Markov Chain Monte Carlo

Edps 590BAY

Carolyn J. Anderson

Department of Educational Psychology
ILLINOIS

Fall 2019

Overview

- Introduction to Bayesian computing.
- Markov Chain
- Metropolis algorithm for mean of normal given fixed variance.
- Revisit anorexia data.
- Practice problem with SAT data.
- Some tools for assessing convergence
- Metropolis algorithm for mean and variance of normal.
- Anorexia data.
- Practice problem.
- Summary

Depending on the book that you select for this course, read either Gelman et al. pp 2751-291 or Kruschke Chapters pp 143–218. I am relying more on Gelman et al.

Introduction to Bayesian Computing

- Our major goal is to approximate the posterior distributions of unknown parameters and use them to estimate parameters.
- The analytic computations are fine for simple problems,
  - Beta-binomial for bounded counts
  - Normal-normal for continuous variables
  - Gamma-Poisson for (unbounded) counts
  - Dirichelt-Multinomial for multicategory variables (i.e., a categorical variable)
  - Models in the exponential family with small number of parameters
- For large number of parameters and more complex models
  - Algebra of analytic solution becomes overwhelming
  - Grid takes too much time.
  - Too difficult for most applications.

Steps in Modeling

Recall that the steps in an analysis:

1. Choose model for data (i.e., $p(y|\theta)$) and model for parameters (i.e., $p(\theta)$ and $p(\theta|y)$).
2. Compute $p(\theta|y)$ or at least find a good approximation of it.
3. Model evaluation.

Target Distribution: $p(\theta|y)$

The distribution we want to simulate is the posterior, $p(\theta|y)$.

Let

- $q(\theta|y)$ be an un-normalized density that is easy to compute.
- $p(\theta|y)$ the target distribution
- The ratio

$$\frac{q(\theta|y)}{p(\theta|y)} = \text{ a constant that depends only y}$$

- We will work with

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

Avoiding Under and Over Flow Errors

- Numbers can become too small (underflow) or too large (overflow) for computers to deal with. This leads to errors.
- Simple example: Note that logically $\exp(\log(1000)) = \log(\exp(1000)) = 1000$; however, if you try them in R ...

$$\exp(\log(1000)) = 1000 \qquad \text{but} \qquad \log(\exp(1000)) = \text{Inf}$$

- By working with log densities we can work with densities we only exponentiate at the very end (if even necessary).
- For example, the normal distribution,

$$\begin{aligned} p(y_1, \ldots, y_n|\theta, \sigma^2) &= (2\pi\sigma^2)^{-n/2} \prod_{i=1}^{n} \exp\left\{ -\frac{1}{2}\left(\frac{y_i - \theta}{\sigma}\right)^2 \right\} \\ \log p(y_1, \ldots, y_n|\theta, \sigma^2) &= (-n/2)\log(2\pi\sigma^2) + \sum_i \left\{ -\frac{1}{2}\left(\frac{y_i - \theta}{\sigma}\right)^2 \right] \end{aligned}$$

Markov Chains

(quotes from Gelman et al.):

"…a Markov Chain is a sequence of of random variables $\theta^1, \theta^2, \ldots,$ for which, for any t, the distribution of θ^t given all previous θ's depends only on the most recent value."

"The key to the method's success, however, is not the Markov property but rather that the approximate distributions are improved at each step in the simulation, in the sense of converging to the target distribution."

"The transition probability distributions must be constructed to converge so that the Markov chain converges to a unique stationary distribution that is the posterior distribution, $p(\theta|y)$.

Example of Markov Chain

Run R function `metroLogNorm(niter,y,current,tau,jump)` where

- niter = number of iterations
- y a random sample from a normal distribution
- current = starting value for algorithm
- tau = a guess at the variance of posterior for the mean
- jump = a value that is a standard deviation of “jumping” distribution

Output:

- the values for each iteration
- figure of values by iteration (shows the random walk)
- histogram of values

If you use RStudio, you have to open up a plot window to “see” it in action.

