppose that we take a $U(-2,2)$ proposal distribution. Note that X has pdf

$$f(x) \propto e^{-\frac{1}{2}x^2}, \qquad -2 < x < 2.$$

The proposal pdf is

$$g(x) \propto 1, \qquad -2 < x < 2.$$

The normalised weights therefore take the form

$$\tilde{w}(x_i) = \frac{e^{-\frac{1}{2}x_i^2}}{\sum_{j=1}^{N} e^{-\frac{1}{2}x_j^2}}, \; i = 1, \ldots N\,.$$

The following R code plots the target, proposal and implements the algorithm.

```
x=seq(-2,2,0.01)
plot(x,dnorm(x)/(pnorm(2)-pnorm(-2)),type="l",ylab="Density",xlab="x")
lines(x,dunif(x,-2,2),type="l",lty=2)
```

```
##Weighted resampling function
wr=function(N)
{
 xs=runif(N,-2,2)
 ws=exp(-xs^2/2)
 x=sample(xs,N,TRUE,ws)  #R can handle the unnormalised weights
 return(x)
}

## Run
par(mfrow=c(3,2))
hist(wr(50),freq=F,main="(a)",ylim=c(0,0.41))
lines(x,dnorm(x)/(pnorm(2)-pnorm(-2)),type="l")
```

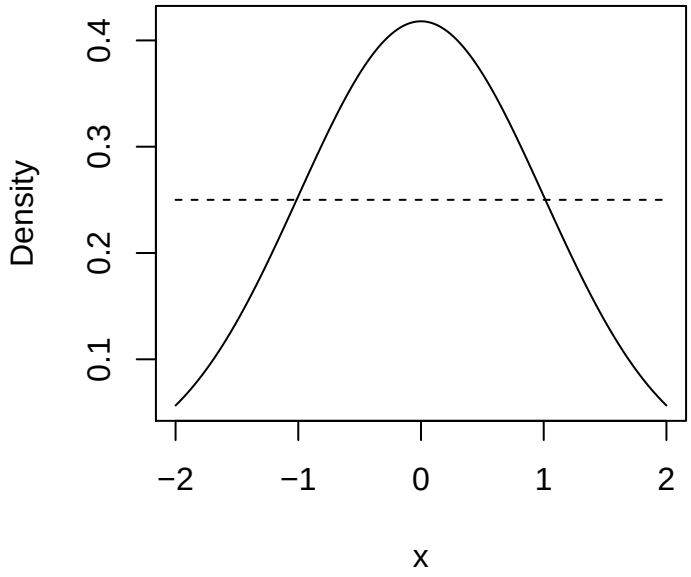

Figure 1.1: Truncated $N(0,1)$ target density (solid) and $U(-2,2)$ proposal density (dotted).

```
hist(wr(100),freq=F,main="(b)")
lines(x,dnorm(x)/(pnorm(2)-pnorm(-2)),type="l")
hist(wr(500),freq=F,main="(c)",ylim=c(0,0.41))
lines(x,dnorm(x)/(pnorm(2)-pnorm(-2)),type="l")
hist(wr(1000),freq=F,main="(d)")
lines(x,dnorm(x)/(pnorm(2)-pnorm(-2)),type="l")
hist(wr(10000),freq=F,main="(e)",ylim=c(0,0.41))
lines(x,dnorm(x)/(pnorm(2)-pnorm(-2)),type="l")
hist(wr(100000),freq=F,main="(f)")
lines(x,dnorm(x)/(pnorm(2)-pnorm(-2)),type="l")
```

It is clear from the plots below that provided N is chosen to be sufficiently large, the sample of N points generated after resampling are a good approximation to f.

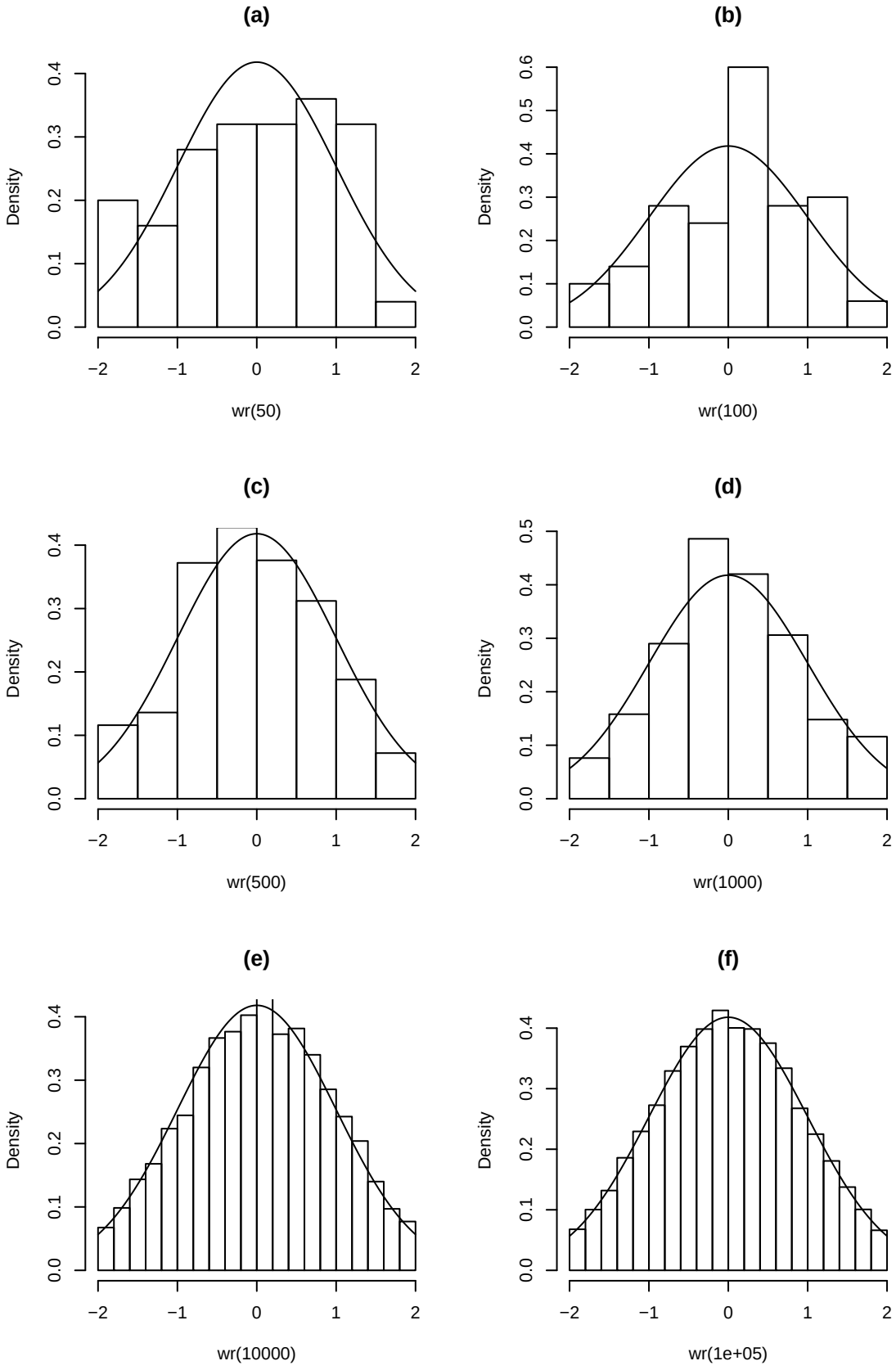

Figure 1.2: Target (solid) and histograms of samples using (a) N=50, (b) N=100, (c) N=500, (d) N=1000, (e) N=10000 and (f) N=100000.

Drawbacks of Monte Carlo integration

1. The variance, $\sigma_h^2 = \mathrm{Var}[h(X)]$ is often large. This problem can be circumvented to some extent by careful choice of $g(x)$ in importance sampling.
2. It can be difficult to simulate from X (in an efficient manner). This is a harder problem to deal with and this is where MCMC will be use- ful. Simulating from univariate distributions (e.g., by rejection sampling or inversion of the cdf) is usually fairly straightforward. For multivariate distributions, however, the problem becomes a lot harder as methods such as rejection sampling perform much worse as the dimension of the problem grows. This means a lot of work is required to get even a small sample from the distribution X.

1.4 Markov chains

Here, we will briefly look at the concepts that underpin Markov chains that are needed to understand MCMC. Note that we will barely scratch the surface of the vast theory underpinning Markov chains! For ease of notation, we will consider here univariate Markov chains but note that all results extend straightforwardly to the multivariate case.

Definition: *A (first order) Markov chain is a stochastic process $X_0, X_1, X_2, \ldots$ with the property that future states are independent of the past states given the present state. Formally, for any $A \subset \mathcal{X}$*

$$\begin{aligned}&\mathbb{P}\left(X_{n+1} \in A | X_n = x, X_{n-1} = x_{n-1}, \ldots, X_0 = x_0\right)\\ &\qquad = \mathbb{P}\left(X_{n+1} \in A | X_n = x\right), \quad \forall x, x_{n-1}, x_{n-2}, \ldots, x_0 \in \mathcal{X}.\end{aligned}$$

The state space $\mathcal{X}$ is just the set of possible values adopted by the random variables $X_0, X_1, X_2, \ldots$. The behaviour of the chain is therefore determined by the probabilities

$$\mathbb{P}\left(X_{n+1} \in A | X_n = x\right).$$

In general, this depends on n, A, and x. However, if there is no n dependence so that

$$\mathbb{P}\left(X_{n+1} \in A | X_n = x\right) = P(A|x), \quad \forall n$$

for a function P, then the Markov chain is said to be *homogeneous*, and the *transition kernel* $p(A|x)$ completely determines the behaviour of the chain. We will only consider homogeneous Markov chains.

We will start by illustrating the basic ideas for discrete state spaces. That is, $\mathcal{X} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$. Now, for all i, j let

$$p_{ij} = \mathbb{P}\left(X_{n+1} = j | X_n = i\right)$$

that is, the probability of moving from state i to state j in a single step, known as the *transition prbability*.

1.4.1 Transition matrices

If the state space is finite with $\mathcal{X} = \{1, \ldots, r\}$, then we can construct the *transition matrix*:

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1r} \\ \vdots & \ddots & \vdots \\ p_{r1} & \cdots & p_{rr} \end{pmatrix}.$$

For example, $p_{1r} = \mathbb{P}(X_{n+1} = r | X_n = 1)$ is the probability of moving from state 1 to state r in a single step. Note that the elements of P must be non-negative and the rows must sum to one: $\sum_j p_{ij} = 1$. Such matrices are called *stochastic matrices*.

Example 1.4.1. (Running example.) Suppose that the state space is $\{\text{Rain}, \text{Sunny}, \text{Cloudy}\}$ (ordered as the first, second and third states) and weather follows a homogeneous Markov process. Thus, the probability of tomorrow's weather simply depends on today's weather, and not any previous days. Suppose the transitions are given by

$$\begin{aligned} Pr(\text{Rain tomorrow} | \text{Rain today}) &= 0.5 \\ Pr(\text{Sunny tomorrow} | \text{Rain today}) &= 0.25 \\ Pr(\text{Cloudy tomorrow} | \text{Rain today}) &= 0.25 \end{aligned}$$

if it is raining today. Given today's weather state is Rain, we have a distribution on the state space for tomorrow's weather. The first row of the transition matrix becomes $(0.5, 0.25, 0.25)$. Suppose the full transition matrix is

$$P = \begin{pmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0 & 0.5 \\ 0.25 & 0.25 & 0.5 \end{pmatrix}.$$

1.4.2 Properties of Markov chains

Definition: *A Markov chain is irreducible if for all i and j, there exists some n such that $\mathbb{P}(X_n = j | X_0 = i) > 0$; i.e. every state is reachable (at some point) from every other state.*
Definition: *An irreducible Markov chain is aperiodic if the greatest common divider (gcd) of*

$\{n : \mathbb{P}(X_n = i | X_0 = i) > 0\}$ is 1. If the gcd is $d > 1$ then the Markov chain is periodic with period d. (The choice of i does not matter: for an irreducible chain the periodicity is the same whatever i is chosen). Quick example matrix 1:

$$P = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.2 & 0.8 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

This is an example of a reducible Markov chain since we can't get from state 1 to state 3, for example. *Quick example matrix 2:*

$$P = \begin{pmatrix} 0 & 0.5 & 0 & 0.5 \\ 0.2 & 0 & 0.8 & 0 \\ 0 & 0.2 & 0 & 0.8 \\ 0.1 & 0 & 0.9 & 0 \end{pmatrix}.$$

This is periodic, it returns to a given state after an even number of steps. (It alternates between odd states, 1 and 3, and even states, 2 and 4.)

1.4.3 Stationary distribution

Definition: *A stationary distribution is a set of probabilities π_i, $i = 1, 2, \ldots$ such that:*

$$\mathbb{P}(X_n = i) = \pi_i \quad \forall i \quad \Rightarrow \quad \mathbb{P}(X_{n+1} = i) = \pi_i \quad \forall i$$

i.e., a Markov chain which starts with a distribution of π will have the same distribution at the next time point.

Remark: a reducible chain can have more than one stationary distribution; but our interest lies in irreducible chains and, for an irreducible chain the stationary distribution is always unique (if it exists).

Now, let the row vector $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots)$. If $\mathbb{P}(X_n = i) = \pi_i \ \forall i$, then we say that X_n is distributed as $\boldsymbol{\pi}$ and write $X_n \sim \boldsymbol{\pi}$. It should be clear that by induction, if $X_n \sim \boldsymbol{\pi}$ then $X_{n+k} \sim \boldsymbol{\pi} \ \forall k \geq 0$. Moreover, we have that

$$\begin{aligned} \mathbb{P}(X_{n+1} = j) &= \sum_i \mathbb{P}(X_{n+1} = j, X_n = i) \\ &= \sum_i \mathbb{P}(X_{n+1} = j | X_n = i) \mathbb{P}(X_n = i) \\ &= \sum_i p_{ij} \mathbb{P}(X_n = i). \end{aligned}$$

Now, if $X_n \sim \boldsymbol{\pi}$, then the above can be written as $\pi_j = \sum_i \pi_i p_{ij} \ \forall j$. Or, written more succinctly as

$$\boldsymbol{\pi} = \boldsymbol{\pi} P.$$

Example 1.4.2. (Example 1.4.1 revisited.) Consider again the three state weather example with

$$P = \begin{pmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0 & 0.5 \\ 0.25 & 0.25 & 0.5 \end{pmatrix}.$$

We can find the stationary distribution by first writing out $\boldsymbol{\pi} = \boldsymbol{\pi} P$ explicitly to give

$$\begin{aligned} \pi_1 &= 0.5\pi_1 + 0.5\pi_2 + 0.25\pi_3 \\ \pi_2 &= 0.25\pi_1 + 0.25\pi_3 \\ \pi_3 &= 0.25\pi_1 + 0.5\pi_2 + 0.5\pi_3 \end{aligned}$$

for which the first and third equations give $\pi_1 = \pi_3$. Hence set $\pi_1 = \pi_3 = \pi^*$ and sub into equation 2 to give $\pi_2 = 0.5\pi^*$. Hence $\boldsymbol{\pi} = (\pi^*, 0.5\pi^*, \pi^*)$. Finally, we use the crucial property that the elements of $\boldsymbol{\pi}$ sum to 1 to deduce $\boldsymbol{\pi} = (2/5, 1/5, 2/5)$.

1.4.4 Limiting distribution

Definition: *A Markov chain $\{X_n\}$ has a limiting distribution (also called its asymptotic distribution), $\boldsymbol{\pi}$, if for all i and j, $\lim_{n\to\infty} \mathbb{P}(X_n = j|X_0 = i) = \pi_j$ where $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots)$.*

Now, by the Markov property we have for all i, j that

$$\begin{aligned}\mathbb{P}(X_{n+1} = j|X_0 = i) &= \sum_k \mathbb{P}(X_{n+1} = j|X_n = k, X_0 = i)\mathbb{P}(X_n = k|X_0 = i)\\ &= \sum_k \mathbb{P}(X_{n+1} = j|X_n = k)\mathbb{P}(X_n = k|X_0 = i)\\ &= \sum_k p_{kj}\mathbb{P}(X_n = k|X_0 = i).\end{aligned}$$

Remarks:

- If the limiting distribution exists then letting $n \to \infty$ we obtain $\pi_j = \sum_k p_{kj}\pi_k$. Equivalently, $\boldsymbol{\pi} = \boldsymbol{\pi}P$: the asymptotic distribution of the chain, $\boldsymbol{\pi}$, is also the stationary distribution of the chain.
- Substituting $n = 1$ into the above expression for $\mathbb{P}(X_{n+1} = j|X_0 = i)$ gives

$$\mathbb{P}(X_2 = j|X_0 = i) = \sum_k p_{kj}p_{ik}, \quad \forall i, j$$

  which we recognise as P^2 (and recall that P is the transition matrix). In general, we obtain the n-step transition matrix as P^n.

Example 1.4.3. (Example 1.4.1 revisited.) Compute $\mathbb{P}(X_n = j|X_0 = Rain)$, $n = 1, \ldots, 5$, for $j \in \{1, 2, 3\}$ corresponding to each weather state Rain, Sunny, Cloudy.

```
## Example: weather, limiting distribution demo

Xprob=c(1,0,0) ## Rain at time 0
## set up transition matrix
P=matrix(c(0.5,0.25,0.25,0.5,0,0.5,0.25,0.25,0.5),ncol=3,byrow=TRUE)
for (i in 1:5)
{
 Xprob=Xprob%*%P
 print(Xprob)
}
```

```
##      [,1] [,2] [,3]
## [1,]  0.5 0.25 0.25
##        [,1]   [,2]  [,3]
## [1,] 0.4375 0.1875 0.375
##         [,1]     [,2]     [,3]
## [1,] 0.40625 0.203125 0.390625
##           [,1]      [,2]      [,3]
## [1,] 0.4023438 0.1992188 0.3984375
##           [,1]      [,2]      [,3]
## [1,] 0.4003906 0.2001953 0.3994141
```

Theorem: *If $\{X_n\}$ is an aperiodic, irreducible Markov chain on a countable statespace with a proper stationary distribution of $\boldsymbol{\pi}$ then for all i and j,*

$$\lim_{n\to\infty} \mathbb{P}(X_n = j | X_0 = i) = \pi_j.$$

Remark: A *proper* distribution $\boldsymbol{\pi}$ has $\sum_i \pi_i = 1$.

1.4.5 Detailed balance

Definition: *A discrete-valued Markov chain with transition matrix P is said to satisfy detailed balance if there is a distribution $\boldsymbol{\nu}$ such that for all i and j,*

$$\nu_i p_{ij} = \nu_j p_{ji}.$$

This is a powerful idea, since checking detailed balance is often an easy way of confirming a probability distribution $\boldsymbol{\pi}$ is the stationary distribution of a Markov chain, as the following Lemma shows.

Lemma: *If a Markov chain with transition matrix P satisfies detailed balance with distribution $\boldsymbol{\nu}$ then $\boldsymbol{\nu} = \boldsymbol{\pi}$ is a stationary distribution of the chain.*

Proof: Take the detailed balance equation and sum both sides over j. For any i we obtain

$$\begin{aligned} \sum_j \nu_i p_{ij} &= \sum_j \nu_j p_{ji} \\ \Rightarrow \nu_i \sum_j p_{ij} &= \sum_j \nu_j p_{ji} \\ \Rightarrow \nu_i &= \sum_j \nu_j p_{ji} \end{aligned}$$

that is, the ith element of $\boldsymbol{\nu} = \boldsymbol{\nu} P$.

1.5 Aim of MCMC

The aim of MCMC is to construct a Markov chain whose stationary distribution is the posterior distribution we are interested in. Then, provided the Markov chain is started in stationarity, a sample from the Markov chain will form a sample from the posterior distribution of interest. It is important to check that the Markov chain we construct is *irreducible* and *aperiodic*.

Since we will mainly be interested in posterior distributions with continuous support, we need a few key results for continuous-valued Markov chains.

1.6 Continuous state-space Markov chains

Next we consider what happens when the state space $\mathcal{X}$ is continuous e.g. $\mathcal{X} \subset \mathbb{R}$. Note that time is still taken to be discrete. Fortunately, most of the results we have already seen in the discrete state-space setting carry over to the continuous case, in particular: sums become integrals and the *transition matrix* becomes a *transition kernel*.

1.6.1 Transition kernels

In a similar way to the discrete case, for a homogeneous chain we can define

$$P(A|x) = \mathbb{P}\left(X_{n+1} \in A | X_n = x\right).$$

For continuous state-spaces we always have $P(X_{n+1} = y|x) = 0$, so instead we define

$$\begin{aligned} P(y|x) &= \mathbb{P}\left(X_{n+1} \leq y | X_n = x\right) \\ &= \mathbb{P}\left(X_1 \leq y | X_0 = x\right) \end{aligned}$$

where the last equality follows from time homogeneity. This is called the *conditional cumulative distribution function*. It is the distributional form of the transition matrix for continuous state-space Markov chains, but we can also define the corresponding density

$$p(y|x) = \frac{\partial}{\partial y} P(y|x).$$

This defines the *density form* of the transition kernel of the chain. It is easier to use than the conditional cumulative distribution function, particularly when the state space is multidimensional, eg. $\mathcal{X} \subset \mathbb{R}^d$.

Example 1.6.1. (Running example.) Consider the AR(1) model:

$$X_{n+1} = \alpha X_n + \epsilon_{n+1}, \quad \epsilon_{n+1} \sim N(0, \sigma^2), \quad |\alpha| < 1.$$

It is clear that the conditional distribution of X_{n+1} given $X_n = x$ is

$$X_{n+1}|(X_n = x) \sim N(\alpha x, \sigma^2),$$

and that it does not depend on any other previous values. Thus, the AR(1) model is a Markov chain and its state-space is the set of real numbers, so it is a continuous Markov chain.

Also, the density form of the transition kernel is just

$$p(y|x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{y-\alpha x}{\sigma}\right)^2\right].$$

Stationary distribution and detailed balance

By analogy to the discrete state-space case, denote a stationary density by π. Then, π satisfies

$$\pi(y) = \int_{\mathcal{X}} p(y|x)\pi(x)dx$$

which is the continuous version of the discrete matrix equation $\boldsymbol{\pi} = \boldsymbol{\pi}P$. Similarly, the detailed balance equation becomes

$$\pi(x)p(y|x) = \pi(y)p(x|y) \quad \forall x, y \in \mathcal{X}.$$

Showing that stationarity can be deduced from this equation is left as an exercise.

