Markov chain Monte Carlo

Adapted from notes by Murali Haran, Penn State University

May / June 2018, 14th Penn State Astrostatistics Summer School

Goal: Describe and implement a simple example of MCMC

Monte Carlo sampling methods, which we describe below, don't always work.

The purpose of this activity is twofold:

- Illustrate how MCMC algorithms are easy to implement, at least in principle, in situations where classical Monte Carlo methods do not work.
- Provide a glimpse of practical MCMC implementation issues.

Basic Monte Carlo Methods

Suppose we want to estimate $m = E_f g(X)$, the expectation of a function $g(x)$ with respect to the probability distribution f.

If m is analytically intractable, a Monte Carlo estimate of m can be obtained by

- simulating N pseudo-random values, say $X_1, X_2, \ldots, X_N$, from the distribution f
- taking the average of $g(X_1), g(X_2), \ldots, g(X_N)$ to estimate m.

Typically, the Law of Large Numbers implies that the estimate converges to the true expectation m as N gets large.

A toy example

Suppose X is a standard normal random variable and we wish to calculate $P(-1 < X < 1)$.

(This can be written in the form $E_f g(X)$ if we take f as the standard normal density and $g(x) = I\{-1 < x < 1\}$.)

```r
xs<-rnorm(10000) # simulate 10,000 draws from N(0,1)
xcount<-sum((xs>-1) & (xs<1)) # count draws in (-1,1)
xcount/10000 # Monte Carlo estimate
```

```
## [1] 0.687
```

```r
pnorm(1) - pnorm(-1) # Compare to exact value
```

```
## [1] 0.6826895
```

Importance sampling: Another MC technique for estimating expectations

- Goal: estimate $m = E_f g(X)$.
- Strategy: Sample $Y_1, \ldots, Y_N$ from density q (NOT f).
- Notice:
$$E_f g(X) = E_q \left[g(Y) \frac{f(Y)}{q(Y)} \right].$$
- Conclusion: Estimate m by
$$\frac{1}{N} \sum_{i=1}^{N} W_i g(Y_i) \quad \text{where} \quad W_i = \frac{f(Y_i)}{q(Y_i)}.$$

When normalizing constants for f or q are unknown, or for numerical stability, the weights W_i may be scaled so that their mean is one. (NB: This should be done only when f and q have the same support, for in that case $E_q W_i = 1$.)

Toy Example 2: Calculate $P(4 < X)$ when $X \sim N(0,1)$

First, naive MC based on $E_f I\{4 < X\}$:

```
x <- rnorm(10000) # simulate 10,000 draws from N(0,1)
mean(x>4) # Estimate probability empirically
```

```
## [1] 0
```

Next, importance sampling using $q(y) = \exp(-(y-4))$ for $y > 4$.

```
y <- rexp(10000)+4 # Sample from q
mean(dnorm(y)/exp(4-y)) # All y are >4
```

```
## [1] 3.143678e-05
```

Here is the exact value:

```
1-pnorm(4) # Compare it to exact answer
```

```
## [1] 3.167124e-05
```

Importance sampling is powerful in a number of situations

- If expectations with respect to several different distributions (say $f_1, \ldots, f_p$) are of interest, all can be estimated from a single set of samples.
- For rare event probabilities, ordinary Monte Carlo could take a huge number of samples for accurate estimates. In such cases, selecting q wisely can give more precise estimates using fewer samples.
- In principle, "selecting q wisely" means trying to make it roughly proportional to $f \times g$.

Discussions of importance sampling in astronomical Bayesian computation appear in papers by Lewis & Bridle (http://xxx.lanl.gov/abs/astro-ph/0205436) and Trotta (http://arxiv.org/abs/0803.4089) for cosmological parameter estimation and Ford (http://xxx.lanl.gov/abs/astro-ph/0512634) for extrasolar planet modeling.

MCMC can help when f is intractable

- Producing pseudo-random draws from f is often infeasible.
- An alternative is to use a Metropolis-Hastings algorithm to produce a Markov chain $X_1, \dots, X_N$ where the X_i are **dependent** draws that are **approximately** from f.
- As before, the mean of $g(X_1), \dots, g(X_N)$ provides an estimate of m.

The Metropolis-Hastings algorithm is very general and hence very useful.

