STAT 576 Bayesian Analysis

Lecture 8: Markov Chain Simulation

Chencheng Cai

Washington State University

Backgrounds on Markov Chain

Let $\{X_t\}$ be a sequence of random variables. We say $\{X_t\}$ is a **Markov chain** if

$$P(X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0) = P(X_{t+1} = x_{t+1} | X_t = x_t).$$

In other words, the future state of the chain depends only on the current state, not on the past states.

Remarks:

- If $t \in \mathbb{Z}^+$, then $\{X_t\}$ is a **discrete-time** Markov chain.
- If $t \in \mathbb{R}^+$, then $\{X_t\}$ is a **continuous-time** Markov chain.
- For discrete-time Markov chain, we can define the **transition probability** $P_{ij,t} = P(X_{t+1} = j | X_t = i)$.
- For continuous-time Markov chain, we can define the **transition rate** $q_{ij,t} = \lim_{s \downarrow t} \frac{P(X_s = j | X_t = i) - \delta_{ij}}{s - t}$.

For now, we will focus on discrete-time Markov chain.

Backgrounds on Markov Chain

For discrete-time Markov chain with transition probability

$$P_{ij,t} = P(X_{t+1} = j | X_t = i),$$

- A Markov chain is **(time-)homogeneous** if the transition probabilities do not depend on t. Otherwise it is **(time-)inhomogeneous**.
- The **state space** of the chain is the set of all possible values of X_t.
- For a finite state space, we represent the transition probabilities in a matrix form, known as **the transition matrix**.
- For a measurable state space, we call it a **Markov chain on a measurable state space**. And we define the **transition kernel**

$$P(x, A) = P(X_{t+1} \in A | X_t = x), \quad x \in \mathcal{X}, A \in \mathcal{B}(\mathcal{X}).$$

For now, we focus on homogenoeous Markov chains with finite state space.

Properties of Markov Chain

- A state i is **accessible** from state j if there exists $n \geq 0$ such that $P(X_n = i | X_0 = j) > 0$. We write $j \to i$.
- A state i is **communicating** with state j if $i \to j$ and $j \to i$. We write $i \leftrightarrow j$.
- A Markov chain is **irreducible** if any two states are communicating.
- The **period** of a state i is the greatest common divisor of all $n \geq 1$ such that $P(X_n = i | X_0 = i) > 0$. If the period is 1, then the state is **aperiodic**.
- A state i is **transient** if there is a non-zero probability that the chain will never return to i. Otherwise, it is **recurrent**.
- A state i is **positive-recurrent** if the expected return time to i is finite. Otherwise, it is **null-recurrent**.
- Periodicity, transience, recurrence and postive-recurrence are class properties.

Stationary Distribution of a Markov Chain

- A distribution π on the state space is **stationary** for a Markov chain with transition matrix P if
$$\pi = \pi P.$$
That is $\pi(i) = \sum_j \pi(j) P_{ji}$ for all i.
- **Existence**:
Every positive recurrent Markov chain has a unique stationary distribution.
- **Uniquess**:
The stationary distribution is unique if the chain is irreducible.
- **Covergence theorem**:
If the chain is irreducible and aperiodic with stationary π, then there exist constants $0 < \alpha < 1$ and $C > 0$ such that
$$\max_x \ \|P^t(x, \cdot) - \pi\|_{TV} \leq C\alpha^t.$$

Ergodic Theorem

- A state i is called **ergodic** if it is positive recurrent and aperiodic.
- Let $V_i(n) = \sum_{t=0}^{n-1} I(X_t = i)$ be the number of visits to state i in the first n steps.
- **Ergodic Theorem**:
  For any irreducible and positive recurrent Markov chain with stationary distribution π, we have
  $$\frac{V_i(n)}{n} \to \pi(i), \quad \text{a.s.}$$

Remarks:

- The convergence theorem indicates the marginal distribution at a future time is close to the stationary distribution (in terms of replications of the chain).
- The ergodic theorem indicates the partial sample from a long chain is close to the stationary distribution.

Detailed Balance

- A distribution π is **detailed balanced** for a Markov chain with transition matrix P if
$$\pi(i)P_{ij} = \pi(j)P_{ji}, \quad \forall i, j.$$
- A detailed balanced distribution is a stationary distribution. (Not vice versa!)
- Example: unique stationary distribution that is not detailed balanced.
Consider a Markov chain with state space $\{1, 2, 3\}$ and transition matrix
$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$
The stationary distribution is $\pi = (1/3, 1/3, 1/3)$, but it is not detailed balanced.

