Introducing Monte Carlo Methods with R Solutions to Odd-Numbered Exercises

Article · January 2010

Source: arXiv

Christian Robert
Université Paris-Dauphine
and
George Casella
University of Florida

Introducing Monte Carlo Methods with R

Solutions to Odd-Numbered Exercises

January 17, 2010

Preface

The scribes didn't have a large enough set from which to determine patterns.

Brandon Sauderson
The Hero of Ages

This partial solution manual to our book *Introducing Monte Carlo Methods with R*, published by Springer Verlag in the User R! series, on December 2009, has been compiled both from our own solutions and from homeworks written by the following Paris-Dauphine students in the 2009-2010 Master in Statistical Information Processing (TSI): Thomas Bredillet, Anne Sabourin, and Jiazi Tang. Whenever appropriate, the `R` code of those students has been identified by a `# (C.) Name` in the text. We are grateful to those students for allowing us to use their solutions. A few solutions in Chapter 4 are also taken *verbatim* from the solution manual to *Monte Carlo Statistical Methods* compiled by Roberto Casarin from the University of Brescia (and only available to instructors from Springer Verlag).

We also incorporated in this manual indications about some typos found in the first printing that came to our attention while composing this solution manual have been indicated as well. Following the new "print on demand" strategy of Springer Verlag, these typos will not be found in the versions of the book purchased in the coming months and should thus be ignored. (Christian Robert's book webpage at Université Paris-Dauphine `www.ceremade.dauphine.fr/~xian/books.html` is a better reference for the "complete" list of typos.)

Reproducing the warning Jean-Michel Marin and Christian P. Robert wrote at the start of the solution manual to *Bayesian Core*, let us stress here that some self-study readers of *Introducing Monte Carlo Methods with* R may come to the realisation that the solutions provided here are too sketchy for them because the way we wrote those solutions assumes some minimal familiarity with the maths, the probability theory and with the statistics be-

hind the arguments. There is unfortunately a limit to the time and to the efforts we can put in this solution manual and studying *Introducing Monte Carlo Methods with* R requires some prerequisites in maths (such as matrix algebra and Riemann integrals), in probability theory (such as the use of joint and conditional densities) and some bases of statistics (such as the notions of inference, sufficiency and confidence sets) that we cannot cover here. Casella and Berger (2001) is a good reference in case a reader is lost with the "basic" concepts or sketchy math derivations.

We obviously welcome solutions, comments and questions on possibly erroneous or ambiguous solutions, as well as suggestions for more elegant or more complete solutions: since this manual is distributed both freely and independently from the book, it can be updated and corrected [almost] in real time! Note however that the R codes given in the following pages are not optimised because we prefer to use simple and understandable codes, rather than condensed and efficient codes, both for time constraints and for pedagogical purposes: some codes were written by our students. Therefore, if you find better [meaning, more efficient/faster] codes than those provided along those pages, we would be glad to hear from you, but that does not mean that we will automatically substitute your R code for the current one, because readability is also an important factor.

A final request: this manual comes in two versions, one corresponding to the odd-numbered exercises and freely available to everyone, and another one corresponding to a larger collection of exercises and with restricted access to instructors only. Duplication and dissemination of the more extensive "instructors only" version are obviously prohibited since, if the solutions to most exercises become freely available, the appeal of using our book as a textbook will be severely reduced. Therefore, if you happen to possess an extended version of the manual, please refrain from distributing it and from reproducing it.

Sceaux and Gainesville **Christian P. Robert and George Casella**
January 17, 2010

Contents

1

Basic R programming

Exercise 1.1

Self-explanatory.

Exercise 1.3

Self-explanatory.

Exercise 1.5

One problem is the way in which `R` handles parentheses. So

```
> n=10
> 1:n
```

produces

```
1  2  3  4  5  6  7  8  9 10
```

but

```
> n=10
> 1:n-1
```

produces

```
0 1  2  3  4  5  6  7  8  9
```

since the `1:10` command is executed first, then 1 is subtracted.

The command `seq(1,n-1,by=1)` operates just as `1:(n-1)`. If n is less than 1 we can use something like `seq(1,.05,by=-.01)`. Try it, and try some other variations.

Exercise 1.7

a. To bootstrap the data you can use the code

```
Boot=2500
B=array(0,dim=c(nBoot, 1))
for (i in 1:nBoot){
    ystar=sample(y,replace=T)
    B[i]=mean(ystar)
    }
```

The quantile can be estimated with `sort(B)[.95*nBoot]`, which in our case/sample is 5.8478.

b. To get a confidence interval requires a double bootstrap. That is, for each bootstrap sample we can get a point estimate of the 95% quantile. We can then run an histogram on these quantiles with `hist`, and get *their* upper and lower quantiles for a confidence region.

```
nBoot1=1000
nBoot2=1000
B1=array(0,dim=c(nBoot1, 1))
B2=array(0,dim=c(nBoot2, 1))
for (i in 1:nBoot1){
   ystar=sample(y,replace=T)
   for (j in 1:nBoot2)
      B2[j]=mean(sample(ystar,replace=T))
   B1[i]=sort(B2)[.95*nBoot2]
   }
```

A 90% confidence interval is given by

```
> c(sort(B1)[.05*nBoot1], sort(B1)[.95*nBoot1])
[1] 4.731 6.844
```

or alternatively

```
> quantile(B1,c(.05,.95))
   5%    95%
4.731  6.844
```

for the data in the book. The command `hist(B1)` will give a histogram of the values.

Exercise 1.9

If you type

```
> mean
function (x, ...)
UseMethod("mean")
<environment: namespace:base>
```

you do not get any information about the function `mean` because it is not written in `R`, while

```
> sd
function (x, na.rm = FALSE)
{
    if (is.matrix(x))
        apply(x, 2, sd, na.rm = na.rm)
    else if (is.vector(x))
        sqrt(var(x, na.rm = na.rm))
    else if (is.data.frame(x))
        sapply(x, sd, na.rm = na.rm)
    else sqrt(var(as.vector(x), na.rm = na.rm))
}
```

shows `sd` is written in `R`. The same applies to `var` and `cov`.

Exercise 1.11

When looking at the description of `attach`, you can see that this command allows to use variables or functions that are in a database rather than in the current `.RData`. Those objects can be temporarily modified without altering their original format. (This is a fragile command that we do not personaly recommend!)

The function `assign` is also rather fragile, but it allows for the creation and assignment of an arbitrary number of objects, as in the documentation example:

```
for(i in 1:6) { #-- Create objects  'r.1', 'r.2', ... 'r.6' --
       nam <- paste("r",i, sep=".")
       assign(nam, 1:i)
      }
```

which allows to manipulate the `r.1`, `r.2`, ..., variables.

Exercise 1.13

This is mostly self-explanatory. If you type the help on each of those functions, you will see examples on how they work. The most recommended `R` function for saving `R` objects is `save`. Note that, when using `write`, the description states

```
The data (usually a matrix)  'x'  are written to file
'file'. If  'x'  is a two-dimensional matrix you need
to transpose it to get the columns in 'file' the same
as those in the internal representation.
```

Note also that `dump` and `sink` are fairly involved and should use with caution.

Exercise 1.15

Take, for example `a=3;x=c(1,2,3,4,5)` to see that they are the same, and, in fact, are the same as `max(which(x == a))`. For `y=c(3,4,5,6,7,8)`, try `match(x,y)` and `match(y,x)` to see the difference. In contrast, `x%in%y` and `y%in%y` return true/false tests.

Exercise 1.17

Running `system.time` on the three sets of commands give

a. 0.004 0.000 0.071
b. 0 0 0
c. 0.000 0.000 0.001

and the vectorial allocation is therefore the fastest.

Exercise 1.19

The `R` code is

```
> A=matrix(runif(4),ncol=2)
> A=A/apply(A,1,sum)
> apply(A%*%A,1,sum)
[1] 1 1
> B=A;for (t in 1:100) B=B%*%B
> apply(B,1,sum)
[1] Inf Inf
```

and it shows that numerical inaccuracies in the product leads to the property to fail when the power is high enough.

Exercise 1.21

The function `xyplot` is part of the `lattice` library. Then

```
> xyplot(age ~ circumference, data=Orange)
> barchart(age ~ circumference, data=Orange)
> bwplot(age ~ circumference, data=Orange)
> dotplot(age ~ circumference, data=Orange)
```

produce different representations of the dataset. Fitting a linear model is simply done by `lm(age ~ circumference, data=Orange)` and using the tree index as an extra covariate leads to

```
>summary(lm(age ~ circumference+Tree, data=Orange))

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    -90.0596    55.5795  -1.620    0.116
circumference    8.7366     0.4354  20.066  < 2e-16 ***
Tree.L        -348.8982    54.9975  -6.344 6.23e-07 ***
Tree.Q         -22.0154    52.1881  -0.422    0.676
Tree.C          72.2267    52.3006   1.381    0.178
Tree^4          41.0233    52.2167   0.786    0.438
```

meaning that only `Tree.L` was significant.

Exercise 1.23

a. A plain representation is

```
> s
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    0    0    0    0    0    6    0    4    0
 [2,]    2    7    9    0    0    0    0    5    0
 [3,]    0    5    0    8    0    0    0    0    2
 [4,]    0    0    2    6    0    0    0    0    0
 [5,]    0    0    0    0    0    0    0    0    0
 [6,]    0    0    1    0    9    0    6    7    3
 [7,]    8    0    5    2    0    0    4    0    0
 [8,]    3    0    0    0    0    0    0    8    5
 [9,]    6    0    0    0    0    0    9    0    1
```

   where empty slots are represented by zeros.

b. A simple cleaning of non-empty (i.e. certain) slots is

```
for (i in 1:9)
for (j in 1:9){
  if (s[i,j]>0) pool[i,j,-s[i,j]]=FALSE
  }
```

c. In `R`, matrices (and arrays) are also considered as vectors. Hence `s[i]` represents the $(1 + \lfloor (i-1)/9 \rfloor, (i-1) \bmod 9 + 1)$ entry of the grid.

d. This is self-explanatory. For instance,

```
> a=2;b=5
> boxa
[1] 1 2 3
> boxb
[1] 4 5 6
```

e. The first loop checks whether or not, for each remaining possible integer, there exists an identical entry in the same row, in the same column or in the same box. The second command sets entries for which only one possible integer remains to this integer.

f. A plain `R` program solving the grid is

```
while (sum(s==0)>0){
  for (i in sample(1:81)){
    if (s[i]==0){
       a=((i-1)%%9)+1
       b=trunc((i-1)/9)+1
       boxa=3*trunc((a-1)/3)+1
       boxa=boxa:(boxa+2)
       boxb=3*trunc((b-1)/3)+1
       boxb=boxb:(boxb+2)

       for (u in (1:9)[pool[a,b,]]){
         pool[a,b,u]=(sum(u==s[a,])+sum(u==s[,b])
                +sum(u==s[boxa,boxb]))==0
         }

       if (sum(pool[a,b,])==1){
         s[i]=(1:9)[pool[a,b,]]
         }

       if (sum(pool[a,b,])==0){
          print("wrong sudoku")
          break()
          }
       }
    }
  }
```

and it stops with the outcome

```
> s
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    1    3    8    5    2    6    7    4    9
 [2,]    2    7    9    3    4    1    8    5    6
 [3,]    4    5    6    8    7    9    3    1    2
 [4,]    7    4    2    6    3    5    1    9    8
 [5,]    9    6    3    1    8    7    5    2    4
 [6,]    5    8    1    4    9    2    6    7    3
 [7,]    8    9    5    2    1    3    4    6    7
 [8,]    3    1    7    9    6    4    2    8    5
 [9,]    6    2    4    7    5    8    9    3    1
```

which is the solved Sudoku.

2

Random Variable Generation

Exercise 2.1

For a random variable X with cdf F, if

$$F^-(u) = \inf\{x, F(x) \le u\},$$

then, for $U \sim \mathcal{U}[0,1]$, for all $y \in \mathbb{R}$,

$$\begin{aligned}\mathbb{P}(F^-(U) \le y) &= \mathbb{P}(\inf\{x, F(x) \le U\} \le y)\\ &= \mathbb{P}(F(y) \ge U) \qquad \text{as } F \text{ is non-decreasing}\\ &= F(y) \qquad\qquad \text{as } U \text{ is uniform}\end{aligned}$$

Exercise 2.3

a. It is easy to see that $\mathbb{E}[U_1] = 0$, and a standard calculation shows that $\text{var}(U_1) = 1/12$, from which the result follows.
b. Histograms show that the tails of the 12 uniforms are not long enough. Consider the code

```
nsim=10000
u1=runif(nsim)
u2=runif(nsim)
X1=sqrt(-2*log(u1))*cos(2*pi*u2)
X2=sqrt(-2*log(u1))*sin(2*pi*u2)
U=array(0,dim=c(nsim,1))
for(i in 1:nsim)U[i]=sum(runif(12,-.5,.5))
par(mfrow=c(1,2))
hist(X1)
hist(U)
a=3
mean(X1>a)
mean(U>a)
```

```
mean(rnorm(nsim)>a)
1-pnorm(a)
```

c. You should see the difference in the tails of the histogram. Also, the numerical output from the above is

```
[1] 0.0016
[1] 5e-04
[1] 0.0013
[1] 0.001349898
```

where we see that the Box-Muller and `rnorm` are very good when compared with the exact `pnorm`. Try this calculation for a range of `nsim` and `a`.

Exercise 2.5

For $U \sim \mathcal{U}_{[0,1]}$, $Y \sim g(y)$, and $X \sim f(x)$, such that $f/g \leq M$, the acceptance condition in the Accept–Reject algorithm is that $U \leq f(Y)/(Mg(Y))$. The probability of acceptance is thus

$$\begin{aligned}
\mathbb{P}(U \leq f(Y)/Mg(Y)) &= \int_{-\infty}^{+\infty} \int_0^{\frac{f(y}{Mg(y)}} \mathrm{d}u g(y)\,\mathrm{d}y \\
&= \int_{-\infty}^{+\infty} \frac{f(y)}{Mg(y)} g(y)\,\mathrm{d}y \\
&= \frac{1}{M} \int_{-\infty}^{+\infty} f(y)\,\mathrm{d}y \\
&= \frac{1}{M}\,.
\end{aligned}$$

Assume f/g is only known up to a normalising constant, i.e. $f/g = k.\tilde{f}/\tilde{g}$, with $\tilde{f}/\tilde{g} \leq \tilde{M}$, $\tilde{M}$ being a well-defined upper bound different from M because of the missing normalising constants. Since $Y \sim g$,

$$\begin{aligned}
\mathbb{P}(U \leq \tilde{f}(Y)/\tilde{M}\tilde{g}(Y)) &= \int_{-\infty}^{+\infty} \int_0^{\frac{\tilde{f}(y}{\tilde{M}\tilde{g}(y)}} \mathrm{d}u g(y)\,\mathrm{d}y \\
&= \int_{-\infty}^{+\infty} \frac{\tilde{f}(y)}{\tilde{M}\tilde{g}(y)} g(y)\,\mathrm{d}y \\
&= \int_{-\infty}^{+\infty} \frac{f(y)}{k\tilde{M}g(y)} g(y)\,\mathrm{d}y \\
&= \frac{1}{k\tilde{M}}\,.
\end{aligned}$$

Therefore the missing constant is given by

$$k = 1 \Big/ \tilde{M}.\mathbb{P}(U \leq \tilde{f}(Y)/\tilde{M}\tilde{g}(Y))\,,$$

which can be estimated from the empirical acceptance rate.

Exercise 2.7

The ratio is equal to

$$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\,\frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}\,x^{\alpha-a}\,(1-x)^{\beta-b}$$

and it will not diverge at $x=0$ only if $a \le \alpha$ and at $x=1$ only if $b \le \beta$. The maximum is attained for

$$\frac{\alpha-a}{x^\star} = \frac{\beta-b}{1-x^\star}\,,$$

i.e. is

$$M_{a,b} = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\,\frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}\,\frac{(\alpha-a)^{\alpha-a}(\beta-b)^{\beta-b}}{(\alpha-a+\beta-b)^{\alpha-a+\beta-b}}\,.$$

The analytic study of this quantity as a function of (a,b) is quite delicate but if we define

```
mab=function(a,b){
  lgamma(a)+lgamma(b)+(alph-a)*log(alph-a)+(beta-b)*log(beta-b)
  -(alph+bet-a-b)*log(alph+bet-a-b)}
```

it is easy to see using `contour` on a sequence of a's and b's that the maximum of $M_{a,b}$ is achieved over integer values when $a = \lfloor \alpha \rfloor$ and $b = \lfloor \beta \rfloor$.

Exercise 2.9

Given θ, exiting the loop is driven by $X = x_0$, which indeed has a probability $f(x_0|\theta)$ to occur. If X is a discrete random variable, this is truly a probability, while, if X is a continuous random variable, this is zero. The distribution of the exiting θ is then dependent on the event $X = x_0$ taking place, i.e. is proportional to $\pi(\theta)f(x_0|\theta)$, which is exactly $\pi(\theta|x_0)$.

Exercise 2.11

a. Try the `R` code

```
nsim<-5000
n=25;p=.2;
cp=pbinom(c(0:n),n,p)
X=array(0,c(nsim,1))
for(i in 1:nsim){
   u=runif(1)
   X[i]=sum(cp<u)
   }
hist(X,freq=F)
lines(1:n,dbinom(1:n,n,p),lwd=2)
```

which produces a histogram and a mass function for the binomial $\mathcal{B}(25, .2)$. To check timing, create the function

```
MYbinom<-function(s0,n0,p0){
  cp=pbinom(c(0:n0),n0,p0)
  X=array(0,c(s0,1))
  for (i in 1:s0){
        u=runif(1)
        X[i]=sum(cp<u)
        }
  return(X)
  }
```

and use `system.time(rbinom(5000,25,.2))` and `system.time(MYbinom(5000,25,.2))` to see how much faster `R` is.

b. Create the `R` functions `Wait` and `Trans`:

```
Wait<-function(s0,alpha){
  U=array(0,c(s0,1))
  for (i in 1:s0){
     u=runif(1)
     while (u > alpha) u=runif(1)
     U[i]=u
     }
  return(U)
  }

Trans<-function(s0,alpha){
  U=array(0,c(s0,1))
  for (i in 1:s0) U[i]=alpha*runif(1)
  return(U)
  }
```

Use `hist(Wait(1000,.5))` and `hist(Trans(1000,.5))` to see the corresponding histograms. Vary n and α. Use the `system.time` command as in part a to see the timing. In particular, `Wait` is very bad if α is small.

Exercise 2.13

The cdf of the Pareto $\mathcal{P}(\alpha)$ distribution is

$$F(x) = 1 - x^{-\alpha}$$

over $(1, \infty)$. Therefore, $F^{-1}(U) = (1 - U)^{-1/\alpha}$, which is also the $-1/\alpha$ power of a uniform variate.

Exercise 2.15

Define the R functions

```
Pois1<-function(s0,lam0){
   spread=3*sqrt(lam0)
   t=round(seq(max(0,lam0-spread),lam0+spread,1))
   prob=ppois(t,lam0)
   X=rep(0,s0)
   for (i in 1:s0){
      u=runif(1)
      X[i]=max(t[1],0)+sum(prob<u)-1
      }
   return(X)
   }
```

and

```
Pois2<-function(s0,lam0){
   X=rep(0,s0)
   for (i in 1:s0){
   sum=0;k=1
   sum=sum+rexp(1,lam0)
   while (sum<1){ sum=sum+rexp(1,lam0);k=k+1}
   X[i]=k
   }
   return(X)
   }
```

and then run the commands

```
> nsim=100
> lambda=3.4
> system.time(Pois1(nsim,lambda))
   user  system elapsed
  0.004   0.000   0.005
> system.time(Pois2(nsim,lambda))
   user  system elapsed
  0.004   0.000   0.004
> system.time(rpois(nsim,lambda))
   user  system elapsed
      0       0       0
```

for other values of `nsim` and `lambda`. You will see that `rpois` is by far the best, with the exponential generator (`Pois2`) not being very good for large λ's. Note also that `Pois1` is not appropriate for small λ's since it could then return negative values.

