MATH 598: TOPICS IN STATISTICS

AUXILIARY VARIABLE METHODS AND THE GIBBS SAMPLER

The principle of *auxiliary* or *latent* variables is to extend the state-space of the Markov chain in such a way that the full conditionals posterior become tractable, or which help to break the dependence amongst the variables. Consider target distribution $\pi_X(x)$ for random variables X, and let U be a collection of random variables such that the joint distribution $\pi_{X,U}(x,u)$ has marginal $\pi_X(x)$

$$\pi_X(x) = \int \pi_{X,U}(x,u)\, du$$

In a Gibbs sampler context, we could consider constructing a Markov chain using the two conditional distributions

$$\pi_{X|U}(x|u) \qquad\qquad \pi_{U|X}(u|x)$$

and then collecting the samples for X, which are necessarily samples from $\pi_X(x)$. Note that

$$\pi_{X,U}(x,u) = \pi_X(x)\pi_{U|X}(u|x)$$

We are free to select the U variables in whichever way is appropriate. We seek to introduce U such that

- the full conditionals are straightforward to sample;
- the dependence structure amongst the X variables is broken.

For example, suppose we have a structure such that

$$\pi_X(x) \;\propto\; \pi_X^{(0)}(x) \prod_{k=1}^{K} b_k(x)$$

where $\pi_X^{(0)}(x)$ can be sampled in straightforward fashion. Let

$$U_k|X = x \sim \text{Uniform}(0, b_k(x)) \qquad k = 1, \ldots, K$$

be a collection of conditionally independent random variables.

$$\begin{aligned}
\pi_{X,U}(x,u) &= \pi_X(x)\pi_{U|X}(u|x) \\
&= \left\{\pi_X^{(0)}(x) \prod_{k=1}^{K} b_k(x)\right\} \left\{\prod_{k=1}^{K} \frac{1}{b_k(x)} \mathbb{1}_{[0,b_k(x)]}(u_k)\right\} \\
&= \left\{\pi_X^{(0)}(x)\right\} \left\{\prod_{k=1}^{K} \mathbb{1}_{[0,b_k(x)]}(u_k)\right\}
\end{aligned}$$

Therefore

$$\pi_{X|U}(x|u) \;\propto\; \pi_X^{(0)}(x) \prod_{k=1}^{K} \mathbb{1}_{[u_k,\infty)}(b_k(x))$$

that is, proportional to $\pi_X^{(0)}(x)$ with support restricted by

$$b_k(x) \geq u_k \qquad k = 1, \ldots, K.$$

To sample this full conditional distribution, we may sample $\pi_X^{(0)}(x)$ – recall this is straightforward – and reject the sampled x values for which the constraints are not met.

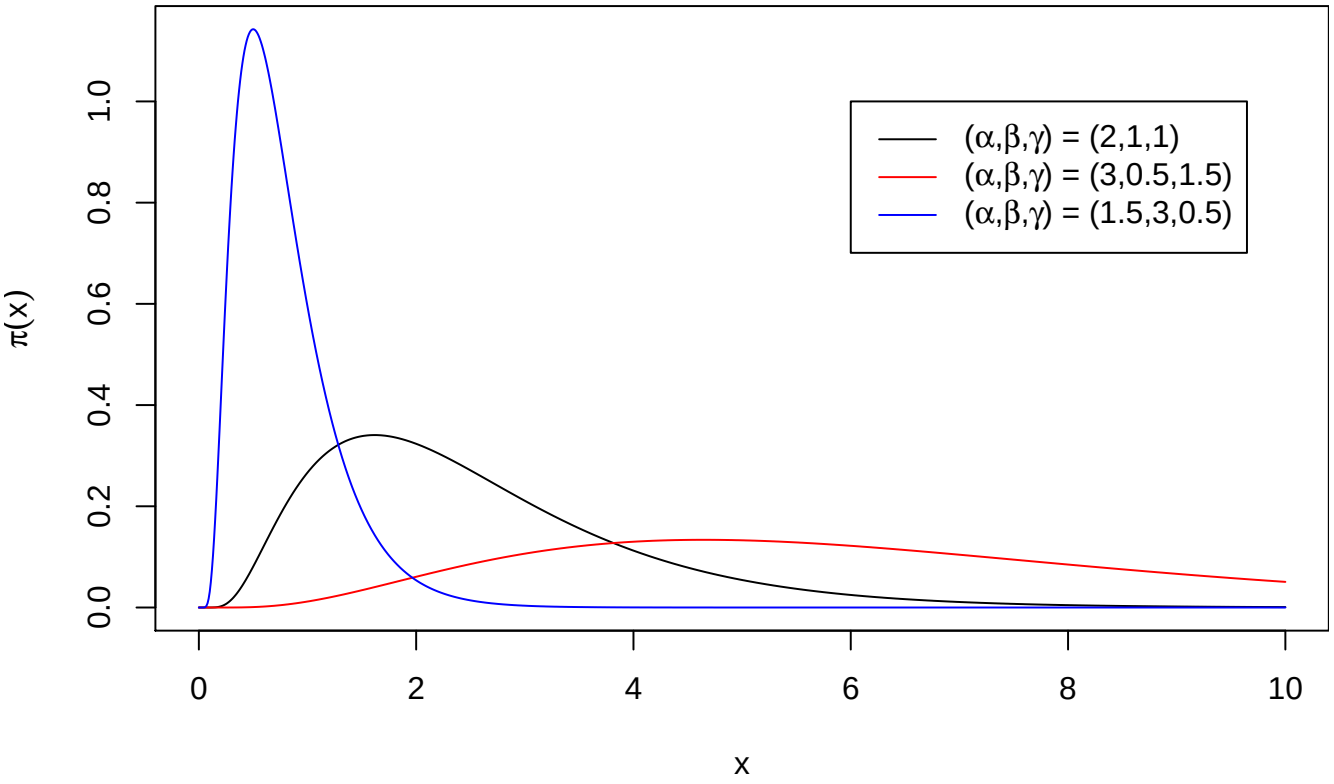
This method is also utilized in the *Slice Sampler* that is now widely use in the MCMC literature.

EXAMPLE: Sampling an awkward distribution
Suppose we wish to produce a sample from the pdf

$$\pi(x) \propto x^{\alpha-1} \exp\left\{-\left[\beta x + \frac{\gamma}{x}\right]\right\} \qquad x > 0$$

for parameters $\alpha, \beta, \gamma > 0$. This pdf is similar to the Gamma pdf, but is not as tractable. The normalizing constant is not trivial to compute in general.

```
pdfunc<-function(x,a,b,g,log=F){
        yv<-(a-1)*log(x)-b*x-g/x
        if(!log){yv<-exp(yv)}
        return(yv)
}
xv<-seq(0,10,by=0.01)
yv0<-pdfunc(xv,2,1,1)
c0<-integrate(pdfunc,lower=0,upper=100,a=2,b=1,g=1)$val
yv1<-pdfunc(xv,3,0.5,1.5)
c1<-integrate(pdfunc,lower=0,upper=100,a=3,b=0.5,g=1.5)$val
yv2<-pdfunc(xv,1.5,3,0.5)
c2<-integrate(pdfunc,lower=0,upper=100,a=1.5,b=3,g=0.5)$val
par(mar=c(4,4,1,0))
plot(xv,yv0/c0,ylim=range(0,max(yv0/c0,yv1/c1,yv2/c2)),xlab='x',ylab=expression(pi(x)),type='l')
lines(xv,yv1/c1,col='red')
lines(xv,yv2/c2,col='blue')
legend(6,1,c(expression(paste('(',alpha,',',beta,',',gamma,') = (2,1,1)')),
expression(paste('(',alpha,',',beta,',',gamma,') = (3,0.5,1.5)')),
expression(paste('(',alpha,',',beta,',',gamma,') = (1.5,3,0.5)'))),lty=1,col=c('black','red','blue'))
```



To sample this distribution using an auxiliary variable Gibbs sampler, we note that

$$\pi(x) \propto \pi^{(0)}(x) b(x)$$

where

$$\pi^{(0)}(x) \propto x^{\alpha-1} \exp\{-\beta x\} \qquad b(x) = \exp\left\{-\frac{\gamma}{x}\right\},$$

that is, $\pi_X^{(0)}(x) \equiv Gamma(\alpha, \beta)$ which may be sampled directly. In line with the auxiliary variable approach outlined above, this suggests introducing variable U where

$$U|X = x \sim Uniform(0, b(x)) \equiv Uniform(0, \exp\{-\gamma/x\})$$

so that the joint pdf for X and U IS

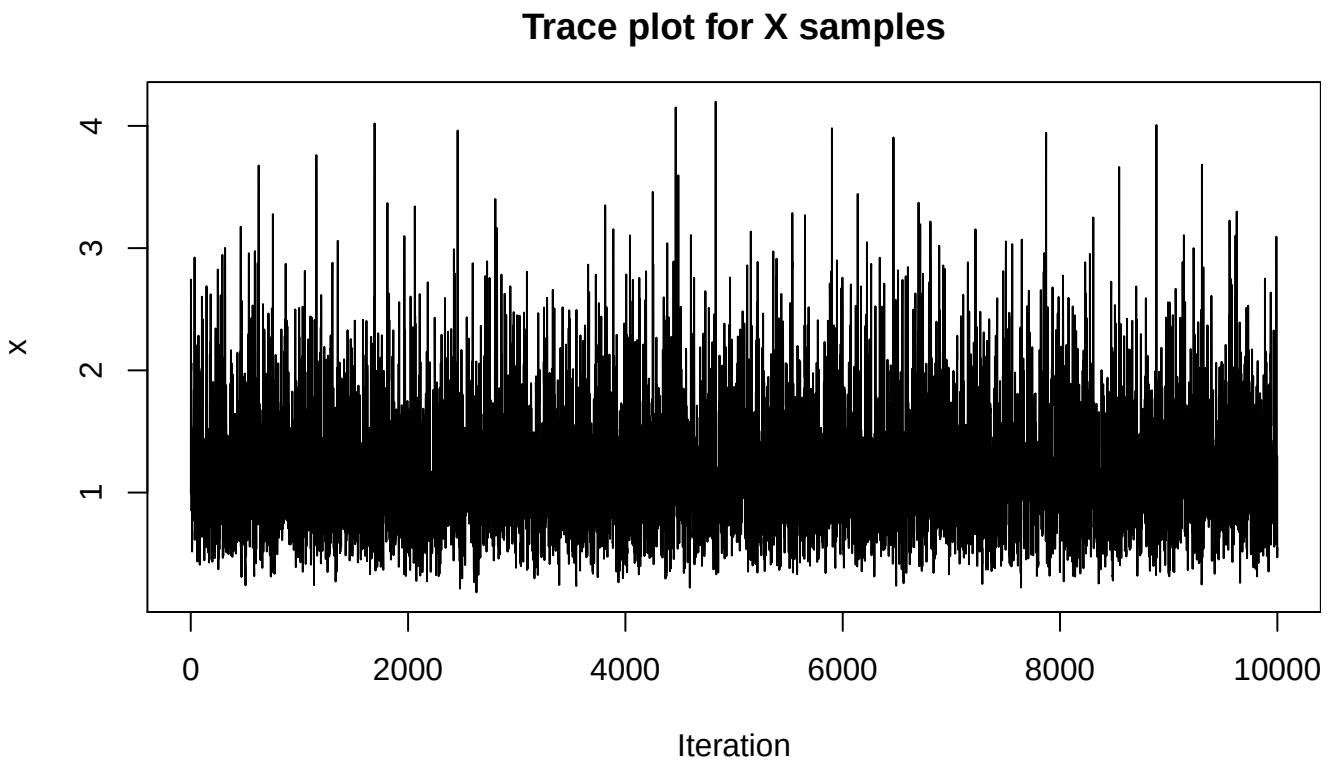
$$\pi_{X,U}(x,u) \quad = \quad \pi_X(x)\pi_{U|X}(u|x) = \left\{\pi_X^{(0)}(x)b(x)\right\}\frac{1}{b(x)}\mathbb{1}_{[0,b(x)]}(u) = \left\{\pi_X^{(0)}(x)\right\}\mathbb{1}_{[0,b(x)]}(u)$$