The following example shows how it can be used for inference for a situation where it would otherwise be impossible to compute desired expectations.

Brief review of statistical inference

- We assume that data Y are distributed according to density $h_\theta(y)$, where θ is the (unknown) object of our scientific interest.
- We observe Y; suppose we denote it $Y_{\rm obs}$.
- The likelihood is $L(\theta) = h_\theta(Y_{\rm obs})$.
- To find the MLE, we'd maximize $L(\theta)$. (Or in practice $\log L(\theta)$)

In Bayesian inference, we assume a prior density $p(\theta)$ for θ. The prior may be based on astrophysical insights (e.g. no source can have negative brightness), past astronomical observation (e.g. stars have masses between 0.08-150 solar masses), and/or purely statistical considerations (e.g. uniform or Jeffreys priors) when it is difficult to obtain good prior information.

The object of inference is the posterior density $\pi(\theta|Y_{\rm obs})$.

The posterior is proportional to likelihood times prior

- That is, $\pi(\theta|Y_{\mathrm{obs}}) \propto L(\theta) = h_\theta(Y_{\mathrm{obs}})p(\theta)$.
- Sometimes the proportionality constant is difficult to calculate.
- MCMC algorithms avoid computation of the proportionality constant (!)

Although the MCMC methods discussed here are often associated with Bayesian computation, this need not be the case.

Essentially, any time samples from a complicated distribution are needed, MCMC may be useful.

Example: X-ray emission time series

Our dataset is from the Chandra Orion Ultradeep Project (COUP), found in the file ‘COUP551_rates.dat’ in the following path:

```r
MCMCpath <- paste("http://www.stat.psu.edu/~dhunter/",
            "astrostatistics/mcmc/", sep="")
```

This is a time series of X-ray emission from a flaring young star in the Orion Nebula Cluster. More information on the Chandra Flares dataset is available at
http://astrostatistics.psu.edu/datasets/Chandra_flares.html.

- The raw data arrive approximately according to a Poisson process
- They consist of the individual photon arrival times (in seconds) and their energies (in keV).
- The processed data we consider here are counts of events obtained by grouping the events into evenly-spaced time bins of width 10,000 seconds.

Assumption: Piecewise homogeneous Poisson process with changepoint

Our goal is to identify the changepoint and estimate the intensities of the Poisson process before and after the change point.

Here's a Bayesian model for the problem from *Bayes and Empirical Bayes Methods for Data Analysis* (Carlin and Louis, 2000, Chpt. 5):

- Let Y_t be the number of occurrences of some event at time t.
- The process is observed for times 1 through n.
- At time k, the event counts are significantly different, higher or lower than before.

In other words,

$$\begin{aligned} Y_t | k, \theta, \lambda \sim & \text{Poisson}(\theta) \text{ for } t = 1, \ldots, k; \\ Y_t | k, \theta, \lambda \sim & \text{Poisson}(\lambda) \text{ for } t = k+1, \ldots, n. \end{aligned}$$

We assume that the Y_t are mutually independent conditional on the parameters.

Priors for piecewise homogeneous Poisson process model

We assume $Y_t|k,\theta,\lambda \sim \begin{cases} \text{Poisson}(\theta) & \text{for } t = 1,\ldots,k; \\ \text{Poisson}(\lambda) & \text{for } t = k+1,\ldots,n. \end{cases}$

We assume the following prior distributions on the parameters:

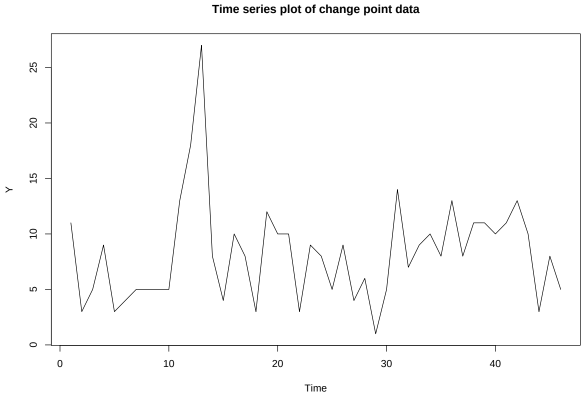
$$\begin{aligned} \theta|b_1 \sim& \text{Gamma}(0.5, b_1) \qquad (\text{pdf}{=}g_1(\theta|b_1)) \\ \lambda|b_2 \sim& \text{Gamma}(0.5, b_2) \qquad (\text{pdf}{=}g_2(\lambda|b_2)) \\ b_1 \sim& \text{IG}(1,1) \qquad (\text{pdf}{=}h_1(b_1)) \\ b_2 \sim& \text{IG}(1,1) \qquad (\text{pdf}{=}h_2(b_2)) \\ k \sim& \text{Uniform}(1,\ldots,n) \qquad (\text{pmf } {=}u(k)) \end{aligned}$$

