

Bayesian Data Analysis

Sourish Das¹

¹Mathematics,
Chennai Mathematical Institute, INDIA

Bayesian Monte Carlo Simulation



Agenda

- ▶ Introduction and Motivation
- ▶ Inverse Transform Method
- ▶ Acceptance Rejection Method
- ▶ Importance Sampling Method
- ▶ Markov Chain Monte Carlo Method
 - ▶ Gibbs Sampler
 - ▶ Metropolis-Hastings
- ▶ MCMC Diagnostics



Introduction

- ▶ Suppose $Y \sim p(y|\theta)$ and $p(\theta)$ is the prior distribution over θ
- ▶ Our objective is to make statistical inference about θ
- ▶ The posterior distribution is:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int_{\Theta} p(y|\theta)p(\theta)d\theta}$$

Introduction

- ▶ The posterior distribution is:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int_{\Theta} p(y|\theta)p(\theta)d\theta}$$

- ▶ Posterior mean: **integration problem**

$$E(\theta|y) = \int_{\Theta} \theta p(\theta|y)d\theta = g(y)$$

- ▶ Posterior median: **integration problem**

$$\int_{-\infty}^{\mu_0} p(\theta|y)d\theta = \frac{1}{2},$$

- ▶ Posterior Mode: **optimization problem**

$$\hat{\theta} = \arg \max_{\theta} p(\theta|y)$$

- ▶ Posterior predictive distribution:

$$p(\tilde{y}|y) = \int_{\Theta} p(\tilde{y}|\theta, y)p(\theta|y)d\theta$$

Introduction

- ▶ In order to get the posterior mean we have to solve this integration
- ▶ Analytical solution does not exist for most of the sophisticated models
- ▶ So we have to resort to simulation technique to solve this integration problem.
- ▶ Typically it is known as **Monte Carlo Integration** method

Monte Carlo Methods

- ▶ Monte Carlo methods rely on
 - ▶ The possibility of generation of endless flow of random variables
 - ▶ For well-known or new distributions.
- ▶ Such a simulation is based on the generation of uniform random variables on the interval $(0, 1)$.
- ▶ We are not concerned with the details of producing uniform random numbers.
- ▶ We assume the existence of such a sequence

Monte Carlo Integration

- ▶ As we want to estimate the posterior mean:

$$E(\theta|y) = \int_{\Theta} \theta p(\theta|y) d\theta = g(y)$$

we can do so by simulating random samples from $p(\theta|y)$

- ▶ Suppose $(\theta^1, \dots, \theta^N)$ are random samples from $p(\theta|y)$, then we can approximate $g(y)$ by

$$\hat{g}(y) = \frac{1}{N} \sum_{s=1}^N \theta^s$$

- ▶ If we can ensure simple random sample then Laws of Large Number ensures

$$\hat{g}(y) = \frac{1}{N} \sum_{s=1}^N \theta^s \rightarrow g(y) = E(\theta|y)$$

as $N \rightarrow \infty$, where N is the simulation size.

Monte Carlo Integration

- ▶ **Convergence:**

$$\bar{h} = \frac{1}{n} \sum_{i=1}^n h(\theta^i) \xrightarrow{a.s.} E(h(\theta)|y) = \int_{\Theta} h(\theta) p(\theta|y) d\theta$$

- ▶ **CLT:**

$$\frac{\bar{h} - E(\theta|y)}{\sqrt{\sigma_n^2}} \xrightarrow{L} N(0, 1)$$

where $\sigma_n^2 = \frac{1}{n} \sum_{i=1}^n [\bar{h} - E(\theta|y)]^2$, such that $\int_{\Theta} h(\theta)^2 p(\theta|y) d\theta < \infty$

Monte Carlo Integration

- ▶ The advantage of CLT is we can evaluate the Monte Carlo error
- ▶ It assumes σ_n^2 is the proper estimate of the variance of \bar{h}_n
- ▶ If σ_n^2 does not converge, converges too slowly, a CLT may not apply. In such cases we will not be able to estimate the Monte Carlo error.

Using the R Generator

- ▶ R has a large number of functions that will generate random samples from standard distributions.
- ▶ If built-in function in R is available then we can use it directly.
- ▶ However, if the built-in functions are not available and the posterior distribution is known upto its kernel then we have to use generic methods to draw samples from such distributions.

The Inverse Transform

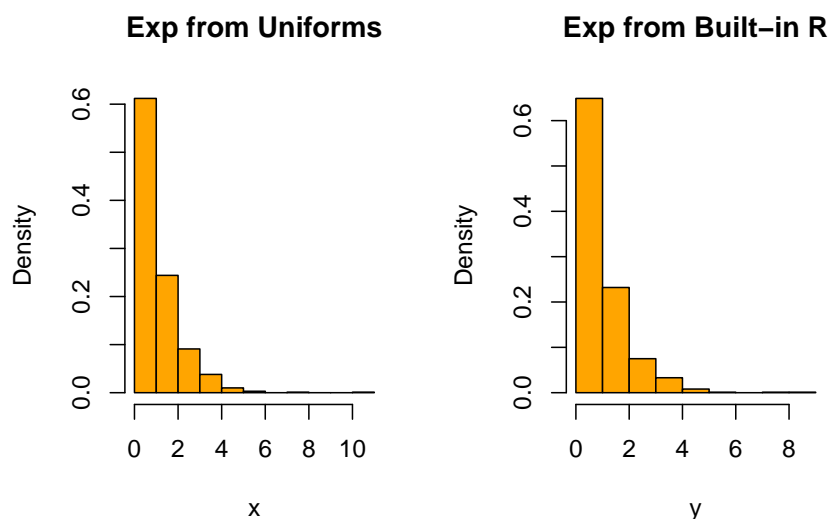
- ▶ The probability integral transform allows us to transform a uniform into any random variable.
- ▶ If X has a density f and cdf F , then we have the relation

$$F(x) = \int_{-\infty}^x f(u) du$$

we know $F(x) \sim Unif(0, 1)$ - so we set $U = F(x)$ and solve for x

- ▶ If $X \sim Exp(1)$ then $F(x) = 1 - e^{-x}$
- ▶ solving for x in $u = 1 - e^{-x}$ gives $x = -\log(1 - u)$

Generating Exponentially Distributed Random Samples



The Inverse Transform from Uniforms

- ▶ This method is useful for other probability distributions ; ones obtained as a transformation of uniform random variables

- ▶ Logistic cdf:

$$F(x) = \frac{1}{1 + \exp\left(-\frac{x-\mu}{\lambda}\right)}$$

- ▶ Cauchy cdf:

$$F(x) = \frac{1}{2} + \frac{1}{\pi} \arctan((x - \mu)/\sigma)$$

General Transform Method

- ▶ If $X_i \stackrel{iid}{\sim} \text{Exp}(1)$; three standard distribution can be derived as

- ▶ $Y = 2 \sum_{i=1}^n X_i \sim \chi_{2n}^2$

- ▶ $Y = \beta \sum_{i=1}^n X_i \sim \text{Gamma}(n, \beta)$

- ▶ $Y = \frac{\sum_{i=1}^{n_1} X_i}{\sum_{i=1}^{n_1+n_2} X_i} \sim \text{Beta}(n_1, n_2)$

where $n \in \mathcal{N} = \{1, 2, \dots\}$

- ▶ These transformations are quite simple and we will use them quite often
- ▶ However, there is a limit to their usefulness. Only when closed form CDF is available the method is available.
- ▶ The method will not work for Gaussian distribution as the CDF is not in closed form.