Overview of Stochastic Methods & Algorithms

Methods

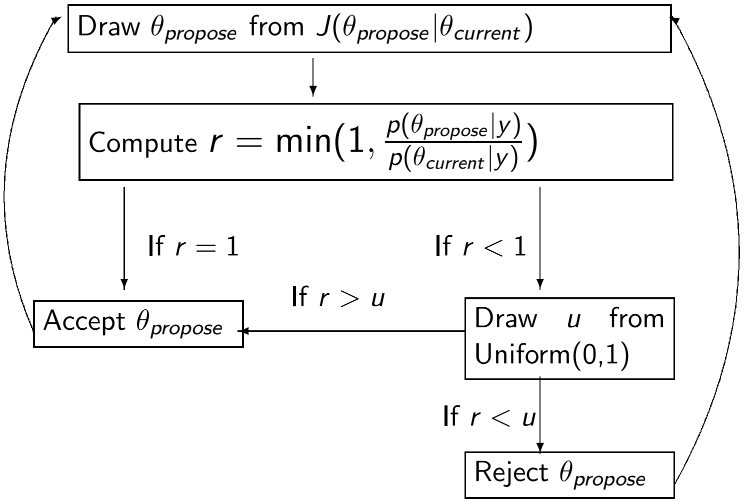
- Metropolis algorithm
- Gibbs Sampling
- Hamiltonian

Some Implementations in R

- Programming in base R
- jags
- rstan

Many useful tools are in the "coda" package and they save time when assessing convergence.

Metropolis Algorithm


Draw $\theta_{propose}$ from $J(\theta_{propose}|\theta_{current})$
Compute $r = \min(1, \frac{p(\theta_{propose}|y)}{p(\theta_{current}|y)})$
If $r = 1$
If $r < 1$
If $r > u$
Accept $\theta_{propose}$
Draw u from Uniform(0,1)
If $r < u$
Reject $\theta_{propose}$

Jumping Distribution

The jumping or "proposal" distribution

- Must be symmetric; that is, $J(\theta_a|\theta_b) = J(\theta_b|\theta_a)$.
- The standard deviation of the jumping distribution impacts how long it take chain to get to stationary point.
- The algorithm is fine for low dimensional problems (i.e., small number of parameters).
- Uses the parameter estimated from the previous iteration. For example, use θ_{t-1} as the mean of the jumping distribution during the t^{th} iteration. In other words, its Markov process.
- Need to ensure that values θ are sampled all over where possible values could be.

$p(\theta_t|y)$ and Accept or Reject Proposed Value

Using Bayes Theorem:

$$\begin{aligned} r &= min\left(1, \frac{p(\theta_t|y)}{p(\theta_{t-1}|y)}\right) \\ \frac{p(\theta_t|y)}{p(\theta_{t-1}|y)} &= \frac{p(y|\theta_t)p(\theta_t)/p(y)}{p(y|\theta_{t-1})p(\theta_{t-1})/p(y)} = \frac{p(y|\theta_t)p(\theta_t)}{p(y|\theta_{t-1})p(\theta_{t-1})} \end{aligned}$$

- Need to choose prior and likelihood.
- If θ_t is better than θ_{t-1}, then $r \geq 1$.
- If θ_t is not as good at θ_{t-1}, we may still accept θ_t.
- Let $u \sim \text{Uniform}(0, 1)$, accept θ_t if $r \geq u$.
- If $r < u$, then reject θ_t and set $\theta_t - \theta_{t-1}$.

Accept or Reject Proposed Value (another way)

- If $p(\theta_t|y) \geq p(\theta_{t-1}|y)$ the proposed values θ_t is "accepted"; that is,
$$\frac{p(\theta_t|y)}{p(\theta_{t-1}|y)} \geq 1$$
- If $p(\theta_t|y) < p(\theta_{t-1}|y)$ the proposed values θ_t may still be accepted with some probability that depends on the ratio and a draw from the uniform distribution, $u \sim \text{Uniform}(0,1)$; that is,
$$\begin{array}{lcl} \text{If } r > u & \rightarrow & \text{accept proposed } \theta \text{ and } \theta_t = \text{ proposed value} \\ \text{If } r < u & \rightarrow & \text{reject proposed } \theta \text{ and } \theta_t = \theta_{t-1} \end{array}$$
- Repeat many, many times and resulting θ_1,θ_2,. . .,$\theta_{\text{large numer}}$ is an approximation of the posterior density.

Metropolis Algorithm

- Once you have $\theta_1, \theta_2, \ldots, \theta_{\text{large numer}}$ you can
  - Graph in various ways
  - Compute statistics: mean, median, mode, standard deviation, intervals, etc.
  - Compute functions of statistics.
  - Can easily do integration via summation; that is,

$$h(\theta) = \int_{-\infty}^{\infty} h(\theta)p(\theta)d(\theta) \approx \frac{1}{S}\sum_{i=1}^{S} h(\theta_s)$$

- BUT
  - Works fine for small problems (e.g., small number of parameters).
  - Needs some tuning for optimal efficiency.

