BAYESIAN STATISTICS

Chapter 12

Instructor: Seokho Lee (이석호)

Hankuk University of Foreign Studies

12. Markov Chain Monte Carlo (MCMC)

12.1. Monte Carlo method

Suppose we are interested in computing the integration of a function $h(x)$ on an interval (a, b):

$$\int_a^b h(x)dx.$$

If $h(x)$ factors into the multiplication of a certain function $g(x)$ and a certain probability density function $f(x)$ of a random variable X, then

$$\int_a^b h(x)dx = \int_a^b g(x)f(x)dx = E[g(X)].$$

If we are able to obtain a random sample $x_1, x_2, \ldots, x_n$ with a reasonably large n, then the expectation can be approximated well by

$$E[g(X)] \approx \frac{1}{n}\sum_{i=1}^{n} g(x_i).$$

This numerical integration is called **Monte Carlo integration**. Monte carlo integration is very useful when the integration is analytically complicated.

Example (12-1)

Calculate the below integration analytically and using Monte Carlo integration. And compare them.

$$\int_{-1}^{1} x^2 dx.$$

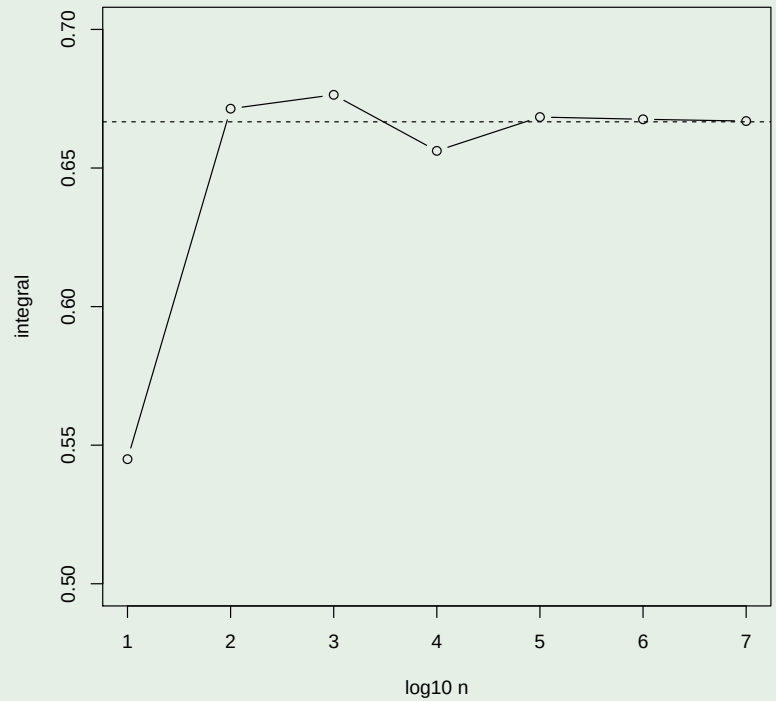
(solution) The exact solution is easily computed as $2/3 = .6667$. For the Monte Carlo integration, we can factor x^2 into

$$x^2 = 2x^2 \cdot \frac{1}{2}.$$

Note that $f(x) = 1/2$ is the probability density function of $U(-1, 1)$. Thus, we can generate $x_1, x_2, \ldots, x_n$ from uniformly on the interval $(-1, 1)$ and calculate $\sum_{i=1}^{n} 2x_i^2/n$.

Example (12-1, continue)

```
> mc<-NULL
> n<-10^(1:7)
> set.seed(1234)
> for(i in 1:7){
+  x<-runif(n[i],min=-1,max=1)
+  mc<-rbind(mc,c(n[i],mean(2*x^2)))
+ }
> mc
      [,1]      [,2]
[1,] 1e+01 0.5449625
[2,] 1e+02 0.6714080
[3,] 1e+03 0.6764102
[4,] 1e+04 0.6561942
[5,] 1e+05 0.6683899
[6,] 1e+06 0.6675851
[7,] 1e+07 0.6669439
> plot(1:7,mc[,2],type='b',xlab='log10 n',
ylab='integral',ylim=c(0.5,0.7))
> abline(h=2/3,lty=2)
> dev.copy2pdf(file='fig12-1.pdf')
```