General Transform Method

- ▶ Box-Muller transform to generate from Normal distribution
- ▶ If U_1 and U_2 are iid from $Unif(0, 1)$

- ▶ The variable X_1 and X_2

$$X_1 = \sqrt{-2 \log(U_1)} \cos(2\pi U_2), \text{ and } X_2 = \sqrt{-2 \log(U_1)} \sin(2\pi U_2)$$

are iid $N(0, 1)$ by virtue of a change of variable argument.

- ▶ The Box-Muller algorithm is exact, not a crude CLT-based approximation
- ▶ Note that this is not the generator implemented in R. It uses the probability inverse transform with a very accurate representation of the Gaussian cdf

Accept-Reject Method

- ▶ There are many distributions where transform methods fail
- ▶ For these cases, we must turn to indirect methods
 - ▶ We generate a candidate random variable
 - ▶ Only accept it subject to passing a test
- ▶ This class of 'Accept-Reject' methods is extremely powerful. It will allow us to simulate from virtually any distribution.
- ▶ **Accept-Reject Methods:**
 - ▶ Only require the functional form of the density f of interest
 - ▶ f : target density, g : candidate density

where it is simpler to simulate random variables from g

Accept-Reject Methods

- ▶ The only constraints we impose on this candidate density g
 - ▶ f and g have compatible supports (i.e., $g(x) > 0$ when $f(x) > 0$).
- ▶ $X \sim f$ can be simulated as follows:
 - ▶ Generate $Y \sim g$ and independently generate $U \sim \text{unif}(0, 1)$
 - ▶ If $U \leq \frac{1}{M} \frac{f(Y)}{g(Y)}$ set $X = Y$
 - ▶ If inequality is not satisfied then discard Y and U and start again.
- ▶ Note $M = \sup_x \frac{f(x)}{g(x)}$
- ▶ $P(\text{Accept}) = \frac{1}{M}$ and expected waiting time M .

Accept-Reject Methods

Accept-Reject Algorithm

1. Generate $Y \sim g$, $U \sim \text{unif}(0, 1)$
2. Accept $X = Y$ if $U \leq f(Y)/Mg(Y)$
3. Return to 1 otherwise

Accept-Reject Methods: Theory

- ▶ Why does this method work?

- ▶ A straightforward probability calculation shows

$$P(Y \leq x | \text{Accept}) = P\left(Y \leq x | U \leq \frac{f(Y)}{Mg(Y)}\right) = P(X \leq x),$$

where $Y \sim g$ and $X \sim f$.

- ▶ Simulating from g , the output of this algorithm is exactly distributed from f .
- ▶ The Accept-Reject method is applicable in any dimension.
- ▶ As long as g is a density over the same space as f .
- ▶ Only need to know f/g upto a constant
- ▶ Only need an upper bound on M

Accept-Rejection Algorithm for Beta Distribution

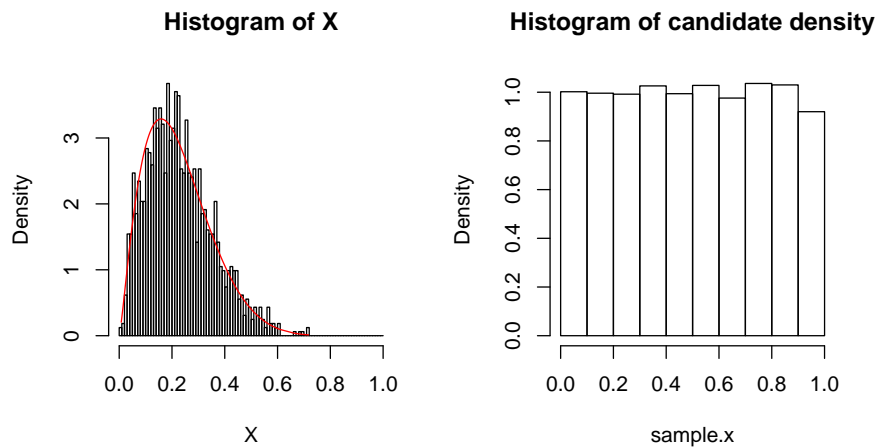
- ▶ Generate $X \sim \text{Beta}(a, b)$
- ▶ No direct method available if a and b are not integers
- ▶ Generate for $a = 2.3$ and $b = 7.9$
- ▶ We can generate if a and b is integer

Accept-Rejection Algorithm for Beta Distribution

- ▶ Candidate distribution $Unif(0, 1)$
- ▶ Target distribution $Beta(2.3, 7.9)$

Acceptance Rate :

[1] 32.4

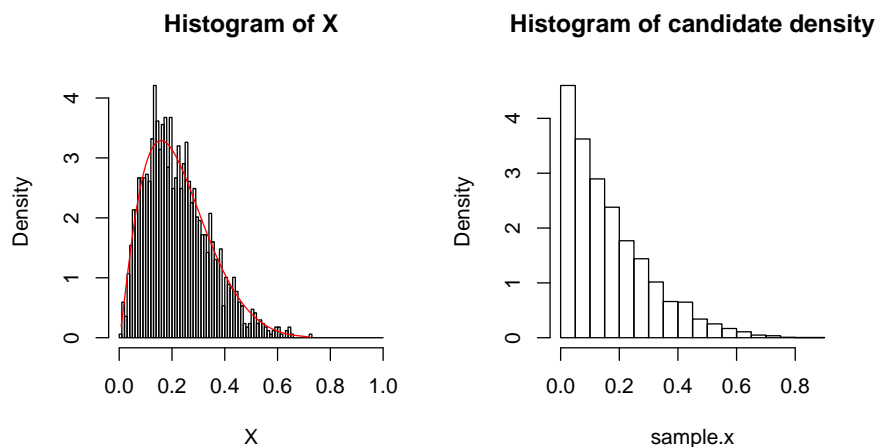


Accept-Rejection Algorithm for Beta Distribution

- ▶ Candidate distribution $Beta(1, 5)$
- ▶ Target distribution $Beta(2.3, 7.9)$

Acceptance Rate :

[1] 33.74

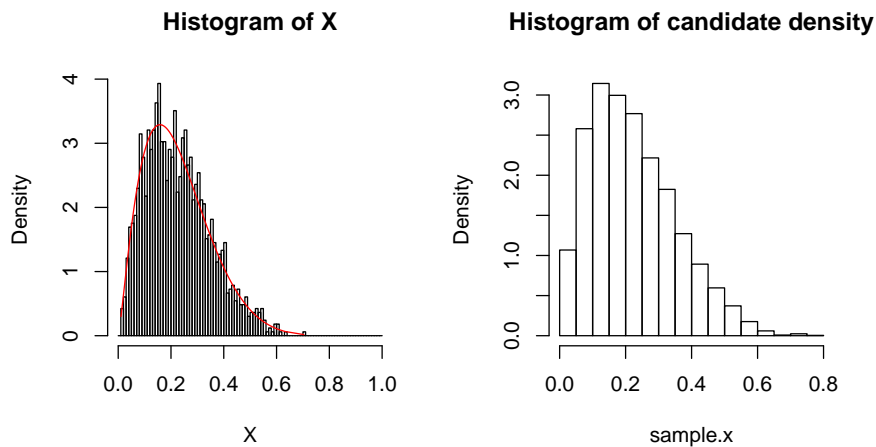


Accept-Rejection Algorithm for Beta Distribution

- ▶ Candidate distribution $Beta(2, 7)$
- ▶ Target distribution $Beta(2.3, 7.9)$

Acceptance Rate :