Therefore

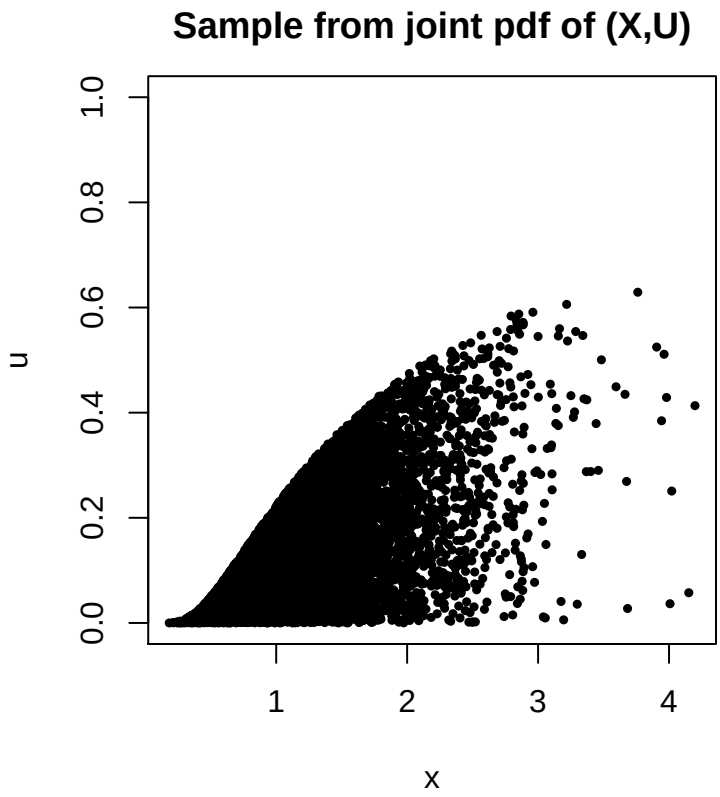
$$\pi_{X|U}(x|u) \ \propto \ \pi_X^{(0)}(x)\mathbb{1}_{[u,\infty)}(b(x))$$

that is, proportional to $\pi_X^{(0)}(x)$ with support restricted by the constraint $b(x) \geq u$ for fixed u. To sample this full conditional distribution, we may sample $\pi_X^{(0)}(x) \equiv Gamma(\alpha, \beta)$ and reject the sampled x values for which the constraint is not met. This is easily implemented in the following Gibbs sampler.

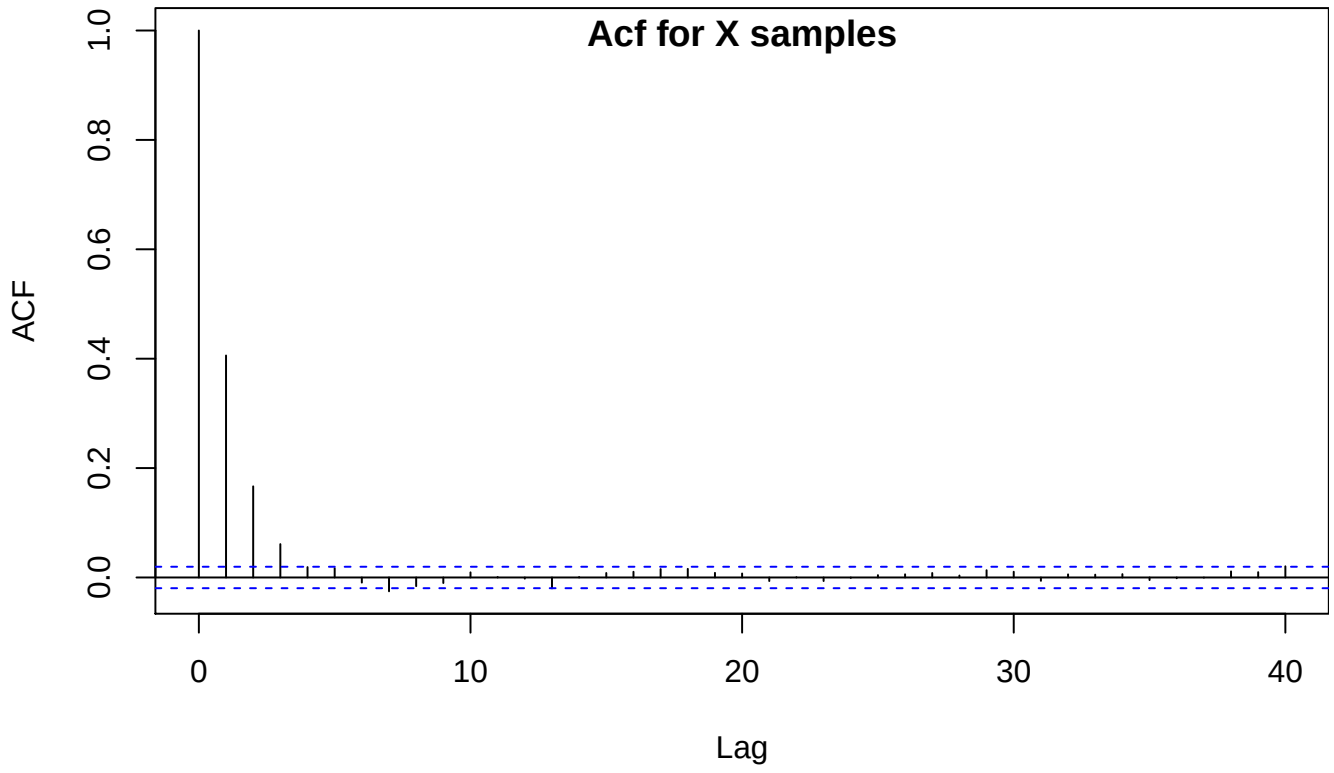
```
al<-2;be<-3;gam<-1.5
old.x<-rgamma(1,al,be);old.u<-runif(1,0,exp(-gam/old.x))
nreps<-10000
xu.samp<-matrix(0,nrow=nreps,ncol=2)
for(irep in 1:nreps){
        old.x<-rgamma(1,al,be)
        while(old.u > exp(-gam/old.x)){old.x<-rgamma(1,al,be)}
        old.u<-runif(1,0,exp(-gam/old.x))
        xu.samp[irep,]<-c(old.x,old.u)
}
par(mar=c(4,4,3,0));plot(xu.samp[,1],type='l',xlab='Iteration',ylab='x')
title('Trace plot for X samples')
```



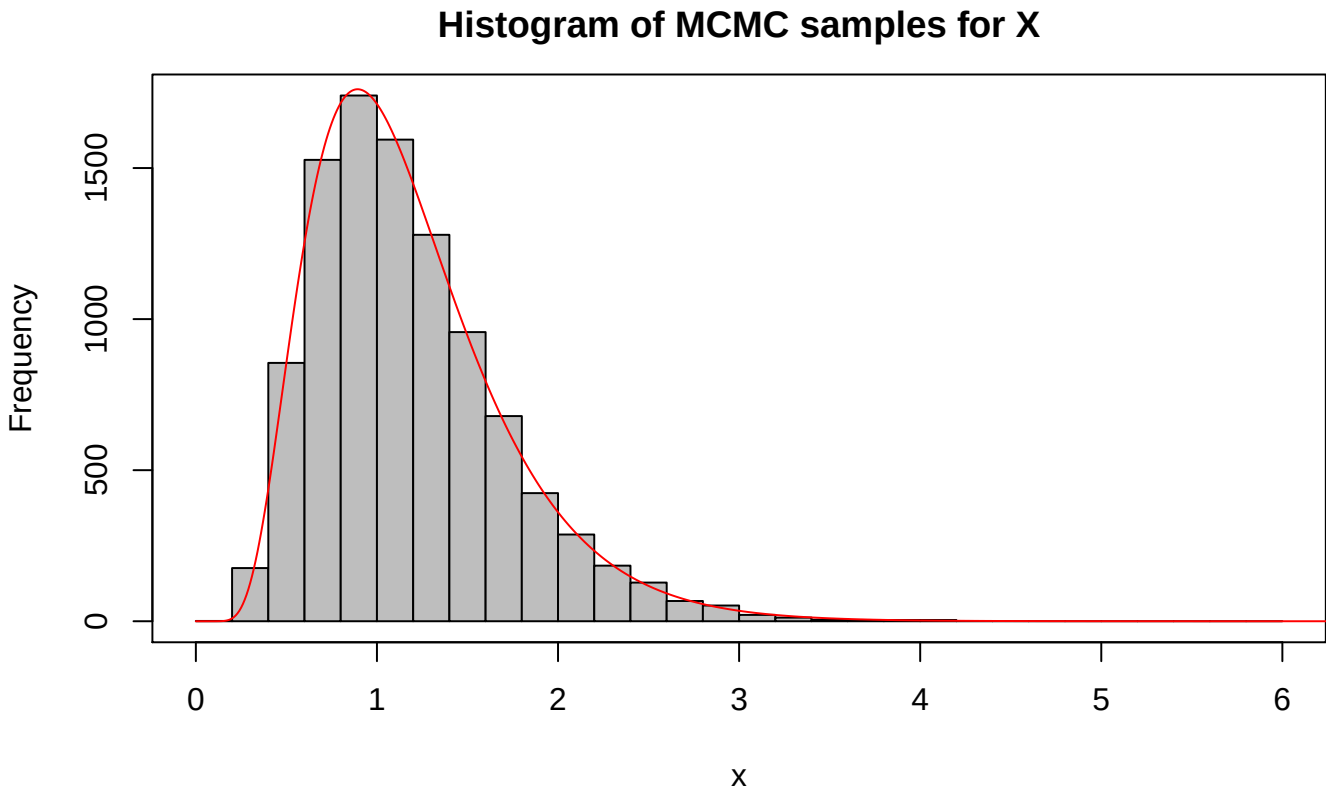

```
par(mar=c(4,4,2,0),pty='s');plot(xu.samp,xlab='x',ylab='u',pch=19,cex=0.5,ylim=range(0,1))
title('Sample from joint pdf of (X,U)',line=1)
```




```
par(mar=c(4,4,1,0),pty='m')
acf(xu.samp[,1],main=' ');title('Acf for X samples',line=-1)
```



```
const<-integrate(pdfunc,lower=0,upper=100,a=al,b=be,g=gam)$val
yv<-pdfunc(xv,al,be,gam)/const
par(mar=c(4,4,2,0))
hist(xu.samp[,1],xlab='x',main='',col='gray',breaks=seq(0,6,by=0.2));box()
lines(xv,yv*nreps*0.2,col='red')
title('Histogram of MCMC samples for X',line=1)
```



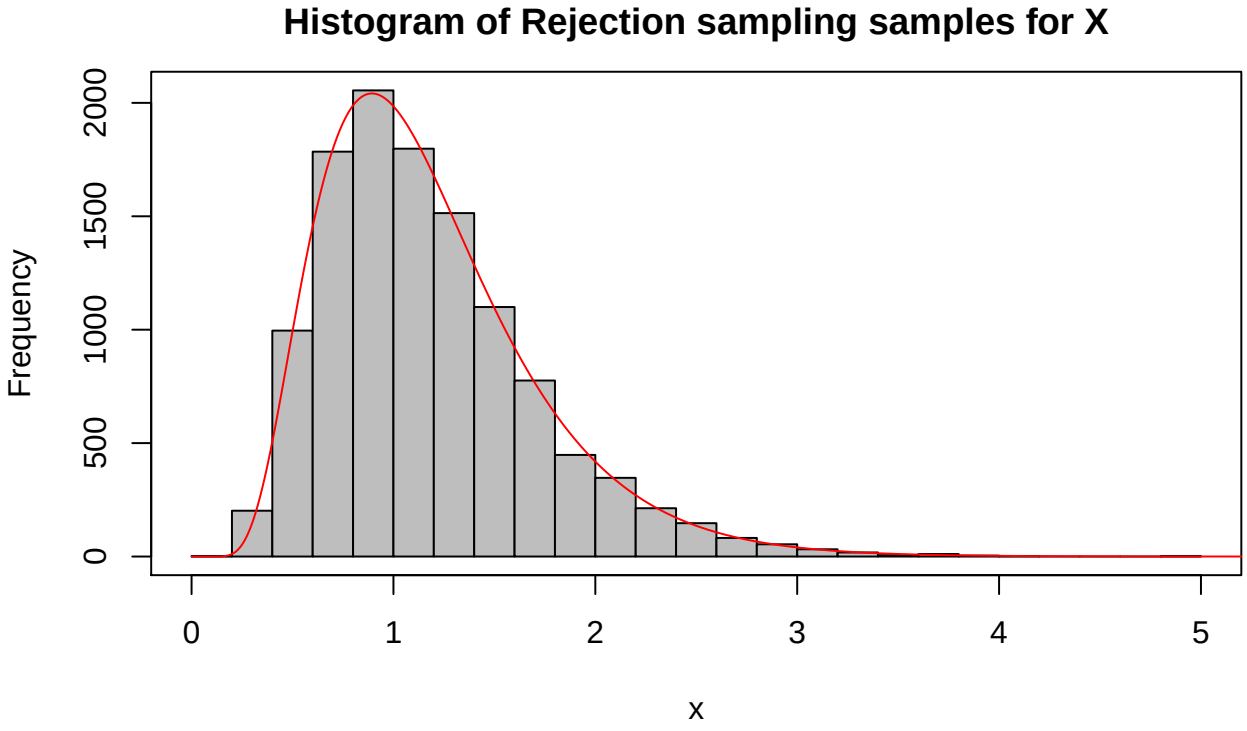
In this problem, we may also use rejection sampling using the proposal density $\widetilde{\pi}(x) \equiv Gamma(\alpha, \beta)$, so that

$$\frac{\pi(x)}{\widetilde{\pi}(x)} \leq c \exp\left\{-\frac{\gamma}{x}\right\} \leq c$$

for some finite constant c. Then the rejection sampling accepts points x generated from $\widetilde{\pi}(x)$ provided a $Uniform(0,1)$ random variate u satisfies

$$u \leq \exp\left\{-\frac{\gamma}{x}\right\}.$$

```
#Rejection sampling

N0<-100000
X0<-rgamma(N0,al,be)
U<-runif(N0)
X<-X0[U<=exp(-gam/X0)]
N<-length(X)
N/N0   #Acceptance Rate

+ [1] 0.11594

hist(X[X<5],xlab='x',main='',col='gray',breaks=seq(0,5,by=0.2));box()
lines(xv,yv*N*0.2,col='red')
title('Histogram of Rejection sampling samples for X',line=1)
```