Exercise 2.17

a. Since, if $X \sim \mathcal{G}a(\alpha, \beta)$, then $\beta X = \sum_{j=1}^{\alpha} \beta X_j \sim \mathcal{G}a(\alpha, 1)$, β is the inverse of a scale parameter.

b. The Accept-Reject ratio is given by

$$\frac{f(x)}{g(x)} \propto \frac{x^{n-1}\, e^{-x}}{\lambda\, e^{-\lambda x}} = \lambda^{-1} x^{n-1} e^{-(1-\lambda)x}\,.$$

The maximum of this ratio is obtained for

$$\frac{n-1}{x^\star} - (1-\lambda) = 0\,, \quad \text{i.e. for} \quad x^\star = \frac{n-1}{1-\lambda}\,.$$

Therefore,

$$M \propto \lambda^{-1} \left(\frac{n-1}{1-\lambda}\right)^{n-1} e^{-(n-1)}$$

and this upper bound is minimised in λ when $\lambda = 1/n$.

c. If g is the density of the $\mathcal{G}a(a, b)$ distribution and f the density of the $\mathcal{G}a(\alpha, 1)$ distribution,

$$g(x) = \frac{x^{a-1} e^{-bx} b^a}{\Gamma(a)} \quad \text{and} \quad f(x) = \frac{x^{\alpha-1} e^{-x}}{\Gamma(\alpha)}$$

the Accept-Reject ratio is given by

$$\frac{f(x)}{g(x)} = \frac{x^{\alpha-1} e^{-x} \Gamma(a)}{\Gamma(\alpha) b^a x^{a-1} e^{-bx}} \propto b^{-a} x^{\alpha-a} e^{-x(1-b)}\,.$$

Therefore,

$$\frac{\partial}{\partial x}\frac{f}{g} = b^a e^{-x(1-b)} x^{\alpha-a-1} \left\{(\alpha - a) - (1-b)x\right\}$$

provides $x^\star = \alpha - a/1 - b$ as the argument of the maximum of the ratio, since $\frac{f}{g}(0) = 0$. The upper bound M is thus given by

$$M(a, b) = b^{-a} \left(\frac{\alpha - a}{1-b}\right)^{\alpha-a} e^{-\left(\frac{\alpha-a}{1-b}\right)*(1-b)} = b^{-a} \left(\frac{\alpha - a}{(1-b)e}\right)^{\alpha-a}\,.$$

It obviously requires $b < 1$ and $a < \alpha$.

d. **Warning: there is a typo in the text of the first printing, it should be:**

Show that the maximum of $b^{-a}(1-b)^{a-\alpha}$ is attained at $b = a/\alpha$, and hence the optimal choice of b for simulating $\mathcal{G}a(\alpha, 1)$ is $b = a/\alpha$, which gives the same mean for both $\mathcal{G}a(\alpha, 1)$ and $\mathcal{G}a(a, b)$.

With this modification, the maximum of $M(a,b)$ in b is obtained by derivation, i.e. for b solution of

$$\frac{a}{b} - \frac{\alpha - a}{1-b} = 0\,,$$

which leads to $b = a/\alpha$ as the optimal choice of b. Both $\mathcal{G}a(\alpha,1)$ and $\mathcal{G}a(a,a/\alpha)$ have the same mean α.

e. Since

$$M(a,a/\alpha) = (a/\alpha)^{-a}\left(\frac{\alpha-a}{(1-a/\alpha)e}\right)^{\alpha-a} = (a/\alpha)^{-a}\alpha^{\alpha-a} = \alpha^{\alpha}/a^{a},,$$

M is decreasing in a and the largest possible value is indeed $a = \lfloor\alpha\rfloor$.

Exercise 2.19

The ratio f/g is

$$\frac{f(x)}{g(x)} = \frac{\exp\{-x^2/2\}/\sqrt{2\pi}}{\alpha\exp\{-\alpha|x|\}/2} = \frac{\sqrt{2/\pi}}{\alpha}\,\exp\{\alpha|x| - x^2/2\}$$

and it is maximal when $x = \pm\alpha$, so $M = \sqrt{2/\pi}\exp\{\alpha^2/2\}/\alpha$. Taking the derivative in α leads to the equation

$$\alpha - \frac{1}{\alpha^2} = 0\,,$$

that is, indeed, to $\alpha = 1$.

Exercise 2.21

Warning: There is a typo in this exercise, it should be:

(i). a mixture representation (2.2), where $g(x|y)$ is the density of χ^2_{p+2y} and $p(y)$ is the density of $\mathcal{P}(\lambda/2)$, and
(ii). the sum of a χ^2_{p-1} random variable and the square of a $\mathcal{N}(\sqrt{\lambda},1)$.

a. Show that both those representations hold.
b. Compare the corresponding algorithms that can be derived from these representations among themselves and also with `rchisq` for small and large values of λ.

If we use the definition of the noncentral chi squared distribution, $\chi^2_p(\lambda)$ as corresponding to the distribution of the squared norm $||x||^2$ of a normal vector $x \sim \mathcal{N}_p(\theta, I_p)$ when $\lambda = ||\theta||^2$, this distribution is invariant by rotation over the normal vector and it is therefore the same as when $x \sim \mathcal{N}_p((0,\ldots,0,\sqrt{\lambda}), I_p)$,

hence leading to the representation (ii), i.e. as a sum of a χ^2_{p-1} random variable and of the square of a $\mathcal{N}(||\theta||, 1)$ variable. Representation (i) holds by a regular mathematical argument based on the series expansion of the modified Bessel function since the density of a non-central chi-squared distribution is

$$f(x|\lambda) = \frac{1}{2}(x/\lambda)^{(p-2)/4} I_{(p-2)/2}(\sqrt{\lambda x}) e^{-(\lambda+x)/2},$$

where

$$I_\nu(t) = \left(\frac{t}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{(z/2)^{2k}}{k!\Gamma(\nu+k+1)}.$$

Since `rchisq` includes an optional non-centrality parameter `nc`, it can be used to simulate directly a noncentral chi-squared distribution. The two scenarios (i) and (ii) lead to the following `R` codes.

```
> system.time({x=rchisq(10^6,df=5,ncp=3)})
   user  system elapsed
> system.time({x=rchisq(10^6,df=4)+rnorm(10^6,mean=sqrt(3))^2})
   user  system elapsed
  1.700   0.056   1.757
> system.time({x=rchisq(10^6,df=5+2*rpois(10^6,3/2))})
   user  system elapsed
  1.168   0.048   1.221
```

Repeated experiments with other values of p and λ lead to the same conclusion that the Poisson mixture representation is the fastest.

Exercise 2.23

Since the ratio $\pi(\theta|\mathbf{x})/\pi(\theta)$ is the likelihood, it is obvious that the optimal bound M is the likelihood function evaluated at the MLE (assuming π is a true density and not an improper prior).

Simulating from the posterior can then be done via

```
theta0=3;n=100;N=10^4
x=rnorm(n)+theta0
lik=function(the){prod(dnorm(x,mean=the))}
M=optimise(f=function(the){prod(dnorm(x,mean=the))},
  int=range(x),max=T)$obj
theta=rcauchy(N)
res=(M*runif(N)>apply(as.matrix(theta),1,lik));print(sum(res)/N)
while (sum(res)>0){le=sum(res);theta[res]=rcauchy(le)
res[res]=(M*runif(le)>apply(as.matrix(theta[res]),1,lik))}
```

The rejection rate is given by 0.9785, which means that the Cauchy proposal is quite inefficient. An empirical confidence (or credible) interval at the level 95% on θ is $(2.73, 3.799)$. Repeating the experiment with $n = 100$ leads (after a while) to the interval $(2.994, 3.321)$, there is therefore an improvement.

3

Monte Carlo Integration

Exercise 3.1

a. The plot of the integrands follows from a simple `R` program:

```
f1=function(t){  t/(1+t*t)*exp(-(x-t)^2/2)}
f2=function(t){  1/(1+t*t)*exp(-(x-t)^2/2)}
plot(f1,-3,3,col=1,ylim=c(-0.5,1),xlab="t",ylab="",ty="l")
plot(f2,-3,3,add=TRUE,col=2,ty="l")
legend("topright", c("f1=t.f2","f2"), lty=1,col=1 :2)
```

Both numerator and denominator are expectations under the Cauchy distribution. They can therefore be approximated directly by

```
Niter=10^4
co=rcauchy(Niter)
I=mean(co*dnorm(co,mean=x))/mean(dnorm(co,mean=x))
```

We thus get

```
> x=0
> mean(co*dnorm(co,mean=x))/mean(dnorm(co,mean=x))
[1] 0.01724
> x=2
> mean(co*dnorm(co,mean=x))/mean(dnorm(co,mean=x))
[1] 1.295652
> x=4
> mean(co*dnorm(co,mean=x))/mean(dnorm(co,mean=x))
[1] 3.107256
```

b. Plotting the convergence of those integrands can be done via

```
# (C.) Anne Sabourin, 2009
x1=dnorm(co,mean=x)
estint2=cumsum(x1)/(1:Niter)
esterr2=sqrt(cumsum((x1-estint2)^2))/(1:Niter)
```

```
x1=co*x1
estint1=cumsum(x1))/(1:Niter)
esterr2=sqrt(cumsum((x1-estint1)^2))/(1:Niter)
par(mfrow=c(1,2))
plot(estint1,type="l",xlab="iteration",ylab="",col="gold")
lines(estint1-2*esterr1,lty=2,lwd=2)
lines(estint1+2*esterr1,lty=2,lwd=2)
plot(estint2,type="l",xlab="iteration",ylab="",col="gold")
lines(estint2-2*esterr2,lty=2,lwd=2)
lines(estint2+2*esterr2,lty=2,lwd=2)
```

Because we have not yet discussed the evaluation of the error for a ratio of estimators, we consider both terms of the ratio separately. The empirical variances $\hat{\sigma}$ are given by `var(co*dnorm(co,m=x))` and `var(dnorm(co,m=x))` and solving $2\hat{\sigma}/\sqrt{n} < 10^{-3}$ leads to an evaluation of the number of simulations necessary to get 3 digits of accuracy.

```
> x=0;max(4*var(dnorm(co,m=x))*10^6,
+ 4*var(co*dnorm(co,m=x))*10^6)
[1] 97182.02
> x=2; 4*10^6*max(var(dnorm(co,m=x)),var(co*dnorm(co,m=x)))
[1] 220778.1
> x=4; 10^6*4*max(var(dnorm(co,m=x)),var(co*dnorm(co,m=x)))
[1] 306877.9
```

c. A similar implementation applies for the normal simulation, replacing `dnorm` with `dcauchy` in the above. The comparison is clear in that the required number of normal simulations when $x = 4$ is 1398.22, to compare with the above 306878.

Exercise 3.3

Due to the identity

$$\mathbb{P}(X > 20) = \int_{20}^{\infty} \frac{\exp(-\frac{x^2}{2})}{\sqrt{2\pi}} \mathrm{d}x = \int_{0}^{1/20} \frac{\exp(-\frac{1}{2*u^2})}{20u^2\sqrt{2\pi}} 20 \mathrm{d}u\,,$$

we can see this integral as an expectation under the $\mathcal{U}(0, 1/20)$ distribution and thus use a Monte Carlo approximation to $\mathbb{P}(X > 20)$. The following `R` code monitors the convergence of the corresponding approximation.

```
# (C.) Thomas Bredillet, 2009
h=function(x){ 1/(x^2*sqrt(2*pi)*exp(1/(2*x^2)))}
par(mfrow=c(2,1))
curve(h,from=0,to=1/20,xlab="x",ylab="h(x)",lwd="2")
I=1/20*h(runif(10^4)/20)
estint=cumsum(I)/(1:10^4)
esterr=sqrt(cumsum((I-estint)^2))/(1:10^4)
```

```
plot(estint,xlab="Iterations",ty="l",lwd=2,
ylim=mean(I)+20*c(-esterr[10^4],esterr[10^4]),ylab="")
lines(estint+2*esterr,col="gold",lwd=2)
lines(estint-2*esterr,col="gold",lwd=2)
```

The estimated probability is $2.505e{-}89$ with an error of $\pm 3.61e{-}90$, compared with

```
> integrate(h,0,1/20)
2.759158e-89 with absolute error < 5.4e-89
> pnorm(-20)
[1] 2.753624e-89
```

Exercise 3.5

Warning: due to the (late) inclusion of an extra-exercise in the book, the "above exercise" actually means Exercise 3.3!!!

When $Z \sim \mathcal{N}(0,1)$, with density f, the quantity of interest is $\mathbb{P}(Z > 4.5)$, i.e. $\mathbb{E}^f[\mathbb{I}_{Z>4.5}]$. When g is the density of the exponential $\mathcal{E}xp(\lambda)$ distribution truncated at 4.5,

$$g(y) = \frac{1_{y>4.5}\lambda \exp(-\lambda y)}{\int_{-4.5}^{\infty} \lambda \exp(-\lambda y)\,\mathrm{d}y} = \lambda e^{-\lambda(y-4.5)}\mathbb{I}_{y>4.5}\,,$$

simulating iid $Y^{(i)}$'s from g is straightforward. Given that the indicator function $\mathbb{I}_{Y>4.5}$ is then always equal to 1, $\mathbb{P}(Z > 4.5)$ is estimated by

$$\hat{h}_n = \frac{1}{n}\sum_{i=1}^{n} \frac{f(Y^{(i)})}{g(Y^{(i)})}.$$

A corresponding estimator of its variance is

$$v_n = \frac{1}{n}\sum_{i=1}^{n}(1-\hat{h}_n)^2 f(Y^{(i)})/g(Y^{(i)})\,.$$

The following `R` code monitors the convergence of the estimator (with $\lambda = 1, 10$)

```
# (C.) Anne Sabourin, 2009
Nsim=5*10^4
x=rexp(Nsim)
par(mfcol=c(1,3))
for (la in c(.5,5,50)){
  y=(x/la)+4.5
  weit=dnorm(y)/dexp(y-4.5,la)
  est=cumsum(weit)/(1:Nsim)
```

```
varest=cumsum((1-est)^2*weit/(1:Nsim)^2)
plot(est,type="l",ylim=c(3e-6,4e-6),main="P(X>4.5) estimate",
sub=paste("based on E(",la,") simulations",sep=""),xlab="",ylab="")
abline(a=pnorm(-4.5),b=0,col="red")
}
```

When evaluating the impact of λ on the variance (and hence on the convergence) of the estimator, similar graphs can be plotted for different values of λ. This experiment does not exhibit a clear pattern, even though large values of λ, like $\lambda = 20$ appear to slow down convergence very much. Figure 3.1 shows the output of such a comparison. Picking $\lambda = 5$ seems however to produce a very stable approximation of the tail probability.

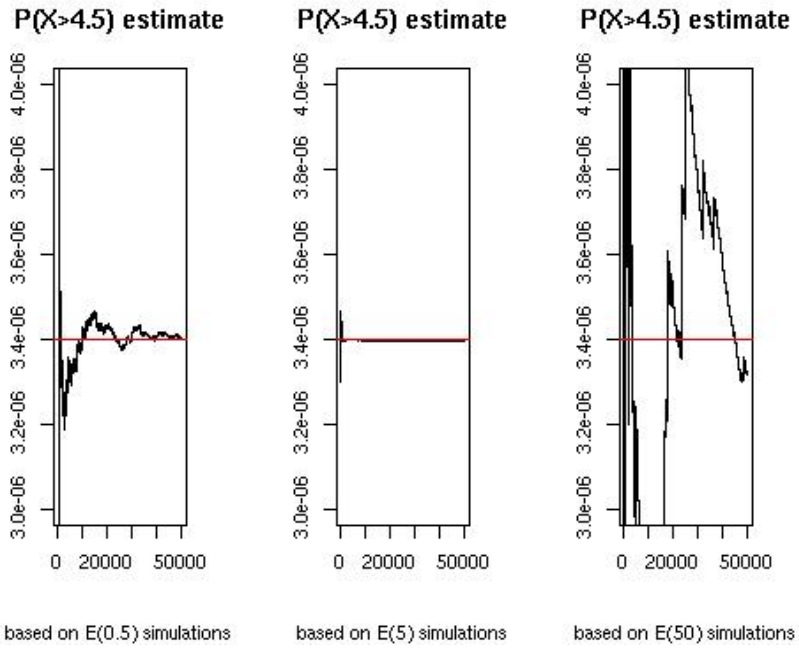

Fig. 3.1. Comparison of three importance sampling approximations to the normal tail probability $\mathbb{P}(Z > 4.5)$ based on a truncated $\mathcal{E}xp(\lambda)$ distribution with $\lambda = .5, 5.50$. The straight red line is the true value.

Exercise 3.7

While the expectation of $\sqrt{x/(1-x)}$ is well defined for $\nu > 1/2$, the integral of $x/(1-x)$ against the t density does not exist for any ν. Using an importance

sampling representation,

$$\int \frac{x}{1-x}\,\frac{f^2(x)}{g(x)}\,\mathrm{d}x = \infty$$

if $g(1)$ is finite. The integral will be finite around 1 when $1/(1-t)g(t)$ is integrable, which means that $g(t)$ can go to infinity at any rate. For instance, if $g(t) \approx (1-t)^{-\alpha}$ around 1, any $\alpha > 0$ is acceptable.

Exercise 3.9

As in Exercise 3.1, the quantity of interest is $\delta^\pi(x) = \mathbb{E}^\pi(\theta|x) = \int \theta\pi(\theta|x)\,\mathrm{d}\theta$ where $x \sim \mathcal{N}(\theta, 1)$ and $\theta \sim \mathcal{C}(0, 1)$. The target distribution is

$$\pi(\theta|x) \propto \pi(\theta)e^{-(x-\theta)^2/2} = f_x(\theta)\,.$$

A possible importance function is the prior distribution,

$$g(\theta) = \frac{1}{\pi(1+\theta^2)}$$

and for every $\theta \in \mathbb{R}$, $\frac{f_x(\theta)}{g(\theta)} \leq M$, when $M = \pi$. Therefore, generating from the prior g and accepting simulations according to the Accept-Reject ratio provides a sample from $\pi(\theta|x)$. The empirical mean of this sample is then a converging estimator of $\mathbb{E}^\pi(\theta|x)$. Furthermore, we directly deduce the estimation error for δ. A graphical evaluation of the convergence is given by the following `R` program:

```
f=function(t){ exp(-(t-3)^2/2)/(1+t^2)}
M=pi
Nsim=2500
postdist=rep(0,Nsim)
for (i in 1:Nsim){
   u=runif(1)*M
   postdist[i]=rcauchy(1)
   while(u>f(postdist[i])/dcauchy(postdist[i])){
     u=runif(1)*M
     postdist[i]=rcauchy(1)
     }}
estdelta=cumsum(postdist)/(1:Nsim)
esterrd=sqrt(cumsum((postdist-estdelta)^2))/(1:Nsim)
par(mfrow=c(1,2))
C1=matrix(c(estdelta,estdelta+2*esterrd,estdelta-2*esterrd),ncol=3)
matplot(C1,ylim=c(1.5,3),type="l",xlab="Iterations",ylab="")
plot(esterrd,type="l",xlab="Iterations",ylab="")
```

Exercise 3.11

a. If $X \sim \mathcal{E}xp(1)$ then for $x \geq a$,

$$\mathbb{P}[a + X < x] = \int_0^{x-a} \exp(-t)\,\mathrm{d}t = \int_a^x \exp(-t+a)\,\mathrm{d}t = \mathbb{P}(Y < x)$$

when $Y \sim \mathcal{E}xp^+(a, 1)$,

b. If $X \sim \chi^2_3$, then

$$\begin{aligned}\mathbb{P}(X > 25) &= \int_{25}^{+\infty} \frac{2^{-3/2}}{\Gamma(\frac{3}{2})}\, x^{1/2} \exp(-x/2)\,\mathrm{d}x \\ &= \int_{12.5}^{+\infty} \frac{\sqrt{(}x)\exp(-12.5)}{\Gamma(\frac{3}{2})} \exp(-x+12.5)\,\mathrm{d}x\,.\end{aligned}$$

The corresponding `R` code

```
# (C.) Thomas Bredilllet, 2009
h=function(x){ exp(-x)*sqrt(x)/gamma(3/2)}
X = rexp(10^4,1) + 12.5
I=exp(-12.5)*sqrt(X)/gamma(3/2)
estint=cumsum(I)/(1:10^4)
esterr=sqrt(cumsum((I-estint)^2))/(1:10^4)
plot(estint,xlab="Iterations",ty="l",lwd=2,
ylim=mean(I)+20*c(-esterr[10^4],esterr[10^4]),ylab="")
lines(estint+2*esterr,col="gold",lwd=2)
lines(estint-2*esterr,col="gold",lwd=2)
```

gives an evaluation of the probability as $1.543e-05$ with a 10^{-8} error, to compare with

```
> integrate(h,12.5,Inf)
1.544033e-05 with absolute error < 3.4e-06
> pchisq(25,3,low=F)
[1] 1.544050e-05
```

Similarly, when $X \sim t_5$, then

$$\mathbb{P}(X > 50) = \int_{50}^{\infty} \frac{\Gamma(3)}{\sqrt{(5*\pi)}\Gamma(2,5)(1+\frac{t^2}{5})^3 \exp(-t+50)} \exp(-t+50)\,\mathrm{d}t$$

and a corresponding `R` code

```
# (C.) Thomas Bredilllet, 2009
h=function(x){ 1/sqrt(5*pi)*gamma(3)/gamma(2.5)*1/(1+x^2/5)^3}
integrate(h,50,Inf)
X = rexp(10^4,1) + 50
I=1/sqrt(5*pi)*gamma(3)/gamma(2.5)*1/(1+X^2/5)^3*1/exp(-X+50)
```

```
estint=cumsum(I)/(1:10^4)
esterr=sqrt(cumsum((I-estint)^2))/(1:10^4)
plot(estint,xlab="Mean and error range",type="l",lwd=2,
ylim=mean(I)+20*c(-esterr[10^4],esterr[10^4]),ylab="")
lines(estint+2*esterr,col="gold",lwd=2)
lines(estint-2*esterr,col="gold",lwd=2)
```

As seen on the graph, this method induces jumps in the convergence patterns. Those jumps are indicative of variance problems, as should be since the estimator does not have a finite variance in this case. The value returned by this approach differs from alternatives evaluations:

```
>  mean(I)
[1] 1.529655e-08
> sd(I)/10^2
[1] 9.328338e-10
> integrate(h,50,Inf)
3.023564e-08 with absolute error < 2e-08
> pt(50,5,low=F)
[1] 3.023879e-08
```

and cannot be trusted.

c. **Warning: There is a missing line in the text of this question, which should read:**

Explore the gain in efficiency from this method. Take $a = 4.5$ in part (a) and run an experiment to determine how many normal $\mathcal{N}(0, 1)$ random variables would be needed to calculate $P(Z > 4.5)$ to the same accuracy obtained from using 100 random variables in this importance sampler.

