# Computational Methods for Bayesian Inference

BDA3 Ch 11, A First Course in Bayesian Statistical Methods (Hoff) Ch 6 and 7

## The Metropolis and Metropolis-Hastings algorithms

The Metropolis-Hastings algorithm is a general term for a family of Markov chain simulation methods useful for sampling from Bayesian posterior distributions.

The Gibbs sampler is a special case of the Metropolis-Hastings algorithm.

## *Metropolis-Hastings Algorithm*

The steps of the Metropolis-Hastings algorithm:

1. Choose an initial value, $\theta^{(0)}$, for which $p\left(\theta^{(0)}\middle|y\right) > 0$.
2. For $t = 1,2, \ldots$:
   a. Sample a proposal $\theta^*$ from a **proposal distribution** (or jumping distribution) at time $t$, $j_t(\theta^*|\theta^{(t-1)})$.
   b. Calculate the acceptance ratio

$$r = \frac{p(\theta^*|y)/j_t(\theta^*|\theta^{(t-1)})}{p\left(\theta^{(t-1)}\middle|y\right)/j_t(\theta^{(t-1)}|\theta^*)}.$$

   c. Set $\theta^{(t)} = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta^{(t-1)} & \text{otherwise.} \end{cases}$

The *Metropolis algorithm* assumes a symmetric proposal distribution.

- A symmetric proposal distribution means that $j_t\left(\theta^* \middle| \theta^{(t-1)}\right) = j_t(\theta^{(t-1)}|\theta^*)$.
- The acceptance ratio simplifies to $r = p(\theta^*|y)/p(\theta^{(t-1)}|y)$.

Some intuition (based on the Metropolis algorithm):

- If we propose a value with greater likelihood, then we move to that value with probability one.
- If we propose a value with smaller likelihood, we will still move to the region with probability $r$.

The Markov chain must sometimes move to less likely regions or it wouldn't explore the entire target distribution.

*Computational Note*

To avoid numerical overflow issues, we should typically implement the acceptance rule on the (natural) log scale.

We calculate $\ln(r)$ and accept $\theta^*$ when $\ln(u) \leq \min(\ln(r), 0)$, where $u$ is a sample from $U(0, 1)$.

*Tips for choosing a proposal distribution*

- It should be easy to sample from $j_t(\theta^*|\theta)$ for all $\theta$.
- It should be easy to compute the acceptance ratio $r$.
- Each proposal value should go a reasonable distance in the parameter space.
- The proposed values should be accepted between 20-50% of the time.

*Specific types of the Metropolis-Hasting algorithm:*

**Random walk**:

- The proposal is symmetric and depends only on the distance between parameters, i.e., $j_t\left(\theta^* \middle| \theta^{(t-1)}\right) = j_t\left(\left|\theta^* - \theta^{(t-1)}\right|\right)$.
- E.g., $j_t\left(\theta^* \middle| \theta^{(t-1)}\right) = N(\theta^* | \theta^{(t-1)}, \sigma^2)$.
- This is a special case of the Metropolis algorithm.

**Independence chain**:

- The proposal distribution doesn't depend on the past.
  - i.e., $j_t\left(\theta^* \middle| \theta^{(t-1)}\right) = j_t(\theta^* | \theta)$ for some fixed value of $\theta$.
- The proposal distribution should resemble the target distribution and cover the tails of the target distribution.
- Simple trick: use the prior as the proposal distribution. In that case,

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

**Componentwise Metropolis-Hastings:**

- Divide the parameter vector $\theta$ into subvectors and update sequentially using Metropolis-Hastings updates to draw samples from each subcomponent.
- This gets a lot more complicated.

## *Example:* **Random walk Metropolis**

Use a random walk Metropolis algorithm to sample from the posterior distribution in the following setting:

*Data distribution*: $y|\theta \sim N(\theta, I)$, where $\theta$ is a vector of length 2 and $I$ is the $2 \times 2$ Identity matrix and the observed $y = (0, 0)$.

*Prior distribution*: $p(\theta) \propto 1$ (i.e., an improper prior)

*Proposal distribution*: $\theta^*|\theta^{(t-1)} \sim N(\theta^{(t-1)}, cI)$, where $c$ is a constant chosen to scale the size of the proposal distribution.

## Assessing MCMC convergence

MCMC convergence checks should be performed to make sure that our MCMC chains have converged and adequately explored the target distribution.

*Discarding warmup values*

It is common to discard the beginning of an MCMC chain to weaken the influence of the starting value.