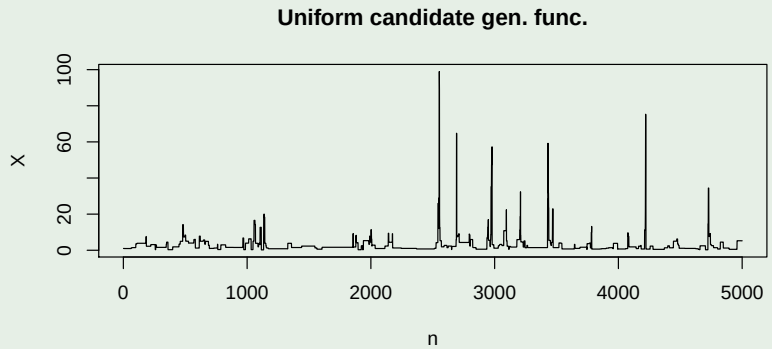


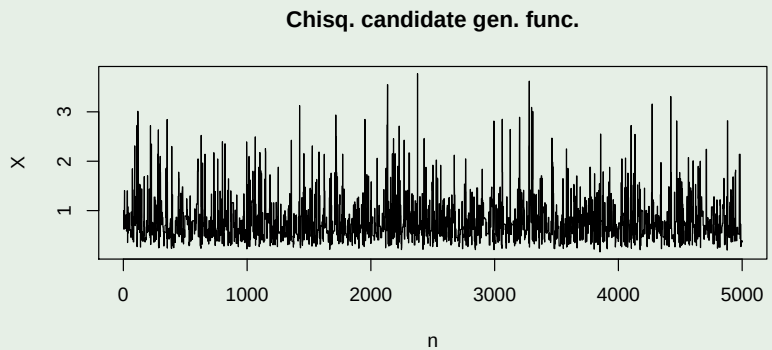
Example (12-2)

Compute the expectation of Gamma(3, 0.2) using a Monte Carlo method.

(solution) The theoretical value is $3 \times 0.2 = 0.6$.

```
> alpha<-3; beta<-0.2
> mc<-NULL
> n<-10^(1:7)
> set.seed(1234)
> for(i in 1:7){
+  x<-rgamma(n[i],shape=alpha,scale=beta)
+  mc<-rbind(mc,c(n[i],mean(x)))
+ }
> mc
      [,1]      [,2]
[1,] 1e+01 0.4360828
[2,] 1e+02 0.6047663
[3,] 1e+03 0.6168883
[4,] 1e+04 0.5945474
[5,] 1e+05 0.5990375
[6,] 1e+06 0.6006146
[7,] 1e+07 0.5999258
> plot(1:7,mc[,2],type='b',xlab='log10 n',
ylab='Monte Carlo mean')
> abline(h=0.6,lty=2)
> dev.copy2pdf(file='fig12-2.pdf')
```





12.2. Markov chain

Stochastic process is the sequence of random variable $\{X_0, X_1, X_2, \dots\}$. Especially, a stochastic process is called the **Markov chain** if it satisfies

$$\Pr(X_{n+1} \in A | X_n, X_{n-1}, \dots, X_1, X_0) = \Pr(X_{n+1} \in A | X_n)$$

for any positive integer n. This implies that the conditional distribution of the current random variable depends only on the latest one.

One of the simple Markov chains is a set of the independent and identically distributed random variables, $\{X_1, X_2, \dots\}$ (why?). Also we can show that the partial sum of independent and identically distributed random variables

$$S_n = X_1 + X_2 + \cdots + X_n, \qquad n = 1, 2, \dots.$$

is a Markov chain (i.e., a stochastic process $\{S_1, S_2, \dots, S_n\}$ is a Markov chain. Why?).

For a Markov chain $\{X_0, X_1, \dots\}$, if the distribution of X_n does not depend on the initial value X_0 for a large n, then this Markov chain has the **stationary distribution**.

The stationary distribution always exists for any Markov chain. However, if a Markov chain meets some requirements, then it has a unique stationary distribution.

12.3. Markov chain monte carlo

Monte carlo method cannot be used if it is not easy to draw random numbers from the density function $f(x)$. In such case, **Markov chain Monte Carlo (MCMC)** can be effectively used.

A basic idea of MCMC is that, instead of generating random numbers from $f(x)$ directly, we generate random numbers from a Markov chain whose stationary distribution is $f(x)$.

- **Step 1.** Construct a Markov chain $\{X_0, X_1, \ldots, X_t\}$ whose stationary distribution is $f(x)$ and keep X_t.
- **Step 2.** Repeat **Step 1** n times. n random variables of X_t are random numbers from $f(x)$.

The quality of random numbers improves as t gets large because X_t will not depend on the initial value X_0. **Burn-in** is the period such that the Markov chain is free from the initial value beyond this period.

12.3.1. Metropolis-Hastings algorithm

Metropolis-Hastings algorithm (M-H algorithm) is a device that makes a Markov chain whose stationary distribution is a target density function.