Example 1.6.2. (Example 1.6.1 revisited.) Consider again the AR(1) process and suppose that $X_0 = 15$, $\alpha = 0.9$ and $\sigma^2 = 1 - 0.9^2 = 0.19$. The R code below can be used to simulate this process over 300 steps.

```
set.seed(3421)
x=15
nt=300
alpha=0.9
xs=rep(0,nt+1)
xs[1]=x
for (i in 1:nt)
{
 x=alpha*x+sqrt(1-alpha^2)*rnorm(1)
 xs[i+1]=x
}
par(mfrow=c(1,2))
plot(0:nt,xs,xlab="t",ylab="x")
plot(0:nt,xs,type="l",xlab="t",ylab="x")
```

Note that a simulation using a different seed (or without setting the seed) would produce a different realisation, but the certain features would be common. The chain starts at $X_0 = 15$ and heads towards 0 initially, then meanders about, mostly staying between -2 and 2. After fewer than 50 iterations it is very close to its asymptotic distribution. This chain has a proper stationary distribution which we may show is $N(0, 1)$. Suppose $X_n \sim N(0, 1)$, then

$$X_{n+1} = \alpha X_n + \epsilon_{n+1}, \quad \epsilon_{n+1} \sim N(0, \sigma^2)$$

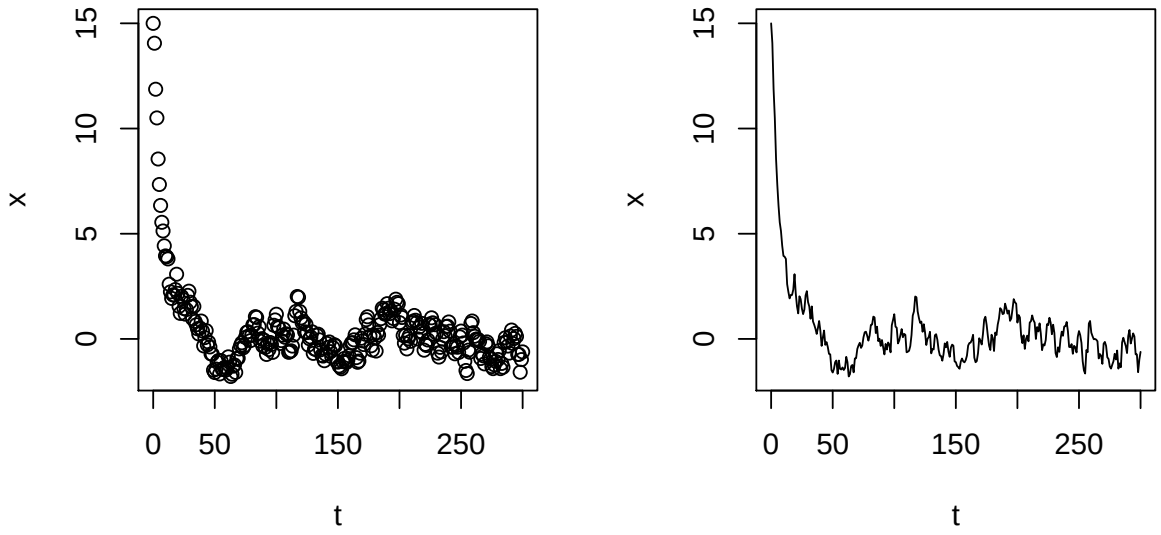

Figure 1.3: AR(1) simulation.

is the sum of two normal random variables giving $X_{n+1} \sim N(0, \alpha^2 + \sigma^2)$. Now, noting that in our example, $\sigma^2 = 1 - \alpha^2$, gives $X_{n+1} \sim N(0, 1)$. We will consider the general form of the stationary distribution for the AR(1) process in the tutorial questions at the end of this chapter.

1.6.2 Properties of continuous state-space Markov chains

Recall that for a continuous state-space $\mathcal{X}$, $\mathbb{P}(X = x) = 0$ for all $x \in \mathcal{X}$. Therefore, when considering definitions of aperiodicity and irreducibility in the context of continuous chains, we use subsets of $\mathcal{X}$, rather than individual values.

Definition: *A Markov chain,$\{X_n\}$, with a state-space $\mathcal{X}$ is μ-irreducible if there exists distribution, μ, on $\mathcal{X}$ such that for all $A \subseteq \mathcal{X}$ with $\mathbb{P}_\mu(X \in A) > 0$, and for all $x \in X$ there exists a positive integer $n = n(x, A)$ such that $\mathbb{P}(X_n \in A | X_0 = x) > 0$. (NB: μ could simply be taken to be π.)*

Definition: *A μ-irreducible Markov chain, $\{X_n\}$, with a state-space of $\mathcal{X}$ is aperiodic if there does not exist $d \geq 2$ and disjoint subsets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_d$ of the state-space $\mathcal{X}$ with $\mathbb{P}(X_{n+1} \in \mathcal{X}_{i+1} | X_n \in \mathcal{X}_i) = 1$ $(i = 1, \ldots, d-1)$ and $\mathbb{P}(X_{n+1} \in \mathcal{X}_1 | X_n \in X_d) = 1$, and such that $\mathbb{P}_\mu(X \in \mathcal{X}_i) > 0$. Otherwise the chain is periodic, with period d.*

Remark: The AR(1) process above is both aperiodic and μ-irreducible with $\mu = \pi$ because from any point x there is a non-zero chance of moving to any set A for which $\mathbb{P}_\pi(X \in A) > 0$.

Under μ-irreducibility and aperiodicity we have an analogous convergence result to that for discrete statespace Markov chains.

Theorem: (See e.g. Roberts and Rosenthal 2004.) *If a Markov chain on $\mathbb{R}^d$ or an open or closed subset of $\mathbb{R}^d$ is μ-irreducible and aperiodic, and has a proper stationary distribution, π, then for all $x \in \mathcal{X}$,*

$$\mathbb{P}(X_n \in A | X_0 \in x) \to \mathbb{P}_\pi(X \in A)$$

for all measurable $A \subseteq \mathcal{X}$.

Contents

Chapter 2

The Metropolis-Hastings algorithm

2.1 Introduction

Markov chain Monte Carlo (MCMC) is a generic tool for simulating analytically intractable distributions, which is particularly useful for Bayesian inference. As we have seen, given a large sample from some target $\pi(\boldsymbol{\theta})$, we can estimate almost any aspect of the distribution. The problem is obtaining that sample in the first place!

The MCMC strategy takes advantage of the fact that it is easy to simulate Markov chains:

- Construct a Markov chain with stationary distribution $\pi(\boldsymbol{\theta})$.
- Simulate a realization of this chain.
- After some 'burn-in' period (discussed later), take the realizations as (dependent) samples from $\pi(\boldsymbol{\theta})$.
- Use this sample to evaluate integrals / perform inference.

This chapter describes a fundamental algorithm that defines such chains: the *Metropolis-Hastings algorithm*.

Note that although interest here will typically be in sampling a posterior $\pi(\boldsymbol{\theta}|\boldsymbol{x})$, we will present the algorithm for a general target $\pi(\boldsymbol{\theta})$. Later, we will consider specific Bayesian problems for which interest will be in the posterior $\pi(\boldsymbol{\theta}|\boldsymbol{x})$.

2.1.1 Inception

Enrico Fermi and Stanislaw Ulam developed Monte Carlo methods for solving integrals in the early 20 th century. After WWII Ulam worked with John von Neumann do develop computational methods for exact independent sampling from univariate distributions. These simple Monte Carlo algorithms didn't work for complex problems. At Los Alamos, Nick Metropolis, along with Arianna Rosenbluth, Marshall Rosenbluth, Edward Teller and Augusta Teller found a way of generating dependent samples through Markov chains.

It's a beautifully simple idea. Explore your distribution by randomly walking through it. They published a paper in 1953, on what later became known as the Metropolis algorithm or random walk Metropolis. The Metropolis algorithm was later generalised by Hastings in 1970 and his

student Peskun (with papers in 1973 and 1981) as a statistical simulation tool that could overcome the curse of dimensionality met by regular Monte Carlo methods.

2.2 A first look at the Metropolis-Hastings algorithm

In addition to the target distribution with density $\pi(\boldsymbol{\theta})$, suppose we have some transition kernel $q(\cdot|\boldsymbol{\theta})$ which we call the *proposal* distribution and might depend on the current state of the chain (say $\boldsymbol{\theta}$). The proposal distribution will typically have the same support as $\pi(\cdot)$ and be easy to simulate from, but does not need to have $\pi(\cdot)$ as a stationary distribution.

The Metropolis-Hastings algorithm with N iterations is as follows:

1. Initialise the chain to $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_d^{(0)})'$ somewhere in the support of $\pi(\boldsymbol{\theta})$. Set the iteration counter $j = 1$.
2. Generate a *proposed* value $\boldsymbol{\theta}^*$ using the transition kernel $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$.
3. Evaluate the *acceptance probability* $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$ of the proposed move, defined by
$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\, q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})\, q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}.$$
4. Put $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^*$ with probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$; otherwise put $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)}$.
5. If $j = N$ stop, otherwise put j to $j + 1$ and go to step 2.

Remarks:

- At each stage a new value is generated from the proposal distribution. This is either accepted, in which case the chain moves, or rejected, in which case the chain stays at the same point.
- The target only enters into the acceptance probability as a ratio, and so the method can be used when the target is only known up to a multiplicative constant. Hence, if the target is a Bayesian posterior $\pi(\boldsymbol{\theta}|\boldsymbol{x})$, the acceptance probability can be written as
$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\, f(\boldsymbol{x}|\boldsymbol{\theta}^*)\, q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})\, f(\boldsymbol{x}|\boldsymbol{\theta})\, q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}.$$

Considerable freedom in the choice of proposal $q(\cdot|\boldsymbol{\theta})$ leaves us wondering what choices might be good, or generally quite useful. In particular we want a chain that

- converges rapidly, and
- *mixes* well ie. it
  - moves often and
  - moves well around the support of $\pi(\cdot)$.

Two commonly used special cases are discussed next.

2.2.1 Independence proposal / sampler

In this case the proposed transition is completely independent of the current position of the chain and so $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}^*)$. The acceptance probability becomes

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*)} \times \frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})}\right\}.$$

Example 2.2.1. (Prior as the proposal in a Bayesian setting.) For a target $\pi(\boldsymbol{\theta}|\boldsymbol{x}) \propto \pi(\boldsymbol{\theta})f(\boldsymbol{x}|\boldsymbol{\theta})$ and a proposal given by the prior $\pi(\boldsymbol{\theta})$, the acceptance probability becomes

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{f(\boldsymbol{x}|\boldsymbol{\theta}^*)}{f(\boldsymbol{x}|\boldsymbol{\theta})}\right\}$$

and hence only depends on the likelihood ratio of the candidate point and the current point. This scheme is likely to have a very high rejection rate unless the posterior is very similar to the prior.

Example 2.2.2. (Gaussian target and proposal.) Consider a target and proposal of the form

$$\pi(\theta) = \frac{1}{\sigma_p\sqrt{2\pi}}e^{-\theta^2/(2\sigma_p^2)},$$
$$q(\theta^*) = \frac{1}{\sigma_q\sqrt{2\pi}}e^{-(\theta^*)^2/(2\sigma_q^2)}$$

with $\sigma_p = 1$ and $\sigma_q = 2$. The initial value of the independence sampler is $\theta^{(0)} = 0$. The graphs below show the result of running the independence sampler for $N = 100$ iterations. The first plot shows the two densities, π and q, the second shows the Markov chain with positions joined by lines (known as a trace plot), the third shows a kernel density plot of the 100 chain values, and the fourth shows the chain again but overlays the proposals in red. Note that proposed points outside of the typical range of θ are likely to be rejected. Also note that from the bottom left panel, the approximation to the Gaussian density is far from perfect, and we might expect estimates obtained using our 101 samples to be poor. What might constitute a reasonable number of samples will be discussed in detail later.

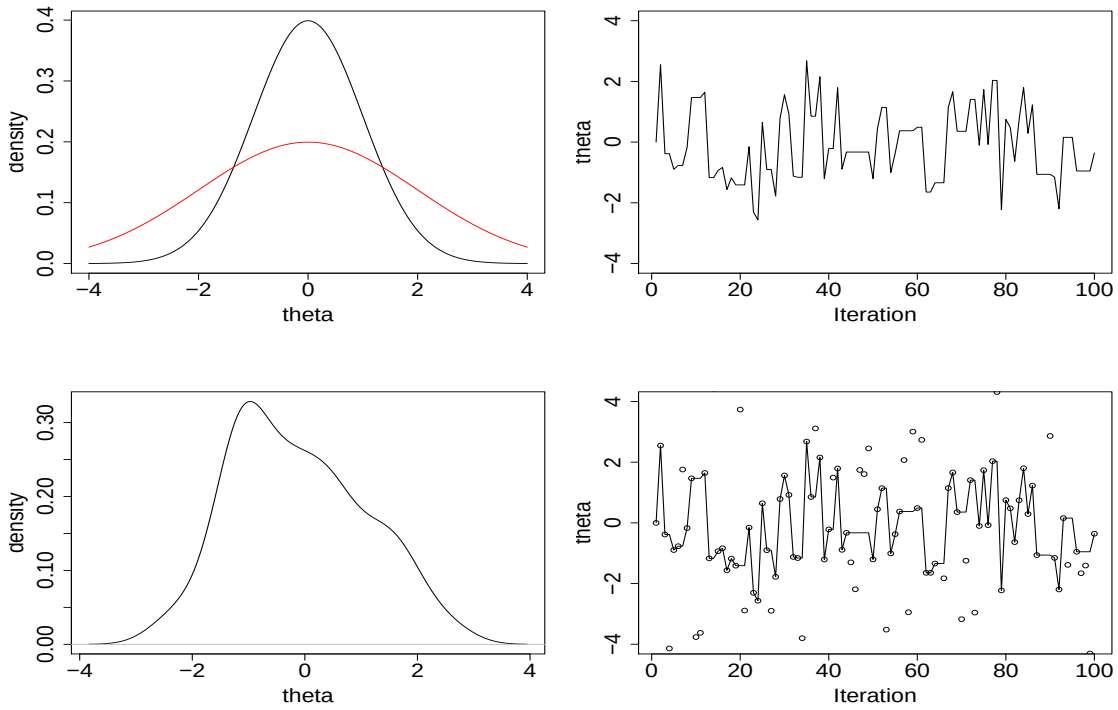

Figure 2.1: Independence sampler output. Top panel: target density (black) and proposal density (red), trace plot. Bottom panel: kernel density estimate, trace plot with proposed points overlaid.

2.2.2 Random walk Metropolis

We can define the proposed move at iteration j to be $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(j-1)} + \boldsymbol{w}^{(j)}$ where $\boldsymbol{w}^{(j)}$ is a $d \times 1$ random vector (completely independent of the state of the chain). Suppose that the $\boldsymbol{w}^j$ have density $g(\cdot)$ which is easy to simulate from. The proposal kernel is then $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = g(\boldsymbol{\theta}^* - \boldsymbol{\theta})$ and this can be used to calculate the acceptance probability. Of course, if $g(\boldsymbol{\theta}^* - \boldsymbol{\theta}) = g(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$, then we have a symmetric random walk chain, and the acceptance probability becomes

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})}\right\}$$

and hence does not involve the proposal density at all.

A commonly used method has $\boldsymbol{w}^{(j)} \sim N_d(\mathbf{0}, \lambda^2 I_d)$ where I_d denotes the $d \times d$ identity matrix and λ is 'tuned' to maximimse efficiency, a concept that we will revisit later in the notes.

Example 2.2.3. (Running example: Gaussian target and RWM method with Gaussian innovations.) Consider a target and proposal of the form

$$\pi(\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\theta^2/(2\sigma^2)},$$
$$q(\theta^*|\theta) = \frac{1}{\lambda\sqrt{2\pi}} e^{-(\theta^*-\theta)^2/(2\lambda^2)}.$$

with $\sigma = 1$ and $\lambda = 2$. The initial value of the RWM algorithm is $\theta^{(0)} = 0$.

The first plot shows the two densities, π, and q when $\theta = 1$, the second shows a trace plot of the Markov chain, the third shows a kernel density plot of the 101 chain values, and the fourth shows the trace plot again but overlays the proposals as points. Note that for a proposal, θ^*, well outside of the typical range of θ, $\pi(\theta^*) << \pi(\theta)$, so the proposed value is very likely to be rejected.

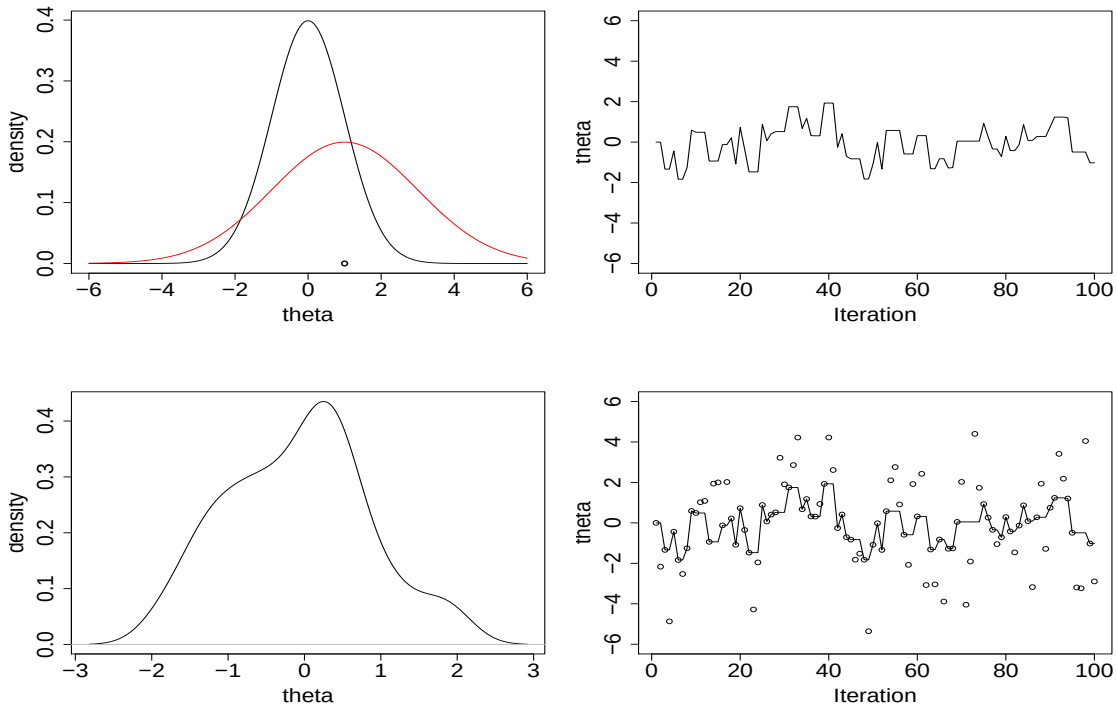

Figure 2.2: RWM output. Top panel: target density (black) and proposal density (red) at $\theta = 1$, trace plot. Bottom panel: kernel density estimate, trace plot with proposed points overlaid.

2.3 Validity

We first show that $\pi(\boldsymbol{\theta})$ is a stationary distribution of the Markov chain generated by the Metropolis-Hastings algorithm, before considering convergence to this stationary distribution.

2.3.1 Detailed balance

Recall that if a Markov chain satisfies detailed balance with respect to a distribution, π, then π is a stationary distribution for the chain. First we need to obtain the transition kernel of the Markov chain being simulated by the Metropolis-Hastings algorithm.

The kernel for the MH algorithm is not simply a probability density since there is often a non-zero probability that the proposal is rejected but including this simply complicates the notation without adding further insight.

Hence, assuming that the chain moves we have the transition kernel as

$$p(\boldsymbol{\phi}|\boldsymbol{\theta}) = \alpha(\boldsymbol{\phi}|\boldsymbol{\theta})q(\boldsymbol{\phi}|\boldsymbol{\theta}) \quad \text{when } \boldsymbol{\theta} \neq \boldsymbol{\phi}.$$

We can then check whether detailed balance holds:

$$\begin{aligned}\pi(\boldsymbol{\theta})p(\boldsymbol{\phi}|\boldsymbol{\theta}) &= \pi(\boldsymbol{\theta})q(\boldsymbol{\phi}|\boldsymbol{\theta})\min\left\{1, \frac{\pi(\boldsymbol{\phi})\, q(\boldsymbol{\theta}|\boldsymbol{\phi})}{\pi(\boldsymbol{\theta})\, q(\boldsymbol{\phi}|\boldsymbol{\theta})}\right\} \\ &= \min\left\{\pi(\boldsymbol{\theta})\, q(\boldsymbol{\phi}|\boldsymbol{\theta}), \pi(\boldsymbol{\phi})\, q(\boldsymbol{\theta}|\boldsymbol{\phi})\right\}.\end{aligned}$$

This expression is clearly symmetric in $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Hence, detailed balance holds and we may conclude that he Metropolis-Hastings algorithm defines a Markov chain with stationary distribution π.

2.3.2 Convergence

We need to show that the Markov chain is μ-irreducible and aperiodic. For simplicity we restrict ourselves to proposals q such that $q(\boldsymbol{\phi}|\boldsymbol{\theta}) > 0$ for all $\boldsymbol{\theta} \in \mathcal{X}$ and all $\boldsymbol{\phi} \in \mathcal{X}$ and densities $\pi(\boldsymbol{\theta}) \leq C$ everywhere, for some $C > 0$.

μ-irreducibility: We choose $\mu = \pi$ and note that since $\pi(\boldsymbol{\theta}) < \infty$ for all $\boldsymbol{\theta} \in \mathcal{X}$:

$$\begin{aligned}\mathbb{P}(\boldsymbol{\theta}^{(1)} \in A|\boldsymbol{\theta}^{(0)} = \boldsymbol{\theta}) &= \int_A q(\boldsymbol{\phi}|\boldsymbol{\theta})\min\left\{1, \frac{\pi(\boldsymbol{\phi})\, q(\boldsymbol{\theta}|\boldsymbol{\phi})}{\pi(\boldsymbol{\theta})\, q(\boldsymbol{\phi}|\boldsymbol{\theta})}\right\} d\boldsymbol{\phi} \\ &= \min\left\{\frac{q(\boldsymbol{\phi}|\boldsymbol{\theta})}{\pi(\boldsymbol{\phi})}\,,\, \frac{q(\boldsymbol{\theta}|\boldsymbol{\phi})}{\pi(\boldsymbol{\theta})}\right\}\pi(\boldsymbol{\phi})d\boldsymbol{\phi} \\ &\geq \frac{1}{C}\int_A \min\left\{q(\boldsymbol{\phi}|\boldsymbol{\theta})\,,\, q(\boldsymbol{\theta}|\boldsymbol{\phi})\right\}\pi(\boldsymbol{\phi})d\boldsymbol{\phi} \\ &= \frac{\mathbb{P}_\pi(A)}{C} \times \mathbb{E}_{\boldsymbol{\phi}\sim\pi^A}\left[\min\left\{q(\boldsymbol{\phi}|\boldsymbol{\theta})\,,\, q(\boldsymbol{\theta}|\boldsymbol{\phi})\right\}\right] \\ &> 0\end{aligned}$$

where π^A is π restricted to the set A.

Aperiodicity: wherever it is currently, the chain has a non-zero chance of moving to any set A where $\mathbb{P}_\pi(\boldsymbol{\theta} \in A) > 0$, so it could move from any set in a partition $\mathcal{X}_1, \ldots, \mathcal{X}_d$ to any other; hence it cannot be periodic.

2.4 Practical considerations

How do we implement MH in practice? How many iterations should we run the sampler for? Should we discard some iterations? We will endeavour to answer these questions in this section.

2.4.1 Accepting a move

Recall that the MH acceptance probability is

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\, q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})\, q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}.$$

Now let

$$R(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \frac{\pi(\boldsymbol{\theta}^*)\, q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})\, q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}$$

so that $\alpha = \min(1, R)$.

