Markov chain Monte Carlo

Timothy Hanson[1] and Alejandro Jara[2]

[1] Division of Biostatistics, University of Minnesota, USA
[2] Department of Statistics, Universidad de Concepción, Chile

IAP-Workshop 2009 Modeling Association and Dependence in Complex Data

Catholic University of Leuven, Leuven, November, 2009

Outline

Metropolis Hastings

- Recall that MCMC constructs a transition kernel $k(\theta|\theta^{k-1})$ that yields the posterior $p(\theta|\mathbf{y})$ as the stationary distribution of the Markov chain.
- MCMC iterates are generated directly from the kernel:

$$\begin{aligned}
\theta^1|\theta^0 &\sim k(\cdot|\theta^0), \\
\theta^2|\theta^1 &\sim k(\cdot|\theta^1), \\
\theta^3|\theta^2 &\sim k(\cdot|\theta^2), \\
&\vdots \\
\theta^k|\theta^{k-1} &\sim k(\cdot|\theta^{k-1})
\end{aligned}$$

Metropolis Hastings transition kernel

- Let $q(\theta^*|\theta)$ be *any* conditional density. Under kernel $k(\cdot|\theta^{k-1})$,

$$\theta^k|\theta^{k-1} \left\{ \begin{array}{ll} \sim \frac{q(\theta|\theta^{k-1})\rho(\theta^{k-1},\theta)}{\int q(\theta|\theta^{k-1})\rho(\theta^{k-1},\theta)d\theta} & \text{with prob. } 1 - s(\theta^{k-1}) \\ = \theta^{k-1} & \text{with prob. } s(\theta^{k-1}) \end{array} \right\}$$

Metropolis Hastings transition kernel

- Let $q(\theta^*|\theta)$ be *any* conditional density. Under kernel $k(\cdot|\theta^{k-1})$,

$$\theta^k|\theta^{k-1} \left\{ \begin{array}{ll} \sim \frac{q(\theta|\theta^{k-1})\rho(\theta^{k-1},\theta)}{\int q(\theta|\theta^{k-1})\rho(\theta^{k-1},\theta)d\theta} & \text{with prob. } 1-s(\theta^{k-1}) \\ = \theta^{k-1} & \text{with prob. } s(\theta^{k-1}) \end{array} \right\}$$

- where

$$\rho(\theta^{k-1},\theta) = \min\left\{1, \frac{p(\theta|y)p(\theta)q(\theta^{k-1}|\theta)}{p(\theta^{k-1}|y)p(\theta^{k-1})q(\theta|\theta^{k-1})}\right\}.$$

Metropolis Hastings transition kernel

- Let $q(\theta^*|\theta)$ be *any* conditional density. Under kernel $k(\cdot|\theta^{k-1})$,

$$\theta^k|\theta^{k-1} \left\{ \begin{array}{ll} \sim \frac{q(\theta|\theta^{k-1})\rho(\theta^{k-1},\theta)}{\int q(\theta|\theta^{k-1})\rho(\theta^{k-1},\theta)d\theta} & \text{with prob. } 1-s(\theta^{k-1}) \\ = \theta^{k-1} & \text{with prob. } s(\theta^{k-1}) \end{array} \right\}$$

- where

$$\rho(\theta^{k-1},\theta) = \min\left\{1, \frac{p(\theta|y)p(\theta)q(\theta^{k-1}|\theta)}{p(\theta^{k-1}|y)p(\theta^{k-1})q(\theta|\theta^{k-1})}\right\}.$$

- $s(\theta^{k-1})$ is the probability of $\theta^k = \theta^{k-1}$, or of θ^k staying at where it is currently:

$$s(\theta^{k-1}) = 1 - \int_{\theta\in\Theta} q(\theta|\theta^{k-1})\rho(\theta^{k-1},\theta)d\theta.$$

- Simulating the chain is easier than it might first appear. Iterate k is generated from θ^{k-1} according to
  1. Draw $\theta^* \sim q(\cdot|\theta^{k-1})$ independent of $u \sim U(0,1)$.
  2. If $u \leq \rho(\theta^{k-1}, \theta^*) = \min\left\{1, \frac{p(\theta^*|y)p(\theta^*)q(\theta^{k-1}|\theta^*)}{p(\theta^{k-1}|y)p(\theta^{k-1})q(\theta^*|\theta^{k-1})}\right\}$ then $\theta^k = \theta^*$ otherwise $\theta^k = \theta^{k-1}$.

- Simulating the chain is easier than it might first appear. Iterate k is generated from θ^{k-1} according to
  1. Draw $\theta^* \sim q(\cdot|\theta^{k-1})$ independent of $u \sim U(0,1)$.
  2. If $u \leq \rho(\theta^{k-1}, \theta^*) = \min\left\{1, \frac{p(\theta^*|y)p(\theta^*)q(\theta^{k-1}|\theta^*)}{p(\theta^{k-1}|y)p(\theta^{k-1})q(\theta^*|\theta^{k-1})}\right\}$ then $\theta^k = \theta^*$ otherwise $\theta^k = \theta^{k-1}$.
- In theory, under mild conditions, $\theta^k \xrightarrow{\mathcal{D}} p(\theta|y)$.