Suppose we would like to draw random samples from a density function $f(x)$ from which a random number is not easily generated. And, moreover, it has a form of $f(x) = C \cdot h(x)$ where a normalizing constant $C = \{\int_{-\infty}^{\infty} h(x)dx\}^{-1}$ is unknown and not calculable.

First, we choose a **candidate generating density** $q(x_2|x_1)$. The candidate generating density can be any density function from which a random number is easily generated. Using a relevant $q(x_2|x_1)$, we define **the probability of move** α as

$$\alpha = \alpha(x_1, x_2) = \min\left\{\frac{f(x_2)q(x_2|x_1)}{f(x_1)q(x_1|x_2)}, 1\right\} = \min\left\{\frac{h(x_2)q(x_2|x_1)}{h(x_1)q(x_1|x_2)}, 1\right\}$$

If α is large, then a Markov chain is quickly generated. If α is small, then it takes longer to obtain a Markov chain.

Definition (Metropolis-Hasting Algorithm)

1. Set a initial value x_0 of $f(x_0) > 0$ or $h(x_0) > 0$.
2. Generate $x \sim q(\cdot|x_j)$ and $u \sim \mathrm{U}(0,1)$.
3. If $\alpha(x_j, x) \geq u$ then $x_{j+1} = x$. Otherwise, $x_{j+1} = x_j$.
4. Set $j \leftarrow j + 1$ and go to step 1.

After burn-in period, the resulting process is known to be a Markov chain whose stationary distribution is $f(x)$. Since the probability of move, α, depends on $q(x_2|x_1)$, the choice of $q(x_2|x_1)$ is important. And it is also important how long we take a burn-in period. All of these questions are out of scope of this lecture.

M-H algorithm can be used for generating random samples from a multivariate random variable too.

Example (12-3)

Using M-H algorithm, generate 10 random numbers from the density

$$f(x) \propto x^{-5/2} e^{-2/x}, \quad x \geq 0$$

(solution) We choose two candidate generating densities, one of which is U(0, 100)

$$q(x_2|x_1) = \frac{1}{100}, \quad 0 \leq x_2 \leq 100$$

and the other is $\chi^2(1)$

$$q(x_2|x_1) = \frac{1}{\sqrt{2\pi}} x_2^{-1/2} e^{-x_2/2}, \quad x \geq 0.$$

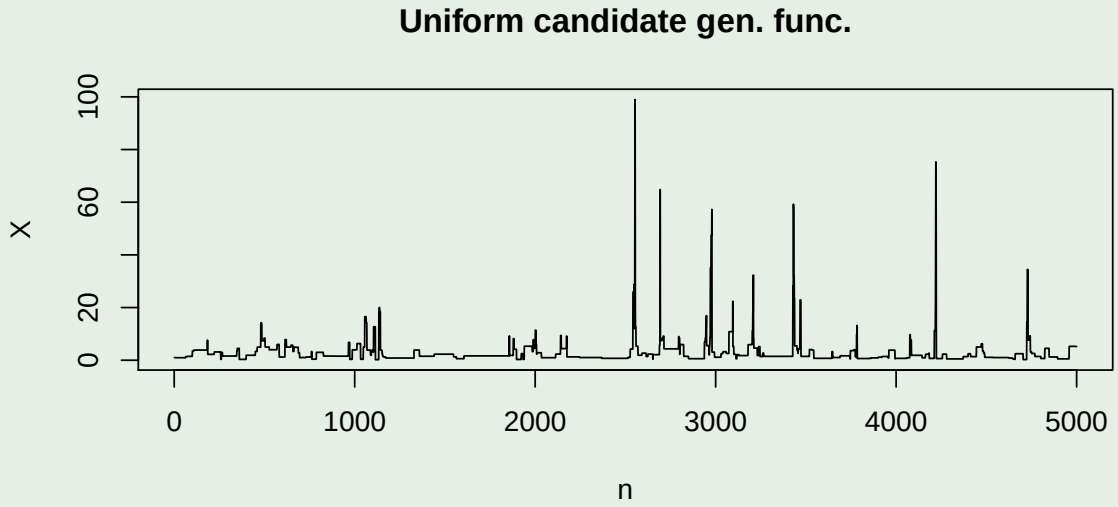
For these two candidates, we can implement them with the initial $x_0 = 1$ as following:

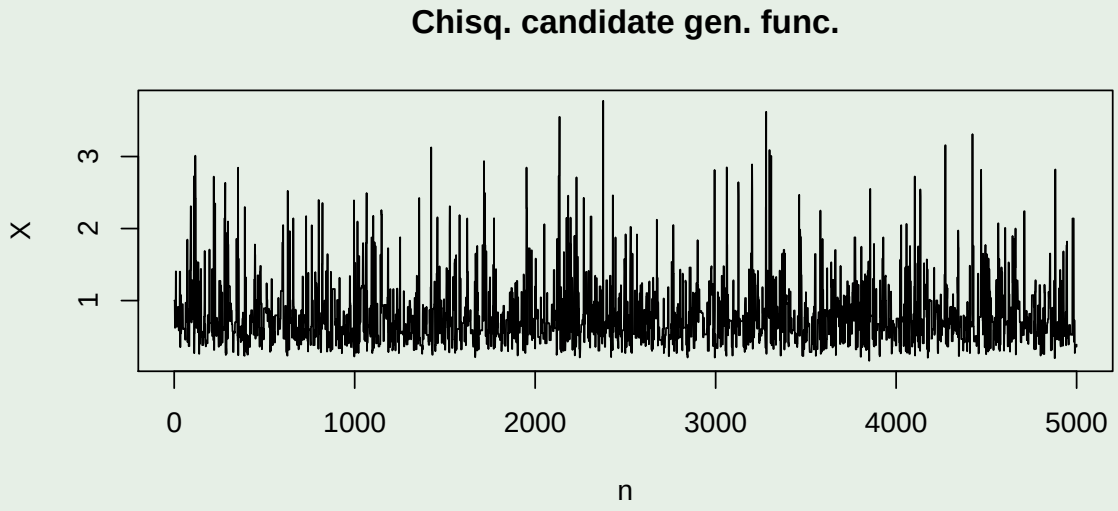
```
> q<-function(d,x1,x2){
+  if(d==1) ans <- 0.01*(0<=x2)*(x2<=100)
+  if(d==2) ans <- dchisq(x2,1)
+  return(ans)
+ }
>
> h<-function(x) x^(-5/2)*exp(-2/x)*(x>=0)
```

Example (12-3, continue)

```
> MH<-function(d,x0,len){
+  if(d==1) x<-runif(len)*100
+  if(d==2) x<-rchisq(len,1)
+  u<-runif(len)
+  X<-NULL
+  for(i in 1:len){
+  a1 <- h(x[i])*q(d,x0,x[i])
+  a2 <- h(x0)*q(d,x[i],x0)
+  alpha<-min(1,a1/a2)
+  if(alpha>=u[i]) {
+  X<-c(X,x[i]); x0<-x[i]
+  } else X<-c(X,x0)
+  }
+  return(X)
+ }
>
> len.mc<-5000
> set.seed(1234)
> par(mfrow=c(2,1))
> d<-1; X<-MH(d,1,len.mc)
> plot(X,type='l',xlab='n',main='Uniform candidate gen. func.')
> d<-2; X<-MH(d,1,len.mc)
> plot(X,type='l',xlab='n',main='Chisq. candidate gen. func.')
> dev.copy2pdf(file='fig12-2.pdf')
```

Example (12-3, continue)



