Lectures 9-13: Monte Carlo Statistical Methods

MAST90125 Bayesian Statistical Learning

Guoqi Qian

School of Mathematics & Statistics
The University of Melbourne

Outline

Introduction

- In finding an MLE of θ using EM algorithm we need evaluate

$$Q(\theta, \theta') = E[\ln f(\mathbf{y}, \mathbf{Z}|\theta)|\mathbf{y}, \theta'] = \int \ln f(\mathbf{y}, \mathbf{z}|\theta) k(\mathbf{z}|\mathbf{y}, \theta') d\mathbf{z}$$

  where $k(\mathbf{z}|\mathbf{y}, \theta')$ is the conditional pdf of $\mathbf{Z}$ given that $\mathbf{y}$ is observed, and $(\mathbf{y}, \mathbf{z})$ is the complete data.
- In finding a Bayes estimator $T = E(\theta|\mathbf{y}_n)$ of θ we need the posterior distribution $p(\theta|\mathbf{y}_n)$ which typically has the form

$$p(\theta|\mathbf{y}_n) = \frac{p(\mathbf{y}_n|\theta)p(\theta)}{\int p(\mathbf{y}_n|\theta)p(\theta)d\theta}.$$

- In these and many other situations, it can be very difficult to evaluate the integrals involved by an analytical method.
- Monte Carlo methods can be used to overcome the difficulties.

What is a Monte Carlo method? (1)

- The essential component of a Monte Carlo method is random number generation from a probability distribution. If we can generate a random sample from a probability distribution, e.g. the posterior pdf $p(\theta|\mathbf{y}_n)$, then any numeric characteristic of the distribution can be approximated to any given precision by certain sample mean, e.g. $E(\tau(\theta)|\mathbf{y}_n) \approx \frac{1}{M}\sum_{m=1}^{M}\tau(\theta_m)$, where $(\theta_1, \cdots, \theta_M)$ is a sample from $p(\theta|\mathbf{y}_n)$.
- In general, if we want to evaluate $\mu = E[h(\mathbf{X})]$ where $\mathbf{X}$ has a pdf $f_{\mathbf{X}}(\mathbf{x})$, we can use a Monte Carlo method by generating an iid sample $\mathbf{X}_1, \cdots, \mathbf{X}_n$ from $f_{\mathbf{X}}(\mathbf{x})$ and estimate μ by

$$\hat{\mu}_{\mathrm{MC}} = \frac{1}{n}\sum_{i=1}^{n} h(\mathbf{X}_i).$$

What is a Monte Carlo method? (2)

- By the **strong law of large numbers (SLLN)**,

  $$\hat{\mu}_{\mathrm{MC}} \overset{a.s.}{\to} \int h(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = \mu \quad \text{as } n \to \infty \text{ under certain conditions.}$$

- As $\mathrm{var}(\hat{\mu}_{\mathrm{MC}}) = \frac{1}{n}\mathrm{var}(h(\mathbf{X})) = \frac{1}{n}E([h(\mathbf{X}) - \mu]^2)$, it can also be estimated by

  $$\widehat{\mathrm{var}}(\hat{\mu}_{\mathrm{MC}}) = \frac{1}{n^2}\sum_{i=1}^{n}[h(\mathbf{X}_i) - \hat{\mu}_{\mathrm{MC}}]^2$$

  with $\frac{1}{n}\sum_{i=1}^{n}[h(\mathbf{X}_i) - \hat{\mu}_{\mathrm{MC}}]^2 \overset{a.s.}{\to} \mathrm{var}(h(\mathbf{X}))$ as $n \to \infty$ by SLLN.
- Monte Carlo approximation has a slow $\mathcal{O}(n^{-1/2})$ convergence rate while some numeric quadrature methods has a fast one $\mathcal{O}(n^{-2})$. However, the latter become computationally infeasible when $\mathbf{X}$ is multi-dimensional whereas the former remains powerful in high-dimensional cases.

What is a Monte Carlo method? (3)

- Generating random numbers from a target probability distribution fundamentally depends on the generation of standard uniform random numbers. The latter has been extensively studied (cf D. Knuth: *The Art of Computer Programming 2: Seminumerical Algorithms*, 1997).
- Many software packages have code to generate random numbers from common distributions.
- We focus on generating random numbers from **difficult target distributions** in this chapter. Most posterior distributions in Bayesian inference are difficult ones as they involve difficult integrations. Many target distributions in modern statistical models are known only up to a multiplicative normalization constant thus are also difficult.

Random number generation methods

- If $F(x)$ is a continuous cdf, the following algorithm is well-known for generating a random number from $F(x)$:

  Algorithm 1 [*Transformation sampling*]

  1° Generate $U = u$ from Unif$(0, 1)$.

  2° Compute $x = F^{-1}(u)$.

  3° Deliver $X = x$.

  Feasibility of Algorithm 1 depends on that of $F^{-1}(u)$.
- In this chapter we introduce four methods to tackle difficult random number generation:
  - Acceptance-Rejection sampling
  - Importance sampling
  - Gibbs sampler
  - Metropolis-Hastings algorithm

  The last two are special cases of the Markov chain Monte Carlo (**MCMC**) methods.

