

Section 5

Bayesian Computing

Bayesian computing

- ▶ Give the prior and data, the posterior is fixed and a Bayesian analysis boils down to summarizing the posterior
- ▶ We need point estimates, credible sets, etc
- ▶ Summarizing a p -dimensional posterior distribution is challenging for large p
- ▶ In the 80's, Bayesian computing was unable to do this for more than a few parameters
- ▶ In the 90's, new algorithms were developed that revolutionized Bayesian statistics
- ▶ Understanding these algorithms is obviously important

Approaches to Bayesian computing

Some approaches to dealing with complicated joint posteriors:

- ▶ Just use a point estimate (e.g., MAP), ignore uncertainty
- ▶ Approximate the posterior as Gaussian
- ▶ Numerical integration
- ▶ Markov Chain Monte Carlo (MCMC) sampling

Outline

These notes cover Chapter 3

- ▶ Deterministic methods
 - ▶ MAP estimation
 - ▶ Numerical integration
 - ▶ Bayesian Central Limit Theorem
- ▶ MCMC algorithms
 - ▶ Gibbs sampling
 - ▶ Metropolis-Hastings sampling
- ▶ Just Another Gibbs Sampler (JAGS)
- ▶ Diagnostic and improving convergence
 - ▶ Setting initial values
 - ▶ Convergence diagnostics
 - ▶ Improving convergence
 - ▶ Dealing with large datasets

MAP estimation

- ▶ Sometimes you don't need an entire posterior distribution and a single point estimate will do
- ▶ Example: prediction in machine learning
- ▶ The Maximum a Posteriori (MAP) estimate is the posterior mode

$$\hat{\theta}_{MAP} = \operatorname{argmax}_{\theta} p(\theta|\mathbf{Y}) = \operatorname{argmax}_{\theta} \log[f(\mathbf{Y}|\theta)] + \log[\pi(\theta)]$$

- ▶ This is similar to the maximum likelihood estimation but includes the prior

Univariate example

Say $Y|\theta \sim \text{Poisson}(N\theta)$ and $\theta \sim \text{Gamma}(a, b)$, find $\hat{\theta}_{MAP}$

Bayesian central limit theorem

- ▶ Another simplification is to approximate the posterior as Gaussian
- ▶ Bernstein-Von Mises Theorem: As the sample size grows the posterior doesn't depend on the prior
- ▶ Frequentist result: As the sample size grows the likelihood function is approximately normal
- ▶ Bayesian CLT: For large n and some other conditions $\theta|\mathbf{Y} \approx \text{Normal}$

Bayesian central limit theorem

- ▶ Bayesian CLT: For large n and some other conditions

$$\theta \sim \text{Normal}[\hat{\theta}_{MAP}, \mathbf{I}(\hat{\theta}_{MAP})^{-1}]$$

- ▶ \mathbf{I} is Fisher's information matrix
- ▶ The (j, k) element of \mathbf{I} is

$$-\frac{\partial^2}{\partial \theta_j \partial \theta_k} \log[p(\theta|\mathbf{Y})]$$

evaluated at $\hat{\theta}_{MAP}$

- ▶ We have marginal and conditional means, standard deviations and intervals for the normal distribution

Univariate example

Say $Y|\theta \sim \text{Binomial}(n, \theta)$ and $\theta \sim \text{Beta}(0.5, 0.5)$, find the Gaussian approximation for $p(\theta|\mathbf{Y})$