- Simulating the chain is easier than it might first appear. Iterate k is generated from θ^{k-1} according to
  1. Draw $\theta^* \sim q(\cdot|\theta^{k-1})$ independent of $u \sim U(0,1)$.
  2. If $u \leq \rho(\theta^{k-1}, \theta^*) = \min\left\{1, \frac{p(\theta^*|y)p(\theta^*)q(\theta^{k-1}|\theta^*)}{p(\theta^{k-1}|y)p(\theta^{k-1})q(\theta^*|\theta^{k-1})}\right\}$ then $\theta^k = \theta^*$ otherwise $\theta^k = \theta^{k-1}$.
- In theory, under mild conditions, $\theta^k \xrightarrow{\mathcal{D}} p(\theta|y)$.
- In reality, k is nowhere near ∞ and $q(\cdot|\theta^{k-1})$ needs to be picked in an intelligent way.

Metropolis algorithm: symmetric proposal

- When $q(\theta^*|\theta) = q(\theta|\theta^*)$, q is said to be *symmetric.*

Metropolis algorithm: symmetric proposal

- When $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)$, q is said to be *symmetric.*
- In this case, the acceptance probability simplifies to $\rho(\boldsymbol{\theta}^{k-1}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{p(\boldsymbol{\theta}^*|y)p(\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{k-1}|y)p(\boldsymbol{\theta}^{k-1})}\right\}$.

Metropolis algorithm: symmetric proposal

- When $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)$, q is said to be *symmetric.*
- In this case, the acceptance probability simplifies to $\rho(\boldsymbol{\theta}^{k-1}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{p(\boldsymbol{\theta}^*|y)p(\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{k-1}|y)p(\boldsymbol{\theta}^{k-1})}\right\}$.
- Metropolis *et al.* (1953) used this algorithm for computing properties of substances composed of interacting individual molecules.

Metropolis algorithm: symmetric proposal

- When $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)$, q is said to be *symmetric.*
- In this case, the acceptance probability simplifies to $\rho(\boldsymbol{\theta}^{k-1}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{p(\boldsymbol{\theta}^*|y)p(\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{k-1}|y)p(\boldsymbol{\theta}^{k-1})}\right\}$.
- Metropolis *et al.* (1953) used this algorithm for computing properties of substances composed of interacting individual molecules.
- Also called a random walk chain.

Metropolis algorithm: symmetric proposal

- When $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)$, q is said to be *symmetric.*
- In this case, the acceptance probability simplifies to $\rho(\boldsymbol{\theta}^{k-1}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{p(\boldsymbol{\theta}^*|y)p(\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{k-1}|y)p(\boldsymbol{\theta}^{k-1})}\right\}$.
- Metropolis *et al.* (1953) used this algorithm for computing properties of substances composed of interacting individual molecules.
- Also called a random walk chain.
- Used *a lot*. Often $\boldsymbol{\theta}^* \sim N_p(\boldsymbol{\theta}, \mathbf{S})$ for some $\mathbf{S}$. "Tuning" required to get good $\mathbf{S}$.

Independence proposal

- When $q(\theta^*|\theta) = q(\theta^*)$, q does not care about the last θ.

Independence proposal

- When $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}^*)$, q does not care about the last $\boldsymbol{\theta}$.
- In this case, the acceptance probability is $\rho(\boldsymbol{\theta}^{k-1}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{p(\boldsymbol{\theta}^*|y)p(\boldsymbol{\theta}^*)q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}^{k-1}|y)p(\boldsymbol{\theta}^{k-1})q(\boldsymbol{\theta}^*)}\right\}$.

Independence proposal

- When $q(\theta^*|\theta) = q(\theta^*)$, q does not care about the last θ.
- In this case, the acceptance probability is $\rho(\theta^{k-1}, \theta^*) = \min\left\{1, \frac{p(\theta^*|y)p(\theta^*)q(\theta)}{p(\theta^{k-1}|y)p(\theta^{k-1})q(\theta^*)}\right\}$.
- Hastings, W.K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97-109. Cited thousands of times.

Independence proposal

- When $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}^*)$, q does not care about the last $\boldsymbol{\theta}$.
- In this case, the acceptance probability is $\rho(\boldsymbol{\theta}^{k-1}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{p(\boldsymbol{\theta}^*|y)p(\boldsymbol{\theta}^*)q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}^{k-1}|y)p(\boldsymbol{\theta}^{k-1})q(\boldsymbol{\theta}^*)}\right\}$.
- Hastings, W.K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97-109. Cited thousands of times.
- Called independence chain. Proposal $q(\cdot)$ should have "fatter tails" in $\mathbb{R}^p$ than $p(\boldsymbol{\theta}|\mathbf{y})$ to get good sample. Same idea as in importance sampling.