If we use the representation

$$\mathbb{P}(Z > 4.5) = \int_{4.5}^{\infty} \varphi(z)\,\mathrm{d}z = \int_{0}^{\infty} \varphi(x + 4.5)\exp(x)\exp(-x)\,\mathrm{d}x\,,$$

the approximation based on 100 realisations from an $\mathcal{E}xp(1)$ distribution, $x_1m \ldots, x_100$, is

$$\frac{1}{100}\sum_{i=1}^{100} \varphi(x_i + 4.5)\exp(x_i)$$

and the `R` code

```
> x=rexp(100)
> mean(dnorm(x+4.5)*exp(x))
[1] 2.817864e-06
> var(dnorm(x+4.5)*exp(x))/100
[1] 1.544983e-13
```

shows that the variance of the resulting estimator is about 10^{-13}. A simple simulation of a normal sample of size m and the resulting accounting of the portion of the sample above 4.5 leads to a binomial estimator with a variance of $\mathbb{P}(Z > 4.5)\mathbb{P}(Z < 4.5)/m$, which results in a lower bound

$$m \geq \mathbb{P}(Z > 4.5)\mathbb{P}(Z < 4.5)/1.510^{-13} \approx 0.7510^7\,,$$

i.e. close to ten million simulations.

Exercise 3.13

For the three choices, the importance weights are easily computed:

```
x1=sample(c(-1,1),10^4,rep=T)*rexp(10^4)
w1=exp(-sqrt(abs(x1)))*sin(x1)^2*(x1>0)/.5*dexp(x1)
x2=rcauchy(10^4)*2
w2=exp(-sqrt(abs(x2)))*sin(x2)^2*(x2>0)/dcauchy(x2/2)
x3=rnorm(10^4)
w3=exp(-sqrt(abs(x3)))*sin(x3)^2*(x3>0)/dnorm(x3)
```

They can be evaluated in many ways, from

```
boxplot(as.data.frame(cbind(w1,w2,w3)))
```

to computing the effective sample size `1/sum((w1/sum(w1))^2)` introduced in Example 3.10. The preferable choice is then g_1. The estimated sizes are given by

```
> 4*10^6*var(x1*w1/sum(w1))/mean(x1*w1/sum(w1))^2
[1] 10332203
> 4*10^6*var(x2*w2/sum(w2))/mean(x2*w2/sum(w2))^2
[1] 43686697
> 4*10^6*var(x3*w3/sum(w3))/mean(x3*w3/sum(w3))^2
[1] 352952159
```

again showing the appeal of using the double exponential proposal. (Note that efficiency could be doubled by considering the absolute values of the simulations.)

Exercise 3.15

a. With a positive density g and the representation

$$m(x) = \int_\Theta f(x|\theta)\frac{\pi(\theta)}{g(\theta)}g(\theta)\,\mathrm{d}\theta\,,$$

we can simulate θ_i's from g to approximate $m(x)$ with

$$\frac{1}{n}\sum_{i=1}^{n}\frac{f(x|\theta_i)\pi(\theta_i)}{g(\theta_i)}\,.$$

b. When $g(x) = \pi(\theta|x) = f(x|\theta)\pi(\theta)/K$, then

$$K\frac{1}{n}\sum_{i=1}^{n}\frac{f(x|X_i)\pi(X_i)}{f(X_i|\theta)\pi(\theta)} = K$$

and the normalisation constant is the exact estimate. If the normalising constant is unknown, we must use instead the self-normalising version (3.7).

c. Since

$$\int_\Theta \frac{\tau(\theta)}{f(x|\theta)\pi(\theta)}\pi(\theta|x)\mathrm{d}\theta = \int_\Theta \frac{\tau(\theta)}{f(x|\theta)\pi(\theta)}\frac{f(x|\theta)\pi(\theta)}{m(x)}\mathrm{d}\theta = \frac{1}{m(x)}\,,$$

we have an unbiased estimator of $1/m(x)$ based on simulations from the posterior,

$$\frac{1}{T}\sum_{t=1}^{T}\frac{\tau(\theta_i^*)}{f(x|\theta_i^*)\pi(\theta_i^*)}$$

and hence a converging (if biased) estimator of $m(x)$. This estimator of the marginal density can then be seen as an harmonic mean estimator, but also as an importance sampling estimator (Robert and Marin, 2010).

Exercise 3.17

Warning: There is a typo in question b, which should read

Let $X|Y = y \sim \mathcal{G}(1, y)$ and $Y \sim \mathcal{E}xp(1)$.

a. If $(X_i, Y_i) \sim f_{XY}(x, y)$, the Strong Law of Large Numbers tells us that

$$\lim_n \frac{1}{n}\sum_{i=1}^{n}\frac{f_{XY}(x^*, y_i)w(x_i)}{f_{XY}(x_i, y_i)} = \int\int \frac{f_{XY}(x^*, y)w(x)}{f_{XY}(x, y)}f_{XY}(x, y)\mathrm{d}x\mathrm{d}y.$$

Now cancel $f_{XY}(x, y)$ and use that fact that $\int w(x)dx = 1$ to show

$$\int\int \frac{f_{XY}(x^*, y)w(x)}{f_{XY}(x, y)}f_{XY}(x, y)\mathrm{d}x\mathrm{d}y = \int f_{XY}(x^*, y)dy = f_X(x^*).$$

b. The exact marginal is

$$\int \left[ye^{-yx}\right]e^{-y}dy = \int y^{2-1}e^{-y(1+x)}dy = \frac{\gamma(2)}{(1+x)^2}.$$

We tried the following `R` version of Monte Carlo marginalization:

```
X=rep(0,nsim)
Y=rep(0,nsim)
for (i in 1:nsim){
```

```
   Y[i]=rexp(1)
   X[i]=rgamma(1,1,rate=Y[i])
   }

MCMarg=function(x,X,Y){
  return(mean((dgamma(x,1,rate=Y)/dgamma(X,1,
      rate=Y))*dgamma(X,7,rate=3)))
  }
True=function(x)(1+x)^(-2)
```

which uses a $\mathcal{G}a(7,3)$ distribution to marginalize. It works ok, as you can check by looking at the plot

```
> xplot=seq(0,5,.05);plot(xplot,MCMarg(xplot,X,Y)-True(xplot))
```

c. Choosing $w(x) = f_X(x)$ leads to the estimator

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\frac{f_{XY}(x^*,y_i)f_X(x_i)}{f_{XY}(x_i,y_i)} &= \frac{1}{n}\sum_{i=1}^{n}\frac{f_X(x^*)f_{Y|X}(y_i|x^*)f_X(x_i)}{f_X(x_i)f_{Y|X}(y_i|x_i)}\\ &= f_X(x^*)\,\frac{1}{n}\sum_{i=1}^{n}\frac{f_{Y|X}(y_i|x^*)}{f_{Y|X}(y_i|x_i)}\end{aligned}$$

which produces $f_X(x^*)$ modulo an estimate of 1. If we decompose the variance of the estimator in terms of

$$\text{var}\left\{\mathbb{E}\left[\frac{f_{XY}(x^*,y_i)w(x_i)}{f_{XY}(x_i,y_i)}\middle| x_i\right]\right\} + \mathbb{E}\left\{\text{var}\left[\frac{f_{XY}(x^*,y_i)w(x_i)}{f_{XY}(x_i,y_i)}\middle| x_i\right]\right\},$$

the first term is

$$\begin{aligned}\mathbb{E}\left[\frac{f_{XY}(x^*,y_i)w(x_i)}{f_{XY}(x_i,y_i)}\middle| x_i\right] &= f_X(x^*)\mathbb{E}\left[\frac{f_{Y|X}(y_i|x^*)}{f_{Y|X}(y_i|x_i)}\middle| x_i\right]\,\frac{w(x_i)}{f_X(x_i)}\\ &= f_X(x^*)\frac{w(x_i)}{f_X(x_i)}\end{aligned}$$

which has zero variance if $w(x) = f_X(x)$. If we apply a variation calculus argument to the whole quantity, we end up with

$$w(x) \propto f_X(x)\Big/\int\frac{f^2_{Y|X}(y|x^*)}{f_{Y|X}(y|x)}\,\mathrm{d}y$$

minimizing the variance of the resulting estimator. So it is likely f_X is *not* optimal...

4

Controling and Accelerating Convergence

Exercise 4.1

a. Since

$$\pi_1(\theta|x) = \tilde{\pi}_1(\theta)/c_1 \text{ and } \pi_2(\theta|x) = \tilde{\pi}_2(\theta)/c_2\,,$$

where only $\tilde{\pi}_1$ and $\tilde{\pi}_2$ are known and where c_1 and c_2 correspond to the marginal likelihoods, $m_1(x)$ and $m_2(x)$ (the dependence on x is removed for simplification purposes), we have that

$$\varrho = \frac{m_1(x)}{m_2(x)} = \frac{\int_{\Theta_1} \pi_1(\theta) f_1(x|\theta)\,\mathrm{d}\theta}{\int_{\Theta_1} \pi_2(\theta) f_2(x|\theta)\,\mathrm{d}\theta} = \int_{\Theta_1} \frac{\pi_1(\theta) f_1(x|\theta)}{\tilde{\pi}_2(\theta)}\,\frac{\tilde{\pi}_2(\theta)}{m_2(x)}\mathrm{d}\theta_1$$

and therefore $\tilde{\pi}_1(\theta)/\tilde{\pi}_2(\theta)$ is an unbiased estimator of ϱ when $\theta \sim \pi_2(\theta|x)$.

b. Quite similarly,

$$\frac{\int \tilde{\pi}_1(\theta)\alpha(\theta)\pi_2(\theta|x)\mathrm{d}\theta}{\int \tilde{\pi}_2(\theta)\alpha(\theta)\pi_1(\theta|x)\mathrm{d}\theta} = \frac{\int \tilde{\pi}_1(\theta)\alpha(\theta)\tilde{\pi}_2(\theta)/c_2\mathrm{d}\theta}{\int \tilde{\pi}_2(\theta)\alpha(\theta)\tilde{\pi}_1(\theta)/c_1\mathrm{d}\theta} = \frac{c_1}{c_2} = \varrho\,.$$

Exercise 4.3

We have

$$\begin{aligned}\mathrm{ESS}_n &= 1\Big/\sum_{i=1}^n \underline{w}_i^2 = 1\Big/\sum_{i=1}^n \left(w_i\Big/\sum_{j=1}^n w_j\right)^2 \\ &= \frac{\left(\sum_{i=1}^n w_i\right)^2}{\sum_{i=1}^n w_i^2} = \frac{\sum_{i=1}^n w_i^2 + \sum_{i\neq j} w_i w_j}{\sum_{i=1}^n w_i^2} \leq n\end{aligned}$$

(This is also a consequence of Jensen's inequality when considering that the $\underline{w}_i$ sum up to one.) Moreover, the last equality shows that

$$ESS_n = 1 + \frac{\sum_{i\neq j} w_i w_j}{\sum_{i=1}^n w_i^2} \geq 1\,,$$

with equality if and only if a single ω_i is different from zero.

Exercise 4.5

Warning: There is a slight typo in the above in that $\bar{\mathbf{X}}_k$ should not be in bold. It should thus read

Establish that

$$\operatorname{cov}(\bar{X}_k, \bar{X}_{k'}) = \sigma^2 / \max\{k, k'\}.$$

Since the X_i's are iid, for $k' < k$, we have

$$\begin{aligned}
\operatorname{cov}(\overline{X}_k, \overline{X}_{k'}) &= \operatorname{cov}\left(\frac{1}{k}\sum_{i=1}^{k} X_i, \frac{1}{k'}\sum_{i=1}^{k'} X_i\right) \\
&= \operatorname{cov}\left(\frac{1}{k}\sum_{i=1}^{k'} X_i, \frac{1}{k'}\sum_{i=1}^{k'} X_i\right) \\
&= \frac{1}{kk'}\operatorname{cov}\left(\sum_{i=1}^{k'} X_i, \sum_{i=1}^{k'} X_i\right) \\
&= \frac{1}{kk'} k' \operatorname{cov}\left(X_i, X_i\right) \\
&= \sigma^2/k \\
&= \sigma^2 / \max\{k, k'\}.
\end{aligned}$$

Exercise 4.9

Warning: There is a missing variance term in this exercise, which should read

Show that

$$\begin{aligned}
\mathbb{E}\left[\exp -X^2 | y\right] &= \frac{1}{\sqrt{2\pi\sigma^2/y}} \int \exp\{-x^2\} \exp\{-(x-\mu)^2 y / 2\sigma^2\} \, \mathrm{d}x \\
&= \frac{1}{\sqrt{2\sigma^2/y + 1}} \exp\left\{-\frac{\mu^2}{1 + 2\sigma^2/y}\right\}
\end{aligned}$$

by completing the square in the exponent to evaluate the integral.

We have

$$\begin{aligned}
2x^2 + (x-\mu)^2 y / 2\sigma^2 &= x^2(2 + y\sigma^{-2}) - 2x\mu y\sigma^{-2} + \mu^2 y\sigma^{-2} \\
&= (2 + y\sigma^{-2}) \left[x - \mu y\sigma^{-2}/(2 + y\sigma^{-2})\right]^2 + \\
&\quad \mu^2 \left[y\sigma^{-2} - y^2\sigma^{-4}/(2 + y\sigma^{-2})\right] \\
&= (2 + y\sigma^{-2}) \left[x - \mu/(1 + 2\sigma^2/y)\right]^2 + \mu^2/(1 + 2\sigma^2/y)
\end{aligned}$$

and thus

$$\begin{aligned}\int \exp\{-x^2\}\,\exp\{-(x-\mu)^2 y/2\sigma^2\}\,\frac{\mathrm{d}x}{\sqrt{2\pi\sigma^2/y}} &= \exp\left\{-\frac{\mu^2}{1+2\sigma^2/y}\right\}\\ &\quad\times\int \exp\left\{-(2+y\sigma^{-2})\left[x-\mu/(1+2\sigma^2/y)\right]^2/2\right\}\,\frac{\mathrm{d}x}{\sqrt{2\pi\sigma^2/y}}\\ &= \exp\left\{-\frac{\mu^2}{1+2\sigma^2/y}\right\}\,\frac{\sqrt{y\sigma^{-2}}}{\sqrt{2+y\sigma^{-2}}}\\ &= \exp\left\{-\frac{\mu^2}{1+2\sigma^2/y}\right\}\,\frac{1}{\sqrt{1+2\sigma^2/y}}\end{aligned}$$

Exercise 4.11

Since $H(U)$ and $H(1_U)$ take opposite values when H is monotone, i.e. one is large when the other is small, those two random variables are negatively correlated.

Exercise 4.13

Warning: Another reference problem in this exercise: Exercise 4.2 should be Exercise 4.1.

a. The ratio (4.9) is a ratio of convergent estimators of the numerator and the denominator in question b of Exercise 4.1 when $\theta_{1i} \sim \pi_1(\theta|x)$ and $\theta_{2i} \sim \pi_2(\theta|x)$. (Note that the wording of this question is vague in that it does not indicate the dependence on x.)

b. If we consider the special choice $\alpha(\theta) = 1/\tilde{\pi}_1(\theta)\tilde{\pi}_2(\theta)$ in the representation of question b of Exercise 4.1, we do obtain $\varrho = \mathbb{E}^{\pi_2}[\tilde{\pi}_2(\theta)^{-1}]/\mathbb{E}^{\pi_1}[\tilde{\pi}_1(\theta)^{-1}]$, assuming both expectations exist. Given that $(i = 1, 2)$

$$\mathbb{E}^{\pi_i}[\tilde{\pi}_i(\theta)^{-1}] = \int_\Theta \frac{1}{\tilde{\pi}_i(\theta)}\,\frac{\tilde{\pi}_i(\theta)}{m_i(x)}\,\mathrm{d}\theta\,,$$

this implies that the space Θ must have a finite measure. If $\mathrm{d}\theta$ represents the dominating measure, Θ is necessarily compact.

Exercise 4.15

Each of those `R` programs compare the range of the Monte Carlo estimates with and without Rao–Blackwellization:

a. For the negative binomial mean, $\mathbb{E}_f(X) = a/b$ since $X \sim \mathcal{N}eg(a, b/(b+1))$.

```
y=matrix(rgamma(100*Nsim,a)/b,ncol=100)
x=matrix(rpois(100*Nsim,y),ncol=100)
matplot(apply(x,2,cumsum)/(1:Nsim),type="l",col="grey80",
lty=1,ylim=c(.4*a/b,2*a/b), xlab="",ylab="")
matplot(apply(y,2,cumsum)/(1:Nsim),type="l",col="grey40",
lty=1,add=T,xlab="",ylab="")
abline(h=a/b,col="gold",lty=2,lwd=2)
```

b. For the generalized t variable, $\mathbb{E}_f(X) = BE_f[X|Y] = 0$. So the improvement is obvious. To make a more sensible comparison, we consider instead $\mathbb{E}_f[X^2] = \mathbb{E}[Y] = a/b$.

```
y=matrix(rgamma(100*Nsim,a)/b,ncol=100)
x=matrix(rnorm(100*Nsim,sd=sqrt(y)),ncol=100)
matplot(apply(x^2,2,cumsum)/(1:Nsim),type="l",col="grey80",
lty=1,ylim=(a/b)*c(.2,2), xlab="",ylab="")
matplot(apply(y,2,cumsum)/(1:Nsim),type="l",col="grey40",
lty=1,add=T,xlab="",ylab="")
abline(h=a/b,col="gold",lty=2,lwd=2)
```

c. **Warning: There is a typo in this question with a missing n in the $\mathcal{B}in(y)$ distribution... It should be**

c. $X|y \sim \mathcal{B}in(n, y)$, $Y \sim \mathcal{B}e(a, b)$ (X is beta-binomial).

