Metropolis and Metropolis-Hastings

Edps 590BAY

Carolyn J. Anderson

Department of Educational Psychology

ILLINOIS

©Board of Trustees, University of Illinois

Fall 2021

Overview

- Metropolis algorithm
- Revisit anorexia data.
- Metropolis algorithm for mean and variance of normal.
- Anorexia data.
- Metropolis-Hastings (MH) Algorithm
- Summary

Depending on the book that you select for this course, read either Gelman et al. pp 275-291 or Kruschke Chapters pp 143–218. I am relying more on Gelman et al.

Recall

- 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.
- Grid computations are fine for 1 or 2 parameters.
- We need algorithms for more complex problems because
  - Algebra of analytic solution becomes overwhelming .
  - Grid takes too much time and becomes computationally prohibitive.
  - Too difficult for most applications.
- Gibbs sampling requires that we an get full conditional distributions for some joint distribution, can sample from conditional, and works best when using conjugate priors.
- Metropolis-Hastings is general algorithm

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 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,

$$p(y_1, \ldots, y_n|\theta, \sigma^2) = (2\pi\sigma^2)^{-n/2} \prod_{i=1}^{n} \exp\left\{-\frac{1}{2}\left(\frac{y_i - \theta}{\sigma}\right)^2\right\}$$

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

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, $p(\theta|y)$.

Overview of Stochastic Methods & Algorithms

MCMC alogrithms:

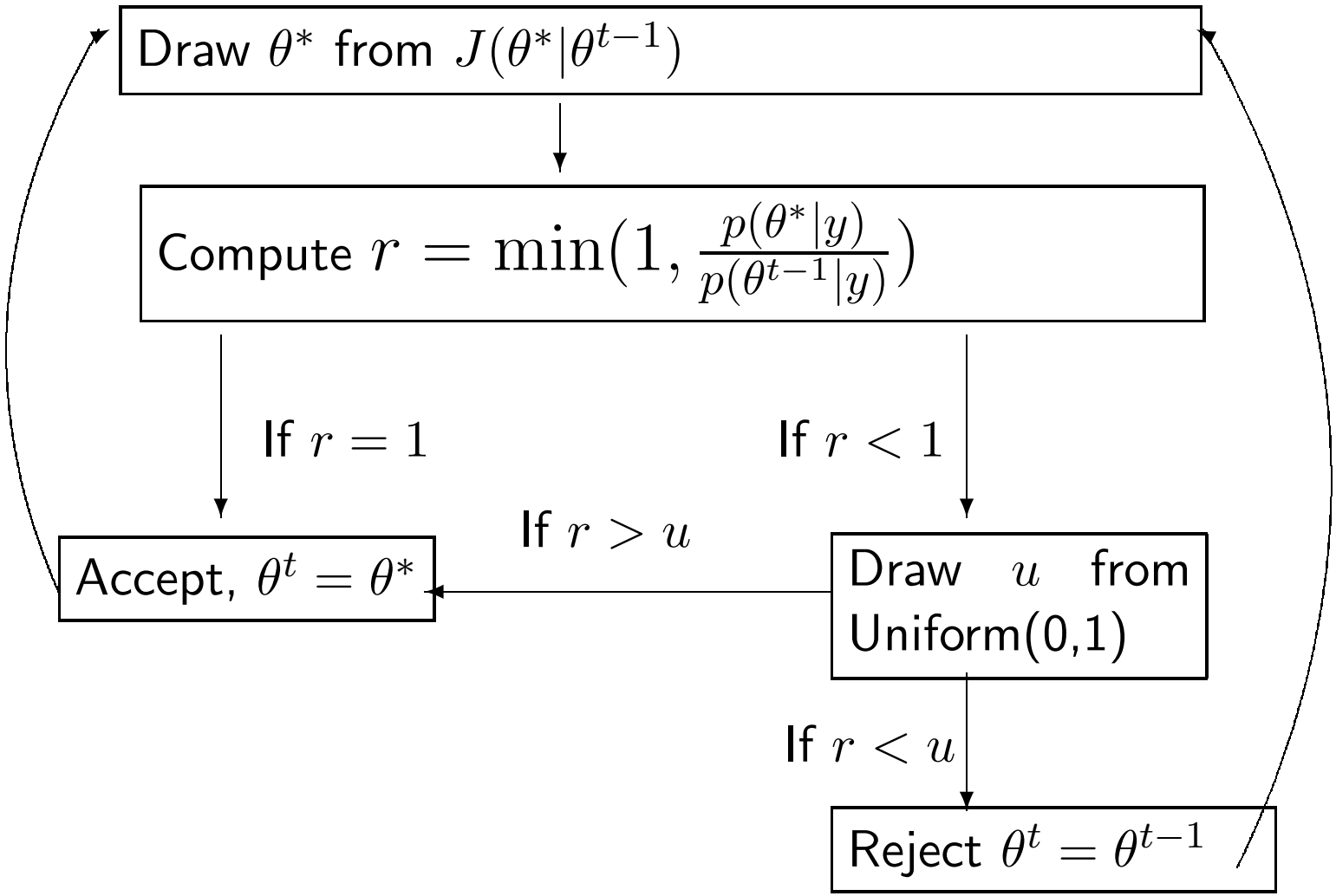
- Gibbs Sampling
- Metropolis and Metroplogis-Hasting algorithms
- Hamiltonian Sampling

Implementations in R

- Programm in base R
- rjags, runjags, jagsUI, etc.
- mcmc, MCMCpack, others?
- rstan (wrappef for Stan)

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

Metropolis Algorithm



Jumping Distribution

The jumping or "proposal" distribution

- Must be symmetric; that is, $J(\theta_a|\theta_b) = J(\theta_b|\theta_a)$. For example,
  - Uniform$[0, 1]$
  - Normal (μ, σ^2)
  - Student-t
- Later, we can allow the jumping distribution to be anything.

Jumping Distribution

The standard deviation of the jumping distribution impacts how long it take chain to get to stationary point; that is, need to ensure that values θ are sampled all over where possible values could be.

- If jump standard deviation is too small, need to take long time to get to convergence. The acceptance ratio will tend to equal 1.
- If jump standard deviation is too large, many θ^*s will be thrown away (i.e., it gets stuck); therefore, need many iterations to get to convergence...it will take a long time.
- If the jump standard deviation is just right, then accept and reject. Convention choose sd so that accept between 20% to 50%.

How to decide?
Run some simulations to try multiple values. This will be illustrated using anorexia data.

Acceptance Ratio

- If $p(\theta^*|y) > p(\theta^{t-1}|y)$ then accept and $\theta_t = \theta^*$.
- If $p(\theta^*|y) < p(\theta_{t-1}|y)$ then θ^* not necessarily new value.

However, to compare them we compute an acceptance Ratio

$$\frac{p(\theta^*|y)}{p(\theta^{t-1}|y)}$$

Bayes Theorem gives us these.

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

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

- Need to choose prior and likelihood.
- If θ^* is better than θ^{t-1}, then $r \geq 1$.
- If θ^* is not as good at θ^{t-1}, we may still accept θ^*.

Accept or Reject Proposed Value

- If $p(\theta^*|y) \geq p(\theta^{t-1}|y)$ the proposed values θ^* is "accepted"; that is,

$$\frac{p(\theta^*|y)}{p(\theta^{t-1}|y)} \geq 1$$

- If $p(\theta^*|y) < p(\theta^{t-1}|y)$ the proposed values θ^* 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,

$$\begin{aligned} \text{If } r > u &\rightarrow \text{ accept proposed and } \theta_t = \theta^* \\ \text{If } r < u &\rightarrow \text{ reject proposed and } \theta_t = \theta_{t-1} \end{aligned}$$

- Repeat many, many times and resulting $\{\theta_1, \theta_2, \ldots, \theta_{\text{large numer}}\}$ 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 (and check convergence)
  - 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

  Needs some tuning for optimal efficiency (e.g., selection of jumping rule)