[1] 33.06



Importance Sampling

- ▶ Importance sampling is based on an alternative formulation of the SLLN
- ▶ For notational convenience we assume $p(\theta|y) = f(\theta)$

$$\begin{aligned} E_f(h(\theta)) &= \int_{\Theta} h(\theta) f(\theta) d\theta \\ &= \int_{\Theta} h(\theta) \frac{f(\theta)}{g(\theta)} g(\theta) d\theta \\ &= E_g \left[\frac{h(\theta) f(\theta)}{g(\theta)} \right] \end{aligned}$$

- ▶ f is the target density
- ▶ g is the candidate density

Importance Sampling

- ▶ So in 'Importance Sampling' all you do is first you generate samples from candidate distribution g
- ▶ Suppose $(\theta^1, \theta^2, \dots, \theta^N)$ are the random samples generated from g
- ▶ You estimates

$$\bar{h} = \frac{1}{N} \sum_{i=1}^N \frac{f(\theta^i)}{g(\theta^i)} h(\theta^i)$$

and by virtue of SLLN we have

$$\bar{h} = \frac{1}{N} \sum_{i=1}^N \frac{f(\theta^i)}{g(\theta^i)} h(\theta^i) \longrightarrow E_g \left[\frac{h(\theta)f(\theta)}{g(\theta)} \right] = E_f(h(\theta))$$

Importance Sampling

- ▶ The logic underlying importance sampling lies in a simple rearrangement of terms in the target integral and multiplying by 1:

$$\int h(\theta)f(\theta)d\theta = \int h(\theta)\frac{f(\theta)}{g(\theta)}g(\theta)d\theta = \int h(\theta)\omega(\theta)g(\theta)d\theta$$

here $g()$ is another density whose support is same as $f()$.

- ▶ $\omega()$ is called importance function
- ▶ a good importance function will be large when the integrand is large and small otherwise.

Importance Sampling (IS)

- ▶ **IS to improve integral approximation** - it reduces the variance of an integral approximation.
- ▶ Another objective of IS when you cannot generate from the density p
- ▶ A third objective of IS is to draw inference or generate sample when the target density is unnormalized

Importance Sampling to improve integral approximation

Improve integral approximation

Consider the function $h(x) = 10 \exp(-2|x - 5|)$. Suppose that we want to calculate $E(h(X))$, where $X \sim \text{Uniform}(0, 10)$. That is we want to calculate

$$\int_0^{10} \exp(-2|x - 5|) dx.$$

The true value for this integral is about 1.

- ▶ The simple way to do this is to generate x_i from the $\text{Uniform}(0, 10)$ and look at the sample mean of $h(x_i)$
- ▶ Notice this is equivalent to importance sampling with importance function $\omega(x) = f(x)$, where $f(x) = \frac{1}{10} I(0 < x < 10)$

Importance Sampling to improve integral approximation

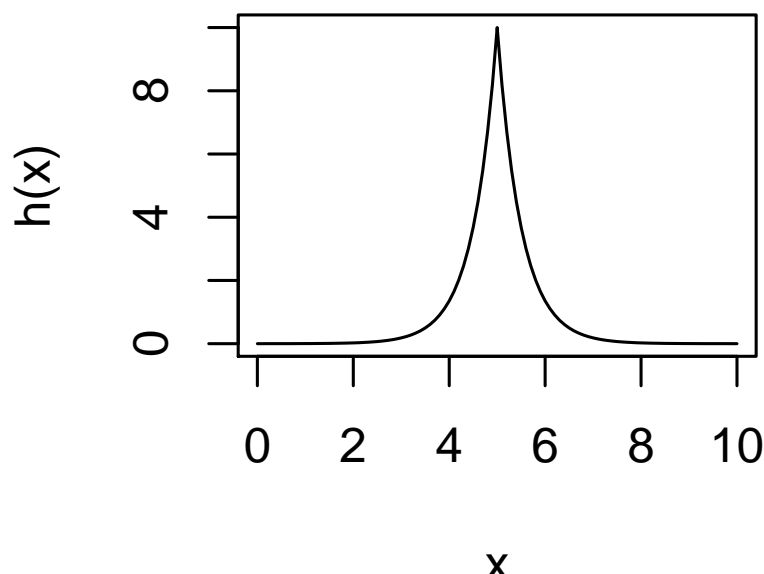
- Note that $h(x) = 10 \exp(-2|x - 5|)$ where $x \sim \text{Unif}(0, 10)$

```
> set.seed(7831)
> sim.size<-50000
> X<-runif(sim.size,0,10)
> Y<-10*exp(-2*abs(X-5))
> c(mean(Y),var(Y))

[1] 0.999026 3.993018
```


Importance Sampling to improve integral approximation

- Note that $h(x) = 10 \exp(-2|x - 5|)$ where $x \sim \text{Unif}(0, 10)$



Importance Sampling to improve integral approximation

- ▶ Note that $h(x) = 10 \exp(-2|x - 5|)$ where $x \sim \text{Unif}(0, 10)$
- ▶ The function $h()$ in this case is peaked at 5, and decays quickly elsewhere, therefore, under the uniform distribution, many of the points are contributing very little to this expectation.
- ▶ Something more like a gaussian function ce^{-x^2} with a peak at 5 and small variance, say, 1, would provide greater precision
- ▶ We can re-write the integral as

$$\int_0^{10} 10 \exp(-2|x - 5|) \frac{1/10}{\frac{1}{\sqrt{2\pi}} e^{-(x-5)^2/2}} \frac{1}{\sqrt{2\pi}} e^{-(x-5)^2/2} dx$$
CHENNAI
MATHEMATICAL
INSTITUTE

Importance Sampling to improve integral approximation

- ▶ We want to estimate $E[h(X)\omega(X)]$ where $X \sim N(5, 1)$
- ▶ So in this case $f(x) \sim \text{Uniform}(0, 10)$, $g(x) \sim N(5, 1)$
- ▶ The weight function is

$$\omega(x) = \frac{\sqrt{2\pi} e^{(x-5)^2/2}}{10}$$

Importance Sampling to improve integral approximation

- We implement the IS approach as

```
> w<-function(x)dunif(x,0,10)/dnorm(x,mean=5,sd=1)
> h<-function(x)10*exp(-2*abs(x-5))
> set.seed(7831)
> sim.size<-50000
> x<-rnorm(sim.size,mean=5,sd=1)
> y<-h(x)*w(x)
> c(mean(y),var(y))

[1] 0.9995830 0.3609514
```

Note that here Monte Carlo variance is much smaller

Importance Sampling - Application

Application:

Suppose daily return of a stock is defined as

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}} \times 100$$

where P_t is the price of the stock on t^{th} day. Suppose it is believed that return of the stock follows $r_t \sim N(0, 1)$.

Objective: We want to estimate $P(r_t < -5)$, i.e., we want to estimate the probability that the price of the stock will drop more than 5% in one day?