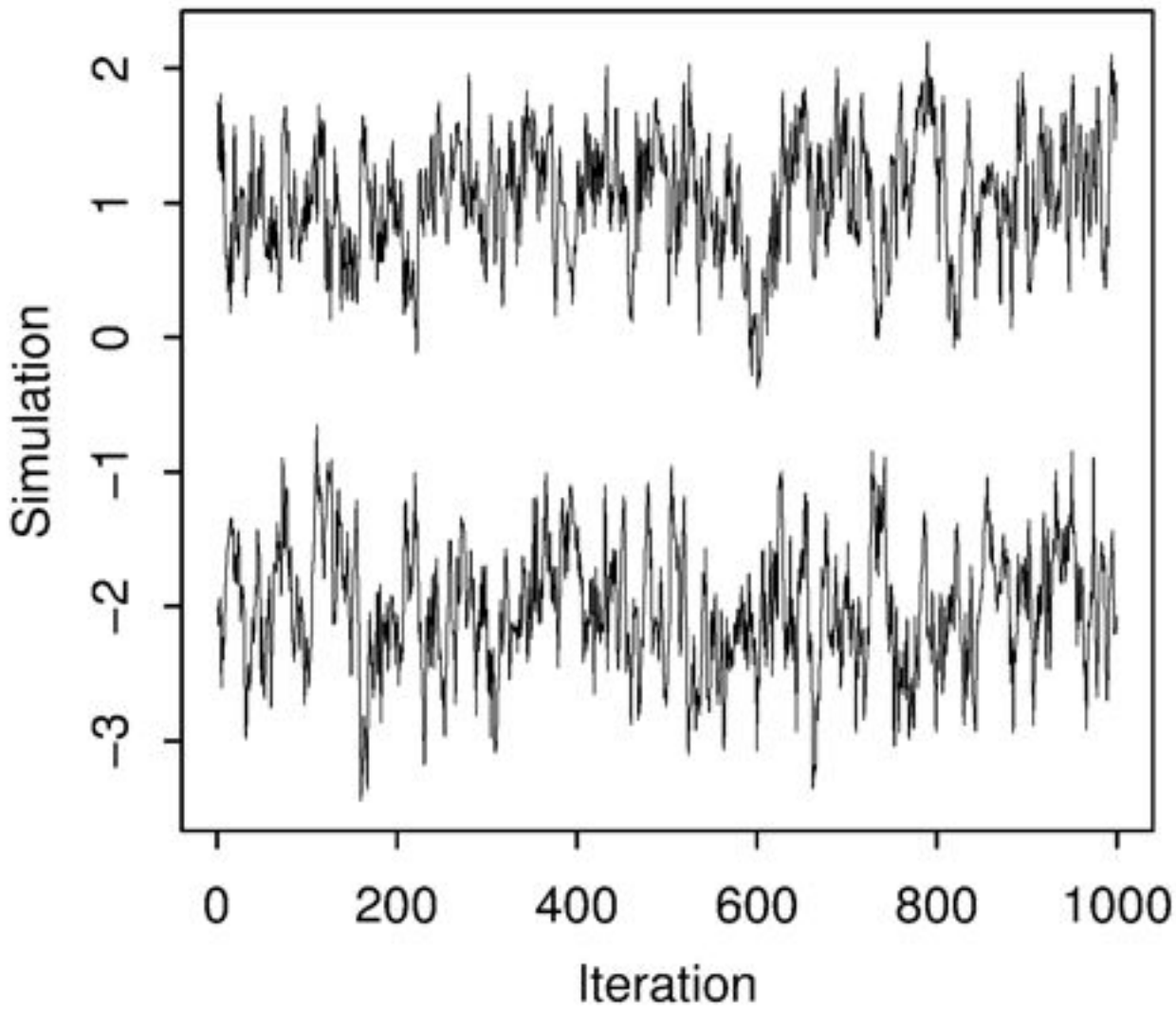
- The discarded period is known as the "burnin" or "warmup".
- A conservative warmup rules is to discard the first half of the MCMC samples.