Illustration of the Bayesian CLT

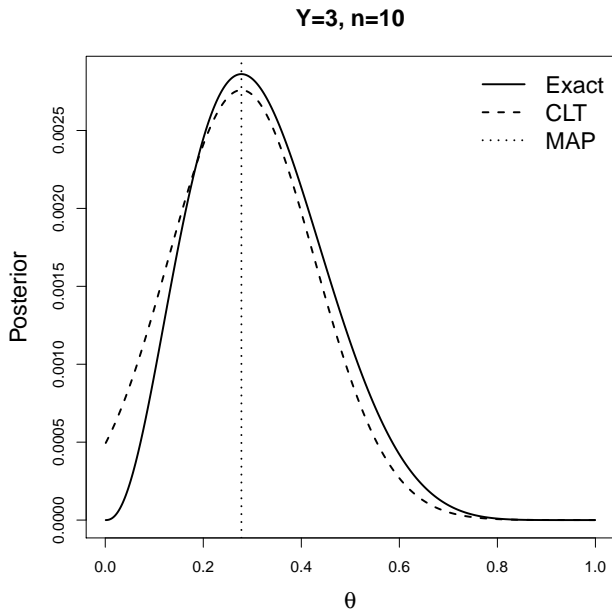


Illustration of the Bayesian CLT

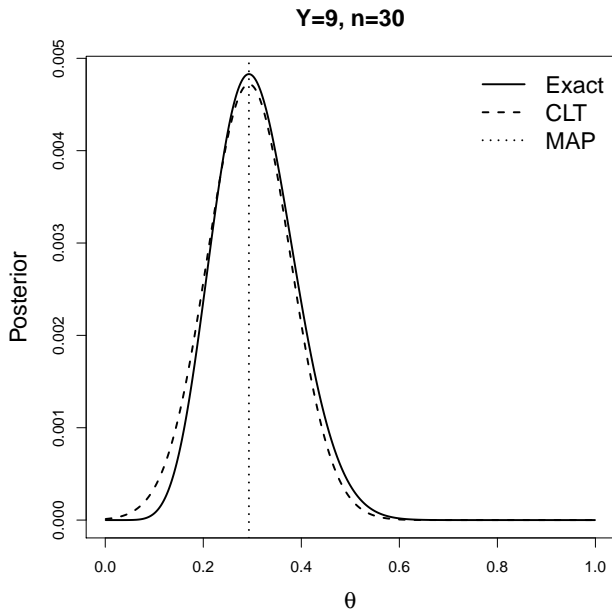
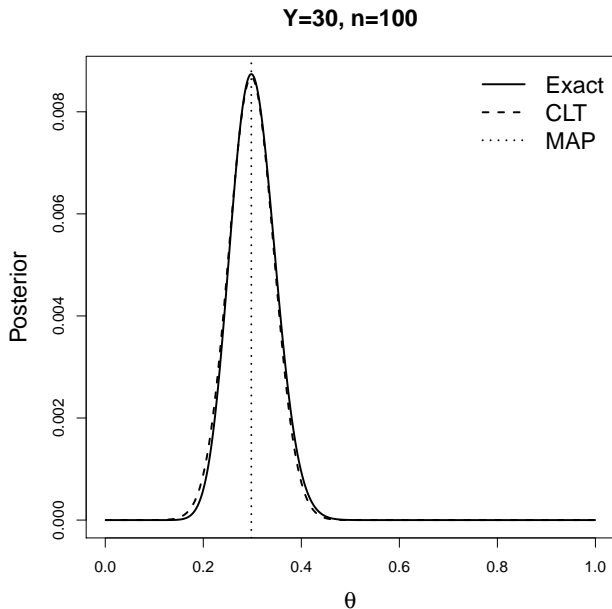


Illustration of the Bayesian CLT



Bayesian central limit theorem

- ▶ For large datasets with a small number of parameters evoking the Bayes CLT is probably the best approach
- ▶ The approximate posterior can be computing using standard software (e.g., `glm` in R)
- ▶ The numerical values (e.g., intervals) will equal the frequentist values, but the interpretation remains Bayesian
- ▶ Why not just do a frequentist analysis? Well, why not just do a Bayesian analysis?

Numerical integration

- ▶ Many posterior summaries of interest are integrals over the posterior
- ▶ Ex: $E(\theta_j|\mathbf{Y}) = \int \theta_j p(\theta) d\theta$
- ▶ Ex: $V(\theta_j|\mathbf{Y}) = \int [\theta_j - E(\theta|\mathbf{Y})]^2 p(\theta) d\theta$
- ▶ These are p dimensional integrals that we usually can't solve analytically
- ▶ A grid approximation is a crude approach
- ▶ Gaussian quadrature is better

Numerical integration

- ▶ Numerical integration is only feasible for small p
- ▶ The Iteratively Nested Laplace Approximation (INLA) is an even more sophisticated method
- ▶ INLA combines Gaussian approximations with numerical integration
- ▶ This works well if most of the parameters are approximately normal and only a few are non-Gaussian and require numerical integration