In this case, $\mathbb{E}_f[X] = n\mathbb{E}_f[Y] = na/(a + b)$.

```
y=1/matrix(1+rgamma(100*Nsim,b)/rgamma(100*Nsim,a),ncol=100)
x=matrix(rbinom(100*Nsim,n,prob=y),ncol=100)
matplot(apply(x,2,cumsum)/(1:Nsim),type="l",col="grey80",lty=1,
ylim=(n*a/(a+b))*c(.2,2), xlab="",ylab="")
matplot(n*apply(y,2,cumsum)/(1:Nsim),type="l",col="grey40",lty=1,add=T,
xlab="",ylab="")
abline(h=n*a/(a+b),col="gold",lty=2,lwd=2)
```

Exercise 4.17

It should be clear from display (4.5) that we only need to delete the n^2 (k^2 in the current notation). We replace it with $2k^2$ and add the last row and column as in (4.5).

Exercise 4.19

a. For the accept-reject algorithm,

$$(X_1, \ldots, X_m) \sim f(x)$$
$$(U_1, \ldots, U_N) \overset{\text{i.i.d.}}{\sim} \mathcal{U}_{[0,1]}$$
$$(Y_1, \ldots, Y_N) \overset{\text{i.i.d.}}{\sim} g(y)$$

and the acceptance weights are the $w_j = \frac{f(Y_j)}{Mg(Y_j)}$. N is the stopping time associated with these variables, that is, $Y_N = X_m$. We have

$$\begin{aligned}\rho_i &= P\left(U_i \leq w_i | N = n, Y_1, \ldots, Y_n\right) \\ &= \frac{P\left(U_i \leq w_i, N = n, Y_1, \ldots, Y_n\right)}{P\left(N = n, Y_1, \ldots, Y_n\right)}\end{aligned}$$

where the numerator is the probability that Y_N is accepted as X_m, Y_i is accepted as one X_j and there are $(m-2)$ X_j's that are chosen from the remaining $(n-2)$ Y_ℓ's. Since

$$P\left(Y_j \text{ is accepted}\right) = P\left(U_j \leq w_j\right) = w_j\,,$$

the numerator is

$$w_i \sum_{(i_1,\ldots,i_{m-2})} \prod_{j=1}^{m-2} w_{i_j} \prod_{j=m-1}^{n-2} (1 - w_{i_j})$$

where

i) $\prod_{j=1}^{m-2} w_{i_j}$ is the probability that among the N Y_j's, in addition to both Y_N and Y_i being accepted, there are $(m-2)$ other Y_j's accepted as X_ℓ's;
ii) $\prod_{j=m-1}^{n-2}(1 - w_{i_j})$ is the probability that there are $(n-m)$ rejected Y_j's, given that Y_i and Y_N are accepted;
iii) the sum is over all subsets of $(1, \ldots, i-1, i+1, \ldots, n)$ since, except for Y_i and Y_N, other $(m-2)$ Y_j's are chosen uniformly from $(n-2)$ Y_j's.

Similarly the denominator

$$P\left(N = n, Y_1, \ldots, Y_n\right) = w_i \sum_{(i_1,\ldots,i_{m-1})} \prod_{j=1}^{m-1} w_{i_j} \prod_{j=m}^{n-1} (1 - w_{i_j})$$

is the probability that Y_N is accepted as X_m and $(m-1)$ other X_j's are chosen from $(n-1)$ Y_ℓ's. Thus

$$\begin{aligned}\rho_i &= P\left(U_i \leq w_i | N = n, Y_1, \ldots, Y_n\right) \\ &= w_i \frac{\sum_{(i_1,\ldots,i_{m-2})} \prod_{j=1}^{m-2} w_{i_j} \prod_{j=m-1}^{n-2} (1 - w_{i_j})}{\sum_{(i_1,\ldots,i_{m-1})} \prod_{j=1}^{m-1} w_{i_j} \prod_{j=m-1}^{n-1} (1 - w_{i_j})}\end{aligned}$$

b. We have

$$\delta_1 = \frac{1}{m}\sum_{i=1}^{m} h\left(X_i\right) = \frac{1}{m}\sum_{j=1}^{N} h\left(Y_j\right)\mathbb{I}_{U_j \le w_j}$$
$$\delta_2 = \frac{1}{m}\sum_{j=1}^{N} \mathbb{E}\left(\mathbb{I}_{U_j \le w_j}|N, Y_1, \ldots, Y_N\right) h\left(Y_j\right) = \frac{1}{m}\sum_{i=1}^{N} \rho_i h\left(Y_i\right)$$

Since $\mathbb{E}\left(\mathbb{E}\left(X|Y\right)\right) = \mathbb{E}\left(X\right)$,

$$\begin{aligned}\mathbb{E}\left(\delta_2\right) &= \mathbb{E}\left(\frac{1}{m}\sum_{j=1}^{N} \mathbb{E}\left(\mathbb{I}_{U_j \le w_j}|N, Y_1, \ldots, Y_N\right)\right) \\ &= \frac{1}{m}\sum_{j=1}^{N} \mathbb{E}\left(\mathbb{I}_{U_j \le w_j}\right) h\left(Y_j\right) \\ &= \mathbb{E}\left(\frac{1}{m}\sum_{j=1}^{N} h\left(Y_j\right)\mathbb{I}_{U_j \le w_j}\right) = \mathbb{E}\left(\delta_1\right)\end{aligned}$$

Under quadratic loss, the risk of δ_1 and δ_2 are:

$$\begin{aligned}R\left(\delta_1\right) &= \mathbb{E}\left(\delta_1 - \mathbb{E}h\left(X\right)\right)^2 \\ &= \mathbb{E}\left(\delta_1^2\right) + \mathbb{E}\left(\mathbb{E}(h(X))\right)^2 - 2\mathbb{E}\left(\delta_1 \mathbb{E}(h(X))\right) \\ &= \mathrm{var}\left(\delta_1\right) - \left(\mathbb{E}(\delta_1)\right)^2 + \mathbb{E}\left(\mathbb{E}\left(h(X)\right)\right)^2 - 2\mathbb{E}\left(\delta_1 \mathbb{E}\left(h(X)\right)\right)\end{aligned}$$

and

$$\begin{aligned}R\left(\delta_2\right) &= \mathbb{E}\left(\delta_2 - \mathbb{E}h\left(X\right)\right)^2 \\ &= \mathbb{E}\left(\delta_2^2\right) + \mathbb{E}\left(\mathbb{E}(h(X))\right)^2 - 2\mathbb{E}\left(\delta_2 \mathbb{E}(h(X))\right) \\ &= \mathrm{var}\left(\delta_2\right) - \left(\mathbb{E}(\delta_2)\right)^2 + \mathbb{E}\left(\mathbb{E}\left(h(X)\right)\right)^2 - 2\mathbb{E}\left(\delta_2 \mathbb{E}\left(h(X)\right)\right)\end{aligned}$$

Since $\mathbb{E}\left(\delta_1\right) = \mathbb{E}\left(\delta_2\right)$, we only need to compare $\mathrm{var}\left(\delta_1\right)$ and $\mathrm{var}\left(\delta_2\right)$. From the definition of δ_1 and δ_2, we have

$$\delta_2(X) = \mathbb{E}\left(\delta_1(X)|Y\right)$$

so

$$\mathrm{var}\left(\mathbb{E}\left(\delta_1\right)\right) = \mathrm{var}\left(\delta_2\right) \le \mathrm{var}\left(\delta_1\right) .$$

Exercise 4.21

a. Let us transform $\mathfrak{I}$ into $\mathfrak{I} = \int \frac{h(y)f(y)}{m(y)} m(y)dy$, where m is the marginal density of Y_1. We have

$$\begin{aligned}\mathfrak{I} &= \sum_{n\in\mathbb{N}} \left[P(N=n) \int \frac{h(y)f(y)}{m(y|N=n)} \right] \\ &= \mathbb{E}_N \left[\mathbb{E}\left[\frac{h(y)f(y)}{m(y)} | N \right] \right].\end{aligned}$$

b. As β is constant, for every function c,

$$\mathfrak{I} = \beta\mathbb{E}[c(Y)] + \mathbb{E}\left[\frac{h(Y)f(Y)}{m(Y)} - \beta c(Y) \right].$$

c. The variance associated with an empirical mean of the

$$\frac{h(Y_i)f(Y_i)}{m(Y_i)} - \beta c(Y_i)$$

is

$$\begin{aligned}\text{var}(\widehat{\mathfrak{I}}) &= \beta^2 \text{var}(c(Y)) + \text{var}\left(\frac{h(Y)f(Y)}{m(Y)} \right) - 2\beta \text{cov}\left[\frac{h(Y)f(Y)}{m(Y)}, c(Y) \right] \\ &= \beta^2 \text{var}(c(Y)) - 2\beta \text{cov}[d(Y), c(Y)] + \text{var}(d(Y)).\end{aligned}$$

Thus, the optimal choice of β is such that

$$\frac{\partial \text{var}(\widehat{\mathfrak{I}})}{\partial \beta} = 0$$

and is given by

$$\beta^* = \frac{\text{cov}[d(Y), c(Y)]}{\text{var}(c(Y))}.$$

d. The first choice of c is $c(y) = \mathbb{I}_{\{y>y_0\}}$, which is interesting when $p = P(Y > y_0)$ is known. In this case,

$$\beta^* = \frac{\int_{y>y_0} hf - \int_{y>y_0} hf \int_{y>y_0} m}{\int_{y>y_0} m - (\int_{y>y_0} m)^2} = \frac{\int_{y>y_0} hf}{p}.$$

Thus, β^* can be estimated using the Accept-reject sample. A second choice of c is $c(y) = y$, which leads to the two first moments of Y. When those two moments m_1 and m_2 are known or can be well approximated, the optimal choice of β is

$$\beta^* = \frac{\int yh(y)f(y)dy - \mathfrak{I}m_1}{m_2}.$$

and can be estimated using the same sample or another instrumental density namely when $\mathfrak{I}' = \int yh(y)f(y)dy$ is simple to compute, compared to $\mathfrak{I}$.

5

Monte Carlo Optimization

Exercise 5.1

This is straightforward in R

```
par(mfrow=c(1,2),mar=c(4,4,1,1))
image(mu1,mu2,-lli,xlab=expression(mu[1]),ylab=expression(mu[2]))
contour(mu1,mu2,-lli,nle=100,add=T)
Nobs=400
da=rnorm(Nobs)+2.5*sample(0:1,Nobs,rep=T,prob=c(1,3))
for (i in 1:250)
for (j in 1:250)
  lli[i,j]=like(c(mu1[i],mu2[j]))
image(mu1,mu2,-lli,xlab=expression(mu[1]),ylab=expression(mu[2]))
contour(mu1,mu2,-lli,nle=100,add=T)
```

Figure 5.1 shows that the log-likelihood surfaces are quite comparable, despite being based on different samples. Therefore the impact of allocating 100 and 300 points to both components, respectively, instead of the random 79 and 321 in the current realisation, is inconsequential.

Exercise 5.3

Warning: as written, this problem has not simple solution! The constraint should be replaced with

$$x^2(1+\sin(y/3)\cos(8x))+y^2(2+\cos(5x)\cos(8y))\le 1\,,$$

We need to find a lower bound on the function of (x,y). The coefficient of y^2 is obviously bounded from below by 1, while the coefficient of x^2 is positive. Since the function is bounded from below by y^2, this means that $y^2<1$, hence that $\sin(y/3)>\sin(-1/3)>-.33$. Therefore, a lower bound on the

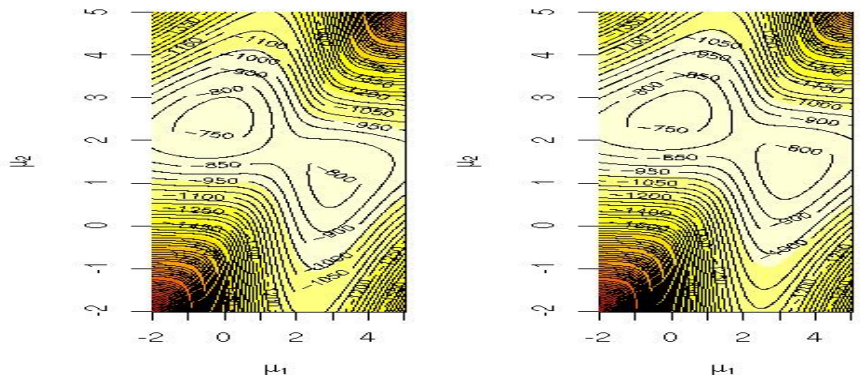

Fig. 5.1. Comparison of two log-likelihood surfaces for the mixture model (5.2) when the data is simulated with a fixed 100/300 ratio in both components *(left)* and when the data is simulated with a binomial $\mathcal{B}(400, 1/4)$ random number of points on the first component.

function is $0.77x^2 + y^2$. If we simulate uniformly over the ellipse $0.77x^2 + y^2 < 1$, we can subsample the points that satisfy the constraint. Simulating the uniform distribution on $0.77x^2 + y^2 < 1$ is equivalent to simulate the uniform distribution over the unit circle $z^2 + y^2 < 1$ and resizing z into $x = z/\sqrt{0.77}$.

```
theta=runif(10^5)*2*pi
rho=runif(10^5)
xunif=rho*cos(theta)/.77
yunif=rho*sin(theta)
plot(xunif,yunif,pch=19,cex=.4,xlab="x",ylab="y")
const=(xunif^2*(1+sin(yunif/3)*cos(xunif*8))+
 yunif^2*(2+cos(5*xunif)*cos(8*yunif))<1)
points(xunif[const],yunif[const],col="cornsilk2",pch=19,cex=.4)
```

While the ellipse is larger than the region of interest, Figure 5.2 shows that it is reasonably efficient. The performances of the method are given by `sum(const)/10^4`, which is equal to 73%.

Exercise 5.5

Since the log-likelihood of the mixture model in Example 5.2 has been defined by

```
#minus the log-likelihood function
like=function(mu){
  -sum(log((.25*dnorm(da-mu[1])+.75*dnorm(da-mu[2]))))
  }
```

in the mcsm package, we can reproduce the R program of Example 5.7 with the function h now defined as `like`. The difference with the function h of

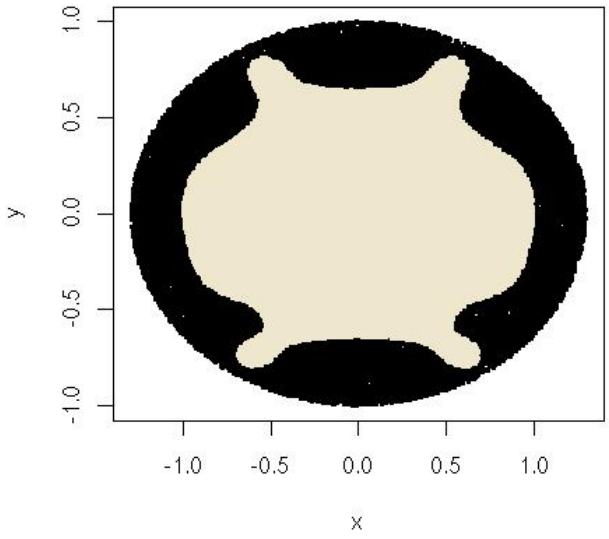

Fig. 5.2. Simulation of a uniform distribution over a complex domain via uniform simulation over a simpler encompassing domain for 10^5 simulations and an acceptance rate of 0.73%.

Example 5.7 is that the mixture log-likelihood is more variable and thus the factors α_j and β_j need to be calibrated against divergent behaviours. The following figure shows the impact of the different choices $(\alpha_j, \beta_j) = (.01/\log(j+1), 1/\log(j+1)^{.5})$, $(\alpha_j, \beta_j) = (.1/\log(j+1), 1/\log(j+1)^{.5})$, $(\alpha_j, \beta_j) = (.01/\log(j+1), 1/\log(j+1)^{.1})$, $(\alpha_j, \beta_j) = (.1/\log(j+1), 1/\log(j+1)^{.1})$, on the convergence of the gradient optimization. In particular, the second choice exhibits a particularly striking behavior where the sequence of (μ_1, μ_2) skirts the true mode of the likelihood in a circular manner. (The stopping rule used in the `R` program is `(diff<10^(-5))`.)

Exercise 5.7

The `R` function `SA` provided in Example 5.9 can be used in the following `R` program to test whether or not the final value is closer to the main mode or to the secondy mode:

```
modes=matrix(0,ncol=2,nrow=100)
prox=rep(0,100)
for (t in 1:100){
```

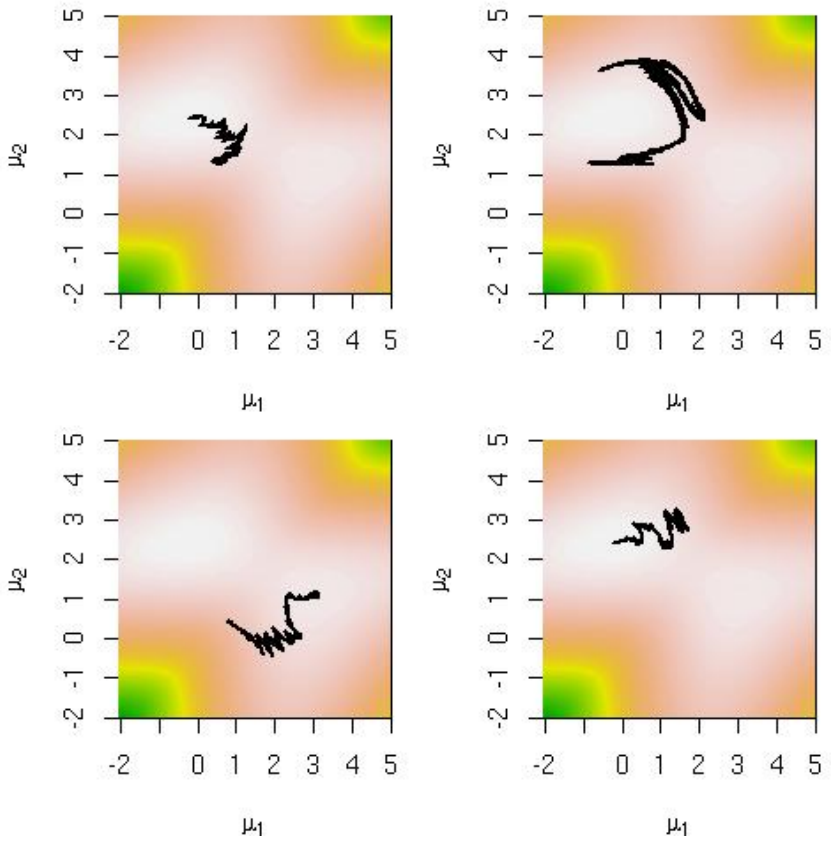

Fig. 5.3. Four stochastic gradient paths for four different choices $(\alpha_j, \beta_j) = (.01/\log(j+1), 1/\log(j+1)^{.5})$ (u.l.h.s.), $(\alpha_j, \beta_j) = (.1/\log(j+1), 1/\log(j+1)^{.5})$ (u.r.h.s.), $(\alpha_j, \beta_j) = (.01/\log(j+1), 1/\log(j+1)^{.1})$ (l.l.h.s.), $(\alpha_j, \beta_j) = (.1/\log(j+1), 1/\log(j+1)^{.1})$ (l.r.h.s.).

```
res=SA(mean(da)+rnorm(2))
modes[t,]=res$the[res$ite,]
diff=modes[t,]-c(0,2.5)
duff=modes[t,]-c(2.5,0)
prox[t]=sum(t(diff)%*%diff<t(duff)%*%duff)
}
```

For each new temperature schedule, the function `SA` must be modified accordingly (for instance by the on-line change `SA=vi(SA)`). Figure 5.4 illustrates the output of an experiment for four different schedules.