Markov Chain Simulation

- We generate a Markov chain with **designed** transition matrix P such that the stationary distribution of the chain is the target distribution.
- We treat the values of the markov chain as samples from the target distribution.
- The chain is generated by the following steps:
  - Start from an initial state X_0.
  - At each step t, generate X_{t+1} from the conditional distribution $P(X_{t+1}|X_t)$.
  - Repeat the above step for n steps.
  - The samples $\{X_0, X_1, \ldots, X_n\}$ are the samples from the target distribution.

The Metropolis Algorithm

Assume the target distribution is $\pi(x)$. The Metropolis algorithm generates a Markov chain with the following steps:

- Start from an initial state X_0.
- At each step t, generate a candidate state Y from a **symmetric proposal** distribution $q(x, y)$.
- Compute the acceptance probability

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}.$$

- Generate X_{t+1} by

$$X_{t+1} = \begin{cases} Y & \text{with probability } \alpha(x, y), \\ X_t & \text{with probability } 1 - \alpha(x, y). \end{cases}$$

- Repeat the above step for n steps.

The Metropolis Algorithm

Justification on the stationary distribution:

- The transition probability from x to y is

$$P(x, y) = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}$$

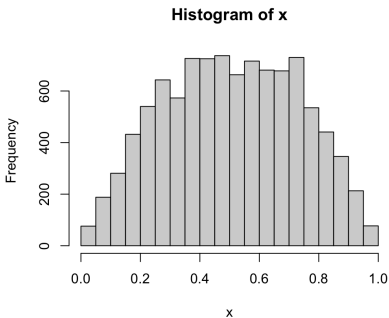
- The target distribution satisfies the detailed balance condition:

$$\pi(x)P(x, y) = \pi(y)P(y, x).$$

Implementation in R

Example: draw samples from $\mathrm{Beta}(2, 2)$ distribution.

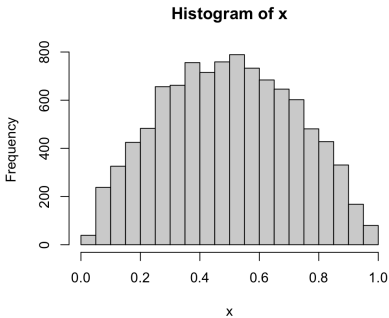
```
n = 10000
x = rep(0, n+1)
x[1] = 0.5
for(i in 1:n){
    y = x[i] + rnorm(1) * 0.5
    a = dbeta(y, 2, 2) / dbeta(x[i],
        2, 2)
    if (runif(1) <= a) x[i+1] = y
    else x[i+1] = x[i]
}
```



Implementation in R

Update: Use random starting point.

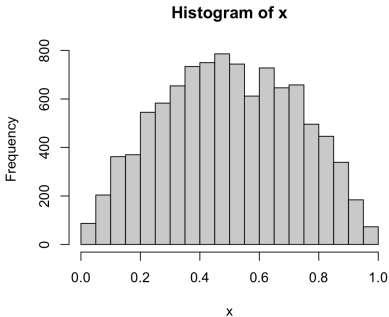
```
n = 10000
x = rep(0, n+1)
x[1] = runif(1)
for(i in 1:n){
    y = x[i] + rnorm(1) * 0.5
    a = dbeta(y, 2, 2) / dbeta(x[i],
        2, 2)
    if (runif(1) <= a) x[i+1] = y
    else x[i+1] = x[i]
}
```



Implementation in R

Update: Add burnin period to get samples under convergence.

```
n = 10000
burnin = 1000
x = rep(0, n+1)
x0 = 0.5
for (i in 1:burnin){
    y = x0 + rnorm(1) * 0.5
    a = dbeta(y, 2, 2) / dbeta(x0,
        2, 2)
    if (runif(1) <= a) x0 = y
}
x[1] = x0
for(i in 1:n){
    y = x[i] + rnorm(1) * 0.5
    a = dbeta(y, 2, 2) / dbeta(x[i],
         2, 2)
    if (runif(1) <= a) x[i+1] = y
    else x[i+1] = x[i]
}
```



Implementation in R

Update: Encapsulate the code into a function.

```
metropolis <- function(target, n, burnin, proposal, initial){
    sample = rep(0, n)
    x = initial()
    for(i in 1:(n+burnin)){
        y = proposal(x)
        a = target(y) / target(x)
        if(runif(1) <= a) x = y
        if(i>burnin)
            sample[i-burnin] = x
    }
    return(sample)
}

target = function(x){dbeta(x, 2, 2)}
proposal = function(x){x + rnorm(1)*0.5}
initial = function(){runif(1)}
s = metropolis(target, n, burnin, proposal, initial)
```