Importance Sampling - Application

- ▶ As we are interested in $P(r_t < -5)$ where $r_t \sim N(0, 1)$, the probability in log-scales, i.e., $\log(P(r_t < -5))$
> `log(pnorm(-5, mean=0, sd=1))`
[1] -15.065
- ▶ Simulating $Z^{(i)}$ from $N(0, 1)$ only produces a hit once in about 3 million iterations !
- ▶ This is a very rare event for standard normal distribution
- ▶ Estimates from Importance sampling
[1] -15.0732

Importance Sampling - Application

	Built-in R	Importance Sampling
$\log(P(r_t < -1))$	-1.841	-1.865
$\log(P(r_t < -2))$	-3.783	-3.803
$\log(P(r_t < -3))$	-6.608	-6.624
$\log(P(r_t < -4))$	-10.360	-10.372
$\log(P(r_t < -5))$	-15.065	-15.073

Importance Sampling for Bayesian Inference

- ▶ In Bayesian inference is the primary example where you want to determine the properties of a probability distribution given upto its kernel function.
- ▶ Suppose $k(\theta)$ is the kernel function of the posterior distribution $p(\theta|y)$, i.e.,

$$f(\theta) = p(\theta|y) = \frac{k(\theta)}{C},$$

where $C = \int k(\theta)d\theta = \int p(y|\theta)p(\theta)d\theta$

- ▶ Typically we do not know C and we know the posterior distribution in its unnormalized form, i.e.,

$$p(\theta|y) \propto p(y|\theta)p(\theta) = k(\theta)$$

Importance Sampling for Bayesian Inference

- ▶ If you want to calculate $E[h(\theta)]$ where unnormalized density of $f(\theta)$ is $k(\theta)$
- ▶ We can rewrite this as

$$\begin{aligned} E[h(\theta)] &= \int h(\theta) \frac{k(\theta)}{C} d\theta = \frac{1}{C} \int h(\theta) \frac{k(\theta)}{g(\theta)} g(\theta) d\theta \\ &= \frac{1}{C} \int h(\theta) \tilde{w}(\theta) g(\theta) d\theta \end{aligned}$$

where $\omega(\theta) = \frac{f(\theta)}{g(\theta)} = \frac{1}{C} \frac{k(\theta)}{g(\theta)} = \frac{1}{C} \tilde{w}(\theta)$ and $f(\theta) = p(\theta|y)$ is the normalized posterior density

Importance Sampling for Bayesian Inference

- ▶ By LLN if $\theta^1, \theta^2, \dots, \theta^N$ are iid random samples from g , then

$$\frac{1}{N} \sum_{i=1}^N h(\theta^i) \tilde{\omega}(\theta^i) \longrightarrow CE[h(\theta)]$$

as $N \longrightarrow \infty$.

- ▶ Also by LLN,

$$\frac{1}{N} \sum_{i=1}^N \tilde{\omega}(\theta^i) \longrightarrow C$$

- ▶ Therefore Monte Carlo estimator for $E[h(\theta)]$ using Importance Sampling Scheme is

$$\bar{h}_{IS} = \frac{\frac{1}{N} \sum_{i=1}^N h(\theta^i) \tilde{\omega}(\theta^i)}{\sum_{i=1}^N \tilde{\omega}(\theta^i)}$$

Importance Sampling for Bayesian Inference

- ▶ The method is only reliable when the weights are not too variable.
- ▶ As a rule of thumb, when

$$ESS = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(\frac{\tilde{\omega}(\theta^i)}{\bar{\omega}} - 1 \right)^2}$$

is less than 5, the IS method is reasonable. Note that $\bar{\omega} = \frac{1}{N} \sum_{i=1}^N \tilde{\omega}(\theta^i)$

- ▶ You will know you chose a bad g when ESS is large

Importance Sampling for Bayesian Inference

- ▶ When $ESS < 5$ the variance \bar{h}_{IS} can be estimated as

$$\sigma_{IS}^2 = \frac{1}{N} \sum_{i=1}^N (\zeta^i - \bar{h}_{IS})^2$$

where

$$\zeta^i = \frac{\tilde{\omega}(\theta^i)h(\theta^i)}{\bar{\omega}}$$

Application - Modeling Road Accident

- ▶ Suppose following data presents number of accidents every month for last 12 months in a particular stretch on national highway 34 (NH34)
- ▶ 6, 2, 2, 1, 2, 1, 1, 2, 3, 5, 2, 1
- ▶ We define y_i as the number of accident on i^{th} month, i.e., $y_1 = 6, y_2 = 2, \dots, y_{12} = 1$.
- ▶ We assume $y_i \stackrel{iid}{\sim} \text{Poisson}(\theta)$ and $\theta \sim \text{log-Normal}(\mu = 2, \sigma = 1)$

Application - Modeling Road Accident

- ▶ In this problem since a non-conjugate prior is being selected the posterior distribution is known upto normalizing constant.
- ▶ Simulating random samples from the unnormalized posterior distribution is difficult
- ▶ So we simulate from a candidate distribution and using the importance sampling method we estimate the θ as
[1] 2.737056

Importance Sampling

- ▶ Note that simulated $(\theta^1, \dots, \theta^N)$ is not from the target density ... it is from the candidate density g
- ▶ IS is an estimation method
- ▶ Later we will see that simulation from target can be done by adding step to IS

Markov Chain Monte Carlo

- ▶ **Markov Chain** a stochastic process in which future states are independent of past states given the present state
- ▶ **Monte Carlo** simulation technique
- ▶ Up until now, we have done Monte Carlo simulation to find integrals instead of doing it analytically. This process is called **Monte Carlo Integration**.

Monte Carlo Integration

- ▶ Suppose we have a distribution $f(\theta)$ (perhaps a posterior) from which we want to draw samples.
- ▶ To derive it analytically, we need to take integrals:

$$I = \int_{\Theta} h(\theta)f(\theta)d\theta$$

where $h(\theta)$ is some function of θ ($h(\theta) = \theta$ for the mean and $h(\theta) = (\theta - E(\theta))^2$ for the variance).

Monte Carlo Integration

- We can approximate the integrals via Monte Carlo Integration by simulating M values from $p(\theta)$ and calculating

$$\hat{I} = \frac{1}{M} \sum_{i=1}^M h(\theta^i)$$

Monte Carlo Integration

- For example, we can compute the expected value of the $Beta(3, 4)$ distribution analytically:

$$E(\theta) = \int_0^1 \theta f(\theta) d\theta = \int_0^1 \theta \theta^{3-1} (1-\theta)^{4-1} d\theta = \frac{3}{7} = 0.4286$$

or via Monte Carlo methods:

```
> set.seed(88732)
> M <- 50000
> beta.sims <- rbeta(M, 3, 4)
> sum(beta.sims)/M
[1] 0.4288105
```


Monte Carlo Integration

- ▶ Our Monte Carlo approximation \hat{I} is a simulation estimator such that $\hat{I} \rightarrow I$ as $M \rightarrow \infty$

This follows from LLN.

Law of Large Number (LLN)

- ▶ Let $\theta_1, \theta_2, \dots$ be a sequence of **independent** and identically distributed random variables, each having a finite mean $\mu = E(\theta_i)$.

Then with probability 1,

$$\frac{\theta_1 + \theta_2 + \dots + \theta_M}{M} \rightarrow \mu \text{ as } M \rightarrow \infty$$

- ▶ In the example, each simulated random sample was independent and distributed from the same $Beta(3, 4)$ distribution.
- ▶ But what if we cannot generate draws that are **independent**?

Fail to Draw Independent Random Samples

- ▶ Suppose we want to draw from our posterior distribution $p(\theta|y)$, but we cannot draw independent sample from it.
- ▶ For example, we often do not know the normalizing constant.
- ▶ We may be able to sample draws from $p(\theta|y)$ that are slightly dependent.
- ▶ If we can draw slightly dependent samples using a **Markov chain**, then can we solve the integration?

What is a Markov Chain?

- ▶ **Definition:** Markov Chain is a stochastic process in which future states are independent of past states given the present state
- ▶ **Stochastic process:** a consecutive set of random (not deterministic) quantities defined on some known state space Θ .
- ▶ Think of Θ as our parameter space
- ▶ consecutive implies a time component, indexed by t .