Monte Carlo sampling

- ▶ Monte Carlo (MC) sampling is the predominant method of Bayesian inference because it can be used for high-dimensional models (i.e., with many parameters)
- ▶ The main idea is to approximate posterior summaries by drawing samples from the posterior distribution, and then using these samples to approximate posterior summaries of interest
- ▶ This requires drawing samples from non-standard distributions
- ▶ It also requires careful analysis to be sure the approximation is sufficiently accurate

Monte Carlo sampling

- ▶ Notation: Let $\theta = (\theta_1, \dots, \theta_p)$ be the collection of all parameters in the model
- ▶ Notation: Let $\mathbf{Y} = (Y_1, \dots, Y_n)$ be entire dataset
- ▶ The posterior $f(\theta|\mathbf{Y})$ is a distribution
- ▶ If $\theta^{(1)}, \dots, \theta^{(S)}$ are samples from $f(\theta|\mathbf{Y})$, then the mean of the S samples approximates the posterior mean
- ▶ This only provides approximations of the posterior summaries of interest.
- ▶ But how to draw samples from some arbitrary distribution $p(\theta|\mathbf{Y})$?

Software options

- ▶ There are now many software options for performing MC sampling
- ▶ There are SAS procs and R functions for particular analyses (e.g., the function `BLR` for linear regression)
- ▶ There are also all-purpose programs that work for virtually any user-specified model: OpenBUGS; JAGS; Proc MCMC; STAN; INLA (not MC)
- ▶ We will use JAGS, but they are all similar

MCMC

We will study the algorithms behind these programs, which is important because it helps:

- ▶ Select models and priors conducive to MC sampling
- ▶ Anticipate bottlenecks
- ▶ Understand error messages and output
- ▶ Design your own sampler if these off-the-shelf programs are too slow

The most common algorithms are **Gibbs** and **Metropolis** sampling

Gibbs sampling

- ▶ Gibbs sampling is attractive because it can sample from high-dimensional posteriors
- ▶ The main idea is to break the problem of sampling from the high-dimensional joint distribution into a series of samples from low-dimensional conditional distributions
- ▶ Updates can also be done in blocks (groups of parameters)
- ▶ Because the low-dimensional updates are done in a loop, samples are not independent
- ▶ The dependence turns out to be a Markov distribution, leading to the name Markov chain Monte Carlo (MCMC)

MCMC for the Bayesian t test

- ▶ Say $Y_i \sim \text{Normal}(\mu, \sigma^2)$ with $\mu \sim \text{Normal}(0, \sigma_0^2)$ and $\sigma^2 \sim \text{InvGamma}(a, b)$
- ▶ In the one-parameter section we saw that if we knew either μ or σ^2 , we can sample from the other parameter
- ▶ $\mu | \sigma^2, \mathbf{Y} \sim \text{Normal} \left[\frac{n\bar{Y}\sigma^{-2} + \mu_0\sigma_0^{-2}}{n\sigma^{-2} + \sigma_0^{-2}}, \frac{1}{n\sigma^{-2} + \sigma_0^{-2}} \right]$
- ▶ $\sigma^2 | \mu, \mathbf{Y} \sim \text{InvGamma} \left[\frac{n}{2} + a, \frac{1}{2} \sum_{i=1}^n (Y_i - \mu)^2 + b \right]$
- ▶ But how to draw from the joint distribution?

Gibbs sampling for the Gaussian model

- ▶ The full conditional (FC) distribution is the distribution of one parameter taking all other as fixed and known

- ▶ FC1: $\mu|\sigma^2, \mathbf{Y} \sim \text{Normal} \left[\frac{n\bar{Y}\sigma^{-2} + \mu_0\sigma_0^{-2}}{n\sigma^{-2} + \sigma_0^{-2}}, \frac{1}{n\sigma^{-2} + \sigma_0^{-2}} \right]$

- ▶ FC2: $\sigma^2|\mu, \mathbf{Y} \sim \text{InvGamma} \left[\frac{n}{2} + a, \frac{1}{2} \sum_{i=1}^n (Y_i - \mu)^2 + b \right]$