Independence proposal

- When $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}^*)$, q does not care about the last $\boldsymbol{\theta}$.
- In this case, the acceptance probability is $\rho(\boldsymbol{\theta}^{k-1}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{p(\boldsymbol{\theta}^*|y)p(\boldsymbol{\theta}^*)q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}^{k-1}|y)p(\boldsymbol{\theta}^{k-1})q(\boldsymbol{\theta}^*)}\right\}$.
- Hastings, W.K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97-109. Cited thousands of times.
- Called independence chain. Proposal $q(\cdot)$ should have "fatter tails" in $\mathbb{R}^p$ than $p(\boldsymbol{\theta}|\mathbf{y})$ to get good sample. Same idea as in importance sampling.
- When is $\rho = 1$? This is the case for Gibbs sampling...

Independence proposal

- When $q(\theta^*|\theta) = q(\theta^*)$, q does not care about the last θ.
- In this case, the acceptance probability is $\rho(\theta^{k-1}, \theta^*) = \min\left\{1, \frac{p(\theta^*|y)p(\theta^*)q(\theta)}{p(\theta^{k-1}|y)p(\theta^{k-1})q(\theta^*)}\right\}$.
- Hastings, W.K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97-109. Cited thousands of times.
- Called independence chain. Proposal $q(\cdot)$ should have "fatter tails" in $\mathbb{R}^p$ than $p(\theta|\mathbf{y})$ to get good sample. Same idea as in importance sampling.
- When is $\rho = 1$? This is the case for Gibbs sampling...
- Often $q(\cdot) = N_p(\cdot|\hat{\theta}, \hat{\mathbf{\Sigma}})$.

Gibbs sampling

- Gibbs sampling can be viewed a series of componentwise Metropolis-Hastings algorithm steps, each with acceptance probability one.

Gibbs sampling

- Gibbs sampling can be viewed a series of componentwise Metropolis-Hastings algorithm steps, each with acceptance probability one.
- From $\boldsymbol{\theta}^{k-1}$, $\boldsymbol{\theta}^k = (\theta_1^k, \ldots, \theta_p^k)$ is generated by "successive substitution sampling." When $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ this boils down to

$$\begin{aligned}
\theta_1^k &\sim p(\theta_1 | \theta_2 = \theta_2^{k-1}, \theta_3 = \theta_3^{k-1}, y) \\
\theta_2^k &\sim p(\theta_2 | \theta_1 = \theta_1^{k} \quad, \theta_3 = \theta_3^{k-1}, y) \\
\theta_3^k &\sim p(\theta_3 | \theta_1 = \theta_1^{k} \quad, \theta_2 = \theta_2^{k} \quad, y)
\end{aligned}$$

Gibbs sampling

- Gibbs sampling can be viewed a series of componentwise Metropolis-Hastings algorithm steps, each with acceptance probability one.
- From $\boldsymbol{\theta}^{k-1}$, $\boldsymbol{\theta}^k = (\theta_1^k, \ldots, \theta_p^k)$ is generated by "successive substitution sampling." When $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ this boils down to

$$\begin{aligned}
\theta_1^k &\sim p(\theta_1 | \theta_2 = \theta_2^{k-1}, \theta_3 = \theta_3^{k-1}, y) \\
\theta_2^k &\sim p(\theta_2 | \theta_1 = \theta_1^{k} \quad, \theta_3 = \theta_3^{k-1}, y) \\
\theta_3^k &\sim p(\theta_3 | \theta_1 = \theta_1^{k} \quad, \theta_2 = \theta_2^{k} \quad, y)
\end{aligned}$$

- Generalizes to any $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_p)'$.

Gibbs sampling

- Gibbs sampling can be viewed a series of componentwise Metropolis-Hastings algorithm steps, each with acceptance probability one.
- From $\boldsymbol{\theta}^{k-1}$, $\boldsymbol{\theta}^k = (\theta_1^k, \ldots, \theta_p^k)$ is generated by "successive substitution sampling." When $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ this boils down to

$$\begin{aligned}
\theta_1^k &\sim p(\theta_1 | \theta_2 = \theta_2^{k-1}, \theta_3 = \theta_3^{k-1}, y) \\
\theta_2^k &\sim p(\theta_2 | \theta_1 = \theta_1^{k} \quad, \theta_3 = \theta_3^{k-1}, y) \\
\theta_3^k &\sim p(\theta_3 | \theta_1 = \theta_1^{k} \quad, \theta_2 = \theta_2^{k} \quad, y)
\end{aligned}$$

- Generalizes to any $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_p)'$.
- Need to be able to sample from all full conditional distributions!

Each Gibbs update special case of M-H step

- Again, for simplicity, $p = 3$.

Each Gibbs update special case of M-H step

- Again, for simplicity, $p = 3$.
- Define $q(\theta^*|\theta^{k-1}) = p(\theta_1^*|\theta_2^{k-1}, \theta_3^{k-1}, y)p(\theta_2^*|\theta_1^*, \theta_3^{k-1}, y)p(\theta_3^*|\theta_1^*, \theta_2^*, y)$.

Each Gibbs update special case of M-H step

- Again, for simplicity, $p = 3$.
- Define $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1}) = p(\theta_1^*|\theta_2^{k-1}, \theta_3^{k-1}, y)p(\theta_2^*|\theta_1^*, \theta_3^{k-1}, y)p(\theta_3^*|\theta_1^*, \theta_2^*, y)$.
- This is transitional distribution (transition kernel) for Gibbs sampler, but also can be viewed as series of componentwise M-H proposals, each with acceptance probability one.