Implementation in R

Update: Use log density for higher precision.

```
metropolis.log <- function(log.target, n, burnin, proposal, initial){
    sample = rep(0, n)
    x = initial()
    for(i in 1:(n+burnin)){
        y = proposal(x)
        du = log.target(y) - log.target(x)
        if(runif(1) <= exp(du)) x = y
        if(i>burnin) sample[i-burnin] = x
    }
    return(sample)
}

log.target = function(x){dbeta(x, 2, 2, log=T)}
proposal = function(x){x + rnorm(1)*0.5}
initial = function(){runif(1)}
s = metropolis.log(log.target, n, burnin, proposal, initial)
```

Implementation in R

Update: Allow parallel sampling with multiple chains

```
metropolis <- function(log.target, n, burnin, proposal, initial, n.chain)
    {
    sample = array(dim=c(n.chain, n))
    x = initial(n.chain)
    for(i in 1:(n+burnin)){
        y = proposal(x)
        du = log.target(y) - log.target(x)
        accept = runif(n.chain) <= exp(du)
        x[accept] = y[accept]
        if(i>burnin) sample[,i-burnin] = x
    }
    return(sample)
}

log.target = function(x){dbeta(x, 2, 2, log=T)}
proposal = function(x){x + rnorm(length(x))*0.5}
initial = function(n){runif(n)}
s = metropolis(log.target, n, burnin, proposal, initial, 4)
```

Perofrmance of Simulating Mutliple Chain

```
for(n.chain in 0:6){
    start = Sys.time()
    for(i in 1:100)
        s = metropolis(log.target, n, burnin, proposal, initial,
                       2**n.chain)
    end = Sys.time()
    print(c(2 ** n.chain, (end-start)/100))
}
```

- Resutls on my laptop, M1 Pro (8-core CPU):

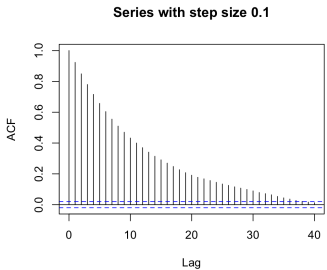
n.chain	1	2	4	8	16	32	64
time (ms)	53	58	65	79	107	173	298

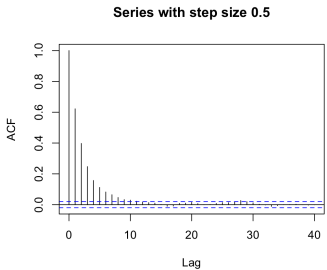
- Results on my desktop, 12900K (16-core CPU):

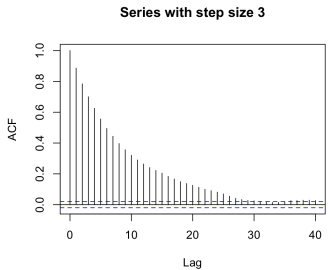
n.chain	1	2	4	8	16	32	64
time (ms)	34	40	44	52	69	106	176

Samples from the Metropolis Algorithm

Samples from the Metropolis algorithm are correlated.







- A more local proposal distribution increases the autocorrelation.
- A more global proposal distribution resulting lower jumps increases the autocorrelation as well.
- One solution is to have asymmetric proposal distribution.

The Metropolis-Hastings Algorithm

Assume the target distribution is $\pi(x)$. The Metropolis-Hastings algorithm generates a Markov chain with the following steps:

- Start from an initial state X_0.
- At each step t, generate a candidate state Y from a **proposal** distribution $q(x, y)$.
- Compute the acceptance probability

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right\}.$$

- Generate X_{t+1} by

$$X_{t+1} = \begin{cases} Y & \text{with probability } \alpha(x, y), \\ X_t & \text{with probability } 1 - \alpha(x, y). \end{cases}$$

- Repeat the above step for n steps.

Implementation in R

```
metropolis.hastings <- function(log.target, n, burnin, proposal, initial,
     n.chain){
    sample = array(dim=c(n.chain, n))
    x = initial(n.chain)
    for(i in 1:(n+burnin)){
        prop = proposal(x)
        du = log.target(prop$y) - log.target(x)
        accept = runif(n.chain) <= exp(du+prop$dlog)
        x[accept] = prop$y[accept]
        if(i>burnin) sample[,i-burnin] = x
    }
    return(sample)
}
proposal = function(x){
    y = (0.5+x)/2 + rnorm(length(x))*0.5
    dlog = dnorm(x, (0.5+y)/2, 0.25, log=T) - dnorm(y, (0.5+x)/2, 0.25,
        log=T)
    return(list(y=y, dlog=dlog))
}
s = metropolis.hastings(log.target, n, burnin, proposal, initial, 1)
```

