



STK4051/STK9051

Computational Statistics

Riccardo De Bin

`debin@math.uio.no`

Outline of the lecture

- Markov chain Monte Carlo
 - Implementation
 - Ensuring Good Mixing and Convergence
 - Simple Graphical Diagnostics
 - Burn-in and Run Length
 - Choice of Proposal
 - Reparameterization
 - Comparing Chains: Effective Sample Size
 - Number of Chains
 - Practical Implementation Advice
 - Using the Results
- Advanced Topics in MCMC

Ensuring Good Mixing and Convergence: summary

Strategies and diagnostics:

1. simple graphical diagnostics;
2. burn-in and run length;
3. choice of proposal;
4. reparameterization;
5. effective sample size;
6. number of chains.

Ensuring Good Mixing and Convergence: Simple Graphical Diagnostics

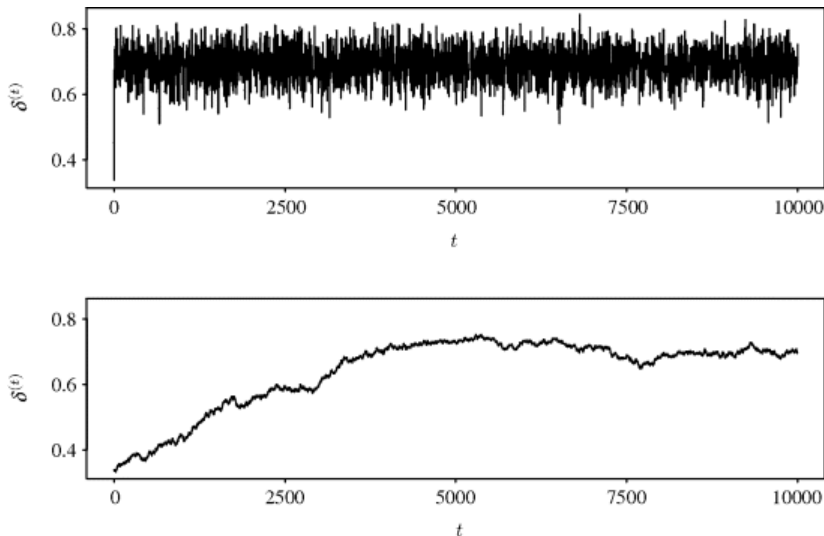
Sample paths:

- iteration number versus realizations of $\mathbf{X}^{(t)}$ for $t = 0, 1, \dots$
- also called trace or history plots;
- we want to see a very wiggly line that (perhaps) moves away from the starting value.

Autocorrelation function (ACF) plot:

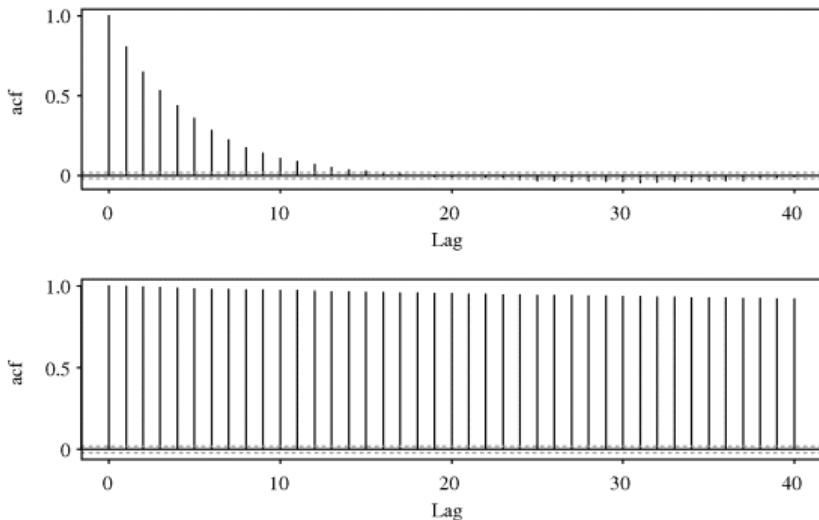
- the lag versus the correlation between iterates at that lag;
- slow decay in the acf suggests poor mixing;
- multi-parameter problems:
 - ▶ examine cross-correlations between related parameters;
 - ▶ high cross-correlations may indicate poor mixing of the chain.