Gibbs sampling

- ▶ In the Gaussian model $\theta = (\mu, \sigma^2)$ so $\theta_1 = \mu$ and $\theta_2 = \sigma^2$
- ▶ The algorithm begins by setting initial values for all parameters, $\theta^{(0)} = (\theta_1^{(0)}, \dots, \theta_p^{(0)})$.
- ▶ Variables are then sampled one at a time from their full conditional distributions,

$$p(\theta_j | \theta_1, \dots, \theta_{j-1}, \theta_{j+1}, \dots, \theta_p, \mathbf{Y})$$

- ▶ Rather than 1 p -dimensional joint sample, we make p 1-dimensional samples.
- ▶ The process is repeated until the required number of samples have been generated.

Gibbs sampling

A Set initial value $\theta^{(0)} = (\theta_1^{(0)}, \dots, \theta_p^{(0)})$

B For iteration t ,

FC1 Draw $\theta_1^{(t)} | \theta_2^{(t-1)}, \dots, \theta_p^{(t-1)}, \mathbf{Y}$

FC2 Draw $\theta_2^{(t)} | \theta_1^{(t)}, \theta_3^{(t-1)}, \dots, \theta_p^{(t-1)}, \mathbf{Y}$

...

FCp Draw $\theta_p^{(t)} | \theta_1^{(t)}, \dots, \theta_{p-1}^{(t)}, \mathbf{Y}$

We repeat step B S times giving posterior draws

$$\theta^{(1)}, \dots, \theta^{(S)}$$

Why does this work?

- ▶ $\theta^{(0)}$ isn't a sample from the posterior, it is an arbitrarily chosen initial value
- ▶ $\theta^{(1)}$ likely isn't from the posterior either. Its distribution depends on $\theta^{(0)}$
- ▶ $\theta^{(2)}$ likely isn't from the posterior either. Its distribution depends on $\theta^{(0)}$ and $\theta^{(1)}$
- ▶ **Theorem:** For any initial values, the chain will eventually converge to the posterior
- ▶ **Theorem:** If $\theta^{(s)}$ is a sample from the posterior, then $\theta^{(s+1)}$ is too

Convergence

- ▶ We need to decide:
 1. When has it converged?
 2. When have we taken enough samples to approximate the posterior?
- ▶ Once we decide the chain has converged at iteration T , we discard the first T samples as “burn-in”
- ▶ We use the remaining $S - T$ to approximate the posterior
- ▶ For example, the posterior mean (marginal over all other parameters) of θ_j is

$$E(\theta_j | \mathbf{Y}) \approx \frac{1}{S - T} \sum_{s=T+1}^S \theta_j^{(s)}$$

Examples

- ▶ `http:`
`//www4.stat.ncsu.edu/~reich/ABA/code/NN2`
- ▶ `http:`
`//www4.stat.ncsu.edu/~reich/ABA/code/SLR`
- ▶ `http:`
`//www4.stat.ncsu.edu/~reich/ABA/code/ttest`
- ▶ All derivations of full conditionals are in the online derivations

Practice problem

Work out the full conditionals for λ and b for the following model:

$$Y|\lambda, b \sim \text{Poisson}(\lambda)$$

$$\lambda|b \sim \text{Gamma}(1, b)$$

$$b \sim \text{Gamma}(1, 1)$$

Metropolis sampling

- ▶ In Gibbs sampling each parameter is updated by sampling from its full conditional distribution
- ▶ This is possible with conjugate priors
- ▶ However, if the prior is not conjugate it is not obvious how to make a draw from the full conditional
- ▶ For example, if $Y \sim \text{Normal}(\mu, 1)$ and $\mu \sim \text{Beta}(a, b)$ then

$$p(\mu|Y) \propto \exp\left[-\frac{1}{2}(Y - \mu)^2\right] \mu^{(a-1)}(1 - \mu)^{b-1}$$

- ▶ For some likelihoods there is no known conjugate prior, e.g., logistic regression
- ▶ In these cases we use Metropolis sampling

