MCMC Diagnostics

Yingbo Li

Clemson University

MATH 9810

Convergence to Posterior Distribution

Theory proves that if a Gibbs sampler iterates enough, the draws will be from the joint posterior distribution (called the target or stationary distribution). Convergence to the target distribution does not depend on the starting point.

- How long do we need to run the Markov Chain to adequately explore the posterior distribution?
- How can we tell if the chain is not converging?

Initial Steps

- Don't want pre-convergence values to influence the summary of the posterior distribution (the draws of the parameters from the MCMC) too much.
- Ideal is to throw away all draws before convergence to target distribution, and use only draws after convergence.
- Hard to know exactly when convergence has happened. Thus, it is standard to throw out the first $p\%$ of draws (default is 50%) as *burn-in* after you are satisfied that convergence has been achieved.
- Inference done with remaining $(100 - p\%)$ of draws.

Three Component Mixture Model

Posterior for μ:

$$\mu \mid Y \sim 0.45 \text{ N}(-3, 1/3) + 0.1 \text{ N}(0, 1/3) + 0.45 \text{ N}(3, 1/3)$$

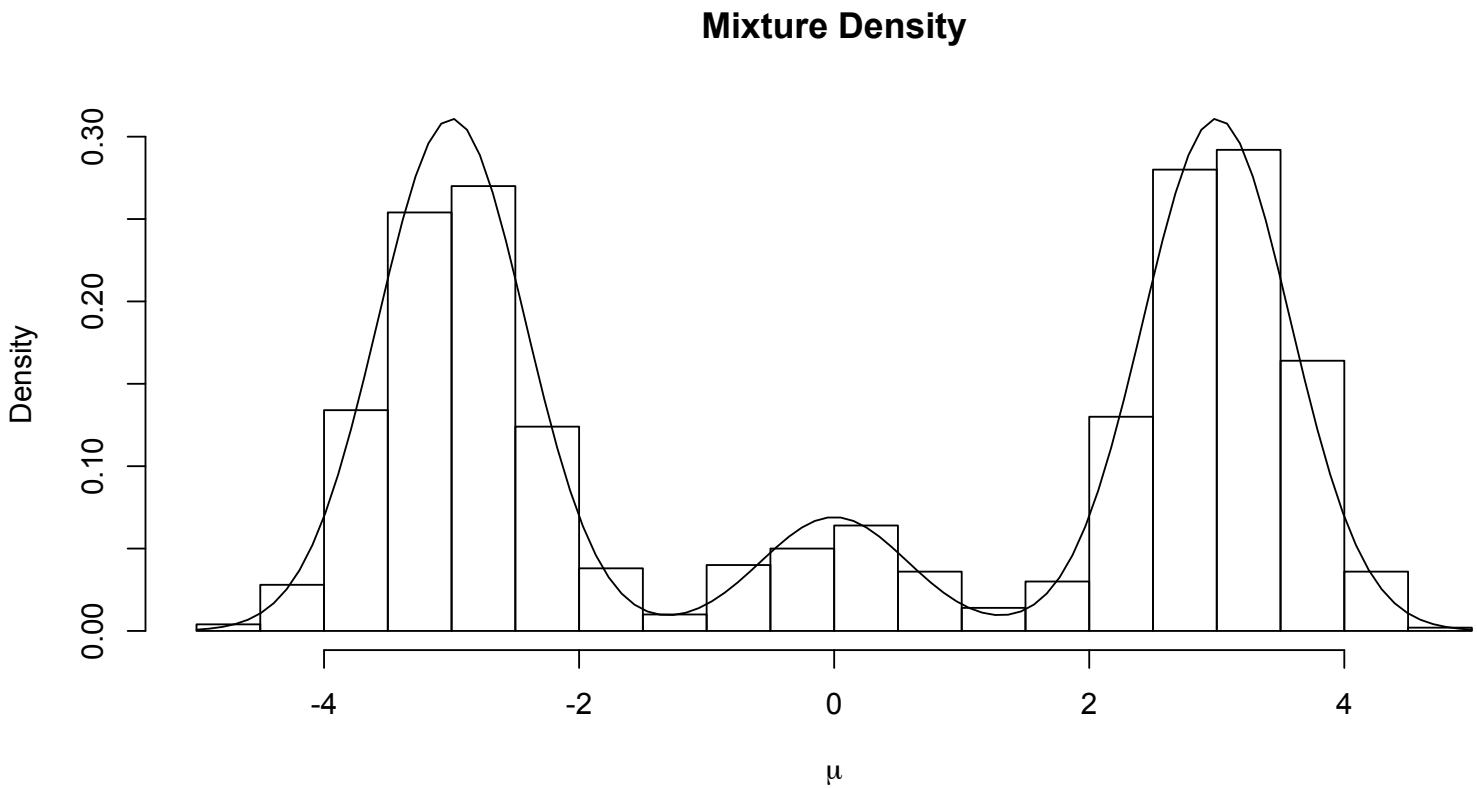
How can we draw samples from the posterior?