What is a Markov Chain?

- ▶ Consider a draw of $\theta^{(t)}$ to be a state at iteration t .
- ▶ The next draw $\theta^{(t+1)}$ is dependent only on the current draw $\theta^{(t)}$, and not on any past draws.

- ▶ **Markov Property:**

$$p(\theta^{(t+1)} | \theta^{(t)}, \theta^{(t-1)}, \dots, \theta^{(1)}) = p(\theta^{(t+1)} | \theta^{(t)})$$

What is Markov Chain?

- ▶ So our Markov chain is a bunch of draws of θ that are each dependent on the previous draw.
- ▶ The chain wanders around the parameter space, remembering only where it has been in the last time.
- ▶ What are the rules that govern how the chain jumps from one state to another at each time?
- ▶ The rule that govern the jump are known as **transition kernel**.

Transition Kernel

- ▶ For discrete state space (k possible states): a $k \times k$ matrix of transition probabilities.
- ▶ **Example:** Suppose $k = 3$. The 3×3 transition matrix \mathbf{P} would be

$p(\theta_1^{(t=1)} \theta_1^{(t)})$	$p(\theta_2^{(t=1)} \theta_1^{(t)})$	$p(\theta_3^{(t=1)} \theta_1^{(t)})$
$p(\theta_1^{(t=1)} \theta_2^{(t)})$	$p(\theta_2^{(t=1)} \theta_2^{(t)})$	$p(\theta_3^{(t=1)} \theta_2^{(t)})$
$p(\theta_1^{(t=1)} \theta_3^{(t)})$	$p(\theta_2^{(t=1)} \theta_3^{(t)})$	$p(\theta_3^{(t=1)} \theta_3^{(t)})$

where the subscripts index the 3 possible values that θ can take.

- ▶ The rows sum to one and define a conditional probability mass function, conditional on the current state.
- ▶ The columns are the marginal probabilities of being in a certain state in the next time.

Markov Chain

Discrete example:

- 1 Define a starting distribution $\Pi^{(0)}$ (a $1 \times k$ vector of probabilities that sum to one).
- ▶ At iteration 1, our distribution $\Pi^{(1)}$ (from which $\theta^{(1)}$ is drawn) is

$$\Pi_{(1 \times k)}^{(1)} = \Pi_{(1 \times k)}^{(0)} \times \mathbf{P}$$

- ▶ At iteration 2, our distribution $\Pi^{(2)}$ (from which $\theta^{(2)}$ is drawn) is

$$\Pi_{(1 \times k)}^{(2)} = \Pi_{(1 \times k)}^{(1)} \times \mathbf{P}$$

- ▶ At iteration t , our distribution $\Pi^{(t)}$ (from which $\theta^{(t)}$ is drawn) is

$$\Pi_{(1 \times k)}^{(t)} = \Pi_{(1 \times k)}^{(t-1)} \times \mathbf{P} = \Pi_{(1 \times k)}^{(0)} \times \mathbf{P}^t$$

Stationarity of Markov Chain

- ▶ Define a stationary distribution π to be some distribution Π such that $\pi = \pi \mathbf{P}$
- ▶ Here we blur the distinction between the Markov chain and its transition matrix \mathbf{P} .
- ▶ If posterior distribution is proper then the Markov Chain of the MCMC algorithm will typically converge to $p(\theta|y)$ regardless of our starting points.
- ▶ So if we can devise a Markov chain whose stationary distribution Π is our desired posterior distribution $p(\theta|y)$ then we can run this chain to get draws that approximately from $p(\theta|y)$ once the chain has converged.

Fundamental Theorem of Markov Chain

- ▶ **Fundamental Theorem of Markov Chain:** If a Markov chain \mathbf{P} is *irreducible*, *aperiodic* and *positive recurrent* then it has an unique stationary distribution π .
- ▶ This is the unique (normalized such that the entries sum to 1) left eigenvector of \mathbf{P} with eigenvalue 1.

Burn-In

- ▶ In light of the Fundamental theorem of MC, we shall refer to an irreducible, aperiodic Markov chain as ergodic.
- ▶ So as long as we can simulate an ergodic Markov chain, irrespective of the starting point the chain will converge to target stationary distribution.
- ▶ However, the time it takes for the chain to converge may vary; depending on the starting point.
- ▶ As a best practice, it is advisable to throw out a certain number of the first draws, known as the **burn-in**.

Markov Chain Monte Carlo Integration

- ▶ Once we have a Markov chain that has converged to the stationary distribution, then the draws in our chain appear to be like draws from $p(\theta|y)$, so it seems like we should be able to use Monte Carlo Integration methods to find quantities of interest.
- ▶ **One Problem:** MC draws are not independent, which we required for Monte Carlo Integration to work (required condition for SLLN to work).
- ▶ The answer is **Ergodic Theorem**

Ergodic Theorem

Ergodic Theorem

Let $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(T)}$ be T values from Markov chain that is aperiodic, irreducible and positive recurrent (that is the chain is ergodic) and $E[g(\theta)] < \infty$ Then with probability 1

$$\frac{1}{T} \sum_{t=1}^T g(\theta^{(t)}) \longrightarrow \int_{\Theta} g(\theta) \pi(\theta) d\theta$$

as $T \longrightarrow \infty$ where π is the stationary distribution.

- ▶ This is the Markov chain analog to the LLN
- ▶ But what does it mean for a chain to be aperiodic, irreducible, and positive recurrent?

Aperiodicity

- ▶ A Markov chain P is aperiodic if for all $x, y \in \Omega$ we have $\gcd\{t : P^t(x, y) > 0\} = 1$.
- ▶ If the only length of time for which the chain repeats some cycle of values is the trivial case with cycle length equal to one then the chain is aperiodic.
- ▶ Intuitively we can say repetition is allowed but as long as the chain is not repeating itself in an identical cycle, then the chain is aperiodic

Irreducibility

- ▶ A Markov chain is *irreducible* if it is possible go from any state to any other state (not necessarily in one step).
- ▶ A Markov chain \mathbf{P} is *irreducible* if for all x, y , there exists some t such that $P^t(x, y) > 0$.
- ▶ We can show if \mathbf{P} is *irreducible*, then \mathbf{P} is aperiodic \iff there exists t such that $P^t(x, y) > 0$ for all $x, y \in \Omega$.

Positive Recurrence

- ▶ A Markov chain is *recurrent* if for any given state i , if the chain starts at i , it will eventually return to i with probability 1.
- ▶ A Markov chain is *positive recurrent* if the expected return time to state i is finite; otherwise it is null recurrent.
- ▶ So if our Markov chain is aperiodic, irreducible, and positive recurrent then it is ergodic and the ergodic theorem allows us to do Monte Carlo Integration by calculating $E[g(\theta)]$ from draws, ignoring the dependence between draws.

What is MCMC?

- ▶ **MCMC** is a class of Monte Carlo methods in which we can simulate dependent sample that are approximately from a posterior probability distribution.
- ▶ We then take the draws and calculate quantities of interest for the posterior distribution.
- ▶ In Bayesian statistics, there are generally two MCMC algorithms that we use: the Gibbs Sampler and the Metropolis-Hastings algorithm.