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

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

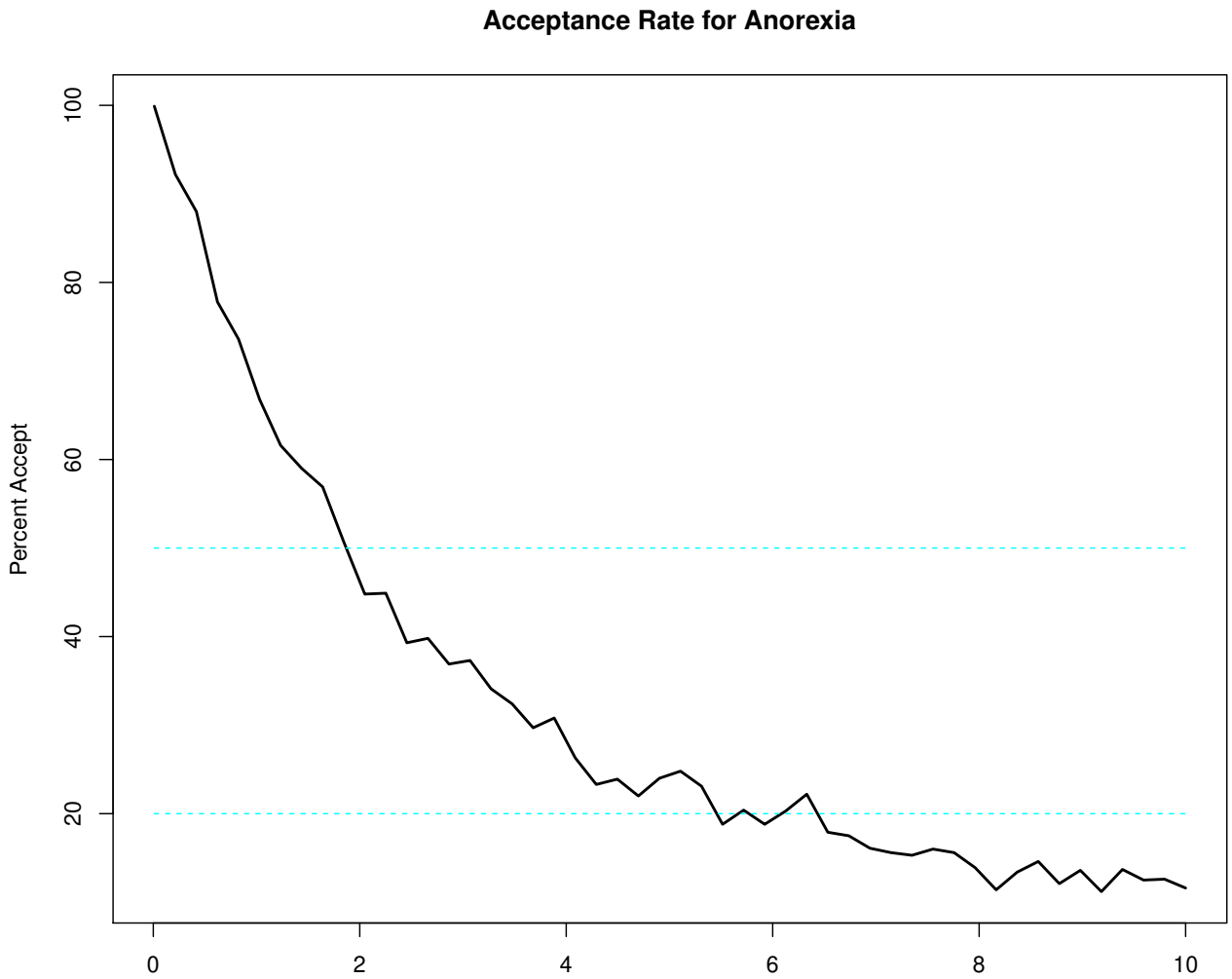
Tuning the Algorithm for Anorexia Data

Tuning was very important with these data.

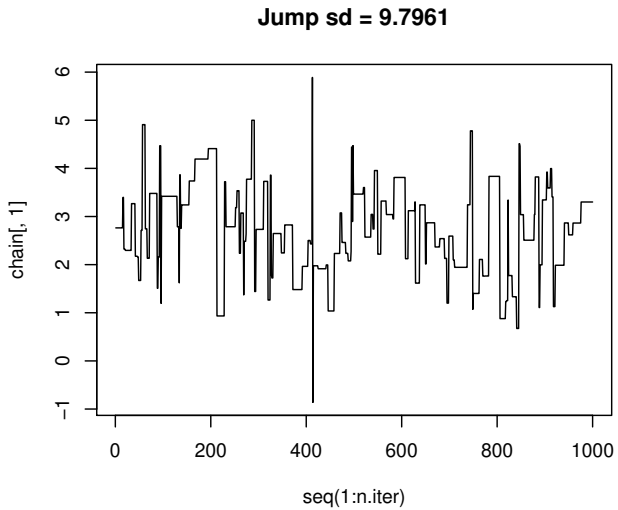
Different inputs were tried:

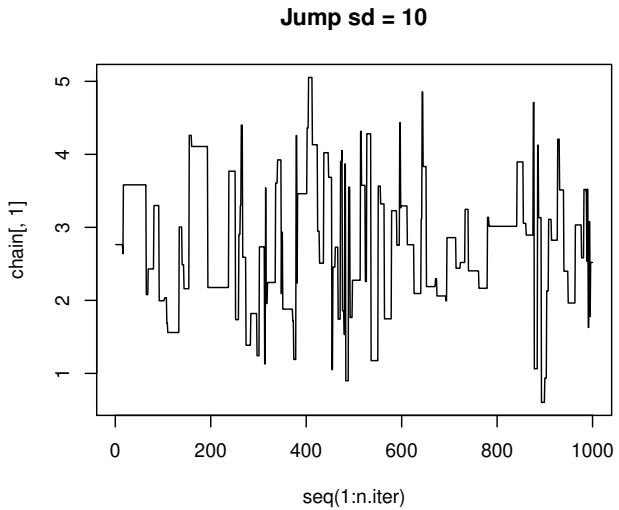
- Starting values for the mean to ensure that chains were stable.
- Set acceptance rate of about 50% to 45% (for lots of parameters aim for 20%).
- Ran a little R script I wrote to help select jump standard deviations to get one that works well –acceptance rate $\approx 40\%$..
- With good input, need fewer iterations.
- I settled on jump standard deviation $= 1.75$ and 5,000 iterations (more than we need?).