EXAMPLE: The Ising Model
The **Ising model** is a joint distribution for the collection of binary random variables $\{X_i\}$ placed on a rectangular $(N \times M)$ lattice, with joint mass function

$$\pi_{\mathbf{X}}(\mathbf{x}) = \frac{\exp\left\{\beta \sum_{i=1}^{NM} \sum_{j \in \partial i} \mathbb{1}_{\{x_i\}}(x_j)\right\}}{Z(\beta)}$$

where ∂i is a suitably defined neighbourhood of i, and where $Z(\beta)$ is the normalizing constant. The support of this mass function 2^{NM} elements, which are all possible configurations of a binary vector of length NM. The *full conditional distribution* for the X_i is a discrete distribution on $\{0, 1\}$, with

$$\pi_i(x_i|x_{(i)}) \propto \exp\left\{\beta \sum_{j \in \partial i} \mathbb{1}_{x_j}(x_i)\right\}.$$

This distribution reduces to

$$\Pr(X_i = 0|X_{(i)} = x_{(i)}) = \frac{e^{\beta n_{i0}}}{e^{\beta n_{i0}} + e^{\beta n_{i1}}}$$

where n_{i0} and n_{i1} are the numbers of neighbours of i that take the values 0 and 1 respectively.

The Gibbs sampler is readily implementable for the Ising model, but can be extremely slow to converge to the stationary distribution. This is due in part to high dependence between the variables. One method for speeding up simulation of variates from the Ising model is to introduce auxiliary variables as escribed below. The most common latent variable method for the Ising model is the *Swendsen-Wang algorithm*.

$$\pi_X(x) \propto \exp\left\{\beta \sum_{l} \sum_{m \in \partial l} \mathbb{1}_{\{x_m\}}(x_l)\right\}.$$

where ∂l constitute the neighbours of site l. In the simplest case, if site l is at Cartesian location (i, j), the neighbours are the four sites immediately

- to the left $(i-1, j)$
- to the right $(i+1, j)$
- above $(i, j+1)$
- below $(i, j-1)$

site (i, j). The full conditional distributions for the Gibbs sampler are given by

$$X_l | X_{\partial l} = x_{\partial l} \sim \text{Bernoulli}(\theta(\beta; x_{\partial l}))$$

that is

$$\Pr[X_l = 1 | X_{\partial l} = x_{\partial l}] = \theta(\beta; x_{\partial l})$$

where

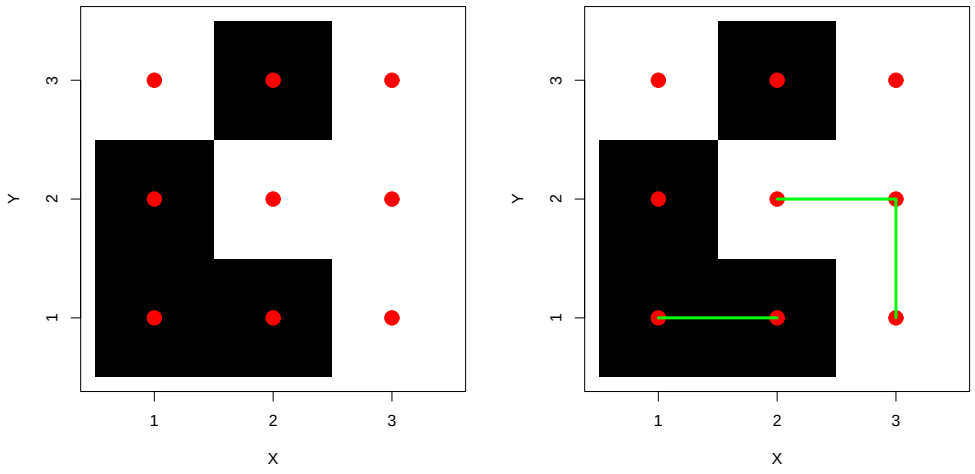
$$\theta(\beta; x_{\partial l}) = \frac{\exp\{\beta N_{l1}\}}{\exp\{\beta N_{l0}\} + \exp\{\beta N_{l1}\}}$$

This algorithm updates one x value at a time, either in a *fixed* or *random* scan fashion.

The Swendsen-Wang algorithm introduces a collection of auxiliary bond variables, u. Consider two neighbouring sites, l and m, say. Let $\omega = 1 - e^{-\beta}$. Define the binary bond random variable U_{lm}, conditional on X_l and X_m, to have distribution described by

$$\Pr[U_{lm} = 1 | X_l = x_l, X_m = x_m] = \begin{cases} \omega & x_l = x_m \\ 0 & x_l \neq x_m \end{cases}$$

such that the collection of U variables are conditionally independent given X variables. The bond is *present* if $U_{lm} = 1$, and *absent* otherwise. The figure below displays bond variables (green) between adjacent, similar x values.



Consider the joint specification $\pi_{X,U}$ given by

$$\pi_{X,U}(x, u) \propto \prod_{l,m} \{\omega \delta_{x_l x_m} \delta_{u_{lm} 1} + (1 - \omega) \delta_{u_{lm} 0}\}$$

where the product is over all pairs of neighbouring sites, and $\delta_{ab} = 1$ if $a = b$, and zero otherwise. We have

$$\pi_{U_{lm}|X}(u|x) = \pi_{U_{lm}|X_l, X_m}(u|x_l, x_m) = \begin{cases} \omega^u (1-\omega)^{1-u} & x_l = x_m \\ \mathbb{1}_{\{0\}}(u) & x_l \neq x_m \end{cases}$$

for $u = 0, 1$. Also marginalizing over the u variables by summing out over $u_{lm} = 0, 1$ can be achieved over each term in the product separately. We have

$$\begin{aligned}\sum_u \pi_{X,U}(x,u) &\propto \prod_{l,m} \{(1-\omega) + \omega\delta_{x_l x_m}\} \\ &= \prod_{l,m} \left\{e^{-\beta} + (1-e^{-\beta})\delta_{x_l x_m}\right\}.\end{aligned}$$

Multiplying through each term by $e^{-\beta}$ yields

$$\sum_u \pi_{X,U}(x,u) \propto \prod_{l,m} \left\{1 + (e^{\beta}-1)\delta_{x_l x_m}\right\}.$$

For each term in the product, when $x_l \neq x_m$, the term contributes 1; when $x_l = x_m$, the term contributes e^{β}. Hence

$$\sum_u \pi_{X,U}(x,u) \propto \prod_{l,m} \exp\{\beta\delta_{x_l x_m}\} = \exp\left\{\beta \sum_{l,m} \delta_{x_l x_m}\right\} = \pi_X(x)$$

and so the **correct marginal Ising model is recovered**. Marginalizing over the X variables, we have

$$\begin{aligned}\sum_x \pi_{X,U}(x,u) &\propto \sum_x \prod_{l,m} \{\omega\delta_{x_l x_m}\delta_{u_{lm}1} + (1-\omega)\delta_{u_{lm}0}\} \\ &\propto \omega^{n(u)}(1-\omega)^{n_b - n(u)}\end{aligned}$$

where $n(u)$ is the total number of bonds present

$$n(u) = \sum_{l,m} u_{lm}$$

and n_b is the total number of bonds possible. We now consider running a Gibbs sampler for $\pi_{X,U}$.

- The full conditional distribution of $U|X = x$ is determined by the definition above, that is,

  $$\pi_{U|X}(u|x) = \prod_{l,m} \pi_{U_{lm}|X_l,X_m}(u_{lm}|x_l, x_m)$$

  where

  $$\pi_{U_{lm}|X_l,X_m}(u_{lm}|x_l,x_m) = \begin{cases} \omega^u(1-\omega)^{1-u} & x_l = x_m \\ \mathbb{1}_{\{0\}}(u) & x_l \neq x_m \end{cases}$$

  for $u = 0, 1$, with the product being over all neighbouring sites. Thus the U variables can be updated independently, conditional on X.

- The full conditional distribution of $X|U$ is defined by the fact that

  $$\pi_{X|U}(x|u) \propto \pi_{X,U}(x,u) \propto \prod_{l,m} \{\omega\delta_{x_l x_m}\delta_{u_{lm}1} + (1-\omega)\delta_{u_{lm}0}\}$$

  and that bonds link "clusters" of sites that have the same value. The conditional distribution does not depend on the **value** of each x_l, merely whether or not $x_l = x_m$.

  Conditional on u, clusters can assume the label 0 or 1, **independently** with **equal probability**.

Thus the Gibbs sampler can be implemented by sampling from the two full conditional distributions

$$\pi_{X|U}(x|u) \qquad \pi_{U|X}(u|x)$$

recursively, both of which involve sampling from trivial distributions. There is some computational overhead in implementing Swendsen-Wang, in particular, tracing the clusters.

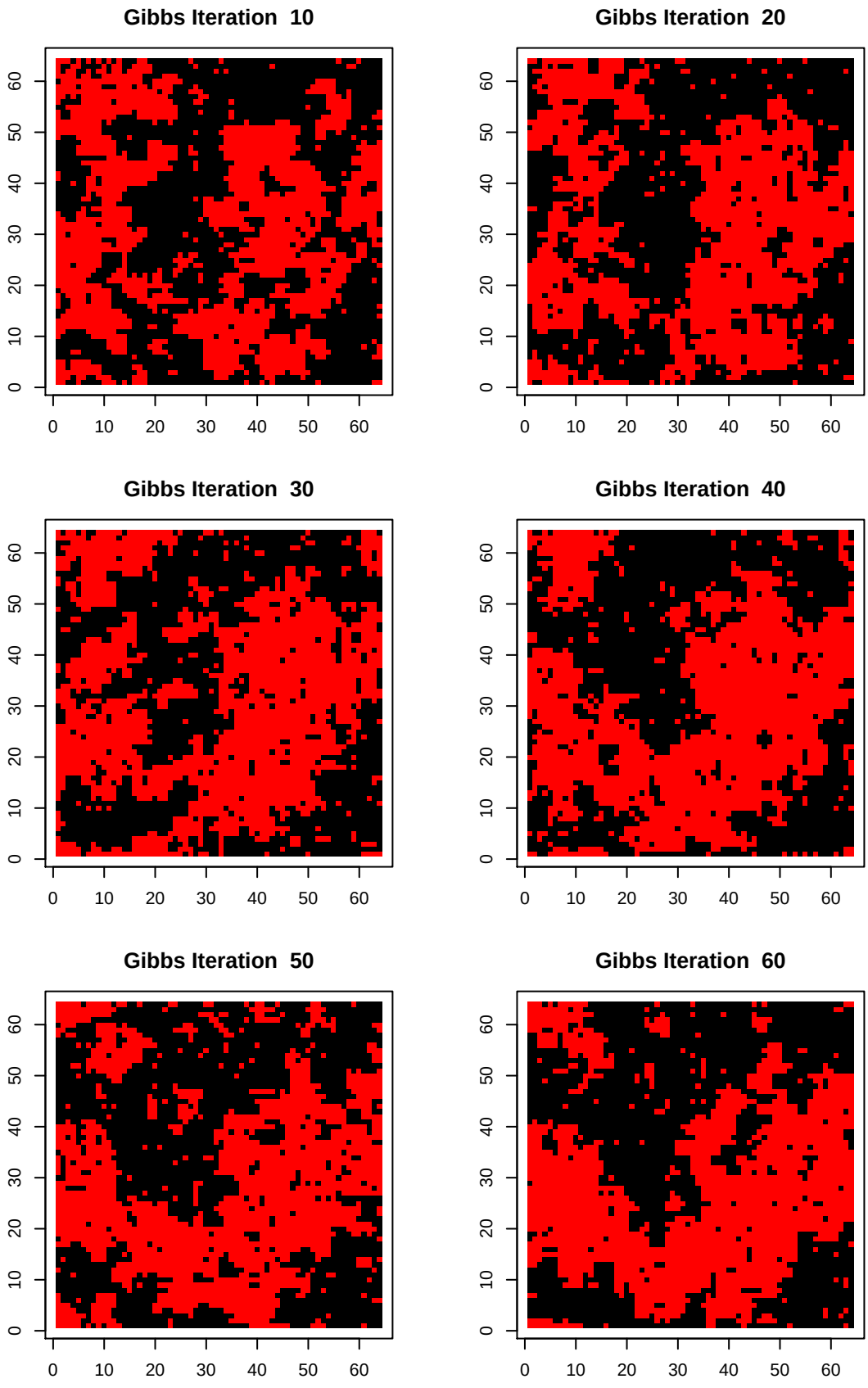
For a specific computational example: with grid size $N = 64$, $\beta = 0.90$, the Gibbs sampler and Swendsen-Wang (SW) algorithms were run in R for 60 iterations. The SW chain converges much more quickly; the Gibbs sampler chain exhibits slower convergence for larger values of β. With $\beta = 0.9$, the performance of the two methods is quite different.