Here, we assume k, θ, and λ are conditionally independent and the gamma(α,β) and inverse gamma(α,β) densities are

$$\frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta} \text{ and } \frac{e^{-1/\beta x}}{\Gamma(\alpha)\beta^\alpha x^{\alpha+1}}, \text{ respectively.}$$

Read data and produce a simple time series plot

```r
chptdat <- read.table(paste(MCMCpath,
      "COUP551_rates.dat", sep=""), skip=1)
Y=chptdat[,2] # store data in Y
ts.plot(Y,main="Time series plot of change point data")
```



The plot suggests a changepoint around 10.

Setting up the MCMC algorithm

Inference is based on the posterior distribution, obtained up to a multiplicative constant by multiplying likelihood times prior:

$$\begin{aligned} f(k,\theta,\lambda,b_1,b_2|\mathbf{Y}) &\propto \prod_{i=1}^{k} f_1(Y_i|\theta,\lambda,k) \prod_{i=k+1}^{n} f_2(Y_i|\theta,\lambda,k) \\ &\quad \times g_1(\theta|b_1) g_2(\lambda|b_2) h_1(b_1) h_2(b_2) u(k) \\ &= \prod_{i=1}^{k} \frac{\theta^{Y_i} e^{-\theta}}{Y_i!} \prod_{i=k+1}^{n} \frac{\lambda^{Y_i} e^{-\lambda}}{Y_i!} \\ &\quad \times \frac{1}{\Gamma(0.5) b_1^{0.5}} \theta^{-0.5} e^{-\theta/b_1} \times \frac{1}{\Gamma(0.5) b_2^{0.5}} \lambda^{-0.5} e^{-\lambda/b_2} \\ &\quad \times \frac{1}{b_1^2} e^{-1/b_1} \frac{1}{b_2^2} e^{-1/b_2} \times \frac{1}{n}. \end{aligned}$$

Our goal is to simulate multiple draws from this posterior distribution.

Full conditional distributions for θ and λ

$$\begin{aligned}
f(k,\theta,\lambda,b_1,b_2|\mathbf{Y}) \;\propto\; & \prod_{i=1}^{k}\frac{\theta^{Y_i}e^{-\theta}}{Y_i!}\prod_{i=k+1}^{n}\frac{\lambda^{Y_i}e^{-\lambda}}{Y_i!} \\
& \times \frac{1}{\Gamma(0.5)b_1^{0.5}}\theta^{-0.5}e^{-\theta/b_1} \times \frac{1}{\Gamma(0.5)b_2^{0.5}}\lambda^{-0.5}e^{-\lambda/b_2} \\
& \times \frac{1}{b_1^2}e^{-1/b_1}\frac{1}{b_2^2}e^{-1/b_2} \times \frac{1}{n}
\end{aligned}$$

gives

$$\begin{aligned}
f(\theta|k,\lambda,b_1,b_2,\mathbf{Y}) &\propto \theta^{\sum_{i=1}^{k}Y_i-0.5}e^{-\theta(k+1/b_1)} \\
&\propto \text{Gamma}\left(\sum_{i=1}^{k}Y_i+0.5,\frac{b_1}{kb_1+1}\right), \\
f(\lambda|k,\theta,b_1,b_2,\mathbf{Y}) &\propto \text{Gamma}\left(\sum_{i=k+1}^{n}Y_i+0.5,\frac{b_2}{(n-k)b_2+1}\right).
\end{aligned}$$