Each Gibbs update special case of M-H step

- Again, for simplicity, $p = 3$.
- Define $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1}) = p(\theta_1^*|\theta_2^{k-1}, \theta_3^{k-1}, y)p(\theta_2^*|\theta_1^*, \theta_3^{k-1}, y)p(\theta_3^*|\theta_1^*, \theta_2^*, y)$.
- This is transitional distribution (transition kernel) for Gibbs sampler, but also can be viewed as series of componentwise M-H proposals, each with acceptance probability one.
- For example $\rho(\boldsymbol{\theta}_1^{k-1}, \boldsymbol{\theta}_1^*|\boldsymbol{\theta}_2^{k-1}, \boldsymbol{\theta}_3^{k-1}, y) = 1$. Show this!

Each Gibbs update special case of M-H step

- Again, for simplicity, $p = 3$.
- Define $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1}) = p(\theta_1^*|\theta_2^{k-1}, \theta_3^{k-1}, y)p(\theta_2^*|\theta_1^*, \theta_3^{k-1}, y)p(\theta_3^*|\theta_1^*, \theta_2^*, y)$.
- This is transitional distribution (transition kernel) for Gibbs sampler, but also can be viewed as series of componentwise M-H proposals, each with acceptance probability one.
- For example $\rho(\theta_1^{k-1}, \theta_1^*|\theta_2^{k-1}, \theta_3^{k-1}, y) = 1$. Show this!
- Gibbs sampler is sequence of special M-H steps where candidate is always accepted.

Each Gibbs update special case of M-H step

- Again, for simplicity, $p = 3$.
- Define $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1}) = p(\theta_1^*|\theta_2^{k-1}, \theta_3^{k-1}, y)p(\theta_2^*|\theta_1^*, \theta_3^{k-1}, y)p(\theta_3^*|\theta_1^*, \theta_2^*, y)$.
- This is transitional distribution (transition kernel) for Gibbs sampler, but also can be viewed as series of componentwise M-H proposals, each with acceptance probability one.
- For example $\rho(\theta_1^{k-1}, \theta_1^*|\theta_2^{k-1}, \theta_3^{k-1}, y) = 1$. Show this!
- Gibbs sampler is sequence of special M-H steps where candidate is always accepted.
- Reviewed and reintroduced by Gelfand and Smith (1990).

Example: Normal data

- Model: $y_1, \ldots, y_n | \mu, \tau \overset{iid}{\sim} N(\mu, \tau^{-1})$.

Example: Normal data

- Model: $y_1, \ldots, y_n | \mu, \tau \overset{iid}{\sim} N(\mu, \tau^{-1})$.
- Prior $\mu \sim N(m, t^{-1})$ independent of $\tau \sim \Gamma(a, b)$.

Example: Normal data

- Model: $y_1, \ldots, y_n | \mu, \tau \overset{iid}{\sim} N(\mu, \tau^{-1})$.
- Prior $\mu \sim N(m, t^{-1})$ independent of $\tau \sim \Gamma(a, b)$.
- Full conditionals:

$$\begin{aligned} \mu|\tau, y &\sim N\left(\frac{n\tau\bar{y} + mt}{n\tau + t}, \frac{1}{n\tau + t}\right) \\ \tau|\mu, y &\sim \Gamma\left(a + 0.5n, b + 0.5\sum_{i=1}^{n}(y_i - \mu)^2\right) \end{aligned}$$

Example: Normal data

- Model: $y_1, \ldots, y_n | \mu, \tau \overset{iid}{\sim} N(\mu, \tau^{-1})$.
- Prior $\mu \sim N(m, t^{-1})$ independent of $\tau \sim \Gamma(a, b)$.
- Full conditionals:

$$\begin{aligned} \mu|\tau, y &\sim N\left(\frac{n\tau\bar{y} + mt}{n\tau + t}, \frac{1}{n\tau + t}\right) \\ \tau|\mu, y &\sim \Gamma\left(a + 0.5n, b + 0.5\sum_{i=1}^{n}(y_i - \mu)^2\right) \end{aligned}$$

- MCMC iterates $\{(\mu^k, \tau^k)\}_{k=1}^{M}$ generated by first picking μ^0 and τ^0. Maybe $\mu^0 = m$ and $\tau = a/b$ (why?). Then...

Gibbs sampler...

$$
\begin{aligned}
\mu^1 &\sim N\left(\frac{n\tau^0\bar{y}+mt}{n\tau^0+t}, \frac{1}{n\tau^0+t}\right) \\
\tau^1 &\sim \Gamma\left(a+0.5n, b+0.5\sum_{i=1}^{n}(y_i-\mu^1)^2\right) \\
\mu^2 &\sim N\left(\frac{n\tau^1\bar{y}+mt}{n\tau^1+t}, \frac{1}{n\tau^1+t}\right) \\
\tau^2 &\sim \Gamma\left(a+0.5n, b+0.5\sum_{i=1}^{n}(y_i-\mu^2)^2\right) \\
\mu^3 &\sim N\left(\frac{n\tau^2\bar{y}+mt}{n\tau^2+t}, \frac{1}{n\tau^2+t}\right) \\
\tau^3 &\sim \Gamma\left(a+0.5n, b+0.5\sum_{i=1}^{n}(y_i-\mu^3)^2\right), \text{etc...}
\end{aligned}
$$

In R...