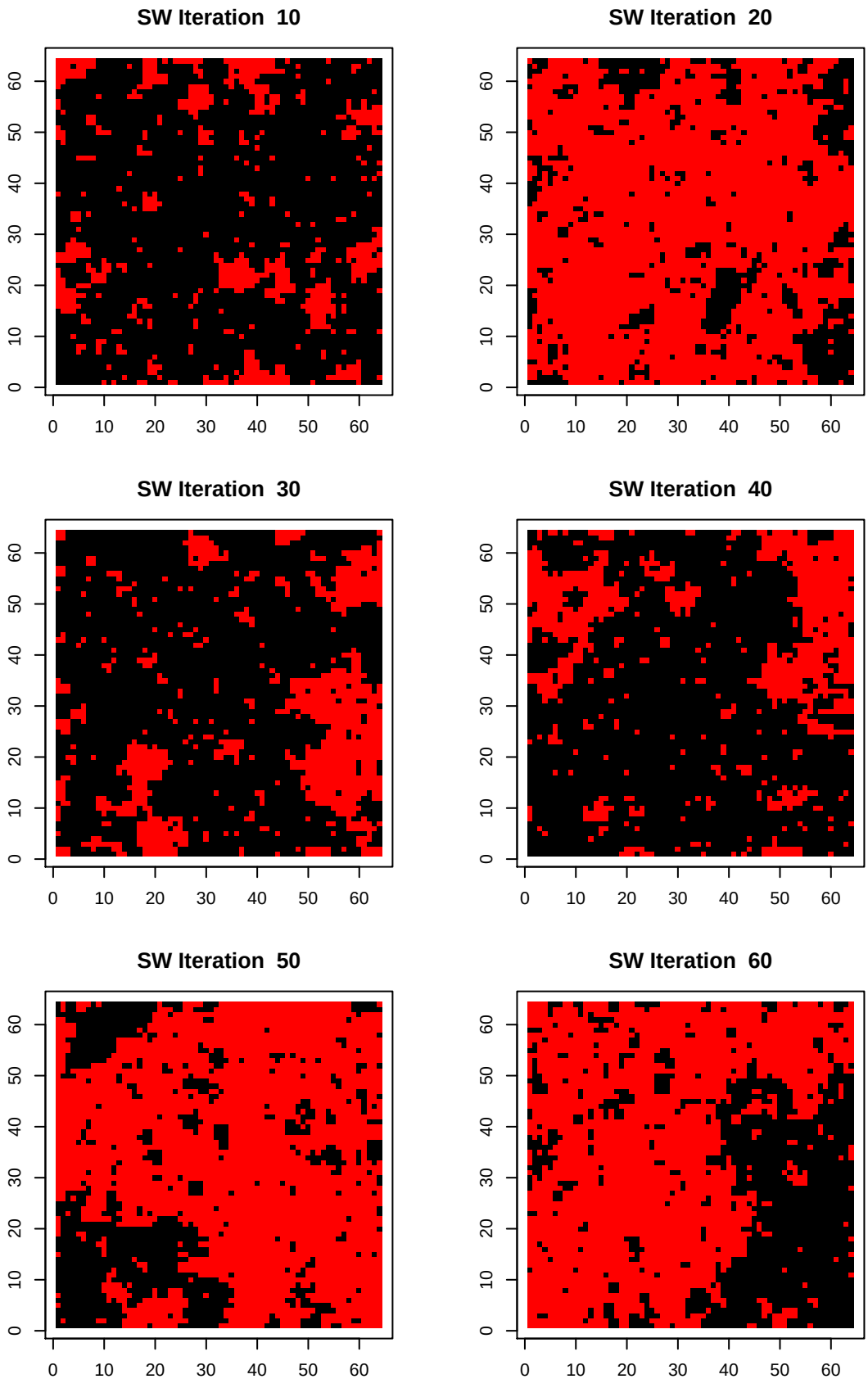
```
run.Gibbs<-function(N,beta.val,nreps,plot.out=T){  #Gibbs sampler for Ising Model
    t0<-proc.time()[3]
    Kval<-beta.val
    pval<-1-exp(-Kval)
    count.z<-c(0,0)
    Match.vec.Gibbs<-rep(0,nreps)

    Bond.list<-numeric()
    for(i in 1:N){
        for(j in 1:N){
            if(i+1 <= N){
                blist<-c(i,j,i+1,j)
                Bond.list<-rbind(Bond.list,blist)
            }
            if(j+1 <= N){
                blist<-c(i,j,i,j+1)
                Bond.list<-rbind(Bond.list,blist)
            }
        }
    }
    Nbonds<-nrow(Bond.list)
    Bonds<-rep(0,Nbonds)

    Grid.X<-rep(c(1:N),N); Grid.Y<-rep(c(1:N),each=N)
    Xvals<-c(1:N); Yvals<-c(1:N)
    par(pty="s",mfrow=c(3,2),mar=c(3,3,3,3))
    for(irep in 1:nreps){
        if(irep == 1){
            New.Grid.Z<-rbinom(Nsq,1,0.5)
        }
            Grid.Z<-matrix(New.Grid.Z,nrow=N,ncol=N)
        count.z[2]<-Grid.Z[1,2]+Grid.Z[2,1]
        count.z[1]<-2-count.z[2]
        prob.z<-exp(Kval*count.z[Grid.Z[1,1]+1])/sum(exp(Kval*count.z))
        Grid.Z[1,1]<-sample(c(Grid.Z[1,1],1-Grid.Z[1,1]),prob=c(prob.z,1-prob.z),size=1)
        for(j in 2:(N-1)){
            count.z[2]<-Grid.Z[1,j-1]+Grid.Z[1,j+1]+Grid.Z[2,j]
            count.z[1]<-3-count.z[2]
            prob.z<-exp(Kval*count.z[Grid.Z[1,j]+1])/sum(exp(Kval*count.z))
            Grid.Z[1,j]<-sample(c(Grid.Z[1,j],1-Grid.Z[1,j]),prob=c(prob.z,1-prob.z),size=1)
        }
        count.z[2]<-Grid.Z[1,N-1]+Grid.Z[2,N-1]
        count.z[1]<-2-count.z[2]
        prob.z<-exp(Kval*count.z[Grid.Z[1,N]+1])/sum(exp(Kval*count.z))
        Grid.Z[1,N]<-sample(c(Grid.Z[1,N],1-Grid.Z[1,N]),prob=c(prob.z,1-prob.z),size=1)
        for(i in 2:(N-1)){
            count.z[2]<-Grid.Z[i,j+1]+Grid.Z[i-1,j]+Grid.Z[i+1,j]
            count.z[1]<-3-count.z[2]
                    prob.z<-exp(Kval*count.z[Grid.Z[i,1]+1])/sum(exp(Kval*count.z))
            Grid.Z[i,1]<-sample(c(Grid.Z[i,1],1-Grid.Z[i,1]),prob=c(prob.z,1-prob.z),size=1)
            for(j in 2:(N-1)){
                count.z[2]<-Grid.Z[i,j-1]+Grid.Z[i,j+1]+Grid.Z[i-1,j]+Grid.Z[i+1,j]
                count.z[1]<-4-count.z[2]
                prob.z<-exp(Kval*count.z[Grid.Z[i,j]+1])/sum(exp(Kval*count.z))
                Grid.Z[i,j]<-sample(c(Grid.Z[i,j],1-Grid.Z[i,j]),prob=c(prob.z,1-prob.z),size=1)
            }
            count.z[2]<-Grid.Z[i,j-1]+Grid.Z[i-1,j]+Grid.Z[i+1,j]
```

```
            count.z[1]<-3-count.z[2]
            prob.z<-exp(Kval*count.z[Grid.Z[i,N]+1])/sum(exp(Kval*count.z))
            Grid.Z[i,N]<-sample(c(Grid.Z[i,N],1-Grid.Z[i,N]),prob=c(prob.z,1-prob.z),size=1)
        }
        count.z[2]<-Grid.Z[N,2]+Grid.Z[N-1,1]
        count.z[1]<-2-count.z[2]
        prob.z<-exp(Kval*count.z[Grid.Z[N,1]+1])/sum(exp(Kval*count.z))
        Grid.Z[N,1]<-sample(c(Grid.Z[N,1],1-Grid.Z[N,1]),prob=c(prob.z,1-prob.z),size=1)
        for(j in 2:(N-1)){
            count.z[2]<-Grid.Z[N,j-1]+Grid.Z[N,j+1]+Grid.Z[N-1,j]
            count.z[1]<-3-count.z[2]
            prob.z<-exp(Kval*count.z[Grid.Z[N,j]+1])/sum(exp(Kval*count.z))
            Grid.Z[N,j]<-sample(c(Grid.Z[N,j],1-Grid.Z[N,j]),prob=c(prob.z,1-prob.z),size=1)
        }
        count.z[2]<-Grid.Z[N,N-1]+Grid.Z[N-1,N]
        count.z[1]<-2-count.z[2]
        prob.z<-exp(Kval*count.z[Grid.Z[N,N]+1])/sum(exp(Kval*count.z))
        Grid.Z[N,N]<-sample(c(Grid.Z[N,N],1-Grid.Z[N,N]),prob=c(prob.z,1-prob.z),size=1)

        if(plot.out & irep %% 10 == 0){
            plot(Grid.X,Grid.Y,type="n",xlab="X",ylab="Y")
            title(paste('Gibbs Iteration ',irep))
            image(Xvals,Yvals,Grid.Z,col=c("black","red"),add=T)
        }
        New.Grid.Z<-as.vector(Grid.Z)

        Matches<-sum(as.numeric(apply(Bond.list,1,function(x){
                    return(Grid.Z[x[1],x[2]]==Grid.Z[x[3],x[4]])})))
        Match.vec.Gibbs[irep]<-Matches
    }
    t1<-proc.time()[3]
    return(list(match=Match.vec.Gibbs,time=t1-t0))
}

run.SW<-function(N,beta.val,nreps,plot.out=T){  #Swendsen-Wang algorithm for Ising Model
    t0<-proc.time()[3]
    Kval<-beta.val
    pval<-1-exp(-Kval)
    Match.vec.SW<-rep(0,nreps)
    Bond.list<-numeric()
    for(i in 1:N){
        for(j in 1:N){
            if(i+1 <= N){
                blist<-c(i,j,i+1,j)
                Bond.list<-rbind(Bond.list,blist)
            }
            if(j+1 <= N){
                blist<-c(i,j,i,j+1)
                Bond.list<-rbind(Bond.list,blist)
            }
        }
    }
    Nbonds<-nrow(Bond.list)
    Bonds<-rep(0,Nbonds)
    Grid.X<-rep(c(1:N),N); Grid.Y<-rep(c(1:N),each=N)
    Xvals<-c(1:N); Yvals<-c(1:N)
    par(pty="s",mfrow=c(3,2),mar=c(3,3,3,3))
    for(irep in 1:nreps){
        if(irep == 1){
            Grid.Z<-rbinom(Nsq,1,0.5)
        }else{
```

```
            Grid.Z<-New.Grid.Z
        }
        New.Z<-matrix(Grid.Z,nrow=N,ncol=N)

        Match<-as.numeric(apply(Bond.list,1,function(x){return(New.Z[x[1],x[2]]==New.Z[x[3],x[4]])}))
        Bonds<-Match
        Bond.Z<-rbinom(length(Match),1,pval)*Match
        Matches<-sum(Match)
        Match.vec.SW[irep]<-Matches
            Clust.mat<-matrix(0,N,N)
            Clust.mat[1,1]<-1
        nclust<-1
        for(i in 1:Nbonds){
            i1<-Bond.list[i,1]
            j1<-Bond.list[i,2]
            i2<-Bond.list[i,3]
            j2<-Bond.list[i,4]
            if(Clust.mat[i1,j1]==0){
                nclust<-nclust+1
                Clust.mat[i1,j1]<-nclust
            }
            if(Bond.Z[i]==1){
                if(Clust.mat[i2,j2] == 0){
                    Clust.mat[i2,j2]<-Clust.mat[i1,j1]
                }else{
                    label1<-Clust.mat[i1,j1]
                    label2<-Clust.mat[i2,j2]
                    new.label<-min(label1,label2)
                    Clust.mat[Clust.mat==label1]<-new.label
                    Clust.mat[Clust.mat==label2]<-new.label
                }
            }
        }

        if(Clust.mat[N,N]==0){Clust.mat[N,N]<-nclust+1;nclust<-nclust+1}

        Clust.vec<-as.vector(Clust.mat)
        Cluster.list<-c(1:max(Clust.vec))
        Clust.col<-rbinom(length(Cluster.list),1,0.5)
        Grid.Z.vec<-Clust.col[Clust.vec]

        New.Grid.Z<-matrix(Grid.Z.vec,N,N)

        if(plot.out & irep %% 10 == 0){
            plot(Grid.X,Grid.Y,type="n",xlab="X",ylab="Y")
            title(paste('SW Iteration ',irep))
            image(Xvals,Yvals,New.Grid.Z,col=c("black","red"),add=T)
        }
    }
    t1<-proc.time()[3]
    return(list(match=Match.vec.SW,time=t1-t0))
}
```

```
#Run
N<-64
Nsq<-N*N
match.G<-run.Gibbs(N,0.9,60);match.S<-run.SW(N,0.9,60)
```