Introduce "mixture component indicator" δ, an unobserved latent variable which simplifies sampling

- $\delta = 1$ then $\mu \mid \delta, Y \sim \text{N}(-3, 1/3)$, and $P(\delta = 1 \mid Y) = 0.45$
- $\delta = 2$ then $\mu \mid \delta, Y \sim \text{N}(0, 1/3)$, and $P(\delta = 1 \mid Y) = 0.1$
- $\delta = 3$ then $\mu \mid \delta, Y \sim \text{N}(3, 1/3)$, and $P(\delta = 1 \mid Y) = 0.45$

Method 1: independent Monte Carlo sampling
Draw δ; Given δ, draw μ

MC density



Histogram of μ from 1000 MC draws with posterior density as a solid line

MC Variation

If we want to find the posterior mean of $g(\theta)$, then the Monte Carlo (MC) estimate based on S MC samples is

$$\hat{g}_{MC} = \frac{1}{S}\sum_{m} g(\theta^{(m)}) \rightarrow E[g(\theta) \mid Y]$$

with variance

$$V_{MC}[\hat{g}_{MC}] = E[\hat{g}_{MC} - E[\hat{g}_{MC}]]^2 = \frac{V[g(\theta) \mid Y]}{S}$$

leading to Monte Carlo Standard Error $\sqrt{V_{MC}[\hat{g}_{MC}]}$.

The posterior mean of $g(\theta)$ should be in the interval $\hat{g}_{MC} \pm 2\sqrt{V_{MC}[\hat{g}_{MC}]}$ for $\approx 95\%$ of repeated MC samples.

To increase accuracy, increase S.