Percent Acceptance

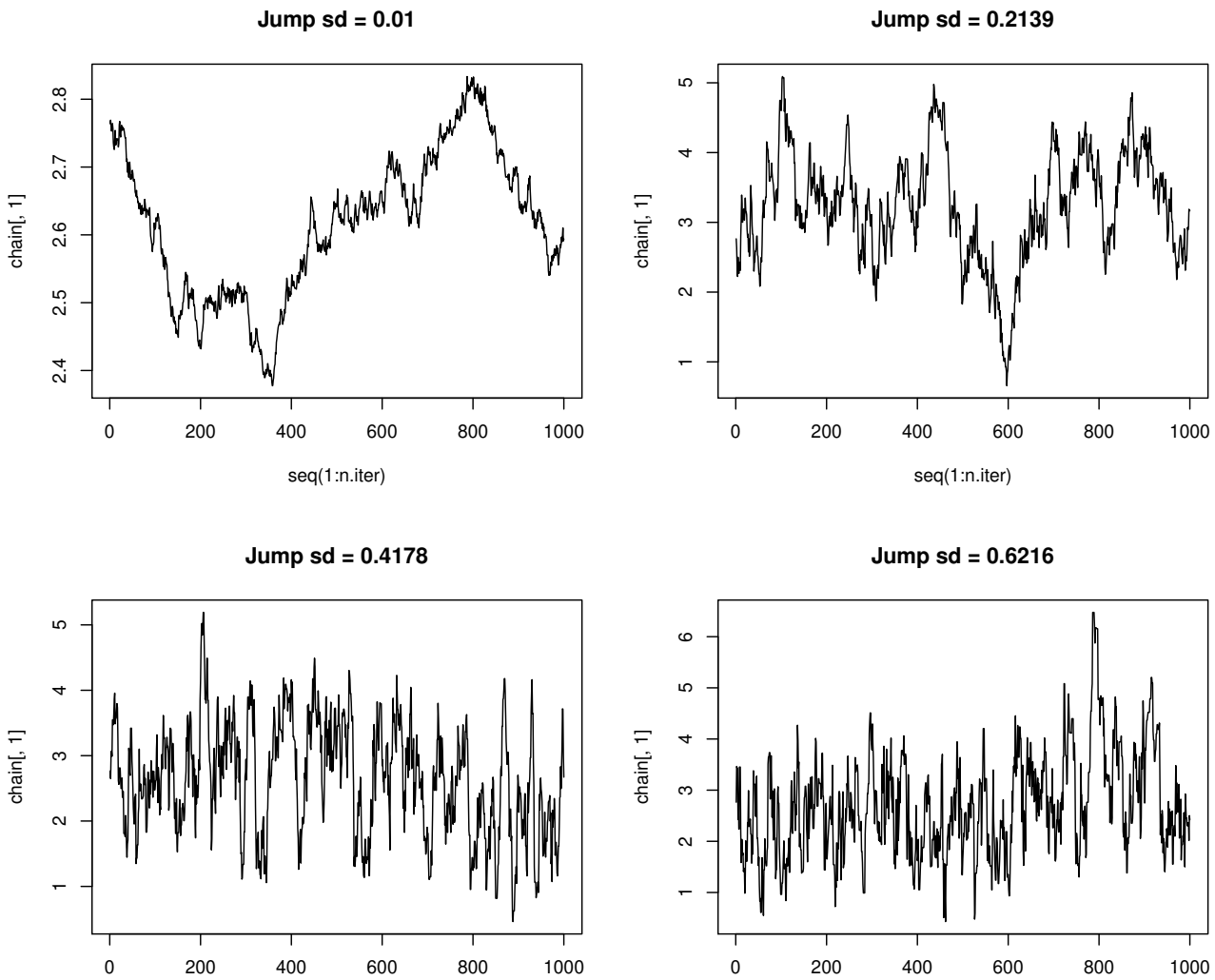


Trace Plots: jumpSD Too Large

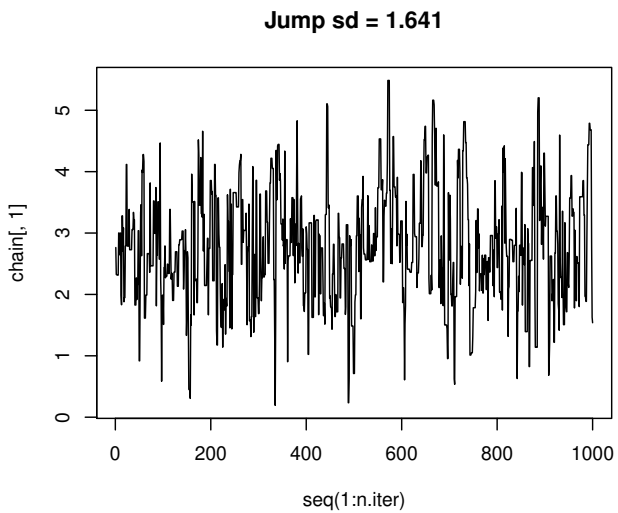


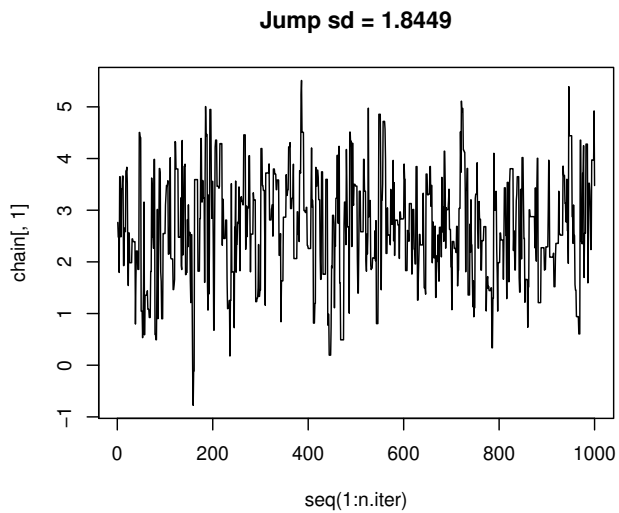


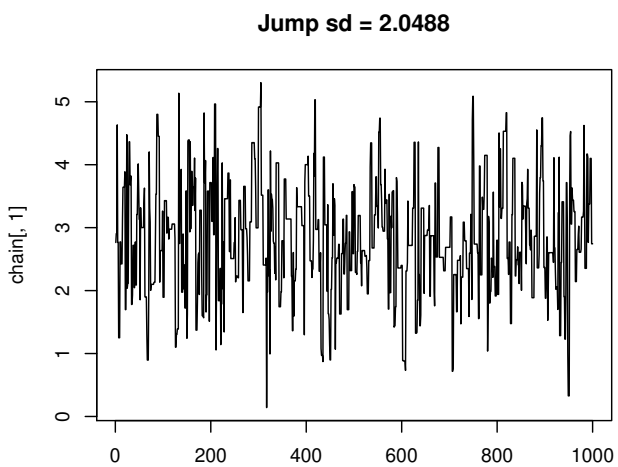
Trace Plots jumpSD Too Small

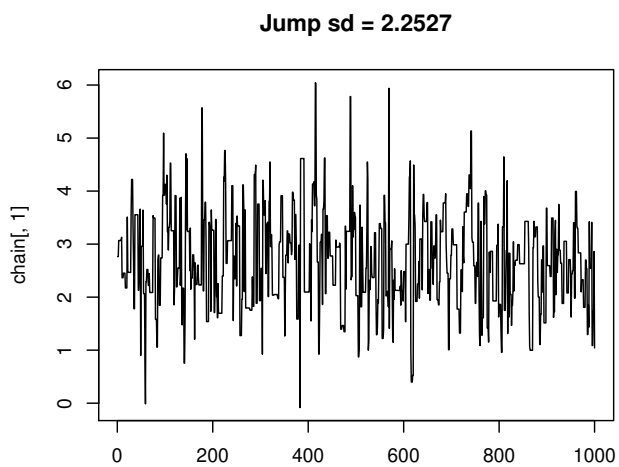


Trace Plots jumpSD Pretty Good







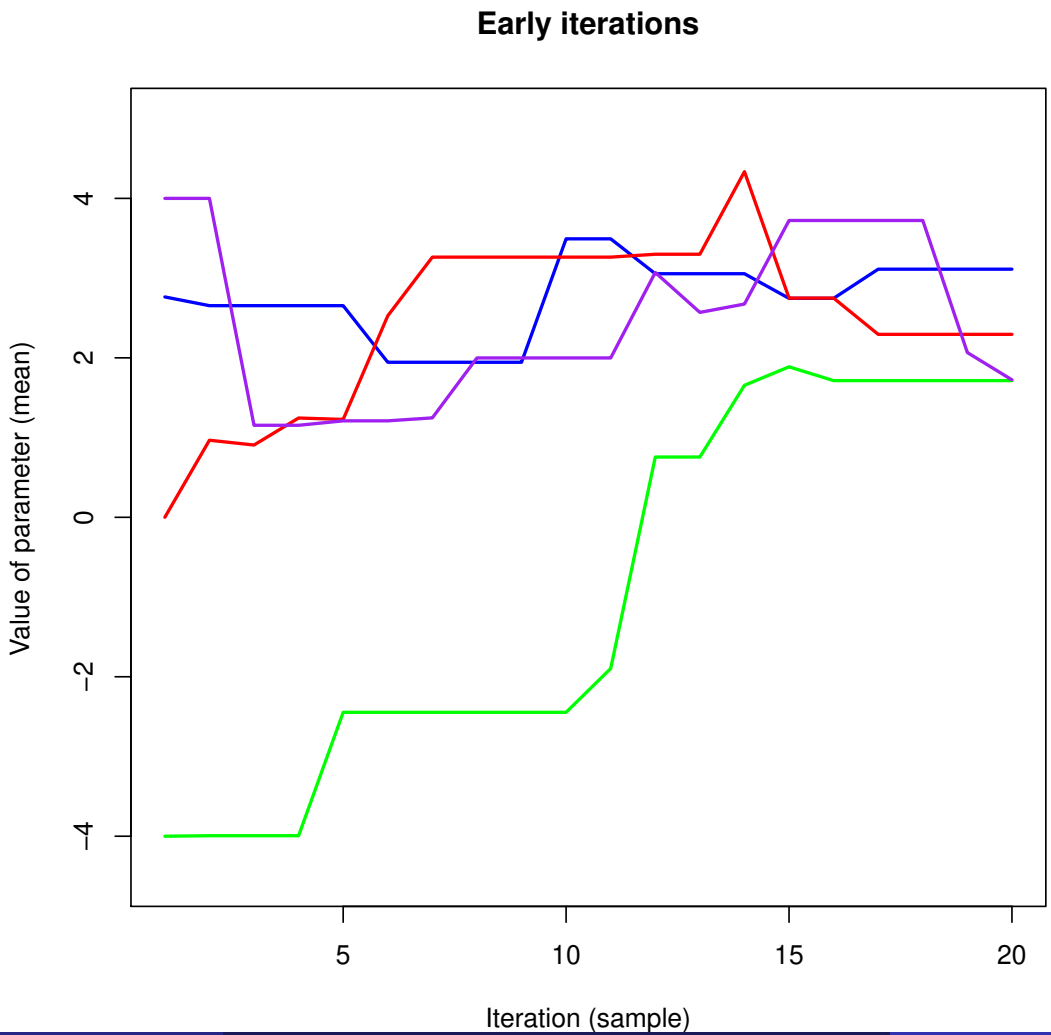


Results for Anorexia Data

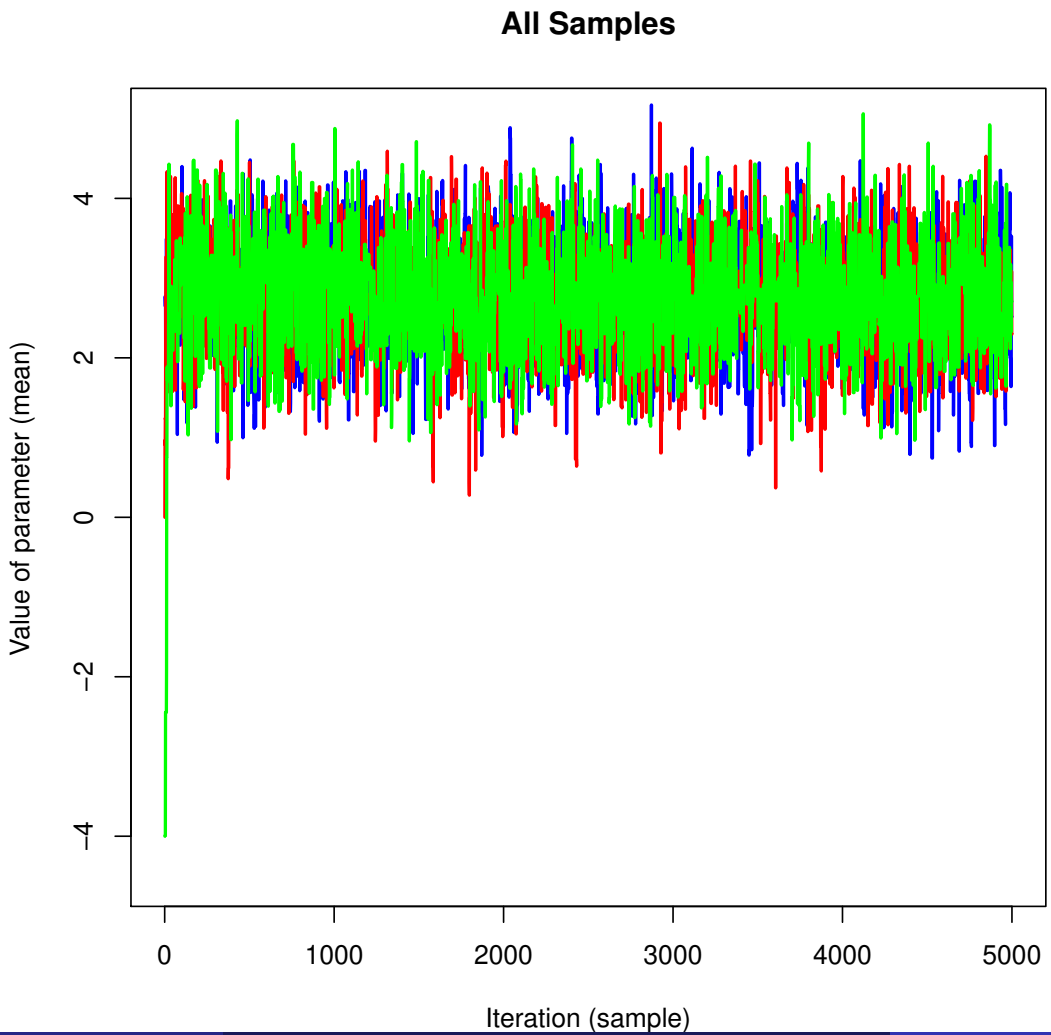
- 4 chains with starting values equal to $\bar{y}$, 0, -4, 4
- $\tau = \sqrt{s^2/n}$ = standard error of mean
- jump std $= 1.75$
- 5000 iterations and saved 4000 $\times$ 4 for posterior.