The simplest way to make the accept/reject decision is to generate a random number u from $U[0,1]$ distribution and accept if $u \leq \alpha$, rejecting otherwise. Since $\alpha \leq 1$ this is equivalent to accepting if $u \leq R$ and rejecting otherwise. In practice (for example when there is a lot of data) the target density can be so small or so large that calculating R can be prone to numerical issues. Since log is an increasing function $a \leq b \iff \log a \leq \log b$, so all practical implementations of Metropolis-Hastings algorithms accept if

$$\log u \leq \log R = \log \pi(\boldsymbol{\theta}^*) - \log \pi(\boldsymbol{\theta}) + \log q(\boldsymbol{\theta}|\boldsymbol{\theta}^*) - \log q(\boldsymbol{\theta}^*|\boldsymbol{\theta}).$$

2.4.2 Overall acceptance rate

We know that the probability of accepting a move given a current value $\boldsymbol{\theta}$ and a proposed value $\boldsymbol{\theta}^*$ is $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta})$. Hence, the probability of accepting a move, averaged over all possible proposed values is

$$\begin{aligned}\mathbb{P}(\text{move}|\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{\phi}\sim q(\boldsymbol{\phi}|\boldsymbol{\theta})}\left[\alpha(\boldsymbol{\phi}|\boldsymbol{\theta})\right] \\ &= \int \alpha(\boldsymbol{\phi}|\boldsymbol{\theta}) q(\boldsymbol{\phi}|\boldsymbol{\theta}) d\boldsymbol{\phi}.\end{aligned}$$

The *theoretical acceptance rate* for a Metropolis-Hastings chain at stationarity is

$$\begin{aligned}\mathbb{P}(\text{move}) &= \mathbb{E}_{\boldsymbol{\theta}\sim\pi(\boldsymbol{\theta})}\left[\mathbb{P}(\text{move}|\boldsymbol{\theta})\right] \\ &= \int \mathbb{P}(\text{move}|\boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}.\end{aligned}$$

This is almost always intractable (except in some simple cases – see tutorial questions); however, we can approximate it via the empirical acceptance rate, the fraction of proposals that have been accepted. As we shall see, the acceptance rate of an algorithm often provides an easy to measure handle on the efficiency of the algorithm.

Example 2.4.1. (Running example: Gaussian target and RWM method with Gaussian innovations.) Recall the target and proposal of the form

$$\pi(\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\theta^2/(2\sigma^2)},$$
$$q(\theta^*|\theta) = \frac{1}{\lambda\sqrt{2\pi}} e^{-(\theta^*-\theta)^2/(2\lambda^2)}.$$

with $\sigma = 1$ and $\lambda = 2$. R code for implementing this example is given below.

```
## RWM (normal target and normal innovations)

rwm=function(N,lambda,theta0)
{
 theta=rep(0,N)  # Store theta values here
 theta[1]=theta0 # Initialise
 count=0 # count acceptances
 for(i in 2:N)
 {
  can=theta[i-1]+rnorm(1,0,lambda)  # Propose
  laprob=dnorm(can,log=TRUE)-dnorm(theta[i-1],log=TRUE)
  if(log(runif(1))<laprob)
  {
   theta[i]=can # Store candidate value
   count=count+1
  }else{
   theta[i]=theta[i-1] # Else store current value
  }
 }
print(count/(N-1))
return(theta)
}

set.seed(3421)
out=rwm(100,2,0)

## [1] 0.4747475
```

Note that the estimated overall acceptance rate is around 47%. We will consider whether or not this is reasonable later in the notes.

2.4.3 Burn-in and mixing

We want each draw $\boldsymbol{\theta}^{(n)}$ to have distribution $\pi(\boldsymbol{\theta})$. However, any simulation cannot be started with $\boldsymbol{\theta}^{(0)} \sim \pi$ – because we cannot simulate from $\pi(\boldsymbol{\theta})$ directly (otheriwse, why use MCMC at all?). Thus we have no *guarantee* that $\boldsymbol{\theta}^{(n)}$ follows the target distribution for any n. However, there are good reasons to believe that, independent of the starting distribution $\pi^{(0)}$, the chain will converge towards the target distribution i.e. for all sufficiently large n we will have $\boldsymbol{\theta}^{(n)} \sim \pi$, *approximately*.

The usual strategy adopted to deal with this issue is to run the simulated chain for a certain number of iterations, say K, disgard these samples, and then regard subsequent iterations as (dependent) samples that are approximately distributed according to $\pi(\boldsymbol{\theta})$. The discarded number of iterations is called the *burn-in* period.

Example 2.4.2. (Running example: Gaussian target and RWM method with Gaussian innovations.) Suppose we initialise the sampler at $\theta^{(0)} = 20$ and run for 500 iterations. The R code below produces trace plots and histograms of the sampler output with and without a burn-in period of 30 values.

```
set.seed(3421)
out=rwm(500,2,20)

## [1] 0.511022

par(mfrow=c(2,2))
plot(ts(out),main="All samples",xlab="Iteration",ylab="theta")
plot(ts(out[31:500]),main="Remove burn-in",xlab="Iteration",ylab="theta")
hist(out,freq=F,main="All samples",xlab="theta",ylim=c(0,0.45))
lines(seq(-20,20,0.001),dnorm(seq(-20,20,0.001)),type="l")
hist(out[31:500],freq=F,main="Remove burn-in",xlab="theta",ylim=c(0,0.45))
lines(seq(-20,20,0.001),dnorm(seq(-20,20,0.001)),type="l")
```

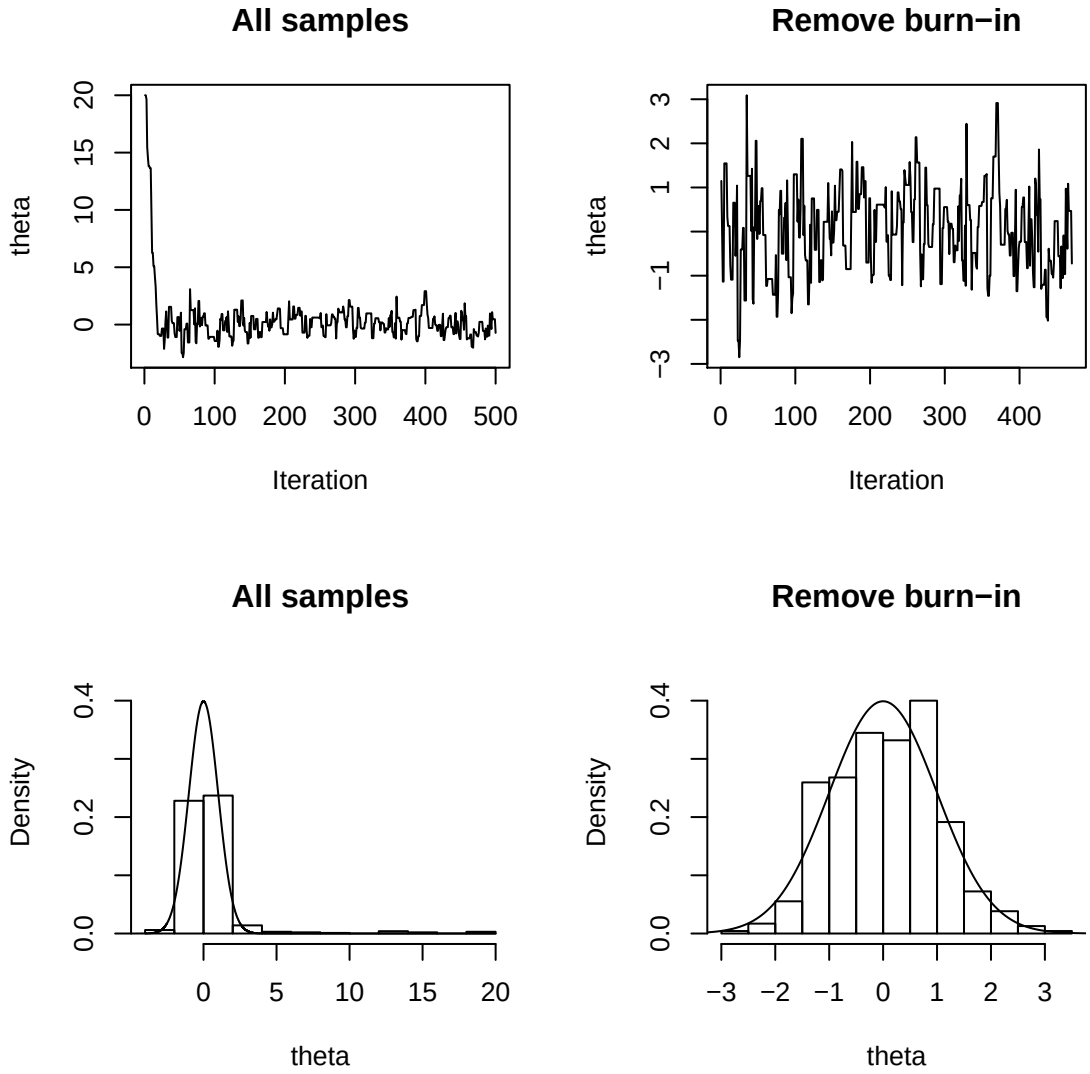

Figure 2.3: Burn-in demonstration for RWM sampler. Top panel: Trace plots. Bottom panel: Histograms with target overlaid.

2.4.4 Measuring / assessing efficiency

For this section, for simplicity of presentation, we assume that the burn-in samples have been removed, and that we have relabelled the points so that we have a sequence $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(N)}$ from the chain.

Trace plots

The following trace plots were obtained from the RWM running example, started from $\theta^{(0)} = 0$ and run for 500 iterations with $\lambda \in \{0.3, 3, 10\}$.

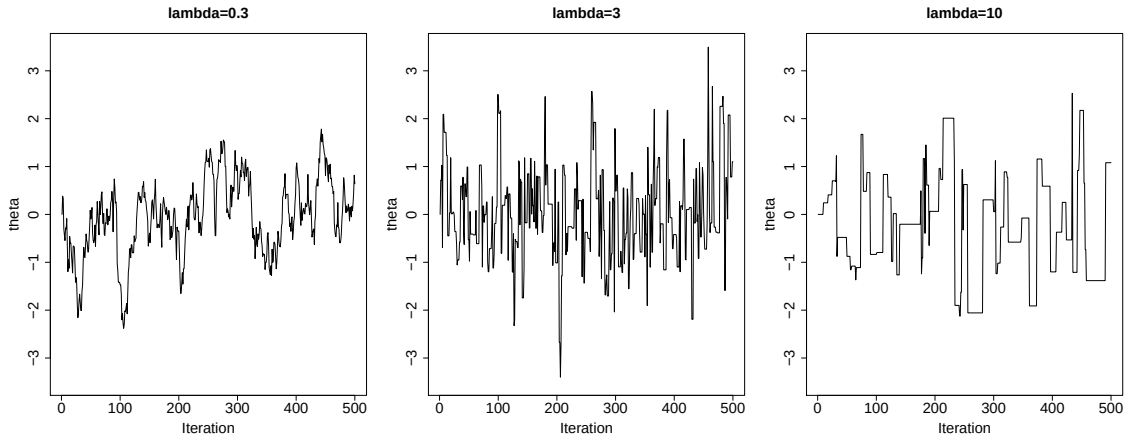

Figure 2.4: Mixing demonstration for RWM sampler. Trace plots based on 500 iterations with the innovation standard deviation $\lambda = 0.3$ (left), $\lambda = 3$ (middle) and $\lambda = 10$ (right).

For $\lambda = 0.3$, the chain is moving in such small jumps (that are mostly accepted) and has not had time to explore the range of "reasonable" values of θ. For $\lambda = 10$, the chain proposes values that do not fit with π and these are rejected. Both chains are said to be *mixing slowly*.

Remarks:

- The above example suggests a value of the innovation standard deviation λ (also known as the *scaling*) that maximises mixing efficiency.
- Consider a general d-dimensional target $\pi(\boldsymbol{\theta})$ and a Gaussian random walk proposal at iteration j of
$$\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(j-1)} + \boldsymbol{w}^{(j)}, \quad \boldsymbol{w}^{(j)} \sim N(\mathbf{0}, V)$$
with a general tuning matrix V.
  - Under certain constraints on the posterior distribution, researchers have shown that the optimal choice of V (for large d) is
$$V = \frac{2.38^2}{d} \operatorname{Var}(\boldsymbol{\theta})$$
and this leads to an optimal acceptance rate of 0.234.

– Of course, we typically don't know the posterior variance $\mathrm{Var}(\boldsymbol{\theta})$. However, we could first run the MCMC algorithm (for example by using a diagonal V) to obtain an estimate of $\mathrm{Var}(\boldsymbol{\theta})$.

– We should also note that in practice, and especially for small d, the above formula for V should just be used as a guide – an acceptance rate anywhere between 0.1 and 0.4 could be close to optimal.

Autocorrelation and ACF plots

The level of dependence in the sample can be measured in terms of the correlation at lag-k:

$$\rho_k = \mathrm{Cor}[h(\boldsymbol{\theta}^{(0)}), h(\boldsymbol{\theta}^{(k)})],$$

also known as the lag-k autocorrelation. Since the chain is (approximately) stationary, this can be obtained empirically, by looking at the correlation between pairs in the sample

$$[h(\boldsymbol{\theta}^{(0)}), h(\boldsymbol{\theta}^{(k)})], [h(\boldsymbol{\theta}^{(1)}), h(\boldsymbol{\theta}^{(k+1)})], \ldots, [h(\boldsymbol{\theta}^{(N-k)}), h(\boldsymbol{\theta}^{(N)})].$$

Typically, in practice we examine the individual components of the vector $\boldsymbol{\theta}$.

The plots below show the estimated lag-k autocorrelation of $h(\theta) = \theta$ as a function of $k = 0, 1, \ldots, 25$ for the RWM example. The plots are often called ACF plots (ACF=autocorrelation function).

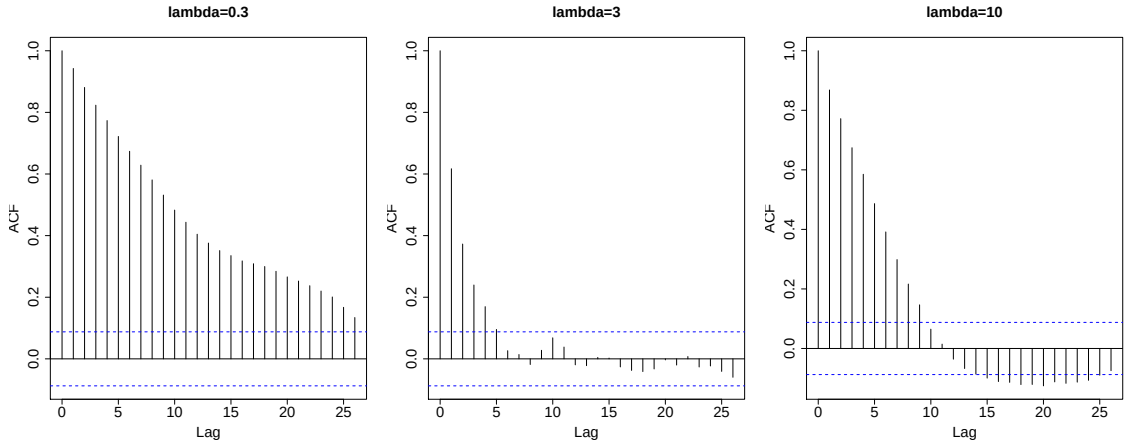

Figure 2.5: RWM sampler. ACF plots based on 500 iterations with the innovation standard deviation $\lambda = 0.3$ (left), $\lambda = 3$ (middle) and $\lambda = 10$ (right).

The dashed, blue lines show the critical value for a hypothesis test of $H_0 : \rho_k = 0$. Samples from the RWM algorithm with $\lambda = 3$ are close to independent by lag 5, whereas even when separated by a lag of 25 the samples from the algorithm with $\lambda = 0.3$ are highly positively correlated.

Effective sample size (ESS)

Recall that for a sequence of iid samples from π, $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}$, if we set

$$\hat{\mu}_h = \frac{1}{N} \sum_{i=1}^{N} h(\boldsymbol{\theta}^{(i)})$$

then

$$\mathbb{E}(\hat{\mu}_h) = \mathbb{E}_\pi[h(\boldsymbol{\theta})] = \mu_h,$$
$$\mathrm{Var}(\hat{\mu}_h) = \frac{1}{N}\mathrm{Var}_\pi[h(\boldsymbol{\theta})] = \frac{\sigma_h^2}{N}.$$

Now, if $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}$ are generated from an MH algorithm, then, since the Markov chain is not quite stationary, $\hat{\mu}_h$ is biased. However, the bias is $O(1/N)$ whereas the standard deviation of $\hat{\mu}_h$ is $\propto 1/\sqrt{N}$, and the impact of this can be substantial.

Consider $\mathrm{Var}(\hat{\mu}_h)$. It can be shown that (e.g. Brockwell and Davis, 1991, dominated convergence Theorem) that

$$\mathrm{Var}(\hat{\mu}_h) \approx C\frac{\sigma_h^2}{N} = \frac{\sigma_h^2}{N/C}.$$

The quantity N/C is known as the effective sample size (ESS) and can be computed from the autocorrelation function, leading to the following definition.

Definition: *The effective sample size (ESS) can be computed as*

$$ESS = \frac{N}{1 + 2\sum_{k=1}^{\infty} \rho_k}.$$

It is the number of iid samples that would lead to the same variance of $\hat{\mu}_h$ as that obtained from the chain.

Remarks:

- Typically we examine the ESS for each component of $\boldsymbol{\theta}$ to ascertain whether sufficient mixing has occurred.
- For the RWM example above, we obtain ESS values of 16.8 for $\lambda = 0.3$, 118.2 for $\lambda = 3$ and 30.3 for $\lambda = 10$.
- A good ESS is one that allows you to calculate what you need to the required accuracy. Often an ESS of 500 is deemed sufficent; if an ESS is below 100, inference from the MCMC sample should be treated with caution.
- Roughly speaking, we have
$$\mathrm{Var}(\hat{\mu}_h) \approx \frac{\sigma_h^2}{\mathrm{ESS}}.$$
Applying the central limit Theorem, we have approximately
$$\hat{\mu}_h \sim N\left(\mu_h\,,\, \frac{\sigma_h^2}{\mathrm{ESS}}\right).$$

By estimating σ_h^2 via the empirical variance of $h(\boldsymbol{\theta}^{(1)}), \ldots, h(\boldsymbol{\theta}^{(N)})$, we obtain $\hat{\sigma}_h^2$. Now, since most of the mass of a gaussian distribution lies within two standard deviations of the expectation, an approximate estimate of the error is $2\hat{\sigma}_h/\sqrt{\mathrm{ESS}}$.

Contents

Chapter 3

The Gibbs sampler

Constructing a suitable proposal density $q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$ may be difficult, especially when the target $\pi(\boldsymbol{\theta})$ is high dimensional. Sometimes, it is more natural to update subsets of components of $\boldsymbol{\theta}$ in a single Metropolis-Hastings step, with steps for these different subsets combined to give a single algorithm. Moreover, for many problems of interest, the corresponding *full conditional distributions (FCDs)* associated with these subsets of components may be available for sampling from. This leads to an MCMC technique known as *the Gibbs sampler*.

3.1 Component-wise and block-wise transitions

Suppose that $\boldsymbol{\theta}$ can be decomposed into $(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)'$ and that, for the moment, $\boldsymbol{\theta}_B$ is fixed and known. We therefore wish to generate samples from a target density $\pi(\boldsymbol{\theta}_A|\boldsymbol{\theta}_B)$ using Metropolis-Hastings with a proposal

$$q(\boldsymbol{\theta}_A^*|\boldsymbol{\theta}_A) := q(\boldsymbol{\theta}_A^*|\boldsymbol{\theta}_A, \boldsymbol{\theta}_B).$$

The MH acceptance probability is $\min\{1, R\}$ where

$$\begin{aligned} R(\boldsymbol{\theta}_A^*|\boldsymbol{\theta}_A, \boldsymbol{\theta}_B) &= \frac{\pi(\boldsymbol{\theta}_A^*|\boldsymbol{\theta}_B)\, q(\boldsymbol{\theta}_A|\boldsymbol{\theta}_A^*)}{\pi(\boldsymbol{\theta}_A|\boldsymbol{\theta}_B)\, q(\boldsymbol{\theta}_A^*|\boldsymbol{\theta}_A)} \\ &= \frac{\pi(\boldsymbol{\theta}_A^*, \boldsymbol{\theta}_B)/\pi(\boldsymbol{\theta}_B)}{\pi(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)/\pi(\boldsymbol{\theta}_B)} \times \frac{q(\boldsymbol{\theta}_A|\boldsymbol{\theta}_A^*)}{q(\boldsymbol{\theta}_A^*|\boldsymbol{\theta}_A)} \\ &= \frac{\pi(\boldsymbol{\theta}_A^*, \boldsymbol{\theta}_B)}{\pi(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)} \times \frac{q(\boldsymbol{\theta}_A|\boldsymbol{\theta}_A^*)}{q(\boldsymbol{\theta}_A^*|\boldsymbol{\theta}_A)}. \end{aligned}$$

Hence, we may use the full target in the acceptance probability; the only difference is that some of the parameters, $\boldsymbol{\theta}_B$, have not changed. If the update is accepted then the current state of the chain becomes $(\boldsymbol{\theta}_A^*, \boldsymbol{\theta}_B)'$ otherwise it is $(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)'$.

Now suppose that $\boldsymbol{\theta}$ is in its stationary distribution. Then, its density is $\pi(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B) = \pi(\boldsymbol{\theta}_B)\pi(\boldsymbol{\theta}_A|\boldsymbol{\theta}_B)$. If we apply the above MH step to $\boldsymbol{\theta}_A$ then:

- By design of the MH step, the conditional distribution $\pi(\boldsymbol{\theta}_A|\boldsymbol{\theta}_B)$ is preserved.
- We have not altered $\boldsymbol{\theta}_B$ so its distribution $\pi(\boldsymbol{\theta}_B)$ is preserved.

Hence, $(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)' \sim \pi(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$.

Problem: If *only* $\boldsymbol{\theta}_A$ is updated, then the Markov chain would be *reducible*, so it would never converge to $\pi(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$ in the first place.

Solution: We alternate updates to $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_B$, with the latter using some proposal density $q(\boldsymbol{\theta}_B^*|\boldsymbol{\theta}_B) := q(\boldsymbol{\theta}_B^*|\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$.

This approach applies more generally, with $\boldsymbol{\theta}$ split into three or more vectors e.g. $\boldsymbol{\theta}_A$, $\boldsymbol{\theta}_B$, $\boldsymbol{\theta}_C$,... This is called a *block update*. Now recall that $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_d)'$ and denote the FCD for the ith component of $\boldsymbol{\theta}$ by

$$\pi(\theta_i|\boldsymbol{\theta}_{-i}) := \pi(\theta_i|\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_d).$$

An algorithm that cycles through MH steps for all components in turn is as follows:

1. Initialise the state of the chain to $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_d^{(0)})'$. Initialise the iteration counter to $j = 1$.
2. Obtain a new value $\boldsymbol{\theta}^{(j)}$ from $\boldsymbol{\theta}^{(j-1)}$ by successive generation of values
   - $\theta_1^{(j)} \sim \pi(\theta_1|\theta_2^{(j-1)}, \ldots, \theta_d^{(j-1)})$ using a Metropolis–Hastings step with proposal distribution $q_1(\phi_1|\theta_1^{(j-1)})$
   - $\theta_2^{(j)} \sim \pi(\theta_2|\theta_1^{(j)}, \theta_3^{(j-1)}, \ldots, \theta_d^{(j-1)})$ using a Metropolis–Hastings step with proposal distribution $q_2(\phi_2|\theta_2^{(j-1)})$

     $\vdots$
   - $\theta_d^{(j)} \sim \pi(\theta_d|\theta_1^{(j)}, \ldots, \theta_{d-1}^{(j)})$ using a Metropolis–Hastings step with proposal distribution $q_d(\theta_d|\theta_d^{(j-1)})$
3. If $j = N$, stop, otherwise put j to $j + 1$, and return to step 2.