Gibbs Iteration 10
Gibbs Iteration 20
Gibbs Iteration 30
Gibbs Iteration 40
Gibbs Iteration 50
Gibbs Iteration 60
0 10 20 30 40 50 60


SW Iteration 10
SW Iteration 20
SW Iteration 30
SW Iteration 40
SW Iteration 50
SW Iteration 60
0 10 20 30 40 50 60

Recall that the Ising model has

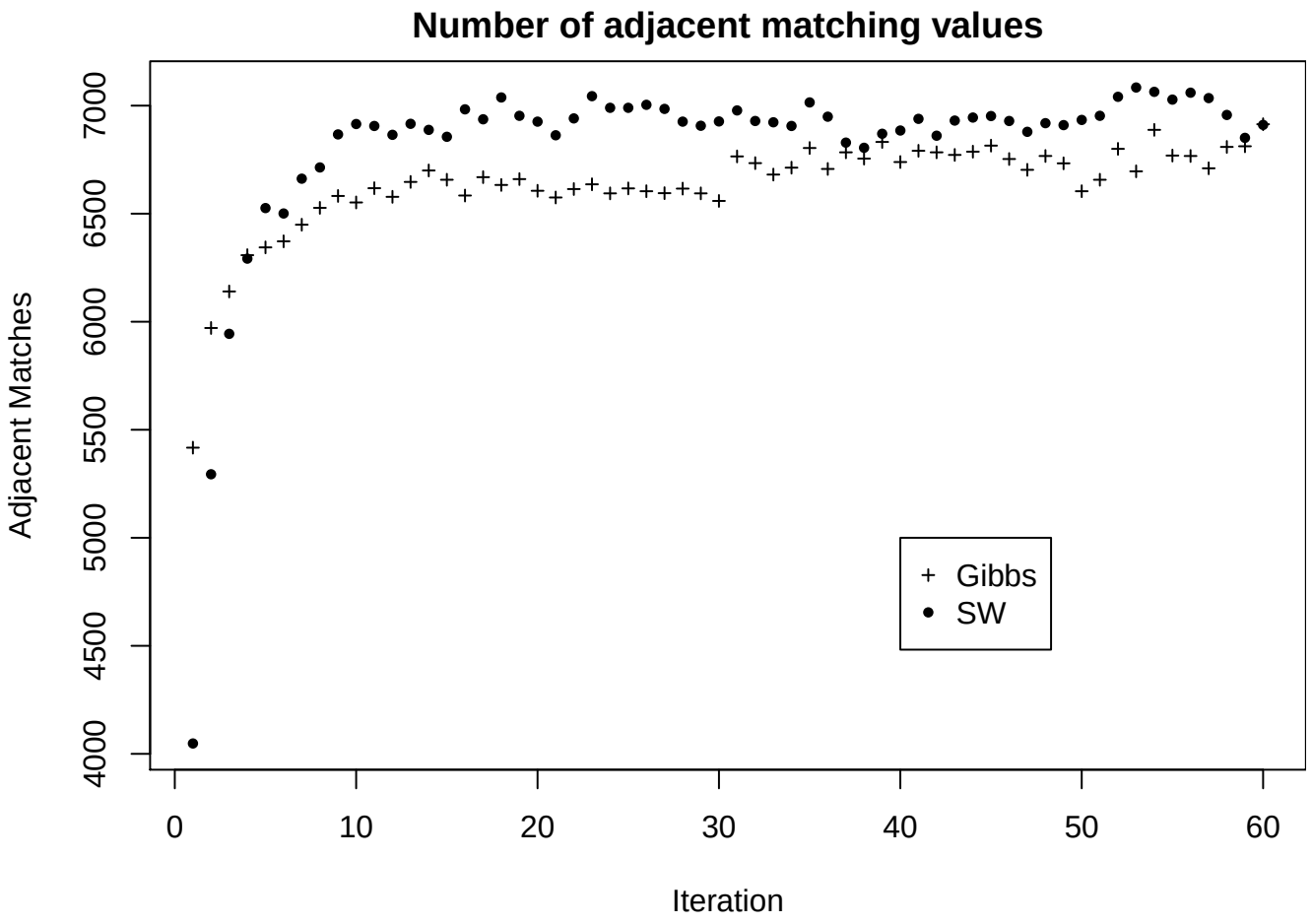
$$\log \pi_{\mathbf{X}}(\mathbf{x}) = \beta \sum_{i=1}^{NM} \sum_{j \in \partial i} \mathbb{1}_{\{x_i\}}(x_j) - \log Z(\beta)$$

so we may track the quantity

$$\sum_{i=1}^{NM} \sum_{j \in \partial i} \mathbb{1}_{\{x_i\}}(x_j)$$

which is the number of adjacent matching values to give an indication of the posterior density value being sampled.

```
print(c(match.G$time,match.S$time))

+ elapsed elapsed
+    5.46    7.99

par(mfrow=c(1,1),mar=c(4,4,2,0))
plot(1:60,match.S$match,xlab='Iteration',ylab='Adjacent Matches',pch=19,cex=0.6,
       ylim=range(c(match.S$match,match.G$match)))
points(1:60,match.G$match,pch=3,cex=0.6)
legend(40,5000,c('Gibbs','SW'),pch=c(3,19),pt.cex=c(0.6,0.6))
title('Number of adjacent matching values')
```



Example: Bayesian analysis of Finite Mixture Models
Suppose that $Y_1 \ldots, Y_n$ are a conditionally i.i.d. from the K component finite mixture model for $y \in \mathbb{R}$

$$f_Y(y;\boldsymbol{\theta}) = \sum_{k=1}^{K} \omega_k f_k(y;\theta_k) \qquad\qquad \sum_{k=1}^{K} \omega_k = 1$$

for $0 < \omega_k < 1$ where $f_1, \ldots, f_K$ are component densities. The likelihood is

$$\mathcal{L}_n(\boldsymbol{\theta}) = \prod_{i=1}^{n} \left\{ \sum_{k=1}^{K} \omega_k f_k(y_i;\theta_k) \right\}$$

For a Bayesian analysis, we specify a prior distribution

$$\pi_0(\boldsymbol{\theta}) = p(\omega_1, \ldots, \omega_K, \theta_1, \ldots, \theta_K)$$

which yields a posterior which is not analytically tractable. Here

$$\pi_0(\boldsymbol{\theta}) \propto \mathcal{L}_n(\boldsymbol{\theta})\pi_0(\boldsymbol{\theta}).$$

Consider the discrete auxiliary variables $U_1, \ldots, U_n$ introduced into the likelihood density for each i via the identity

$$f_Y(y;\boldsymbol{\theta}) = \sum_{k=1}^{K} f_{Y|U}(y|u;\boldsymbol{\theta}) f_U(u)$$

where U has the $Multinomial(1, \omega_1, \ldots, \omega_K)$ distribution

$$f_U(u) = \prod_{k=1}^{K} \omega_k^{\mathbb{1}_{\{k\}}(u)}$$

and $f_{Y|U,\boldsymbol{\theta}}(y|u,\boldsymbol{\theta}) \equiv f_u(y|\boldsymbol{\theta})$. The U variables are discrete labels indicating which of the component densities, $f_1, \ldots, f_K$, each data point is sampled from, with

$$\Pr[U = k] = \omega_k \qquad k = 1, \ldots, K$$

Consider the joint posterior distribution on the unknown (unobservable) quantities

$$\pi_{\boldsymbol{\theta},\mathbf{U}}(\boldsymbol{\theta}, \mathbf{u}|\mathbf{y}) \propto \mathcal{L}_n(\boldsymbol{\theta}, \mathbf{u})\pi_0(\boldsymbol{\theta}, \mathbf{u})$$

Here

$$\pi_0(\boldsymbol{\theta}, \mathbf{u}) = \pi_0(\boldsymbol{\theta})\pi_0(\mathbf{u}|\boldsymbol{\theta}) = \pi_0(\boldsymbol{\theta}) \prod_{i=1}^{n} \left\{ \prod_{k=1}^{K} \omega_k^{\mathbb{1}_{\{k\}}(u)} \right\}$$

where $\pi_0(\boldsymbol{\theta})$ is the joint prior on the ω and θ variables. Running the Gibbs sampler on the full conditional distributions

$$\pi_{\boldsymbol{\theta}|\mathbf{U}}(\boldsymbol{\theta}|\mathbf{u}, \mathbf{y}) \qquad\qquad \pi_{\mathbf{U}|\boldsymbol{\theta}}(\mathbf{u}|\boldsymbol{\theta}, \mathbf{y})$$

yields samples from the joint posterior. Marginally, the Gibbs sampler produces a sample from

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\mathbf{y})$$

which is the target posterior of interest.

For the first conditional, given $\mathbf{U} = \mathbf{u}$, we have a factorization

$$\pi_{\boldsymbol{\theta}|\mathbf{U}}(\boldsymbol{\theta}|\mathbf{u}, \mathbf{y}) = \pi_{\theta_1,\ldots,\theta_K|\mathbf{U}}(\theta_1, \ldots, \theta_K|\mathbf{u}, \mathbf{y})\pi_{\omega|\mathbf{U}}(\omega_1, \ldots, \omega_K|\mathbf{u}, \mathbf{y})$$

and can update these parameters independently in two parts.

- $\pi_{\boldsymbol{\theta}|\mathbf{U}}(\boldsymbol{\theta}|\mathbf{u},\mathbf{y})$ (Part I): if the prior on the θ parameters factorizes into independent components, then

$$\pi_{\theta_1,\ldots,\theta_K|\mathbf{U}}(\theta_1,\ldots,\theta_K|\mathbf{u},\mathbf{y}) = \prod_{k=1}^{K} \pi_{\theta_k|\mathbf{U}}(\theta_k|\mathbf{u},\mathbf{y})$$

  where

$$\pi_{\theta_k|\mathbf{U}}(\theta_k|\mathbf{u},\mathbf{y}) \propto \left\{ \prod_{i:u_i=k} f_k(y_i|\theta_k) \right\} p(\theta_k)$$

  and we can update the θ parameters independently conditional on $\mathbf{u}$. This may require a Metropolis-Hastings update.