Let's look at fitting a simple normal model to $n = 30$ young rats whose weights were measured weekly for five weeks. Only the first week is considered here.

```
#################################################################################
# Weight of n=30 rats in grams; ll is log posterior distribution for model given in notes
#################################################################################
ll=function(mean,precision,data,a,b,m,t){
 ll=0
 for(i in 1:length(data)){ll=ll+dnorm(data[i],mean,1/sqrt(precision),log=TRUE)}
 ll=ll+dnorm(mean,m,1/sqrt(t),log=TRUE)
 ll=ll+dgamma(precision,a,b,log=TRUE)
}

weight=c(151,145,147,155,135,159,141,159,177,134,160,143,154,171,163,160,142,156,157,
         152,154,139,146,157,132,160,169,157,137,153)
MCtotal=1000
mean=1:MCtotal; precision=1:MCtotal # store MCtotal MCMC iterates here
```

Random walk proposal M-H: R code

```
#############################################################################
# Metropolis-Hastings, random walk proposal for (mean,precision)
#############################################################################
mean[1]=mean(weight); precision[1]=1/var(weight) # starting values
a=0.01; b=0.01; m=150; t=0.1                     # prior values
for(i in 2:MCtotal){
 mean.star=rnorm(1,mean[i-1],1); prec.star=rnorm(1,precision[i-1],0.001)
 rho=exp(ll(mean.star,prec.star,weight,a,b,m,t)-ll(mean[i-1],precision[i-1],weight,a,b,m,t))
 u=runif(1)
 if(u<=rho){ mean[i]=mean.star; precision[i]=prec.star }
 if(u>rho) { mean[i]=mean[i-1]; precision[i]=precision[i-1] }
}

sum(mean[2:MCtotal]-mean[1:(MCtotal-1)]!=0)/(MCtotal-1) # accept rate for (mu,tau)
plot(mean,precision) # Monte Carlo estimate of posterior
quantile(mean,c(0.025,0.5,0.975)) # 95% CI and posterior median mu
quantile(precision,c(0.025,0.5,0.975)) # 95% CI and posterior median tau
```

Independence proposal M-H: R code

```
#############################################################################
# Metropolis-Hastings, independence proposal for (mean,precision)
#############################################################################
mean[1]=mean(weight); precision[1]=1/var(weight) # starting values
a=0.01; b=0.01; m=150; t=0.1                     # prior values
for(i in 2:MCtotal){
 mean.star=rnorm(1,152,2); prec.star=rnorm(1,0.008,0.002)
 rho=exp(ll(mean.star,prec.star,weight,a,b,m,t)-ll(mean[i-1],precision[i-1],weight,a,b,m,t)
     -dnorm(mean.star,152,2,log=TRUE)+dnorm(mean[i-1],152,2,log=TRUE)
     -dnorm(prec.star,0.008,0.002,log=TRUE)+dnorm(precision[i-1],0.008,0.002,log=TRUE))
 u=runif(1)
 if(u<=rho){ mean[i]=mean.star; precision[i]=prec.star }
 if(u>rho) { mean[i]=mean[i-1]; precision[i]=precision[i-1] }
}

sum(mean[2:MCtotal]-mean[1:(MCtotal-1)]!=0)/(MCtotal-1) # accept rate for (mu,tau)
plot(mean,precision) # Monte Carlo estimate of posterior
quantile(mean,c(0.025,0.5,0.975)) # 95% CI and posterior median mu
quantile(precision,c(0.025,0.5,0.975)) # 95% CI and posterior median tau
```

Gibbs sampling: R code

```
#############################################################################
# Gibbs sampler for (mean,precision)
#############################################################################
mean[1]=mean(weight); precision[1]=1/var(weight) # starting values
a=0.01; b=0.01; m=150; t=0.1; n=length(data)     # prior values
for(i in 2:MCtotal){
 mu.var=1/(n*precision[i-1]+t)
 mu.mean=mu.var*(n*precision[i-1]*mean(weight)+m*t)
 mean[i]=rnorm(1,mu.mean,sqrt(mu.var))
 precision[i]=rgamma(1,a+0.5*n,b+0.5*sum((weight-mean[i])**2))
}

sum(mean[2:MCtotal]-mean[1:(MCtotal-1)]!=0)/(MCtotal-1) # accept rate for (mu,tau)
plot(mean,precision) # Monte Carlo estimate of posterior
quantile(mean,c(0.025,0.5,0.975)) # 95% CI and posterior median mu
quantile(precision,c(0.025,0.5,0.975)) # 95% CI and posterior median tau
```

WinBUGS and OpenBUGS

- WinBUGS is a free, Windows-based platform for MCMC inference in general models; it is no longer being updated, but still available.

WinBUGS and OpenBUGS

- WinBUGS is a free, Windows-based platform for MCMC inference in general models; it is no longer being updated, but still available.
- OpenBUGS is an open-source version of BUGS, which runs on Windows machines, Unix/Linux, or Macintosh. See http://www.openbugs.info/w/FrontPage.