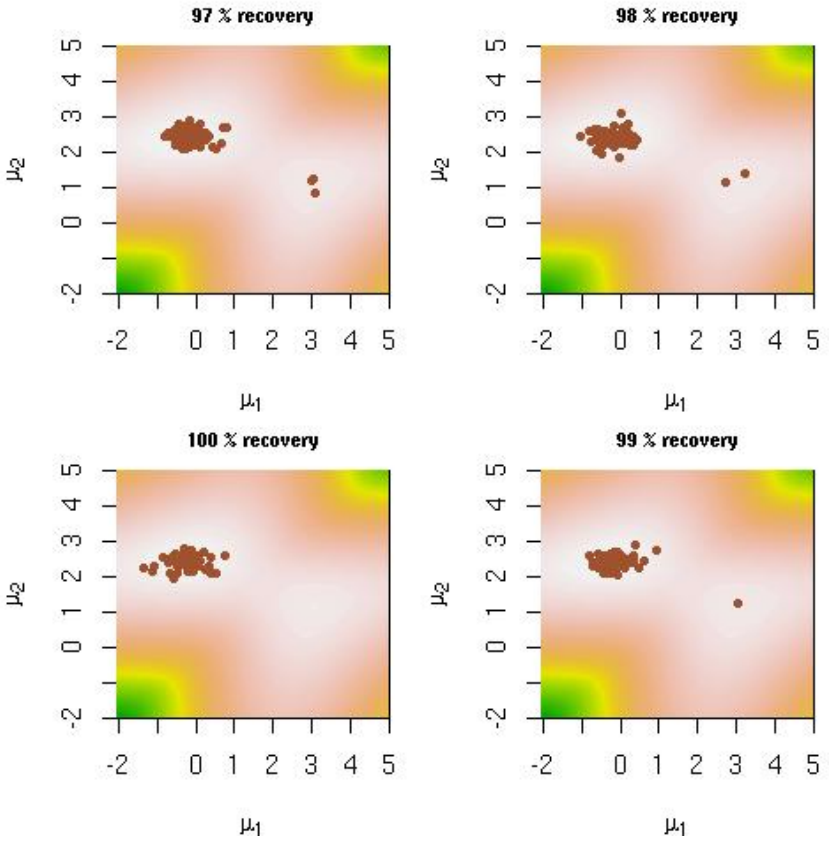

Fig. 5.4. Four simulated annealing outcomes corresponding to the temperature schedules $T_t = 1/1\log(1+t)$, $T_t = 1/10\log(1+t)$, $T_t = 1/10\sqrt{\log(1+t)}$, and $T_t = (.95)^{1+t}$, based on 100 replications. (The percentage of recoveries of the main mode is indicated in the title of each graph.)

Exercise 5.9

In principle, $Q(\theta'|\theta, \mathbf{x})$ should also involve the logarithms of 1/4 and 1/3, raised to the powers $\sum Z_i$ and $\sum(1 - Z_i)$, respectively. But, due to the logarithmic transform, the expression does not involve the parameter $\theta = (\mu_1, \mu_2)$ and can thus be removed from $Q(\theta'|\theta, \mathbf{x})$ with no impact on the optimization problem.

Exercise 5.11

Warning: there is a typo in Example 5.16. The EM sequence should be

$$\hat{\theta}_1 = \left\{ \frac{\theta_0\, x_1}{2+\theta_0} + x_4 \right\} \Big/ \left\{ \frac{\theta_0\, x_1}{2+\theta_0} + x_2 + x_3 + x_4 \right\}.$$

instead of having x_4 in the denominator.
Note first that some 1/4 factors have been removed from every term as they were not contributing to the likelihood maximisation. Given a starting point θ_0, the EM sequence will always be the same.

```
x=c(58,12,9,13)
n=sum(x)
start=EM=cur=diff=.1
while (diff>.001){ #stopping rule

  EM=c(EM,((cur*x[1]/(2+cur))+x[4])/((cur*x[1]/(2+cur))+x[2]+x[3]+x[4]))
  diff=abs(cur-EM[length(EM)])
  cur=EM[length(EM)]
  }
```

The Monte Carlo EM version creates a sequence based on a binomial simulation:

```
M=10^2
MCEM=matrix(start,ncol=length(EM),nrow=500)
for (i in 2:length(EM)){
  MCEM[,i]=1/(1+(x[2]+x[3])/(x[4]+rbinom(500,M*x[1],
  prob=1/(1+2/MCEM[,i-1]))/M))
  }
plot(EM,type="l",xlab="iterations",ylab="MCEM sequences")
upp=apply(MCEM,2,max);dow=apply(MCEM,2,min)
polygon(c(1:length(EM),length(EM):1),c(upp,rev(dow)),col="grey78")
lines(EM,col="gold",lty=2,lwd=2)
}
```

and the associated graph shows a range of values that contains the true EM sequence. Increasing `M` in the above `R` program obviously reduces the range.

Exercise 5.13

The `R` function for plotting the (log-)likelihood surface associated with (5.2) was provided in Example 5.2. We thus simply need to apply this function to the new sample, resulting in an output like Figure 5.5, with a single mode instead of the usual two modes.

Exercise 5.15

Warning: there is a typo in question a where the formula should involve capital Z_i's, namely

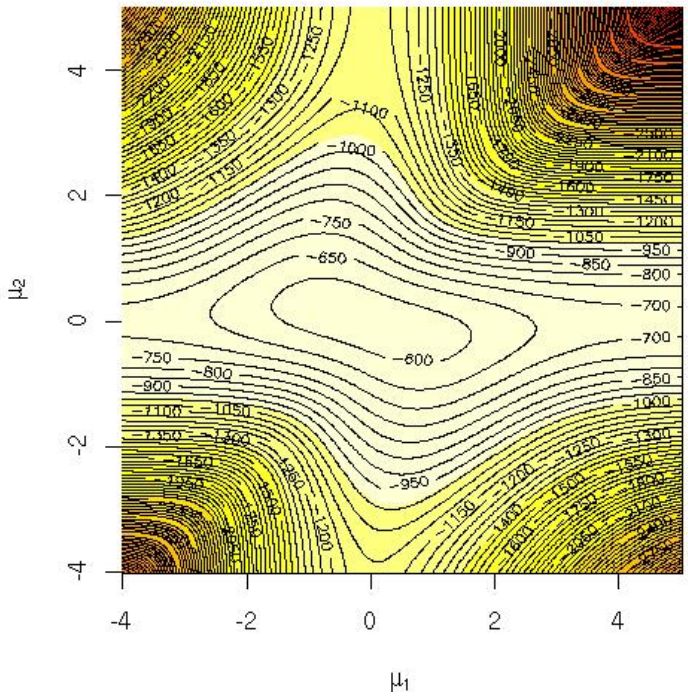

Fig. 5.5. Log-likelihood surface of a mixture model applied to a five component mixture sample of size 400.

$$P(Z_i = 1) = 1 - P(Z_i = 2) = \frac{p\lambda \exp(-\lambda x_i)}{p\lambda \exp(-\lambda x_i) + (1-p)\mu \exp(-\mu x_i)}.$$

a. The likelihood is

$$L(\theta|\mathbf{x}) = \prod_{i=1}^{12} [p\lambda e^{-\lambda x_i} + (1-p)\mu e^{-\mu x_i}],$$

and the complete-data likelihood is

$$L^c(\theta|\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{12} [p\lambda e^{-\lambda x_i} \mathbb{I}_{(z_i=1)} + (1-p)\mu e^{-\mu x_i} \mathbb{I}_{(z_i=2)}],$$

where $\theta = (p, \lambda, \mu)$ denotes the parameter, using the same arguments as in Exercise 5.14.

b. The EM algorithm relies on the optimization of the expected log-likelihood

$$Q(\theta|\hat{\theta}_{(j)},\mathbf{x}) = \sum_{i=1}^{12} \Big[\log\left(p\lambda e^{-\lambda x_i}\right) P_{\hat{\theta}_{(j)}}(Z_i = 1|x_i) + \log\left((1-p)\mu e^{-\mu x_i}\right) P_{\hat{\theta}_{(j)}}(Z_i = 2|x_i)\Big] .$$

The arguments of the maximization problem are

$$\begin{cases} \hat{p}_{(j+1)} = \hat{P}/12 \\ \hat{\lambda}_{(j+1)} = \hat{S}_1/\hat{P} \\ \hat{\mu}_{(j+1)} = \hat{S}_2/\hat{P}, \end{cases}$$

where

$$\begin{cases} \hat{P} = \sum_{i=1}^{12} P_{\hat{\theta}_{(j)}}(Z_i = 1|x_i) \\ \hat{S}_1 = \sum_{i=1}^{12} x_i P_{\hat{\theta}_{(j)}}(Z_i = 1|x_i) \\ \hat{S}_2 = \sum_{i=1}^{12} x_i P_{\hat{\theta}_{(j)}}(Z_i = 2|x_i) \end{cases}$$

with

$$P_{\hat{\theta}_{(j)}}(Z_i = 1|x_i) = \frac{\hat{p}_{(j)}\hat{\lambda}_{(j)} e^{-\hat{\lambda}_{(j)} x_i}}{\hat{p}_{(j)}\hat{\lambda}_{(j)} e^{-\hat{\lambda}_{(j)} x_i} + (1-\hat{p}_{(j)})\hat{\mu}_{(j)} e^{-\hat{\mu}_{(j)} x_i}} .$$

An `R` implementation of the algorithm is then

```
x=c(0.12,0.17,0.32,0.56,0.98,1.03,1.10,1.18,1.23,1.67,1.68,2.33)
EM=cur=c(.5,jitter(mean(x),10),jitter(mean(x),10))
diff=1
while (diff*10^5>1){

  probs=1/(1+(1-cur[1])*dexp(x,cur[3])/(cur[1]*dexp(x,cur[2])))
  phat=sum(probs);S1=sum(x*probs);S2=sum(x*(1-probs))
  EM=rbind(EM,c(phat/12,S1/phat,S2/phat))
  diff=sum(abs(cur-EM[dim(EM)[1],]))
  cur=EM[dim(EM)[1],]
  }
```

and it always produces a single component mixture.

Exercise 5.17

Warning: Given the notations of Example 5.14, the function ϕ in question b should be written φ...

a. The question is a bit vague in that the density of the missing data $(Z_{n-m+1},\ldots,Z_n)$ is a normal $\mathcal{N}(\theta,1)$ density if we do not condition on $\mathbf{y}$. Conditional upon $\mathbf{y}$, the missing observations Z_i are truncated in a, i.e. we know that they are larger than a. The conditional distribution of

the Z_i's is therefore a normal $\mathcal{N}(\theta, 1)$ distribution truncated in a, with density

$$f(z|\theta, y) = \frac{\exp\{-(z_i-\theta)^2/2\}}{\sqrt{2\pi}\, P_\theta(Y > a)}\, \mathbb{I}z \geq a\,. = \frac{\varphi(z-\theta)}{1-\Phi(a-\theta)}\, \mathbb{I}z \geq a\,.$$

where φ and Φ are the normal pdf and cdf, respectively.

b. We have

$$\begin{aligned}
\mathbb{E}_\theta[Z_i|Y_i] &= \int_a^\infty z\, \frac{\varphi(z-\theta)}{1-\Phi(a-\theta)}\, \mathrm{d}z \\
&= \theta + \int_a^\infty (z-\theta)\, \frac{\varphi(z-\theta)}{1-\Phi(a-\theta)}\, \mathrm{d}z \\
&= \theta + \int_{a-\theta}^\infty y\, \frac{\varphi(y)}{1-\Phi(a-\theta)}\, \mathrm{d}y \\
&= \theta + [-\varphi(x)]_{a-\theta}^\infty \\
&= \theta + \frac{\varphi(a-\theta)}{1-\Phi(a-\theta)},
\end{aligned}$$

since $\varphi'(x) = -x\varphi(x)$.

Exercise 5.19

Running `uniroot` on both intervals

```
> h=function(x){(x-3)*(x+6)*(1+sin(60*x))}
> uniroot(h,int=c(-2,10))
$root
[1] 2.999996
$f.root
[1] -6.853102e-06
> uniroot(h,int=c(-8,1))
$root
[1] -5.999977
$f.root
[1] -8.463209e-06
```

misses all solutions to $1 + \sin(60x) = 0$

Exercise 5.21

Warning: this Exercise duplicates Exercise 5.11 and should not have been included in the book!

6

Metropolis-Hastings Algorithms

Exercise 6.1

A simple `R` program to simulate this chain is

```
# (C.) Jiazi Tang, 2009
x=1:10^4
x[1]=rnorm(1)
r=0.9
for (i in 2:10^4){
  x[i]=r*x[i-1]+rnorm(1) }
hist(x,freq=F,col="wheat2",main="")
curve(dnorm(x,sd=1/sqrt(1-r^2)),add=T,col="tomato"
```

Exercise 6.3

When $q(y|x) = g(y)$, we have

$$\begin{aligned}\rho(x,y) &= \min\left(\frac{f(y)}{f(x)}\frac{q(x|y)}{q(y|x)}, 1\right)\\ &= \min\left(\frac{f(y)}{f(x)}\frac{g(x)}{g(y)}, 1\right)\\ &= \min\left(\frac{f(y)}{f(x)}\frac{g(x)}{g(y)}, 1\right)\,.\end{aligned}$$

Since the acceptance probability satisfies

$$\frac{f(y)}{f(x)}\frac{g(x)}{g(y)} \geq \frac{f(y)/g(y)}{\max f(x)/g(x)}$$

it is larger for Metropolis–Hastings than for accept-reject.

Exercise 6.5

a. The first property follows from a standard property of the normal distribution, namely that the linear transform of a normal is again normal. The second one is a consequence of the decomposition $y = X\beta + \epsilon$, when $\epsilon \sim \mathcal{N}_n(0, \sigma^2 I_n)$ is independent from $X\beta$.

b. This derivation is detailed in Marin and Robert (2007, Chapter 3, Exercise 3.9).
Since
$$\mathbf{y}|\sigma^2, X \sim \mathcal{N}_n(X\tilde{\beta}, \sigma^2(I_n + nX(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}))\,,$$
integrating in σ^2 with $\pi(\sigma^2) = 1/\sigma^2$ yields
$$f(\mathbf{y}|X) = (n+1)^{-(k+1)/2}\pi^{-n/2}\Gamma(n/2)\left[\mathbf{y}^{\mathrm{T}}\mathbf{y} - \frac{n}{n+1}\mathbf{y}^{\mathrm{T}}X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\mathbf{y} - \frac{1}{n+1}\tilde{\beta}^{\mathrm{T}}X^{\mathrm{T}}X\tilde{\beta}\right]^{-n/2}.$$
Using the `R` function `dmt(mnormt)`, we obtain the marginal density for the swiss dataset:

```
> y=log(as.vector(swiss[,1]))
> X=as.matrix(swiss[,2:6])
> library(mnormt)
> dmt(y,S=diag(length(y))+X%*%solve(t(X)%*%X)%*%t(X),d=length(y)-1)
[1] 2.096078e-63
```

with the prior value $\tilde{\beta} = 0$.

Exercise 6.7

a. We generate an Metropolis-Hastings sample from the $\mathcal{B}e(2.7, 6.3)$ density using uniform simulations:

```
# (C.) Thomas Bredillet, 2009
Nsim=10^4
a=2.7;b=6.3
X=runif(Nsim)
last=X[1]
for (i in 1:Nsim) {
        cand=rbeta(1,1,1)
        alpha=(dbeta(cand,a,b)/dbeta(last,a,b))/
      (dbeta(cand,1,1)/dbeta(last,1,1))
        if (runif(1)<alpha)
           last=cand
        X[i]=last
}
hist(X,pro=TRUE,col="wheat2",xlab="",ylab="",main="Beta(2.7,3) simulation")
curve(dbeta(x,a,b),add=T,lwd=2,col="sienna2")
```

The acceptance rate is estimated by

```
> length(unique(X))/5000
[1] 0.458
```

If instead we use a $\mathcal{B}e(20, 60)$ proposal, the modified lines in the `R` program are

```
cand=rbeta(20,60,1)
alpha=(dbeta(cand,a,b)/dbeta(last,a,b))/
      (dbeta(cand,20,60)/dbeta(last,20,60))
```

and the acceptance rate drops to zero!

b. In the case of a truncated beta, the following `R` program

```
Nsim=5000
a=2.7;b=6.3;c=0.25;d=0.75
X=rep(runif(1),Nsim)
test2=function(){
  last=X[1]
  for (i in 1:Nsim){
        cand=rbeta(1,2,6)
        alpha=(dbeta(cand,a,b)/dbeta(last,a,b))/
      (dbeta(cand,2,6)/dbeta(last,2,6))
        if ((runif(1)<alpha)&&(cand<d)&&(c<cand))
           last=cand
        X[i]=last}
  }
test1=function(){
  last=X[1]
  for (i in 1:Nsim){
        cand=runif(1,c,d)
        alpha=(dbeta(cand,a,b)/dbeta(last,a,b))
        if ((runif(1)<alpha)&&(cand<d)&&(c<cand))
           last=cand
        X[i]=last
        }
}
system.time(test1());system.time(test2())
```

shows very similar running times but more efficiency for the beta proposal, since the acceptance rates are approximated by 0.51 and 0.72 for `test1` and `test2`, respectively. When changing to $c = 0.25$, $d = 0.75$, `test1` is more efficient than `test2`, with acceptances rates of approximately 0.58 and 0.41, respectively.

Exercise 6.9

a. The Accept–Reject algorithm with a Gamma $\mathcal{G}(4, 7)$ candidate can be implemented as follows

```
# (C.) Jiazi Tang, 2009
g47=rgamma(5000,4,7)
u=runif(5000,max=dgamma(g47,4,7))
x=g47[u<dgamma(g47,4.3,6.2)]
par(mfrow=c(1,3),mar=c(4,4,1,1))
hist(x,freq=FALSE,xlab="",ylab="",col="wheat2",
main="Accept-Reject with Ga(4.7) proposal")
curve(dgamma(x,4.3,6.2),lwd=2,col="sienna",add=T)
```

The efficiency of the simulation is given by

```
> length(x)/5000
[1] 0.8374
```

b. The Metropolis-Hastings algorithm with a Gamma $\mathcal{G}(4,7)$ candidate can be implemented as follows

```
# (C.) Jiazi Tang, 2009
X=rep(0,5000)
X[1]=rgamma(1,4.3,6.2)
for (t in 2:5000){
    rho=(dgamma(X[t-1],4,7)*dgamma(g47[t],4.3,6.2))/
        (dgamma(g47[t],4,7)*dgamma(X[t-1],4.3,6.2))
    X[t]=X[t-1]+(g47[t]-X[t-1])*(runif(1)<rho)
    }
hist(X,freq=FALSE,xlab="",ylab="",col="wheat2",
main="Metropolis-Hastings with Ga(4,7) proposal")
curve(dgamma(x,4.3,6.2),lwd=2,col="sienna",add=T)
```

Its efficiency is

```
> length(unique(X))/5000
[1] 0.79
```

c. The Metropolis-Hastings algorithm with a Gamma $\mathcal{G}(5,6)$ candidate can be implemented as follows

```
# (C.) Jiazi Tang, 2009
g56=rgamma(5000,5,6)
X[1]=rgamma(1,4.3,6.2)
for (t in 2:5000){
   rho=(dgamma(X[t-1],5,6)*dgamma(g56[t],4.3,6.2))/
       (dgamma(g56[t],5,6)*dgamma(X[t-1],4.3,6.2))
   X[t]=X[t-1]+(g56[t]-X[t-1])*(runif(1)<rho)
   }
hist(X,freq=FALSE,xlab="",ylab="",col="wheat2",
main="Metropolis-Hastings with Ga(5,6) proposal")
curve(dgamma(x,4.3,6.2),lwd=2,col="sienna",add=T)
```

Its efficiency is

```
> length(unique(X))/5000
[1] 0.7678
```

which is therefore quite similar to the previous proposal.