Markov Chain Monte Carlo: Gibbs Sampling

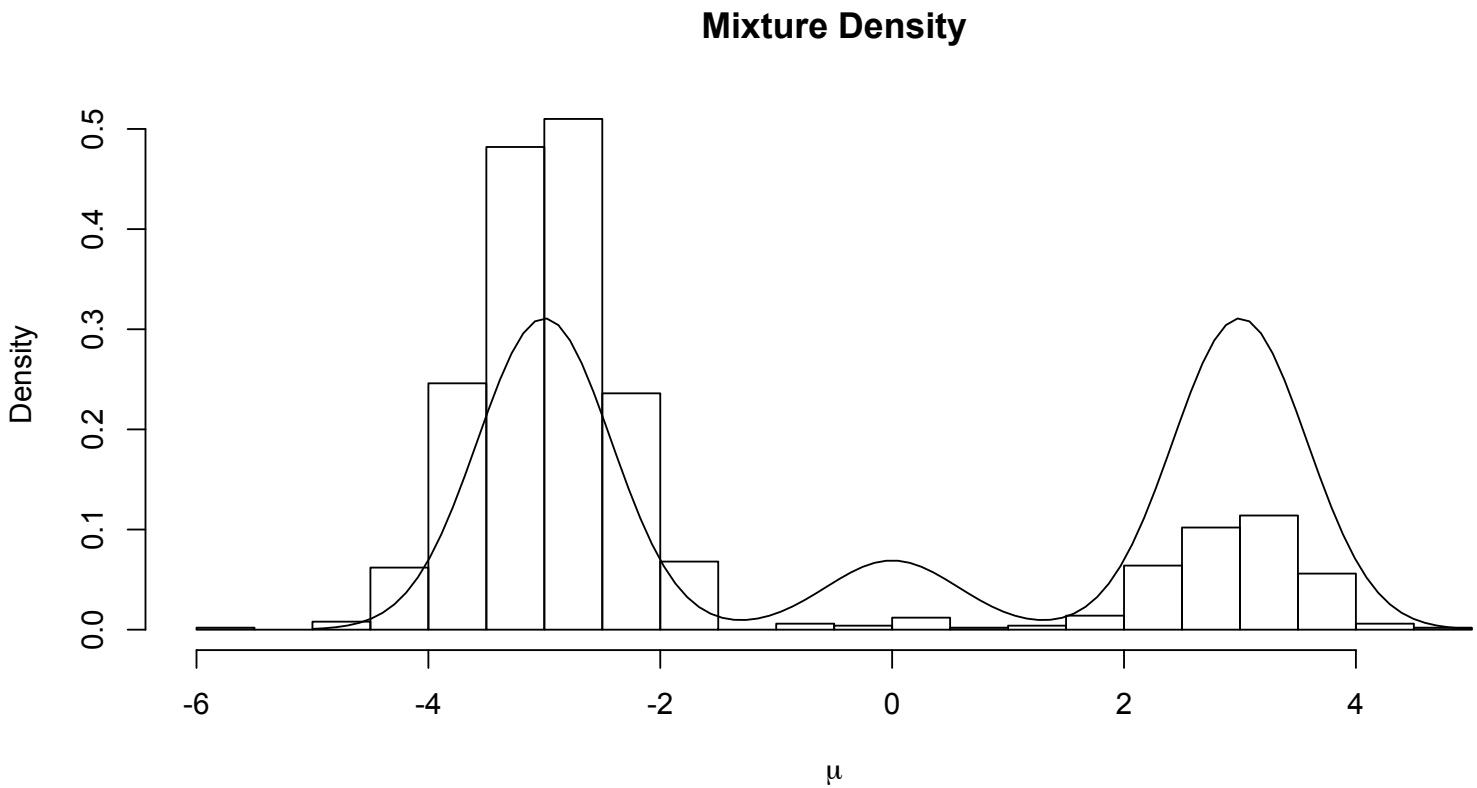
- Full conditional for $\mu \mid \delta, Y$
  - $\mu \mid \delta = 1, Y \sim \mathsf{N}(-3, 1/3)$
  - $\mu \mid \delta = 2, Y \sim \mathsf{N}(0, 1/3)$
  - $\mu \mid \delta = 3, Y \sim \mathsf{N}(3, 1/3)$

- Full conditional for $\delta \mid \mu, Y$ (use Bayes Theorem):

$$P(\delta = d \mid \mu, Y) = \frac{P(\delta = d \mid Y)\ \mathsf{dnorm}(\mu, m_d, s_d)}{\sum_{d'=1}^{3} P(\delta = d' \mid Y)\ \mathsf{dnorm}(\mu, m_{d'}, s_{d'})}$$

