ST 740: Markov Chain Monte Carlo

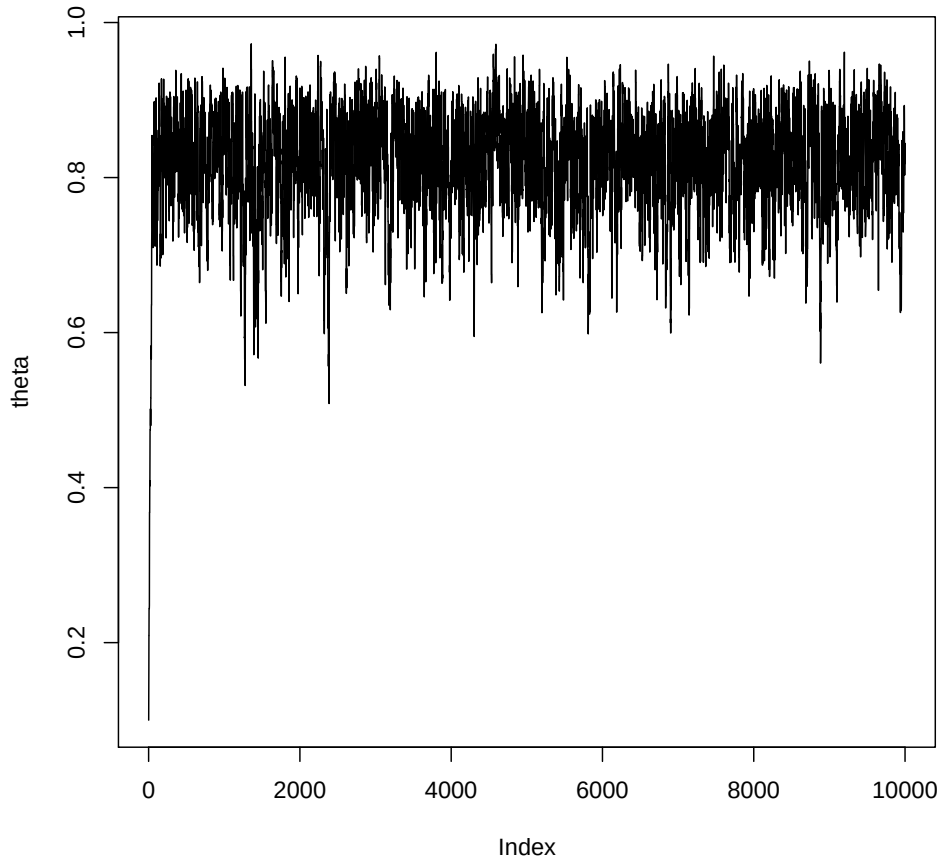
Alyson Wilson

Department of Statistics
North Carolina State University

October 14, 2012

Convergence Diagnostics: Trace Plots

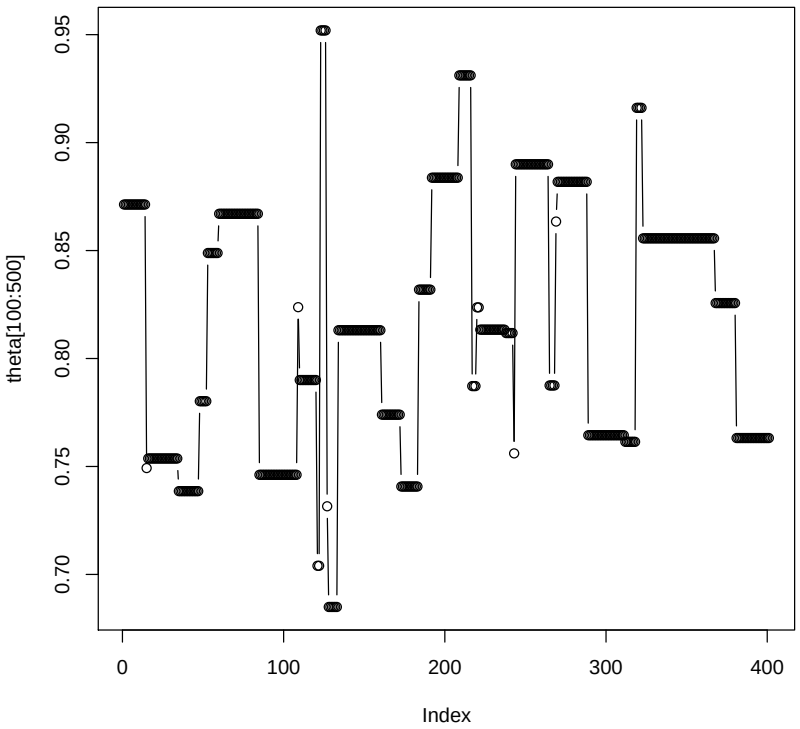
An increasing or decreasing trend in the values of the parameter indicate that the burn-in period is not over.