Metropolis sampling

- ▶ Metropolis sampling is a version of rejection sampling
- ▶ Let θ_j^* be the current value of the parameter being updated and $\theta_{(j)}$ be the current value of all other parameters
- ▶ You propose a random candidate based on the current value, e.g.,

$$\theta_j^c \sim \text{Normal}(\theta_j^*, s_j^2)$$

- ▶ The candidate is accepted with probability

$$R = \min \left\{ 1, \frac{p(\theta_j^c | \theta_{(j)}, \mathbf{Y})}{p(\theta_j^* | \theta_{(j)}, \mathbf{Y})} \right\}$$

- ▶ If the candidate is not accepted then you simply retain the previous value and move to the next step

Metropolis sampling

- ▶ The candidate standard deviation s_j is a tuning parameter
- ▶ Ideally s_j is tuned to give acceptance probability around 0.3-0.4
- ▶ If s_j is too small:
- ▶ If s_j is too large:
- ▶ Off-the-shelf programs have default values, and many allow you to change the value if the results are unsatisfactory

Metropolis-Hastings sampling

- ▶ Denote $\theta_j^c \sim q(\theta|\theta^*)$ as the candidate distribution
- ▶ The candidate distribution is symmetric if

$$q(\theta^*|\theta_j^c) = q(\theta_j^c|\theta^*)$$

- ▶ For example, if $\theta_j^c \sim \text{Normal}(\theta_j^*, s_j^2)$ then

$$q(\theta_j^c|\theta^*) = \frac{1}{\sqrt{2\pi}s_j} \exp \left[-\frac{(\theta_j^c - \theta_j^*)^2}{2s_j^2} \right] = q(\theta^*|\theta_j^c).$$

Metropolis-Hastings sampling

- ▶ Metropolis-Hastings (MH) sampling generalizes Metropolis sampling to allow for asymmetric candidate distributions
- ▶ For example, if $\theta_j \in [0, 1]$ then a reasonable candidate is

$$\theta_j^c | \theta_j^* \sim \text{Beta}[10\theta_j^*, 10(1 - \theta_j^*)]$$

- ▶ Then $q(\theta_j^* | \theta_j^c)$ and $q(\theta_j^c | \theta_j^*)$ are both beta PDFs
- ▶ MH proceeds exactly like Metropolis except the acceptance probability is

$$R = \min \left\{ 1, \frac{p(\theta_j^c | \theta_{(j)}, \mathbf{Y}) q(\theta_j^* | \theta_j^c)}{p(\theta_j^* | \theta_{(j)}, \mathbf{Y}) q(\theta_j^c | \theta_j^*)} \right\}$$

Metropolis-Hastings sampling

- ▶ What if we take the candidate distribution to be the full conditional distribution

$$\theta_j^c \sim p(\theta_j^c | \theta_{(j)}, \mathbf{Y})$$

- ▶ What is the acceptance ratio?
- ▶ What does this say about the relationship between Gibbs and Metropolis Hastings sampling?

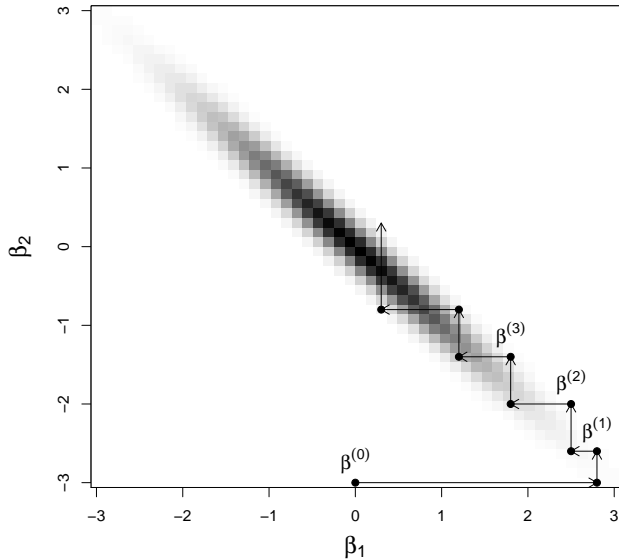
Variants

- ▶ You can combine Gibbs and Metropolis in the obvious way, sampling directly from full conditional when possible and Metropolis otherwise
- ▶ Adaptive MCMC varies the candidate distribution throughout the chain
- ▶ Hamiltonian MCMC uses the gradient of the posterior in the candidate distribution and is used in STAN