Full conditional distribution for k

$$
\begin{aligned}
f(k,\theta,\lambda,b_1,b_2|\mathbf{Y}) \;\propto\; & \prod_{i=1}^{k} \frac{\theta^{Y_i} e^{-\theta}}{Y_i!} \prod_{i=k+1}^{n} \frac{\lambda^{Y_i} e^{-\lambda}}{Y_i!} \\
& \times \frac{1}{\Gamma(0.5) b_1^{0.5}} \theta^{-0.5} e^{-\theta/b_1} \times \frac{1}{\Gamma(0.5) b_2^{0.5}} \lambda^{-0.5} e^{-\lambda/b_2} \\
& \times \frac{1}{b_1^2} e^{-1/b_1} \frac{1}{b_2^2} e^{-1/b_2} \times \frac{1}{n}
\end{aligned}
$$

gives

$$
\begin{aligned}
f(k|\theta,\lambda,b_1,b_2,\mathbf{Y}) & \propto \prod_{i=1}^{k} \frac{\theta^{Y_i} e^{-\theta}}{Y_i!} \prod_{i=k+1}^{n} \frac{\lambda^{Y_i} e^{-\lambda}}{Y_i!} \\
& \propto \theta^{\sum_{i=1}^{k} Y_i} \lambda^{\sum_{i=k+1}^{n} Y_i} e^{-k\theta-(n-k)\lambda}.
\end{aligned}
$$

Full conditional distributions for b_1 and b_2

$$\begin{aligned} f(k,\theta,\lambda,b_1,b_2|\mathbf{Y}) \;\propto\; & \prod_{i=1}^{k} \frac{\theta^{Y_i} e^{-\theta}}{Y_i!} \prod_{i=k+1}^{n} \frac{\lambda^{Y_i} e^{-\lambda}}{Y_i!} \\ & \times \frac{1}{\Gamma(0.5) b_1^{0.5}} \theta^{-0.5} e^{-\theta/b_1} \times \frac{1}{\Gamma(0.5) b_2^{0.5}} \lambda^{-0.5} e^{-\lambda/b_2} \\ & \times \frac{1}{b_1^2} e^{-1/b_1} \frac{1}{b_2^2} e^{-1/b_2} \times \frac{1}{n} \end{aligned}$$

gives

$$f(b_1|k,\theta,\lambda,b_2,\mathbf{Y}) \propto b_1^{-2.5} e^{-(1+\theta)/b_1} \propto IG(1.5, 1/(\theta+1)).$$

and

$$f(b_2|k,\theta,\lambda,b_1|\mathbf{Y}) \propto b_2^{-2.5} e^{-(1+\lambda)/b_2} \propto IG(1.5, 1/(\lambda+1)).$$

Setting up an MCMC algorithm

Many MCMC algorithms cycle through each univariate full conditional in sequence, creating a Markov chain whose stationary distribution is the desired posterior distribution.

We have provided code to do this in R:

```
source(paste(MCMCpath, "MCMCchpt.R", sep=""))
```

Run the MCMC algorithm, with or without fixing $k = 10$:

```
mchain <- mhsampler(dat=Y, MCMCiterations=5000)
```

```
## Markov chain algorithm ran for 5000 iterations
##  acc. rate for k:  0.0720144
```

```
mchain2<-mhsampler(dat=Y,MCMCiterations=5000,kfixed=TRUE)
```

```
## Markov chain algorithm ran for 5000 iterations
```

MCMC output analysis

- We have a series of 5-dimensional vectors.
- This approx. posterior sample can be used in multiple ways.
- E.g., since θ is the first dimension, $E(theta)$ is estimated by

```
mean(mchain[1,])
```

```
## [1] 5.580691
```

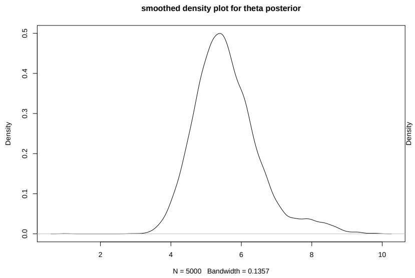
- To get estimates for means of medians for all parameters:

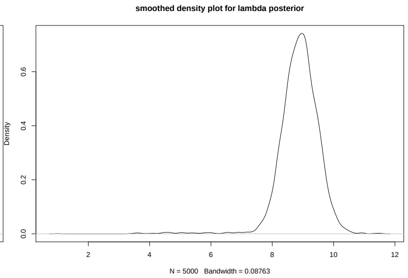
```
rbind(apply(mchain,1,mean), # compute row means
    apply(mchain,1,median)) # compute row medians
```

```
##         theta   lambda       k        b1        b2
## [1,] 5.580691 8.876345 10.4022 12.007125 18.292898
## [2,] 5.477154 8.911640 10.0000  5.402742  8.263059
```