	acceptance rate	Sample Statistics of Posterior 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

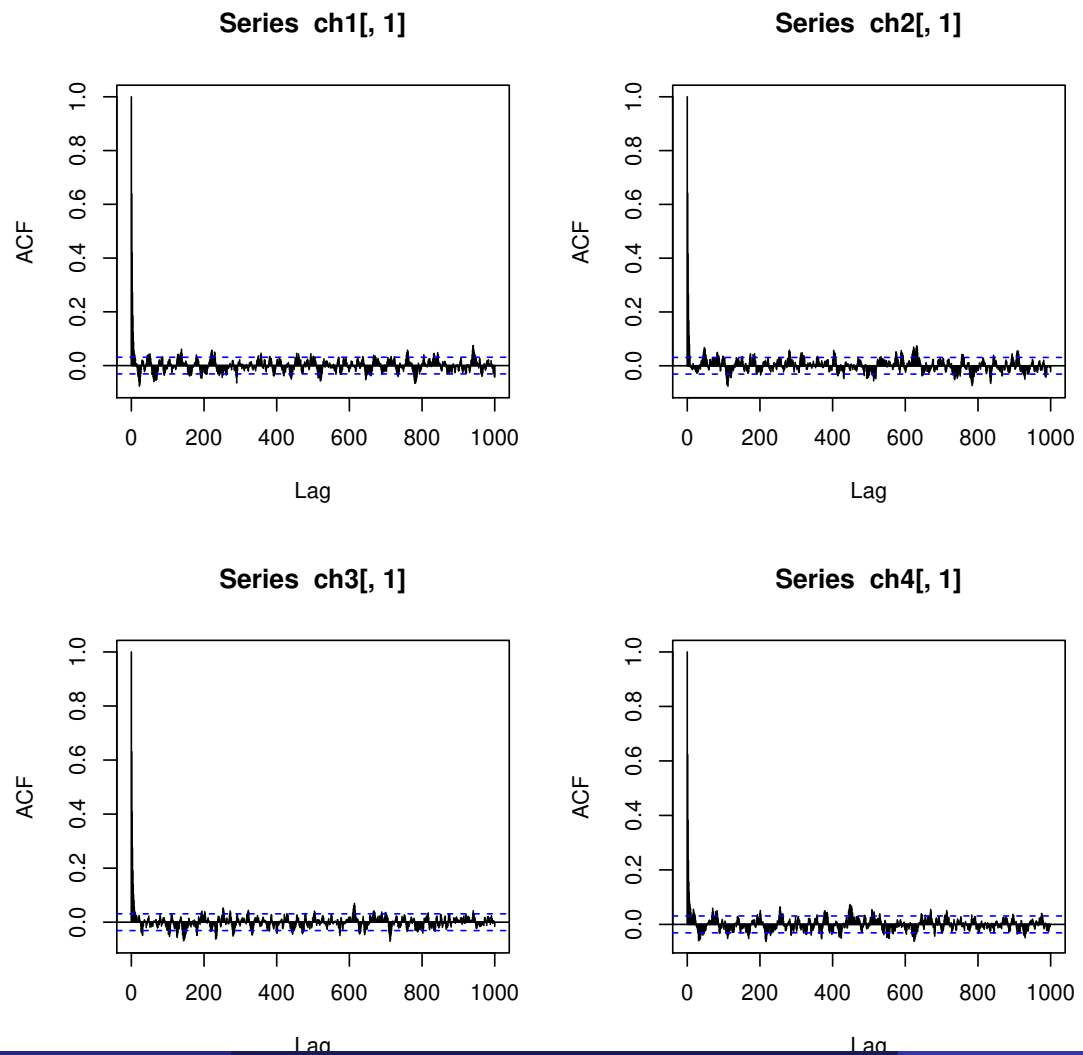
Anorexia Data: Early iterations of 4 chains