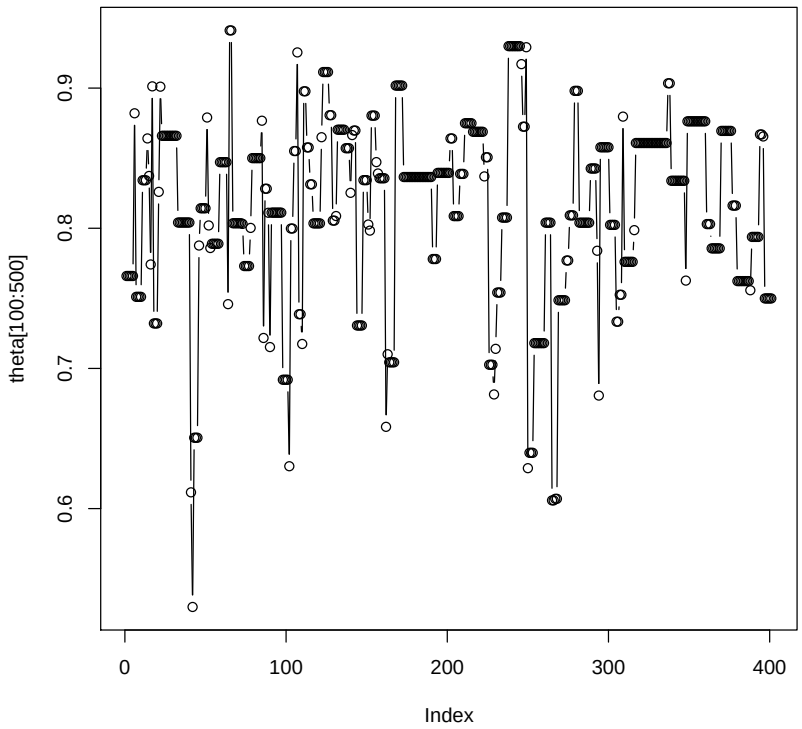
R package `coda`, function `plot(as.mcmc())`

Convergence Diagnostics: Trace Plots

Metropolis-Hastings: not moving enough (acceptance rate = 8%)