Blocked Gibbs/Metropolis

- ▶ If a group of parameters are highly correlated convergence can be slow
- ▶ One way to improve Gibbs sampling is a block update
- ▶ For example, in linear regression might iterate between sampling the block $(\beta_1, \dots, \beta_p)$ and σ^2
- ▶ Blocked Metropolis is possible too
- ▶ For example, the candidate for $(\beta_1, \dots, \beta_p)$ could be a multivariate normal

Posterior correlation leads to slow convergence



Summary

- ▶ With the combination of Gibbs and Metropolis-Hastings sampling we can fit virtually any model
- ▶ In some cases Bayesian computing is actually preferable to maximum likelihood analysis
- ▶ In most cases Bayesian computing is slower
- ▶ However, in the opinion of many it is worth the wait for improved uncertainty quantification and interpretability
- ▶ In all cases it is important to carefully monitor convergence

Options for coding MCMC

- ▶ Writing your own code
- ▶ Bayesian options in SAS procedures
- ▶ R packages for specific models
- ▶ All-purpose software like JAGS, BUGS, PROC MCMC, and STAN

Bayes in SAS procedures and R functions

- ▶ Here is a SAS proc

```
proc phreg data=VALung;  
  class PTherapy(ref='no') Cell(ref='large')  
  Therapy(ref='standard');  
  model Time*Status(0) = KPS Duration;  
  bayes seed=1 outpost=cout coeffprior=uniform  
  plots=density;  
run;
```

- ▶ In R you can use `BLR` for linear regression, `MCMClogit` for logistic regression, etc.

Why Just Another Gibbs Sampler (JAGS)?

- ▶ You can fit virtually any model
- ▶ You can call JAGS from \mathbb{R} which allows for plotting and data manipulation in \mathbb{R}
- ▶ It runs on all platforms: LINUX, Mac, Windows
- ▶ There is a lot of help online
- ▶ \mathbb{R} has many built in packages for convergence diagnostics

How does JAGS work?

- ▶ You specify the model by declaring the likelihood and priors
- ▶ JAGS then sets up the MCMC sampler, e.g., works out the full conditional distributions for all parameters
- ▶ It returns MCMC samples in a matrix or array
- ▶ It also automatically produces posterior summaries like means, credible sets, and convergence diagnostics
- ▶ **User's manual:** http://blue.for.msu.edu/CSTAT_13/jags_user_manual.pdf

Running JAGS from R has the following steps

1. Install JAGS: `https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Windows/`
2. Download `rjags` from CRAN and load the library
3. Specify the model as a string
4. Compile the model using the function `jags.model`
5. Draw burn-in samples using the function `update`
6. Draw posterior samples using the function `coda.samples`
7. Inspect the results using the `plot` and `summary` functions

Examples

- ▶ The course website has many example of Bayesian analyses using JAGS
- ▶ There are also comparisons with other software
- ▶ For moderately-sized problems JAGS is competitive with these methods
- ▶ For really big and/or complex analyses STAN is preferred
- ▶ JAGS is easier to code and so we will use it through the course, but you should be familiar with other software
- ▶ Once you understand JAGS, switching to the others is straightforward

Tuning the MCMC algorithm

- ▶ MCMC is beautiful because it can handle virtually any statistical model and it is usually pretty easy to write functional code
- ▶ However, for hard problems great care must be taken to ensure that the algorithm has converged
- ▶ There are three main decisions:
 - ▶ Selecting the initial values
 - ▶ Determining if/when the chain(s) has converged
 - ▶ Selecting the number of samples needed to approximate the posterior