WinBUGS and OpenBUGS

- WinBUGS is a free, Windows-based platform for MCMC inference in general models; it is no longer being updated, but still available.
- OpenBUGS is an open-source version of BUGS, which runs on Windows machines, Unix/Linux, or Macintosh. See http://www.openbugs.info/w/FrontPage.
- Note that OpenBUGS can be called from within R or SAS.

WinBUGS and OpenBUGS

- WinBUGS is a free, Windows-based platform for MCMC inference in general models; it is no longer being updated, but still available.
- OpenBUGS is an open-source version of BUGS, which runs on Windows machines, Unix/Linux, or Macintosh. See http://www.openbugs.info/w/FrontPage.
- Note that OpenBUGS can be called from within R or SAS.
- The rest of the slides will focus on WinBUGS, as it is slightly easier to use initially

WinBUGS and OpenBUGS

- WinBUGS is a free, Windows-based platform for MCMC inference in general models; it is no longer being updated, but still available.
- OpenBUGS is an open-source version of BUGS, which runs on Windows machines, Unix/Linux, or Macintosh. See http://www.openbugs.info/w/FrontPage.
- Note that OpenBUGS can be called from within R or SAS.
- The rest of the slides will focus on WinBUGS, as it is slightly easier to use initially
- WinBUGS suffers in terms of analyzing the output though. Complex analyses require imputting WinBUGS output into R or SAS anyway.

WinBUGS and OpenBUGS

- WinBUGS is a free, Windows-based platform for MCMC inference in general models; it is no longer being updated, but still available.
- OpenBUGS is an open-source version of BUGS, which runs on Windows machines, Unix/Linux, or Macintosh. See http://www.openbugs.info/w/FrontPage.
- Note that OpenBUGS can be called from within R or SAS.
- The rest of the slides will focus on WinBUGS, as it is slightly easier to use initially
- WinBUGS suffers in terms of analyzing the output though. Complex analyses require imputting WinBUGS output into R or SAS anyway.
- After today, all examples will be almost exclusively in DPpackage, but it's good to know about BUGS, BayesX, etc.

How does WinBUGS sample the posterior?

WinBUGS uses Gibbs sampling exclusively. This has good and bad points.

- Good point: generating samples is more or less automatic.

How does WinBUGS sample the posterior?

WinBUGS uses Gibbs sampling exclusively. This has good and bad points.

- Good point: generating samples is more or less automatic.
- Bad point: high correlation among elements of $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)'$ in the posterior makes Gibbs sampling really inefficient.

How does WinBUGS sample the posterior?

WinBUGS uses Gibbs sampling exclusively. This has good and bad points.

- Good point: generating samples is more or less automatic.
- Bad point: high correlation among elements of $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)'$ in the posterior makes Gibbs sampling really inefficient.
- Good point: By construction, almost all Gibbs samplers are $p(\boldsymbol{\theta}|\mathbf{y})$-irreducible, aperiodic, and positive Harris recurrent (Tierney, 1994). This means they're ergodic and all ergodic theorems apply (LLN, convergence of quantiles, etc.).

How does WinBUGS sample the posterior?

WinBUGS uses Gibbs sampling exclusively. This has good and bad points.

- Good point: generating samples is more or less automatic.
- Bad point: high correlation among elements of $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)'$ in the posterior makes Gibbs sampling really inefficient.
- Good point: By construction, almost all Gibbs samplers are $p(\boldsymbol{\theta}|\mathbf{y})$-irreducible, aperiodic, and positive Harris recurrent (Tierney, 1994). This means they're ergodic and all ergodic theorems apply (LLN, convergence of quantiles, etc.).
- Bad point: there are precious few practical ways to improve mixing in a Gibbs sampler.

“Intractable” full conditionals

Not all full conditionals are “recognizable” as Γ, normal, uniform, etc. Alternatives in order of preference for WinBUGS:

- Adaptive rejection sampling (if full conditional is log-concave). (Gilks and Wild, 1992).

“Intractable” full conditionals

Not all full conditionals are “recognizable” as Γ, normal, uniform, etc. Alternatives in order of preference for WinBUGS:

- Adaptive rejection sampling (if full conditional is log-concave). (Gilks and Wild, 1992).
- Slice sampling. (Neal, 2004). Restricted range, used with censored data.

“Intractable” full conditionals

Not all full conditionals are “recognizable” as Γ, normal, uniform, etc. Alternatives in order of preference for WinBUGS:

- Adaptive rejection sampling (if full conditional is log-concave). (Gilks and Wild, 1992).
- Slice sampling. (Neal, 2004). Restricted range, used with censored data.
- Metropolis step. (Tierney, 1994). Unrestricted range.