Fundamental theorem of A-R sampling

Theorem (Fundamental theorem of A-R sampling)

Let $f(x)$ and $g(x)$ be two probability density functions (pdf's), and $h(x)$ be a given real-valued function. Consider the following acceptance-rejection sampling algorithm:

Algorithm 2 *[A-R sampling]*

1° *Generate $X = x$ from $f(x)$.*

2° *Generate $Y = y$ from $g(y)$.*

3° *If $y \leq h(x)$, then deliver $Z = x$; otherwise go to 1°.*

Then the pdf of Z is $p(z) = \dfrac{f(z)G(h(z))}{\int_{-\infty}^{\infty} f(s)G(h(s))ds}$

where $G(y) = \int_{-\infty}^{y} g(s)ds$ is the cdf corresponding to $g(y)$.

Remarks on the fundamental theorem of A-R sampling

- To see potential applications of the theorem, note that even when the pdf's $f(x)$ and $g(x)$ are simple functions, the resultant pdf $p(z)$ can be quite sophisticated. This implies that random numbers from some complicated pdf can be simulated from simple pdf's through an acceptance-rejection procedure.
- The price paid for this simplification is the acceptance probability $P(Y \leq h(X))$, which we want to be as large as possible in simulations.
- The distribution of the random numbers generated from A-R sampling is **exactly** $p(x)$.
- The variables x, y and z in the theorem can be multivariate.

Proof of the fundamental theorem of A-R sampling (1)

Proof: Suppose steps 1° and 2° need to be repeated i times before a random number Z can be generated.
Denote $\{X_1, Y_1\}, \cdots, \{X_i, Y_i\}$ be the i independent pairs of $\{X, Y\}$ generated in the first i cycles of steps 1° and 2°. Now the event $\{Z \leq t\}$ can be partitioned into the following mutually exclusive sub-events:

$$\begin{aligned}\{Z \leq t\} &= \bigcup_{i=1}^{\infty} \{Z = X_i, Z \leq t\} \\ &= \bigcup_{i=1}^{\infty} \{X_i \leq t, Y_1 > h(X_1), \cdots, Y_{i-1} > h(X_{i-1}), Y_i \leq h(X_i)\}.\end{aligned}$$

Proof of the fundamental theorem of A-R sampling (2)

Proof(continued): Thus

$$\begin{aligned}
P\{Z \le t\} &= \sum_{i=1}^{\infty} P\{X_i \le t, Y_1 > h(X_1), \cdots, Y_{i-1} > h(X_{i-1}), Y_i \le h(X_i)\} \\
&= \sum_{i=1}^{\infty} P\{X_i \le t, Y_i \le h(X_i)\} \times P\{Y_1 > h(X_1), \cdots, Y_{i-1} > h(X_{i-1})\} \\
&= P\{X \le t, Y \le h(X)\} \sum_{i=1}^{\infty} P\{Y > h(X)\}^{i-1} \\
&= \frac{P\{X \le t, Y \le h(X)\}}{1 - P\{Y > h(X)\}} = \frac{P\{X \le t, Y \le h(X)\}}{P\{Y \le h(X)\}}
\end{aligned}$$

Proof of the fundamental theorem of A-R sampling (3)

Proof(continued):

$$
\begin{aligned}
P\{Z \le t\} &= \frac{P\{X \le t, Y \le h(X)\}}{P\{Y \le h(X)\}} \\
&= \frac{\int_{-\infty}^{t}\int_{-\infty}^{h(x)} f(x)g(y)dydx}{\int_{-\infty}^{\infty}\int_{-\infty}^{h(x)} f(x)g(y)dydx} \\
&= \int_{-\infty}^{t} \frac{f(x)\int_{-\infty}^{h(x)} g(y)dy}{\int_{-\infty}^{\infty} f(x)\left(\int_{-\infty}^{h(x)} g(y)dy\right)dx}dx \\
&= \int_{-\infty}^{t} \frac{f(x)G(h(x))}{\int_{-\infty}^{\infty} f(x)G(h(x))dx}dx = \int_{-\infty}^{t} \frac{f(x)G(h(x))}{\int_{-\infty}^{\infty} f(s)G(h(s))ds}dx.
\end{aligned}
$$

Now the pdf of Z is $p(z) = \frac{dP\{Z \le z\}}{dz} = \frac{f(z)G(h(z))}{\int_{-\infty}^{\infty} f(s)G(h(s))ds}$. □

Application 1: Rejection sampling I (1)

Let $p(x)$ be the target pdf from which we want to simulate a random sample. Suppose a simple integrable upper bound function $M(x)$, also called **envelope function**, can be found for $p(x)$, i.e. $p(x) \leq M(x)$ for every x in the support (i.e. domain) of $p(x)$. It is easy to know that $M(x)$ can be standardized to induce a new pdf $f(x) = c^{-1}M(x)$ where $c = \int_{-\infty}^{\infty} M(x)dx$.

Consider the following algorithm:

Algorithm 3 [*Rejection Sampling 1*]

1° Generate $X = x$ from $f(x)$.

2° Generate $Y = y$ from Unif$(0, 1)$ independently.

3° If $y \leq \frac{p(x)}{M(x)}$, then deliver $Z = x$; otherwise go to 1°.

Application 1: Rejection sampling I (2)

By the Fundamental Theorem of A-R sampling, Z generated by the above algorithm has the pdf

$$\frac{f(z)G\left(\frac{p(z)}{M(z)}\right)}{\int_{-\infty}^{\infty} f(s)G\left(\frac{p(s)}{M(s)}\right)ds} = \frac{\frac{M(z)}{c}\times\frac{p(z)}{M(z)}}{\int_{-\infty}^{\infty}\frac{M(s)}{c}\times\frac{p(s)}{M(s)}ds} = \frac{p(z)}{\int_{-\infty}^{\infty}p(s)ds} = p(z)$$

noting that the cdf of Unif(0,1) is $G(x) = x$ for $x \in [0,1]$.