Anorexia Data

Sample statistics:

$$\bar{y} = 2.7638, \qquad s^2 = 63.7378, \qquad n = 72$$

Analytic results where $\mu_o = 0$, $\tau_o^2 = 1000$, and $\sigma = 0$

$$\theta_{57} = \mu_{57} = 2.683, \qquad \tau_{57}^2 = 1.123$$

Analytic results with extra 15 and $\kappa_0 = 1$

$$\theta_n = 2.761, \qquad \tau_n^2 = 0.889$$

Metropolis algorithm setting $\sigma = \text{sd}(\textit{data})$, start=0, $\tau_o^2 = 0.1$, jump standard deviation $= 0.3$, and 2,000 iterations:

$$\theta = 2.7684, \qquad \text{sd}(\theta) = 0.1026$$

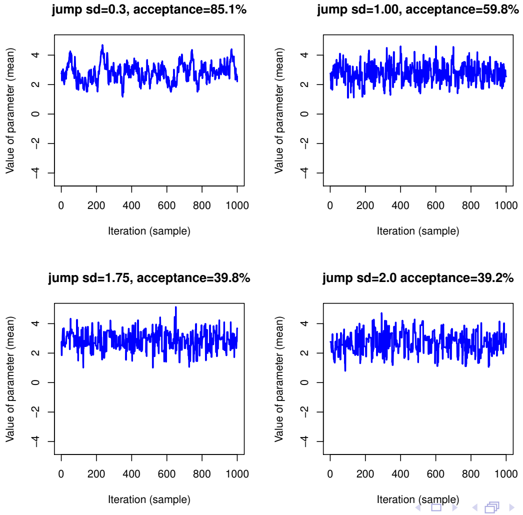
Tuning the Algorithm for Anorexia Data

Tuning was very important with these data.

I different inputs:

- I tried different starting values for the mean to ensure that chains were stable.
- I wanted and acceptance rate of about 45% (for more complex aim for 23%, see Gelman for justification).
- I tried different jump standard deviations to get one that works well –acceptance rate $\approx$ 40%..
- With good input, need fewer iterations.
- I settled on jump standard deviation $= 1.75$ and 5,000 iterations (more than we need?).

Tuning Picture

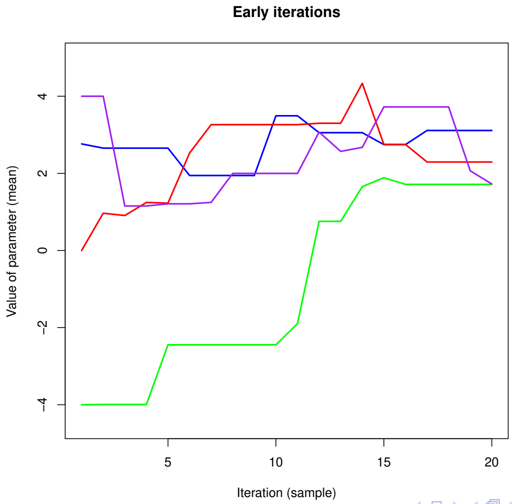


Results for Anorexia Data

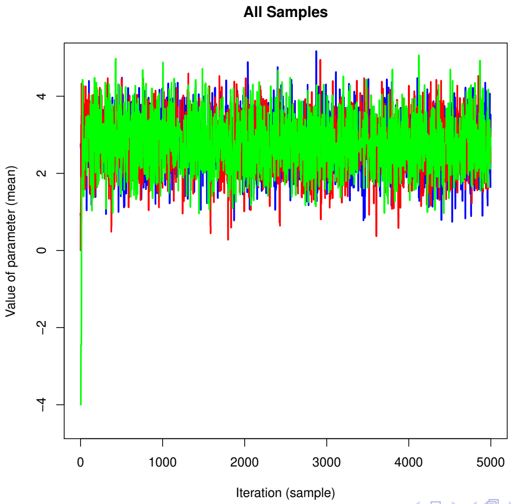
- 4 chains with starting values equal to $\bar{y}$, 0, -4, 4
- $\tau = \sqrt{s^2/n}$ = standard error of mean
- jump std $= 1.75$
- 5000 iterations and saved 4000×4 for posterior.