More MCMC output analysis

Using the mchain object, we can obtain marginal posterior density estimates for θ or λ:

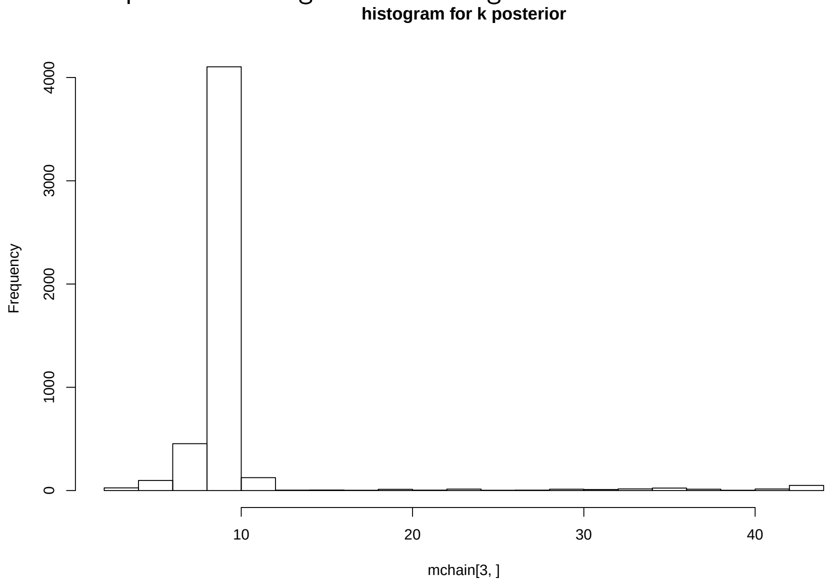




We can similarly analyze the mchain2 object, in which all k values (in the 3rd row) are fixed at 10.

More MCMC output analysis

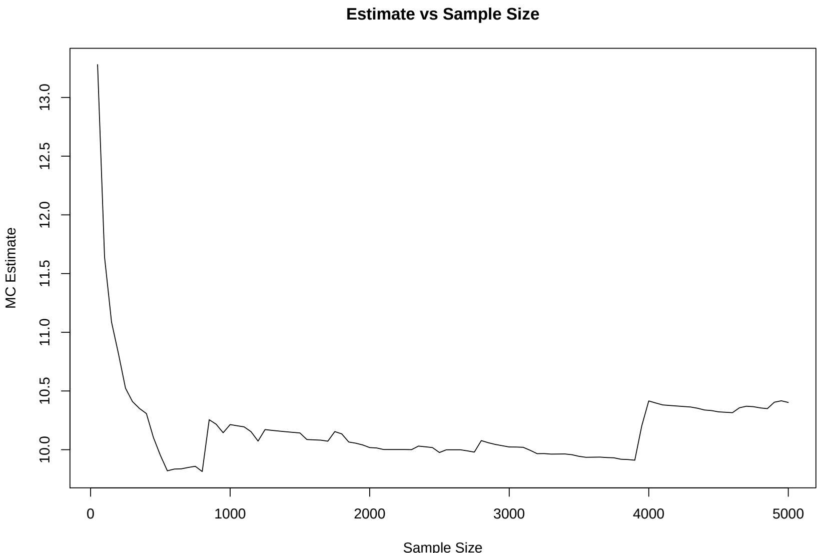
Here is a posterior histogram for k using mchain:



Mean estimates as a function of iteration

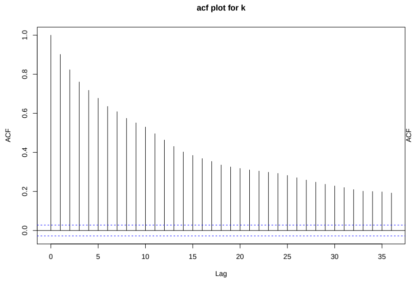
The ‘batchmeans’ package has a function that depicts this.