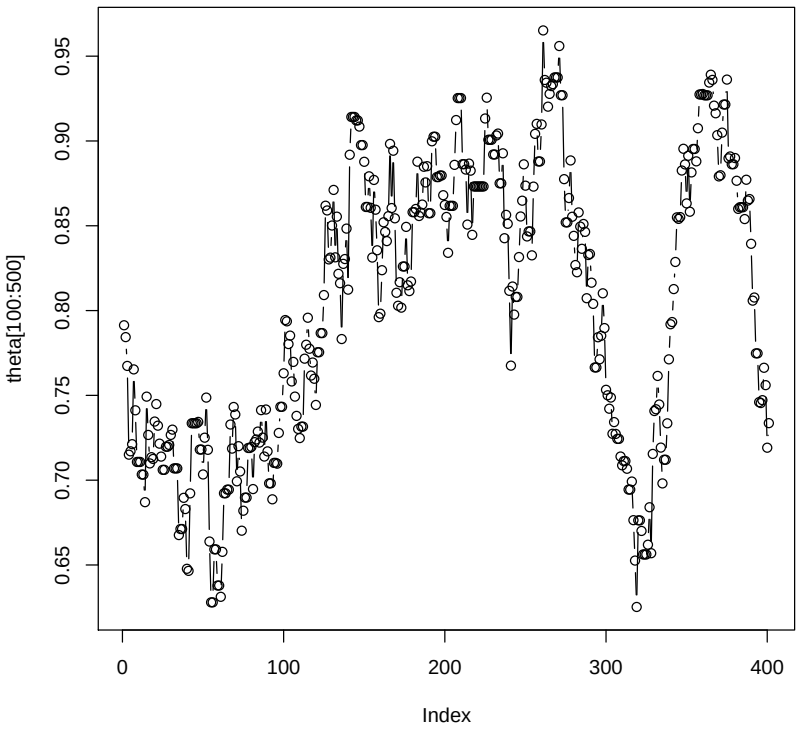


Metropolis-Hastings: moving about right (acceptance rate = 29%)

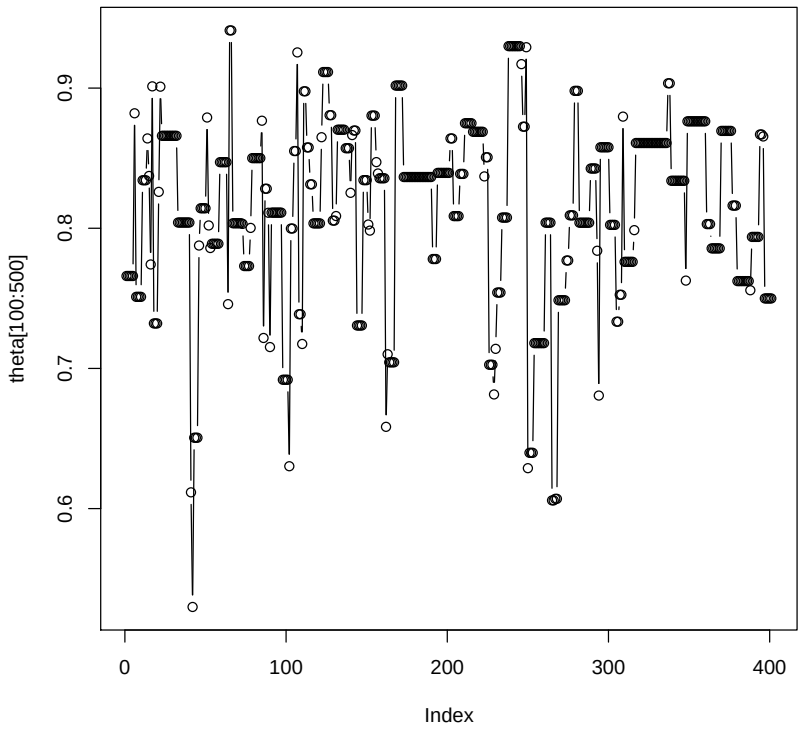


Convergence Diagnostics: Trace Plots

Metropolis-Hastings: moving too often (acceptance rate = 89%)



Metropolis-Hastings: moving about right (acceptance rate = 29%)



Convergence Diagnostics: Autocorrelation

For both Gibbs sampling and Metropolis-Hastings, $\theta^{(i+1)}$ is correlated with $\theta^{(i)}$. If this correlation is strong, then successive draws from the chain provide only marginally more information about the distribution than does a single simulated value. A chain with high correlation is said to exhibit *poor mixing*.