Ensuring Good Mixing and Convergence: Simple Graphical Diagnostics



(Givens & Hoeting, 2013, figure 7.5)

Ensuring Good Mixing and Convergence: Simple Graphical Diagnostics



(Givens & Hoeting, 2013, figure 7.8)

Burn-in and run length: Burn-in

Burn-in:

- initial MCMC iterates,
 - do not have exactly the correct marginal distribution;
 - strong dependency on the initial point (or distribution);
- to reduce the severity of this problem, the first several thousand values from the chain are typically discarded.

Gelman and Rubin statistic, R :

- commonly used to determine burn-in period and run length;
- motivated by an analysis of variance (ANOVA);
- based on the comparison of the output of several chains;
- the burn-in period or MCMC run-length should be increased if
 - between-chain variance $\gg$ the within-chain variance.

Burn-in and run length: Gelman and Rubin statistic

Initial steps:

- run the MCMC algorithm J separate times;
- to produce equal-length chains ($J \geq 2$);
- with starting values all over the support of the target density.

Notation:

- L is the length of each chain w/o the D burn-in iterations;
- X is the variable of interest,
 - ▶ with $x_j^{(t)}$ be its value at the t th iteration of the j th chain;
- for the j th chain,
 - ▶ the D values $x_j^{(0)}, \dots, x_j^{(D-1)}$ are discarded;
 - ▶ the L values $x_j^{(D)}, \dots, x_j^{(D+L-1)}$ are retained.

Burn-in and run length: Gelman and Rubin statistic

Let $\bar{x}_j = \frac{1}{L} \sum_{t=D}^{D+L-1} x_j^{(t)}$ and $\bar{x} = \frac{1}{J} \sum_{j=1}^J \bar{x}_j$ and define the between-chain variance as

$$B = \frac{L}{J-1} \sum_{j=1}^J (\bar{x}_j - \bar{x})^2.$$

Now if we define the within-chain variance for the j th chain to be $s_j^2 = \frac{1}{L-1} \sum_{t=D}^{D+L-1} (x_j^{(t)} - \bar{x}_j)^2$, then

$$W = \frac{1}{J} \sum_{j=1}^J s_j^2$$

represents the mean of the J within-chain estimated variances.

Burn-in and run length: Gelman and Rubin statistic

With these pieces, we can define the Gelman and Rubin statistic as

$$R = \frac{\frac{L-1}{L}W + \frac{1}{L}B}{W}.$$

Note:

- if all the chains are stationary, then both the numerator and the denominator should estimate the marginal variance of X ;
- if there are notable differences between the chains, then the numerator will exceed the denominator;
- in theory, $\sqrt{R} \rightarrow 1$ as $L \rightarrow \infty$,
 - ▶ true for the adjusted estimator $\hat{R} = \frac{J+1}{J}R + \frac{L-1}{JL}$;
- practical advice: $\sqrt{R} < 1.2$ is acceptable (or $\sqrt{\hat{R}} < 1.1$).

Burn-in and run length: Gelman and Rubin statistic

Remarks:

- if the chosen burn-in period did not yield an acceptable result,
 - ▶ D should be increased;
 - ▶ L should be increased;
 - ▶ or preferably both D and L should be increased;
- if the $x_j^{(t)}$ iterates are transformed so that their distribution is approximately normal, better performance of this diagnostic;
- reparameterize the model and rerun the chain.

Possible issues:

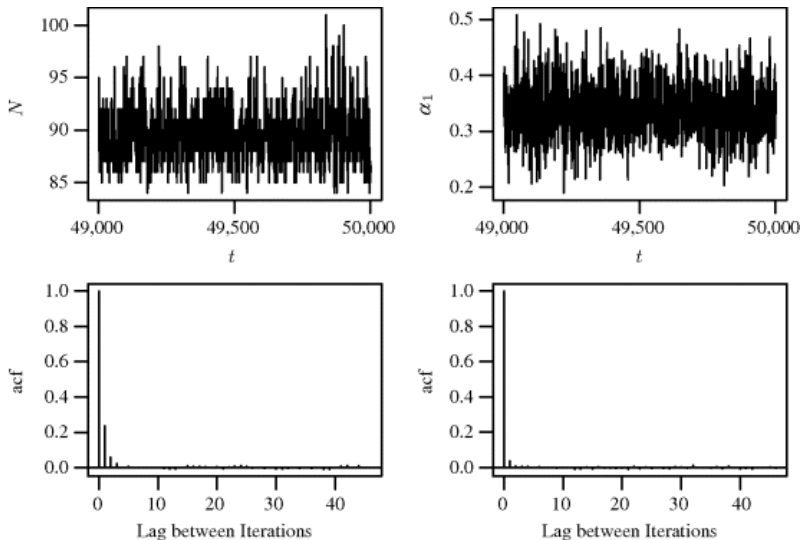
- hard to select suitable starting values,
 - ▶ especially tricky in the case of multimodal f ;
 - ▶ the procedure does not work if all of the chains become stuck in the same subregion or mode;
- possible misleading results when f is multidimensional.