Therefore, the random numbers generated by Algorithm 3 indeed follow the pdf $p(x)$.

Note Algorithm 3 is still feasible if $p(x) = \tilde{c}\tilde{p}(x)$ with intractable $\tilde{c}$; and in such a situation we only need an envelope $\tilde{M}(x)$ for $\tilde{p}(x)$; then 3° becomes $y \le \frac{\tilde{p}(x)}{\tilde{M}(x)}$, not involving $\tilde{c}$.

Example 1 (1)

Example 1 Derive an A-R sampling algorithm for generating random numbers from Beta(a, b) distribution in the situation when $a \geq 1$ and $b \geq 1$.

- The pdf of Beta(a, b) is
$$p(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} I(0 < x < 1).$$
- It is easy to see that an upper bound function for $p(x)$ is
$$M(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} I(0 < x < 1); \quad \text{i.e. } p(x) \leq M(x).$$
- The pdf induced by $M(x)$ is $f(x) = I(0 < x < 1)$ which is simply that of Unif(0,1). Also $\dfrac{p(x)}{M(x)} = x^{a-1}(1-x)^{b-1}$.

Example 1 (2)

Therefore, the required A-R sampling algorithm is:

Algorithm 4 [*A-R sampling for Beta (a,b) with* $a \geq 1$, $b \geq 1$]

1° Generate $X = x$ from Unif$(0, 1)$.

2° Generate $Y = y$ from Unif$(0, 1)$ independently.

3° If $y \leq x^{a-1}(1-x)^{b-1}$, then deliver $Z = x$; otherwise go to 1°.

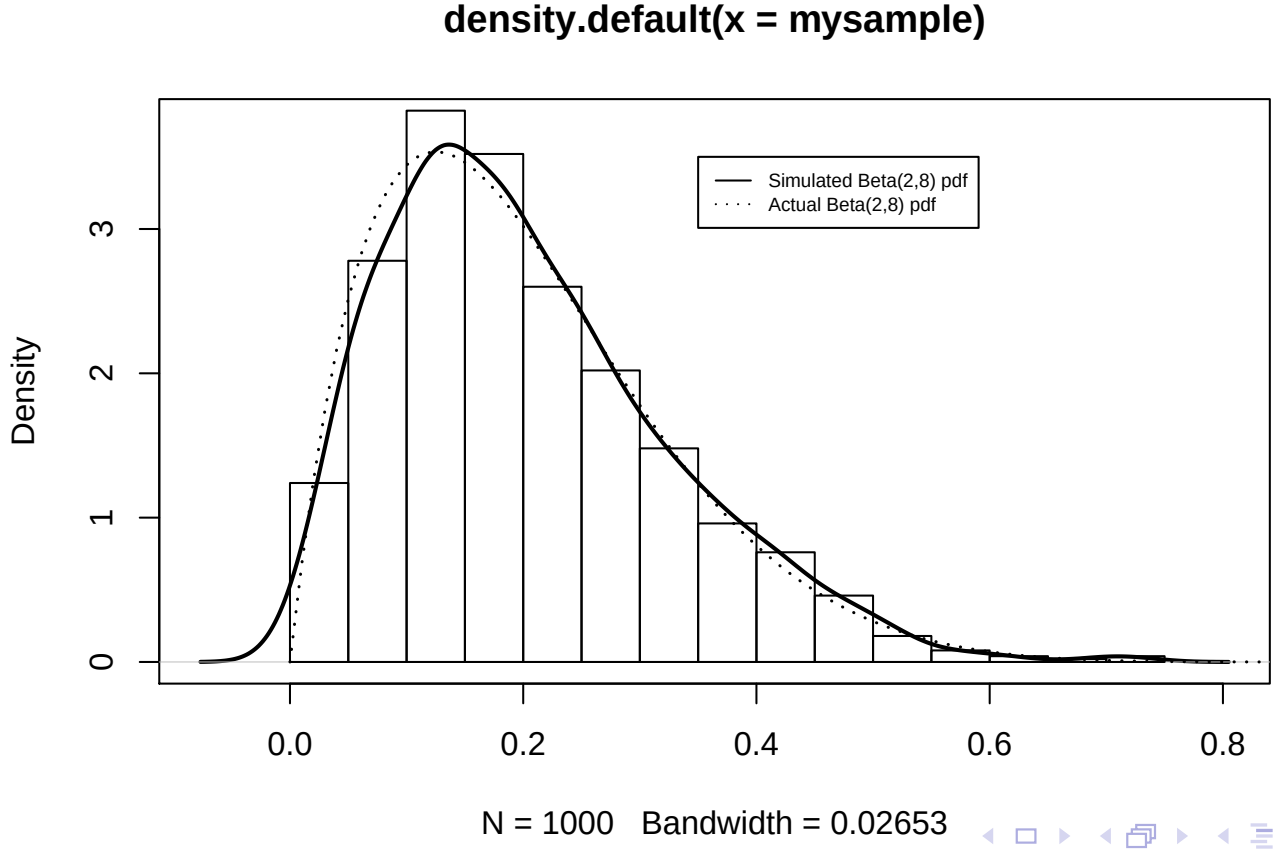
Example 1 (3)