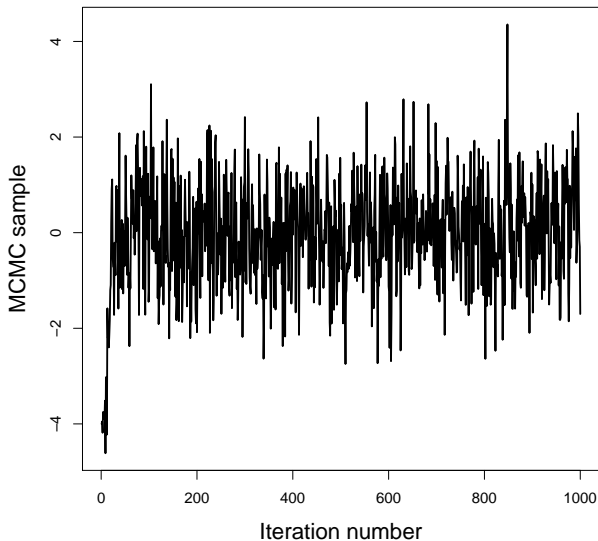
Initial values

- ▶ The algorithm will eventually converge no matter what initial values you select
- ▶ However taking time to select good initial values will speed up convergence
- ▶ It is important to try a few initial values to verify they all give the same result
- ▶ Usually 3-5 separate chains is sufficient
- ▶ **Option 1:** Select good initial values using method of moments or MLE
- ▶ **Option 2:** Purposely pick bad but different initial values for each chain to check convergence

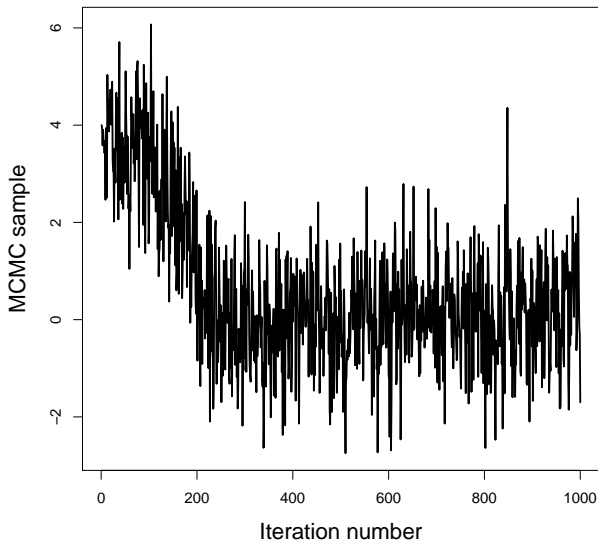
Convergence

- ▶ The first few samples are probably not draws from the posterior distribution
- ▶ It can take a few dozen, hundreds or even thousands of iterations to move from the initial values to the posterior
- ▶ When the sampler reaches the posterior this is called convergence
- ▶ Samples before convergence are discarded as **burn-in**
- ▶ After convergence the samples should not converge to a single point!
- ▶ They should be draws from the posterior, and ideally look like a caterpillar or bar code