Comparison of Metropolis vs Metropolis-Hastings

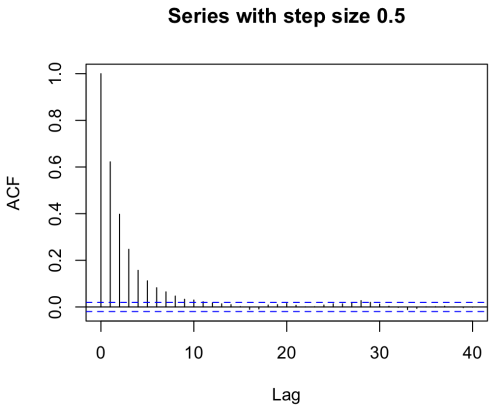


Figure: $q(x, y) = \mathcal{N}(x, 0.5^2)$

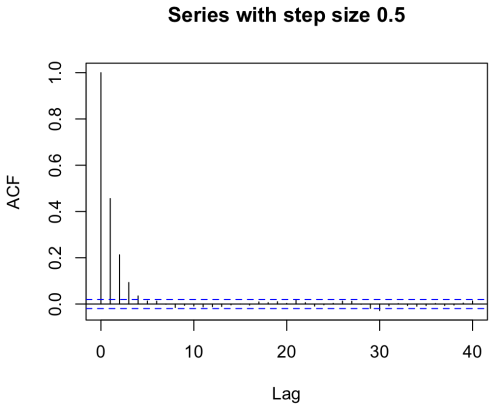


Figure: $q(x, y) = \mathcal{N}\left(\frac{x+0.5}{2}, 0.5^2\right)$

Some Special Cases

- **Random Walk Metropolis**:
  The proposal step is
  $$y = x + \sigma\epsilon,$$
  for some spherical symmetric random variable ϵ.

- **Independence Sampler**:
  The proposal step is
  $$y = q(x),$$
  where q is a proposal distribution.

Gibbs Sampler

- The Gibbs sampler is a special case of the Metropolis-Hastings algorithm.
- Suppose the variable X is partitioned into $(X_1, X_2, \ldots, X_d)$.
- The Gibbs sampler makes the Metropolis-Hastings move to update each component X_i in turn.
- **Random-Scan Gibbs Sampler**:
  At iteration $t+1$, we update X by
  - Choose i at random from $\{1, 2, \ldots, d\}$.
  - Draw
    $$X_i^{(t+1)} \sim \pi(X_i | X_1^{(t)}, X_2^{(t)}, \ldots, X_{i-1}^{(t)}, X_{i+1}^{(t)}, \ldots, X_d^{(t)}).$$
  - Update $X_{[-i]}^{(t+1)} = X_{[-i]}^{(t)}$.
- **Systematic-Scan Gibbs Sampler**:
  At iteration $t+1$, we update X by
  - For $i = 1, 2, \ldots, d$, draw

$$X_i^{(t+1)} \sim \pi(X_i | X_1^{(t+1)}, X_2^{(t+1)}, \ldots, X_{i-1}^{(t+1)}, X_{i+1}^{(t)}, \ldots, X_d^{(t)}).$$

Justifications for Gibbs Sampler

Consider the step:

- Draw $X_i^{(t+1)} \sim \pi(X_i | X_{[-i]}^{(t)})$

The proposal transition kernel from $X^{(t)} = (X_{[-i]}^{(t)}, X_i^{(t)})$ to $X^{(t+1)} = (X_{[-i]}^{(t)}, X_i^{(t+1)})$ is

$$q(X^{(t)}, X^{(t+1)}), (X_{[-i]}^{(t)}, X_i^{(t+1)})) = \pi(X_i^{(t+1)} | X_{[-i]}^{(t)}).$$

The acceptance probability from the Metropolis-Hastings algorithm is

$$\begin{aligned}
\alpha &= \min \left\{ 1, \frac{\pi(X^{(t+1)}) q(X^{(t+1)}, X^{(t)})}{\pi(X^{(t)}) q(X^{(t)}, X^{(t+1)})} \right\} \\
&= \min \left\{ 1, \frac{\pi(X_{[-i]}^{(t)}) \pi(X_i^{(t+1)} | X_{[-i]}^{(t)}) \pi(X_i^{(t)} | X_{[-i]}^{(t)})}{\pi(X_{[-i]}^{(t)}) \pi(X_i^{(t)} | X_{[-i]}^{(t)}) \pi(X_i^{(t+1)} | X_{[-i]}^{(t)})} \right\} = 1
\end{aligned}$$