Implementation: Fur Seal Pup Capture - Recapture Example

Diagnostics in the Fur Seal Pup Capture - Recapture Example:

- basic Gibbs sampler;
- $J = 5$ runs of 100000 iterations, each with a burn-in of 50000;
- for N :
 - ▶ the Gelman–Rubin statistic is equal to 0.999995;
 - ▶ the chain $N^{(t)}$ is roughly stationary;
- for $(\alpha_1, \dots, \alpha_7)$:
 - ▶ the Gelman–Rubin statistics are all close to one;
 - ▶ the capture probabilities exhibit little correlation between MCMC samples.

Implementation: Fur Seal Pup Capture - Recapture Example



(Givens & Hoeting, 2013, figure 7.9)

Ensuring Good Mixing and Convergence: Choice of Proposal

MCMC algorithms are affected by the proposal distribution:

- aim for a proposal distribution that is more diffuse than f ,
 - ▶ tail behaviour of g more important than its resemblance to f ;
 - ▶ faster convergence if f/g is bounded;
- the variance of the proposal distribution can be selected through an informal iterative process:
 - ▶ start a chain, and monitor the proportion of acceptances;
 - ▶ adjust the spread of the proposal distribution accordingly;
- for a Metropolis algorithm with normal target and proposal distributions, aim for an acceptance between 25 and 50%,
 - ▶ best choice for one-dimensional problems being about 44%;
 - ▶ about 23.4% for higher-dimensional problems.

Ensuring Good Mixing and Convergence: Reparameterization

To apply such rules, care must be taken:

- target and proposal distributions must be roughly normal;
- at least assure they have simple, unimodal distributions;
- if the target distribution is multimodal,
 - the chain may get stuck in one mode;
 - without adequately explore the rest of the parameter space;
- issue (for most MCMC methods): we need as much knowledge as possible about the unknown target distribution.

Advanced methods (e.g., Adaptive Markov chain Monte Carlo) tune the proposal distribution during the MCMC algorithm:

- in some implementations, do not require the user to stop, tune, and restart the algorithm on multiple occasions.

Ensuring Good Mixing and Convergence: Reparameterization

Model reparameterization can provide substantial improvements:

- high correlation between the elements of $\mathbf{X}$ can slow down the convergence;
- reparameterization is the primary strategy for reducing it;
- example with a Gibbs sampler:
 - ▶ f bivariate normal with very strong positive correlation;
 - ▶ both univariate conditionals will allow only small steps;
 - ▶ $\mathbf{Y} = (\mathbf{X}_1 + \mathbf{X}_2, \mathbf{X}_1 - \mathbf{X}_2)$ yields one univariate conditional on the axis of maximal variation in $\mathbf{X}$;
 - ▶ easier to move from one point to any other point.

Reparameterization approaches are typically model specific:

- center and scale the covariates to reduce correlations;
- use hierarchical centering: parameters in a hierarchical Bayesian model are moved down the hierarchy.

Implementation: Fur Seal Pup Capture - Recapture Example

Fur Seal Pup Capture - Recapture Ex. with hybrid Gibbs sampler:

- a Metropolis-Hastings step (with random walk) is required to sample (θ_1, θ_2) from the conditional posterior;
- the prior distribution restricts (θ_1, θ_2) to be larger than 0;
- this constraint can impede MCMC performance, particularly if there is high posterior density near the boundary;
- to improve performance, transform (θ_1, θ_2) to

$$\mathbf{U} = (U_1, U_2) = (\log \theta_1, \log \theta_2);$$

- proposal from $N(0, 0.085^2 I_2) \rightarrow$ acceptance rate $\approx 23\%$;
- this requires transformation of the conditional densities. . .