Remarks:

- This is in fact (close to) the original form of the Metropolis algorithm.
- If the FCD $\pi(\theta_i|\boldsymbol{\theta}_{-i})$ is available for sampling from directly for a particular component θ_i, it is easy to see that the resulting acceptance probability is 1. When all FCDs are available for sampling from, we obtain an algorithm known as the Gibbs sampler.

3.2 The Gibbs sampler

We will consider first the algorithm and then examine its properties. The algorithm has the following form.

1. Initialise the state of the chain to $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_d^{(0)})'$. Initialise the iteration counter to $j = 1$.
2. Obtain a new value $\boldsymbol{\theta}^{(j)}$ from $\boldsymbol{\theta}^{(j-1)}$ by successive generation of values
   - $\theta_1^{(j)} \sim \pi(\theta_1|\theta_2^{(j-1)}, \ldots, \theta_d^{(j-1)})$

- $\theta_2^{(j)} \sim \pi(\theta_2|\theta_1^{(j)}, \theta_3^{(j-1)}, \ldots, \theta_d^{(j-1)})$

  $\vdots$

- $\theta_d^{(j)} \sim \pi(\theta_d|\theta_1^{(j)}, \ldots, \theta_{d-1}^{(j)})$

3. If $j = N$, stop, otherwise put j to $j + 1$, and return to step 2.

Remarks:

- The algorithm above updates the components of $\boldsymbol{\theta}$ in order. This is known as the *deterministic scan* Gibbs sampler.
- The algorithm clearly defines a homogeneous Markov chain, as each simulated value depends only on the previous simulated value and not on any previous value or the iteration counter j.

3.2.1 Validity

A note on reversible chains

If $\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}$ is a Markov chain then it is straightforward to show that the reversed sequence of states $\boldsymbol{\theta}^{(N)}, \boldsymbol{\theta}^{(N-1)}, \ldots, \boldsymbol{\theta}^{(0)}$ is also a Markov chain. Let $p(\boldsymbol{\phi}|\boldsymbol{\theta})$ denote the transition density for the forward chain. The reversed chain is not homogeneous in general, however, if the chain is in equilibrium then the transition density for the reversed chain is, via Bayes Theorem

$$p^*(\boldsymbol{\phi}|\boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta}|\boldsymbol{\phi})\pi(\boldsymbol{\phi})}{\pi(\boldsymbol{\theta})}.$$

Definition: *The chain is reversible if* $p^*(\boldsymbol{\phi}|\boldsymbol{\theta}) = p(\boldsymbol{\phi}|\boldsymbol{\theta})$

A consequence of reversibility is the *detailed balance equation*:

$$\pi(\boldsymbol{\theta})p(\boldsymbol{\phi}|\boldsymbol{\theta}) = \pi(\boldsymbol{\phi})p(\boldsymbol{\theta}|\boldsymbol{\phi}).$$

Alternatively, if a chain satisfies detailed balance then it is also reversible. Unfortunately, the deterministic scan Gibbs sampler is **not** reversible, so we can't use detailed balance to check that $\pi(\boldsymbol{\theta})$ is stationary.

Direct checking

The Gibbs sampler admits a Markov chain with transition density

$$p(\boldsymbol{\phi}|\boldsymbol{\theta}) = \prod_{i=1}^{d} \pi(\phi_i|\phi_1, \ldots, \phi_{i-1}, \theta_{i+1}, \ldots, \theta_d).$$

To check stationarity directly, we show that

$$\pi(\boldsymbol{\phi}) = \int p(\boldsymbol{\phi}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}.$$

Starting with the RHS, we have

$$\begin{aligned}
\int p(\boldsymbol{\phi}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} &= \int \cdots \int \pi(\boldsymbol{\theta})d\theta_1\, \pi(\phi_1|\theta_2,\ldots,\theta_d)\ldots\pi(\phi_d|\phi_1,\ldots,\phi_{d-1})d\theta_2\ldots d\theta_d \\
&= \int \cdots \int \pi(\phi_1,\theta_2,\ldots,\theta_d)d\theta_2\, \pi(\phi_2|\phi_1,\theta_3,\ldots,\theta_d)\ldots\pi(\phi_d|\phi_1,\ldots,\phi_{d-1})d\theta_3\ldots d\theta_d \\
&= \ldots \\
&= \int \pi(\phi_1,\ldots,\phi_{d-1},\theta_d)d\theta_d\, \pi(\phi_d|\phi_1,\ldots,\phi_{d-1}) \\
&= \pi(\phi_1,\ldots,\phi_d)
\end{aligned}$$

as required.

3.2.2 Reversible Gibbs samplers

While the standard *deterministic scan* version of the Gibbs sampler presented above is not reversible, each *component update* is in fact reversible. Hence there are several variations of the Gibbs sampler that *are* reversible, and hence do satisfy the detailed balance equation. Let's start by considering each component update separately.

Suppose we wish to update component i, that is, we update $\boldsymbol{\theta}$ by replacing θ_i with ϕ_i drawn from $\pi(\phi_i|\boldsymbol{\theta}_{-i})$, where $\boldsymbol{\theta}_{-i} = (\theta_1,\ldots,\theta_{i-1},\theta_{i+1},\ldots,\theta_d)'$. All the other components are not changed. The transition kernel for this update is

$$p(\boldsymbol{\phi}|\boldsymbol{\theta}) = \pi(\phi_i|\boldsymbol{\theta}_{-i})I(\boldsymbol{\theta}_{-i} = \boldsymbol{\phi}_{-i})$$

where

$$I(E) = \begin{cases} 1 & \text{if } E \text{ is true} \\ 0 & \text{if } E \text{ is false} \end{cases}.$$

Note that the density is zero for a transition changing any but the i-th component. Now we can check detailed balance:

$$\begin{aligned}
\pi(\boldsymbol{\theta})p(\boldsymbol{\phi}|\boldsymbol{\theta}) &= \pi(\boldsymbol{\theta})\pi(\phi_i|\boldsymbol{\theta}_{-i})I(\boldsymbol{\theta}_{-i} = \boldsymbol{\phi}_{-i}) \\
&= \pi(\boldsymbol{\theta}_{-i})\pi(\theta_i|\boldsymbol{\theta}_{-i})\pi(\phi_i|\boldsymbol{\theta}_{-i})I(\boldsymbol{\theta}_{-i} = \boldsymbol{\phi}_{-i}) \\
&= \pi(\boldsymbol{\phi}_{-i})\pi(\theta_i|\boldsymbol{\phi}_{-i})\pi(\phi_i|\boldsymbol{\phi}_{-i})I(\boldsymbol{\theta}_{-i} = \boldsymbol{\phi}_{-i}) \quad \text{since } \boldsymbol{\phi}_{-i} = \boldsymbol{\theta}_{-i} \\
&= \pi(\boldsymbol{\phi})\pi(\theta_i|\boldsymbol{\phi}_{-i})I(\boldsymbol{\theta}_{-i} = \boldsymbol{\phi}_{-i}) \\
&= \pi(\boldsymbol{\phi})p(\boldsymbol{\theta}|\boldsymbol{\phi}).
\end{aligned}$$

Detailed balance therefore holds, and hence the update is reversible with stationary distribution $\pi(\boldsymbol{\theta})$.

Remarks:

- Recall that the chain defined by a single update is *reducible* and hence will not converge to the stationary distribution from an arbitrary starting point.
- In order to ensure *irreducibility* we need to make sure that we update each component of the chain sufficiently often. As we have seen, one way to do this is to update each component in a fixed order. The drawback with this is that we lose reversibility when we do this.
- An alternative is to pick a component at random at each stage, and update that. This gives a reversible chain with the required stationary distribution, and is known as the *random scan* Gibbs sampler.

3.2.3 Gibbs sampling in practice

Example 3.2.1. (Bivariate Gaussian target.) Consider a bivariate normal target with zero mean and unit variance for the marginals, but a correlation of ρ between the two components. We have that

$$\begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix} \sim N\left\{\begin{pmatrix}0\\ 0\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right\}$$

with associated density

$$\pi(\theta_1,\theta_2) \propto \exp\left\{-\frac{1}{2(1-\rho^2)}\left(\theta_1^2+\theta_2^2-2\rho\theta_1\theta_2\right)\right\}, \quad -\infty<\theta_1,\theta_2<\infty.$$

Let us construct a Gibbs sampler for this target.

The full conditional density for θ_1 is

$$\begin{aligned}\pi(\theta_1|\theta_2) &= \frac{\pi(\theta_1,\theta_2)}{\pi(\theta_2)}\\ &\propto \pi(\theta_1,\theta_2)\\ &\propto \exp\left\{-\frac{1}{2(1-\rho^2)}\left(\theta_1^2-2\rho\theta_1\theta_2\right)\right\}\\ &\propto \exp\left\{-\frac{1}{2(1-\rho^2)}\left(\theta_1-\rho\theta_2\right)^2\right\}\end{aligned}$$

for which we find the full conditional distribution for θ_1 as

$$\theta_1|\theta_2 \sim N\left(\rho\theta_2, 1-\rho^2\right).$$

By symmetry, the full conditional distribution for θ_2 is

$$\theta_2|\theta_1 \sim N\left(\rho\theta_1, 1-\rho^2\right).$$

The Gibbs sampler for the target above then has the following form.

1. Initialise the state of the chain to $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \theta_2^{(0)})'$. Initialise the iteration counter to $j=1$.
2. Obtain a new value $\boldsymbol{\theta}^{(j)}$ from $\boldsymbol{\theta}^{(j-1)}$ by successive generation of values

- $\theta_1^{(j)} \sim N\left(\rho\theta_2^{(j-1)}, 1-\rho^2\right)$
- $\theta_2^{(j)} \sim N\left(\rho\theta_1^{(j)}, 1-\rho^2\right)$

3. If $j = N$, stop, otherwise put j to $j+1$, and return to step 2.

The following R code implements this sampler.

```
## Example: Gibbs sampler for a bivariate normal

gibbs=function(N,rho)
{
 mat=matrix(ncol=2,nrow=N)
 th1=0
 th2=0
 mat[1, ]=c(th1,th2)
 for (i in 2:N)
 {
  th1=rnorm(1,rho*th2,sqrt(1-rho^2))
  th2=rnorm(1,rho*th1,sqrt(1-rho^2))
  mat[i, ]=c(th1,th2)
 }
 return(mat)
}

## uncomment to run and plot output
#set.seed(3421)
#out=gibbs(1000,0.5)
#par(mfrow=c(3,2))
#plot(out,col=1:1000,xlab="theta1",ylab="theta2")
#plot(out,type="l",xlab="theta1",ylab="theta2")
#plot(ts(out[,1]),xlab="Iteration",ylab="theta1")
#plot(ts(out[,2]),xlab="Iteration",ylab="theta2")
#hist(out[,1],40,freq=FALSE,main="",xlab="theta1")
#hist(out[,2],40,freq=FALSE,main="",xlab="theta2")
```

The following plots summarise the sampler output using $N = 1000$ and $\rho \in \{0.5, 0.99\}$. Note the poor mixing when correlation between the two components is high.

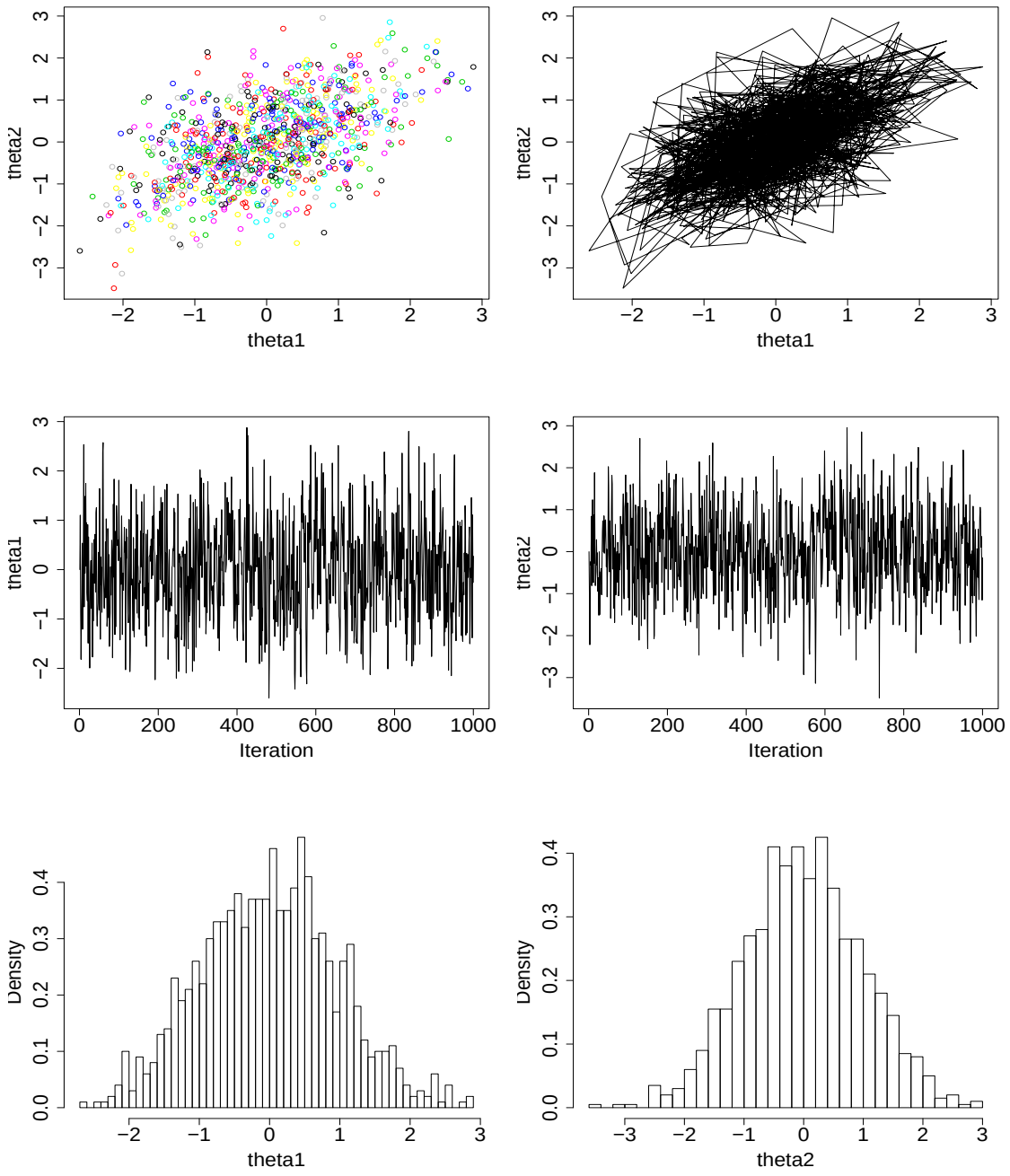

Figure 3.1: Bivariate normal target. Gibbs sampler output when $\rho = 0.5$.

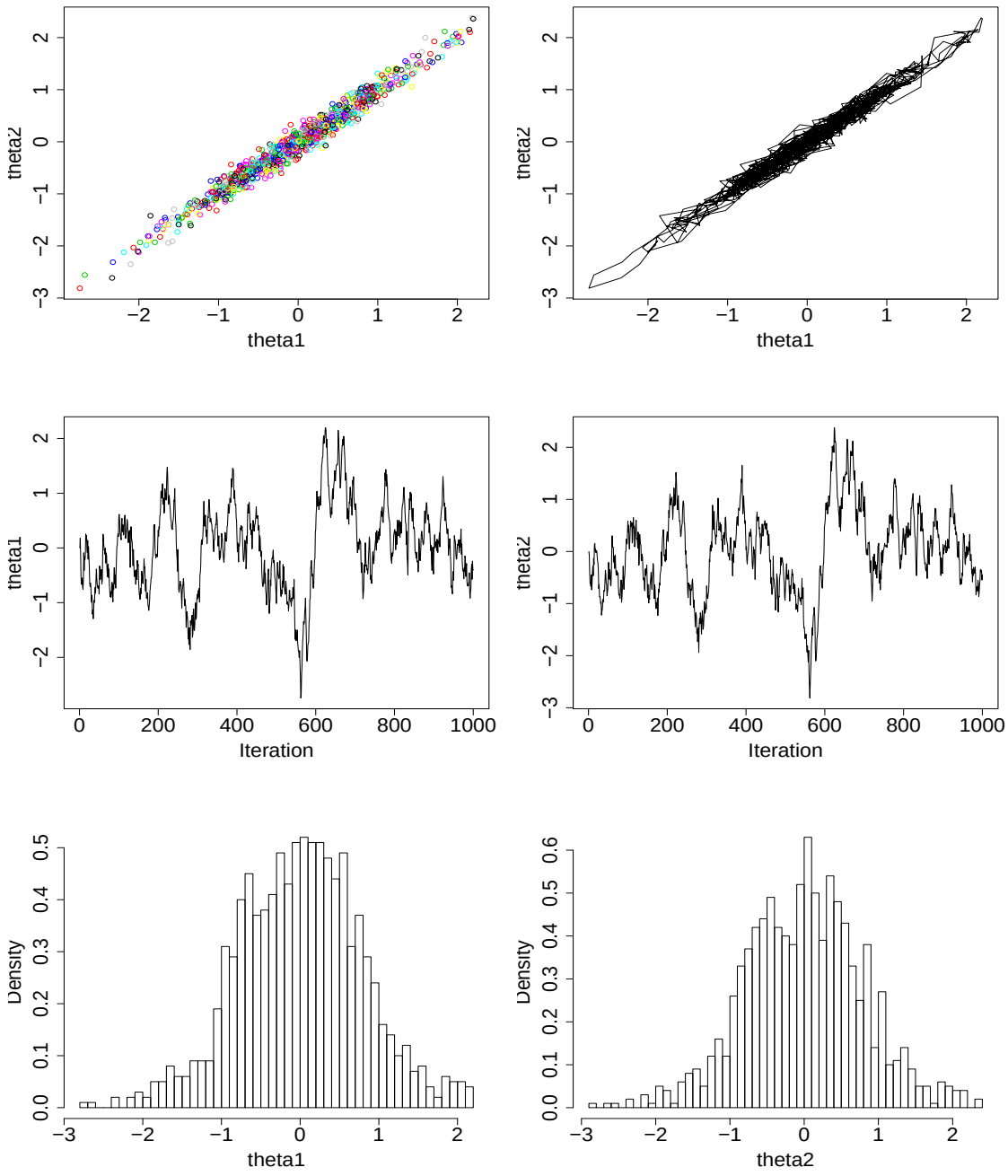

Figure 3.2: Bivariate normal target. Gibbs sampler output when $\rho = 0.99$.

Contents

Chapter 4

Bayesian Inference via MCMC

In this chapter, we will consider the use of Metropolis-Hastings and the Gibbs sampler in two applications of increasing complexity.

4.1 Application I: Normal model with unknown mean and precision

Reconsider the problem from Chapter 1 of Bayesian inference for the mean and precision of a normally distributed random sample. In particular, consider the non-conjugate approach based on the independent prior distributions for the mean and precision. We have that

$$\begin{aligned} X_i|\mu,\tau &\sim N(\mu, 1/\tau) \text{ independently, for } i = 1,\ldots,n \\ \tau &\sim \text{Gamma}(g,h) \\ \mu &\sim N(b, 1/c). \end{aligned}$$

The joint posterior for μ and τ based on a sample $\boldsymbol{x}$ of size n can be written in terms of the sufficient statistics

$$\bar{x} = \frac{1}{n}\sum_1^n x_i \quad \text{and} \quad s^2 = \frac{1}{n}\sum_1^n (x_i - \bar{x})^2.$$

The posterior takes the form

$$\pi(\mu,\tau|\boldsymbol{x}) \propto \tau^{\frac{n}{2}+g-1} \exp\left\{-\frac{n\tau}{2}\left[s^2 + (\bar{x}-\mu)^2\right] - h\tau - \frac{c}{2}(\mu-b)^2\right\}.$$

As explained previously, this distribution is non-standard, and the analysis is not conjugate.

Setup

- Suppose that we have observed data with $n = 100$, $\bar{x} = 12$ and $s^2 = 1$. Let us take $\mu \sim N(10, 100)$ and $\tau \sim Gamma(1, 0.1)$ *a priori*.
- We will consider:
  - A Gibbs sampler,
  - random walk Metropolis,
  - a Metropolis-Hastings independence sampler.

4.1.1 Gibbs sampler

A *semi-conjugate* analysis is possible: the two conditional distributions $\pi(\mu|\tau, \boldsymbol{x})$ and $\pi(\tau|\mu, \boldsymbol{x})$ are of standard form, and are of the same form as the independent priors. In fact, $\mu|\tau, \boldsymbol{x}$ is Normally distributed and $\tau|\mu, \boldsymbol{x}$ is Gamma:

$$\tau|\mu, \boldsymbol{x} \sim \text{Gamma}\left(g + \frac{n}{2}, h + \frac{n}{2}\left[s^2 + (\bar{x} - \mu)^2\right]\right)$$
$$\mu|\tau, \boldsymbol{x} \sim N\left(\frac{bc + n\tau\bar{x}}{c + n\tau}, \frac{1}{c + n\tau}\right).$$

It therefore seems natural to apply the Gibbs sampler to this problem. The (deterministic scan) Gibbs sampler takes the following form:

1. Initialise the state of the chain $(\mu^{(0)}, \tau^{(0)})^T$ e.g. using the prior mean $(10, 10)'$. Initialise the iteration counter to $j = 1$.
2. Obtain a new values $\mu^{(j)}$ and $\tau^{(j)}$ from by successive generation of values
   - $\mu^{(j)} \sim N\left(\frac{bc+n\tau^{(j-1)}\bar{x}}{n\tau^{(j-1)}+c}, \frac{1}{n\tau^{(j-1)}+c}\right)$
   - $\tau^{(j)} \sim \text{Gamma}\left(g + \frac{n}{2}, h + \frac{n}{2}\left[s^2 + (\bar{x} - \mu^{(j)})^2\right]\right)$
3. If $j = N$ stop, otherwise change counter j to $j + 1$, and return to step 2.

Gibbs sampler: R code

```
## Application I: Gibbs sampler

gibbs=function(N,s,xbar,n,b,c,g,h)
{
 mat=matrix(0,ncol=2,nrow=N) #store sampled mu and tau here
 mu=b
 tau=g/h
 mat[1,]=c(mu,tau) #initialise at prior mean
 for (i in 2:N)
 {
  mu=rnorm(1,(b*c+n*tau*xbar)/(n*tau+c),sqrt(1/(n*tau+c)))
  tau=rgamma(1,g+0.5*n,h+0.5*n*(s^2+(xbar-mu)^2))
  mat[i,]=c(mu,tau) #store
 }
 return(mat)
}

set.seed(3421)
out=gibbs(N=5000,s=1,xbar=12,n=100,b=10,c=1/100,g=1,h=0.1)
```

```
#Plots

par(mfrow=c(3,3))
plot(ts(out[100:5000,1]),main="mu",xlab="iteration",ylab="value")
acf(out[100:5000,1],main="")
hist(out[100:5000,1],freq=F,main="mu",xlab="value")
plot(ts(out[100:5000,2]),main="tau",xlab="iteration",ylab="value")
acf(out[100:5000,2],main="")
hist(out[100:5000,2],freq=F,main="tau",xlab="value")
plot(ts(1/sqrt(out[100:5000,2])),main="sigma",xlab="iteration",ylab="value")
acf(1/sqrt(out[100:5000,2]),main="")
hist(1/sqrt(out[100:5000,2]),freq=F,main="sigma",xlab="value")
```

4.1.2 Random walk Metropolis

Suppose that we wish to sample $\pi(\mu, \tau|\boldsymbol{x})$ using a Metropolis-Hastings algorithm with a random walk proposal that uses Gaussian innovations. At this point, it is helpful to define $\boldsymbol{\theta} = (\theta_1, \theta_2)' = (\mu, \tau)'$.