Example: Bivariate Normal

Draw samples from the bivariate normal distribution:

$$\mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right)$$

The target density function is

$$\pi(x_1, x_2) \propto \exp\left\{-\frac{1}{2}\begin{pmatrix}x_1\\x_2\end{pmatrix}^T \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}^{-1} \begin{pmatrix}x_1\\x_2\end{pmatrix}\right\} = \exp\left\{-\frac{x_1^2 + x_2^2 - 2\rho x_1 x_2}{2(1-\rho^2)}\right\}$$

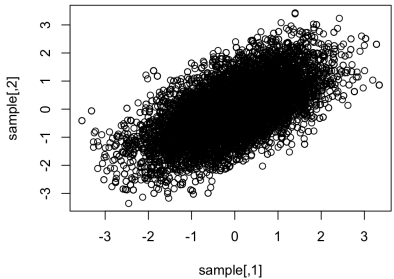
We can run a random-walk Metropolis-Hastings algorithm:

- Proposal: $y = x + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$.
- Accept probability:

$$\alpha = \min\left\{1, \exp\left\{-\frac{y_1^2 + y_2^2 - 2\rho y_1 y_2 - x_1^2 - x_2^2 + 2\rho x_1 x_2}{1-\rho^2}\right\}\right\}$$

Example: Bivariate Normal

```
n = 10000
burnin = 5000
rho=0.6
sigma = matrix(c(1, rho, rho, 1), nrow=2)

sample = array(dim=c(n, 2))
x = c(0, 0)
for(i in 1:(n+burnin)){
    y = x + rnorm(2) * 0.5
    du = -1/2 * (t(y)%*%solve(sigma)%*%y - t
        (x)%*%solve(sigma)%*%x)
    if(runif(1) <= exp(du)) x = y
    if(i>burnin) sample[i-burnin,] = x
}
```



Example: Bivariate Normal

Draw samples from the bivariate normal distribution:

$$\mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right)$$

The conditional distributions are given by

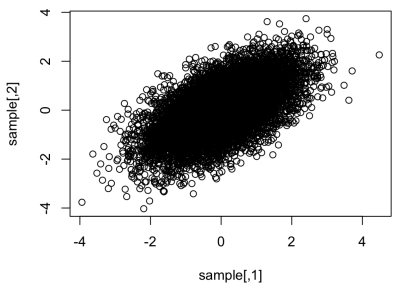
$$x_1 \mid x_2 \sim \mathcal{N}(\rho x_2, 1-\rho^2), \quad x_2 \mid x_1 \sim \mathcal{N}(\rho x_1, 1-\rho^2)$$

The Gibbs sampler udpate step:

- Draw $x_1^{(t+1)} \sim \mathcal{N}(\rho x_2^{(t)}, 1-\rho^2)$
- Draw $x_2^{(t+1)} \sim \mathcal{N}(\rho x_1^{(t+1)}, 1-\rho^2)$

Example: Bivariate Normal

```
sample = array(dim=c(n, 2))
x = c(0, 0)
for(i in 1:(n+burnin)){
    x[1] = rho*x[2] + rnorm(1) * sqrt(1-rho*
        *2)
    x[2] = rho*x[1] + rnorm(1) * sqrt(1-rho*
        *2)
    if(i>burnin) sample[i-burnin,] = x
}
```



Comparison

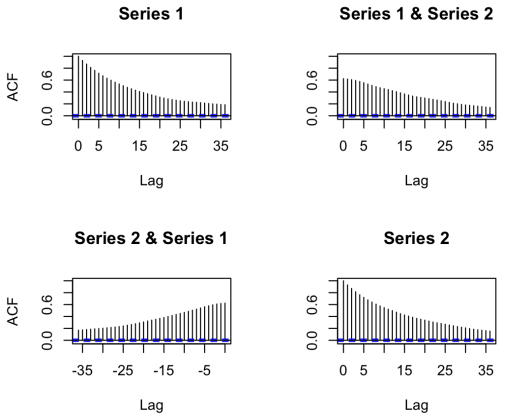


Figure: Random-Walk Metropolis

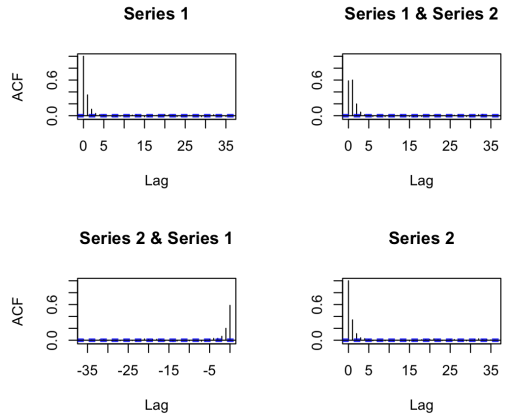


Figure: Gibbs Sampler

Issues

- Discard the first few samples (burnin) to ensure stationary.
- Use multiple chains to get independent samples.
- The effective sample size is related to the autocorrelation:

$$n_{\text{eff}} = \frac{nm}{1 + 2\sum_{k=1}^{\infty} \rho_k},$$

where

- m: number of chains,
- n: number of samples in each chain,
- ρ_k: autocorrelation at lag k.