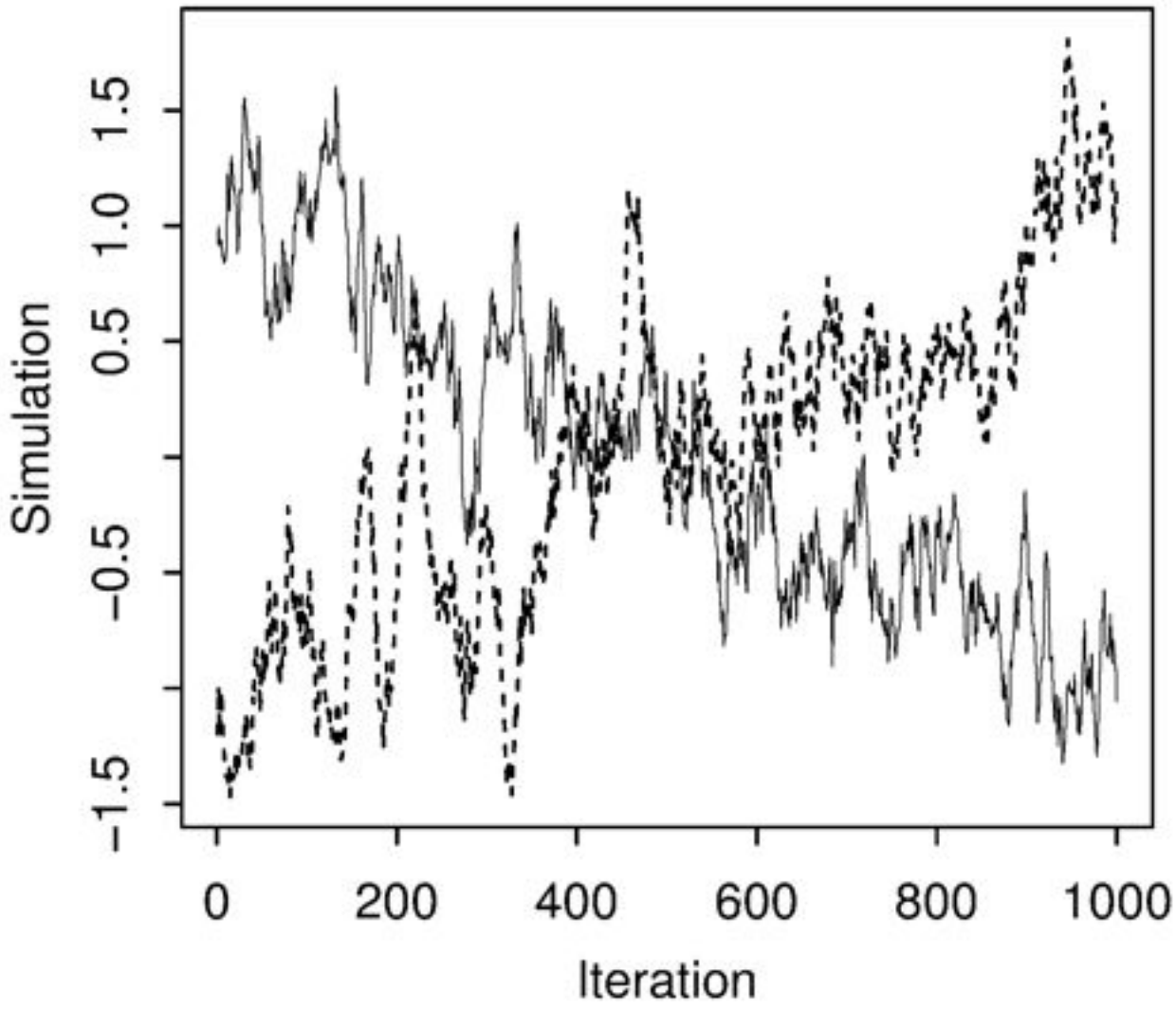
*Methods for assessing convergence to the target distribution*

**Trace plots** are plots of the sampled parameter values versus iteration number.

This is done separately for each parameter, though multiple chains of the same parameter are plotted on the same plot.

We want to see the chains converging to the same distribution and moving around a lot. The lines should be "wiggily".


Simulation
2
1
0
−1
−2
−3
0
200
400
600
800
1000
Iteration


Simulation
−1.5
−0.5
0.5
1.0
1.5
0
200
400
600
800
1000
Iteration

The **coda** R package provides many popular diagnostics for assessing convergence of MCMC output.

- Effective Sample Size: `effectiveSize()`
- Autocorrelation: `autocorr.plot()`
- Cross Variable Correlations: `crosscorr.plot()`
- Geweke: `geweke.diag()`
- Gelman-Rubin:`gelman.diag()`
- Heidelberg & Welch: `heidel.diag()`
- Raftery-Lewis: `raftery.diag()`

**Autocorrelation plots** can be used to determine how much correlation exists between the values.

- The autocorrelation should drop relatively quickly.
- It is a very bad sign if the correlation in the autocorrelation plots decays very slowly.

The **effective sample size** is the (estimated) number of independent observations our sample is equivalent to.

- $B$ samples obtained from an MCMC algorithm are NOT equivalent to $B$ independent samples because there is correlation between our sampled values.
- Larger effective samples sizes are better than smaller effective sample sizes.
- The greater the correlation between observations, the smaller the effective sample size will be.