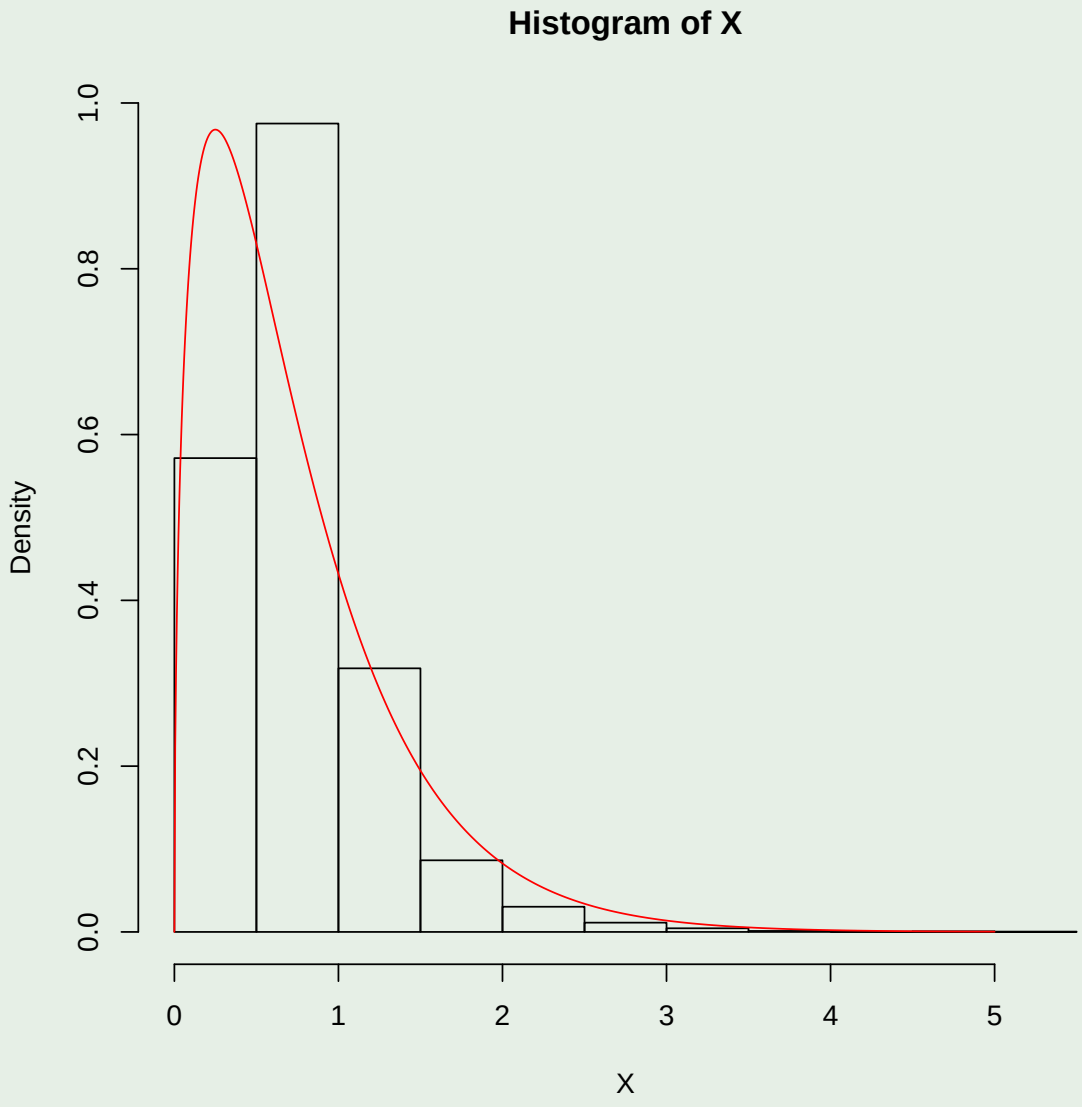


Example (12-3, continue)

When the uniform candidate is used, the corresponding Markov chain is not mixing well even after a long burn-in. However, the Markov chain induced by the chi-square candidate seems mixing very well. So we use the chi-square candidate generating function.

```
> X<-NULL; N<-5000
> len.mc<-100 # burn-in
> set.seed(56778)
> for(i in 1:N){
+  mc.MH<-MH(2,1,len.mc)
+  X<-c(X,mc.MH[len.mc])
+ }
> hist(X,freq=FALSE)
> xx<-seq(0,5,length=1000)
> yy<-dgamma(xx,shape=3/2,scale=1/2)
> lines(xx,yy,col='red')
> dev.copy2pdf(file='fig12-3.pdf')
```

Example (12-3, continue)



12.3.2. Gibbs sampler

It is difficult to generate random vectors from d-dimensional random variable. **Gibbs sample** enables us to generate the multivariate random numbers using several conditional distributions from which a random number is easily generated.

Suppose we are interested in generating random vectors from the d-dimensional multivariate density function

$$f(\mathbf{x}), \quad \mathbf{x} = (x_1, x_2, \ldots, x_d)^T.$$

Define $d-1$ dimensional vector by deleting the ith component from $\mathbf{x}$:

$$\mathbf{x}_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)^T, \quad i = 1, 2, \ldots, d.$$

Assume that we can easily generate a random number from the conditional distribution $f(x_i|\mathbf{x}_{-i})$, $i = 1, 2, \ldots, d$.

Definition (Gibbs sampler)

1. Set $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_d^{(0)})^T$
2. Obtain $\mathbf{x}^{(j+1)} = (x_1^{(j+1)}, x_2^{(j+1)}, \ldots, x_d^{(j+1)})^T$ as follows:

$$
\begin{aligned}
x_1^{(j+1)} &\sim f(x_1 | x_2^{(j)}, x_3^{(j)}, \ldots, x_d^{(j)}) \\
x_2^{(j+1)} &\sim f(x_2 | x_1^{(j+1)}, x_3^{(j)}, \ldots, x_d^{(j)}) \\
x_3^{(j+1)} &\sim f(x_3 | x_1^{(j+1)}, x_2^{(j+1)}, x_4^{(j)} \ldots, x_d^{(j)}) \\
&\vdots \\
x_d^{(j+1)} &\sim f(x_d | x_1^{(j+1)}, x_2^{(j+1)}, \ldots, x_{d-1}^{(j+1)})
\end{aligned}
$$

3. $j \leftarrow j + 1$ and go to step 2.