WinBUGS code

WinBUGS code looks a lot like R or S-plus. Here's code for fitting a simple normal model to $n = 30$ young rats whose weights were measured weekly for five weeks. Only the first week is considered here.

```
model{
 for(i in 1:n){ y[i] ~ dnorm(mu, tau) }
 mu ~ dnorm(0,0.001)
 tau ~ dgamma(0.001,0.001)
}

list(mu=150, tau=0.1)  # starting values

list(y=c(151,145,147,155,135,159,141,159,177,134,160,143,
154,171,163,160,142,156,157,152,154,139,146,157,
132,160,169,157,137,153), n=30)  # data
```

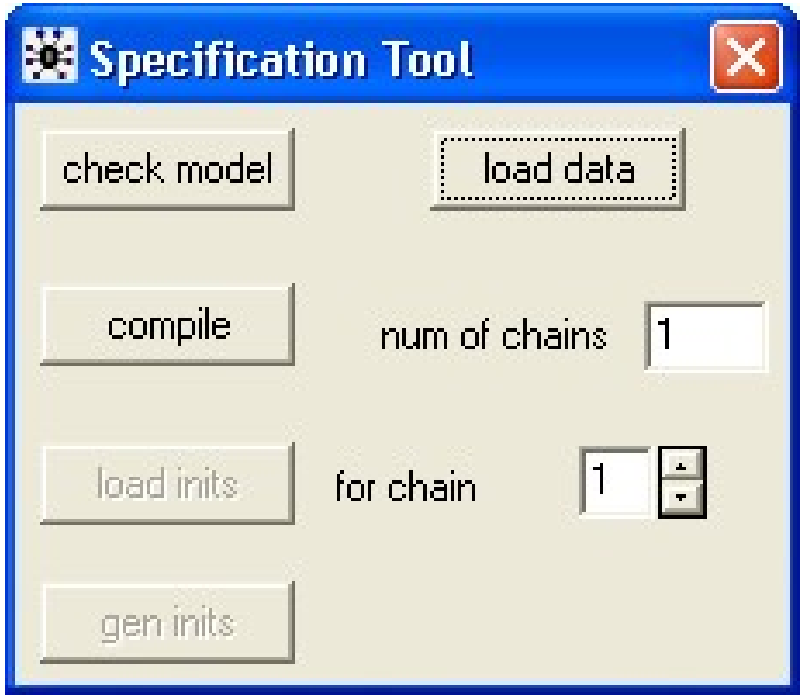
- The `model` statement specifies the full probability model. No improper priors are allowed.

- The `model` statement specifies the full probability model. No improper priors are allowed.
- After the model specification, `list(mu=150, tau=0.1)` specifies the starting values of the Markov chain ($\mu^0 = 150, \tau^0 = 0.1$). Starting values should be in a region of non-negligible posterior support. More on this later.

- The `model` statement specifies the full probability model. No improper priors are allowed.
- After the model specification, `list(mu=150, tau=0.1)` specifies the starting values of the Markov chain ($\mu^0 = 150, \tau^0 = 0.1$). Starting values should be in a region of non-negligible posterior support. More on this later.
- Finally, the data is `list`ed. Data can also be listed one or more columns. Again, more on this later.

WinBUGS tools:

There are three main tools you will use frequently. The winBUGS manual explains the use of each in detail. Under **Model** in the toolbar, pick **Specification...** and the **Specification Tool** appears:



- To make sure the model is at least *syntactically* correctly specified, double click on `model` in the WinBUGS code (it will then become highlighted), then click check model in the **Specification Tool**.

- To make sure the model is at least *syntactically* correctly specified, double click on `model` in the WinBUGS code (it will then become highlighted), then click check model in the **Specification Tool**.
- If the model is okay, WinBUGS will tell you that "the model is syntactically correct" in the lower left hand corner, otherwise you'll get an error message.

- To make sure the model is at least *syntactically* correctly specified, double click on `model` in the WinBUGS code (it will then become highlighted), then click check model in the **Specification Tool**.
- If the model is okay, WinBUGS will tell you that "the model is syntactically correct" in the lower left hand corner, otherwise you'll get an error message.
- If the model is okay, next double click on the `list` that holds the data and click load data. WinBUGS will tell you "data loaded" if there were no problems, otherwise you'll find out something is not right with your `data` list or perhaps your `model`.

- To make sure the model is at least *syntactically* correctly specified, double click on `model` in the WinBUGS code (it will then become highlighted), then click check model in the **Specification Tool**.
- If the model is okay, WinBUGS will tell you that "the model is syntactically correct" in the lower left hand corner, otherwise you'll get an error message.
- If the model is okay, next double click on the `list` that holds the data and click load data. WinBUGS will tell you "data loaded" if there were no problems, otherwise you'll find out something is not right with your `data` list or perhaps your `model`.
- Click compile.

Starting values

- You can either provide intelligent starting values for the Markov chain or let WinBUGS generate them from the prior. If the prior is vague, the former is a good idea. Double click on the `list` that holds the starting values, then click load inits. You are ready to generate samples!

Starting values

- You can either provide intelligent starting values for the Markov chain or let WinBUGS generate them from the prior. If the prior is vague, the former is a good idea. Double click on the `list` that holds the starting values, then click load inits. You are ready to generate samples!
- Note: you can also provide *partial* starting values, for example just the mean value in this example. This is handy in random effects models where there may be several hundred random effects. Click gen inits to simulate initial values for remaining parameters.