	acceptance	Sample Statistics of Posterior			
	rate	mean	median	25%	95%
chain1	41.7%	2.762	2.760	2.306	3.214
chain2	40.7%	2.740	2.744	2.296	3.21
chain3	40.7%	2.788	2.782	2.322	3.25
chain4	41.3%	2.754	2.782	2.322	3.195
grand		2.754	2.760	2.305	3.198

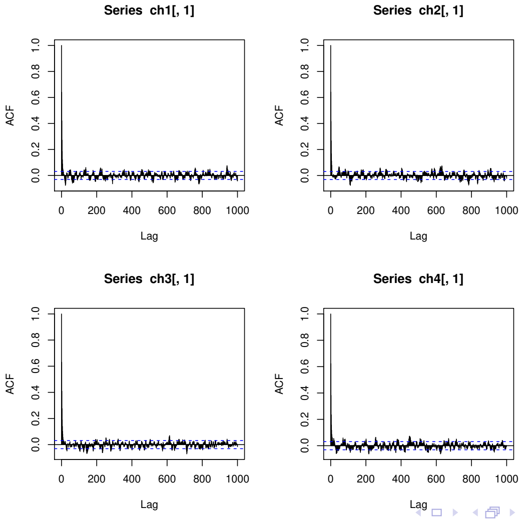
Anorexia Data: Early iterations of 4 chains



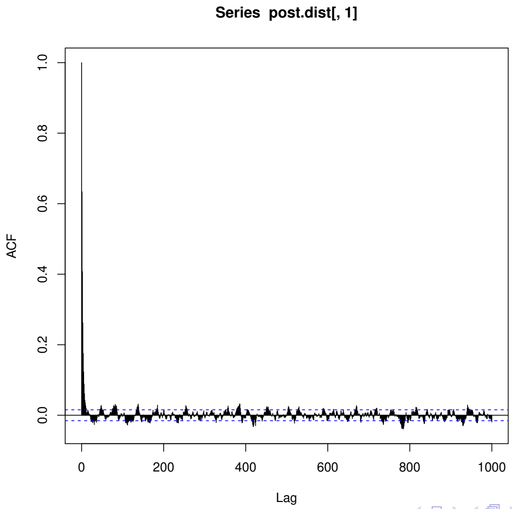
Anorexia Data: Trace Plots of 4 chains



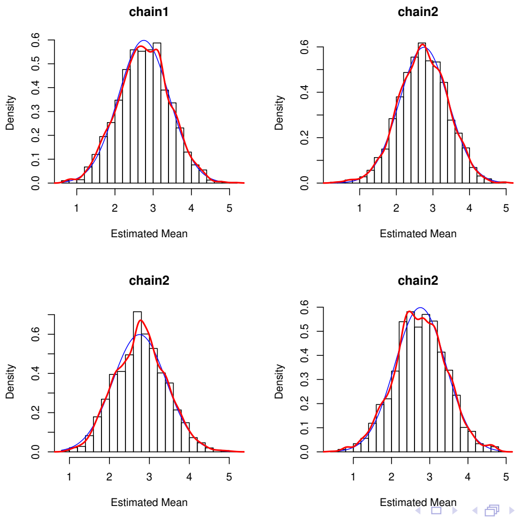
Anorexia Data: Auto-Correlations



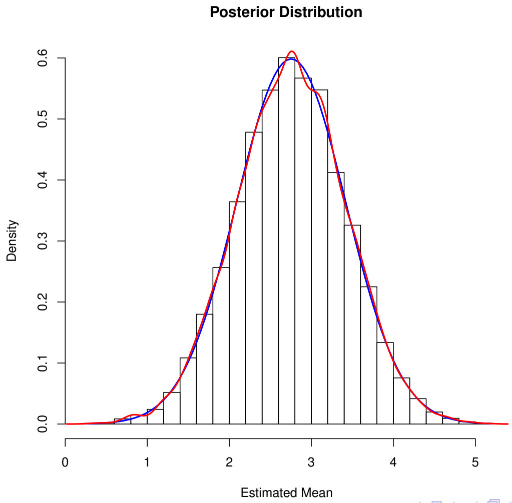
Anorexia Data: Auto-Correlations for Posterior



Anorexia Data: Density Estimates



Anorexia Data: Posterior Distribution & Density



Getting What you Paid for

Use the metroLogNorm function to estimate the mean for fixed variance:

`metroLogNorm(niter,y,current,tau,jump)` where

- niter = number of iterations
- y is data (i.e., it needs to be named "y")
- current = prior value of mean
- tau = prior variance of the mean
- jump = a value that is a standard deviation of "jumping" distribution

Output:

- the values for each iteration
- figure of values by iteration (shows the random walk)
- histogram of values

If you use RStudio, you have to open up a plot window to "see" it in action.