Example: Hierarchical Model

Diet	Measurements
A	62, 60, 63, 59
B	63, 67, 71, 64, 65, 66
C	68, 66, 71, 67, 68, 68
D	56, 62, 60, 61, 63, 64, 63, 59

Table 11.2 *Coagulation time in seconds for blood drawn from 24 animals randomly allocated to four different diets. Different treatments have different numbers of observations because the randomization was unrestricted. From Box, Hunter, and Hunter (1978), who adjusted the data so that the averages are integers, a complication we ignore in our analysis.*

The hierarchical model is

$$\mu, \sigma, \tau \propto 1/\sigma$$
$$\theta_j \mid \mu, \tau \sim \mathcal{N}(\mu, \tau^2), i.i.d.$$
$$y_{ij} \mid \theta_j, \sigma \sim \mathcal{N}(\theta_j, \sigma^2), i.i.d.$$

Example: Hierarchical Model

Then the posterior is

$$p(\theta, \mu, \sigma, \tau) \propto \sigma^{-1} \prod_{j=1}^{4} \mathcal{N}(\theta_j \mid \mu, \tau^2) \prod_{j=1}^{J} \prod_{i=1}^{n_j} \mathcal{N}(y_{ij} \mid \theta_j, \sigma^2)$$

The conditional distributions:

- $\theta_j \mid \mu, \sigma, \tau, y \sim \mathcal{N}(\hat{\theta}_j, V_j)$ with

$$\hat{\theta}_j = V_j \left(\frac{\mu}{\tau^2} + \frac{n_j \bar{y}_j}{\sigma^2} \right), \quad V_j = \left(\frac{1}{\tau^2} + \frac{n_j}{\sigma^2} \right)^{-1}$$

- $\mu \mid \theta, \sigma, \tau, y \sim \mathcal{N}(\hat{\mu}, \tau^2/J)$ with $\hat{\mu} = \bar{\theta}_j$.
- $\sigma^2 \mid \theta, \mu, \tau, y \sim \text{Inv-}\chi^2(n, \hat{\sigma}^2)$ with $\hat{\sigma}^2 = \frac{1}{n} \sum_{j=1}^{J} \sum_{i=1}^{n_j} (y_{ij} - \theta_j)^2$.
- $\tau^2 \mid \theta, \mu, \sigma, y \sim \text{Inv-}\chi^2(J-1, \hat{\tau}^2)$ with $\hat{\tau}^2 = \frac{1}{J-1} \sum_{j=1}^{J} (\theta_j - \mu)^2$.

Example: Hierarchical Model

Load the dataset and pre-compute some constants.

```
library(faraway)
group = aggregate(coagulation$coag, by=list(Diet=coagulation$diet),
                  FUN=function(col){
                      c(n=length(col),
                      avg=mean(col),
                      sumx=sum(col),
                      sumxx=sum(col**2))
                  })$x

J = nrow(group)
sumxx = sum(group[, 'sumxx'])
sumx = group[, 'sumx']
nj = group[, 'n']
```

Example: Hierarchical Model

Prepare for sampling.

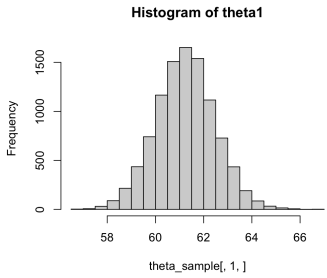
```
n = 1000
m = 10
burnin = 100

theta_sample=array(dim=c(m, J,n))
mu_sample = array(dim=c(m, n))
sigma2_sample = array(dim=c(m, n))
tau2_sample = array(dim=c(m, n))

mu = rnorm(m)
sigma2 = 1 / rgamma(m, 1)
tau2 = 1 / rgamma(m, 1)
theta = matrix(rnorm(J*m), nrow=m)
```

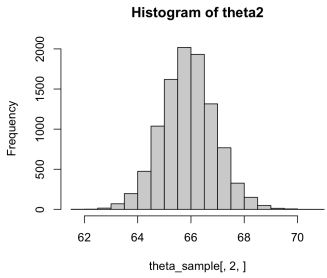
Example: Hierarchical Model

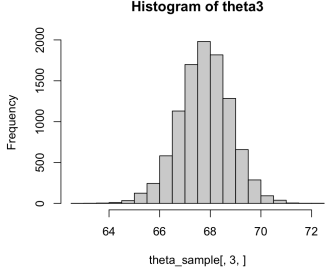
Gibbs sampler.