An R program implementing Algorithm 4 is below:

```
> mybeta.f=function(n, aa,bb){
## This function is for generating a sample of "n"
## observations from beta(aa,bb) distribution with the
## restriction that aa >=1 and bb>=1.
    Z <- rep(-1, n)
    for(i in 1:n) {
        repeat { x <- runif(1)
                 y <- runif(1)
                 if(y <= x^(aa-1)*(1-x)^(bb-1)) {Z[i] <- x
                break } } }
    return(Z)}

> set.seed(456)
> mysample=mybeta.f(1000,2,8)
> plot(density(mysample),lwd=2)
> curve(dbeta(x,2,8),from=0,to=1, add=T,lwd=1.5, lty=3)
> legend(x=0.35, y=3.5, legend=c("Simulated Beta(2,8) pdf",
   "Actual Beta(2,8) pdf"), lty=c(1,3),cex=0.6)
```

Example 1 (4)

A sample of 1000 random numbers are generated from Beta(2,8) using Algorithm 4. The density curve of the sample and the true Beta(2,8) pdf curve are displayed in the following figure.



Example 1 (5)

Remarks:

- It is more difficult to derive an A-R algorithm for Beta(a, b) and implement it when $a \geq 1$ and $b \geq 1$ are not true.
- For example, when $a < 1$ and $b < 1$, $M(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \left[2^{1-b} x^{a-1} I(0 < x \leq \frac{1}{2}) + 2^{1-a}(1-x)^{b-1} I(\frac{1}{2} < x < 1)\right]$ is an upper bound function of $p(x)$; and the induced pdf is $f(x) = \frac{ab}{a+b} \left[2^a x^{a-1} I(0 < x \leq \frac{1}{2}) + 2^b (1-x)^{b-1} I(\frac{1}{2} < x < 1)\right]$.
- Then it is not trivial to generate a random number from $f(x)$ which is required in A-R sampling. One may use a transformation algorithm to do this, where it can be found

$$F^{-1}(u) = \left[\frac{(a+b)u}{2^a b}\right]^{\frac{1}{a}} I(0 < u \leq \frac{b}{a+b}) + \left(1 - \left[\frac{(a+b)(1-u)}{2^b a}\right]^{\frac{1}{b}}\right) I(\frac{b}{a+b} < u < 1).$$

Example 2 (1)

Example 2 The one-parameter logistic pdf is defined as

$$p(x) = \frac{\theta e^{\theta x}}{(1+e^{\theta x})^2}, \quad -\infty < x < \infty; \quad \theta > 0.$$

Derive an A-R algorithm to generate random numbers from logistic(θ).

It can be seen that $p(x) = \begin{cases} \frac{\theta e^{\theta x}}{(1+e^{\theta x})^2} \le \theta e^{\theta x} & \text{when } x \le 0, \\ \frac{\theta e^{-\theta x}}{(1+e^{-\theta x})^2} \le \theta e^{-\theta x} & \text{when } x > 0. \end{cases}$

So $M(x) = \theta e^{-\theta|x|}$ is an upper bound of $p(x)$. As $\int_{-\infty}^{\infty} M(x)dx = 2$, the induced pdf from $M(x)$ is $f(x) = \frac{1}{2}\theta e^{-\theta|x|}$.

Example 2 (2)

The cdf corresponding to the induced pdf is

$$F(x) = \int_{-\infty}^{x} f(t)dt = \int_{-\infty}^{x} \frac{1}{2}\theta e^{-\theta|t|}dt = \begin{cases} \frac{1}{2}e^{\theta x} & \text{if } x \leq 0, \\ 1 - \frac{1}{2}e^{-\theta x} & \text{if } x > 0. \end{cases}$$

Thus the inverse pdf is $F^{-1}(u) = \begin{cases} \theta^{-1}\ln(2u) & \text{if } u \leq \frac{1}{2}, \\ -\theta^{-1}\ln(2-2u) & \text{if } u > \frac{1}{2}. \end{cases}$

Now the required algorithm is

Algorithm 5 [*A-R & transform. mix. algorithm for logistic*(θ)]

1° Generate $U = u$ from Unif$(0, 1)$.

2° If $u \leq \frac{1}{2}$, deliver $x = \theta^{-1}\ln(2u)$; otherwise deliver $x = \theta^{-1}\ln(2-2u)$.

3° Generate $Y = y$ from Unif$(0, 1)$ independently.

4° If $y \leq (1 + e^{-\theta|x|})^{-2}$, then deliver $Z = x$; otherwise go to 1°.