We will use a proposal of the form

$$\boldsymbol{\theta}^* = \boldsymbol{\theta} + \boldsymbol{w}, \qquad \boldsymbol{w} \sim N(\boldsymbol{0}, V)\,.$$

Thus, $\boldsymbol{\theta}^* \sim N(\boldsymbol{\theta}, V)$. We therefore need to be able to simulate multivariate Normal random quantities. This is possible in R with `mvrnorm` which requires the MASS package, or alternatively we can devise our own algorithm. A simple way to generate d-dimensional $N(\boldsymbol{m}, V)$ random quantities using just univariate $N(0, 1)$ realisations is as follows.

1. Generate d independent $N(0, 1)$ realisations, $z_1, z_2, \ldots, z_d$ and stack these in the $d \times 1$ vector $\boldsymbol{z}$.
2. Find a matrix A such that $V = AA^T$. For example, use the *Cholesky* decomposition so that A is lower triangular.
3. Take $\boldsymbol{m} + A\boldsymbol{z}$ as a $N(\boldsymbol{m}, V)$ realisation.

These steps can be implemented in R as follows.

```
#simulate a N(m,V) random vector
rmvn=function(m,V)
{
 d=length(m)
 z=rnorm(d)
 return(m+t(chol(V))%*%z)
}
```

The random walk Metropolis algorithm has the following form.

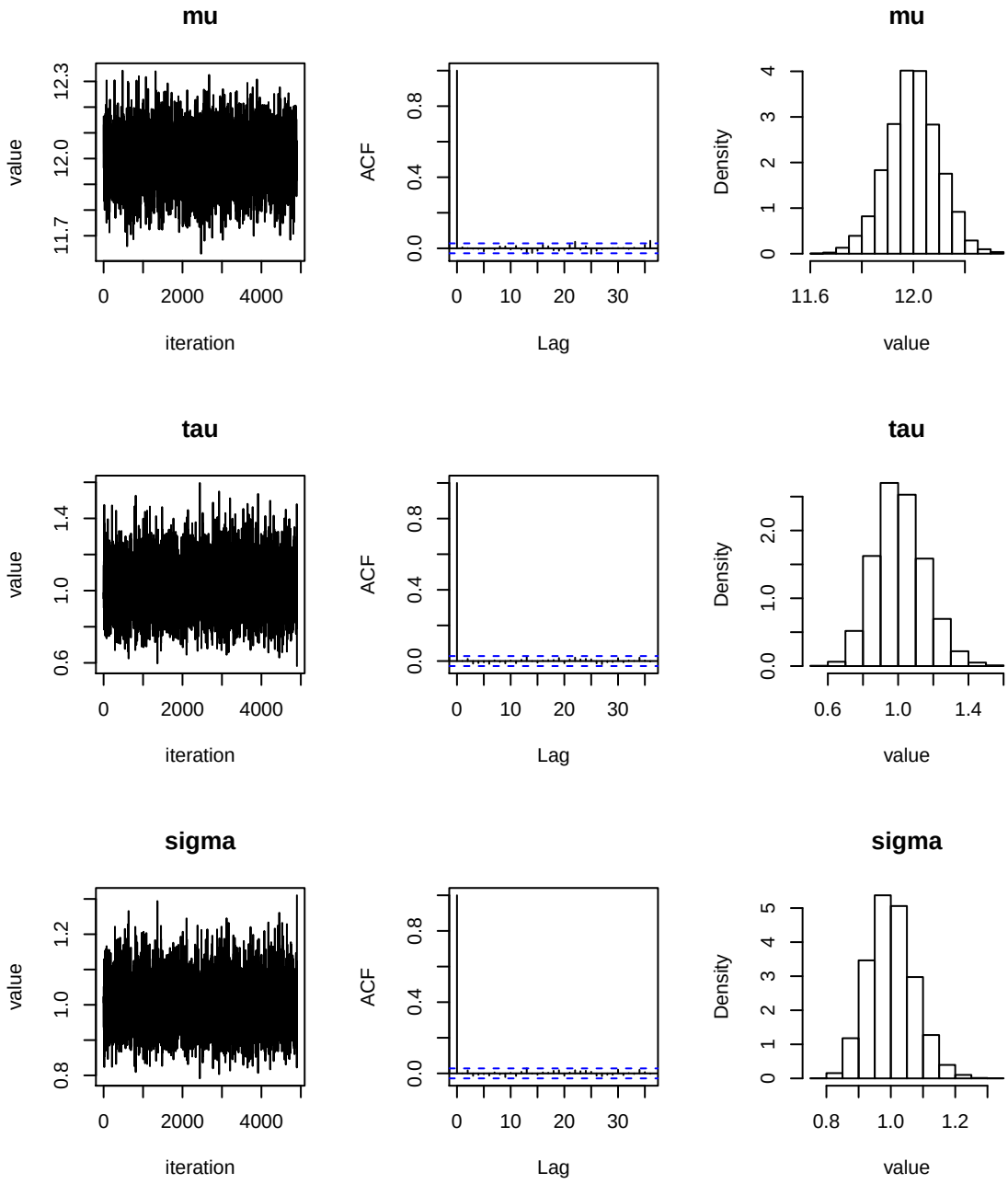

Figure 4.1: Application I. Gibbs sampler output.

1. Initialise the chain to $\boldsymbol{\theta}^{(0)} = (10, 10)'$. Initialise the iteration counter to $j = 1$.
2. Generate a *proposed* value
$$\boldsymbol{\theta}^* \sim N(\boldsymbol{\theta}^{(j-1)}, V)\,.$$
3. Evaluate the *acceptance probability* $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$ of the proposed move,
$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\, f(\boldsymbol{x}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^{(j-1)})\, f(\boldsymbol{x}|\boldsymbol{\theta}^{(j-1)})}\right\}.$$
4. Put $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^*$ with probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$; otherwise put $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)}$.
5. If $j = N$, stop, otherwise put j to $j + 1$ and go to step 2.

Recall that:

- The prior density is $\pi(\boldsymbol{\theta}) = \pi(\mu)\pi(\tau)$ due to the independent prior specification.
- The likelihood is $f(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{n} N(x_i; \mu, \tau^{-1})$.

Random walk Metropolis: R code

```
## Application I: Random walk Metropolis

# Function to evaluate the log of the posterior density
# up to an additive constant

lpost=function(theta,s,xbar,n,b,c,g,h)
{
 mu=theta[1]
 tau=theta[2]
 logprior=dnorm(mu,b,sqrt(1/c),log=T)+dgamma(tau,g,h,log=T)
 loglike=(n/2)*log(tau)-n*tau/2*(s^2+(xbar-mu)^2)
 logpost=logprior+loglike
 return(logpost)
}

# Random walk Metropolis sampler -- uses lpost and rmvn

rwm=function(N,V,s,xbar,n,b,c,g,h)
{
 mat=matrix(0,ncol=2,nrow=N) #store sampled mu and tau here
 theta=c(b,g/h)
 mat[1,]=theta    #initialise at prior mean
 for (i in 2:N)
 {
  can=rmvn(theta,V) #propose
  laprob=lpost(can,s,xbar,n,b,c,g,h)-lpost(theta,s,xbar,n,b,c,g,h)
```

```
  u=runif(1)
  if (log(u) < laprob)
  {
    theta=can
  }
  mat[i,]=theta
 }
 return(mat)
}
```

Note that there's a potential problem here! The sampler may propose a negative value of τ, which is fine in theory as this should just be rejected (as the support of τ *a posteriori* is the positive reals). However, the above function would give an error message. We can deal with this in one of two ways:

1. Amend the code to reject any proposed moves for which $\tau < 0$ (before calling `lpost`.
2. Design a better Metropolis-Hastings sampler (e.g. one that works with $\log(\tau)$ rather than τ).

To run the code, we need a suitable value of the tuning parameter V. As we already have realisations of μ and τ from the posterior (as given by the Gibbs sampler) let's cheat (!) and take

$$V = \frac{2.38^2}{2}\widehat{Var}(\boldsymbol{\theta}|\boldsymbol{x})\,.$$

This can be achieved with

```
# Obtain sensible tuning matrix from Gibbs output
postvar=var(out[100:5000,])
vartune=(2.38^2/2)*postvar
```

In fact, for this example posterior correlation appears fairly low (≈ -0.018) and so there is probably little to be gained by using a correlated (versus uncorrelated) random walk in this example. Now we're ready to implement the algorithm.

```
#Run
set.seed(3421)
out2=rwm(N=5000,V=postvar,s=1,xbar=12,n=100,b=10,c=1/100,g=1,h=0.1)

#Plots

par(mfrow=c(3,3))
plot(ts(out2[500:5000,1]),main="mu",xlab="iteration",ylab="value")
acf(out2[500:5000,1],main="")
```

```
hist(out2[500:5000,1],freq=F,main="mu",xlab="value")
plot(ts(out2[500:5000,2]),main="tau",xlab="iteration",ylab="value")
acf(out2[500:5000,2],main="")
hist(out2[500:5000,2],freq=F,main="tau",xlab="value")
plot(ts(1/sqrt(out2[500:5000,2])),main="sigma",xlab="iteration",ylab="value")
acf(1/sqrt(out2[500:5000,2]),main="")
hist(1/sqrt(out2[500:5000,2]),freq=F,main="sigma",xlab="value")
```

The usual trace and acf plots can be seen below. The mixing is adequate, although apparently not as good as for the Gibbs sampler. In fact, the *minimum effective sample size* (ESS, with the minimum taken over each parameter chain) for the Gibbs sampler is 4900 versus 87 for random walk Metropolis, suggesting that a longer run of the latter is required.

4.1.3 Independence sampler

Finally, suppose that we wish to sample $\pi(\mu, \tau|\boldsymbol{x})$ using a Metropolis-Hastings algorithm with an independence proposal given by the prior. That is, candidate values of μ and τ are drawn from $\pi(\mu, \tau) = \pi(\mu)\pi(\tau)$. The algorithm is as follows.

1. Initialise the chain to $\boldsymbol{\theta}^{(0)} = (10, 10)'$. Initialise the iteration counter to $j = 1$.
2. Generate a *proposed* value

$$\theta_1^* \sim N(10, 100) \qquad \text{and} \qquad \theta_2^* \sim Gamma(1, 0.1).$$

3. Evaluate the *acceptance probability* $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$ of the proposed move,

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)}) = \min\left\{1, \frac{f(\boldsymbol{x}|\boldsymbol{\theta}^*)}{f(\boldsymbol{x}|\boldsymbol{\theta}^{(j-1)})}\right\}.$$

4. Put $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^*$ with probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$; otherwise put $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)}$.
5. Put If $j = N$, stop, otherwise set j to $j + 1$ and go to step 2.

Note that the prior density cancels in the acceptance ratio and we only need to evaluate the (log) likelihood.

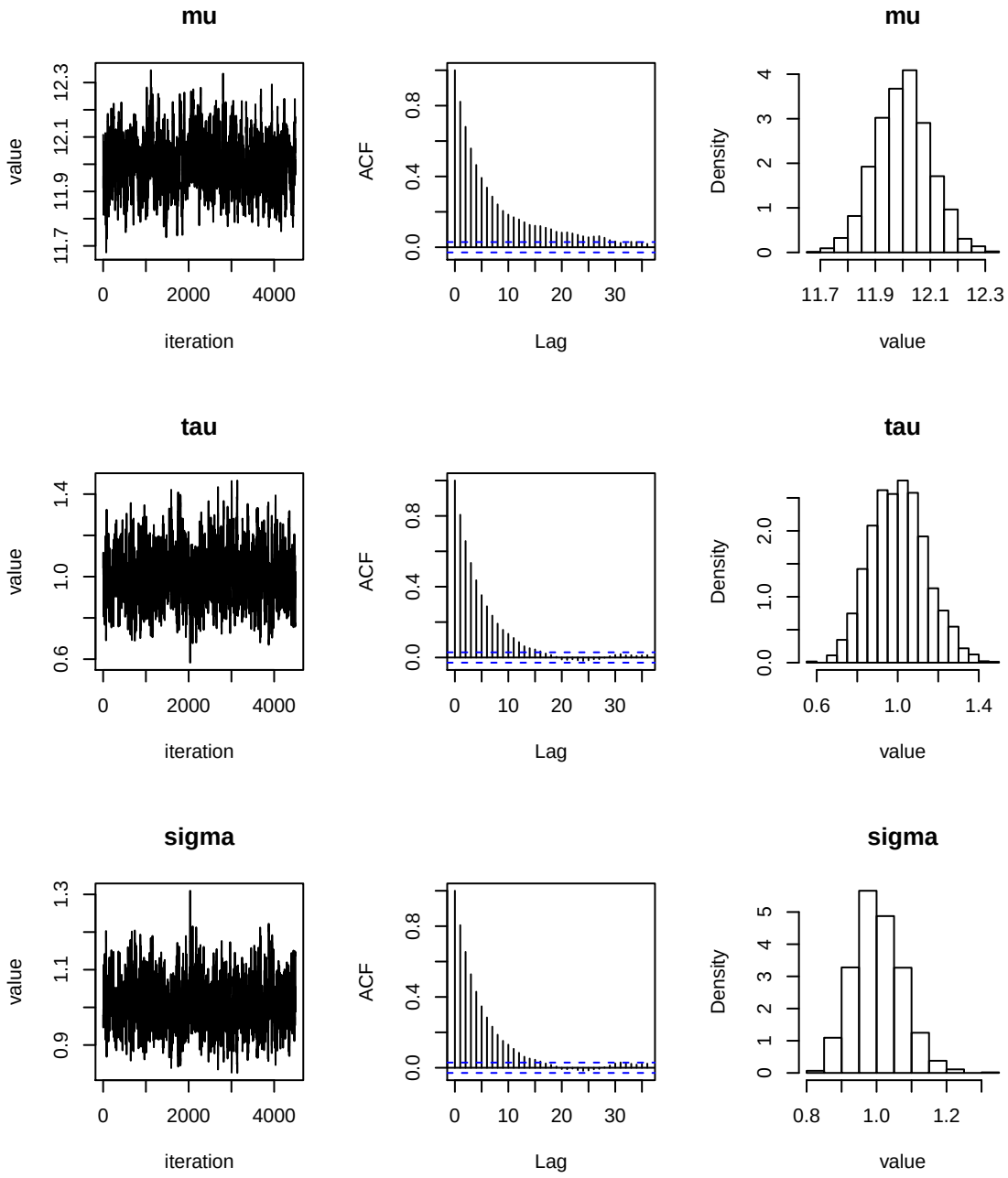

Figure 4.2: Application I. Random walk Metropolis output.

Independence sampler: R code

```
## Application I: Independence sampler

# Function to evaluate log-likelihood up to an additive constant

llike=function(theta,s,xbar,n)
{
 mu=theta[1]
 tau=theta[2]
 loglike=(n/2)*log(tau)-n*tau/2*(s^2+(xbar-mu)^2)
 return(loglike)
}

# Indepence sampler

mh=function (N,s,xbar,n,b,c,g,h)
{
 can=rep(0,2)
 mat=matrix(0,ncol=2,nrow=N) #store sampled mu and tau here
 theta=c(b,g/h)
 mat[1,]=theta #initialise at prior mean
 for (i in 2:N)
 {
  can[1] = rnorm(1,b,sqrt(1/c))
  can[2] = rgamma(1,g,h) #propose from prior
  laprob = llike(can,s,xbar,n)-llike(theta,s,xbar,n)
  u = runif(1)
  if (log(u) < laprob)
  {
    theta = can
  }
  mat[i,] = theta
 }
 return(mat)
}
```

We then call the `mh` function as follows. Note the number of iterations ($N = 500,000$) and the output is *thinned*[1] so that the subsequent plots are based on $5,000$ samples.

[1] A thin of a factor k means storing $\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(2k)}, \ldots$ It is typically used to reduce storage costs. Estimations based on the thinned chain are still valid, however, thinning throws away information.

```
#Run

set.seed(3421)
out3=mh(N=500000,s=1,xbar=12,n=100,b=10,c=1/100,g=1,h=0.1)

#Plots

par(mfrow=c(3,3))
plot(ts(out3[(1:5000)*100,1]),main="mu",xlab="iteration",ylab="value")
acf(out3[(1:5000)*100,1],main="")
hist(out3[(1:5000)*100,1],freq=F,main="mu",xlab="value")
plot(ts(out3[(1:5000)*100,2]),main="tau",xlab="iteration",ylab="value")
acf(out3[(1:5000)*100,2],main="")
hist(out3[(1:5000)*100,2],freq=F,main="tau",xlab="value")
plot(ts(1/sqrt(out3[(1:5000)*100,2])),main="sigma",xlab="iteration",ylab="value")
acf(1/sqrt(out3[(1:5000)*100,2]),main="")
hist(1/sqrt(out3[(1:5000)*100,2]),freq=F,main="sigma",xlab="value")
```

Independence sampler: output

The usual trace and acf plots can be seen below. Note that the mixing is poor, despite a very long. The minimum ESS calculated from the thinned sample is around 200. This is not the full story! The time taken to run the sampler on my laptop is around 8s. Hence the minimum ESS/s is 200/8 versus 87/0.17 for the random walk Metropolis algorithm. The problem with the independence sampler used here is that the prior is quite diffuse compared to the posterior, so most proposed parameter values are rejected, leading to a small acceptance probability.

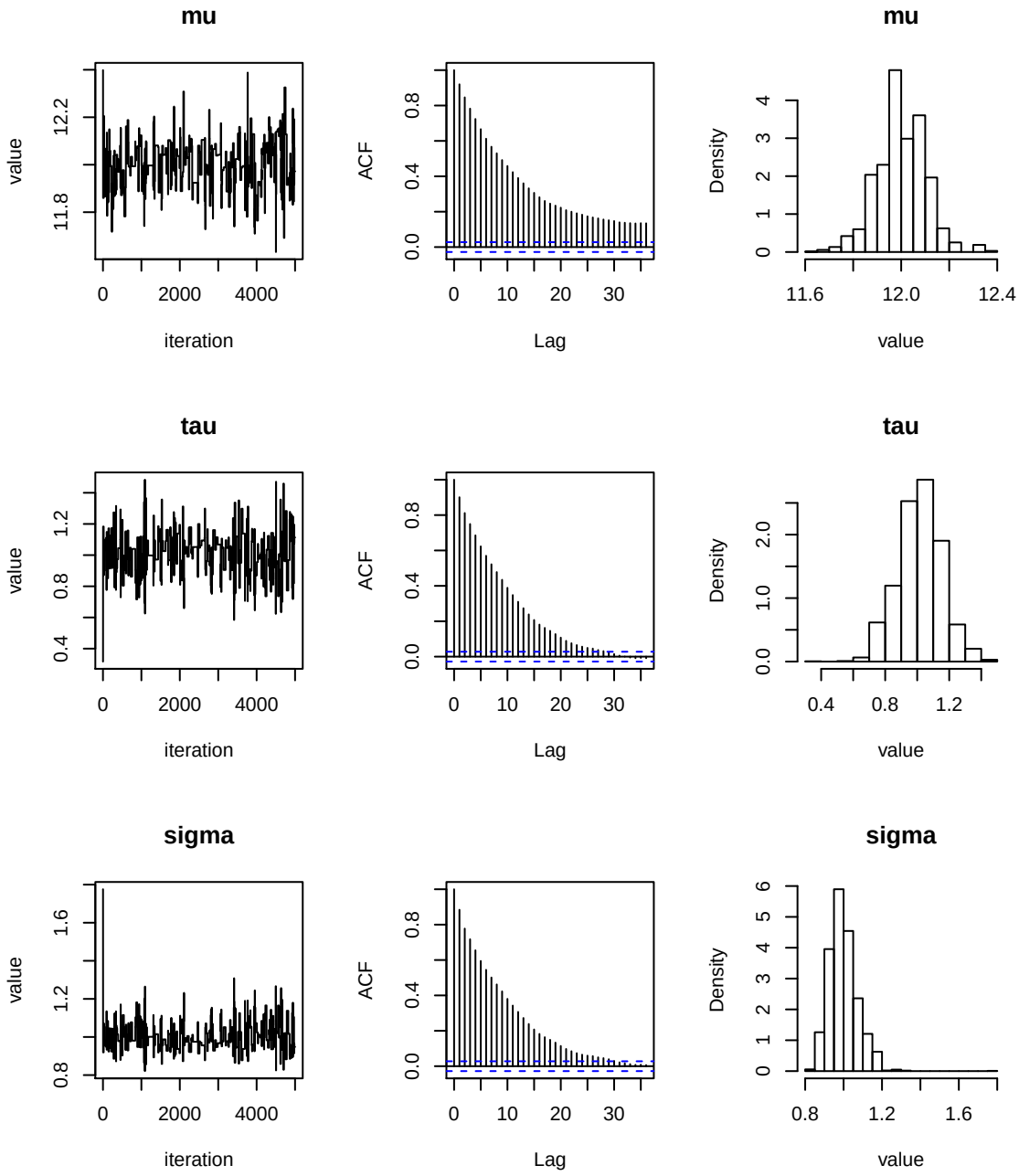

Figure 4.3: Application I. Independence sampler output.

4.2 Application II: Stochastic volatility model

In this section we consider the problem of modelling an asset's price. Let $Y_0^*, Y_1^*, \ldots, Y_n^*$ be (daily) readings of some quantity such as share price of a given company. Define $Y_t = \log Y_t^* - \log Y_{t-1}^*$ be be the logarithm of the relative change in share value since the previous day. These daily differences often exhibit changes in variability. To see this consider the *Standard and Poors 500* index, for which a snapshot of the raw index and daily differences are plotted below.

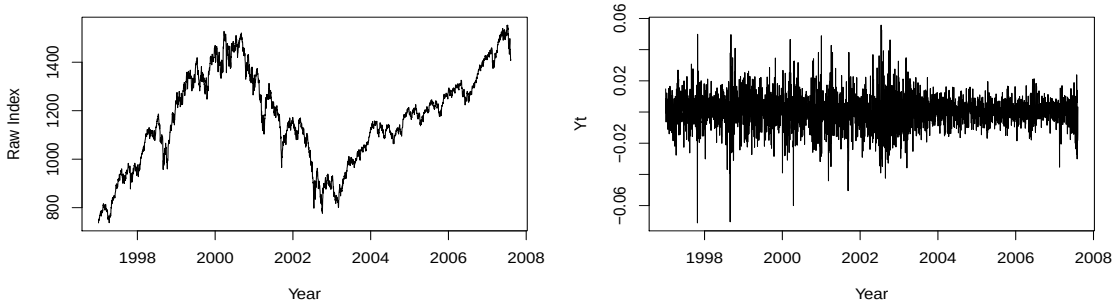

Figure 4.4: S& P 500 data. Left panel: raw index. Right panel: log differences.

Clearly, the day-to-day variability (known in the financial time-series context as *volatility*) is not constant through time. One commonly used model to capture this behaviour is the *stochastic volatility model* given by

$$\begin{aligned} X_0 &\sim N\left(0,\, \frac{\sigma^2}{1-\phi^2}\right), \\ X_t | X_{t-1} = x_{t-1} &\sim N\left(\phi x_{t-1},\, \sigma^2\right), \\ Y_t | X_t = x_t &\sim N\left(0,\, \kappa^2 \exp\{x_t\}\right), \end{aligned}$$

for $t = 1, \ldots, n$. Plainly, $\{X_t\}_{t=0}^n$ is the AR(1) process that we looked at in Chapter 1 when examining Markov chains with continuous state space. Note that here it is started from its stationary distribution.