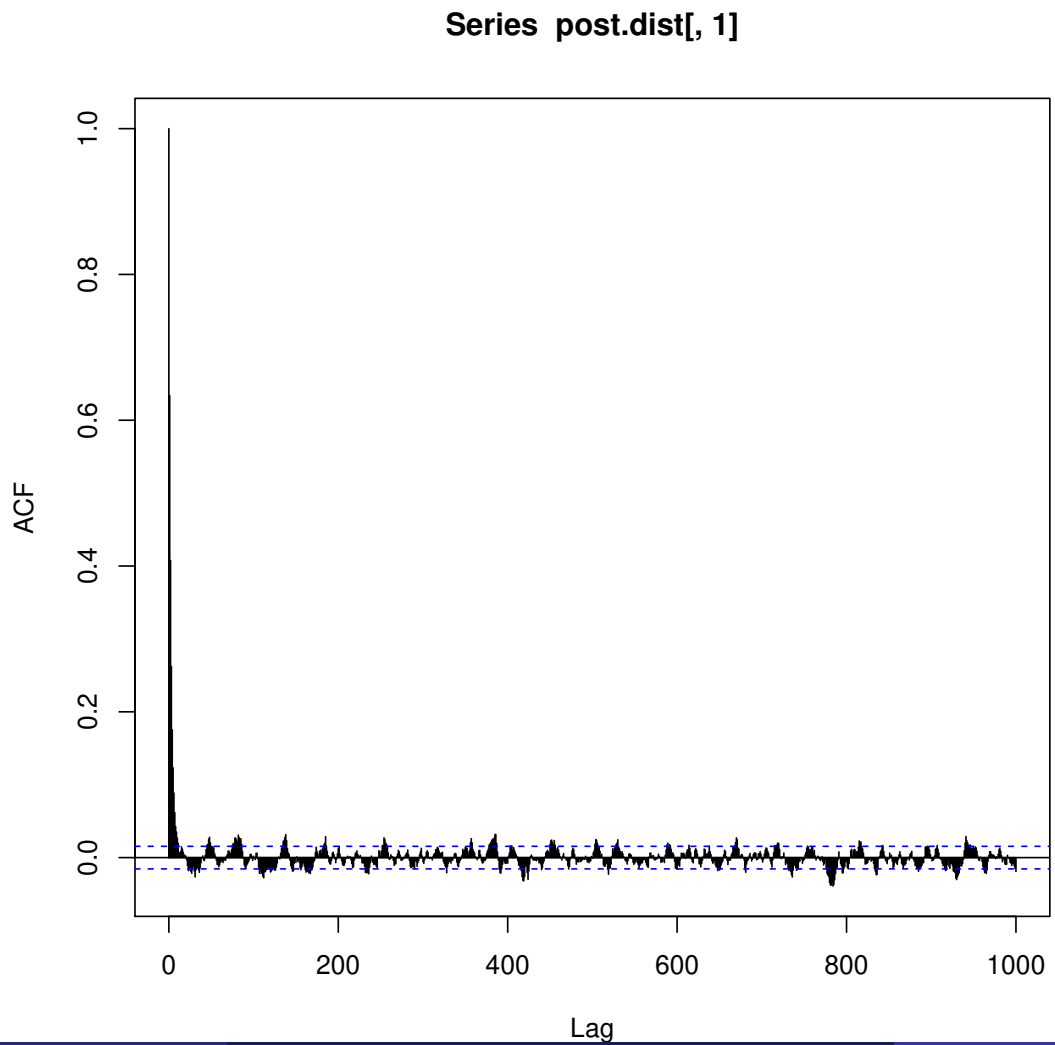
Anorexia Data: Trace Plots of 4 chains



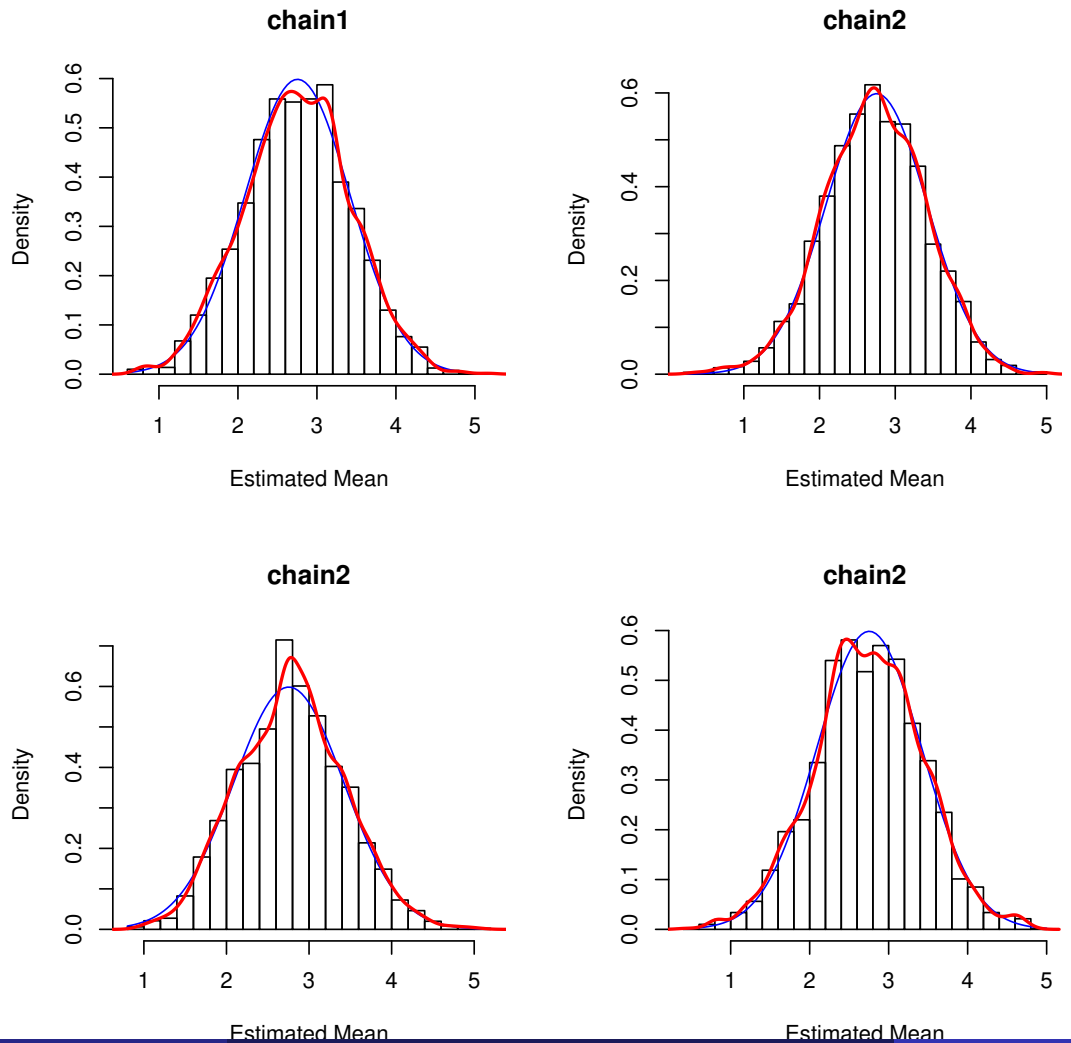
Anorexia Data: Auto-Correlations



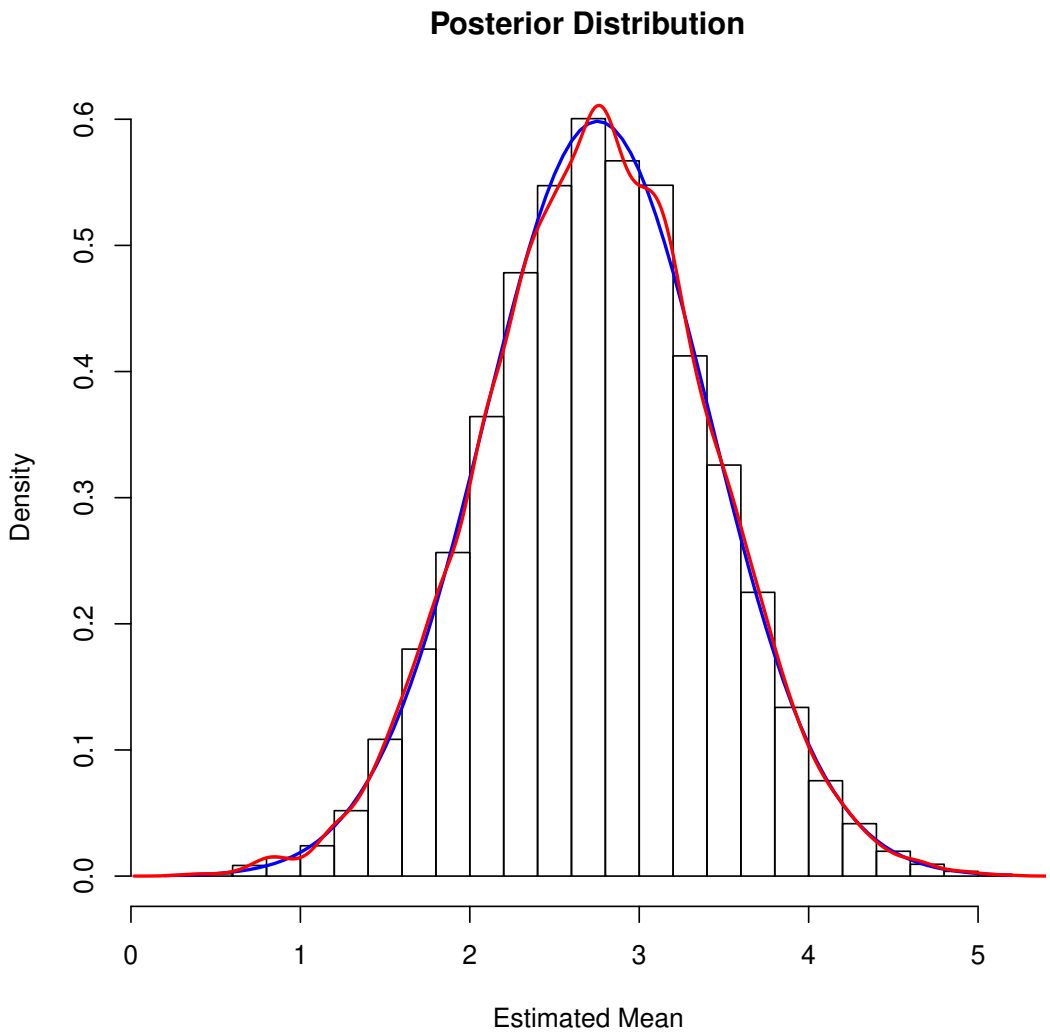
Anorexia Data: Auto-Correlations for Posterior



Anorexia Data: Density Estimates



Anorexia Data: Posterior Distribution & Density



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

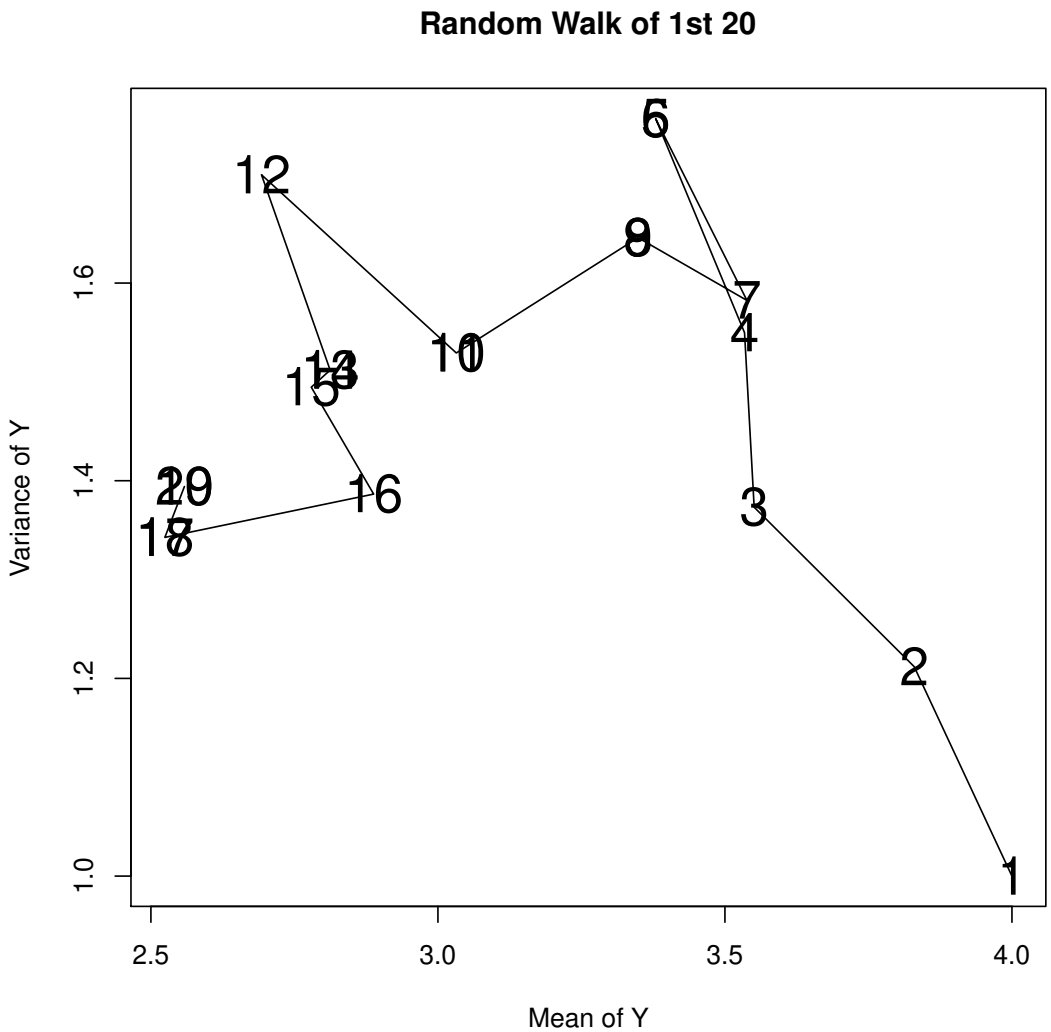
First we need some data:

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

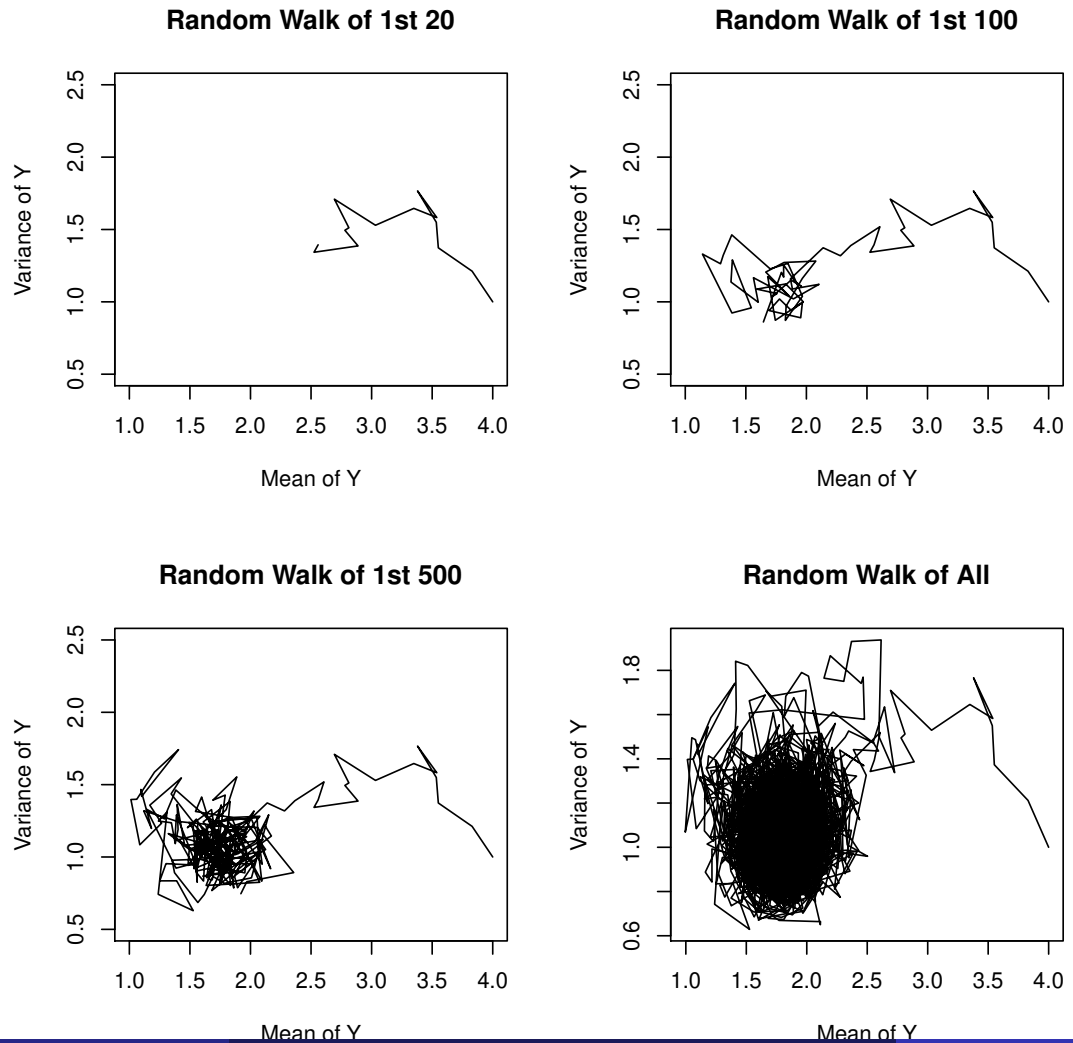
Set the starting values for μ and σ and run the function:

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

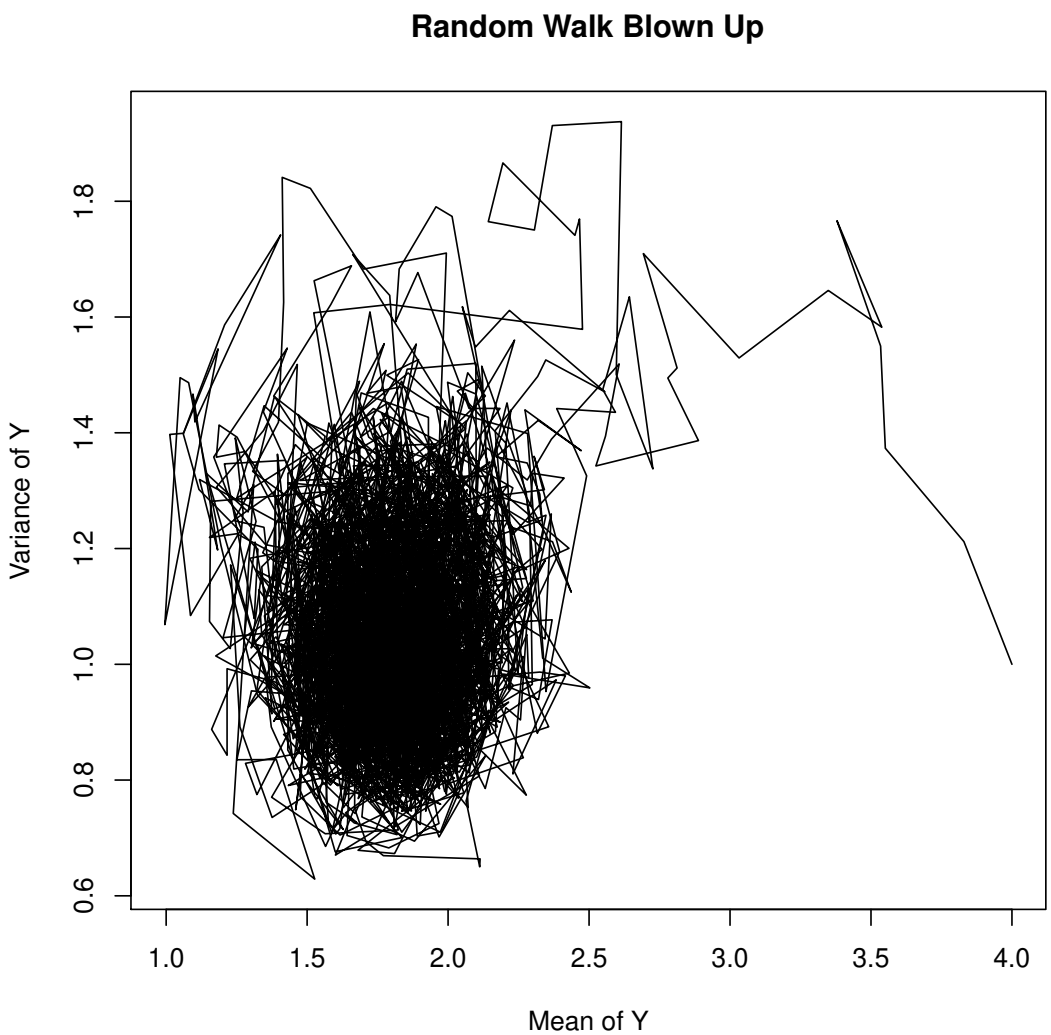
Random Walk Through the Parameter Space



More Random Walks Through the Parameter Space

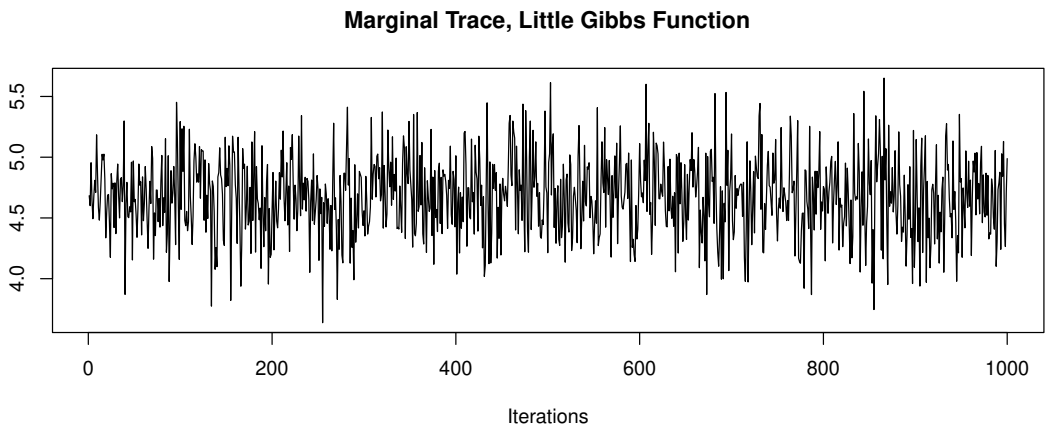


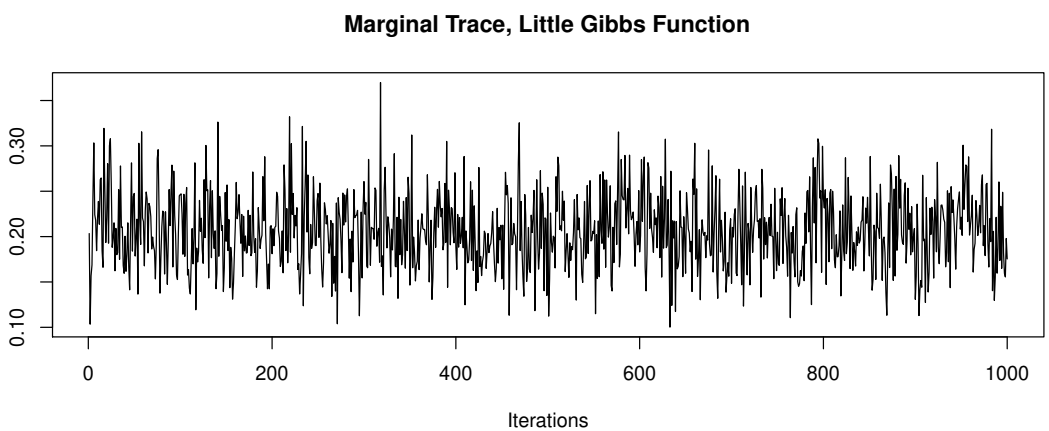
Longest Random Walk Through the Parameter Space