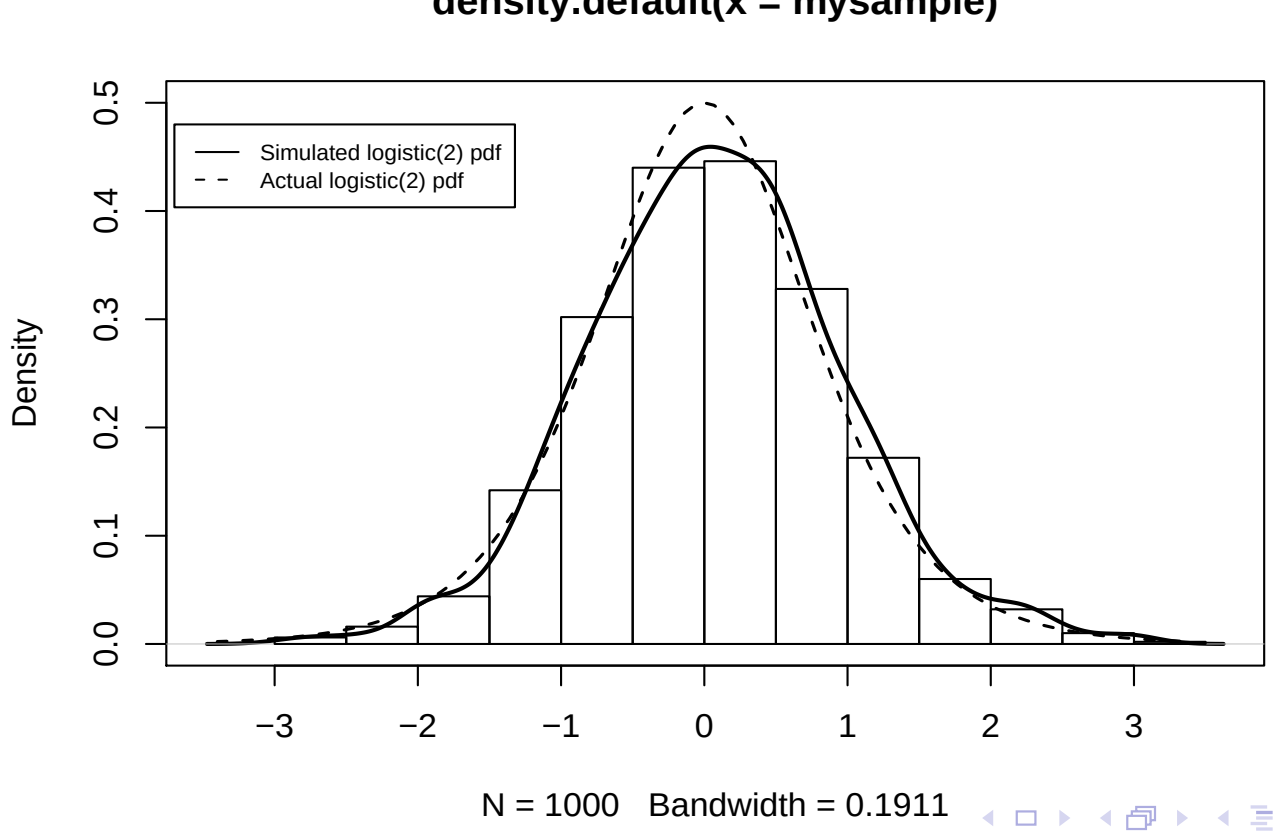
Example 2 (3)

An R program implementing Algorithm 5 is below:

```
> mylogistic.f=function(n, theta){
    Z <- rep(-1, n)
    for(i in 1:n) {
        repeat {temp <- runif(1)
                if(temp <= 0.5) {x <- log(2 * temp)/theta}
                else {x <-  - log(2 - 2 * temp)/theta}
            y <- runif(1)
            if(y <= (1 + exp( - theta * abs(x)))^(-2)) {Z[i] <- x
                break}}}
    return(Z)}

set.seed(456)
mysample=mylogistic.f(1000,2)
plot(density(mysample),lwd=2, ylim=c(0,0.5))
curve(dlogis(x,loc=0,sca=0.5),from=-3.4,to=3.4, add=T,lwd=1.5, lty=2)
legend(x=-3.7, y=0.48, legend=c("Simulated logistic(2) pdf",
  "Actual logistic(2) pdf"), lty=c(1,2),cex=0.7)
```

Example 2 (4)

A sample of 1000 random numbers are generated from logistic(2) using Algorithm 5. The density curve of the sample and the true logistic(2) pdf curve are displayed in the following figure.



Application 2: Rejection sampling II

Suppose the target pdf has the form $p(x) = Lh(x)g(x)$ where $g(x)$ is a pdf easy to simulate, $h(x)$ satisfies $0 \leq h(x) \leq 1$, and L is a positive constant may or may not be tractable. Then by the Fundamental Theorem of A-R sampling, the following algorithm can be used to generate random numbers from $p(x)$.

Algorithm 6 [*Rejection sampling II*]

1° Generate $X = x$ from the pdf $g(x)$.

2° Generate $U = u$ from Unif$(0, 1)$ independently.

3° If $u \leq h(x)$, then deliver $Z = x$; otherwise go to 1°.

Example 3 (1)

Example 3 A half-normal distribution has the pdf

$$p(x) = \sqrt{2\pi^{-1}} e^{-x^2/2} I(x \geq 0).$$

Since $p(x)$ can be written as $p(x) = Lh(x)g(x)$ with $L = \sqrt{2e\pi^{-1}}$, $h(x) = e^{-(x-1)^2/2}$ satisfying $0 \leq h(x) \leq 1$, and $g(x) = e^{-x} I(x \geq 0)$ being an Exponential($\theta = 1$) pdf, the resultant Rejection Sampling II algorithm is

Algorithm 7 [*A-R sampling algorithm for half-normal*]

1° Generate $X = x$ from Exp(1).

2° Generate $U = u$ from Unif$(0, 1)$ independently.

3° If $u \leq e^{-(x-1)^2/2}$, then deliver $Z = x$; otherwise go to 1°.

Example 3 (2)

It can be shown that the following algorithm also generates random numbers from the half-normal pdf $p(x)$ (why?):

Algorithm 8 [*A-R sampling algorithm for half-normal*]

1° Generate $U_1 = u_1$ and $U_2 = u_2$ from Unif$(0, 1)$ independently.

2° Set $x = -\ln u_1$ and $y = -\ln u_2$.

3° If $y \geq \frac{1}{2}(x-1)^2$, deliver $Z = x$; otherwise go to 1°.