4.2.1 Inference for latent variable models

Note that in practice, we observe $\boldsymbol{y} = (y_1, \ldots, y_n)'$ but NOT $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)'$. Let $\boldsymbol{\theta} = (\phi, \sigma, \kappa)'$ denote the parameter vector of interest. The *observed data likelihood* is given by

$$\begin{aligned} f(\boldsymbol{y}|\boldsymbol{\theta}) &= \int f(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x} \\ &= \int f(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x}. \end{aligned}$$

Typically, this integral is high dimensional and intractable.

We deal with this problem in the Bayesian paradigm by formulating the joint posterior density for $\boldsymbol{\theta}$ and $\boldsymbol{x}$ as

$$\begin{aligned}\pi(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y}) &\propto \pi(\boldsymbol{\theta}) f(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}) \\ &\propto \pi(\boldsymbol{\theta}) f(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta}).\end{aligned}$$

We then generate samples from this joint posterior using MCMC. If interest is only in $\boldsymbol{\theta}$ then we only retain parameter samples when running the MCMC scheme.

For the stochastic volatility model we have that

$$f(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}) = N\left(x_0;\, 0,\, \frac{\sigma^2}{1-\phi^2}\right) \prod_{t=1}^{n} N\left(x_t;\, \phi x_{t-1},\, \sigma^2\right) \times \prod_{t=1}^{n} N\left(y_t;\, 0,\, \kappa^2 \exp\{x_t\}\right).$$

4.2.2 MCMC sampler

To make the MCMC scheme more efficient, we will work with the transformed parameters

$$\begin{aligned}\theta_1 &= \log[\phi/(1-\phi)], \\ \theta_2 &= \log \sigma, \\ \theta_3 &= \log \kappa.\end{aligned}$$

We will use *simulated data* which were generated using ground truth values of $(\phi, \sigma, \kappa) = (0.9, 0.1, 0.5)$. Hence $\boldsymbol{\theta} \approx (-0.105, -2.303, -0.693)$. The code for generating and plotting the data is as follows.

```
## Generate synthetic data

set.seed(3421)
x=0
nt=100
xs=rep(0,nt+1)
xs[1]=x; ydata=rep(0,nt)
for (i in 1:nt)
{
 x=0.9*x+rnorm(1,0,0.1)
 xs[i+1]=x
 ydata[i]=rnorm(1,0,0.5*exp(0.5*x))
}

## Plot

par(mfrow=c(1,2))
plot(ts(xs,start=0,deltat=1),ylab="Xt",xlab="Day")
plot(ts(ydata,start=0,deltat=1),ylab="Yt",xlab="Day")
```

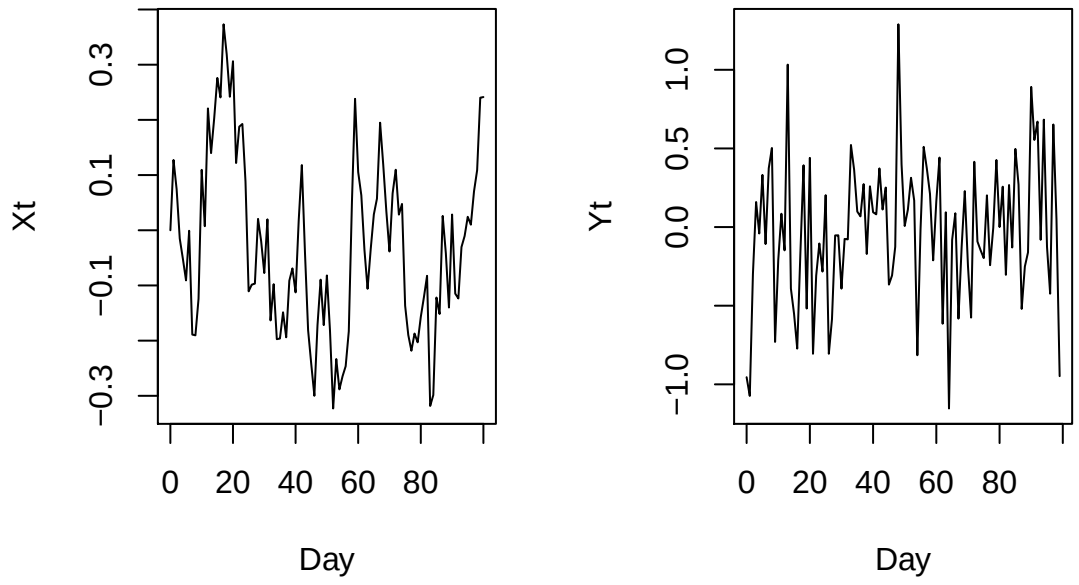

Figure 4.5: Application II. Synthetic data. Left panel: X_t (unobserved). Right panel: Y_t (observed).

The data consist of 100 observations $y_1, \ldots, y_{100}$ and are plotted above, with the corresponding values $x_0, x_1, \ldots, x_{100}$. We adopt an independent prior specification and take $\theta_1 \sim N(2, 0.1)$, $\theta_2 \sim N(0, 1)$ and $\theta_3 \sim N(0, 100)$. The following two R functions can be used to evaluate the log-prior $\log \pi(\boldsymbol{\theta})$ and log-likelihood $\log f(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta})$.

```
## Function to evaluate log-prior

lprior=function(theta)
{
 return(sum(dnorm(theta,c(2,0,0),sqrt(c(0.1,1,100)),log=TRUE)))
}

## Function to evaluate log-likelihood

llike=function(theta,data,xproc)
{
 phi=exp(theta[1])/(1+exp(theta[1]))
 sigma=exp(theta[2])
 kappa=exp(theta[3])
 ny=length(data)

 init=dnorm(xproc[1],0,sigma/sqrt(1-phi^2),log=TRUE)
```

```
 fx=sum(dnorm(xproc[2:(ny+1)],phi*xproc[1:ny],sigma,log=TRUE))
 fy=sum(dnorm(data,0,kappa*exp(0.5*xproc[2:(ny+1)]),log=TRUE))

 return(init+fx+fy)
}
```

The MCMC sampler we will use updates $\boldsymbol{\theta}$ and $\boldsymbol{x}$ in separate blocks, via random walk proposals. Hence, each iteration of the scheme has two parts:

- Update $\boldsymbol{\theta}|\boldsymbol{x}, \boldsymbol{y}$ via a 3-dimensional random walk Metropolis move, with proposal $\boldsymbol{\theta}^* \sim N(\boldsymbol{\theta}, V)$ where $V = \lambda_1^2 I_3$. The scaling λ_1 is chosen to achieve an acceptance rate of approximately 30%.
- Update $\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y}$ via an $n+1$-dimensional random walk Metropolis move, with a proposal of $\boldsymbol{x}^* \sim N(\boldsymbol{x}, \lambda_2^2 B)$, where B is the variance matrix of an AR(1) process with $\sigma = 0.1$ and $\phi = 0.9$. The scaling λ_2 is chosen to achieve an acceptance rate of approximately 25%.

Now, using that for a stationary AR(1) process,

$$Cov(X_s, X_t) = \frac{\sigma^2}{1-\phi^2}\phi^{|s-t|}$$

we use the following code to set V and B.

```
## Generate initial tuning matrix for theta update

V=diag(rep(0.05,3)) # diagonal matrix

## Generate initial tuning matrix for x update

phi=0.9
sigma=0.1
times=seq(0,length(ydata),1)
B=0.05*sigma^2/(1-phi^2)*phi^abs(outer(times,times,"-"))
```

Finally, the Metropolis sampler can be implemented via the following R function.

```
## Metropolis sampler with two blocks

metropolis=function(N,data,Vtune,Btune,theta0,x0)
{
 ny=length(data)
 mat=matrix(0,nrow=N,ncol=3) # store theta samples here
 matX=matrix(0,nrow=N,ncol=ny+1) # store x samples here
 #Initialise
```

```
 theta=theta0
 mat[1,]=theta
 x=x0
 matX[1,]=x0
 count1=0; count2=0

 for(i in 2:N)
 {
  #update parameters
  can=rmvn(theta,Vtune)
  laprob=lprior(can)+llike(can,data,x)-lprior(theta)-llike(theta,data,x)
  u=runif(1)
  if (log(u) < laprob)
  {
   theta=can
   count1=count1+1
  }
  mat[i,]=theta

  #update x process
  xcan=rmvn(x,Btune)
  laprob=llike(theta,data,xcan)-llike(theta,data,x)
  u=runif(1)
  if (log(u) < laprob)
  {
   x=xcan
   count2=count2+1
  }
  matX[i,]=x
 }
 print(c(count1/(N-1),count2/(N-1)))
 return(list(mat,matX))
}
```

We call the function with $N = 50,000$ iterations, and $\boldsymbol{\theta}$ and $\boldsymbol{x}$ initialised at the ground truth.

```
# Run and plot parameter output -- uncomment to execute
#set.seed(3421)
#out=metropolis(50000,ydata,V,B,c(2,log(0.1),log(0.5)),xs)

#phiVec=exp(out[[1]][,1])/(1+exp(out[[1]][,1]))
#sigVec=exp(out[[1]][,2])
#kappaVec=exp(out[[1]][,3])
```

```
#par(mfrow=c(3,3))
#plot(ts(phiVec),main="phi",xlab="iteration",ylab="value")
#acf(phiVec,main="",lag.max=100)
#hist(phiVec,freq=F,main="phi",xlab="value")
#plot(ts(sigVec),main="sigma",xlab="iteration",ylab="value")
#acf(sigVec,main="",lag.max=100)
#hist(sigVec,freq=F,main="sigma",xlab="value")
#plot(ts(kappaVec),main="kappa",xlab="iteration",ylab="value")
#acf(kappaVec,main="",lag.max=100)
#hist(kappaVec,freq=F,main="kappa",xlab="value")
```

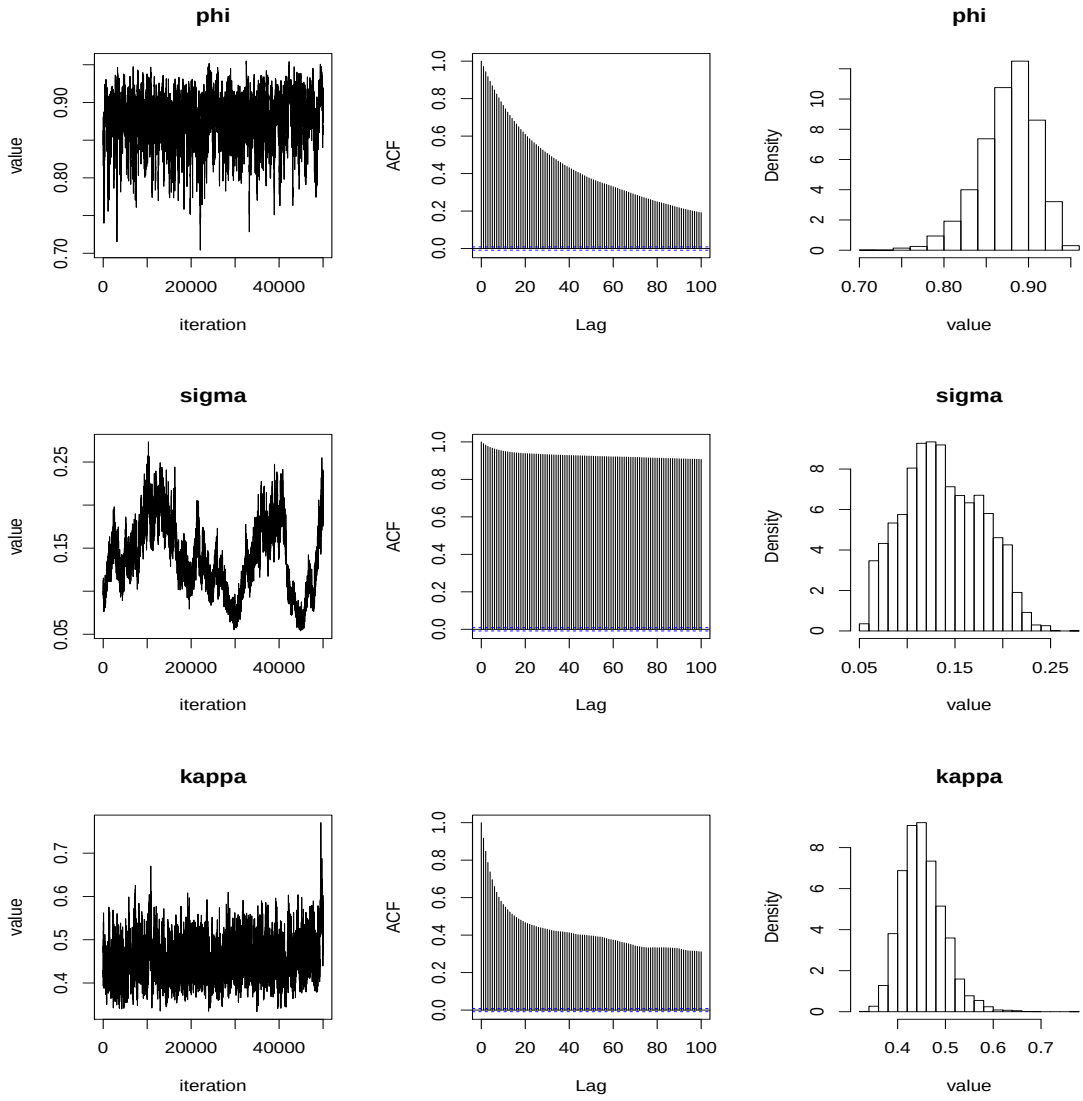

Figure 4.6: Application II. Metropolis sampler parameter output.

Remarks:

- The effective sample sizes for logit ϕ, $\log\sigma$ and $\log\kappa$ are 499, 16 and 427, suggesting that a run of at least 10 times the current length would be preferred.
- The acceptance rates for the $\boldsymbol{\theta}$ and $\boldsymbol{x}$ updates are 0.14 and 0.37 (– a little high and low resp. Needs further tuning!) respectively.
- The worst mixing chain is that for $\theta_2 = \log\sigma$. This is the *volatility of the volatility* parameter, and is well known to be hard to estimate.
- We can summarise the marginal posterior distribution of the dynamic X_t process at each time t by computing the posterior mean and a 95% equi-tailed credible interval via the following code.

```
# Summarise X process output

#xmean=apply(out[[2]],2,mean)
#lq=apply(out[[2]],2,quantile,0.025) #lower 2.5 percentile
#uq=apply(out[[2]],2,quantile,0.975) #upper 2.5 percentile

#plot(ts(xmean),ylab="Xt",ylim=c(-3,3))
#lines(ts(lq))
#lines(ts(uq))
#lines(ts(xs),col=2) #ground truth
```

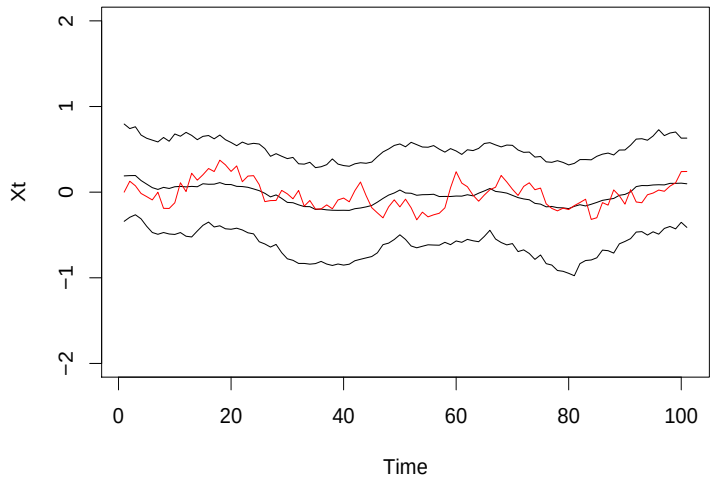

Figure 4.7: Application II. Posterior mean and 95% equi-tailed credible interval for X_t against t. The ground truth is the red line.

Contents

Chapter 5

Sequential Monte Carlo

Often, observations arrive sequentially in time and one is interested in performing inference *on-line*. From a Bayesian perspective, it is therefore necessary to update the posterior distribution as data become available. Examples include tracking aircraft using radar measurements, estimating a communications signal using noisy measurements or estimating volatility of financial instruments using stock market data.

The MCMC applications that we have looked at so far can be thought of as *batch analyses*, that is, we run the sampler for all available data we are in receipt of. A naive approach to implementing MCMC in a sequential fashion involves re-running the sampler (from scratch) as each observation arrives. This is wasteful (in terms of both storage and computational cost). A different approach is therefore required. This will involve a cunning implementation of the *weighted resampling* algorithm of Chapter 1.

5.1 Motivating examples

Suppose that we have data $y_{0:T} = (y_0, \ldots, y_T)'$, of a process observed discretely and noisily over a time interval $[0, T]$.

Example 5.1.1. (Stochastic volatility.) The model is

$$\begin{aligned} X_0 &\sim N\left(0, \frac{\sigma^2}{1-\phi^2}\right), \\ X_t | X_{t-1} = x_{t-1} &\sim N\left(\phi x_{t-1}, \sigma^2\right), \quad t = 1, \ldots, T, \\ Y_t | X_t = x_t &\sim N\left(0, \kappa^2 \exp\{x_t\}\right), \quad t = 0, \ldots, T. \end{aligned}$$

Remarks:

- X_t, $t = 0, \ldots, T$ is a *hidden Markov process* since it is unobserved and Markovian.
- Conditional on $x_{0:T}$, a particular observation only depends on the value of the hidden process at the observation time and is independent of all other observations.

Example 5.1.2. (Tracking a bird/plane/missile (in 1-d).) Consider the following model for the evolution of the position s and velocity v of a bird/plane/missile and a noisy observation of its position y:

$$\begin{aligned} S_0 &\sim N(0,\, 1/4), \quad V_0 = \frac{\sigma}{\sqrt{1-\phi^2}}\epsilon_0, \\ S_t &= S_{t-1} + V_{t-1}, \quad t = 1, \ldots, T, \\ V_t &= \phi V_{t-1} + \sigma\epsilon_t, \quad t = 1, \ldots, T \\ Y_t &= S_t + \omega_t, \quad t = 0, \ldots, T \end{aligned}$$

where $\epsilon_t \sim t_5$ are iid, $\omega_t \sim N(0, \kappa^2)$ are iid, $\phi \in (0, 1)$, $\sigma > 0$ and $\kappa > 0$. The structure is the same as the SV model: $X_t = (S_t, V_t)'$ is Markovian and is unobserved; the observation Y_t is a noisy function of X_t.

Suppose that $\phi = 0.9$, $\sigma = 0.5$ and $\kappa = 1$. The following suite of R functions can be used to simulate the process with $T = 50$.

```
## Tracking model

## Function to simulate initial condition

x0Sim=function(theta)
{
 phi=theta[1]; sigma=theta[2]; kappa=theta[3]
 return(c(0.5*rnorm(1),rt(1,df=5)*sigma/sqrt(1-phi^2)))
}

## Function to simulate X_t|X_{t-1}=x

xtSim=function(x,theta)
{
 phi=theta[1]; sigma=theta[2]; kappa=theta[3]
 xnew=rep(0,2)
 xnew=c(x[1]+x[2],phi*x[2]+sigma*rt(1,df=5))
 return(xnew)
}

## Function to simulate Y_t|X_t=x

ytSim=function(x,theta)
{
 phi=theta[1]; sigma=theta[2]; kappa=theta[3]
 return(x[1]+kappa*rnorm(1))
}

## Function to simulate {X_t} and {Y_t} over [0,T]
```

```
modelSim=function(T,theta)
{
 Xmat=matrix(0,nrow=T+1,ncol=2)
 Yvec=rep(0,T+1)
 x=x0Sim(theta); y=ytSim(x,theta)
 Xmat[1,]=x; Yvec[1]=y
 for(i in 2:(T+1))
 {
  x=xtSim(x,theta); y=ytSim(x,theta)
  Xmat[i,]=x; Yvec[i]=y
 }
 return(list(Xmat,Yvec))
}

## Run and plot

set.seed(5)
out=modelSim(50,c(0.9,0.5,1))

par(mfrow=c(1,2))
plot(ts(out[[1]][,2]),ylab="Vt",xlab="t")
plot(ts(out[[1]][,1]),ylab="St",xlab="t")
lines(ts(out[[2]]),type="p")
```

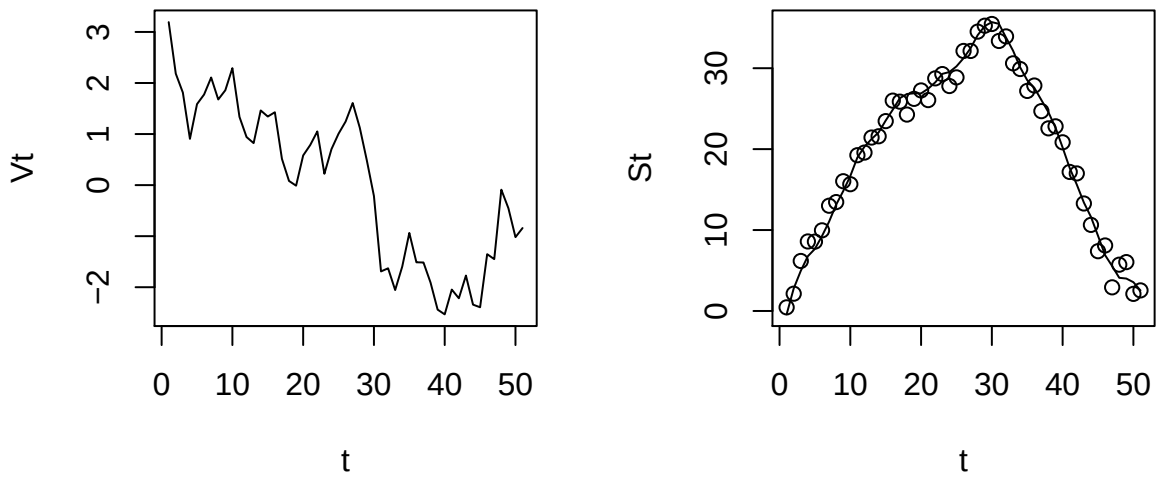

Figure 5.1: Tracking model. Left panel: Velocity (unobserved). Right panel: Position (unobserved) and noisy observations thereof.

5.2 Hidden Markov models and aims

A discretely observed *hidden Markov model (HMM)* is a Markov model where the true state is not observed; instead only a noisy function of the state is observed, and only at a discrete set of time points. An HMM is also known as a *state-space model*.

Definition: *A state-space model is a time series model that consists of two discrete-time processes* $\{X_t, t \geq 0\}$ *and* $\{Y_t, t \geq 0\}$ *taking values respectively in spaces* $\mathcal{X}$ *and* $\mathcal{Y}$*. The model is specified by densities* $p(x_0|\boldsymbol{\theta})$*,* $p(x_t|x_{0:t-1}, \boldsymbol{\theta}) = p(x_t|x_{t-1}, \boldsymbol{\theta})$ *and* $p(y_t|x_t, \boldsymbol{\theta})$*, for some parameter vector* $\boldsymbol{\theta}$*.*

Remarks:

- The above definition describes a generative probabilistic model, where X_0 is drawn from the initial density $p(x_0|\boldsymbol{\theta})$, and then each X_t is drawn conditionally on the previous draw $X_{t-1} = x_{t-1}$ according to the density $p(x_t|x_{t-1}, \boldsymbol{\theta})$, and each Y_t conditionally on the most recent $X_t = x_t$ from $p(y_t|x_t, \boldsymbol{\theta})$.
- The model can be represented graphically by having variables as nodes and an edge between two variables that are related by one of the kernels in the above definition.