- $\pi_{\boldsymbol{\theta}|\mathbf{U}}(\boldsymbol{\theta}|\mathbf{u},\mathbf{y})$ (Part II): a standard choice for the prior for the ω parameters is the Dirichlet prior

$$\begin{aligned} p(\omega_1,\ldots,\omega_K) &\propto \prod_{k=1}^{K} \omega_k^{\alpha_k-1} \\ \implies p(\omega_1,\ldots,\omega_K|\mathbf{u},\mathbf{y}) &\propto \prod_{k=1}^{K} \omega_k^{n_k+\alpha_k-1} \end{aligned}$$

  where, for $k = 1,\ldots,K$, $n_k = \sum_{i=1}^{n} \mathbb{1}_{\{k\}}(u_i)$ is the count of the U variables that are set equal to k. Thus the full conditional distribution is also Dirichlet, and can be sampled directly.

- $\pi_{\mathbf{U}|\boldsymbol{\theta}}(\mathbf{u}|\boldsymbol{\theta},\mathbf{y})$: by conditional independence of the Y given $\boldsymbol{\theta}$, we must have

$$\pi_{\mathbf{U}|\boldsymbol{\theta}}(\mathbf{u}|\boldsymbol{\theta},\mathbf{y}) = \prod_{i=1}^{n} \pi_{U_i|\boldsymbol{\theta}}(u_i|\boldsymbol{\theta},y_i)$$

  where, by Bayes theorem

$$\pi_{U_i|\boldsymbol{\theta}}(u_i|\boldsymbol{\theta},y_i) = \frac{f_{u_i}(y_i;\boldsymbol{\theta}) f_{U_i}(u_i;\boldsymbol{\theta})}{\sum_{k=1}^{K} f_k(y_i;\boldsymbol{\theta}) f_{U_i}(k;\boldsymbol{\theta})} = \frac{f_{u_i}(y_i;\theta_{u_i})\omega_{u_i}}{\sum_{k=1}^{K} f_k(y_i;\theta_k)\omega_k}$$

  which is a discrete distribution on $\{1,\ldots,K\}$. Thus updating the U variables is straightforward.

This formulation reveals a connection with the EM algorithm; the full conditional distribution of U_i is identical to the conditional distribution of the missing data, given the observed data. Indeed, the entire formulation is similar, the only difference is that we replace the Expectation and Maximization steps by sampling steps. The Gibbs sampler does not perform maximization to get the ML estimate, it produces samples from the posterior distribution of the parameters.

Note: Some care is needed in interpreting the Gibbs sampler output, as the component labels are, in fact, arbitrarily assigned in the mixture formulation. Therefore, the U variables can exhibit *label-switching*. A simple solution is to add an additional Metropolis-Hastings step at the end of each Gibbs sampler iteration that permutes the labels on the U variables, that is, for $K = 3$, if the random permutation is $(3,1,2)$, then

- All Us labelled 1 are relabelled 3
- All Us labelled 2 are relabelled 1
- All Us labelled 3 are relabelled 2

This Metropolis-Hastings proposal is **always accepted**, as it does not change the posterior density value.

For a specific example, consider $n = 200$ data generated from the two component mixture pdf

$$f_Y(y;\theta) = \omega_1 f_1(y;\theta_1) + \omega_2 f_2(y;\theta_2)$$

where specifically

- $f_1(y;\theta_1) \equiv Normal(\mu_1, \sigma_1^2)$, with $\theta_1 = (\mu_1, \sigma_1^2)$;
- $f_2(y;\theta_2) \equiv Normal(\mu_2, \sigma_2^2)$, with $\theta_2 = (\mu_2, \sigma_2^2)$;
- $0 < \omega_1, \omega_2 < 1, \omega_1 + \omega_2 = 1$.

This is a five parameter model. We introduce the auxiliary variables $U_1, \ldots, U_n$ as described above to indicate the mixture component from which each of the data originates. Using a conjugate prior structure

$$\pi_0(\mu_1, \sigma_1^2, \mu_2, \sigma_2^2, \omega_1) = \pi_0(\mu_1|\sigma_1^2)\pi_0(\sigma_1^2)\pi_0(\mu_2|\sigma_2^2)\pi_0(\sigma_2^2)\pi_0(\omega_1)$$

where for $k = 1, 2$

- $\pi_0(\sigma_k^2) \equiv InvGamma(\alpha/2, \beta/2)$
- $\pi_0(\mu_k|\sigma_k^2) \equiv Normal(b, \sigma_k^2/\lambda)$
- $\pi_0(\omega_1) \equiv Uniform(0, 1)$.

In the Gibbs sampler, we may sample from the full conditional posterior distributions for these parameters, and under this prior, the full conditionals are also available in closed form: for $k = 1, 2$

- $\pi_n(\sigma_k^2) \equiv InvGamma(\alpha_{nk}/2, \beta_{nk}/2)$ where

$$\alpha_{nk} = \alpha + n_k \qquad \beta_{nk} = \beta + \sum_{i=1}^{n} \mathbb{1}_{\{k\}}(u_i)(y_i - \overline{y}_k)^2 + \frac{n_k \lambda}{n_k + \lambda}(\overline{y}_k - b)^2$$

  and

$$n_k = \sum_{i=1}^{n} \mathbb{1}_{\{k\}}(u_i) \qquad \overline{y}_k = \frac{1}{n_k} \sum_{i=1}^{n} \mathbb{1}_{\{k\}}(u_i)$$

  are the component-specific summary statistics conditional on the auxiliary variables.

- $\pi_n(\mu_k|\sigma_k^2) \equiv Normal(b_{nk}, \sigma_k^2/\lambda_{nk})$ where

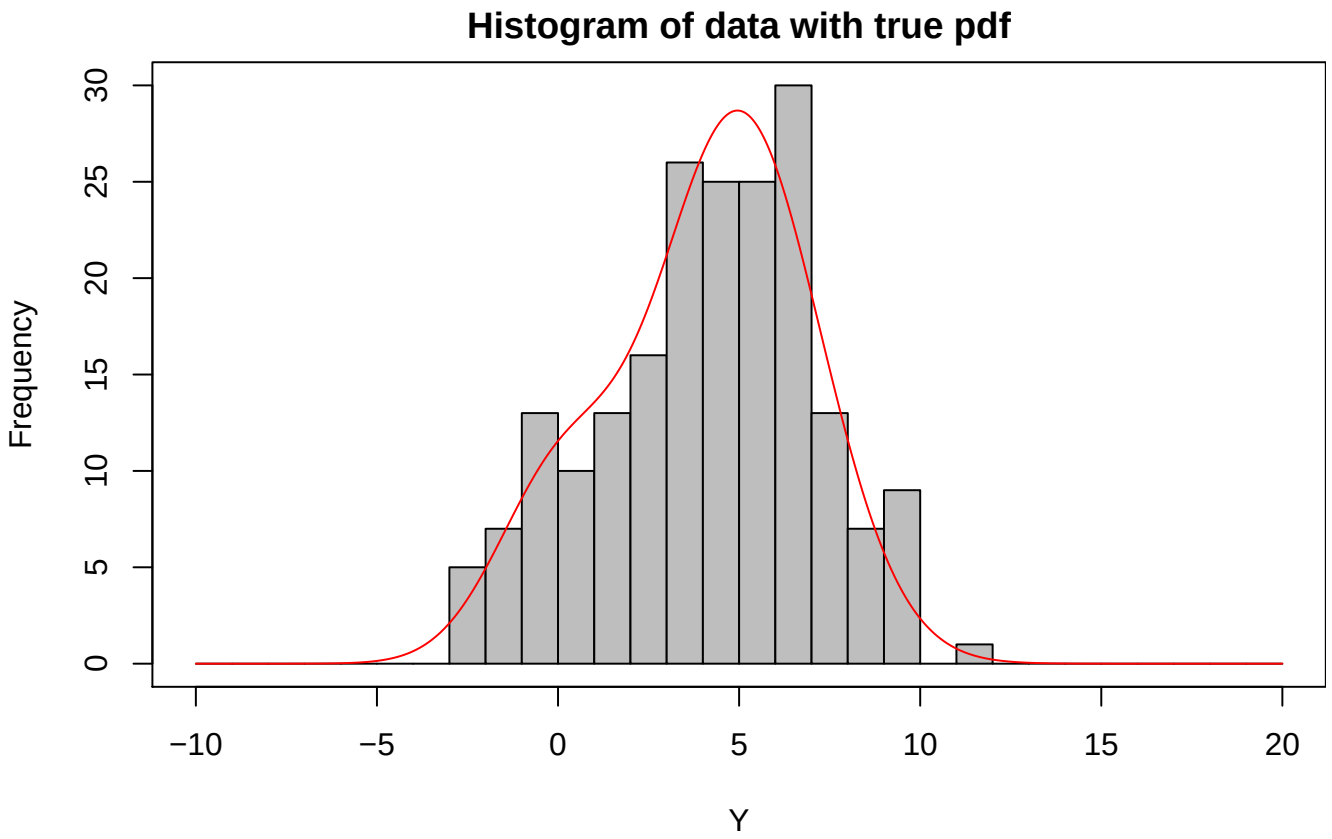
$$b_{nk} = \frac{n_k \overline{y}_k + \lambda b}{n_k + \lambda} \qquad \lambda_{nk} = \lambda + n_k$$

- $\pi_n(\omega_1) \equiv Beta(n_1 + 1, n_2 + 1)$.

In the following simulation, we have

$$\omega_1 = 0.2 \qquad \mu_1 = 0 \qquad \sigma_1^2 = 3 \qquad \mu_2 = 5 \qquad \sigma_2^2 = 5.$$

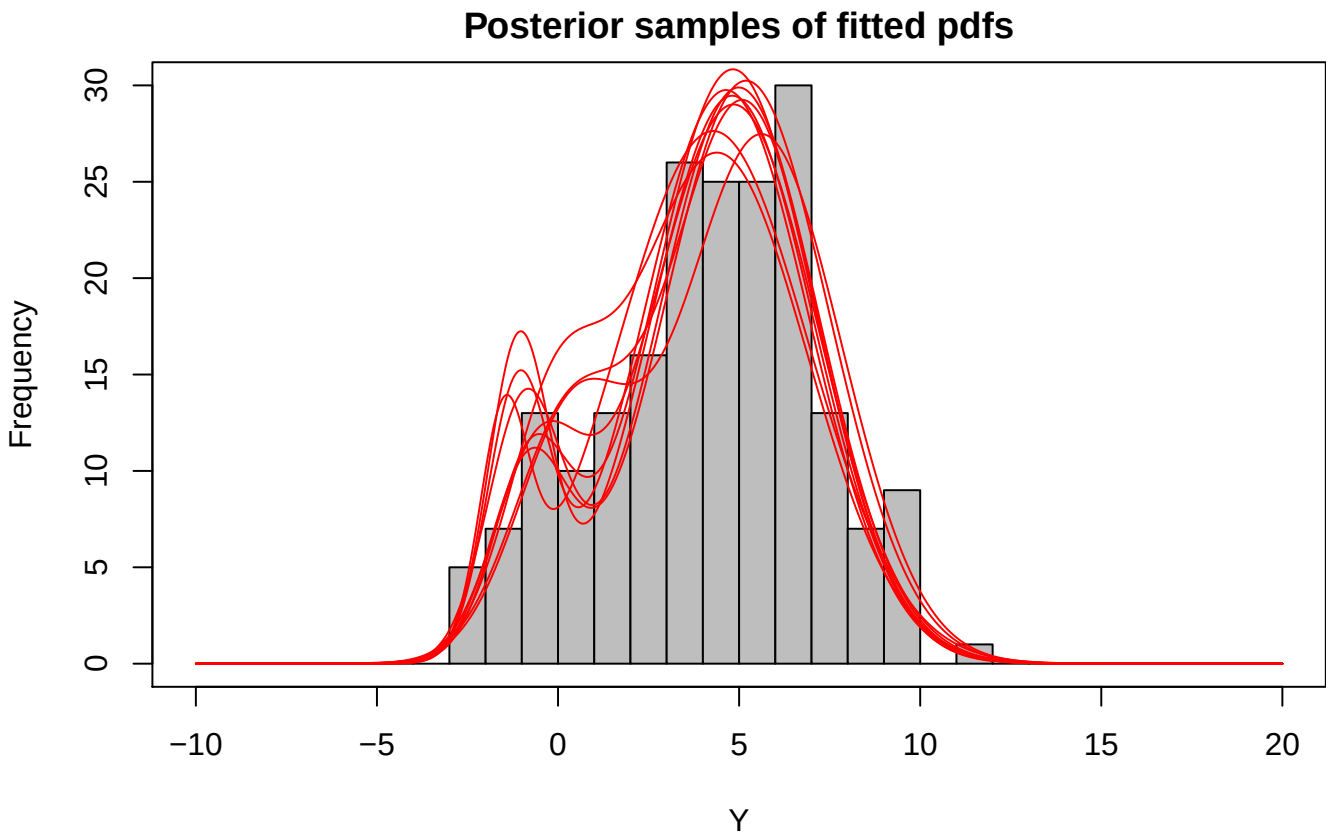
```
set.seed(2342)
n<-200
K<-2
omega.vec<-c(0.2,0.8);mu.vec<-c(0,5);sig.vec<-c(3,5)

U<-sample(c(1:2),size=n,rep=T,prob=omega.vec)
Y<-rnorm(n,mu.vec[U],sqrt(sig.vec[U]))

xv<-seq(-10,20,by=0.01)
yv<-xv*0
for(k in 1:K){
        yv<-yv+omega.vec[k]*dnorm(xv,mu.vec[k],sqrt(sig.vec[k]))
}
par(mar=c(4,4,2,0))
hist(Y,breaks=seq(-10,20,by=1),col='gray',main='Histogram of data with true pdf');box()
lines(xv,yv*n*1,col='red')
```