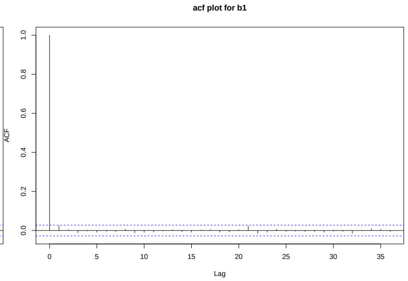
```r
estvssamp(mchain[3,])
```



Assessing behavior of the chain

- We would like to assess whether our Markov chain is moving around quickly enough to produce good estimates, a property often called ‘good mixing’.
- This is hard to do rigorously, but some tests are possible.
- An example: Estimates of autocorrelation for k and b_1





Strong autocorrelation, MCMC mixing and efficiency

- Using ‘acf’ on each parameter in turn shows that only the k parameter exhibits strong autocorrelation.
- This isn’t a big problem in this case; it is easy to run the sampler for a long time. This may not be true in all cases.
- The Metropolis Hastings acceptance rate for k is under 10%, so a different proposal might help the chain mix better.
- Carefully constructed proposals can have a major impact on the efficiency of an MCMC algorithm.

Choosing an MCMC starting value

- In general, any value we believe to be reasonable under the posterior distribution will suffice.
- Sometimes much is made about running the chain for a certain number of “burnin” iterations before any sampling is done.
- In theory, longer burnin periods will cause the chain to “forget” its starting value so that the influence of this value will be lessened. While this sounds good, in practice burnin is probably not necessary as long as the initial value is plausible.
- See http://users.stat.umn.edu/∼geyer/mcmc/burn.html for an amusing rant entitled “Burn-in is unnecessary” from one of the world’s authorities on MCMC.

Assessing accuracy and determining chain length

For classical i.i.d. Monte Carlo samplers,

- Calculating standard errors for sample means is easy using the usual formula (SD estimate divided by the square root of N).
- Chain length is the same thing as sample size.

Assessing accuracy and determining chain length

For MCMC,

- Since Markov chains produce dependent draws, computing precise Monte Carlo standard errors for such samplers is a difficult problem in general.
- The algorithm produces draws that are only asymptotically from the correct distribution.
- All the draws we see after a finite number of iterations are therefore only approximately from the correct distribution.
- Determining how long we have to run the chain before we feel sufficiently confident that the MCMC algorithm has produced reasonably accurate draws from the distribution is therefore a very difficult problem.
- Most rigorous solutions are too specific or tailored towards relatively simple situations while more general approaches tend to be heuristic.

Standard error estimates for MCMC output

The ‘batchmeans’ package allows for this:

```
apply(mchain, 1, function(a) bm(a)$est)
```

```
##     theta    lambda         k        b1        b2
##  5.580863  8.876791 10.410664 12.029801 18.331369
```

```
apply(mchain, 1, function(a) bm(a)$se)
```

```
##      theta     lambda          k         b1         b2
## 0.04476278 0.03966915 0.39629732 0.48591634 0.81886488
```

Are these standard errors acceptable?

Length of MCMC chain

Often MCMC users do not run their simulations long enough.

- There is a vast literature on different proposals for dealing with how long to run the chain.
- They are all heuristics at best.
- One method that is fairly simple, theoretically justified in some cases and seems to work reasonably well in practice is as follows: run the MCMC algorithm and periodically compute Monte Carlo standard errors. Once the Monte Carlo standard errors are below some (user-defined) threshold, stop the simulation.

Making changes to the model

- For certain types of changes, only minor changes may be needed in the code. For instance, if the Inverse Gamma prior on b_1 and b_2 were replaced by Gamma(0.01,100) prior on them, we would only have to change the lines in the code corresponding to the updates of b_1 and b_2.
- An obvious modification to this model would be to allow for more than one change point. It is possible to treat the number of changepoints as unknown. The posterior distribution is then a mixture over distributions of varying dimensions, which requires an advanced version of the Metropolis-Hastings algorithm known as **reversible-jump Metropolis Hastings** due to Peter Green (*Biometrika*, 1995).

Additional resources

- The ‘coda’ and ‘BOA’ packages in R implement many well-known output analysis techniques.
- Charlie Geyer’s ‘MCMC’ package in R is another free resource.
- JAGS and Stan are popular platforms for implementing MCMC samplers.
- There is a vast literature on various aspects of MCMC. Murali Haran has provided a list of references at http://sites.stat.psu.edu/~mharan/MCMCtut/suppl.html as a starting point.