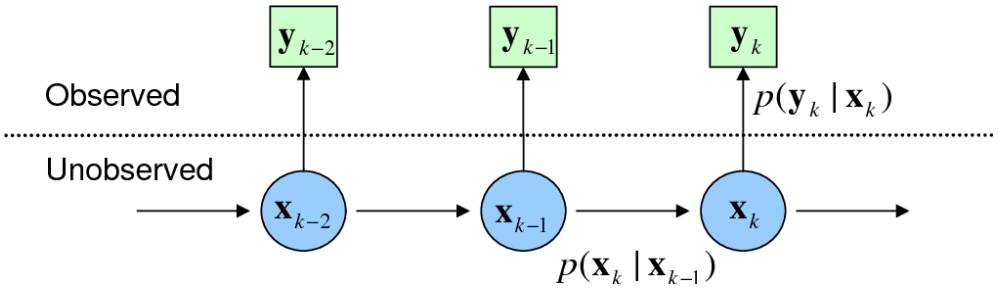

Figure 5.2: Partially observed Markov process as a directed graphical model.

5.2.1 Aims

When analysing HMMs, there are typically 3 key aims:

- **Filtering:** after a new observation arrives, we wish to learn the corresponding hidden state via the filtering density $p(x_t|y_{0:t})$.
- **Smoothing:** given all of the data, learn the hidden states via the smoothing density $p(x_{0:t}|y_{0:T})$.
- **Parameter inference:** learn the parameters via the (marginal) posterior density $\pi(\boldsymbol{\theta}|y_{0:T})$.

The first two aims are tractable (that is, given the parameters, the filtering and smoothing distributions are available in closed form) when:

1. the unobserved Markov chain has a finite number of states or

2. the unobserved Markov chain is a Gaussian autoregressive process (explored further in the tutorial questions).

Except in these two cases, *sequential Monte Carlo* methods provide the best generic methods for filtering and smoothing. These methods are known as *particle filtering* and *particle smoothing*. The remainder of this chapter will focus on *particle filtering*, which is based on the idea of approximating distributions by *weighted samples*. We will therefore have a quick refresher of the weighted resampling algorithm, and it's use within the context of an HMM.

For now we will assume that we know the parameter values and drop them from the notation where possible.

5.3 Weighted resampling for HMMs

Consider a target density $f(x)$ and a proposal density $g(x)$. We can view weighted resampling algorithmically as follows:

1. Gnerate N iid samples $x^{(1)}, \ldots, x^{(N)}$ from $g(x)$.
2. Construct and normalise the importance weights. For $i = 1, \ldots, N$:
$$w(x^{(i)}) = \frac{f(x^{(i)})}{g(x^{(i)})}, \quad \tilde{w}(x^{(i)}) = \frac{w(x^{(i)})}{\sum_{k=1}^{N} w(x^{(k)})}.$$
3. Resample (with replacement) from the discrete distribution on $\{x^{(1)}, \ldots, x^{(N)}\}$ with associated probabilities $\{\tilde{w}(x^{(1)}), \ldots, \tilde{w}(x^{(N)})\}$.

Remarks:

- Recall that after step 2, we have an *empirical approximation* of $f(x)$ as
$$\hat{f}(x) = \sum_{i=1}^{N} \tilde{w}(x^{(i)}) \delta(x - x^{(i)})$$
where δ denotes the Dirac mass function.
- After step 3, we have an equally weighted sample $\{x^{(1)}, \ldots, x^{(N)}\}$, approximately distributed according to f.
- An obvious question is how to choose N. The accuracy of an estimator of $\mathbb{E}_f[h(X)]$ will depend on the weight function and $h(\cdot)$.
- As a rough guide, the *effective sample size*, which is estimated as
$$\text{ESS} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}(x^{(i)})\right)^2}$$
gives the number of iid samples necessary to obtain (roughly) the same accuracy as the estimator based on the weighted sample. Note, this form is specific to the notion of weighted resampling and should not be confused with the form of ESS used to assess MCMC output accuracy.

5.3.1 Application to HMMs

Consider a target of the form

$$p(x_{0:T}|y_{0:T}) \propto p(x_{0:T})p(y_{0:T}|x_{0:T})$$

where

$$p(x_{0:T}) = p(x_0)\prod_{t=1}^{T} p(x_t|x_{t-1}),$$
$$p(y_{0:T}|x_{0:T}) = \prod_{t=0}^{T} p(y_t|x_t).$$

Now take the proposal density to be $g(x_{0:T}) = p(x_{0:T})$. To sample a path, we first simulate the initial value x_0 using $p(x_0)$, then given this initial value we draw x_1 from $p(x_1|x_0)$ etc. The weight for sample $x_{0:T}^{(i)}$ is

$$w(x_{0:T}^{(i)}) = p(y_{0:T}|x_{0:T}^{(i)}).$$

Example 5.3.1. (Tracking model revisited.) Consider the tracking model and data. Using $g(x_{0:T}) = p(x_{0:T})$ as the proposal density gives the weight of a sample $x_{0:T}^{(i)}$ as

$$w(x_{0:T}^{(i)}) = \prod_{t=0}^{T} N\left(y_t;\, s_t^{(i)},\, \kappa^2\right).$$

Take $N = 5$. The plot below shows the data and the 5 proposed realisations.

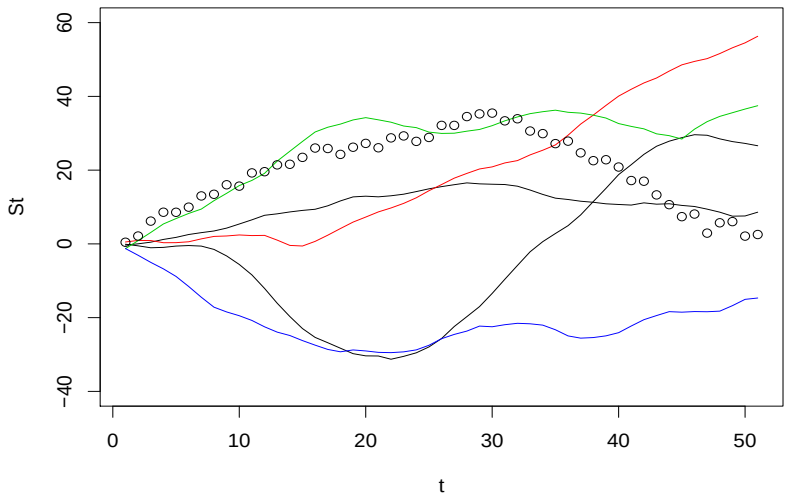

Figure 5.3: Tracking model. Data and 5 realisations of $X_{0:T}$.

The normalised weights are $1, 0, 0, 0, 0,$ and the ESS is 1. If we reduce T to 4 (giving 5 observations) and increase N to 100, we obtain an ESS of around 3. However, reducing T to 1 (giving 2 observations) gives an ESS of around 40, and reducing T further still to 0 (giving 1 observation) gives an ESS of around 96.

Remarks:

- The effective sample size can degrade very quickly as the number of observations increases; typically, the degradation is exponential in the number of observations.
- The degradation is manageable with just one observation.
- Applying weighted resampling to each observation in turn might work! In fact, we can do this by exploiting the structure of the HMM.

First, we will describe the ideal 'one-at-a-time' updates before considering the weighted resampling approximation to these updates; use of weighted resampling in this way is known as *particle filtering*.

5.4 Filtering recursions

We need the *filtering densities* $p(x_t|y_{0:t})$, $t = 0, \ldots, T$. It will be helpful to note that *the special structure* of the HMM means that

$$p(x_t|x_{t-1}, y_{0:t-1}) = p(x_t|x_{t-1}), \quad \text{and} \quad p(y_t|x_t, y_{0:t-1}) = p(y_t|x_t).$$

- Consider $t = 0$. Using Bayes Theorem, we have that

$$p(x_0|y_0) = \frac{p(x_0)p(y_0|x_0)}{p(y_0)}$$

where the numerator is $p(x_0, y_0)$. Note that the *normalising constant* is $p(y_0) = \int p(x_0, y_0)dx_0$. Hence,

$$\boxed{p(x_0|y_0) \propto p(x_0)p(y_0|x_0).}$$

- Now consider $t \in \{1, 2, \ldots, T\}$ and suppose that we 'know' $p(x_{t-1}|y_{0:t-1})$. Bayes Theorem gives

$$\begin{aligned} p(x_t|y_{0:t-1}, y_t) &= \frac{p(x_t, y_t|y_{0:t-1})}{p(y_t|y_{0:t-1})} \\ &= \frac{p(x_t|y_{0:t-1})p(y_t|x_t, y_{0:t-1})}{p(y_t|y_{0:t-1})} \\ &= \frac{p(x_t|y_{0:t-1})p(y_t|x_t)}{p(y_t|y_{0:t-1})} \end{aligned}$$

where the last line uses the special structure of the HMM. Now, write the first term in the numerator as

$$\begin{aligned} p(x_t|y_{0:t-1}) &= \int p(x_t, x_{t-1}|y_{0:t-1})dx_{t-1} \\ &= \int p(x_{t-1}|y_{0:t-1})p(x_t|x_{t-1}, y_{0:t-1})dx_{t-1} \\ &= \int p(x_{t-1}|y_{0:t-1})p(x_t|x_{t-1})dx_{t-1} \end{aligned}$$

where the last line again uses the special structure of the HMM. Putting it all together gives

$$\boxed{p(x_t|y_{0:t}) \propto p(y_t|x_t) \int p(x_{t-1}|y_{0:t-1})p(x_t|x_{t-1})dx_{t-1}.}$$

Note that the *normalising constant* is $p(y_t|y_{0:t-1})) = \int p(x_t, y_t|y_{0:t-1})dx_t$.

- The *observed data likelihood* is

$$p(y_{0:T}) = p(y_0) \prod_{t=1}^{T} p(y_t|y_{0:t-1}).$$

As already stated, these densities are intracatable except in two special cases. We will therefore look at a Monte Carlo method for approximating the filtering densities.

5.5 The particle filter

Consider weighted resampling to approximate $p(x_0|y_0)$ by drawing $x_0^{(i)}$ from $p(x_0)$, $i = 1, \ldots, N$ and constructing weights $w_0(x_0^{(i)}) = p(y_0|x_0^{(i)})$. Normalising the weights gives

$$\tilde{w}_0(x_0^{(i)}) = \frac{w_0(x_0^{(i)})}{\sum_{k=1}^{N} w_0(x_0^{(k)})}.$$

Hence, we may approximate $p(x_0|y_0)$ via

$$\widehat{p}(x_0|y_0) = \sum_{i=1}^{N} \tilde{w}_0(x_0^{(i)})\delta(x_0 - x_0^{(i)}).$$

The particle filter then samples N times from this distribution to get a new, *unweighted* sample from $\widehat{p}(x_0|y_0) \approx p(x_0|y_0)$. This is equivalent to resampling (with replacement) from the existing sample with probabilities given by the normalised weights.

These steps can be summarised as follows:

1. **Initialise** with an unweighted sample of size N drawn from the prior distribution of x_0, $p(x_0)$.
2. **Weight** each sample $x_0^{(i)}$ by the likelihood $p(y_0|x_0^{(i)})$.

3. **Resample**: simulate N values of x_0 with probabilities proportional to the weights to obtain an approximate unweighted sample from $p(x_0|y_0)$. Denote the resulting sample by $x_0^{(1)}, \ldots, x_0^{(N)}$.

Now we observe y_1 and we wish to obtain a sample approximately distributed according to $p(x_1|y_{0:1})$. We construct the approximation

$$\widehat{p}(x_1|y_{0:1}) \propto p(y_1|x_1)\widehat{p}(x_1|y_0).$$

Crucially, $\widehat{p}(x_1|y_0)$ is **easy** to sample from, by taking each $x_0^{(i)}|y_0$ from step 3 above and drawing from $p(x_1|x_0^{(i)})$ to obtain $x_1^{(i)}|y_0$, $i = 1, \ldots, N$. Hence, applying weighted resampling to the target $\widehat{p}(x_1|y_{0:1})$ with proposal density $\widehat{p}(x_1|y_0)$ performs the following steps:

1. **Propagate**: for each sample $x_0^{(i)}|y_0$, draw from $p(x_1|x_0^{(i)})$ to obtain $x_1^{(i)}|y_0$, $i = 1, \ldots, N$.
2. **weight** each $x_1^{(i)}|y_0$ by the likelihood $p(y_1|x_1^{(i)})$.
3. **Resample**: simulate N values of x_1 with probabilities proportional to the weights to obtain an approximate unweighted sample from $p(x_1|y_{0:1})$. Denote the resulting sample by $x_1^{(1)}, \ldots, x_1^{(N)}$.

Repeating these steps, in general to deal with observation y_t gives:

1. **Propagate**: for each sample $x_{t-1}^{(i)}|y_{0:t-1}$, draw from $p(x_t|x_{t-1}^{(i)})$ to obtain $x_t^{(i)}|y_{0:t-1}$, $i = 1, \ldots, N$.
2. **weight** each $x_t^{(i)}|y_{0:t-1}$ by the likelihood $p(y_t|x_t^{(i)})$.
3. **Resample**: simulate N values of x_t with probabilities proportional to the weights to obtain an approximate unweighted sample from $p(x_t|y_{0:t})$. Denote the resulting sample by $x_t^{(1)}, \ldots, x_t^{(N)}$.

Remarks:

- After each step 2, we have the approximation

$$\widehat{p}(x_t|y_{0:t}) = \sum_{i=1}^{N} \tilde{w}_t(x_t^{(i)})\delta(x_t - x_t^{(i)})$$

  based on the weighted sample.

- After each step 3, this becomes

$$\widehat{p}(x_t|y_{0:t}) = \frac{1}{N}\sum_{i=1}^{N} \delta(x_t - x_t^{(i)})$$

  and can be (straightforwardly) used to estimate expectations of the form $\mathbb{E}_{X_t|y_{0:t}}[h(X_t)]$ by the equivalent sample average.

- Executing steps 1–3 recursively corresponds to the *sample-importance-resampling* filter of Gordon, Salmond and Smith (1993), also known as the *bootstrap particle filter*.

Example 5.5.1. (Tracking model revisited.) The following R code implements the bootstrap particle filter (BPF) for this model. The first function is used for weight evaluation in the BPF.

```
## Bootstrap particle filter for tracking model
## Uses functions x0Sim and xtSim above

## Function to evaluate log p(y_t|x_t)

llike=function(y,x,theta)
{
 phi=theta[1]; sigma=theta[2]; kappa=theta[3]
 return(-0.5*(y-x[1])^2/kappa^2)
}

## Function to implement particle filter to track position

BPF=function(N,theta,ydata)
{
 endT=length(ydata)
 # Store unweighted samples from filtering density for position here:
 mat=matrix(0,nrow=N,ncol=endT)
 wts=rep(0,N) # Store weights here
 x=matrix(0,nrow=N,ncol=2) # Samples of current state X_t=(S_t,V_t)
 # Initialise (t=0)
 for(j in 1:N)
 {
  x[j,]=x0Sim(theta) #Sample prior
  wts[j]=llike(ydata[1],x[j,1],theta) # Weight
 }
 x=x[sample(1:N,N,TRUE,exp(wts)),] # Resample
 mat[,1]=x[,1] # Store position
 # Loop for times t=1,...,T
 for(i in 2:endT)
 {
  for(j in 1:N)
  {
    x[j,]=xtSim(x[j,],theta)  # Propagate
    wts[j]=llike(ydata[i],x[j,1],theta) # Weight
  }
  x=x[sample(1:N,N,TRUE,exp(wts)),] # Resample
  mat[,i]=x[,1] # Store position
 }
 return(mat)
}
```

We then generate the data and run the BPF as folows.

```
## Generate synthetic data and store in out
set.seed(5)
out=modelSim(50,c(0.9,0.5,1))

## Run BPF
filt=BPF(100,c(0.9,0.5,1),out[[2]])

## Construct and plot filtering summaries
filt_mean=apply(filt,2,mean)
filt_lq=apply(filt,2,quantile,0.025)
filt_uq=apply(filt,2,quantile,0.975)

plot(ts(filt_mean),ylab="St",xlab="t",ylim=c(-2,40))
lines(ts(filt_lq))
lines(ts(filt_uq))
lines(ts(out[[1]][,1]),type="p") # ground truth position
```

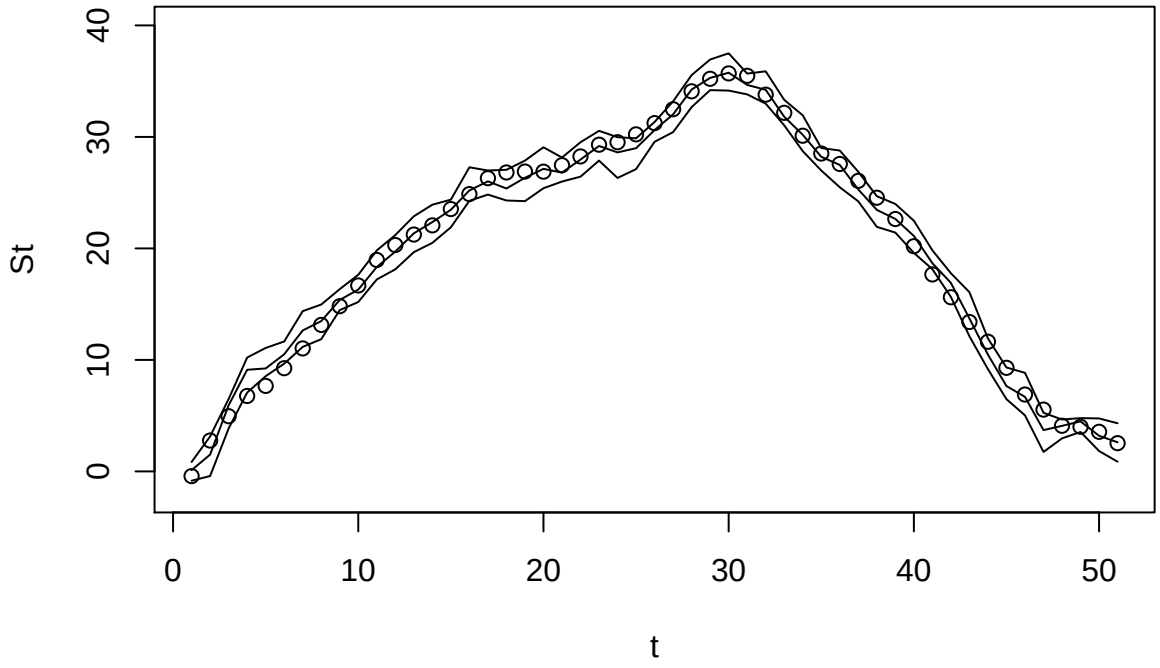

Figure 5.4: Tracking model, filtering distribution summaries. Mean and 95 percent credible interval for hidden position. The ground truth values of position are shown by the circles.

5.6 Observed data likelihood

Recall that the *observed data likelihood* is

$$p(y_{0:T}) = p(y_0) \prod_{t=1}^{T} p(y_t|y_{0:t-1}).$$

Now, we have that $\widehat{p}(x_0) = \frac{1}{N} \sum_{i=1}^{N} \delta(x_0 - x_0^{(i)})$, so

$$\begin{aligned} p(y_0) &= \int p(x_0)p(y_0|x_0)dx_0 \\ &\approx \int \widehat{p}(x_0)p(y_0|x_0)dx_0 \\ &= \frac{1}{N} \sum_{i=1}^{N} p(y_0|x_0^{(i)}) \\ &= \frac{1}{N} \sum_{i=1}^{N} w_0(x_0^{(i)}). \end{aligned}$$

After y_{t-1} has been assimilated and the hidden process simulated forward to time t, we have that $\widehat{p}(x_t|y_{0:t-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta(x_t - x_t^{(i)})$. Hence,

$$\begin{aligned} p(y_t|y_{0:t-1}) &= \int p(x_t|y_{0:t-1})p(y_t|x_t)dx_t \\ &\approx \widehat{p}(x_t|y_{0:t-1})p(y_t|x_t)dx_t \\ &= \frac{1}{N} \sum_{i=1}^{N} p(y_t|x_t^{(i)}) \\ &= \frac{1}{N} \sum_{i=1}^{N} w_t(x_t^{(i)}). \end{aligned}$$

Therefore, we may estimate the observed data likelihood via

$$\widehat{p}(y_{0:T}) = \widehat{p}(y_0) \prod_{t=1}^{T} \widehat{p}(y_t|y_{0:t-1}).$$

Remarks:

- The above estimator is *unbiased* (although hard to prove, and beyond the scope of this course).
- The estimator is the product (over time points) of the average un-normalised weight.

Example 5.6.1. (Tracking model revisited.) The `BPF` function can be easily modified to calculate the average weight stored in the `wts` vector. Taking the logarithm of this average and adding up over time points gives an estimate of the log (observed data) likelihood. For example, calculating 3 realisations of the log-likelihood for different values of N gives:

```
## N=10
# -77.34225
# -91.22439
# -114.1485
## N=50
# -63.63864
# -58.62918
# -62.57907
## N=500
# -55.24049
# -57.28235
# -57.2842
```

The ground truth is around -56.1. Plainly, increasing N will lead to greater accuracy but increases computational cost.

5.7 Better particle filters

Upon receipt of the observation y_t, the bootstrap particle filter targets

$$\widehat{p}(x_t|y_{0:t}) \propto p(y_t|x_t)\widehat{p}(x_t|y_{0:t-1})$$

by using $\widehat{p}(x_t|y_{0:t-1})$ as the proposal density. The proposal is sampled by taking each $x_{t-1}^{(i)}|y_{0:t-1}$ and drawing from $p(x_t|x_{t-1}^{(i)})$ to obtain $x_t^{(i)}|y_{0:t-1}$, $i = 1, \ldots, N$. This leads to the weight

$$w_t(x_t^{(i)}) = \frac{\widehat{p}(x_t^{(i)}|y_{0:t})}{\widehat{p}(x_t^{(i)}|y_{0:t-1})} = p(y_t|x_t^{(i)}).$$

Note that the particle $x_t^{(i)}$ is generated via forward simulation from $p(x_t|x_{t-1}^{(i)})$, which does not depend on y_t at all. In fact, we are 'free' to generate particles from an arbitrary proposal density $g(x_t|x_{t-1})$ for which the importance weight is

$$w_t(x_t^{(i)}) = \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{g(x_t^{(i)}|x_{t-1}^{(i)})}.$$

It can be shown that (see tutorial questions) $g(x_t|x_{t-1}) = p(x_t|x_{t-1}, y_t)$ is optimal in the sense of minimising the variance of the weights. Unfortunately, $p(x_t|x_{t-1}, y_t)$ is rarely tractable, and we instead seek some approximation thereof, for propagating particles.

5.8 Particle filter theory in brief

There is extensive theory underpinning particle filters. In a nutshell, if the model mixes quickly, then it is possible to develop particle filters whose Monte Carlo variability is bounded in time. We call such filters *stable*. Roughly, the idea of quick mixing is related to requiring the amount the distribution $X_T|y_{0:T}, x_t$ depends on x_t to decay exponentially as $T-t$ increases. Put another way, to prevent the asymptotic variance from blowing up, we need the impact of past time steps to vaish; that is, that the past is forgotten in some sense.