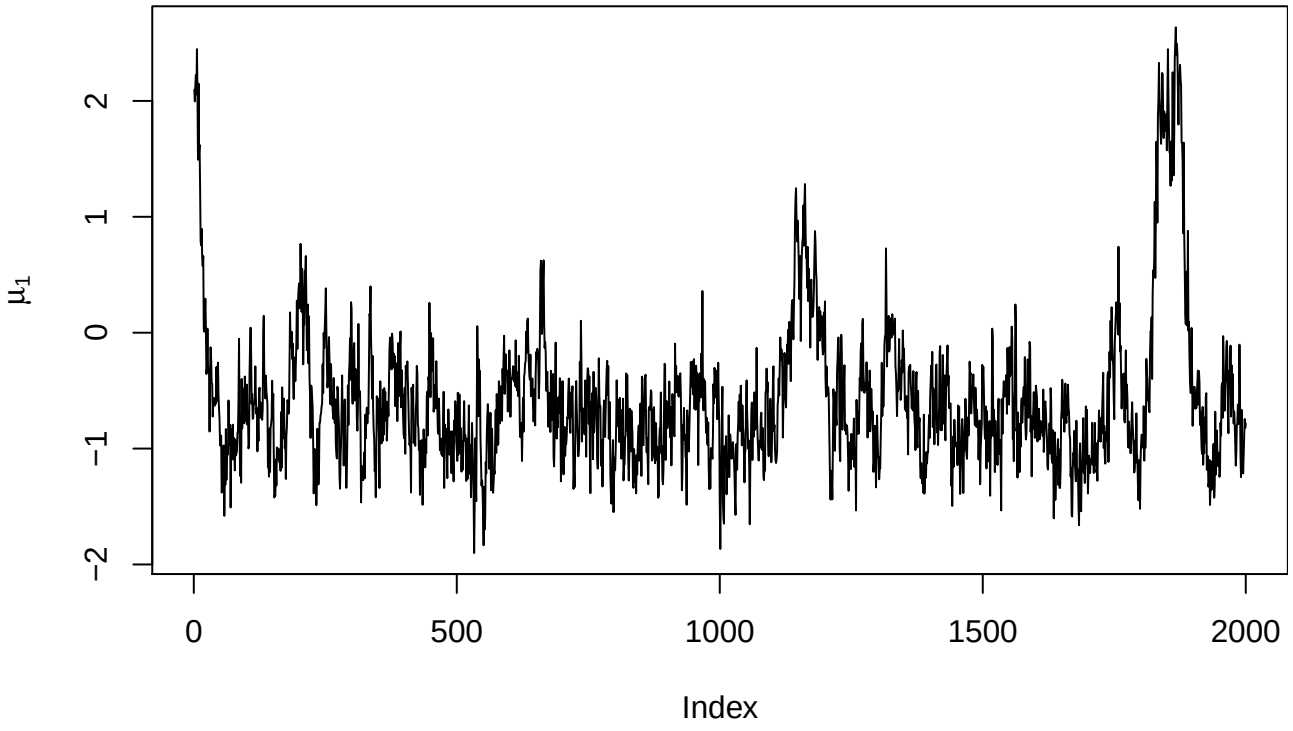
The Gibbs sampler is run for 2000 iterations.

```
old.U<-sample(c(1:2),size=n,rep=T)
old.mu<-quantile(Y,prob=c(0.25,0.75))
old.sigsq<-c(4,4)
old.omega<-rep(1/K,K)

nreps<-2000

al.prior<-5
be.prior<-5
b.prior<-0
lam.prior<-0.01

mu.samples<-sig.samples<-omega.samples<-matrix(0,nrow=nreps,ncol=K)
log.posterior<-rep(0,nreps)

dinvgamma<-function(x,al,be,log=T){

        fv<-be*log(al)-lgamma(al)-(al+1)*log(x)-be/x
        if(!log){fv<-exp(xv)}
        return(fv)

}

par(mar=c(4,4,2,0))
hist(Y,breaks=seq(-10,20,by=1),col='gray',main='Posterior samples of fitted pdfs');box()


for(irep in 1:nreps){
        Mu.mat<-matrix(rep(old.mu,n),ncol=K,byrow=T)
        Sig.mat<-matrix(rep(sqrt(old.sigsq),n),ncol=K,byrow=T)
```

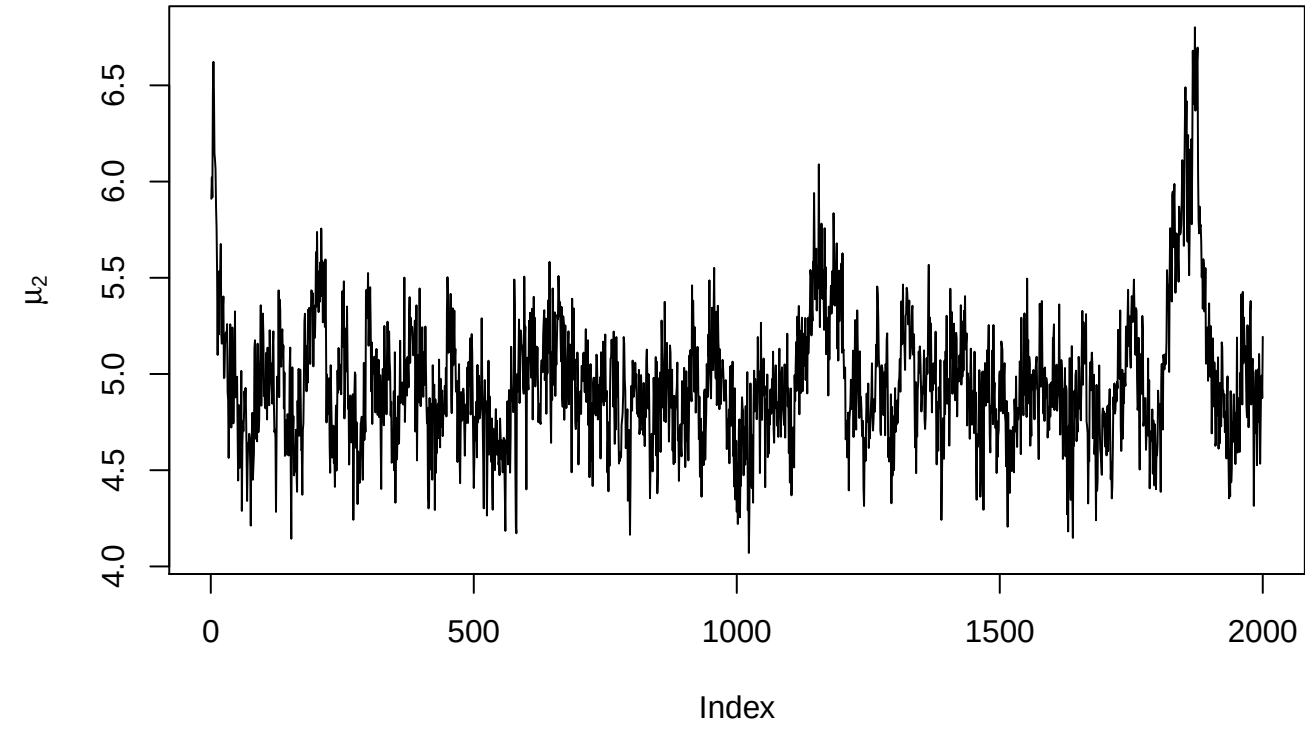
```
        Omega.mat<-matrix(rep(old.omega,n),ncol=K,byrow=T)
        Bmat<-exp(-0.5*((Y-Mu.mat)/Sig.mat)^2)*Omega.mat/Sig.mat
        Btot<-apply(Bmat,1,sum)
        Bmat<-Bmat/Btot

        old.U<-apply(Bmat,1,function(x){sample(c(1:K),size=1,prob=x)})

        nvec<-rep(0,K)
        for(k in 1:K){
                Ysub<-Y[old.U == k]
                nk<-length(Ysub)
                nvec[k]<-nk
                if(length(nk) > 0){
                        ybar<-mean(Ysub)
                        yssq<-sum((Ysub-ybar)^2)
                        al.post<-nk+al.prior
                        be.post<-yssq+be.prior+nk*lam.prior*(ybar-b.prior)^2/(nk+lam.prior)
                        old.sigsq[k]<-1/rgamma(1,al.post/2,be.post/2)
                        b.post<-(nk*ybar+lam.prior*b.prior)/(nk+lam.prior)
                        tau.post<-old.sigsq[k]/(nk+lam.prior)
                        old.mu[k]<-rnorm(1,b.post,sqrt(tau.post))
                }else{
                        old.sigsq[k]<-1/rgamma(1,al.prior/2,be.prior/2)
                        old.mu[k]<-rnorm(1,b.prior,sqrt(old.sigsq[k]/lam.prior))
                }
                mu.samples[irep,]<-old.mu
                sig.samples[irep,]<-old.sigsq
        }

        old.omega<-rgamma(K,nvec+1,1)
        old.omega<-old.omega/sum(old.omega)
        omega.samples[irep,]<-old.omega

        Mu.mat<-matrix(rep(old.mu,n),ncol=K,byrow=T)
        Sig.mat<-matrix(rep(sqrt(old.sigsq),n),ncol=K,byrow=T)
        Omega.mat<-matrix(rep(old.omega,n),ncol=K,byrow=T)
        Bmat<-exp(-0.5*((Y-Mu.mat)/Sig.mat)^2)*Omega.mat/Sig.mat

        log.posterior[irep]<-sum(log(apply(Bmat,1,sum)))+
                        sum(dnorm(old.mu,b.prior,sqrt(old.sigsq/lam.prior),log=T))+
                              sum(dinvgamma(old.sigsq,al.prior/2,be.prior/2,log=T))
        if(irep %% 200 == 0){
                yv<-xv*0
                for(k in 1:K){
                        yv<-yv+old.omega[k]*dnorm(xv,old.mu[k],sqrt(old.sigsq[k]))
                }
                lines(xv,yv*n*1,col='red')
        }
}
```