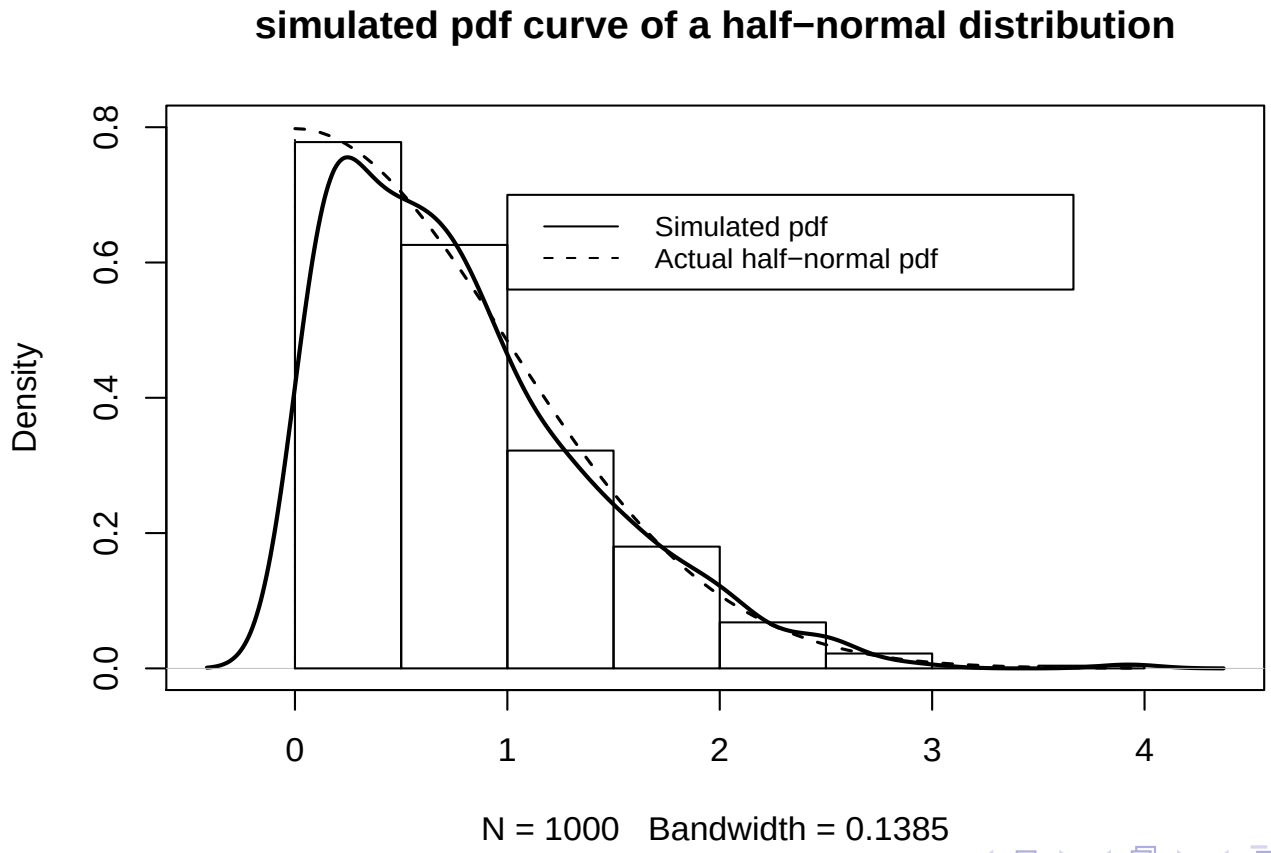
Example 3 (3)

An R program implementing Algorithm 8 is below:

```
myhalfnorm.f=function(n){
## This function is for generating a sample of "n"
## observations from half-normal distribution.
    Z <- rep(-1, n)
    for(i in 1:n) {repeat {x <- -log(runif(1))
            y <- -log(runif(1))
            if(y >= 0.5*(x-1)^2) {Z[i] <- x
                break } } }
    return(Z)}

set.seed(456)
mysample=myhalfnorm.f(1000)
plot(density(mysample),lwd=2, ylim=c(0,0.8),
    main="simulated pdf curve of a half-normal distribution")
curve(sqrt(2/pi)*exp(-0.5*x^2),from=0,to=4, add=T,lwd=1.5, lty=2)
legend(x=1.0, y=0.7, legend=c("Simulated pdf", "Actual half-normal pdf"),
    lty=c(1,2),cex=0.8)
lines(x=c(0,0),y=c(0,0.8),lty=3)
```

Example 3 (4)

A sample of 1000 random numbers are generated using Algorithm 8. The density curve of the sample is given below.



Example 4 (1)

Example 4 Consider a random sample $Y_n = (Y_1, \cdots, Y_n)$ with $Y_i \stackrel{d}{=} \text{Poisson}(e^{\beta x_i})$ and x_i being given, $i = 1, \cdots, n$. Let the prior pdf of β be $N(0,1)$. The posterior pdf of β can be found to be

$$p(\beta|\mathbf{y}_n, \mathbf{x}_n) = \frac{f(\mathbf{y}_n|\beta, \mathbf{x}_n)p(\beta)}{\int_B f(\mathbf{y}_n|\beta, \mathbf{x}_n)p(\beta)d\beta}$$

$$= \frac{(\prod_{i=1}^n y_i!)^{-1} e^{-\sum_{i=1}^n e^{\beta x_i}} e^{\beta \sum_{i=1}^n x_i y_i} (\sqrt{2\pi})^{-1} e^{-\beta^2/2}}{\int_{-\infty}^{\infty} (\prod_{i=1}^n y_i!)^{-1} e^{-\sum_{i=1}^n e^{\beta x_i}} e^{\beta \sum_{i=1}^n x_i y_i} (\sqrt{2\pi})^{-1} e^{-\beta^2/2} d\beta}$$

$$\stackrel{denoted}{=} L \times e^{-\sum_{i=1}^n e^{\beta x_i}} \times \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\beta - \sum_{i=1}^n x_i y_i)^2}.$$

The Bayes estimator of β is
$T = E(\beta|\mathbf{y}_n, \mathbf{x}_n) = \int_{-\infty}^{\infty} \beta p(\beta|\mathbf{y}_n, \mathbf{x}_n)d\beta$,
which has no closed form but can be evaluated using a Monte Carlo method.