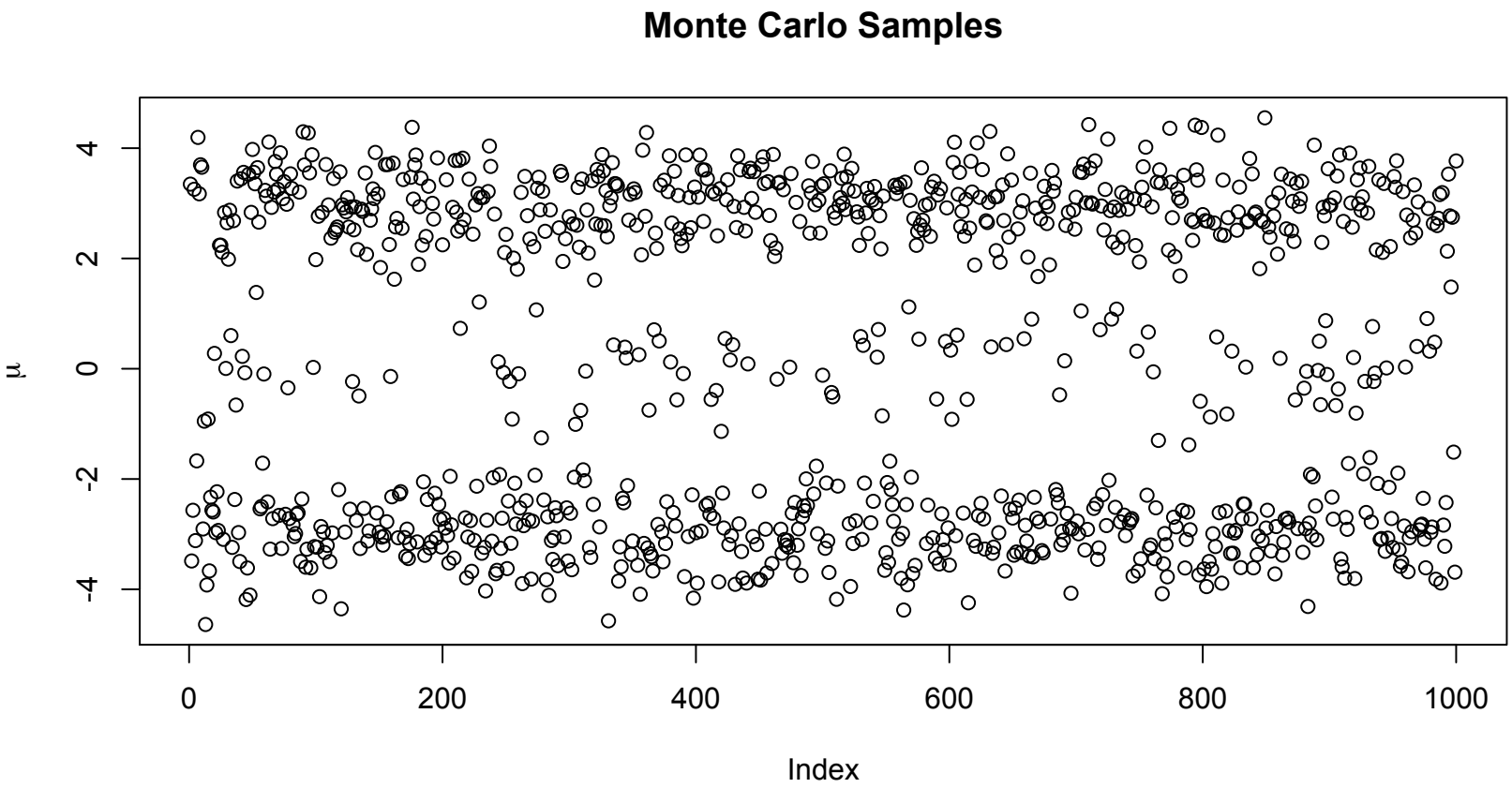
  where dnorm is the normal density.