Two MCMC Samplers and MCMC Diagnostics

1. Gibbs Sampling
2. Metropolis Hastings Algorithms
3. MCMC Diagnostics

Gibbs Sampling

- ▶ Suppose we have a joint posterior distribution $p(\theta_1, \dots, \theta_k | \text{data})$ that we want to sample from.
- ▶ We can use the Gibbs sampler to sample from the target distribution if we know the **full conditional posterior** distributions for each parameter.
- ▶ For each parameter, the **full conditional posterior** distribution is the distribution of the parameter conditional on the known information and all the other parameters: $p(\theta_j | \theta_{-j}, y)$

The Hammersley-Clifford Theorem

- ▶ For two blocks - Suppose we have a joint density $f(x, y)$. The theorem proves that we can write out the joint density in terms of the conditional densities $f(x|y)$ and $f(y|x)$

$$f(x, y) = \frac{f(y|x)}{\int \frac{f(y|x)}{f(x|y)} dy}$$

- ▶ We can write the denominator as

$$\begin{aligned} \int \frac{f(y|x)}{f(x|y)} dy &= \int \frac{\frac{f(x,y)}{f(x)}}{\frac{f(x,y)}{f(y)}} dy \\ &= \int \frac{f(y)}{f(x)} dy \\ &= \frac{1}{f(x)} \end{aligned}$$

The Hammersley-Clifford Theorem

- ▶ Thus, our right-hand side is

$$\frac{f(y|x)}{\frac{1}{f(x)}} = f(y|x)f(x) \\ = f(x, y)$$

- ▶ The theorem shows that knowledge of the conditional densities allows us to get the joint density.
- ▶ This works for more than two blocks of parameters.
- ▶ But how do we figure out the full conditionals?

Steps to Calculating Full Conditional Distributions

- ▶ Target posterior density $p(\theta|y)$
- ▶ To calculate the full conditionals for each θ :
 1. consider full posterior ignoring constant proportion
 2. split the parameter vector into 2, say $\theta = (\theta_1, \theta_2)$
 3. consider a block of parameters (say, θ_1) and ignore everything that doesn't depend on θ_1 .
 4. Use the knowledge of distributions to figure out what is the normalizing constant (and thus what is the full conditional distribution $p(\theta_1|\theta_{-1}, y)$)
 5. Repeat the steps for other blocks

You can generalize it for k blocks

Gibbs Sampler Steps

- ▶ Suppose that we are interested in sampling from the posterior $p(\theta|y)$, where θ is a vector of three parameters, $\theta_3, \theta_2, \theta_1$.
- ▶ The steps to a Gibbs Sampler are
 1. Pick a vector of starting values $\theta^{(0)}$ (Defining a starting distribution $\Pi^{(0)}$ and drawing $\theta^{(0)}$ from it.)
 2. Draw a value $\theta_1^{(1)}$ from the full conditional $p(\theta_1|\theta_2^{(0)}, \theta_3^{(0)}, y)$.
 3. Draw a value $\theta_2^{(1)}$ from the full conditional $p(\theta_2|\theta_1^{(1)}, \theta_3^{(0)}, y)$.
(Note that we must use the updated value of $\theta_1^{(1)}$.)
 4. Draw a value $\theta_3^{(1)}$ from the full conditional $p(\theta_3|\theta_1^{(1)}, \theta_2^{(1)}, y)$.

Gibbs Sampler Steps

- ▶ Steps 2-4 are analogous to multiplying $\Pi^{(0)}$ and \mathbf{P} to get $\Pi^{(1)}$ and then drawing $\theta^{(1)}$ from $\Pi^{(1)}$.
 - 5 Draw $\theta^{(2)}$ using $\theta^{(1)}$ and continually using the most updated.
 - 6 Repeat until you have M draws with each draw being a vector $\theta^{(t)}$ values.
 - 7 Optional burn-in and/or thinning.
- ▶ 'Gibbs sample' is a Markov chain with a bunch of draws of θ that are approximately from our posterior.

The Metropolis-Hastings Algorithm

- ▶ Suppose we have a posterior $p(\theta|y)$ that we want to sample from, but
 - ▶ the posterior does not look like any known distribution that we know
 - ▶ If the posterior consists of more than 2 parameters then grid approximations is not possible.
 - ▶ some (or all) of the full conditionals do not follow known distributions. Therefore Gibbs sampling is not possible for those whose full conditionals are not known.
- ▶ In such cases, we can use the **Metropolis-Hastings** algorithm.

The Metropolis-Hastings Algorithm

The Metropolis-Hastings Algorithm :

1. Choose starting value $\theta^{(0)}$
2. At iteration t , draw a candidate θ^* from a proposal distribution $q_t(\theta^*|\theta^{(t-1)})$.
3. Compute the acceptance ratio

$$r = \frac{p(\theta^*|y)/q_t(\theta^*|\theta^{(t-1)})}{p(\theta^{(t-1)}|y)/q_t(\theta^{(t-1)}|\theta^*)}$$

The Metropolis-Hastings Algorithm

- 4 Accept $\theta^{(t)} = \theta^*$ if $U < \min(1, r)$ where $U \sim \text{uniform}(0, 1)$
Otherwise $\theta^{(t)} = \theta^{(t-1)}$
- 5 Repeat steps 2-4 M times to get M draws from $p(\theta|y)$ with optional burn-in and/or thinning

Step 1: Choose a starting value $\theta^{(0)}$

- This is similar to draw sample from initial stationary distribution.
- Note that $\theta^{(0)}$ must have positive probability.

$$p(\theta^{(0)}|y) > 0$$

Step 2: Draw $\theta^{(*)}$ from $q_t(\theta^*|\theta^{(t-1)})$

- ▶ The proposal distribution $q_t(\theta^*|\theta^{(t-1)})$ determines where it moves in the next iteration of the Markov chain.
- ▶ The proposal distribution is analogous to transition kernel.
- ▶ The support of proposal density must contain the support of the posterior
- ▶ If a symmetric proposal distribution that is dependent on $\theta^{(t-1)}$, then it is known as **random walk Metropolis sampling**.

Step 2: Draw $\theta^{(*)}$ from $q_t(\theta^*|\theta^{(t-1)})$

- ▶ If the proposal distribution does not depend on $\theta^{(t-1)}$

$$q_t(\theta^*|\theta^{(t-1)}) = q_t(\theta^*)$$

then it is known as **independent Metropolis-Hasting sampling**

- ▶ In this case all candidate draws θ^* are drawn from the same distribution, regardless of where the previous draw was.
- ▶ This can be very efficient or very inefficient, depending how close the proposal density with respect to posterior density.
- ▶ Typically, if proposal density has heavier tails than the posterior, then chain will behave nicely

Step 3: Compute acceptance ratio r .

$$r = \frac{p(\theta^*|y)/q_t(\theta^*|\theta^{(t-1)})}{p(\theta^{(t-1)}|y)/q_t(\theta^{(t-1)}|\theta^*)}$$

- ▶ In the case where our proposal density is symmetric

$$r = \frac{p(\theta^*|y)}{p(\theta^{(t-1)}|y)}$$

- ▶ If the candidate sample has higher probability than the current sample, then the candidate is better so we definitely accept it.
- ▶ Otherwise, the candidate is accepted according to the ratio of the probabilities of the candidate and current samples.
- ▶ Since r is a ratio, we only need to know the $p(\theta|y)$ upto a constant.

Step 4: Decide to whether accept θ^*

- ▶ Accept θ^* as $\theta^{(t)}$ with probability $\min(r, 1)$
- ▶ For each θ^* , draw a value u from the $Uniform(0, 1)$ distribution.
- ▶ If $u \leq r$ accept θ^* as $\theta^{(t)}$. Otherwise $\theta^{(t-1)} = \theta^{(t)}$
- ▶ Candidate sample with higher density than the current samples are always accepted
- ▶ Unlike in rejection sampling, each iteration always produces a sample, either θ^* or $\theta^{(t-1)}$.