R package `coda`, function `autocorr.diag` calculates the autocorrelations, `autocorr.plot` plots

Convergence Diagnostics: Autocorrelation

It can sometimes be useful to think about the "equivalent number of independent observations." The formula is

$$N = n/\kappa$$

where

$$\kappa = 1 + 2\sum_{k=1}^{\infty} \rho(k)$$

where $\rho(k)$ are the lag k autocorrelations.

R package `coda`, function `effectiveSize`

Ergodic Theorem

The *ergodic theorem* is analogous to the strong law of large numbers but for Markov chains. It states that any specified function of the posterior distribution can be estimated with samples from a Markov chain in its ergodic state.

More formally, let $\theta^{(i+1)}, \ldots \theta^{(i+n)}$ be n (not necessarily consecutive) values from a Markov chain that has reached its ergodic (stationary) distribution. A statistic of interest, $h(\theta)$, can be calculated empirically

$$h(\theta) \approx \frac{1}{n} \sum_{j=i+1}^{i+n} h(\theta^{(j)})$$

Idea: We can take MCMC simulation values from the stationary distribution and safely ignore the serial nature of their production.

How many observations: Batch Means

Because the draws from MCMC algorithms are not independent, it is difficult to calculate the standard errors of MCMC-based estimators. (Note that I am talking about the errors in the estimates coming from *simulation error*, not about the inherent uncertainty in the posterior distribution.)

How many observations: Batch Means

Suppose we are interested in calculating an expectation: for concreteness, let's assume we are interested in calculating the posterior mean of θ_i. To compute the *MCMC standard error*,

- Divide the stream of simulated values $\theta_i^{(1)}, \ldots, \theta_i^{(M)}$ into b batches of size v, where $M = bv$.
- Calculate the sample mean for each batch. For batch j, denote this by $\bar{\theta}_{i,j}$.
- Suppose that you have chosen v so that the lag 1 autocorrelation in the sequence of batch means is small, say under 0.1.
- Then the standard error of the estimate $\bar{\theta}_i$ can be estimated by the sample standard deviation of the batch means divided by the square root of the number of batches.

$$s^B_{\bar{\theta}_i} = \sqrt{\frac{\sum_{l=1}^{b}(\bar{\theta}_{i,l} - \bar{\theta}_i)^2}{(b-1)b}}$$

How many observations: Batch Means

This standard error is useful for determining the accuracy of posterior means that are computed in the simulation run. If the MCMC standard error is to large, rerun the MCMC algorithm for a larger number of iterations.

R package `coda`, function `summary`, "Time-series SE"

How many observations: Batch Means

Here's the intuition (Gentle 2003):

$$
\begin{aligned}
\text{Var}(\bar{\theta}_i) &= \text{Var}\left(\frac{1}{M}\sum_{j=1}^{M}\theta_i^{(j)}\right) \\
&= \text{Var}\left(\frac{1}{b}\sum_{k=1}^{b}\left(\frac{1}{v}\sum_{j=(k-1)v+1}^{kv}\theta_i^{(j)}\right)\right) \\
&\approx \frac{1}{b^2}\sum_{k=1}^{b}\text{Var}\left(\frac{1}{v}\sum_{j=(k-1)v+1}^{kv}\theta_i^{(j)}\right) \\
&\approx \frac{1}{b}\text{Var}(\bar{\theta}_{i(v)})
\end{aligned}
$$

where $\bar{\theta}_{i(v)}$ is the mean of a batch of length v.

How many observations: Batch Means

- You estimate $\text{Var}(\bar{\theta}_v)$ with the sample variance of the b batch means.
- If the batches are long enough, it may be reasonable to assume the means have a common variance.
- You also want the batches to be as small as possible and still have means that are independent.
- Default choice is $v = 100$.

How many observations: Raftery-Lewis

This diagnostic considers how *accurate* you want your posterior summaries to be. You input three values.