Exercise 6.11

1.. Using the candidate given in Example 6.3 mean using the `Braking R` program of our package `mcsm`. In the earlier version, there is a missing link in the `R` function which must then be corrected by changing

```
data=read.table("BrakingData.txt",sep = "",header=T)
x=data[,1]
y=data[,2]
```

into

```
x=cars[,1]
y=cars[,2]
```

In addition, since the original `Braking` function does not return the simulated chains, a final line

```
list(a=b1hat,b=b2hat,c=b3hat,sig=s2hat)
```

must be added into the function.

2.. If we save the chains as `mcmc=Braking()` (note that we use 10^3 simulations instead of 500), the graphs assessing convergence can be plotted by

```
par(mfrow=c(3,3),mar=c(4,4,2,1))
plot(mcmc$a,type="l",xlab="",ylab="a");acf(mcmc$a)
hist(mcmc$a,prob=T,main="",yla="",xla="a",col="wheat2")
plot(mcmc$b,type="l",xlab="",ylab="b");acf(mcmc$b)
hist(mcmc$b,prob=T,main="",yla="",xla="b",col="wheat2")
plot(mcmc$c,type="l",xlab="",ylab="c");acf(mcmc$c)
hist(mcmc$c,prob=T,main="",yla="",xla="c",col="wheat2")
```

Autocorrelation graphs provided by `acf` show a strong correlation across iterations, while the raw plot of the sequences show poor acceptance rates. The histograms are clearly unstable as well. This 10^3 iterations do not appear to be sufficient in this case.

3.. Using

```
> quantile(mcmc$a,c(.025,.975))
     2.5%     97.5%
-6.462483 12.511916
```

and the same for b and c provides converging confidence intervals on the three parameters.

Exercise 6.13

Warning: There is a typo in question b in that the candidate must also be a double-exponential for α, since there is no reason for α to be positive...

1. The dataset `challenger` is provided with the `mcsm` package, thus available as

```
> library(mcsm)
> data(challenger)
```

Running a regular logistic regression is a simple call to `glm`:

```
> temper=challenger[,2]
> failur=challenger[,1]
> summary(glm(failur~temper, family = binomial))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0611  -0.7613  -0.3783   0.4524   2.2175

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  15.0429     7.3786   2.039   0.0415 *
temper       -0.2322     0.1082  -2.145   0.0320 *
---
Signif. codes:  0 "***" .001 "**" .01 "**" .05 "." .1 "" 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 28.267  on 22  degrees of freedom
Residual deviance: 20.315  on 21  degrees of freedom
AIC: 24.315
```

The MLE's and the associated covariance matrix are given by

```
> challe=summary(glm(failur~temper, family = binomial))
> beta=as.vector(challe$coef[,1])
> challe$cov.unscaled
            (Intercept)      temper
(Intercept)  54.4441826 -0.79638547
temper       -0.7963855  0.01171512
```

The result of this estimation can be checked by

```
plot(temper,failur,pch=19,col="red4",
xlab="temperatures",ylab="failures")
curve(1/(1+exp(-beta[1]-beta[2]*x)),add=TRUE,col="gold2",lwd=2)
```

and the curve shows a very clear impact of the temperature.

2. The Metropolis–Hastings resolution is based on the `challenge(mcsm)` function, using the same prior on the coefficients, $\alpha \sim \mathcal{N}(0, 25)$, $\beta \sim \mathcal{N}(0, 25/s_x^2)$, where s_x^2 is the empirical variance of the temperatures.

```
Nsim=10^4
x=temper
y=failur
sigmaa=5
sigmab=5/sd(x)

lpost=function(a,b){
  sum(y*(a+b*x)-log(1+exp(a+b*x)))+
  dnorm(a,sd=sigmaa,log=TRUE)+dnorm(b,sd=sigmab,log=TRUE)
   }

a=b=rep(0,Nsim)
a[1]=beta[1]
b[1]=beta[2]
#scale for the proposals
scala=sqrt(challe$cov.un[1,1])
scalb=sqrt(challe$cov.un[2,2])

for (t in 2:Nsim){
  propa=a[t-1]+sample(c(-1,1),1)*rexp(1)*scala
  if (log(runif(1))<lpost(propa,b[t-1])-
    lpost(a[t-1],b[t-1])) a[t]=propa
  else a[t]=a[t-1]
  propb=b[t-1]+sample(c(-1,1),1)*rexp(1)*scalb
  if (log(runif(1))<lpost(a[t],propb)-
    lpost(a[t],b[t-1])) b[t]=propb
  else b[t]=b[t-1]
  }
```

The acceptance rate is low

```
> length(unique(a))/Nsim
[1] 0.1031
> length(unique(b))/Nsim
[1] 0.1006
```

but still acceptable.

3. Exploring the output can be done via graphs as follows

```
par(mfrow=c(3,3),mar=c(4,4,2,1))
plot(a,type="l",xlab="iterations",ylab=expression(alpha))
hist(a,prob=TRUE,col="wheat2",xlab=expression(alpha),main="")
acf(a,ylab=expression(alpha))
```

```
plot(b,type="l",xlab="iterations",ylab=expression(beta))
hist(b,prob=TRUE,col="wheat2",xlab=expression(beta),main="")
acf(b,ylab=expression(beta))
plot(a,b,type="l",xlab=expression(alpha),ylab=expression(beta))
plot(temper,failur,pch=19,col="red4",
   xlab="temperatures",ylab="failures")
for (t in seq(100,Nsim,le=100)) curve(1/(1+exp(-a[t]-b[t]*x)),
  add=TRUE,col="grey65",lwd=2)
curve(1/(1+exp(-mean(a)-mean(b)*x)),add=TRUE,col="gold2",lwd=2.5)
postal=rep(0,1000);i=1
for (t in seq(100,Nsim,le=1000)){ postal[i]=lpost(a[t],b[t]);i=i+1}
plot(seq(100,Nsim,le=1000),postal,type="l",
  xlab="iterations",ylab="log-posterior")
abline(h=lpost(a[1],b[1]),col="sienna",lty=2)
```

which shows a slow convergence of the algorithm (see the `acf` graphs on Figure 6.1!)

4. The predictions of failure are given by

```
> mean(1/(1+exp(-a-b*50)))
[1] 0.6898612
> mean(1/(1+exp(-a-b*60)))
[1] 0.4892585
> mean(1/(1+exp(-a-b*70)))
[1] 0.265691
```

Exercise 6.15

Warning: There is a typo in question c, which should involve $\mathcal{N}(0,\omega)$ candidates instead of $\mathcal{L}(0,\omega)$...

a. An `R` program to produce the three evaluations is

```
# (C.) Thomas Bredillet, 2009
Nsim=5000
A=B=runif(Nsim)
alpha=1;alpha2=3
last=A[1]
a=0;b=1
cand=ifelse(runif(Nsim)>0.5,1,-1) * rexp(Nsim)/alpha
for (i in 1:Nsim){
        rate=(dnorm(cand[i],a,b^2)/dnorm(last,a,b^2))/
       (exp(-alpha*abs(cand[i]))/exp(-alpha*abs(last)))
        if (runif(1)<rate) last=cand[i]
        A[i]=last
        }
cand=ifelse(runif(Nsim)>0.5,1,-1) * rexp(Nsim)/alpha2
```

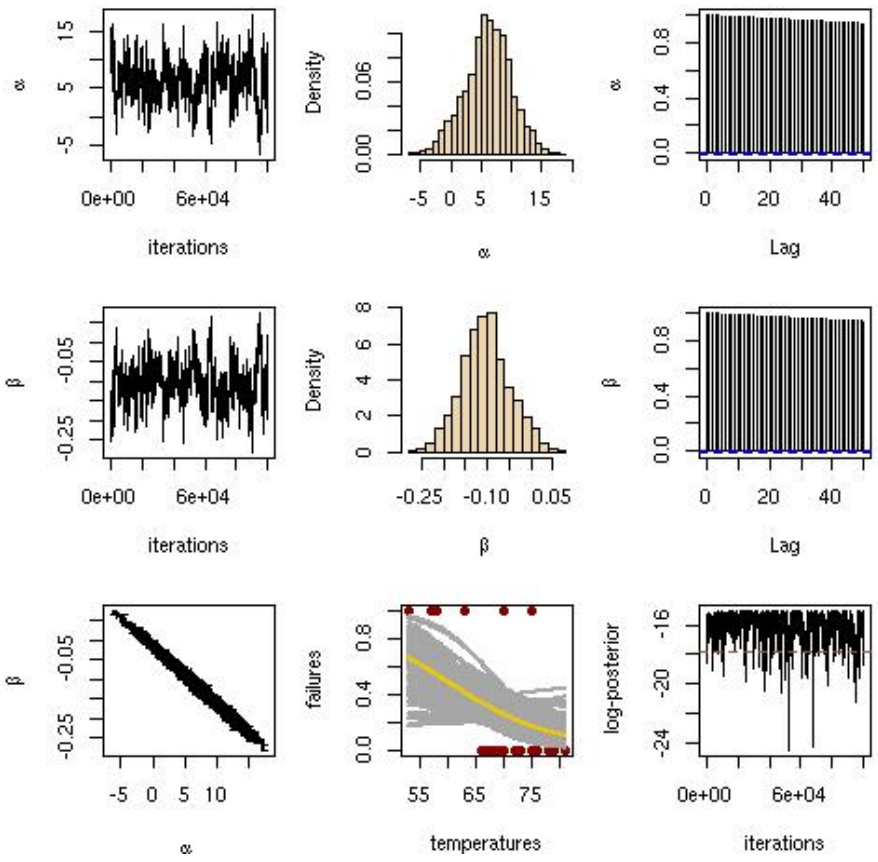

Fig. 6.1. Graphical checks of the convergence of the Metropolis–Hastings algorithm associated with the challenger dataset and a logistic regression model.

```
for (i in 1:Nsim) {
       rate=(dnorm(cand[i],a,b^2)/dnorm(last,a,b^2))/
       (exp(-alpha2*abs(cand[i]))/exp(-alpha2*abs(last)))
       if (runif(1)<rate) last=cand[i]
       B[i]=last
       }
par (mfrow=c(1,3),mar=c(4,4,2,1))
est1=cumsum(A)/(1:Nsim)
est2=cumsum(B)/(1:Nsim)
plot(est1,type="l",xlab="iterations",ylab="",lwd=2)
lines(est2,lwd="2",col="gold2")
acf(A)
acf(B)
```

b. The acceptance rate is given by `length(unique(B))/Nsim`, equal to 0.49 in the current simulation. A plot of the acceptance rates can be done via the `R` program

```
alf=seq(1,10,le=50)
cand0=ifelse(runif(Nsim)>0.5,1,-1) * rexp(Nsim)
acce=rep(0,50)
for (j in 1:50){
  cand=cand0/alf[j]
  last=A[1]
  for (i in 2:Nsim){
    rate=(dnorm(cand[i],a,b^2)/dnorm(last,a,b^2))/
    (exp(-alf[j]*abs(cand[i]))/exp(-alf[j]*abs(last)))
    if (runif(1)<rate) last=cand[i]
    A[i]=last
    }
  acce[j]=length(unique(A))/Nsim
  }
par(mfrow=c(1,3),mar=c(4,4,2,1))
plot(alf,acce,xlab="",ylab="",type="l",main="Laplace iid")
```

The highest acceptance rate is obtained for the smallest value of α.

c. The equivalent of the above `R` program is

```
ome=sqrt(seq(.01,10,le=50))
cand0=rnorm(Nsim)
acce=rep(0,50)
for (j in 1:50){
  cand=cand0*ome[j]
  last=A[1]
  for (i in 2:Nsim){
    rate=(dnorm(cand[i],a,b^2)/dnorm(last,a,b^2))/
    (dnorm(cand[i],sd=ome[j])/dnorm(last,sd=ome[j]))
    if (runif(1)<rate) last=cand[i]
    A[i]=last
    }
  acce[j]=length(unique(A))/Nsim
  }
plot(ome^2,acce,xlab="",ylab="",type="l",main="Normal iid")
```

The highest acceptance rate is (unsurprisingly) obtained for ω close to 1.

d. The equivalent of the above `R` program is

```
alf=seq(.1,10,le=50)
cand0=ifelse(runif(Nsim)>0.5,1,-1) * rexp(Nsim)
acce=rep(0,50)
for (j in 1:50){
  eps=cand0/alf[j]
```

```
    last=A[1]
    for (i in 2:Nsim){
      cand[i]=last+eps[i]
      rate=dnorm(cand[i],a,b^2)/dnorm(last,a,b^2)
      if (runif(1)<rate) last=cand[i]
      A[i]=last
      }
    acce[j]=length(unique(A))/Nsim
    }
  plot(alf,acce,xlab="",ylab="",type="l",main="Laplace random walk")
```

Unsurprisingly, as α increases, so does the acceptance rate. However, given that this is a random walk proposal, higher acceptance rates do not mean better performances (see Section 6.5).

7

Gibbs Samplers

Exercise 7.1

The density g_t of (X_t, Y_t) in Algorithm 7 is decomposed as

$$g_t(X_t, Y_t|X_{t-1}, \dots X_0, Y_{t-1}, \dots Y_0) = g_{t,X|Y}(X_t|Y_t, X_{t-1}, \dots X_0, Y_{t-1}, \dots Y_0) \times g_{t,Y}(Y_t|X_{t-1}, \dots X_0, Y_{t-1}, \dots Y_0)$$

with

$$g_{t,Y}(Y_t|X_{t-1}, \dots X_0, Y_{t-1}, \dots Y_0) = f_{Y|X}(Y_t|X_{t-1})$$

which only depends on $X_{t-1}, \dots X_0, Y_{t-1}, \dots Y_0$ through X_{t-1}, according to Step 1. of Algorithm 7. Moreover,

$$g_{t,X|Y}(X_t|Y_t, X_{t-1}, \dots X_0, Y_{t-1}, \dots Y_0) = f_{X|Y}(X_t|Y_t)$$

only depends on $X_{t-2}, \dots X_0, Y_t, \dots Y_0$ through Y_t. Therefore,

$$g_t(X_t, Y_t|X_{t-1}, \dots X_0, Y_{t-1}, \dots Y_0) = g_t(X_t, Y_t|X_{t-1})\,,$$

which shows this is truly an homogeneous Markov chain.

Exercise 7.5

a. The (normal) full conditionals are defined in Example 7.4. An `R` program that implements this Gibbs sampler is

```
# (C.) Anne Sabourin, 2009
T=500 ;p=5 ;r=0.25
X=cur=rnorm(p)
for (t in 1 :T){
  for (j in 1 :p){
    m=sum(cur[-j])/(p-1)
    cur[j]=rnorm(1,(p-1)*r*m/(1+(p-2)*r),
```

```
        sqrt((1+(p-2)*r-(p-1)*r^2)/(1+(p-2)*r)))
    }
  X=cbind(X,cur)
  }
par(mfrow=c(1,5))
for (i in 1:p){
   hist(X[i,],prob=TRUE,col="wheat2",xlab="",main="")
   curve(dnorm(x),add=TRUE,col="sienna",lwd=2)}
```

b. Using instead

```
J=matrix(1,ncol=5,nrow=5)
I=diag(c(1,1,1,1,1))
s=(1-r)*I+r*J
rmnorm(500,s)
```

and checking the duration by `system.time` shows `rmnorm` is about five times faster (and exact!).

c. If we consider the constraint

$$\sum_{i=1}^{m} x_i^2 \le \sum_{i=m+1}^{p} x_i^2$$

it imposes a truncated normal full conditional on *all* components. Indeed, for $1 \le i \le m$,

$$x_i^2 \le \sum_{j=m+1}^{p} x_j^2 - \sum_{j=1, j\neq i}^{m} x_j^2 \,,$$

while, for $i > m$,

$$x_i^2 \ge \sum_{j=m+1, j\neq i}^{p} x_j^2 - \sum_{j=1}^{m} x_j^2 \,.$$

Note that the upper bound on x_i^2 when $i \le m$ *cannot be negative* if we start the Markov chain under the constraint. The `cur[j]=rnorm(...` line in the above `R` program thus needs to be modified into a truncated normal distribution. An alternative is to use a hybrid solution (see Section 7.6.3 for the validation): we keep generating the x_i's from the same plain normal full conditionals as before and we only change the components for which the constraint remains valid, i.e.

```
for (j in 1:m){
  mea=sum(cur[-j])/(p-1)
  prop=rnorm(1,(p-1)*r*mea/(1+(p-2)*r),
     sqrt((1+(p-2)*r-(p-1)*r^2)/(1+(p-2)*r)))
  if (sum(cur[(1:m)[-j]]^2+prop^2)<sum(cur[(m+1):p]^2))
    cur[j]=prop
  }
```

```
for (j in (m+1):p){
  mea=sum(cur[-j])/(p-1)
  prop=rnorm(1,(p-1)*r*mea/(1+(p-2)*r),
     sqrt((1+(p-2)*r-(p-1)*r^2)/(1+(p-2)*r)))
  if (sum(cur[(1:m)]^2)<sum(cur[((m+1):p)[-j]]^2+prop^2))
    cur[j]=prop
  }
```

Comparing the histograms with the normal $\mathcal{N}(0,1)$ shows that the marginals are no longer normal.

Exercise 7.7

Warning: There is a typo in Example 7.6, namely that the likelihood function involves $\Phi(\theta-a)^{n-m}$ in front of the product of normal densities... For coherence with Examples 5.13 and 5.14, in both Example 7.6 and Exercise 7.7, x should be written y, z z, $\bar{x}$ $\bar{y}$ and x_i y_i.

a. The complete data likelihood is associated with the distribution of the uncensored data

$$(y_1, \ldots, y_m, z_{m+1}, \ldots, z_n)\,,$$

which constitutes an iid sample of size n. In that case, a sufficient statistics is $\{m\bar{y} + (n - m(\bar{z})\}/n$, which is distributed as $\mathcal{N}(\theta, 1/n)$, i.e. associated with the likelihood

$$\exp\left\{\frac{-n}{2}\left(\frac{m\bar{x}+(n-m)\bar{z}}{n}-\theta\right)^2\right\}/\sqrt{n}\,.$$

In this sense, the likelihood is proportional to the density of $\theta \sim \mathcal{N}(\{m\bar{x}+(n-m)\bar{z}\}/n, 1/n)$. (We acknowledge a certain vagueness in the wording of this question!)

b. The full `R` code for the Gibbs sampler is

```
xdata=c(3.64,2.78,2.91,2.85,2.54,2.62,3.16,2.21,4.05,2.19,
2.97,4.32,3.56,3.39,3.59,4.13,4.21,1.68,3.88,4.33)
m=length(xdata)
n=30;a=3.5            #1/3 missing data
nsim=10^4
xbar=mean(xdata)
that=array(xbar,dim=c(nsim,1))
zbar=array(a,dim=c(nsim,1))
for (i in 2:nsim){
   temp=runif(n-m,min=pnorm(a,mean=that[i-1],sd=1),max=1)
   zbar[i]=mean(qnorm(temp,mean=that[i-1],sd=1))
   that[i]=rnorm(1,mean=(m*xbar+(n-m)*zbar[i])/n,
                 sd=sqrt(1/n))
```

```
   }
par(mfrow=c(1,2),mar=c(5,5,2,1))
hist(that[500:nsim],col="grey",breaks=25,
xlab=expression(theta),main="",freq=FALSE)
curve(dnorm(x,mean(that),sd=sd(that)),add=T,lwd=2)
hist(zbar[500:nsim],col="grey",breaks=25
main="",xlab= expression(bar(Z)),freq=FALSE)
curve(dnorm(x,mean(zbar),sd=sd(zbar)),add=T,lwd=2)
```

(We added the normal density curves to check how close to a normal distribution the posteriors are.)