Assessing Convergence using CODA

- Trace Plots: parameter value $\times$ iteration. These are useful for assessing whether chain have stabilized and see where could set "burn in" or "warm up".
- Geweke Statistic: Test whether the mean of first part of chain (first 10% of θs) equals the mean of the later part of the chain (last 50%). This is based on the assumption that the first and last parts of the chain are (asymptotically) independent, such that difference between the means should be 0. The statistics is $N(0, 1)$.
- Auto-correlations: plot auto-correlations $\times$ iterations. These show the dependencies between candidate θs in a chain, at convergence they should be 0.

Assessing Convergence (continued)

- Effective Sample Size: Even if you have a large number of values in the simulated posterior distribution, due to the dependency between θs we need a correction to the sample size:
$$\text{ESS} = \frac{mn}{1 + 2\sum_t \text{ACF}_t},$$
where ACF_t is the autocorrelation of sequence at lag t, m is 2 times number of chains and n is length of chain.
- Trace plots of multiple chains: Determine whether the chains are mixing well or there is an "orphan". Will see wether starting values have impact on results.

Assessing Convergence (continued)

- Gelman-Rubin diagnostic or the "potential scale reductions" or the "shrink factor". The between chain variance relative to the within chain variance should be about the same if all chains have settled. A value > 1.1 is "cause for concern".
  - After deleting warm-ups, split each of the chains into 2, let $n =$ length of split chain, and $m =$ number of split chains.
  - Compute $\bar{\theta}_{.j} = \frac{1}{n}\sum_{i=1}^{n}\theta_{ij}$, $\quad \bar{\theta}_{..} = (1/m)\sum_{j=1}^{m}\bar{\theta}_{.j}$,

$$\begin{aligned} B &= \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\theta}_{.j}-\bar{\theta}_{..})^2 \\ W &= \frac{1}{m}\sum_{j=1}^{m}s_j^2, \text{ where } s_j^2 = \frac{1}{n-1}(\theta_{ij}-\bar{\theta}_j)^2 \end{aligned}$$

  - $\widehat{\text{var}}(\theta|y)$ is the marginal posterior variance of estimand of θ,

$$\text{PSRF} = \text{Rhat} = \hat{R} = \sqrt{\frac{\frac{n-1}{n}W + \frac{1}{n}B}{W}} = \sqrt{\frac{\widehat{\text{var}}(\theta|y)}{W}}$$

Assessing Convergence (continued)

- Plots of Gelman-Rubin statistics $\times$ iterations.
- Density Estimation: Plot of multiple chains as densities, there should basically be the same.
- High density intervals for multiple chains should all be very similar.
- Descriptive statistics from different simulated distributions for different chains should be similar in value. Note that the Standard Error of Mean can be extended to MCMC using

$$\text{MCSE} = \text{sd}\theta\text{s}/\sqrt{\text{ESS}}$$

Metropolis Algorithm for Two Parameters: μ and σ^2

- Works pretty much the same as when just estimating μ for fixed σ, but it is more transparent in terms of the parts.
- Get the file “metropolis_log_with_examples.tex” from the course web-site (there are example simulations after metroLogNorm2 function).
- The following slides are the results of simulations and R-commands to produce the plots and statistics described in the pervious section on convergence.