Implementation: Fur Seal Pup Capture - Recapture Example

Reparametrised conditional posterior densities:

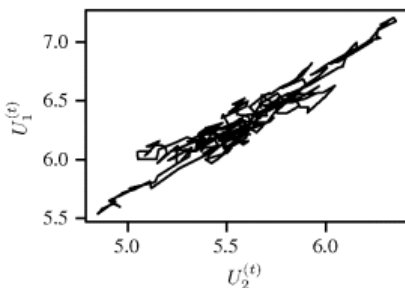
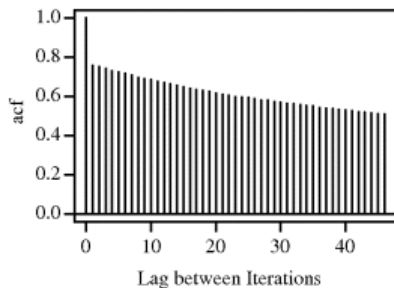
$$\begin{aligned} N - 84 | \cdot &\sim \text{NegBin} \left(84, 1 - \prod_{i=1}^7 (1 - \alpha_i) \right) \\ \alpha_i | \cdot &\sim \text{Beta}(c_i + \exp\{u_1\}, N - c_i + \exp\{u_2\}) \quad \text{for } i = 1, \dots, 7 \\ \theta_1, \theta_2 | \cdot &\sim k_u \exp\{u_1 + u_2\} \left[\frac{\Gamma(\exp\{u_1\} + \exp\{u_2\})}{\Gamma(\exp\{u_1\})\Gamma(\exp\{u_2\})} \right]^7 \\ &\quad \prod_{i=1}^7 \alpha_i^{\exp\{u_1\}} (1 - \alpha_i)^{\exp\{u_2\}} \exp \left\{ -\frac{\exp\{u_1\} + \exp\{u_2\}}{1000} \right\}. \end{aligned}$$

Implementation: Fur Seal Pup Capture - Recapture Example

Results based on a chain of 100000 iterations with the first 50000 iterations discarded for burn-in:

- the Gelman and Rubin statistic for N is equal to 1.00047;
- which suggests the $N^{(t)}$ chain is roughly stationary;
- however, the autocorrelation plots indicate high correlation between iterations,
 - ▶ correlation of 0.6 between MCMC samples 40 iterations apart;
 - ▶ for comparison, it was 0 for lag 2 in the simple Gibbs sampler;
- the bivariate sample path of $U_1^{(t)}$ vs $U_2^{(t)}$ indicates a high correlation between the parameters too;
- lack of convergence of the MCMC algorithm (or at least poor behaviour of the chain) for the hybrid algorithm.

Implementation: Fur Seal Pup Capture - Recapture Example



(Givens & Hoeting, 2013, figure 7.10)

Comparing Chains: Effective Sample Size

If MCMC realizations are highly correlated,

- the information gained from each iteration of the MCMC algorithm will be much less than suggested by the run length;
- the reduced information is called the **effective sample size**;
- it is equivalent to that contained in a smaller i.i.d. sample;
- the difference indicates the loss of efficiency.

Comparing Chains: Effective Sample Size

To estimate the effective sample size,

- estimate the autocorrelation time, a measure of the autocorrelation between realizations and their rate of decay, by

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

where $\rho(k)$ is the autocorrelation between realizations that are k iterations apart;

- common approach: truncate the summation when $\hat{\rho} < 0.1$;
- the effective sample size for an MCMC run with L iterations after burn-in can be estimated using $L/\hat{\tau}$.

Comparing Chains: Effective Sample Size

Effective sample size can be used:

- to compare the efficiency of competing MCMC samplers;
 - ▶ for a fixed number of iterations, an MCMC algorithm with a larger effective sample size is likely to converge more quickly;
 - ▶ e.g., blocked vs unblocked Gibbs sampler;
- as a diagnostic tool:
 - ▶ example: Bayesian model with two parameters (α, β) ;
 - ▶ MCMC algorithm run for $L = 10000$ iterations after burn-in;
 - ▶ effective sample size of 9500 iterations for α ;
 - ▶ effective sample size of 500 iterations for β ;
 - ▶ convergence for β is highly suspect.

Ensuring Good Mixing and Convergence: Number of Chains

A difficult problem to diagnose: chain has become stuck in one or more modes of the target distribution.

Partial solution: run multiple chains from diverse starting values and then compare the within- and between-chain behavior.

Diagnostics: sample path plots.