## *Gelman-Rubin diagnostic* ($\hat{R}$)

Gelman and Rubin (1992) proposed a general approach to monitoring convergence of MCMC output in which multiple chains are run.

- Use different starting values that are overdispersed relative to the posterior distribution.
- Convergence is diagnosed when the chains have "forgotten" their initial values, and the output from all chains is indistinguishable.
- The diagnostic is applied to a single variable from the chain. It is based a comparison of within-chain and between-chain variances (like a classical analysis of variance).
- Assumes the target is normal.
- Values of $\hat{R}$ near 1 (<1.1) suggest convergence.

## *Heidelberg-Welch diagnostic*

A convergence test that uses the Cramer-von-Mises statistic to test the null hypothesis that the sampled values come from a stationary distribution.

- The test is successively applied, firstly to the whole chain, then after discarding the first 10%, 20%, of the chain until either the null hypothesis is accepted, or 50% of the chain has been discarded.
- The latter outcome constitutes “failure” of the stationarity test and indicates that a longer MCMC run is needed.
- If the stationarity test is passed, the number of iterations to keep and the number to discard (burn-in) are reported.

If the stationarity test is passed, the diagnostic then performs a half-width test.

- The ratio of the margin of error (the half-width) for a confidence interval is compared to the estimated mean.
- If this ratio is less than $\epsilon$, some desired level of precision, then the chain is long enough.
- If this ratio is more than $\epsilon$, then the chain should be extended.
- The default value for $\epsilon$ in the **coda** package is 0.1.

## *Raftery Diagnostic*

This diagnostic produces an estimate of the number of iterations needed to estimate a quantile $q$ to within an accuracy of $+/-\ r$ with probability $p$.

This diagnostic estimates $I$, the dependence factor, of the extent to which autocorrelation inflates the required sample size.

- Values of $I$ larger than 5 indicate strong autocorrelation.

## *Geweke Diagnostic*

Geweke (1992) proposed a convergence diagnostic for Markov chains based on a test for equality of the means of the first and last part of a Markov chain (by default the first 10% and the last 50%).

- If the samples are drawn from the stationary distribution of the chain, the two means are equal and Geweke's statistic has an asymptotically standard normal distribution.
- The test statistic is a standard Z-score: the difference between the two sample means divided by its estimated standard error, so it should be between -2 and 2 if the chain has converged to its stationary distribution.

## *Example: M-H independence chain for Binomial proportion*

Implement the Metropolis-Hastings algorithm with an independence sampler to estimate the posterior for $\theta$, the proportion of female placenta previa births, for the placenta previa data introduced in Chapter 2.

*Data distribution*: $y|\theta \sim \text{Binomial}(980, \theta)$ with observed $y = 437$.

*Prior distribution*: $\theta \sim U(0,1)$

*Proposal distribution:* $\theta^* \sim Beta(\alpha, \beta)$

## *Example: Componentwise M-H for $N(\mu, \sigma^2)$*

Implement a componentwise Metropolis-Hastings sampler to draw samples from the posterior for the parameters $\mu$ and $\sigma^2$ for the midge data introduced in Chapter 3.

*Data distribution:* $y_1, \ldots, y_n | \mu, \sigma^2 \overset{\text{i.i.d.}}{\sim} N(\mu, \sigma^2)$
*Prior distributions*:
$\mu|\sigma^2 \sim N(\mu_0, \sigma^2/\kappa_0)$ with $\mu_0 = 1.9$ and $\kappa_0 = 1$.
$\sigma^2 \sim$ Inv- $\chi^2_{\nu_0}(\sigma_0^2)$ with $\nu_0 = 1$ and $\sigma_0^2 = 0.1^2$.
*Proposal distributions:*
$\mu^*|\mu^{(t-1)} \sim N(\mu^{(t-1)}, \sigma_\mu^2)$ with $\sigma_\mu^2 = 0.2^2$.
$\sigma^{2^*} \sim$ Inv-$\chi^2_{\nu_*}(s_*^2)$ with $\nu_* = 2$ and $s_*^2 = 0.1^2$.

## *Example: M-H independence chain for Mixture distribution*

Implement an independence chain Metropolis-Hastings sampler to draw samples from the posterior distribution of $\theta$ using a sample coming from a mixture distribution.

*Data distribution:* $y_1, \dots, y_n|\theta \overset{\text{i.i.d.}}{\sim}$ with $p(y_i|\theta) = \theta N(7, 0.5^2) + (1 - \theta)N(10, 0.5^2)$.

*Prior distribution*: $\theta \sim \text{Beta}(1,1)$

*Proposal distribution:* $\theta^* \sim \text{Beta}(\alpha, \beta)$