How to Use metroNorm2

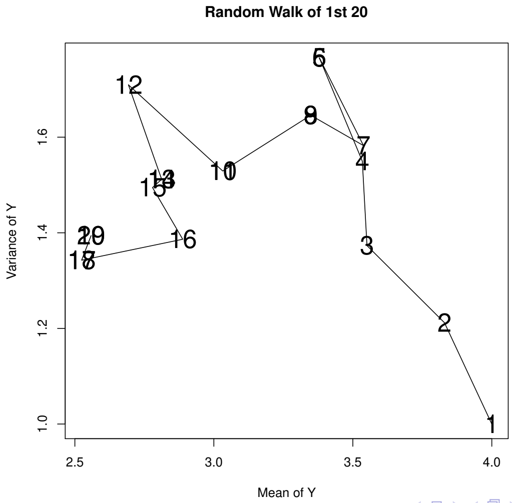
First we need some data:

```
mu = 2
std = 1
N = 20
y <- rnorm(N,mean=mu,sd=std)
```

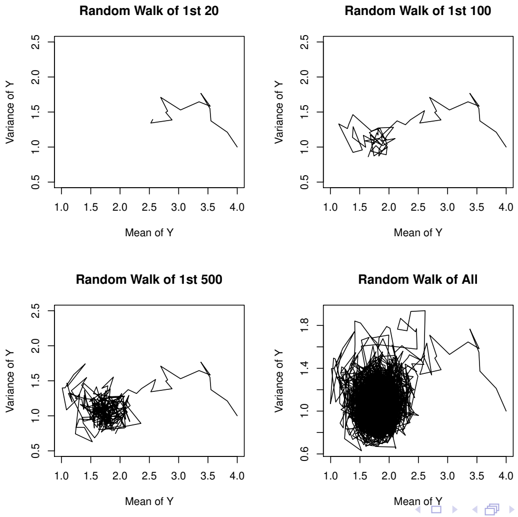
Set the starting values for μ and σ and run the function:

```
start ← c(0.00,1.0)
chain1 ← metroNorm2(start,5000)
```

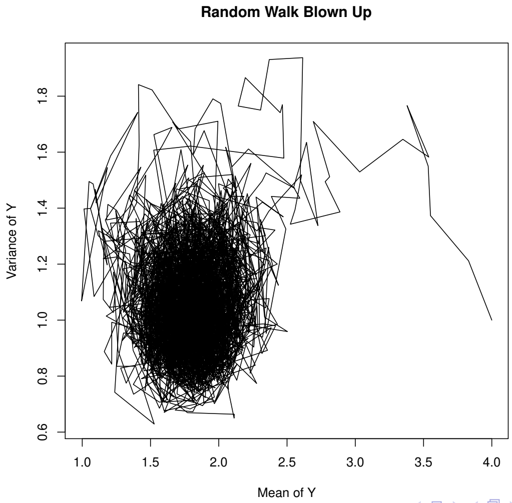
Random Walk Through the Parameter Space



More Random Walks Through the Parameter Space

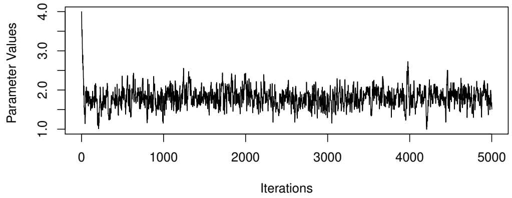


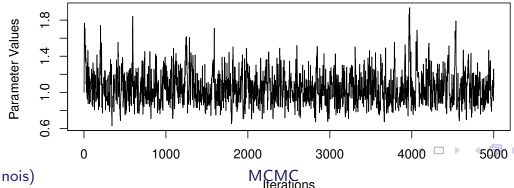
Longest Random Walk Through the Parameter Space



Individual Trace Plots

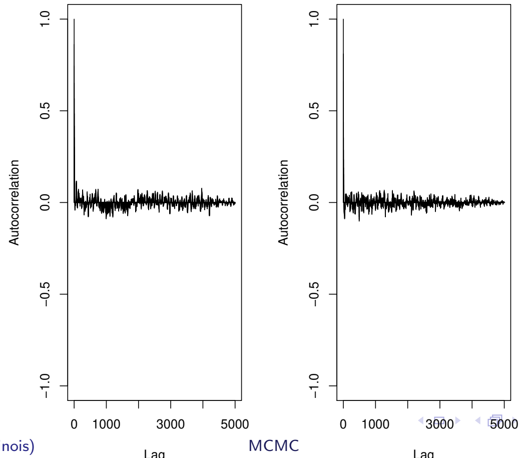
```
traceplot(chain1,smooth=F,type='l',xlab='Iterations',ylab=Parameter Values')
```





Plots of Auto-Correlations

```
autocorr.plot(chain1,iterations,auto.layout=TRUE)
```