Example 4 (2)

The following rejection sampling II algorithm may be used to generate random numbers from this posterior pdf:

Algorithm 9 [*A-R sampling algorithm for* $p(\beta|\mathbf{y}_n, \mathbf{x}_n)$]

1° Generate a β value from $N(\sum_{i=1}^{n} x_i y_i, 1)$.

2° Generate a u value from Unif$(0, 1)$ independently.

3° If $u \leq e^{-\sum_{i=1}^{n} e^{\beta x_i}}$, then deliver the generated β value; otherwise go to 1°.

Acceptance probability and efficiency of A-R sampling (1)

From 3° of the A-R sampling algorithm 2, it follows that the probability of a generated X being accepted as value of Z is

$$\tau = P(Y \leq h(X)) = \int_{-\infty}^{\infty} \int_{-\infty}^{h(x)} f(x, y) dy dx.$$

We call τ the **efficiency** of the A-R sampling algorithm. Further, denote by T the number of trials (rounds of executing steps 1° to 3°) to achieve an acceptance of $Z = x$. Then

$$P(T = t) = P(Y > h(X))^{t-1} P(Y \leq h(X)) = \tau(1-\tau)^{t-1}; \quad t = 1, 2, \cdots$$

Thus T follows a Geometric(τ) distribution with $E(T) = \tau^{-1}$ and $\mathrm{Var}(T) = \tau^{-2}(1 - \tau)$.

Acceptance probability and efficiency of A-R sampling (2)

It implies that the larger the τ is, the smaller is T expected to be; thus the higher is the efficiency.

- For the rejection sampling I algorithm, the efficiency

$$\tau = P\left(Y \leq \frac{p(X)}{M(X)}\right) = \int_{-\infty}^{\infty} \frac{p(x)}{M(x)} f(x)dx = \frac{1}{c} = \frac{1}{\int_{-\infty}^{\infty} M(x)dx}.$$

  Hence, in order that τ is high, the upper bound $M(x)$ should be as close to $p(x)$ as possible.

- For the rejection sampling II algorithm, the efficiency

$$\tau = P(U \leq h(X)) = \int_{-\infty}^{\infty} h(x)g(x)dx = \frac{1}{L}.$$

Application 3: Squeezed rejection sampling (1)

- Algorithm 3 step 3° requires one evaluation of $p(x)$ for every candidate draw X. This is computationally expensive if $p(x)$ is so .
- Simulation speed can be improved by introducing a **squeezing function** $s(x)$ such that $s(x) \leq p(x)$ on the support of $p(x)$.
- Specifically, we replace Algorithm 3 step 3° by first evaluating $y \leq \frac{s(x)}{M(x)}$ to decide acceptance then evaluating $y \leq \frac{p(x)}{M(x)}$ further if required.
- Note this change does not change the overall acceptance probability or efficiency τ comparing with Algorithm 3.

Application 3: Squeezed rejection sampling (2)

Using the same notation as in Algorithm 3 and the squeezing function $s(x)$, we have:

Algorithm 10 [*Squeezed Rejection Sampling*]

1° Generate $X = x$ from $f(x)$.

2° Generate $Y = y$ from Unif$(0, 1)$ independently.

3° If $y \leq \frac{s(x)}{M(x)}$, then deliver $Z = x$.

4° Otherwise, determine whether or not $y \leq \frac{p(x)}{M(x)}$. If yes, deliver $Z = x$; otherwise go to 1°.

Application 4: Adaptive rejection sampling (1)

1. The most challenging part of the rejection sampling is to find a suitable envelope $M(x)$ for the target pdf $p(x)$.
2. Gilks and Wild (1992, *Applied Statistics*) proposed an automatic envelope generation strategy for squeezed rejection sampling for a continuous, differentiable, log-concave density on a connected region of support.
3. Let $p(x)$ be the target pdf on a (possibly infinite) interval $[a, b]$. Let $\ell(x) = \log p(x)$.
4. The *adaptive rejection sampling* is initiated by evaluating ℓ and ℓ' at k points, $x_1 < x_2 < \cdots < x_k$. If $a = -\infty$, choose x_1 such that $\ell'(x_1) > 0$. If $b = +\infty$, choose x_k such that $\ell'(x_k) < 0$. Let $T_k = \{x_1, \cdots, x_k\}$.