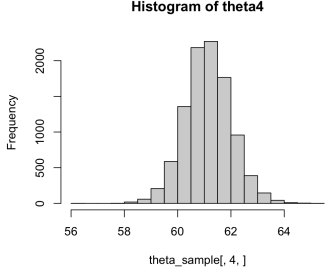
```
for(i in 1:(n+burnin)){
    V_j = 1/(1/tau2 + outer(1/sigma2, group[,'n']))
    theta_hat = V_j * (mu/tau2 + outer(1/sigma2, group[,'sumx']))
    theta = matrix(rnorm(J*m), nrow=m) * sqrt(V_j) + theta_hat
    mu = rowMeans(theta) + rnorm(m) * sqrt(tau2/J)
    sigma2_hat_n = (sumxx - 2*theta%*%sumx + (theta**2)%*%nj)[,1]
    sigma2 = sigma2_hat_n / rchisq(m, nn)
    tau2_hat_J = rowSums((theta - mu)**2)
    tau2 = tau2_hat_J / rchisq(m, J-1)
    if(i > burnin){
        theta_sample[,,i-burnin] = theta
        mu_sample[, i-burnin] = mu
        sigma2_sample[, i-burnin] = sigma2
        tau2_sample[, i-burnin] = tau2
    }
}
```

Example: Hierarchical Model

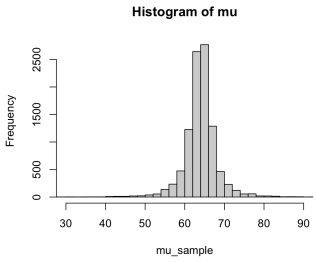


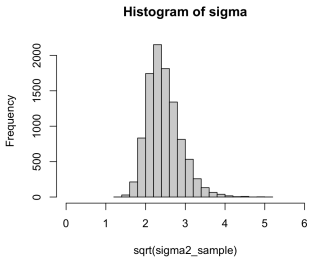


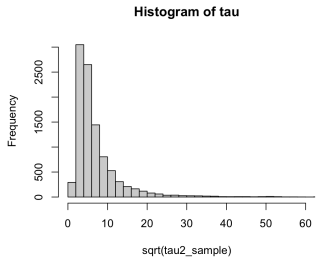




Example


Histogram of mu
Frequency
0
500
1500
2500
30
40
50
60
70
80
90
mu_sample


Histogram of sigma
Frequency
0
500
1000
1500
2000
0
1
2
3
4
5
6
sqrt(sigma2_sample)


Histogram of tau
Frequency
0
500
1500
2500
0
10
20
30
40
50
60
sqrt(tau2_sample)

Exampl: Hierarchical Model

Quantile	2.5%	25%	Median	75%	97.5%
θ_1	58.83	60.44	61.24	62.04	63.69
θ_2	63.94	65.24	65.89	66.54	67.94
θ_3	65.70	67.12	67.78	68.46	69.76
θ_4	59.43	60.58	61.13	61.71	62.91
μ	54.99	62.24	64.05	65.82	73.37
σ	1.806	2.171	2.403	2.699	3.426
τ	1.946	3.533	5.150	8.144	26.204

Data Augmentation Algorithm

Bayesian Missing Value Problem:

- Suppose we have a dataset Y with missing values.
- We denote the observed and missing values as Y_{obs} and Y_{mis}.
- We should have a tractable model for the joint distribution $p(Y_{\text{obs}}, Y_{\text{mis}} \mid \theta)$.
- The posterior is usually intractable:

$$p(\theta \mid Y_{\text{obs}}) = \int p(\theta \mid Y_{\text{obs}}, Y_{\text{mis}}) p(Y_{\text{mis}} \mid Y_{\text{obs}}) dY_{\text{mis}}.$$

- If we have samples from $p(Y_{\text{mis}} \mid Y_{\text{obs}})$, we can use the Monte Carlo average to approximate the integral.