After a sufficient burn-in period, the multivariate process becomes a Markov chain whose stationary distribution is $f(\mathbf{x})$.

Note that multivariate random vector can be generated from a 1-dimensional conditional distribution.

Example (12-4)

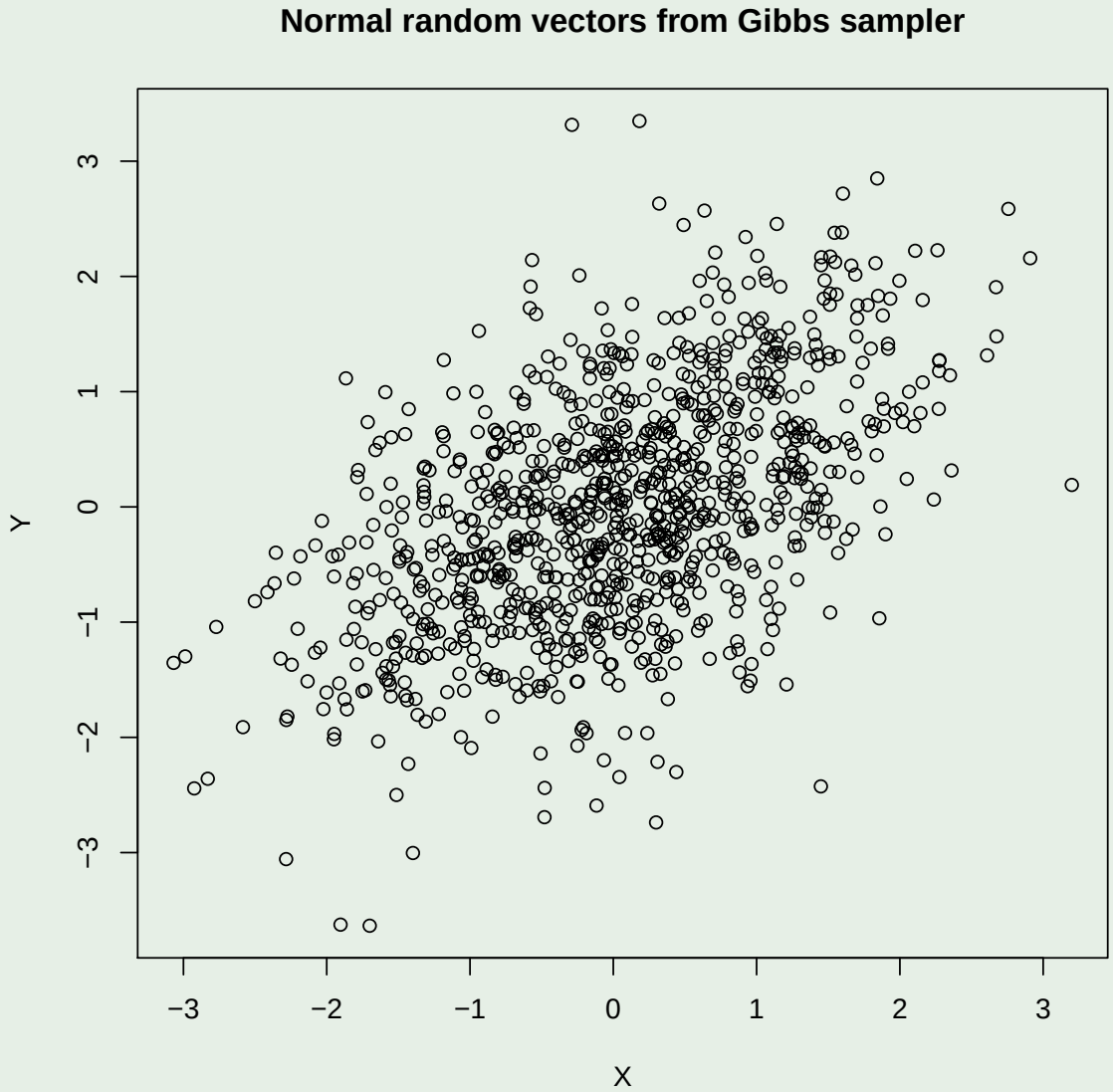
Write an R code to generate a bivariate normal distribution $N_2\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 0\end{pmatrix}\right)$ *using Gibbs sampler.*