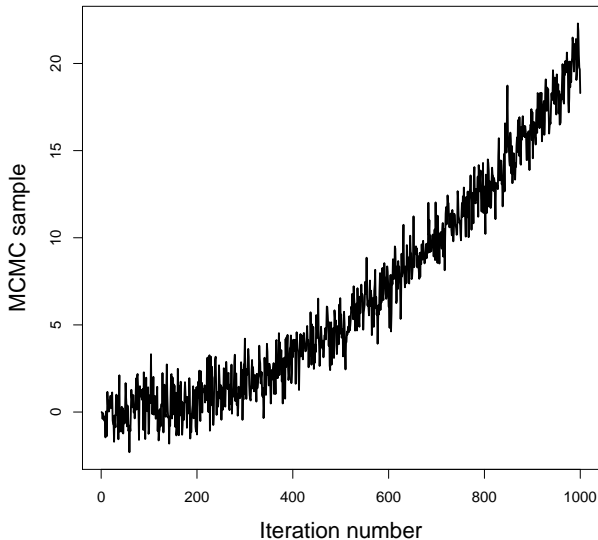
Convergence in a few iterations



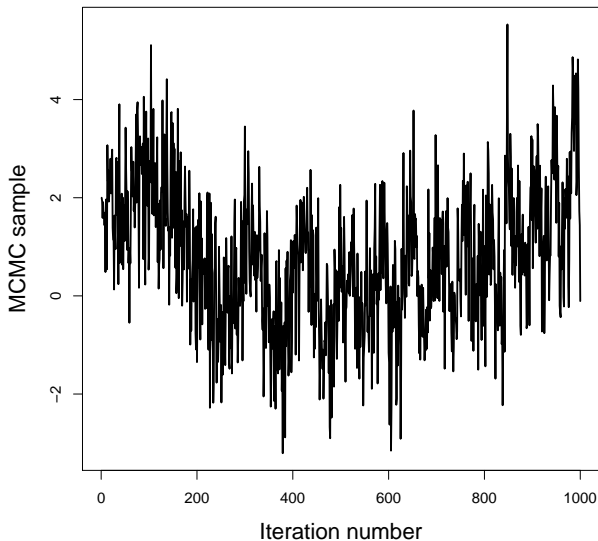
Convergence in a few hundred iterations



This one never converged



Convergence is questionable



Convergence diagnostics

- ▶ So far we have visually inspected the chains for convergence
- ▶ There are many formal diagnostics
- ▶ The CODA package in R has dozens of diagnostics
- ▶ Most give a measure of convergence for each parameter
- ▶ Checking convergence using these one-number summaries is more efficient and objective than visual inspection

Convergence diagnostics

- ▶ Did my chains converge?
 - ▶ Geweke
 - ▶ Gelman-Rubin
- ▶ Did I run the sampler long enough after convergence?
 - ▶ Effective sample size
 - ▶ Standard errors for the posterior mean estimate

Examples

- ▶ The JAGS function `coda.samples` returns sample is the format that can be passed to the `CODA` function which actually computes the diagnostics
- ▶ The course website uses CODA to assess convergence for a best-case and a worst-case scenario

Geweke diagnostic

- ▶ Compares the mean in the beginning of the chain with the mean at the end of the chain
- ▶ Can be used for a single chain
- ▶ Done separately for each parameter
- ▶ The JAGS default is to compare the first 10% with the last 50%
- ▶ The test accounts for autocorrelation
- ▶ The test statistic is a z-score, so $|Z| > 2$ indicates poor convergence

Gelman-Rubin statistic

- ▶ If we run multiple chains, we hope that all chains give same result
- ▶ The Gelman-Rubin statistics measures agreement between chains
- ▶ Is it essentially an ANOVA test of whether the chains have the same mean
- ▶ It is scaled so that 1 is perfect and 1.1 is decent but not great convergence
- ▶ JAGS plots the statistic over iteration
- ▶ When the statistic reaches one this indicates convergence

Autocorrelation

- ▶ Ideally the samples would be independent across iteration
- ▶ The autocorrelation function $\rho(h)$ is the correlation between samples h iterations apart
- ▶ JAGS plots the autocorrelation as a function of h
- ▶ Lower values are better, but if the chains are long enough even large values can be OK
- ▶ **Thinning:** If autocorrelation is zero after lag h you can thin the samples by h to achieve independence
- ▶ This is always less efficient than using all samples, but can save memory

Effective sample size

- ▶ Highly correlated samples have less information than independent samples
- ▶ Say S is the actual number of MCMC samples
- ▶ The **effective samples size** is

$$ESS = \frac{S}{1 + 2 \sum_{h=1}^{\infty} \rho(h)}$$

- ▶ The correlated MCMC sample of length S has the same information as ESS independent samples
- ▶ ESS should be at least a few thousand for all parameters

Standard errors of posterior mean estimates

- ▶ The sample mean of the MCMC draws is an estimate of the posterior mean
- ▶ The standard error of this estimate as another diagnostic
- ▶ Assuming independence the standard error is

$$\text{Naive SE} = \frac{s}{\sqrt{S}}$$

where s is the sample SD and S is the number of samples

- ▶ A more realistic standard error is

$$\text{Times-series SE} = \frac{s}{\sqrt{ESS}}$$

What to do if the chains haven't converged?

What to do for massive datasets?

- ▶ MAP estimation
- ▶ Bayesian CLT
- ▶ Variation Bayes: Approximates the posterior by assuming the posterior is independent across parameters (fast, but questionable statistical properties)
- ▶ Parallel computing: MCMC is inherently sequential, but often some steps can be done in parallel, e.g., onerous likelihood computations
- ▶ Divide and Conquer: Split the data into batches and analysis them in parallel, and then carefully combine the result of the batch analyses