Update and Sample Monitor tools

- Under **Model** choose **Update...** to bring up the **Update Tool**, and under **Inference** choose **Samples...** to bring up the **Sample Monitor Tool**.

Update and Sample Monitor tools

- Under **Model** choose **Update...** to bring up the **Update Tool**, and under **Inference** choose **Samples...** to bring up the **Sample Monitor Tool**.
- The **Sample Monitor Tool**: In the slot by **node** enter "`mu`", then click on the set button, then enter "`tau`" and click on the set button again. This slot tells WinBUGS what quantities you want to keep track of in your analysis. To tell WinBUGS you are done entering nodes, put an asterisk in the slot. The window then looks like:

Update and Sample Monitor tools

- Under **Model** choose **Update...** to bring up the **Update Tool**, and under **Inference** choose **Samples...** to bring up the **Sample Monitor Tool**.
- The **Sample Monitor Tool**: In the slot by **node** enter "`mu`", then click on the set button, then enter "`tau`" and click on the set button again. This slot tells WinBUGS what quantities you want to keep track of in your analysis. To tell WinBUGS you are done entering nodes, put an asterisk in the slot. The window then looks like:

- The path of the Markov chain can be monitored by clicking on trace button.

The **Update tool**

- Click the update button in the **Update Tool**. The trace plot will dynamically show the actual MCMC iterates being generated with each click of update in the **Update Tool**.

  Click update several times. Be careful, this can be quite hypnotic. Update the Markov chain until 11000 iterates are generated, i.e. until 11000 is in the **iteration** box. You can update more or less than 1000 at a time by typing a different value in the **updates** box.

- A *burn in* value can be specified in the **beg** box of the **Sample Monitor Tool**. Type “1001” in this box. We will discuss the notion of burn in carefully later, but roughly, we are throwing out the first 1000 iterates to eliminate any dependence posterior inferences might have on our starting values.

- A *burn in* value can be specified in the **beg** box of the **Sample Monitor Tool**. Type “1001” in this box. We will discuss the notion of burn in carefully later, but roughly, we are throwing out the first 1000 iterates to eliminate any dependence posterior inferences might have on our starting values.
- Go back to the **Sample Monitor Tool** and click on stats to get your posterior estimates.

Comments:

- Under **Help** you can find **Examples Vol I** and **Examples Vol II**. These examples are an excellent source of models, data, and ideas.

Comments:

- Under **Help** you can find **Examples Vol I** and **Examples Vol II**. These examples are an excellent source of models, data, and ideas.
- The error codes in WinBUGS are mysterious. A few are explained in the manual. The most common errors have to do with defining nodes twice and when WinBUGS has trouble sampling, typically with censored data.

Comments:

- Under **Help** you can find **Examples Vol I** and **Examples Vol II**. These examples are an excellent source of models, data, and ideas.
- The error codes in WinBUGS are mysterious. A few are explained in the manual. The most common errors have to do with defining nodes twice and when WinBUGS has trouble sampling, typically with censored data.
- WinBUGS 1.4 now gives the user some leeway in "tweaking" M-H burn-in values, allows for blocked updates, and also allows choosing update options. This can improve convergence dramatically.

et cetera...

- Multiple chains.

et cetera...

- Multiple chains.
- history gives entire iteration history for all nodes being monitored. Good for assessing burn-in and convergence to posterior.

et cetera...

- Multiple chains.
- history gives entire iteration history for all nodes being monitored. Good for assessing burn-in and convergence to posterior.
- density gives (marginal) posterior kernel-smoothed density estimates.

et cetera...

- Multiple chains.
- history gives entire iteration history for all nodes being monitored. Good for assessing burn-in and convergence to posterior.
- density gives (marginal) posterior kernel-smoothed density estimates.
- stats gives posterior summary statistics.

et cetera...

- Multiple chains.
- history gives entire iteration history for all nodes being monitored. Good for assessing burn-in and convergence to posterior.
- density gives (marginal) posterior kernel-smoothed density estimates.
- stats gives posterior summary statistics.
- coda gives the actual Gibbs iterates for use in other programs.

et cetera...

- Multiple chains.
- `history` gives entire iteration history for all nodes being monitored. Good for assessing burn-in and convergence to posterior.
- `density` gives (marginal) posterior kernel-smoothed density estimates.
- `stats` gives posterior summary statistics.
- `coda` gives the actual Gibbs iterates for use in other programs.
- `quantiles` gives “running quantiles” useful for assessing burn-in and convergence.

et cetera...

- bgr daig gives a convergence diagnostic based on running several chains simultaneously from different starting values.

et cetera...

- bgr daig gives a convergence diagnostic based on running several chains simultaneously from different starting values.
- auto cor gives the ACF for each node. Useful for determining how well the chain is mixing and/or determining thinning values.

et cetera...

- bgr daig gives a convergence diagnostic based on running several chains simultaneously from different starting values.
- auto cor gives the ACF for each node. Useful for determining how well the chain is mixing and/or determining thinning values.
- **Inference** then **DIC...** ultimately gives the *deviance information driterion* for a model. Basically a measure of how "complex" a model is along with an overall measure of fit. This statistic can be informally used to compare models and provides a measure of model fit penalized by complexity much in the same manner as the AIC or BIC.