(solution) Note that, when $\begin{pmatrix}X\\Y\end{pmatrix} \sim \mathrm{N}_2\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 0\end{pmatrix}\right)$

$$X|Y \sim \mathrm{N}(\rho Y, 1-\rho^2), \quad Y|X \sim \mathrm{N}(\rho X, 1-\rho^2)$$

```
> gibbs.bvn<-function(rho,len){
+  y<-0
+  for(i in 1:len){
+  x<-rnorm(1,rho*y,sqrt(1-rho^2))
+  y<-rnorm(1,rho*x,sqrt(1-rho^2))
+  }
+  return( c(x,y) )
+ }
> rand.bvn<-function(n,rho){
+  len<-100; xy<-NULL
+  for(i in 1:n) xy<-rbind(xy,gibbs.bvn(rho,len))
+  return(xy)
+ }
> rho<-0.5
> set.seed(123456)
> xy<-rand.bvn(1000,rho)
> plot(xy,xlab='X',ylab='Y',main='Normal random vectors from Gibbs sampler')
> dev.copy2pdf(file='fig12-4.pdf')
```

Example (12-4, continue)



Example (12-4, continue)

```
> colMeans(xy) # close to (0,0)
[1]  0.02027136 -0.01810503
> apply(xy,2,var) # close to (1,1)
[1] 1.073975 1.058566
> cor(xy) # off-diagonal is close to 0.5
          [,1]      [,2]
[1,] 1.0000000 0.5210288
[2,] 0.5210288 1.0000000
> library(moments)
Warning message:
패키지 'moments'는 버전 2.12.1의 R미만에서 작성되어졌습니다
> apply(xy,2,skewness) # close to (0,0)
[1] -0.07193844  0.03960001
> apply(xy,2,kurtosis) # close to (3,3)
[1] 2.781610 3.089416
```

12.4. Posterior inference using MCMC

Bayesian inference is always based on posterior distribution of parameters.

However, posterior computation and relating statistics involve the integration of the complicated functions unless the conjugate prior is available. In such case, MCMC is an good option.

When the prior is $\pi(\theta)$ and the likelihood is $\ell(\mathbf{x}|\theta)$, the posterior $h(\theta|\mathbf{x})$ is given as

$$h(\theta|\mathbf{x}) = C \cdot \ell(\mathbf{x}|\theta)\pi(\theta)$$

with

$$C = \left\{ \int_\Theta \ell(\mathbf{x}|\theta)\pi(\theta)d\theta \right\}^{-1}$$

We can generate random numbers (or random vectors) from $h(\theta|\mathbf{x})$ using M-H algorithm or Gibbs sampler. And then we examine the properties of the posterior distribution using random numbers.

For example, if we generate n random numbers $\theta_1, \theta_2, \ldots, \theta_n$ from the posterior distribution, then the posterior mean $E(\theta|\mathbf{x})$ is

$$E(\theta|\mathbf{x}) = \int_\Theta \theta h(\theta|\mathbf{x})d\theta \approx \frac{1}{n}\sum_{i=1}^{n} \theta_i$$

as an approximation.

12.4.1. Using Gibbs sampler

Suppose the d-dimensional parameter $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_d)^T$ has the posterior $h(\boldsymbol{\theta}|\mathbf{x})$ and we can easily draw a sample from $h(\theta_i|\boldsymbol{\theta}_{-i}, \mathbf{x})$. With a burn-in period M,

1. Set $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_d^{(0)})^T$
2. Obtain $\boldsymbol{\theta}^{(j+1)} = (\theta_1^{(j+1)}, \theta_2^{(j+1)}, \ldots, \theta_d^{(j+1)})^T$ as follows:

$$
\begin{aligned}
\theta_1^{(j+1)} &\sim f(\theta_1|\theta_2^{(j)}, \theta_3^{(j)}, \ldots, \theta_d^{(j)}, \mathbf{x}) \\
\theta_2^{(j+1)} &\sim f(\theta_2|\theta_1^{(j+1)}, \theta_3^{(j)}, \ldots, \theta_d^{(j)}, \mathbf{x}) \\
\theta_3^{(j+1)} &\sim f(\theta_3|\theta_1^{(j+1)}, \theta_2^{(j+1)}, \theta_4^{(j)} \ldots, \theta_d^{(j)}, \mathbf{x}) \\
&\vdots \\
\theta_d^{(j+1)} &\sim f(\theta_d|\theta_1^{(j+1)}, \theta_2^{(j+1)}, \ldots, \theta_{d-1}^{(j+1)}, \mathbf{x})
\end{aligned}
$$

3. If $j = M$ then stop and keep $\boldsymbol{\theta}^{(M)}$. Otherwise, $j \leftarrow j + 1$ and go to step 2.

12.4.2. Using M-H algorithm

Rather than Gibbs sampler, M-H algorithm doesn't need to obtain the conditional distribution. Thus, M-H algorithm is a good option when it is difficult to obtain the conditional distribution.