MCMC density

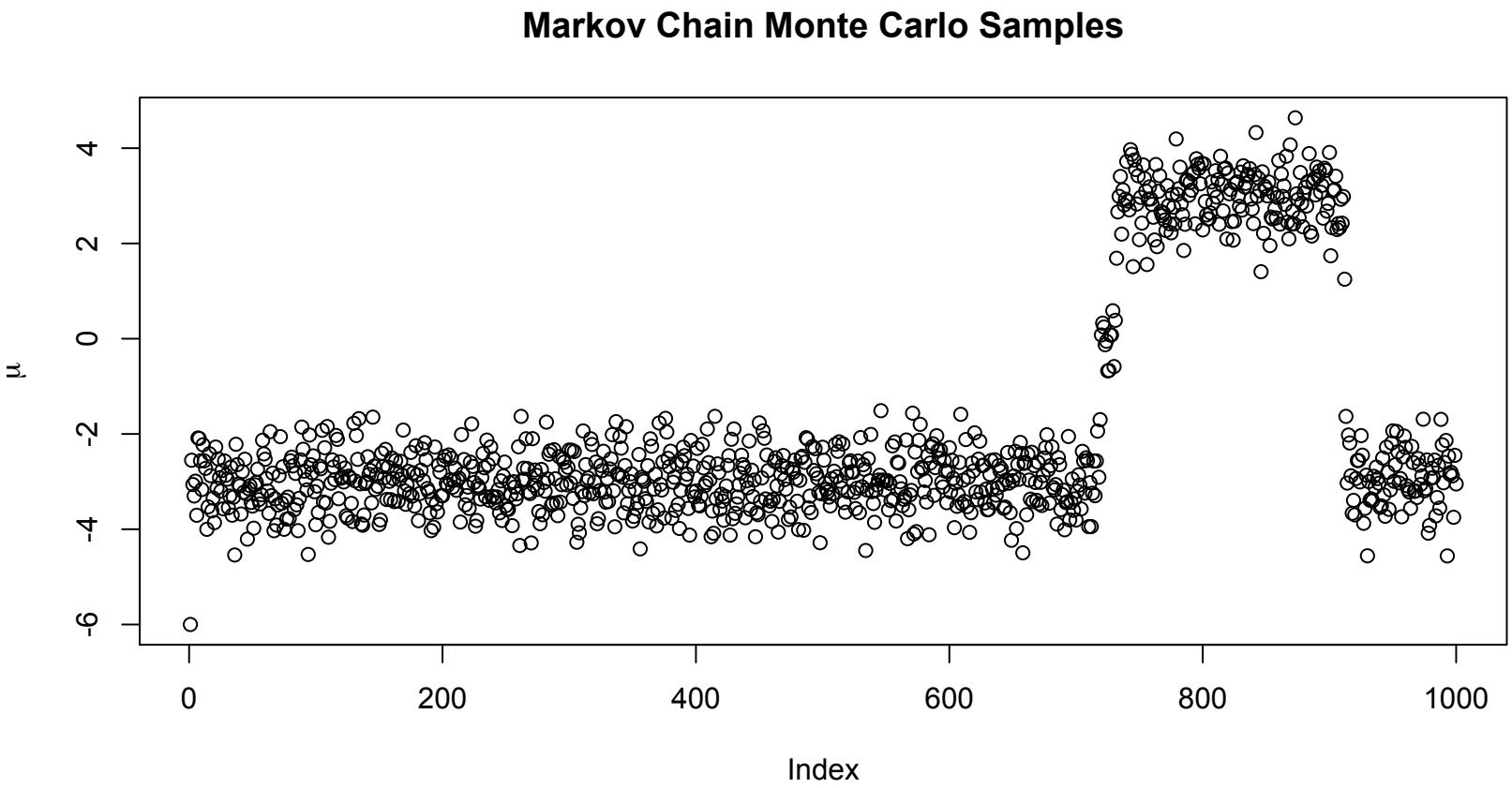


1000 draws using MCMC starting at $\mu = -6$ and $\delta = 1$

MC Trace Plot



MCMC Trace Plot



Stationarity

Partition the parameter space as follows:

- $A_1 = (-\infty, -1.5)$
- $A_2 = (-1.5, 1.5)$
- $A_3 = (1.5, \infty)$

Under the posterior, we should have

$$P(A_1) = P(A_3) > P(A_2)$$

We need the MCMC sample size S to be big enough so that

1. *stationarity or convergence*: move out of A_2 (or another areas of low probability) into higher posterior regions
2. *mixing* move between A_1 and A_3 and other high probability regions

Stickiness

The traceplots show

- MC samples can move from one region to another in 1 step (perfect mixing)
- MCMC quickly moves away from the starting value -6
- MCMC has difficulty moving between the different components and tends to get "stuck" in one component for a while (stickiness)