Exercise 7.9

a. Given the information provided in Table 7.1, since we can reasonably assume independence between the individuals, the distribution of the blood groups is a multinomial distribution whose density is clearly proportional to

$$(p_A^2 + 2p_Ap_O)^{n_A}(p_B^2 + 2p_Bp_O)^{n_B}(p_Ap_B)^{n_{AB}}(p_O^2)^{n_O} .$$

the proportionality coefficient being the multinomial coefficient

$$\binom{n}{n_A\ n_B\ n_{AB}\ n_O} .$$

b. If we break n_A into Z_A individuals with genotype `AA` and $n_A - Z_A$ with genotype `AO`, and similarly, n_B into Z_B individuals with genotype `BB` and $n_B - Z_B$ with genotype `BO`, the complete data likelihood corresponds to the extended multinomial model with likelihood proportional to

$$(p_A^2)^{Z_A}(2p_Ap_O)^{n_A-Z_A}(p_B^2)^{Z_B}(2p_Bp_O)^{n_B-Z_B}(p_Ap_B)^{n_{AB}}(p_O^2)^{n_O} .$$

c. The Gibbs sampler we used to estimate this model is

```
nsim=5000;nA=186;nB=38;nAB=13;nO=284;
pA=array(.25,dim=c(nsim,1));pB=array(.05,dim=c(nsim,1));
for (i in 2:nsim){
  pO=1-pA[i-1]-pB[i-1]
  ZA=rbinom(1,nA,pA[i-1]^2/(pA[i-1]^2+2*pA[i-1]*pO));
  ZB=rbinom(1,nB,pB[i-1]^2/(pB[i-1]^2+2*pB[i-1]*pO));
  temp=rdirichlet(1,c(nA+nAB+ZA+1,nB+nAB+ZB+1,
                  nA-ZA+nB-ZB+2*nO+1));
  pA[i]=temp[1];pB[i]=temp[2];
  }
par(mfrow=c(1,3),mar=c(4,4,2,1))
hist(pA,main=expression(p[A]),freq=F,col="wheat2")
hist(pB,main=expression(p[B]),freq=F,col="wheat2")
hist(1-pA-pB,,main=expression(p[O]),freq=F,col="wheat2")
```

It uses the Dirichlet generator `rdirichlet` found in the `mcsm` package.

Exercise 7.11

a. For the target density $f_X(x) = \frac{1}{2}e^{-\sqrt{x}}$, a slice sampling algorithm is based on the full conditionals
 a) $U^{(t+1)} \sim \mathcal{U}_{[0, f_X(x^{(t)})]}$
 b) $X^{(t+1)} \sim \mathcal{U}_{A^{(t+1)}}$ with $A^{(t+1)} = \{x, f(x) \geq u^{(t+1)}\}$
 Therefore, $U|x \sim \mathcal{U}(0, \frac{1}{2}e^{-\sqrt{x}})$ and, since $A = \{x, \frac{1}{2}e^{-\sqrt{x}} \geq u\}$, i.e. $A = \{x, 0 \leq x \leq \log(2u)\}$, owe also deduce that $X|u \sim \mathcal{U}(0, (\log(2u))^2)$. The corresponding `R` code is

```
T=5000
f=function(x){
   1/2*exp(-sqrt(x))}
X=c(runif(1)) ;U=c(runif(1))
for (t in 1:T){
  U=c(U,runif(1,0,f(X[t])))
  X=c(X,runif(1,0,(log(2*U[t+1]))^2))
  }
par(mfrow=c(1,2))
hist(X,prob=TRUE,col="wheat2",xlab="",main="")
acf(X)
```

b. If we define $Y = \sqrt{X}$, then

$$\begin{aligned} P(Y \leq y) &= P(X \leq y^2) \\ &= \int_0^y \frac{1}{2} e^{-\sqrt{x}} d \, \mathrm{d}x \end{aligned}$$

When we differentiate against y, we get the density

$$f_Y(y) = y \exp(-y)$$

which implies that $Y \sim \mathcal{G}a(2, 1)$. Simulating X then follows from $X = Y^2$. This method is obviously faster and more accurate since the sample points are then independent.

Exercise 7.13

a. The linear combinations $X + Y$ and $X - Y$ also are normal with null expectation and with variances $2(1 + \rho)$ and $2(1 - \rho)$, respectively. The vector $(X + Y, X - Y)$ itself is equally normal. Moreover,

$$\mathrm{cov}(X + Y, X - Y) = \mathbb{E}((X + Y)(X - Y)) = \mathbb{E}(X^2 - Y^2) = 1 - 1 = 0$$

implies that $X + Y$ and $X - Y$ are independent.

b. If, instead,

$$(X,Y) \sim \mathcal{N}(0, \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix})$$

then $\sigma_x^2 \neq \sigma_y^2$ implies that $(X+Y)$ and $(X-Y)$ are dependent since $\mathbb{E}((X+Y)(X-Y)) = \sigma_x^2 - \sigma_y^2$. In this case, $X|Y=y \sim \mathcal{N}(\rho\frac{\sigma_x}{\sigma_y}y, \sigma_x^2(1-\rho^2))$. We can simulate (X,Y) by the following Gibbs algorithm

```
T=5000;r=0.8;sx=50;sy=100
X=rnorm(1);Y=rnorm(1)
for (t in 1:T){
    Yn=rnorm(1,r*sqrt(sy/sx)*X[t],sqrt(sy*(1-r^2)))
    Xn=rnorm(1,r*sqrt(sx/sy)*Yn,sqrt(sx*(1-r^2)))
    X=c(X,Xn)
    Y=c(Y,Yn)
    }
par(mfrow=c(3,2),oma=c(0,0,5,0))
hist(X,prob=TRUE,main="",col="wheat2")
hist(Y,prob=TRUE,main="",col="wheat2")
acf(X);acf(Y);plot(X,Y);plot(X+Y,X-Y)
```

c. If $\sigma_x \neq \sigma_y$, let us find $a \in \mathbb{R}$ such that $X+aY$ and Y are independent. We have $\mathbb{E}[(X+aY)(Y)] = 0$ if and only if $\rho\sigma_x\sigma_y + a\sigma_y^2 = 0$, i.e. $a = -\rho\sigma_x/\sigma_y$. Therefore, $X - \rho\sigma_x/\sigma_y Y$ and Y are independent.

Exercise 7.15

a. The likelihood function naturally involves the tail of the Poisson distribution for those observations larger than 4. The full conditional distributions of the observations larger than 4 are obviously truncated Poisson distributions and the full conditional distribution of the parameter is the Gamma distribution associated with a standard Poisson sample. Hence the Gibbs sampler.

b. The `R` code we used to produce Figure 7.13 is

```
nsim=10^3
lam=RB=rep(313/360,nsim)
z=rep(0,13)
for (j in 2:nsim){
  top=round(lam[j -1]+6*sqrt(lam[j -1]))
  prob=dpois(c(4:top),lam[j -1])
  cprob=cumsum(prob/sum(prob))
  for(i in 1:13) z[i] = 4+sum(cprob<runif(1))
  RB[j]=(313+sum(z))/360
  lam[j]=rgamma(1,360*RB[j],scale=1/360);
  }
par(mfrow=c(1,3),mar=c(4,4,2,1))
```

```
hist(lam,col="grey",breaks=25,xlab="",
     main="Empirical average")
plot(cumsum(lam)/1:nsim,ylim=c(1,1.05),type="l",
     lwd=1.5,ylab="")
lines(cumsum(RB)/1:nsim,col="sienna",lwd=1.5)
hist(RB,col="sienna",breaks=62,xlab="",
     main="Rao-Blackwell",xlim=c(1,1.05))
```

c. When checking the execution time of both programs with `system.time`, the first one is almost ten times faster. And completely correct. A natural way to pick `prob` is

```
> qpois(.9999,lam[j-1])
[1] 6
```

Exercise 7.17

a. The `R` program that produced Figure 7.14 is

```
nsim=10^3
X=Y=rep(0,nsim)
X[1]=rexp(1)            #initialize the chain
Y[1]=rexp(1)            #initialize the chain
for(i in 2:nsim){
        X[i]=rexp(1,rate=Y[i-1])
        Y[i]=rexp(1,rate=X[i])
        }
st=0.1*nsim
par(mfrow=c(1,2),mar=c(4,4,2,1))
hist(X,col="grey",breaks=25,xlab="",main="")
plot(cumsum(X)[(st+1):nsim]/(1:(nsim-st)),type="l",ylab="")
```

b. Using the Hammersley–Clifford Theorem *per se* means using $f(y|x)/f(x|y) = x/y$ which is *not integrable*. If we omit this major problem, we have

$$f(x,y) = \frac{x\,\exp\{-xy\}}{x\displaystyle\int \frac{\mathrm{d}y}{y}} \propto \exp\{-xy\}$$

(except that the proportionality term is infinity!).

c. If we constrain both conditionals to $(0, B)$, the Hammersley–Clifford Theorem gives

$$f(x,y) = \frac{\exp\{-xy\}/(1-e^{-xB})}{\displaystyle\int \frac{1-e^{-yB}}{y(1-e^{-xB})}\,\mathrm{d}y}$$
$$= \frac{\exp\{-xy\}}{\displaystyle\int \frac{1-e^{-yB}}{y}\,\mathrm{d}y}$$
$$\propto \exp\{-xy\}\,,$$

since the conditional exponential distributions are truncated. This joint distribution is then well-defined on $(0,B)^2$. A Gibbs sampler simulating from this joint distribution is for instance

```
B=10
X=Y=rep(0,nsim)
X[1]=rexp(1)            #initialize the chain
Y[1]=rexp(1)            #initialize the chain
for(i in 2:nsim){ #inversion method
        X[i]=-log(1-runif(1)*(1-exp(-B*Y[i-1])))/Y[i-1]
        Y[i]=-log(1-runif(1)*(1-exp(-B*X[i])))/X[i]
        }
st=0.1*nsim
marge=function(x){ (1-exp(-B*x))/x}
nmarge=function(x){
       marge(x)/integrate(marge,low=0,up=B)$val}
par(mfrow=c(1,2),mar=c(4,4,2,1))
hist(X,col="wheat2",breaks=25,xlab="",main="",prob=TRUE)
curve(nmarge,add=T,lwd=2,col="sienna")
plot(cumsum(X)[(st+1):nsim]/c(1:(nsim-st)),type="l",
     lwd=1.5,ylab="")
```

where the simulation of the truncated exponential is done by inverting the cdf (and where the true marginal is represented against the histogram).

Exercise 7.19

Let us define

$$f(x) = \frac{b^a x^{a-1} e^{-bx}}{\Gamma(a)}\,,$$
$$g(x) = \frac{1}{x} = y\,,$$

then we have

$$\begin{aligned} f_Y(y) &= f_X\left(g^{-1}(y)\right) \mid \frac{d}{dy} g^{-1}(y) \mid \\ &= \frac{b^a}{\Gamma(a)} (1/y)^{a-1} \exp(-b/y) \frac{1}{y^2} \\ &= \frac{b^a}{\Gamma(a)} (1/y)^{a+1} \exp(-b/y) \ , \end{aligned}$$

which is the $\mathcal{IG}(a, b)$ density.

Exercise 7.21

Warning: The function `rtnorm` requires a predefined `sigma` that should be part of the arguments, as in
`rtnorm=function(n=1,mu=0,lo=-Inf,up=Inf,sigma=1).`

Since the `rtnorm` function is exact (within the precision of the `qnorm` and `pnorm` functions, the implementation in `R` is straightforward:

```
h1=rtnorm(10^4,lo=-1,up=1)
h2=rtnorm(10^4,up=1)
h3=rtnorm(10^4,lo=3)
par(mfrow=c(1,3),mar=c(4,4,2,1))
hist(h1,freq=FALSE,xlab="x",xlim=c(-1,1),col="wheat2")
dnormt=function(x){ dnorm(x)/(pnorm(1)-pnorm(-1))}
curve(dnormt,add=T,col="sienna")
hist(h2,freq=FALSE,xlab="x",xlim=c(-4,1),col="wheat2")
dnormt=function(x){ dnorm(x)/pnorm(1)}
curve(dnormt,add=T,col="sienna")
hist(h3,freq=FALSE,xlab="x",xlim=c(3,5),col="wheat2")
dnormt=function(x){ dnorm(x)/pnorm(-3)}
curve(dnormt,add=T,col="sienna")
```

Exercise 7.23

a. Since $(j = 1, 2)$

$$(1-\theta_1-\theta_2)^{x_5+\alpha_3-1} = \sum_{i=0}^{x_5+\alpha_3-1} \binom{x_5+\alpha_3-1}{i} (1-\theta_j)^i \theta_{3-j}^{x_5+\alpha_3-1-i} \ ,$$

when α_3 is an integer, it is clearly possible to express $\pi(\theta_1, \theta_2|x)$ as a sum of terms that are products of a polynomial function of θ_1 and of a polynomial function of θ_2. It is therefore straightforward to integrate those terms in either θ_1 or θ_2.

b. For the same reason as above, rewriting $\pi(\theta_1, \theta_2|x)$ as a density in (θ_1, ξ) leads to a product of polynomials in θ_1, all of which can be expanded and integrated in θ_1, producing in the end a sum of functions of the form

$$\xi^{\delta}/(1+\xi)^{x_1+x_2+x_5+\alpha_1+\alpha_3-2}\,,$$

namely a mixture of F densities.

c. The Gibbs sampler based on (7.9) is available in the `mcsm` package.

Exercise 7.25

Warning: There is a typo in Example 7.3, `sigma` should be defined as `sigma2` and `sigma2{1}` should be `sigma2[1]`...

a. In Example 7.2, since $\theta|x \sim \mathcal{B}e(x+a, n-x+b)$, we have clearly $\mathbb{E}[\theta|x] = (x+a)/(n+a+b)$ (with a missing parenthesis). The comparison between the empirical average and of the Rao–Blackwellization version is of the form

```
plot(cumsum(T)/(1:Nsim),type="l",col="grey50",
     xlab="iterations",ylab="",main="Example 7.2")
lines(cumsum((X+a))/((1:Nsim)*(n+a+b)),col="sienna")
```

All comparisons are gathered in Figure 7.1.

b. In Example 7.3, equation (7.4) defines two standard distributions as full conditionals. Since $\pi(\theta|\mathbf{x}, \sigma^2)$ is a normal distribution with mean and variance provided two lines below, we obviously have

$$\mathbb{E}[\theta|\mathbf{x}, \sigma^2] = \frac{\sigma^2}{\sigma^2 + n\tau^2}\,\theta_0 + \frac{n\tau^2}{\sigma^2 + n\tau^2}\,\bar{x}$$

The modification in the `R` program follows

```
plot(cumsum(theta)/(1:Nsim),type="l",col="grey50",
     xlab="iterations",ylab="",main="Example 7.3")
ylab="",main="Example 7.3")
lines(cumsum(B*theta0+(1-B)*xbar)/(1:Nsim)),col="sienna")
```

c. The full conditionals of Example 7.5 given in Equation (7.7) are more numerous but similarly standard, therefore

$$\mathbb{E}[\theta_i|\bar{X}_i, \sigma^2] = \frac{\sigma^2}{\sigma^2 + n_i\tau^2}\mu + \frac{n_i\tau^2}{\sigma^2 + n_i\tau^2}\bar{X}_i$$

follows from this decomposition, with the `R` lines added to the `mcsm` `randomeff` function

```
plot(cumsum(theta1)/(1:nsim),type="l",col="grey50",
     xlab="iterations",ylab="",main="Example 7.5")
lines(cumsum((mu*sigma2+n1*tau2*x1bar)/(sigma2+n1*tau2))/
      (1:nsim)),col="sienna")
```

d. In Example 7.6, the complete-data model is a standard normal model with variance one, hence $\mathbb{E}[\theta|x,z] = \dfrac{m\bar{x}+(n-m)\bar{z}}{n}$. The additional lines in the `R` code are

```
plot(cumsum(that)/(1:Nsim),type="l",col="grey50",
     xlab="iterations",ylab="",main="Example 7.6")
lines(cumsum((m/n)*xbar+(1-m/n)*zbar)/(1:Nsim)),
      col="sienna")
```

e. In Example 7.12, the full conditional on λ, $\lambda_i|\beta,t_i,x_i \sim \mathcal{G}(x_i+\alpha,t_i+\beta)$ and hence $\mathbb{E}[\lambda_i|\beta,t_i,x_i] = (x_i+\alpha)/(t_i+\beta)$. The corresponding addition in the `R` code is

```
plot(cumsum(lambda[,1])/(1:Nsim),type="l",col="grey50",
     xlab="iterations",ylab="",main="Example 7.12")
lines(cumsum((xdata[1]+alpha)/(Time[1]+beta))/(1:Nsim)),
      col="sienna")
```

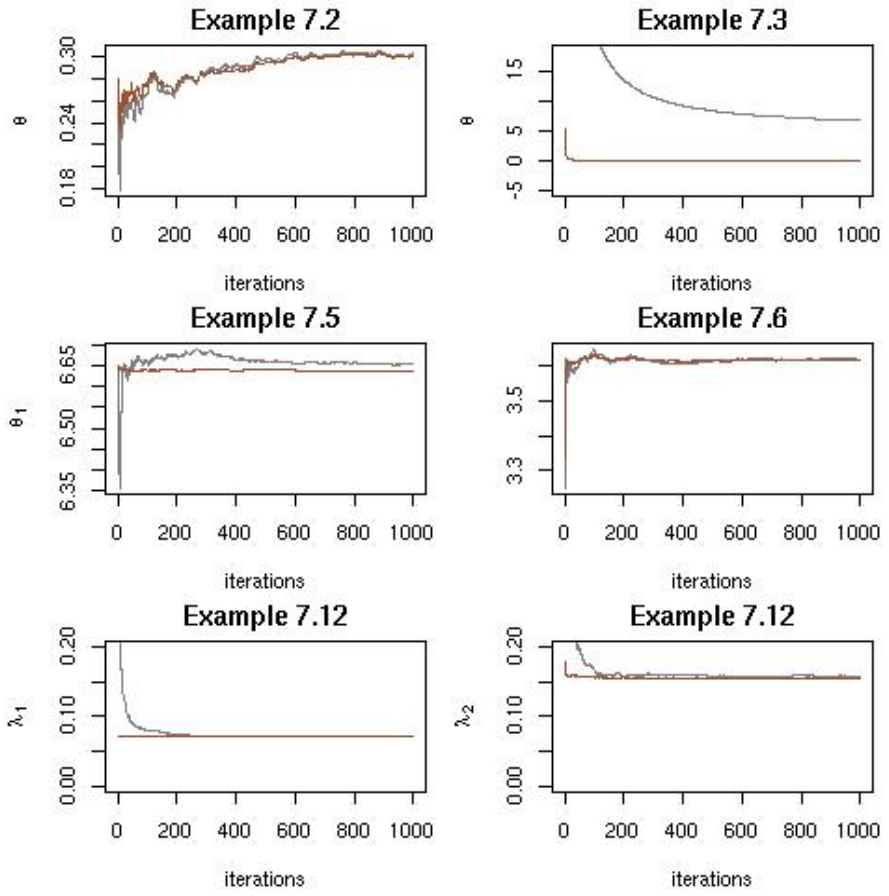

Fig. 7.1. Comparison of the convergences of the plain average with its Rao-Blackwellized counterpart for five different examples. The Rao-Blackwellized is plotted in sienna red and is always more stable than the original version.

8

Convergence Monitoring for MCMC Algorithms

Exercise 8.1

Warning: Strictly speaking, we need to assume that the Markov chain $(x^{(t)})$ has a finite variance for the h transform, since the assumption that $\mathbb{E}_f[h^2(X)]$ exists is not sufficient (see Meyn and Tweedie, 1993.
This result was established by MacEachern and Berliner (1994). We have the proof detailed as Lemma 12.2 in Robert and Casella (2004) (with the same additional assumption on the convergence of the Markov chain missing!).