1. Which quantiles are you most interested in? Usually 2.5% or 97.5%, since they are the basis of a 95% interval estimate (and perhaps harder to sample from than the middle of the distribution).
2. How close to the nominal levels would you like the estimated quantiles to be? The default is 0.5%, so that if the left side of the interval is supposed to be at 2.5%, it is actually between 2.0% and 3.0%.
3. What is the minimum probability with which you want to achieve these accuracy goals? The default is 95%.

R package `coda`, function `raftery.diag`, you feed in a preliminary MCMC run

How many observations: Raftery-Lewis

Outputs of interest

- Recommended length of burn-in, beyond what you've already done
- Recommended total run length (including burn-in)

To Thin or Not to Thin

Thinning the Markov chain means keeping only every k^{th} draw, where k is chosen so the autocorrelation is small.

Whether or not the sequences are thinned, if the sequences have reached approximate convergence, they can be used directly for inference.

Convergence Diagnostics: Gelman-Rubin

- Operationally, effective convergence of Markov chain simulation has been reached when inferences for quantities of interest do not depend on the starting point of the simulations.
- Start off with crude estimates and possibly a mode-based approximation to the posterior distribution.
- Run a few chains from different starting points: sample from the mode-based approximation (think multivariate t).
- Compare the trace plots for each of the parameters and make sure they look similar.
- Compare 95% posterior credible intervals from each chain for different parameters; also compare these with normal approximation results

Correlated Parameters

If there is high correlation between parameters, mixing of the Markov chains is often slow.

1. Consider sampling from parameter "batches." Use a proposal density that has a covariance structure like the distribution you are sampling from. (Think `laplace` and multivariate t.)
2. Use linear transformations to approximately orthogonalize the parameters. If the approximate covariance matrix from the normal approximation is $\Sigma = KK^T$, where K is calculated using the Cholesky decomposition. Consider variables $\eta = K^{-1}\theta$.
3. Center covariates.

Markov Chain Monte Carlo Strategy

- Start off with crude estimates and possibly a mode-based approximation to the posterior distribution.
- If possible, simulate from the posterior distribution directly.
- If not, choose a different strategy. Most likely, the best approach will be MCMC. The updating can be done one parameter at a time or with parameters in batches. Work on the log scale. Consider transforming all of the parameters to the real line.
- For parameters (or batches of parameters) whose conditional posterior distributions have standard forms, use the Gibbs sampler.
- For parameters whose conditional distributions do not have standard forms, consider Metropolis-Hastings. Tune the scale of the proposal distribution so that acceptance rates are between 25% and 40%.

Markov Chain Monte Carlo Strategy

- Note: In practice, the Metropolis-Hastings algorithm is often found as a substep in a larger Gibbs sampling algorithm, used to generate from awkward full conditionals. Such hybrid Gibbs-Metropolis applications are sometimes known as *Metropolis within Gibbs* or *Metropolis substeps*. Users sometimes worry about how many substeps should be used. Fortunately, a single substep is sufficient. When we encounter an awkward full conditional, we simply draw on Metropolis candidate, accept or reject it, and move on.
- Consider constructing a transformation so that the parameters are approximately independent.
- Run one Markov chain and monitor the mixing of the sequence. Run until approximate convergence appears to have been reached for each parameter of interest. If you've run for a long time and you aren't converging, see "constructing a transformation."

Markov Chain Monte Carlo Strategy

- Once you're basically happy with the MCMC algorithm, run the Raftery diagnostic to get an approximate number of samples to draw.
- Start several MCMC chains with parameter values taken from the crude estimates or mode-based approximations, with noise added so they are overdispersed with respect to the target distribution.
- Make sure all of the chains are sampling from the same distribution.
- If you feel you have converged, summarize inference about the posterior distribution by using the iterates from all of the "converged part" of the simulated sequences.
- Compare the posterior inferences from the Markov chain to the approximate distributions used to start the simulation. If they are not similar with respect to locations and *approximate* distributional shape, check for errors before believing the Markov chain has produced a better result.