Acceptance Rate

- ▶ It is important to monitor the *acceptance rate* (the fraction of candidate samples that are accepted) of your Metropolis-Hastings algorithm.
- ▶ If acceptance rate is too high, the chain is probably not mixing well. That is the chain is not moving around the parameter space quickly enough.
- ▶ If the acceptance rate is too low, your algorithm is too inefficient, that is rejecting too many candidate samples.
- ▶ What is too high and too low depends on the specific algorithm, but generally
 - ▶ **random walk**: somewhere between 0.25 and 0.50 is recommended
 - ▶ **independent**: something close to 1 is preferred

MCMC Convergence

- ▶ After the model has converged, samples from the conditional distributions are used to summarize the posterior distribution of parameters of interest θ .
- ▶ **Convergence** refers to the idea that eventually the Gibbs Sampler or other MCMC technique that we choose will eventually reach a stationary distribution.
- ▶ From this point onwards it stays in this distribution and moved about or "mixes" throughout the subspace forever.

MCMC Convergence

- ▶ The general questions for us:
 1. At what point do we know that have we converged to the stationary distribution? (i.e. how long should our “burn-in” period be?
 2. After we have reached the stationary distribution, how many iterations will it take to summarize the posterior distribution?
- ▶ The answers to both of these questions remain a bit *ad hoc* because the desirable results that we depend on are only true asymptotically, and we donot want to wait for an infinite number of draws.

Possible problems with MCMC

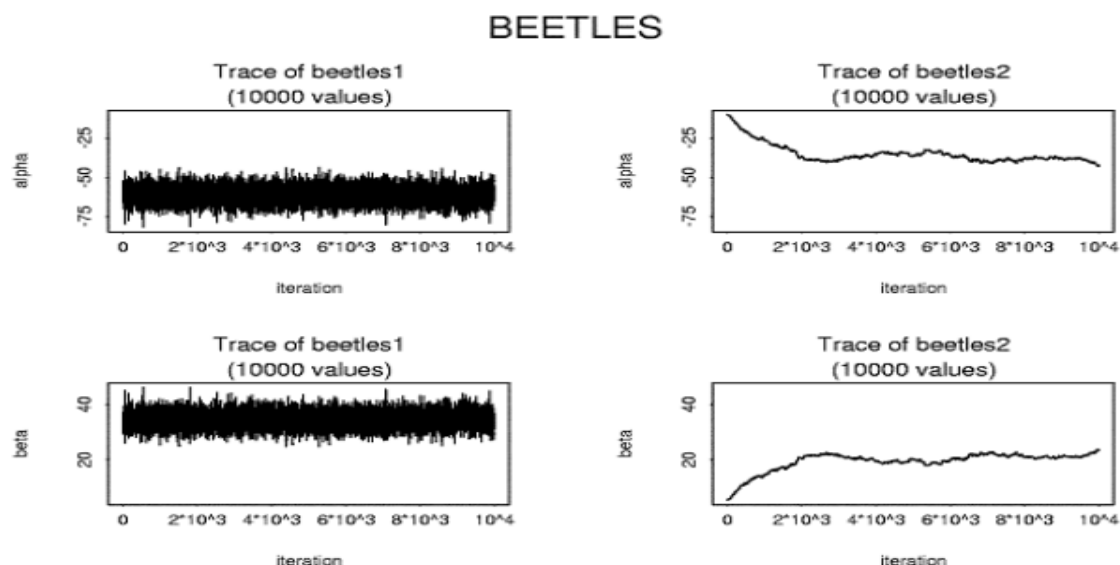
- ▶ The assumed model may not be realistic from a substantive point of view or may not fit.
- ▶ Errors in calculation or programming!
 - Often, simple syntax mistakes may be responsible; however, it is possible that the algorithm may not converge to a proper distribution.
- ▶ **Slow convergence:** this is the problem we are most likely to run into. The simulation can remain for many iterations in a region heavily influenced by the starting distribution. If the iterations are used to summarize the target distribution, they can yield falsely precise estimates.

this will be the focus of our discussion.

Traceplots

- ▶ One intuitive and easily implemented diagnostic tool is a **traceplot** (or history plot) which plots the parameter value at time t against the iteration number.
- ▶ If the model has converged, the traceplot will move snake around the mode of the distribution.
- ▶ A clear sign of non-convergence with a traceplot occurs when we observe some trending in the sample space.
- ▶ The problem with traceplots is that it may appear that we have converged, however, the chain trapped (for a finite time) in a local region rather exploring the full posterior.

Examples of Apparent Convergence and Non-Convergence Based on a trace plot



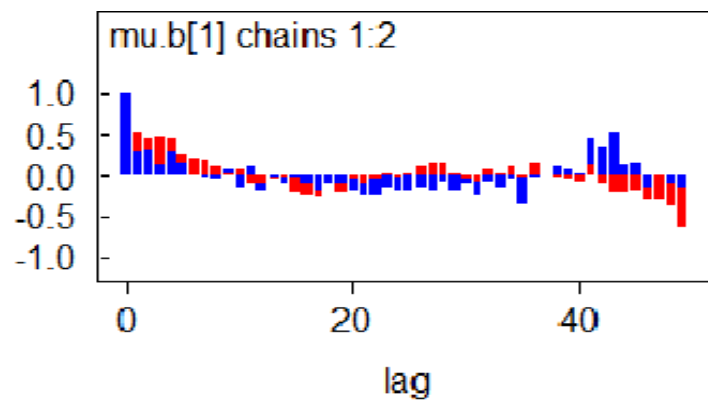
Autocorrelation Diagnostic

- ▶ **Autocorrelation** refers to a pattern of serial correlation in the chain, where sequential draws of a parameter, say θ_1 , from the conditional distribution are correlated.
- ▶ The cause of autocorrelation is that the parameters in our model may be highly correlated, so the Gibbs Sampler will be slow to explore the entire posterior distribution.
- ▶ The reason why autocorrelation is important is that it will take a very long time to explore the entire posterior distribution.

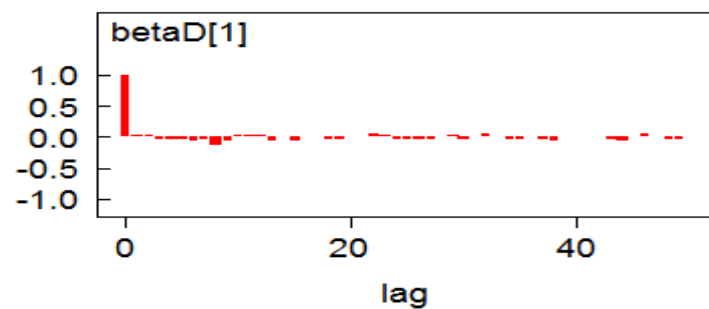
Autocorrelation Diagnostic

- ▶ Note that if the level of autocorrelation is high for a parameter of interest, then a traceplot will be a poor diagnostic for convergence.
- ▶ Typically, the level of autocorrelation will decline with an increasing number of lags in the chain (e.g. as we go from the 1000th to the 1010th lags, the level of autocorrelation will often decline.) When this dampening doesn't occur, then you have a problem and will probably want to re-parameterize the model (more on this below).