Resampling is crucial to obtaining long-term stability of particle filter algorithms (when it is possible).

Filters for the *path* of the hidden state are not *stable* in time; nor is the estimate of the marginal likelihood.

For a stable process, subject to conditions, we have approximately, if N is large enough (e.g., Chopin, 2004, Ann. Stat.) and T is fixed:

$$\sum_{i=1}^{N} w_T(x_T^{(i)})h(x_T^{(i)}) - \mathbb{E}\left[h(X_T)|y_{0:T}\right] \sim N(0, \frac{1}{N}\lambda_{h,T}^2),$$

for some $\lambda_{h,T}^2$. For fixed T, the approximation to $\mathbb{E}[h(X_T)|y_{0:T}]$ gets better and better as $N \to \infty$ with the usual $1/N$ rate for Monte Carlo approximations. Importantly, for a fixed h, $\lambda_{h,T}$ is bounded as $t \to \infty$; thus for calculating expectations at the final time point, the behaviour is stable when the number of particles is fixed.

Now set $\widehat{p}(y_{0:T}) = \prod_{t=0}^{T} \frac{1}{N}\sum_{i=1}^{N} w_t(x_t^{(i)})$. For any fixed T and large N, approximately

$$\frac{1}{p(y_{0:T})}\{\widehat{p}(y_{0:T}) - p(y_{0:T})\} \stackrel{D}{=} \frac{1}{\sqrt{N}}V_T$$

for some fixed random variable V_T.

Finally, if T is large and N is of similar size, then approximately:

$$\log \widehat{p}(y_{0:T}) - \log p(y_{0:T}) \sim N\left(-\frac{T}{2N}\lambda^2\,,\, \frac{T}{N}\lambda^2\right)$$

for some λ^2. Hence, as the length of the time series increases, for stable behaviour in the calculation of the observed data likelihood, the number of particles should be increased in proportion to T.

Contents

Chapter 6

Other methods

This chapter provides a short explanation of other methods you might come across, with references for further, self-directed study (where appropriate).

6.1 Laplace approximation

While for many models, Markov chain Monte Carlo is the approximate inference method of choice, the Laplace approximation still provides the simplest deterministic method available.

In essence, the Laplace approximation entails finding a Gaussian approximation to a continuous probability density. Let's consider a univariate continuous random variable θ with probability density function $p(\theta)$. This could be a Bayesian posterior. Suppose that

$$p(\theta) = \frac{1}{Z} f(\theta)$$

where $Z = \int f(\theta) d\theta$ is the normalising constant. The goal of the Laplace approximation is to find a Gaussian approximation $q(\theta)$ to the target $p(\theta)$. The mean of this approximation is specified to be the *mode* of the target, which we will denote by θ_0. Now, we construct a Taylor expansion of $\log f(\theta)$ about θ_0 to provide an approximation for $\log f(\theta)$ as

$$\log f(\theta) \approx \log f(\theta_0) - \frac{1}{2} H \left(\theta - \theta_0\right)^2$$

where

$$H = -\frac{d^2}{d\theta^2} \log f(\theta) \bigg|_{\theta=\theta_0}.$$

Exponentiating gives

$$f(\theta) \approx f(\theta_0) \exp\left\{-\frac{1}{2} H \left(\theta - \theta_0\right)^2\right\}$$

which is proportional to a Gaussian kernel with mean θ_0 and variance H^{-1}. Normalising gives the Laplace approximation

$$q(\theta) = \frac{H^{1/2}}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2} \frac{(\theta - \theta_0)^2}{H^{-1}}\right\}.$$

We can extend the Laplace approximation for a multivariate distribution. For the d-dimensional target $p(\boldsymbol{\theta}) \propto f(\boldsymbol{\theta})$ we have that

$$\log f(\boldsymbol{\theta}) \approx \log f(\boldsymbol{\theta}_0) - \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)' H (\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

where H is the *Hessian* matrix, that is, the matrix of second-order partial derivatives which describes the local curvature of $\log f(\boldsymbol{\theta})$ at $\boldsymbol{\theta}_0$. In particular, the (i,j)th element of the $d \times d$ matrix H is

$$(H)_{i,j} = \frac{\partial^2 f}{\partial \theta_i \partial \theta_j}.$$

Exponentiating and including the normalisation constant for a multivariate normal gives the approximate multivariate distribution

$$q(\boldsymbol{\theta}) = \frac{|H|^{1/2}}{(2\pi)^{d/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)' H (\boldsymbol{\theta} - \boldsymbol{\theta}_0)\right\}.$$

To summarise, the Laplace approximation is

$$q(\boldsymbol{\theta}) = N\left(\boldsymbol{\theta}; \boldsymbol{\theta}_0, H^{-1}\right).$$

6.2 Variational Bayes

Consider the independence (MCMC) sampler from Chapter 3. This will work well when the proposal density q is 'close' to the target density π. For example, provided the dimension of the target is not too large, we might use the Laplace approximation so that the proposal distribution is a multivariate normal with parameters chosen so that the approximation is reasonable. Variational Bayes seeks to find the "best" (over the possible parameter values) approximation q, to π across a particular class of distributions such as all multivariate normal distributions, or product distributions with particular univariate distributions for each component. It then uses this "best" q directly for inference as if it were the true target.

Suppose that the target is a Bayesian posterior $\pi(\boldsymbol{\theta}|\boldsymbol{x})$. "Best" is defined according to a particular measure, the *Kullback-Leibler divergence*:

$$KL\left(q(\boldsymbol{\theta})||\pi(\boldsymbol{\theta}|\boldsymbol{x})\right) = \mathbb{E}_q\left[\log\left(\frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}|\boldsymbol{x})}\right)\right].$$

Now, $KL(q(\boldsymbol{\theta})||\pi(\boldsymbol{\theta}|\boldsymbol{x})) \geq 0$ with equality if and only if $q = \pi$. Moreover,

$$\begin{aligned} KL\left(q(\boldsymbol{\theta})||\pi(\boldsymbol{\theta}|\boldsymbol{x})\right) &= \int q(\boldsymbol{\theta}) \log q(\boldsymbol{\theta}) d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \log \pi(\boldsymbol{\theta}|\boldsymbol{x}) d\boldsymbol{\theta} \\ &= \int q(\boldsymbol{\theta}) \log q(\boldsymbol{\theta}) d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \log\{\pi(\boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta})\} d\boldsymbol{\theta} + \int q(\boldsymbol{\theta}) \log p(\boldsymbol{x}) d\boldsymbol{\theta} \\ &= \int q(\boldsymbol{\theta}) \log q(\boldsymbol{\theta}) d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \log\{\pi(\boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta})\} d\boldsymbol{\theta} + \log p(\boldsymbol{x}). \end{aligned}$$

Now, since $p(\boldsymbol{x})$ is a constant, minimising the KL-divergence of the class of densities for q is equivalent to minimising

$$\int q(\boldsymbol{\theta}) \log q(\boldsymbol{\theta}) d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \log\{\pi(\boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta})\} d\boldsymbol{\theta}.$$

As with MCMC therefore, we do not need to know the normalising constant $p(\boldsymbol{x})$.

Further reading: Blei, Kucukelbir and McAuliffe (2017), "Variational inference: a review for statisticians", JASA.

6.3 Simulated annealing

This is a method for finding global extrema of arbitrary functions. Consider first the problem of finding the global mode(s) of a distribution. We could estimate the mode of a distribution with density $f(\boldsymbol{x})$ by generating samples $\{\boldsymbol{x}^{(n)}\}$ and finding the sample with maximal density. However, this is inefficient since the sampling procedure will explore the whole distribution rather than the mode.

This suggests modifying the distribution so that it is more concentrated around the mode(s). One of way of achieving this is to consider the density

$$f_\beta(\boldsymbol{x}) \propto [f(\boldsymbol{x})]^\beta$$

for very large values of β. As an example, consider $X \sim N(\mu, \sigma^2)$ and raise the density to the power β to see that the effect is to scale the variance by $1/\beta$, so that the larger we make β, the more concentrated the distribution around the mode μ. It turns out that this result holds in general and as $\beta \to \infty$, $f_\beta(\boldsymbol{x})$ converges to a distribution that has all mass on the mode(s) of f.

Hence, as a first attempt at mode finding, we could use MCMC to generate samples from $f_\beta(\boldsymbol{x})$ with β chosen to be some large value. The problem here is that the resulting Markov chain will have poor mixing properties, for example, by finding it difficult to move away from a local extremum surrounded by areas of low probability (the density of such a distribution will have many local extrema separated y areas where the density is effectively 0).

The idea of *simulated annealing* is to sample from a target distribution that changes over time: $f_{\beta_t}(\boldsymbol{x})$ with $\beta_t \to \infty$. Two possible methods for updating β_t include *logarithmic tempering* with $\beta_t = \frac{\log(1+t)}{\beta_0}$ and geometric tempering with $\beta_t = a^t \beta_0$ for some $a > 1$.

Suppose now that we want to find the global minimum of a function $h(\boldsymbol{x})$ with $h : E \to \mathbb{R}$. Finding the global minimum of $h(\boldsymbol{x})$ is equivalent to finding the mode of a distribution with density

$$f(\boldsymbol{x}) \propto \exp\{-h(\boldsymbol{x})\}, \quad \boldsymbol{x} \in E,$$

if such a distribution exists. We then raise f to large powers to obtain a distribution

$$f_{\beta_t}(\boldsymbol{x}) = [f(\boldsymbol{x})]^{\beta_t} \propto \exp\{-\beta_t h(\boldsymbol{x})\}, \quad \boldsymbol{x} \in E.$$

We hope to find the (global) minimum of $h(\boldsymbol{x})$, which is the (global) model of the distribution with density $f_{\beta_t}(\boldsymbol{x})$, by sampling from a Metropolis-Hastings algorithm. This yields the following steps.

1. Initialise with $\boldsymbol{x}^{(0)}$ and $\beta_0 > 0$. Set $j = 1$.
2. Increase β_{j-1} to β_j. Propose $\boldsymbol{x}^* \sim q(\cdot|\boldsymbol{x}^{(j-1)})$.
3. Compute the acceptance probability

$$\alpha(\boldsymbol{x}^*|\boldsymbol{x}) = \min\left\{1\,,\,\exp\left(-\beta_j[h(\boldsymbol{x}^*) - h(\boldsymbol{x})]\right) \times \frac{q(\boldsymbol{x}|\boldsymbol{x}^*)}{q(\boldsymbol{x}^*|\boldsymbol{x})}\right\}.$$

4. With probability $\alpha(\boldsymbol{x}^*|\boldsymbol{x}^{(j-1)})$ set $\boldsymbol{x}^{(j)} = \boldsymbol{x}^*$ otherwise set $\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(j-1)}$.
5. Put $j := j + 1$ and go to step 2.

The algorithm will converge to a global minimum of $h(\cdot)$. Whether or not it will find the global minimum depends on how slowly β increases.

Finally, it is worth mentioning the related method of *parallel tempering*. For a multimodal target density $f(\boldsymbol{x})$, a density proportional to $f(\boldsymbol{x})^{1/\beta}$, $\beta > 1$, has broader, flatter modes and troughs between the modes that are less "deep"; hence, running an MCMC scheme to target this density will find it easier to move between modes. Parallel tempering uses this idea by running multiple chains at different β values while allowing exchange moves between chains. The chain with $\beta = 1$ admits dependent samples from the target of interest.

Further reading: Kirkpatrick, Gelatt and Vecchi (1983), "Optimization by simulated annealing", Science. Chapter 10 of Liu (2001), "Monte Carlo Strategies in Scientific Computing", New York: Springer.

6.4 MALA

Like the independence sampler and random walk Metropolis (RWM), the *Metropolis adjusted Langevin algorithm* (MALA) is a special case of the Metropolis-Hastings algortihm. It is similar to RWM, except that the proposal is not centered on the current value but is shifted uphill" from the current value. For a target $\pi(\boldsymbol{\theta})$, the proposal takes the form

$$\boldsymbol{\theta}^*|\boldsymbol{\theta} \sim N\left(\boldsymbol{\theta} + \frac{1}{2}\lambda^2 V \nabla \log \pi(\boldsymbol{\theta})\,,\,\lambda^2 V\right).$$

If we write $\ell(\boldsymbol{\theta}) = \log \pi(\boldsymbol{\theta})$ for simplicity, then the *gradient vector* is

$$\nabla \log \pi(\boldsymbol{\theta}) = \left(\frac{\partial \ell}{\partial \theta_1}, \ldots, \frac{\partial \ell}{\partial \theta_d}\right)'\Bigg|_{\boldsymbol{\theta}}.$$

The acceptance probability is the standard Metropolis-Hastings probability:

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = \min\left\{1\ \frac{\pi(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}.$$

Remarks:

- The algorithm will work for a Bayesian posterior known up to proportionality since, for a constant c,
$$\frac{\partial}{\partial\theta_j}\log(c\pi) = \frac{\partial}{\partial\theta_j}\log c + \frac{\partial}{\partial\theta_j}\log\pi = \frac{\partial}{\partial\theta_j}\log\pi.$$
- If V correctly represents the shape of the target then the inclusion of V in the offset allows the deterministic aspect of the proposal to take account of the shape, just as the random part does.
- MALA can work much better than RWM when the target is moderate to high dimensional, since MALA can tolerate a larger scaling λ than RWM, while still admitting a reasonable acceptance rate. The rule for choosing λ in MALA is to obtain an acceptance rate of around 0.574. As with RWM, in low to moderate dimensions, an acceptancerate a little higher than the theoretical optimum is usually better.
- The MALA proposal is derived from the transition kernel of a *Langevin* diffusion, whose equilibrium distribution is the target. The diffusion process itself is typically intractable, precluding direct simulation.

Further reading: Roberts and Rosenthal, (1998) "Optimal scaling of discrete approximations to Langevin diffusions." J. R. Statist. Soc. B.

6.5 Particle MCMC

Consider the hidden Markov model (HMM) formalism of Section 5.2. Consider data $\boldsymbol{y} = y_{0:T} = (y_0, \ldots, y_T)'$ and suppose interest lies in the marginal posterior density

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta})$$

where the *observed data likelihood* is

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \int p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x}$$

and $\boldsymbol{x} = (x_0, \ldots, x_T)'$. Although the observed data likelihood is typically intractable, we can run a particle filter to estimate it (see Section 5.6). It will be helpful to denote the estimator by $\widehat{p}_U(\boldsymbol{y}|\boldsymbol{\theta})$ where U denotes all random variables used to construct the estimator and $U \sim g(u)$ for some density g. Whilst beyond the scope of this course, it can be shown that $\widehat{p}_U(\boldsymbol{y}|\boldsymbol{\theta})$ is a non-negative, unbiased estimator. That is

$$\mathbb{E}_g[\widehat{p}_U(\boldsymbol{y}|\boldsymbol{\theta})] = \int \widehat{p}_u(\boldsymbol{y}|\boldsymbol{\theta})g(u)du = p(\boldsymbol{y}|\boldsymbol{\theta}).$$

This property is crucial in constructing a particle MCMC scheme to generate draws from $\pi(\boldsymbol{\theta}|\boldsymbol{y})$. Particle MCMC refers to a suite of algorithms that belong to a much more general class of algorithms known as *pseudo-marginal* MCMC. The pseudo-marginal "trick" is to run a standard MCMC scheme targeting a joint density over $\boldsymbol{\theta}$ and U, for which the target $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ is a marginal.

Hence, retaining $\boldsymbol{\theta}$ draws from the scheme will give (dependent) samples from the target. To this end, consider a joint density

$$\pi(\boldsymbol{\theta}, u) \propto \pi(\boldsymbol{\theta})\widehat{p}_u(\boldsymbol{y}|\boldsymbol{\theta})g(u)$$

for which integrating out u gives

$$\int \pi(\boldsymbol{\theta})\widehat{p}_u(\boldsymbol{y}|\boldsymbol{\theta})g(u)du = \pi(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta}) \propto \pi(\boldsymbol{\theta}|\boldsymbol{y}).$$

Hence, $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ is a marginal of $\pi(\boldsymbol{\theta}, u)$. Now, consider a Metropolis-Hastings scheme with target $\pi(\boldsymbol{\theta}, u)$ and proposal $q([\boldsymbol{\theta}^*, u^*] \,|\, [\boldsymbol{\theta}, u]) = q(\boldsymbol{\theta}^*|\boldsymbol{\theta})g(u^*)$. The acceptance probability is

$$\begin{aligned}
\alpha([\boldsymbol{\theta}^*, u^*] \,|\, [\boldsymbol{\theta}, u]) &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*, u^*)}{\pi(\boldsymbol{\theta}, u)} \times \frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)g(u)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})g(u^*)}\right\} \\
&= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\widehat{p}_{u^*}(\boldsymbol{y}|\boldsymbol{\theta}^*)g(u^*)}{\pi(\boldsymbol{\theta})\widehat{p}_u(\boldsymbol{y}|\boldsymbol{\theta})g(u)} \times \frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)g(u)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})g(u^*)}\right\} \\
&= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\widehat{p}_{u^*}(\boldsymbol{y}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})\widehat{p}_u(\boldsymbol{y}|\boldsymbol{\theta})} \times \frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}
\end{aligned}$$

which doesn't require that we evaluate $g(u)$ at all. In fact, we don't even need to store the auxiliary variables as part of the sampler. If we use a particle filter to obtain $\widehat{p}_{u^*}(\boldsymbol{y}|\boldsymbol{\theta}^*)$ then we have a particular particle MCMC scheme, known as the *particle marginal Metropolis-Hastings scheme*. Algorithmically, the scheme runs as follows:

1. Initialise with $\boldsymbol{\theta}^{(0)}$. Set $j = 1$.
2. Propose $\boldsymbol{\theta}^* \sim q(\cdot|\boldsymbol{\theta}^{(j-1)})$.
3. Run a particle filter (conditional on $\boldsymbol{\theta}^*$) with N particles to obtain $\widehat{p}_{u^*}(\boldsymbol{y}|\boldsymbol{\theta}^*)$.
4. Compute the acceptance probability

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\widehat{p}_{u^*}(\boldsymbol{y}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^{(j-1)})\widehat{p}_{u^{(j-1)}}(\boldsymbol{y}|\boldsymbol{\theta}^{(j-1)})} \times \frac{q(\boldsymbol{\theta}^{(j-1)}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})}\right\}$$

5. With probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(j-1)})$ set $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^*$ and $\widehat{p}_{u^{(j)}}(\boldsymbol{y}|\boldsymbol{\theta}^{(j)}) = \widehat{p}_{u^*}(\boldsymbol{y}|\boldsymbol{\theta}^*)$, otherwise set $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)}$ and $\widehat{p}_{u^{(j)}}(\boldsymbol{y}|\boldsymbol{\theta}^{(j)}) = \widehat{p}_{u^{(j-1)}}(\boldsymbol{y}|\boldsymbol{\theta}^{(j-1)})$.
6. Put $j := j + 1$ and go to step 2.

Remark: the number of particles N should be chosen to balance computational and mixing efficiency (beyond the scope of the course, but the interested student might like to see Sherlock, Thiery, Roberts and Rosenthal (2015), "On the effciency of pseudomarginal random walk Metropolis algorithms", the Annals of Statistics).

Further reading: Andrieu, Doucet and Holenstein (2010), "Particle Markov chain Monte Carlo methods (with discussion)", J. R. Statist. Soc. B.

6.6 Approximate Bayesian computation

In many statistical applications it is straightforward to simulate from the model but calculating the likelihood of the data is impracticable if not impossible. Suppose that $f(\boldsymbol{x}|\boldsymbol{\theta})$ is not known even up to a normalising constant. In this case the posterior is *completely intractable*.

Consider first the following algorithm.

1. Draw $\boldsymbol{\theta}^* \sim \pi(\cdot)$.
2. Accept $\boldsymbol{\theta}^*$ as a draw from $\pi(\boldsymbol{\theta}|\boldsymbol{x})$ with probability $f(\boldsymbol{x}|\boldsymbol{\theta}^*)$.

If $f(\boldsymbol{x}|\boldsymbol{\theta})$ cannot be evaluated, but simulation from the model is straightforward then an equivalent algorithm is:

1. Draw $\boldsymbol{\theta}^* \sim \pi(\cdot)$.
2. Draw $\boldsymbol{z} \sim f(\cdot|\boldsymbol{\theta}^*)$.
3. Accept $\boldsymbol{\theta}^*$ as a draw from $\pi(\boldsymbol{\theta}|\boldsymbol{x})$ if $\boldsymbol{z} = \boldsymbol{x}$.

It is straightforward to prove that this rejection algorithm samples from the posterior $\pi(\boldsymbol{\theta}|\boldsymbol{x})$.

For discrete $\boldsymbol{x}$, the above algorithm will have a non-zero acceptance rate. However, in most interesting problems the rejection rate will be intolerably high. In particular, the acceptance rate will typically be zero for continuous valued $\boldsymbol{x}$.

The ABC 'approximation' is to accept values provided that $\boldsymbol{z}$ is 'sufficiently close' to the data $\boldsymbol{x}$. Formally, we define some distance $\rho(\boldsymbol{z}, \boldsymbol{x})$ and retain a candidate value $\boldsymbol{\theta}^*$ if $\rho(\boldsymbol{z}, \boldsymbol{x}) \leq \epsilon$ for some small pre-defined ϵ. Hence, the ABC rejection algorithm runs as follows.

1. Draw $\boldsymbol{\theta}^* \sim \pi(\cdot)$.
2. Draw $\boldsymbol{z} \sim f(\cdot|\boldsymbol{\theta}^*)$.
3. Accept $\boldsymbol{\theta}^*$ as a draw from $\pi(\boldsymbol{\theta}|\boldsymbol{x})$ if $\rho(\boldsymbol{z}, \boldsymbol{x}) \leq \epsilon$.

A typical choice of $\rho(\boldsymbol{z}, \boldsymbol{x})$ is Euclidean distance

$$\rho(\boldsymbol{z}, \boldsymbol{x}) = ||\boldsymbol{z} - \boldsymbol{x}|| = \sqrt{(z_1 - x_1)^2 + \ldots + (z_n - x_n)^2}\,.$$

For suitable small choice of ϵ, the ABC rejection algorithm will closely approximate the true posterior. However, smaller choices of ϵ will lead to higher rejection rates. This will be a particular problem in the context of high-dimensional $\boldsymbol{x}$, where it is often unrealistic to expect a close match between all components of $\boldsymbol{x}$ and the simulated data $\boldsymbol{z}$, even for a good choice of $\boldsymbol{\theta}^*$. In this case, it makes more sense to look for good agreement between particular aspects of $\boldsymbol{x}$, such as the mean, or variance, or auto-correlation, depending on the exact problem and context.

In the simplest case, this is done by forming a (vector of) summary statistic(s), $\boldsymbol{s}(\boldsymbol{z})$, and accepting provided that $\rho(\boldsymbol{s}(\boldsymbol{z}), \boldsymbol{s}(\boldsymbol{x})) \leq \epsilon$ for some suitable choice of metric and ϵ. Ideally, the summary statistics should be *sufficient* for the parameter $\boldsymbol{\theta}$, although determining sufficient statistics requires a closed form expression of the likelihood, which we don't have, otherwise why would we want to implement ABC?!

There are various flavours of ABC such as ABC-MCMC and ABC-SMC, with the latter typically preferred.

Further reading: Kypraios, Neal and Prangle (2017): "A tutorial introduction to Bayesian inference for stochastic epidemic models using approximate Bayesian computation", Mathematical Biosciences.