MCMC Variation

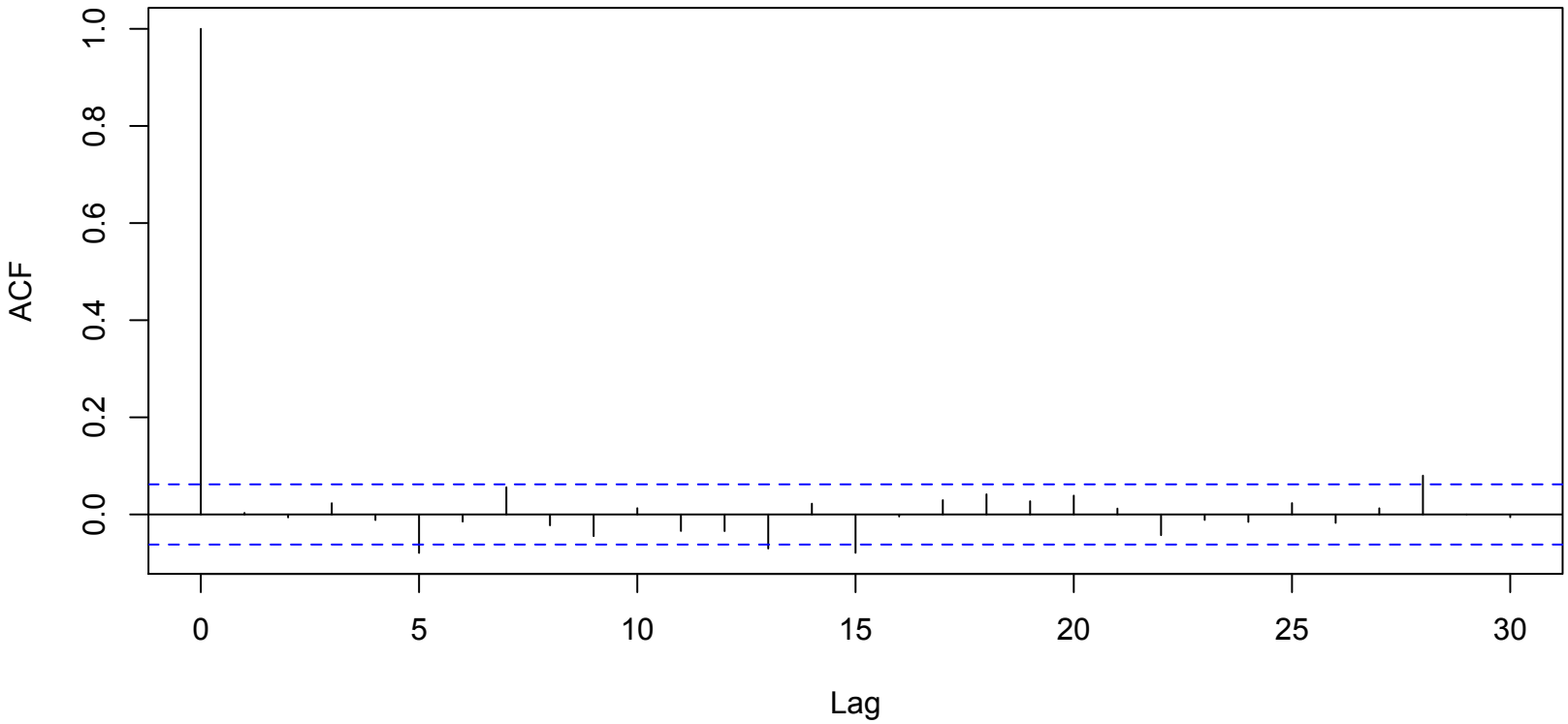
Define $\hat{g}$ like $\hat{g}_{MC}$, but using MCMC draws of $g(\theta)$. Then,

$$V_{\text{MCMC}}[\hat{g}] = V_{\text{MC}}(\hat{g}) + \sum_{s \neq t} E\left[\left(g(\theta^{(s)}) - E[g(\theta)|Y]\right)\left(g(\theta^{(t)}) - E[g(\theta)|Y]\right)\right]/S^2$$