Individual Trace Plots

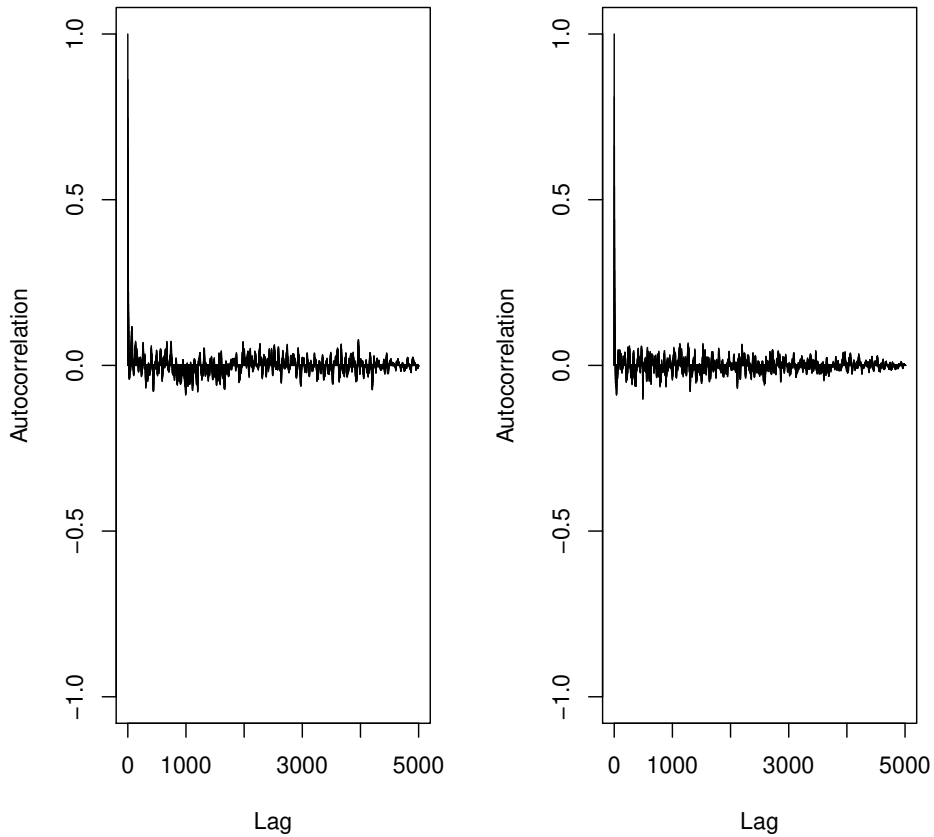
```
traceplot(chain1,smooth=F,type='l',xlab='Iterations',ylab=Parameter Values')
```