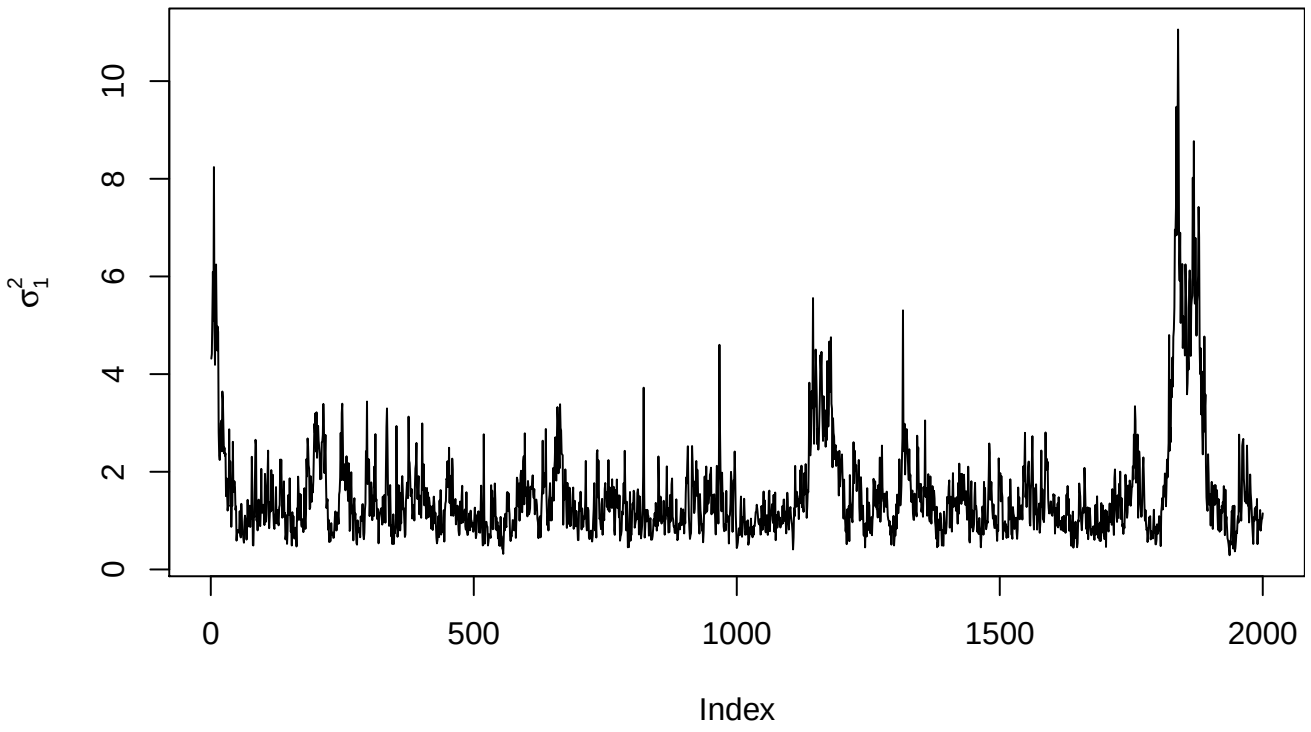

```
par(mar=c(4,5,1,0));plot(mu.samples[,1],type='l',ylab=expression(mu[1]))
```



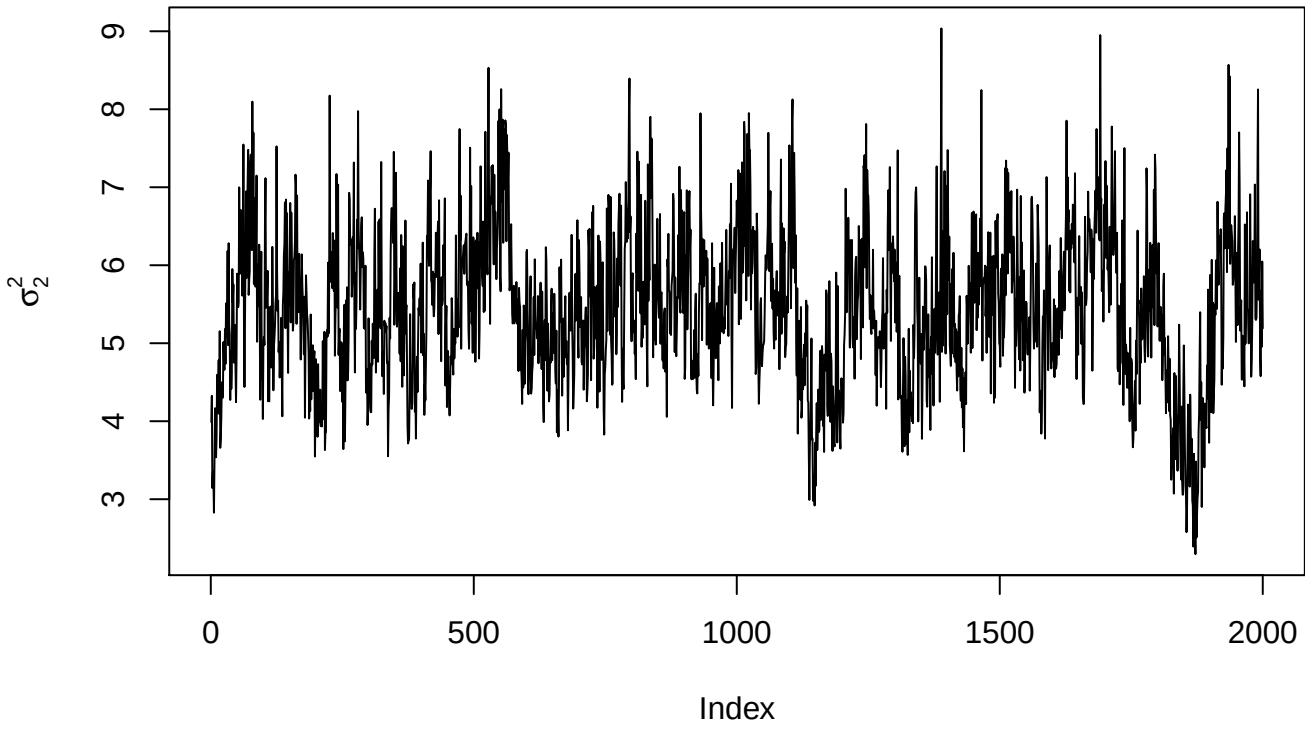
```
par(mar=c(4,5,1,0));plot(mu.samples[,2],type='l',ylab=expression(mu[2]))
```



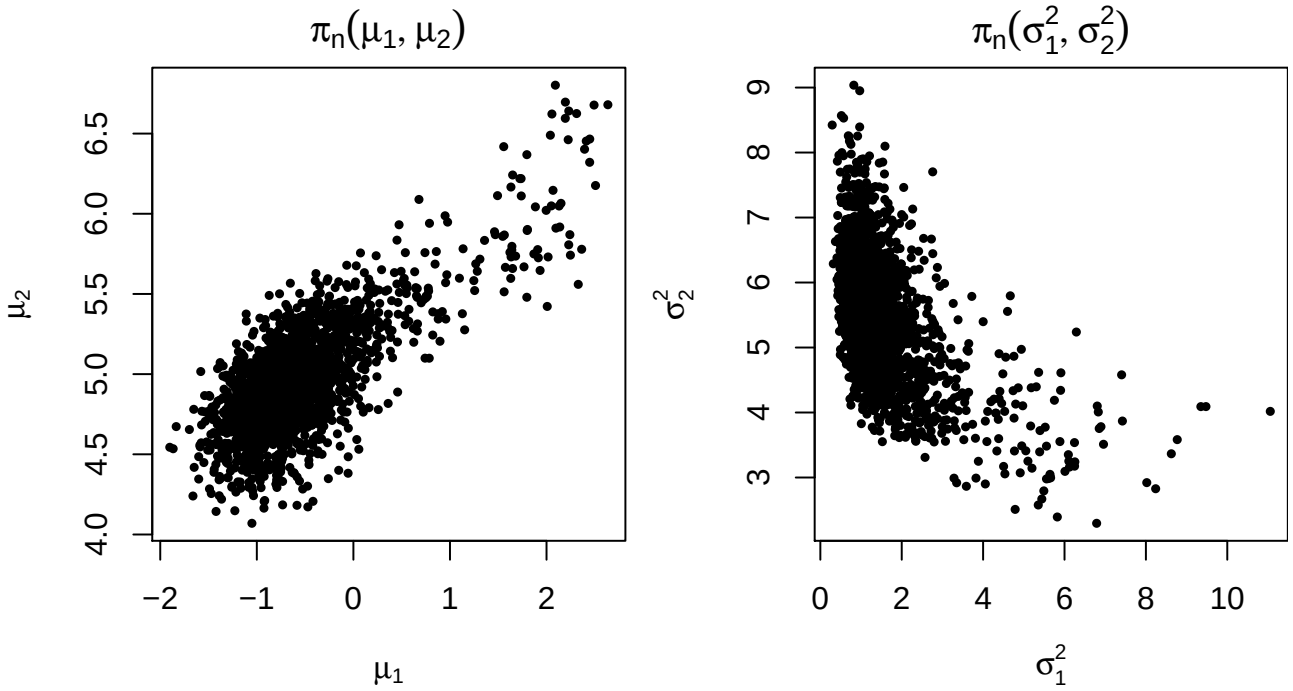

```
par(mar=c(4,5,1,0));plot(sig.samples[,1],type='l',ylab=expression(sigma[1]^2))
```



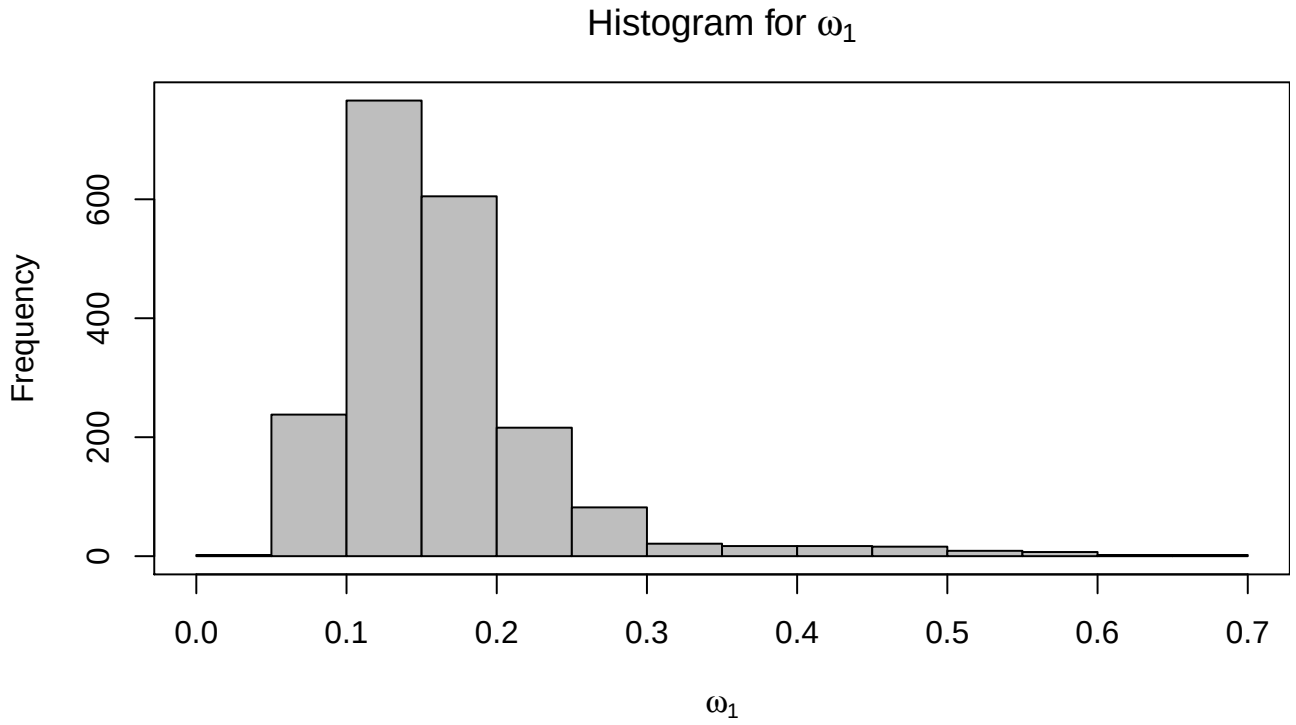

```
par(mar=c(4,5,1,0));plot(sig.samples[,2],type='l',ylab=expression(sigma[2]^2))
```




```
par(mar=c(4,5,2,0),pty='s',mfrow=c(1,2))
plot(mu.samples,pch=19,cex=0.5,xlab=expression(mu[1]),ylab=expression(mu[2]))
title(expression(pi[n](mu[1],mu[2])))
plot(sig.samples,pch=19,cex=0.5,xlab=expression(sigma[1]^2),ylab=expression(sigma[2]^2))
title(expression(pi[n](sigma[1]^2,sigma[2]^2)))
```



```
par(mar=c(4,5,3,0),pty='m',mfrow=c(1,1))
hist(omega.samples[,1],col='gray',main=expression(paste('Histogram for ',omega[1])),
        xlab=expression(omega[1]));box()
```




```
par(mar=c(4,5,2,0),pty='m',mfrow=c(1,1))
plot(log.posterior,type='l',xlab='Iteration',ylab='Log posterior')
```