Ensuring Good Mixing and Convergence: Practical Advice

In practice, what values should be used for:

- the number of chains?
- the number of iterations for burn-in?
- the length of the chain after burn-in?

No general guidelines,

- problem specific;
- availability of computational time.

As computing power continues to grow, so too will the scope and intensity of MCMC efforts.

Using the Results: Marginalization

Marginalization:

- if $\{\mathbf{X}^{(t)}\}$ represents a p -dimensional Markov chain, then the limiting distribution of $\{X_i^{(t)}\}$ is the i th marginal of f ;
- if the focus is only on a property of this marginal,
 - ▶ discard the rest of the simulation;
 - ▶ analyse the realizations of $X_i^{(t)}$.

Probability estimates:

- the probability of any event can be estimated by the frequency of that event in the chain.

Using the Results: Mean

Mean:

- the most used estimator is based on the empirical average;
- discard the burn-in, then calculate the desired statistic by

$$E\{\widehat{h(X)}\} = \frac{1}{L} \sum_{t=D}^{D+L-1} h(\mathbf{X}^{(t)})$$

where L is the chain's length after discarding D iterates,

- ▶ consistent estimator even if the $X(t)$ are serially correlated;
 - ▶ asymptotic arguments in favour of using no burn-in;
 - ▶ burn-in employed to reduce the influence of the initial iterates.
- variance reduction techniques such as Rao–Blackwellization can be used to reduce the Monte Carlo variability of estimators based on the chain output.

Using the Results: standard error

Monte Carlo standard error:

- estimate of the variability in the estimator if the MCMC algorithm were to be run repeatedly;
- naive approach:
 - ▶ standard error of realizations (after burn-in) divided by $\sqrt{L}$;
 - ▶ can underestimate the standard error because correlation;
- obvious correction:
 - ▶ use a systematic subsample of every k th iterate after burn-in;
 - ▶ inefficient approach;
- batch method:
 - ▶ separate the L iterates into batches of b consecutive iterations;
 - ▶ compute the mean of each batch;
 - ▶ take the standard deviation of these means divided by $\sqrt{L/b}$;
 - ▶ recommended batch size: $b = \lfloor L^{1/a} \rfloor$, with $a = 2$ or 3 .

Using the Results

Density/histogram:

- common to use histogram of the raw iterates;
- once could use a kernel density estimate.

Quantiles:

- just use the empirical quantiles;
- used to compute the $100(1 - \alpha)\%$ credible intervals.

Other summary statistics: Be creative!

Advanced Topics in MCMC: Adaptive MCMC

MCMC algorithms requires tuning to improve convergence:

- e.g., variance of a normally distributed proposal distribution in a Metropolis-Hastings algorithm;
- in the exercise class, trial and fail;
- impractical when the number of parameters is large.

Adaptive Markov chain Monte Carlo (AMCMC) algorithms:

- allow for automatic tuning of the proposal distribution as iterations progress;
- conditions to ensure that the chain produced by the algorithm has the correct stationary distribution;
 - ▶ diminishing adaptation: as $t \rightarrow \infty$, the parameters in the proposal depend less and less on earlier states of the chain;
 - ▶ bounded convergence: consider time until near convergence;

Advanced Topics in MCMC: Reversible Jump MCMC

Reversible jump Markov chain Monte Carlo:

- useful when the dimensionality of $X(t)$ change with t ;
- develops a chain that allows for changes in the dimension of the parameter space from one iteration to the next;
- the jumps are the switches from one parameter space to another;
- e.g., posterior for different models with different variables,
 - ▶ compare models of different dimension and select the best;
 - ▶ predict a new values for all models.

Advanced Topics in MCMC: other approaches

Many other approaches in the book (Chapter 8), including:

- Auxiliary Variable Methods,
 - ▶ Simulated Tempering;
 - ▶ Slice Sampler;
- further Metropolis-Hastings Algorithms,
 - ▶ Hit-and-Run Algorithm;
 - ▶ Multiple-Try Metropolis-Hastings Algorithm;
 - ▶ Langevin Metropolis-Hastings Algorithm;
- Perfect Sampling.

Note:

- not so “advanced” any more;
- constant newer methods / improvements:
 - ▶ Hamiltonian Monte Carlo;
 - ▶ mode jumping MCMC;
 - ▶ ...

References |

GIVENS, G. H. & HOETING, J. A. (2013). *Computational Statistics*. Wiley.