1. Set the initial value $\boldsymbol{\theta}_0$ satisfying $\pi(\boldsymbol{\theta}_0)\ell(\mathbf{x}|\boldsymbol{\theta}_0) > 0$.
2. With a relevant candidate generating function, generate $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1)$ and generate $u \sim \mathrm{U}(0,1)$.
3. Compute the probability of move

$$\alpha(\boldsymbol{\theta}_j, \boldsymbol{\theta}^*) = \min\left\{\frac{\pi(\boldsymbol{\theta}^*)\ell(\mathbf{x}|\boldsymbol{\theta}^*)q(\boldsymbol{\theta}^*|\boldsymbol{\theta}_j)}{\pi(\boldsymbol{\theta}_j)\ell(\mathbf{x}|\boldsymbol{\theta}_j)q(\boldsymbol{\theta}_j|\boldsymbol{\theta}^*)}, 1\right\}$$

   and $\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}^*$ if $\alpha(\boldsymbol{\theta}_j, \boldsymbol{\theta}^*) \geq u$ or $\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j$ otherwise.
4. If $j = M$ then stop and keep $\boldsymbol{\theta}_M$. Otherwise, $j \leftarrow j + 1$ and go to step 2.

Example (12-5)

Suppose $X \sim B(10, \theta)$ *and the prior distribution* $\pi(\theta)$ *is*

$$\pi(\theta) = \frac{1}{\sqrt{2\pi}} \frac{1}{\theta(1-\theta)} e^{-\frac{1}{2}\left\{\log\left(\frac{\theta}{1-\theta}\right)\right\}^2}, \quad 0 < \theta < 1.$$

Then, $\rho = \log(\theta/(1-\theta)) \sim N(0,1)$. *When we observe* $X = 5$, *obtain the posterior mean of* θ *using MCMC.*

(solution) The posterior distribution is

$$h(\theta|x) = C \cdot \pi(\theta)\theta^x(1-\theta)^{10-x}$$

which is not a familiar form because the prior is not beta distribution that is a conjugate prior. Thus, we use M-H algorithm. We choose the candidate generating function as a uniform distribution:

$$q(\theta_2|\theta_1) = 1, \quad 0 < \theta_2 < 1.$$

Then, the probability of move is given as

$$\alpha(\theta_1, \theta_2) = \min\left\{\frac{\pi(\theta_2)\ell(x|\theta_2)}{\pi(\theta_1)\ell(x|\theta_1)}, 1\right\}$$

Example (12-5, continue)

```
> prior<-function(theta){
+  tmp1<-1/theta/(1-theta)/sqrt(2*pi)
+  tmp2<- -0.5*(log(theta/(1-theta)))^2
+  return(tmp1*exp(tmp2))
+ }
>
> ell<-function(x,theta) dbinom(x,10,theta)
>
> MH.bin<-function(len,x){
+  theta0<-0.5
+  theta.s<-runif(len)
+  u<-runif(len)
+  X<-NULL
+  for(i in 1:len){
+  a1<-prior(theta.s[i])*ell(x,theta.s[i])
+  a0<-prior(theta0)*ell(x,theta0)
+  alpha<-min(a1/a0,1)
+  if(alpha>=u[i]){
+  X<-c(X,theta.s[i]); theta0<-theta.s[i]
+  } else X<-c(X,theta0)
+  }
+  return(X)
+ }
```

Example (12-5, continue)

```
> M<-100 # burn-in
> x<-5 # date
> n<-100 # number of random numbers
> theta<-NULL
> for(i in 1:n){
+  X<-MH.bin(M,x)
+  theta<-c(theta,X[M])
+ }
> mean(theta)
[1] 0.4542045
```

Example (12-5, continue)

```
> set.seed(12334)
> M<-500 # burn-in
> x<-5 # date
> n<-1000 # number of random numbers
> theta<-NULL
> for(i in 1:n){
+  X<-MH.bin(M,x)
+  theta<-c(theta,X[M])
+ }
> par(mfrow=c(2,1))
> mn<-cumsum(theta)/(1:n)
> plot(1:n,mn,type='l',xlab='n',ylab='posterior mean of theta')
> abline(h=0.5,col='red',lwd=2)
> hist(theta,main='',xlab='theta',ylab='frequency')
> abline(v=0.5,col='red',lwd=2)
> dev.copy2pdf(file='fig12-5.pdf')
```

Example (12-5, continue)