Geweke & ESS Statistics

```
geweke.diag(chain1,frac1=0.1,frac2=0.5)
```

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

    var1              var2
-0.5678            1.5574
```

```
effectiveSize(chain1)
```

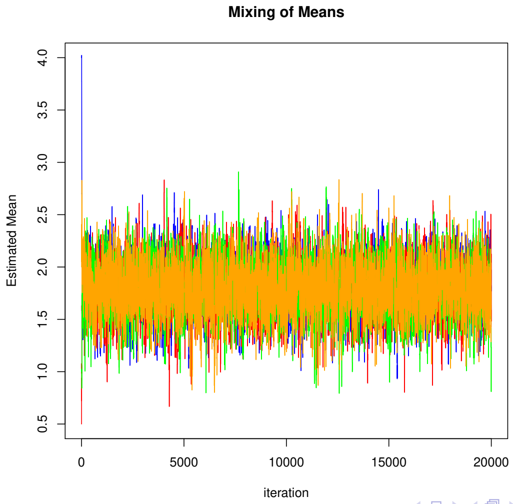
```
     var1      var2
375.7252  524.1868
```

How to Improve Simulation

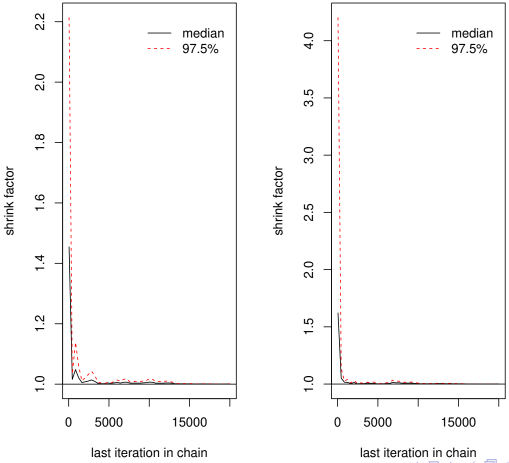
Although this simulation is pretty good, ways that could possibly improve simulation include

- Run the algorithm for more iterations (The first time I ran for 1,000 but increased to 5,000)
- Tune of jump std.
- Use more appropriate $p(y|\theta)$ and $p(\theta)$.
- Thinning.
- Larger sample size (data, more ys...only have $n = 20$)
- Run multiple chains to ensure all parts of the parameter space are visited and that chains are mixing well. Chains should become stable.
- Drop the first 1,000 or first half of the chains before assessing convergence and estimating parameters.

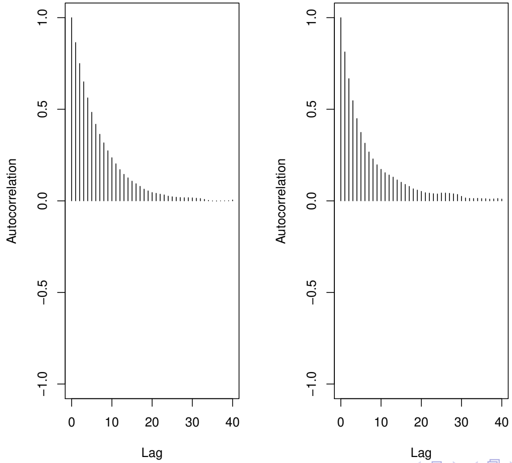
Iterations=20,000, chain=4



Shrink Factor



Autocorrelations (one chain)



Effective Sample Size

```
    var1     var2
5585.972 6880.365
```

summary(all.chains)

Sample statistics:
$\bar{y} = 1.7990,\ s^2 = 0.9953,\ \sqrt{s^2/n} = .2231.$

Bayesian estimates:
Iterations = 1:20001
Thinning interval = 1
Number of chains = 4
Sample size per chain = 20001

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

Mean	SD	Naive SE	Time-series SE
1.797	0.2439	0.0008623	0.003275
1.067	0.1968	0.0006957	0.002440

summary(all.chains)

Sample statistics:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.440	1.135	1.638	1.799	2.209	4.304

Bayesian estimates:

2. Quantiles for each variable:

2.5%	25%	50%	75%	97.5%
1.3163	1.6394	1.797	1.953	2.280
0.7684	0.9322	1.040	1.172	1.511

Getting What you Paid for

- Estimate the mean and variance of state average total SAT scores.
- Run multiple chains
- Combine chains (mcmc objects) use command, for example, all.chains ← mcmc.list(chain1,chain2,chain3,chain4)
- Assess convergence