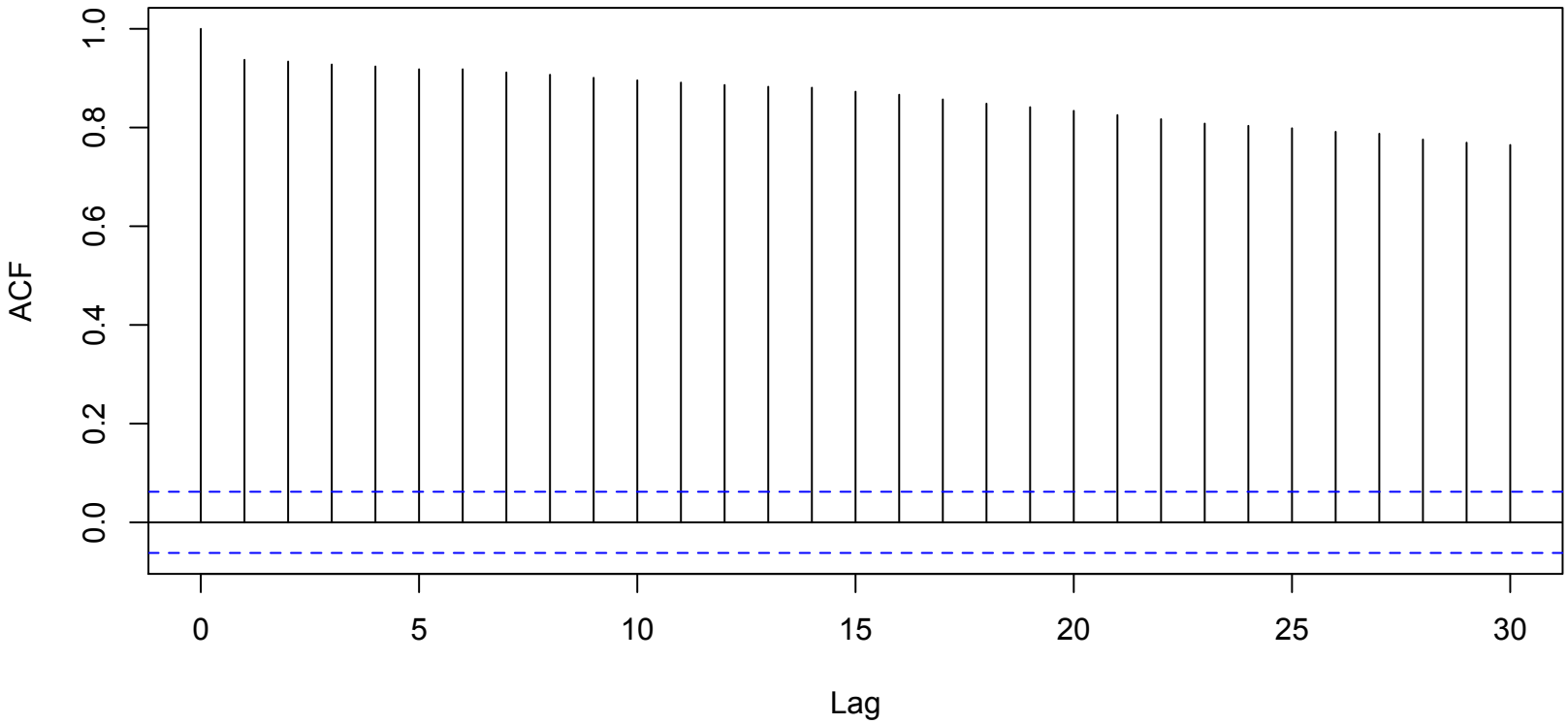
The second term depends on the autocorrelation of samples within the Markov chain

- Autocorrelation of lag t is the correlation between $g(\theta^{(s)})$ and $g(\theta^{(s+t)})$ (elements that are t time steps apart)
- often positive so MCMC variance is larger than MC variance
- high correlation is an indicator of poor mixing
- need a larger S to make V_{MCMC} comparable to V_{MC}

ACF Plots Monte Carlo Sample



ACF Plots MCMC Sample



Useful Diagnostics/Functions

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

- Trace plots: `traceplott()`
- Autocorrelation: `autocorr.plot()`
- Effective Sample Size: `effectiveSize()`
- Geweke: `geweke.diag()`
- Gelman-Rubin: `gelman.diag()`
- Heidelberg & Welch: `heidel.diag()`
- Raftery-Lewis: `raftery.diag()`

Trace Plots

Trace plots show the progress of the chain over time. The ideal trace plot has a lot of movement around the mode from iteration to iteration and not a lot of "stickiness."

Trace plots with a clear trend are evidence that MCMC sampler has not yet converged, and it needs to run longer (or there is an error in your code).

Trace plots with extreme stickiness suggest that the chain may not have explored the entire parameter space yet, and it needs to run longer.