Define $\delta_k^1, \ldots, \delta_k^{k-1}$ as the shifted versions of $\delta_k = \delta_k^0$; that is,

$$\delta_k^i = \frac{1}{T} \sum_{t=1}^{T} h(\theta^{(tk-i)}), \qquad i = 0, 1, \ldots, k-1 \; .$$

The estimator δ_1 can then be written as $\delta_1 = \frac{1}{k} \sum_{i=0}^{k-1} \delta_k^i$, and hence

$$\begin{aligned} \text{var}(\delta_1) &= \text{var}\left(\frac{1}{k} \sum_{i=0}^{k-1} \delta_k^i \right) \\ &= \text{var}(\delta_k^0)/k + \sum_{i \neq j} \text{cov}(\delta_k^i, \delta_k^j)/k^2 \\ &\leq \text{var}(\delta_k^0)/k + \sum_{i \neq j} \text{var}(\delta_k^0)/k^2 \\ &= \text{var}(\delta_k) \; , \end{aligned}$$

where the inequality follows from the Cauchy–Schwarz inequality

$$|\text{cov}(\delta_k^i, \delta_k^j)| \leq \text{var}(\delta_k^0).$$

Exercise 8.3

This is a direct application of the Ergodic Theorem (see Section 6.2). If the chain $(x^{(t)})$ is ergodic, then the empirical average above converges (almost surely) to $\mathbb{E}_f[\varphi(X)/\tilde{f}(X)] = 1/C$. This assumes that the support of φ is *small enough* (see Exercise 4.13). For the variance of the estimator to be finite, a necessary condition is that

$$\mathbb{E}_f[\varphi(X)/\tilde{f}(X)] \propto \int \frac{\varphi^2(x)}{f(x)}\,\mathrm{d}x < \infty\,.$$

As in Exercise 8.1, we need to assume that the convergence of the Markov chain is regular enough to ensure a finite variance.

Exercise 8.5

The modified `R` program using bootstrap is

```
ranoo=matrix(0,ncol=2,nrow=25)
for (j in 1:25){
 batch=matrix(sample(beta,100*Ts[j],rep=TRUE),ncol=100)
 sigmoo=2*sd(apply(batch,2,mean))
 ranoo[j,]=mean(beta[1:Ts[j]])+c(-sigmoo,+sigmoo)
 }
polygon(c(Ts,rev(Ts)),c(ranoo[,1],rev(ranoo[,2])),col="grey")
lines(cumsum(beta)/(1:T),col="sienna",lwd=2)
```

and the output of the comparison is provided in Figure 8.1.

Exercise 8.7

Warning: Example 8.9 contains several typos, namely $Y_k \sim \mathcal{N}(\theta_i, \sigma^2)$ instead of $Y_i \sim \mathcal{N}(\theta_i, \sigma^2)$, the μ_i's being also iid normal **instead of** the θ_i's being also iid normal**...**

Warning: Exercise 8.7 also contains a typo in that the posterior distribution on μ cannot be obtained in a closed form. It should read

Show that the posterior distribution on α in Example 8.9 can be obtained in a closed form.

Since

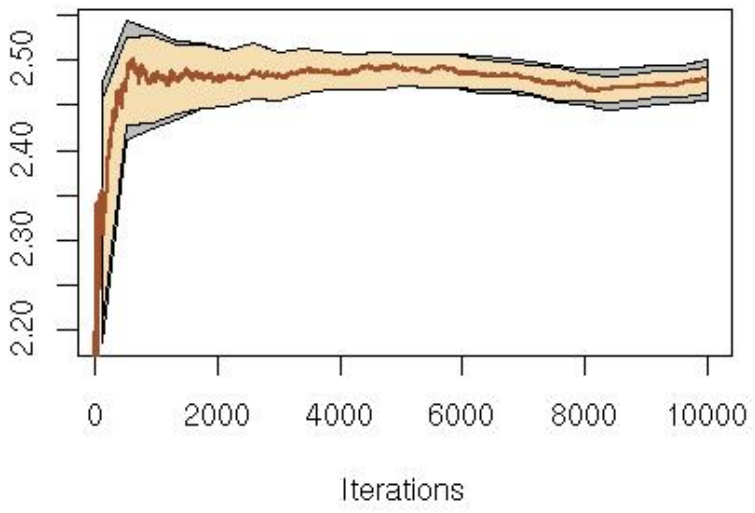

Fig. 8.1. Comparison of two evaluations of the variance of the MCMC estimate of the mean of β for the pump failure model of Example 8.6.

$$\begin{aligned}\theta|\mathbf{y},\mu,\alpha &\sim \pi(\theta|\mathbf{y},\mu,\alpha)\\ &\propto \alpha^{-9}\exp\frac{-1}{2}\left\{\sum_{i=1}^{18}\left[\sigma^{-2}(y_i-\theta_i)^2+\alpha^{-1}(\theta_i-\mu)^2\right]\right\}\\ &\propto \exp\frac{-1}{2}\left(\sum_{i=1}^{18}\left\{(\sigma^{-2}+\alpha^{-1})\left[\theta_i-(\sigma^{-2}+\alpha^{-1})^{-1}(\sigma^{-2}y_i+\alpha^{-1}\mu)\right]^2\right.\right.\\ &\quad\left.\left.+(\alpha+\sigma^2)^{-1}\sum_{i=1}^{18}(y_i-\mu)^2\right\}\right)\end{aligned}$$

(which is also a direct consequence of the marginalization $Y_i \sim \mathcal{N}(\mu, \alpha+\sigma^2)$), we have

$$\begin{aligned}\pi(\alpha,\mu|\mathbf{y}) &\propto \frac{\alpha^{-3}}{(\alpha+\sigma^2)^9}\,\exp\left\{-\frac{1}{2(\alpha+\sigma^2)}\sum_{i=1}^{18}(y_i-\mu)^2-\frac{\mu^2}{2}-\frac{2}{\alpha}\right\}\\ &\propto \frac{\alpha^{-3}}{(\alpha+\sigma^2)^9}\,\exp\left\{-\frac{2}{\alpha}\right.\\ &\quad -\frac{1+n(\alpha+\sigma^2)^{-1}}{2}\left[\mu-(\alpha+\sigma^2)^{-1}\sum_{i=1}^{18}y_i\big/(1+n(\alpha+\sigma^2)^{-1})\right]^2\\ &\quad \left.-\frac{1}{2(\alpha+\sigma^2)}\sum_{i=1}^{18}y_i^2+\frac{(\alpha+\sigma^2)^{-2}}{2(1+n(\alpha+\sigma^2)^{-1})}\left(\sum_{i=1}^{18}y_i\right)^2\right\}\end{aligned}$$

and thus

$$\begin{aligned}\pi(\alpha|\mathbf{y}) &\propto \frac{\alpha^{-3}(1+n(\alpha+\sigma^2)^{-1})^{-1/2}}{(\alpha+\sigma^2)^9}\,\exp\left\{-\frac{2}{\alpha}\right.\\ &\quad \left.-\frac{1}{\alpha+\sigma^2}\sum_{i=1}^{18}y_i^2+\frac{(\alpha+\sigma^2)^{-2}}{1+n(\alpha+\sigma^2)^{-1}}\left(\sum_{i=1}^{18}y_i\right)^2\right\}\end{aligned}$$

Therefore the marginal posterior distribution on α has a closed (albeit complex) form. (It is also obvious from $\pi(\alpha,\mu|\mathbf{y})$ above that the marginal posterior on μ does not have a closed form.)

The baseball dataset can be found in the `amcmc` package in the `baseball.c` program and rewritten as

```
baseball=c(0.395,0.375,0.355,0.334,0.313,0.313,0.291,
0.269,0.247,0.247,0.224,0.224,0.224,0.224,0.224,0.200,
0.175,0.148)
```

The standard Gibbs sampler is implemented by simulating

$$\begin{aligned}\theta_i|y_i,\mu,\alpha &\sim \mathcal{N}\left(\frac{\alpha^{-1}\mu+\sigma^{-2}y_i}{\alpha^{-1}+\sigma^{-2}},(\alpha^{-1}+\sigma^{-2})^{-1}\right),\\ \mu|\theta,\alpha &\sim \mathcal{N}\left(\frac{\alpha^{-1}\sum_{i=1}^{18}\theta_i}{1+n\alpha^{-1}},(n\alpha^{-1}+1)^{-1}\right),\\ \alpha|\theta,\mu &\sim \mathcal{IG}\left(11,2+\sum_{i=1}^{18}(\theta_i-\mu)^2/2\right)\end{aligned}$$

which means using an `R` loop like

```
Nsim=10^4
sigma2=0.00434;sigmam=1/sigma2
theta=rnorm(18)
mu=rep(rnorm(1),Nsim)
alpha=rep(rexp(1),Nsim)
```

```
for (t in 2:Nsim){
  theta=rnorm(18,mean=(mu[t-1]/alpha[t-1]+sigmam*baseball)/
  (1/alpha[t-1]+sigmam),sd=1/sqrt(1/alpha[t-1]+sigmam))
  mu[t]=rnorm(1,mean=sum(theta)/(1/alpha[t-1]+n),
        sd=1/sqrt(1+n/alpha[t-1]))
  alpha[t]=(2+0.5*sum((theta-mu[t])^2))/rgamma(1,11)
}
```

The result of both `coda` diagnostics on α is

```
> heidel.diag(mcmc(alpha))

     Stationarity start     p-value
     test         iteration
var1 passed       1         0.261

     Halfwidth Mean  Halfwidth
     test
var1 passed    0.226 0.00163
> geweke.diag(mcmc(alpha))

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

   var1
-0.7505
```

If we reproduce the Kolmogorov–Smirnov analysis

```
ks=NULL
M=10
for (t in seq(Nsim/10,Nsim,le=100)){
alpha1=alpha[1:(t/2)]
alpha2=alpha[(t/2)+(1:(t/2))]
alpha1=alpha1[seq(1,t/2,by=M)]
alpha2=alpha2[seq(1,t/2,by=M)]
ks=c(ks,ks.test(alpha1,alpha2)$p)
}
```

Plotting the vector `ks` by `plot(ks,pch=19)` shows no visible pattern that would indicate a lack of uniformity.

Comparing the output with the true target in α follows from the definition

```
marge=function(alpha){
(alpha^(-3)/(sqrt(1+18*(alpha+sigma2)^(-1))*(alpha+sigma2)^9))*
exp(-(2/alpha) - (.5/(alpha+sigma2))*sum(baseball^2) +
.5*(alpha+sigma2)^(-2)*sum(baseball)^2/(1+n*(alpha+sigma2)^(-1)))
}
```

Figure 8.2 shows the fit of the simulated histogram to the above function (when normalized by `integrate`).

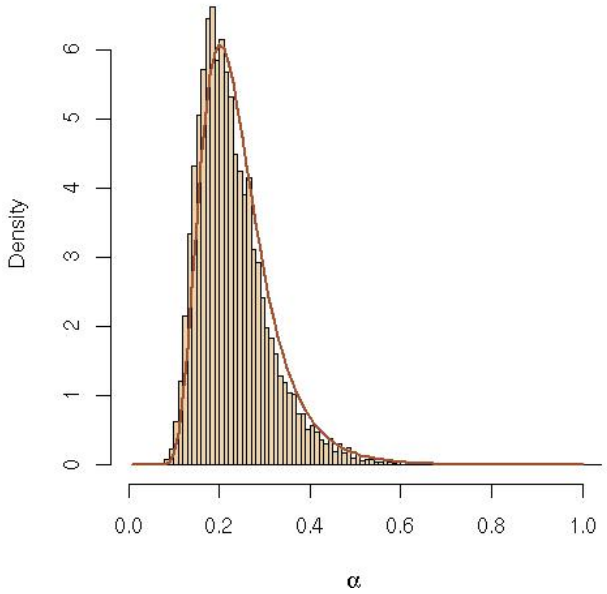

Fig. 8.2. Histogram of the $(\alpha^{(t)})$ chain produced by the Gibbs sampler of Example 8.9 and fit of the exact marginal $\pi(\alpha|\mathbf{y})$, based on 10^4 simulations.

Exercise 8.9

a. We simply need to check that this transition kernel K satisfies the detailed balance condition (6.3), $f(x)K(y|x) = f(y)K(x|y)$ when f is the $\mathcal{B}e(\alpha, 1)$ density: when $x \neq y$,

$$\begin{aligned} f(x)K(x,y) &= \alpha x^{\alpha-1}\, x\,(\alpha+1)\, y^{\alpha} \\ &= \alpha(\alpha+1)(xy)^{\alpha} \\ &= f(y)K(y,x) \end{aligned}$$

so the $\mathcal{B}e(\alpha, 1)$ distribution is indeed stationary.

b. Simulating the Markov chain is straightforward:

```
alpha=.2
Nsim=10^4
x=rep(runif(1),Nsim)
y=rbeta(Nsim,alpha+1,1)
```

```
for (t in 2:Nsim){
   if (runif(1)<x[t-1]) x[t]=y[t]
   else x[t]=x[t-1]
   }
```

and it exhibits a nice fit to the beta $\mathcal{B}e(\alpha, 1)$ target. However, running `cumuplot` shows a lack of concentration of the distribution, while the two standard stationarity diagnoses are

```
> heidel.diag(mcmc(x))

     Stationarity start     p-value
     test         iteration
var1 passed       1001      0.169

     Halfwidth Mean  Halfwidth
     test
var1 failed    0.225 0.0366
> geweke.diag(mcmc(x))

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

 var1
3.277
```

are giving dissonant signals. The `effectiveSize(mcmc(x))}` is then equal to 329. Moving to 10^6 simulations does not modify the picture (but may cause your system to crash!)

c. The corresponding Metropolis–Hastings version is

```
alpha=.2
Nsim=10^4
x=rep(runif(1),Nsim)
y=rbeta(Nsim,alpha+1,1)
for (t in 2:Nsim){
   if (runif(1)<x[t-1]/y[t]) x[t]=y[t]
   else x[t]=x[t-1]
   }
```

It also provides a good fit and also fails the test:

```
> heidel.diag(mcmc(x))

     Stationarity start     p-value
     test         iteration
var1 passed       1001      0.0569
```

```
     Halfwidth Mean  Halfwidth
     test
var1 failed    0.204 0.0268
> geweke.diag(mcmc(x))

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

 var1
1.736
```

Exercise 8.11

a. A possible `R` definition of the posterior is

```
postit=function(beta,sigma2){
  prod(pnorm(r[d==1]*beta/sigma2))*prod(pnorm(-r[d==0]*beta/sigma2))*
  dnorm(beta,sd=5)*dgamma(1/sigma2,2,1)}
```

and a possible `R` program is

```
r=Pima.tr$ped
d=as.numeric(Pima.tr$type)-1
mod=summary(glm(d~r-1,family="binomial"))
beta=rep(mod$coef[1],Nsim)
sigma2=rep(1/runif(1),Nsim)
for (t in 2:Nsim){
  prop=beta[t-1]+rnorm(1,sd=sqrt(sigma2[t-1]*mod$cov.unscaled))
  if (runif(1)<postit(prop,sigma2[t-1])/postit(beta[t-1],
sigma2[t-1])) beta[t]=prop
  else beta[t]=beta[t-1]
  prop=exp(log(sigma2[t-1])+rnorm(1))
  if (runif(1)<sigma2[t-1]*postit(beta[t],prop)/(prop*
      postit(beta[t], sigma2[t-1]))) sigma2[t]=prop
  else sigma2[t]=sigma2[t-1]
  }
```

(Note the Jacobian $1/\sigma^2$ in the acceptance probability.)

b. Running 5 chains in parallel is easily programmed with an additional loop in the above. Running `gelman.diag` on those five chains then produces a convergence assessment:

```
> gelman.diag(mcmc.list(mcmc(beta1),mcmc(beta2),mcmc(beta3),
+ mcmc(beta4),mcmc(beta5)))
Potential scale reduction factors:
     Point est. 97.5% quantile
[1,]       1.02           1.03
```

Note also the good mixing behavior of the chain:

```
> effectiveSize(mcmc.list(mcmc(beta1),mcmc(beta2),
+ mcmc(beta3),mcmc(beta4),mcmc(beta5)))
    var1
954.0543
```

c. The implementation of the traditional Gibbs sampler with completion is detailed in Marin and Robert (2007), along with the appropriate `R` program. The only modification that is needed for this problem is the introduction of the non-identifiable scale factor σ^2.

Exercise 8.13

In the `kscheck.R` program available in `mcsm`, you can modify G by changing the variable `M` in

```
subbeta=beta[seq(1,T,by=M)]
subold=oldbeta[seq(1,T,by=M)]
ks=NULL
for (t in seq((T/(10*M)),(T/M),le=100))
   ks=c(ks,ks.test(subbeta[1:t],subold[1:t])$p)
```

(As noted by a reader, the syntax `ks=c(ks,res)` is very inefficient in system time, as you can check by yourself.)

Exercise 8.15

Since the Markov chain $(\theta^{(t)})$ is converging to the posterior distribution (in distribution), the density at time t, π_t, is also converging (pointwise) to the posterior density $\pi(\theta|x)$, therefore ω_t is converging to

$$\frac{f(x|\theta^{(\infty)})\pi(\theta^{(\infty)})}{\pi(\theta^{(\infty)}|x)} = m(x)\,,$$

for all values of $\theta^{(\infty)}$. (This is connected with Chib's (1995) method, discussed in Exercise 7.16.)

Exercise 8.17

If we get back to Example 8.1, the sequence `beta` can be checked in terms of effective sample via an `R` program like

```
ess=rep(1,T/10)
for (t in 1:(T/10)) ess[t]=effectiveSize(beta[1:(10*t)])
```

where the subsampling is justified by the computational time required by `effectiveSize`. The same principle can be applied to any chain produced by an MCMC algorithm.

Figure 8.3 compares the results of this evaluation over the first three examples of this chapter. None of them is strongly conclusive about convergence...

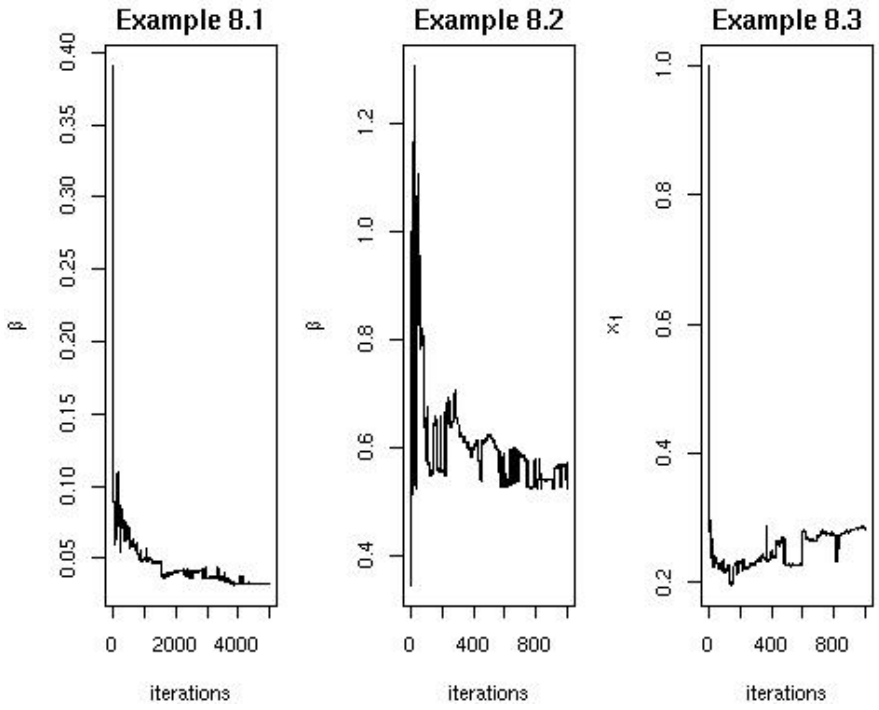

Fig. 8.3. Evolution of the effective sample size across iterations for the first three examples of Chapter 8.

References

Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90:1313–1321.

MacEachern, S. and Berliner, L. (1994). Subsampling the Gibbs sampler. *The American Statistician*, 48:188–190.

Marin, J.-M. and Robert, C. (2007). *Bayesian Core*. Springer–Verlag, New York.

Meyn, S. and Tweedie, R. (1993). *Markov Chains and Stochastic Stability*. Springer–Verlag, New York.

Robert, C. and Casella, G. (2004). *Monte Carlo Statistical Methods,* second edition. Springer–Verlag, New York.

Robert, C. and Marin, J.-M. (2010). Importance sampling methods for Bayesian discrimination between embedded models. In Chen, M.-H., Dey, D. K., Mueller, P., Sun, D., and Ye, K., editors, *Frontiers of Statistical Decision Making and Bayesian Analysis*. (To appear.).