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)
```

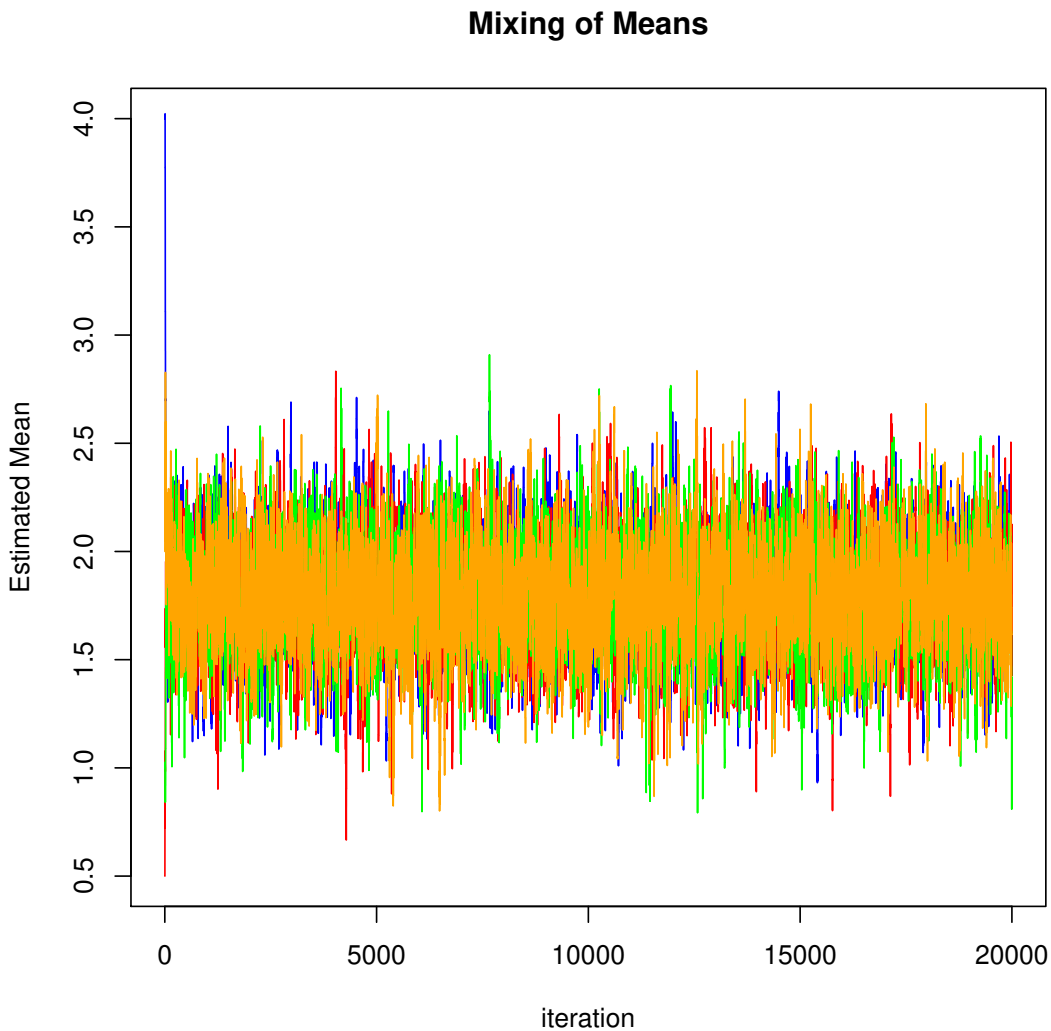
var1	var2
375.7252	524.1868

How to Improve Simulation

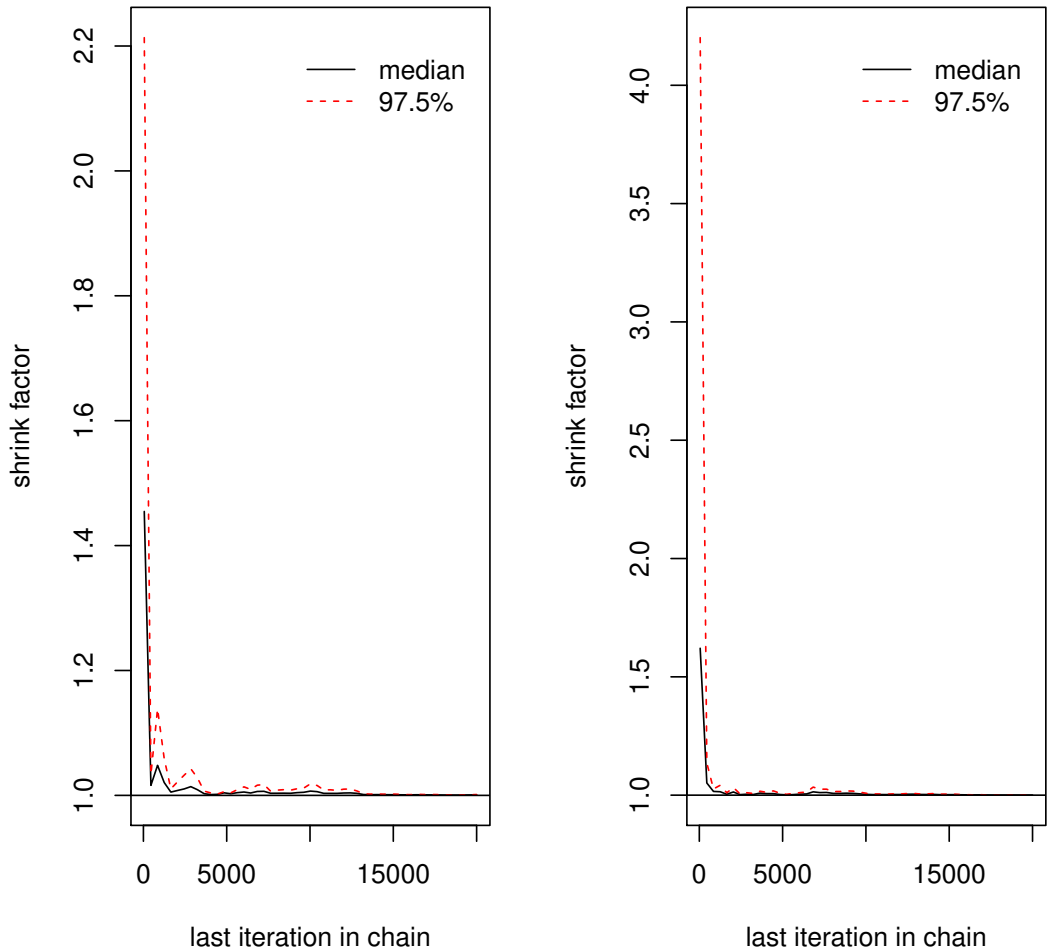
This simulation is pretty good, if you need to improve simulation consider

- 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.

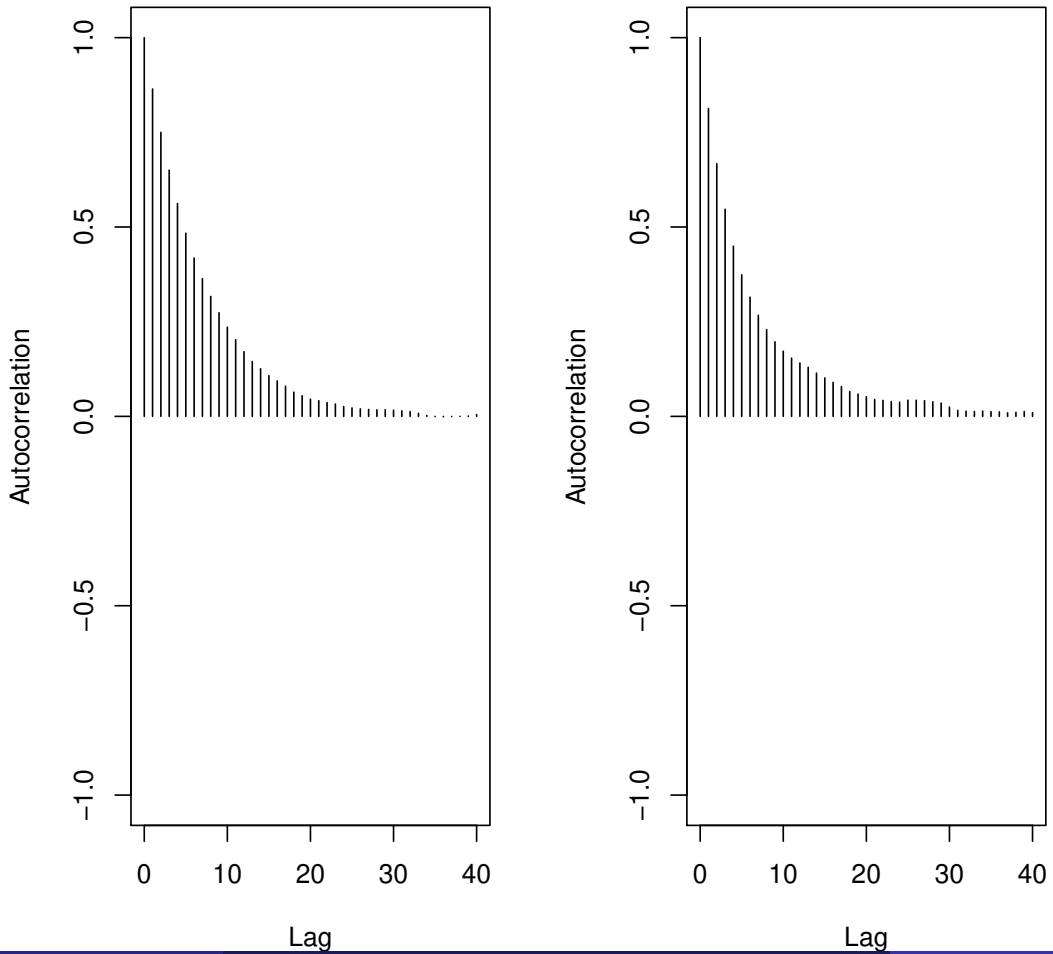
Iterations=20,000, 4 chains



Shrink Factor

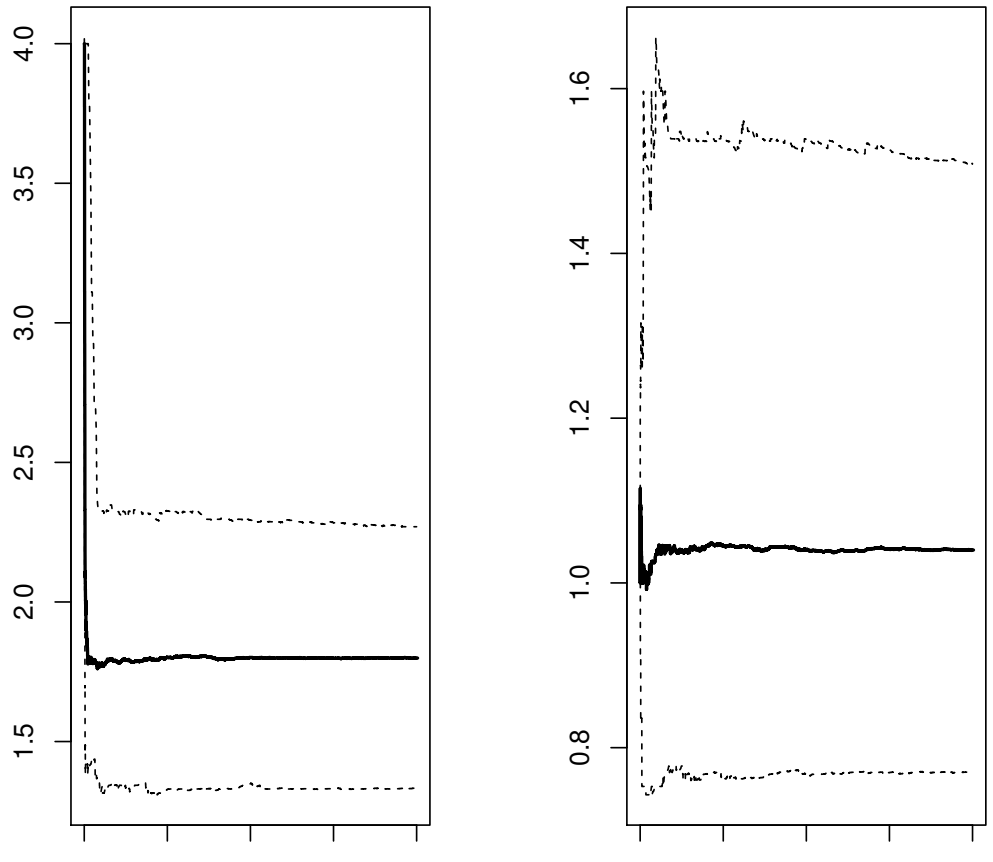


Autocorrelations (one chain)



Cumulative Quantile Plot

The 2.5^{th}, $.50^{th}$, and 97.5^{th} percentiles.



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

Metropolis-Hastings Alogrithm

A very general MCMC algorithm. Special cases include

- Metropolis algorithm
- Gibbs sampling
- Hamiltonian sampling

Extension of Metropolis to allow for

- Asymmetric jumping distribution ("jumping rules"), i.e., $J(\theta_a|\theta_b) \neq J(\theta_b|\theta_a)$.
- Adds a correction factor to deal with asymmetry in the distribution

$$r = \frac{p(\theta^*|y)/J(\theta^*|\theta^{t-1})}{p(\theta^{t-1}|y)/J(\theta^{t-1}|\theta^*)} = \frac{p(y|\theta^*)p(\theta^*)}{p(y|\theta^{t-1})p(\theta^{t-1})} \times \underbrace{\frac{J(\theta^{t-1}|\theta^*)}{J(\theta^*|\theta^{t-1})}}_{correction\,factor}$$

Metropolis-Hastings Algorithm

Allowing asymmetric jumping

- Can increase speed of random walk
- Convergence to target distribution can be proven.
- Metropolis is a special case where jumping distribution is symmetric, in which case the correction factor $= 1$.
- Gibbs is also special case, . . .

Metropolis-Hastings Algorithm

The jumping distribution for Gibbs at jth step and iteration t, the

$$J_{j,t}^{gibbs} = \begin{cases} p(\theta_j^*|\theta_{-j}^{t-1}) & \text{if } \theta_{-j}^* = \theta_{-j}^{t-1} \\ 0 & \text{otherwise} \end{cases}$$

$$r = \frac{p(\theta^*|y)/J_{j,t}^{gibbs}(\theta_{j,t}^*|\theta_{-j}^{t-1})}{p(\theta^{t-1}|y)/J_{j,t}^{gibbs}(\theta_{j,t}^{t-1}|\theta_{-j}^{t-1})} = \frac{p(\theta^*|y)/p(\theta_j^*|\theta_{-j}^{t-1},y)}{p(\theta^{t-1}|y)/p(\theta_j^{t-1}|\theta_{-j}^{t-1},y)}$$

$$= \frac{p(\theta_{-j}^{t-1}|y)}{p(\theta_{-j}^{t-1}y)}$$

$$= 1 \longleftarrow \text{always accept, no rejection}$$

Since $\theta_{-j}^* = \theta_{-j}^{t-1}$; that is, parameters are the same for those other than j; therefore, $p(\theta^*|y) = p(\theta_j^*, \theta_{-j}^*|y) = p(\theta_j^*|\theta_{-j}^{t-1}, y)p(\theta_{-j}^{t-1}|y)$.
Also, $p(\theta^{t-1}|y) = p(\theta_j^{t-1}, \theta_{-j}^{t-1}|y) = p(\theta_j^{t-1}|\theta_{-j}^{t-1}, y)p(\theta_{-j}^{t-1}|y)$.

Increases in Efficiency

- Efficient Gibbs (& Metropolois): transform or re-parameterize so that have independent components — will show this via brms.
- Auxiliary variables — an example of this is Hamiltonian sampling $\rightarrow$ Next topic.
- And more, but this goes beyond this course