Example of Model with autocorrelation



Example of Model without autocorrelation



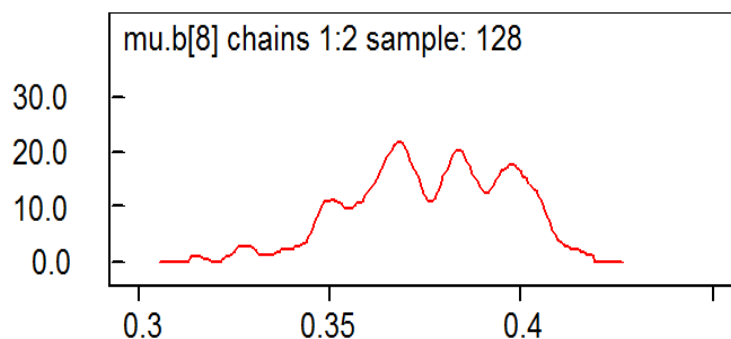
Running Means of parameters

- ▶ **Running means:** Once you have taken enough draws to summarize the posterior distribution, then if the model has converged, further samples from a parameter's posterior distribution should not influence the calculation of the mean.
- ▶ A plot of the average draw from the conditional distribution of draws 1 through t against t is useful for identifying convergence.
- ▶ Note: that you could probably get the same effect with a traceplot.

Kernel Density Plot

- ▶ Kernel density plots (a.k.a. smoothed density; histograms) may be useful diagnostic.
- ▶ Sometimes non-convergence is reflected in multimodal distributions. This is especially true if the kernel density plot isn't just multi-modal, but lumpy (you will know what I mean when you see it).
- ▶ When you get a lumpy posterior, it may be important to let the algorithm run a bit longer. Often, doing this will allow you to get a much more reasonable summary of the posterior.

Example of a problematic kernel density plot



A more satisfactory kernel density plot would look more bell-shaped, though it need not be symmetric

Geweke Time-Series Diagnostic

- ▶ **The Geweke Time-Series Diagnostic:** if a model has converged, then if simulate a large number of draws, the mean (and variance) of a parameter's posterior distribution from the first half of the chain will be equal to the mean (and variance) from the second half of the chain.
- ▶ Technically, this statistic is based on spectral density functions that are beyond the purview of this course and WinBugs does not estimate this statistic directly, but if you export the CODA chain to R the programs CODA and BOA report the Geweke statistic.
- ▶ However, a perfectly reasonable way to proceed is look to see whether the posterior means (and variances) of your parameters are approximately the same for different halves of your simulated chain.

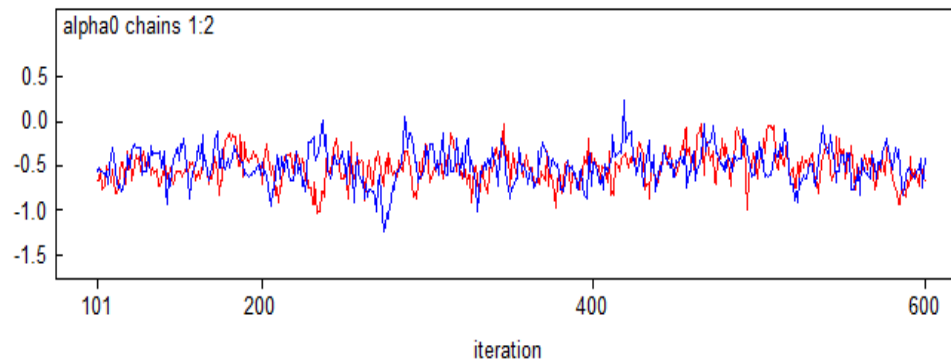
Geweke Time-Series Diagnostic

- ▶ **The Geweke Time-Series Diagnostic:** if a model has converged, then if simulate a large number of draws, the mean (and variance) of a parameter's posterior distribution from the first half of the chain will be equal to the mean (and variance) from the second half of the chain.
- ▶ The value of this approach is that by allowing the algorithm to run for a very long time, it may reach areas of the posterior distribution that may not otherwise be reached.

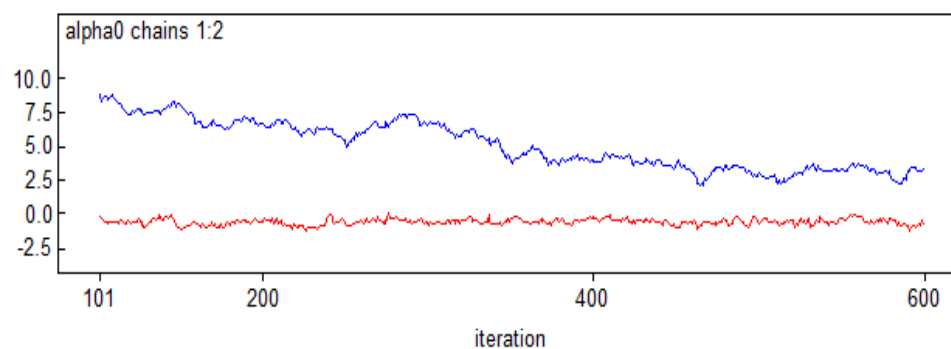
Gelman-Rubin Diagnostic

- ▶ Gelman (especially) argues that the best way to identify non-convergence is to simulate multiple sequences for over-dispersed starting points.
- ▶ The intuition is that the behavior of all of the chains should be basically the same.
- ▶ Or, as Gelman and Rubin put it, the variance within the chains should be the same as the variance across the chains.
- ▶ I think this can be diagnosed pretty easily through traceplots of multiple chains. You want to see if it looks like the mean and the variance of the two chains are the same.

Examples where convergence seems reasonable



Examples where convergence seems unreasonable



Gelman-Rubin Diagnostic

- ▶ The Gelman-Rubin statistic is based on the following procedure:
 1. estimate your model with a variety of different initial values and iterate for an n-iteration burn-in and an n-iteration monitored period.
 2. take the n-monitored draws of m parameters and calculate the following statistics:
 - 2.1 Within chain variance $W = \frac{1}{m(m-1)} \sum_{j=1}^m \sum_{i=1}^n (\theta_j^i - \bar{\theta}_j)^2$
 - 2.2 Between chain variance $B = \frac{m}{m-1} \sum_{j=1}^m (\bar{\theta}_j - \bar{\theta})^2$
 - 2.3 Estimated variance $\hat{V}(\theta) = (1 - \frac{1}{n}) W + \frac{1}{n} B$
 - 2.4 Gelman-Rubin Statistics: $\sqrt{R} = \sqrt{\frac{\hat{V}(\theta)}{W}}$

Gelman Rubin Diagnostic

- ▶ Before convergence, W underestimates total posterior variance in θ because it has not fully explored the target distribution.
- ▶ $V(\theta)$ on the other hand overestimates variance in θ because the starting points are over-dispersed relative to the target.
- ▶ Once convergence is reached, W and $V(\theta)$ should be almost equivalent because variation within the chains and variations between the chains should coincide, so R should approximately equal one.
- ▶ The drawback of this stat is that its value depends a great deal on the choice of initial values.

Convergence Diagnostic Summary

- ▶ You can never prove that something has converged, you can only tell when something has not converged.
- ▶ If your model has not converged and you are confident that YOU have not made any mistake, then the best thing is to just let the model run a long time. CPU time is often "cheaper" than your time.
- ▶ For models with large numbers of parameters you should let the model run for a long time.
- ▶ There are a number of easy way to implement tricks (mostly reparamerizations) that will help to speed convergence in most regression-based models.
- ▶ **Convergence does not mean that you have a good model!!! Convergence should be the beginning of model assessment, not its end.**

Thank You

sourish@cmi.ac.in

www.cmi.ac.in/~sourish