Hard to tell convergence from trace plots alone, because you might be stuck in one region of the parameter space in all of the iterations.

Correlation Diagnostics

Independent (MC) draws have zero autocorrelations. Dependent draws (MCMC) have non-zero, usually positive, autocorrelations.

Large autocorrelations indicate that chain is not mixing well, i.e., the chain possibly has not explored the full space of the posterior distribution (has not converged).

Large autocorrelations often arise in multi-parameter MCMC algorithms when the parameters are highly correlated, e.g., coefficients in regression models.

This usually means you need to run the chains longer to feel comfortable about convergence. It also implies a smaller effective sample size....

Effective Sample Size

The effective sample size, S_{eff}, is the number of independent draws (MC samples) that your MCMC draws equate to. Computed for variance of the posterior mean of θ, so that (using definitions on previous slides)

$$V_{\text{MCMC}}(\hat{g}) = \frac{V(g)}{S_{\text{eff}}} = \frac{SV_{\text{MC}}(\hat{g})}{S_{\text{eff}}}$$

With highly correlated chains, S_{eff} can be small, even for large S. When S_{eff} is small, you need to run the chains longer to get an accurate summary of the posterior distribution.

```
> library(coda);
> Mu.MCMC = as.mcmc(Mu.MCMC);
> effectiveSize(Mu.MCMC);
    var1
2.979007
```

The precision of the MCMC estimate of the posterior mean based on 1000 samples is a good as taking 3 independent samples!

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 parts of Markov chain (by default the first 10% and the last 50%, after burn-in).

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.

coda package reports 2-tailed Z-scores. Absolute value beyond 2 suggest poor mixing.

```
> geweke.diag(Mu.MCMC)

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

  var1
-1.076
```

Gelman-Rubin Diagnostic

Gelman and Rubin (1992) propose running $m > 1$ separate Markov chains each of length S. When convergence is reached, the draws from the chains are all from the target distribution. Hence, at convergence, the output from all chains is indistinguishable.

Diagnostic based on the contrapositive: if output from chains is not indistinguishable, convergence has not been reached. Diagnostic applied to one variable at a time.

Use different starting values that are overdispersed (really spread out) relative to the posterior distribution. This helps ensure all the chains are not stuck in the same region.

Gelman-Rubin Diagnostic

Gelman-Rubin diagnostic based a comparison of within-chain and between-chain variances, which is similar to a classical analysis of variance.

For parameter θ of interest, compute its mean in each chain: $\bar{\theta}^{(1)}, \ldots, \bar{\theta}^{(m)}$. Compute sample-size weighted between variance,

$$B = S \sum_{i=1}^{m} (\bar{\theta}^{(i)} - \bar{\theta})^2 / (m-1)$$

where $\bar{\theta} = \sum_i \bar{\theta}^{(i)} / m$

Compute average within-chain variance

$$W = (1/m) \sum_{i=1}^{m} \sum_{j=1}^{S} (\theta^{(ij)} - \bar{\theta}^{(i)})^2 / (S-1)$$

where $\theta^{(ij)}$ is jth draw of θ from ith chain.

Gelman-Rubin Diagnostic

Compute ratio $\sqrt{\hat{R}} = \sqrt{V/W}$ where $V = \frac{1}{S}B + \frac{S-1}{S}W$. This is called the potential scale reduction factor.

Values of $\sqrt{\hat{R}}$ that are far from 1 suggest lack of convergence. Even better if upper 97.5% limit of interval associated with $\sqrt{\hat{R}}$ is close to one as well.

The second chain: 1000 draws using MCMC starting at $\mu = 6$ and $\delta = 3$

```
> gelman.diag(list(Mu.MCMC, Mu2.MCMC))
Potential scale reduction factors:

     Point est. Upper C.I.
[1,]       2.26       7.86
```

Summary

- Diagnostics cannot guarantee that chain has converged
- Can indicate that it has not converged

Solutions?

- Run longer and (possibly) thin output: save one output in every n iterations
- Standardize variables in multi-parameter problems (can reduce autocorrelations)
- Reparametrize model to reduce correlations among parameters
- More advanced solutions: "Block" correlated parameters together, add auxiliary variables