Data Augmentation Algorithm

$$p(\theta \mid Y_{\text{obs}}) = \int p(\theta \mid Y_{\text{obs}}, Y_{\text{mis}}) p(Y_{\text{mis}} \mid Y_{\text{obs}}) dY_{\text{mis}}.$$

- Let $g(\theta)$ be an approximation for $p(\theta \mid Y_{\text{obs}})$.
- Draw m copies of $y_{\text{mis}}^{(1)}, \ldots, y_{\text{mis}}^{(m)}$ from

$$\tilde{p}(Y_{\text{mis}}) = \int p(Y_{\text{mis}} \mid Y_{\text{obs}}, \theta) g(\theta) d\theta.$$

  - How: first draw $\theta^{(i)}$ from $g(\theta)$, then draw $y_{\text{mis}}^{(i)}$ from $p(Y_{\text{mis}} \mid Y_{\text{obs}}, \theta^{(i)})$.
- Update the approximation for $p(\theta \mid Y_{\text{obs}})$ by

$$g_{new}(\theta) = \frac{1}{m} \sum_{i=1}^{m} p(\theta \mid Y_{\text{obs}}, y_{\text{mis}}^{(i)})$$

Data Augmentation Algorithm

The data augmentation algorithm is as follows:

- For $t = 1, \ldots, T$ (large T):
  - For $j = 1, \ldots, m$:
    - (DA1) Draw l from the set $\{1, \ldots, m\}$ uniformly.
    - (DA2) Draw a θ^* from $p(\theta \mid Y_{\text{obs}}, y_{\text{mis}}^{(t-1,l)})$.
    - (DA3) Draw $y_{\text{mis}}^{(t,j)}$ from $p(Y_{\text{mis}} \mid Y_{\text{obs}}, \theta^*)$.
  - End for
- End for

The steps (DA1) and (DA2) draw a sample θ^* from

$$g_{t-1}(\theta) = \frac{1}{m} \sum_{j=1}^{m} p(\theta \mid Y_{\text{obs}}, y_{\text{mis}}^{(t-1,j)})$$

The data augmentation algorithm iteratively updates θ and y_{mis} in an alternating order. It is related to Gibbs sampling when we draw jointly from $p(\theta, Y_{\text{mis}} \mid Y_{\text{obs}})$ when $m = 1$.

Example: Hierarchical Model

In the previous hierarchical model, we have

$$\begin{aligned} \mu, \sigma, \tau &\propto 1/\sigma \\ \theta_j \mid \mu, \tau &\sim \mathcal{N}(\mu, \tau^2), i.i.d. \\ y_{ij} \mid \theta_j, \sigma &\sim \mathcal{N}(\theta_j, \sigma^2), i.i.d. \end{aligned}$$

In this case, we have the following correspondence:

$$Y_{\text{obs}} = y, \quad Y_{\text{mis}} = \theta$$

The steps are:

- (DA2) Draw $\mu^{t+1}, \sigma^{t+1}, \tau^{t+1}$ from $p(\mu, \sigma, \tau \mid y, \theta^t)$. (Use one step of Gibbs sampling)
- (DA3) Draw θ^{t+1} from $p(\theta \mid y, \mu^{t+1}, \sigma^{t+1}, \tau^{t+1})$.

Same code as before for $m = 1$.

Example: t-distribution

Consider the following model:

$$\mu, \sigma \propto 1/\sigma$$
$$y_1, \ldots, y_n \sim t_\nu(\mu, \sigma^2), i.i.d.$$

Recall the marginal distribution of the normal-inverse-gamma is t-distribution.
Update the model by introducing an auxiliary variable V:

$$\begin{aligned} \mu, \sigma &\propto 1/\sigma \\ V_i &\sim \text{Inv}{-}\chi^2(\nu, \sigma^2), i.i.d. \\ y_i \mid V_i &\sim t_\nu(\mu, V_i), independent. \end{aligned}$$

Now the correspondence is

$$Y_{\text{obs}} = y, \quad Y_{\text{mis}} = V$$

Example: t-distribution

$$\begin{aligned} \mu, \sigma &\propto 1/\sigma \\ V_i &\sim \text{Inv}{-}\chi^2(\nu, \sigma^2), i.i.d. \\ y_i \mid V_i &\sim t_\nu(\mu, V_i), independent. \end{aligned}$$

- (DA2) Draw $\mu^{(t+1)}, \sigma^{(t+1)}$ from $p(\mu, \sigma \mid y, V^{(t)})$ with one step of Gibbs sampler:

$$p(\mu \mid \sigma, y, V) \sim \mathcal{N}\left(\frac{\sum_i y_i/V_i}{\sum_i 1/V_i}, \frac{1}{\sum_i 1/V_i}\right)$$

$$p(\sigma^2 \mid \mu, y, V) \sim \text{Gamma}\left(\frac{n\nu}{2}, \frac{\nu}{2}\sum_{i=1}^{n}\frac{1}{V_i}\right)$$

- (DA3) Draw $V^{(t+1)}$ from $p(V \mid y, \mu^{(t+1)}, \sigma^{(t+1)})$:

$$p(V_i \mid y, mu, \sigma) \sim \text{Inv}{-}\chi^2\left(\nu + 1, \frac{\nu\sigma^2 + (y_i - \mu)^2}{\nu + 1}\right)$$