Application 4: Adaptive rejection sampling (2)

5. Define the *rejection envelope* on T_k to be the exponential of the piecewise linear **upper hull** of ℓ formed by the tangents to ℓ at each point in T_k. Denote such upper hull of ℓ as $\tilde{e}_k$. It can be proved that

$$\tilde{e}_k(x) = \ell(x_i) + (x - x_i)\ell'(x_i) \quad \text{for } x \in [z_{i-1}, z_i]$$

where $z_i = \frac{\ell(x_{i+1}) - \ell(x_i) - x_{i+1}\ell'(x_{i+1}) + x_i\ell'(x_i)}{\ell'(x_i) - \ell'(x_{i+1})}$, $i = 1, \cdots, k-1$, is where the tangents at x_i and x_{i+1} intersect, and $z_0 = a$, $z_k = b$.

6. Once $\tilde{e}_k$ is obtained, the **rejection envelope function** for $p(x)$ is $e_k(x) = \exp\{\tilde{e}_k(x)\}$.

Application 4: Adaptive rejection sampling (3)

7. Define the *squeezing functitoon* on T_k to be the exponential of the piecewise linear **lower hull** of ℓ formed by the chords between adjacent points in T_k. Denote such lower hull of ℓ as $\tilde{s}_k$. It can be proved that

$$\tilde{s}_k(x) = \frac{(x_{i+1} - x)\ell(x_i) + (x - x_i)\ell(x_{i+1})}{x_{i+1} - x_i} \quad \text{for } x \in [x_i, x_{i+1}]$$

and $i = 1, \cdots, k-1$. When $x < x_1$ or $x > x_k$, let $\tilde{s}_k(x) = -\infty$.

8. Once $\tilde{s}_k$ is obtained, the **squeezing function** for $p(x)$ is $s_k(x) = \exp\{\tilde{s}_k(x)\}$.

Application 4: Adaptive rejection sampling (4)

Algorithm 11 [*Adaptive Rejection Sampling*]

1° Choose an initial grid T_k; create the envelope $e_k(x)$ and the squeezer $s_k(x)$, and the pdf $f_k(x)$ induced from $e_k(x)$.

2° Generate $X = x$ from $f_k(x)$.

3° Generate $Y = y$ from Unif$(0, 1)$ independently.

4° If $y \leq \frac{s_k(x)}{e_k(x)}$, deliver $Z = x$ and go to 7°.

5° If $y \leq \frac{p(x)}{e_k(x)}$, deliver $Z = x$ and update T_k by including x to get $T_{k+1} = \{x_1, \cdots, x_{k+1}\}$. Update $e_k(x)$, $s_k(x)$ and $f_k(x)$ to $e_{k+1}(x)$, $s_{k+1}(x)$ and $f_{k+1}(x)$ accordingly. Then go to 7°.

6° If $y > \frac{p(x)}{e_k(x)}$, go to 2°.

7° Return to 2°, with the updated T_{k+1}, $e_{k+1}(x)$, $s_{k+1}(x)$ and $f_{k+1}(x)$ if any, to generate new samples from $p(x)$. Continue until a sample of the desired size are generated from $p(x)$.

Application 4: Adaptive rejection sampling (5)

Remarks:

1. It is easy to see that Algorithm 11 still works when the target pdf has the form $p(x) = \tilde{c}\tilde{p}(x)$ with $\tilde{c}$ being an intractable constant.
2. When $\ell(x) = \log p(x)$ is not differentiable but $p(x)$ is log-concave with a connected support region, a derivative-free piecewise linear upper hull for ℓ can be found to be $e_k(x) = \exp\{\tilde{e}_k(x)\}$ with

$$\tilde{e}_k(x) = \begin{cases} \min\{L_{i-1}(x), L_{i+1}(x)\} & \text{for } x \in [x_i, x_{i+1}], \\ L_1(x) & \text{for } x < x_1, \\ L_{k-1}(x) & \text{for } x > x_k \end{cases}$$

where $L_i(x)$ is the straight line function connecting $(x_i, \ell(x_i))$ and $(x_{i+1}, \ell(x_{i+1}))$ for $i = 1, \cdots, k-1$, and $L_0(x) = L_k(x) = \infty$.

Importance sampling (1)

- Let $\mu = E[h(\mathbf{X})] = \int h(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}$ where $f(\mathbf{x})$ is the pdf of $\mathbf{X}$.
- We have seen a simple Monte Carlo estimator of μ is

$$\hat{\mu}_{MC} = \frac{1}{n} \sum_{i=1}^{n} h(\mathbf{X}_i),$$

  where $\mathbf{X}_1, \cdots, \mathbf{X}_n$ are iid samples from $f(\mathbf{x})$. We have been focusing on how to generate samples from $f(\mathbf{x})$.
- Actually better Monte Carlo estimators of $E[h(\mathbf{X})]$ can be derived in some situations. Here **better** means more precise, smaller variance or quicker convergence.

Importance sampling (2)

Fundamental equations for importance sampling:

$$\mu = E[h(\mathbf{X})] = \int h(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = \int h(\mathbf{x}) \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} \quad (1)$$

$$\mu = E[h(\mathbf{X})] = \frac{\int h(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}}{\int f(\mathbf{x}) d\mathbf{x}} = \frac{\int h(\mathbf{x}) \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x}) d\mathbf{x}}{\int \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x}) d\mathbf{x}} \quad (2)$$

where $g(\mathbf{x})$ is another pdf of $\mathbf{X}$ with its support covering that of $h(\mathbf{x})f(\mathbf{x})$ in case (1) and that of $f(\mathbf{x})$ in case (2). We call $g(\mathbf{x})$ an **importance sampling function** or **envelope**.

Importance sampling (3)

- Equation (1) suggests estimating $E[h(\mathbf{X})]$ using iid samples $\mathbf{X}_1, \cdots, \mathbf{X}_n$ generated from $g(\mathbf{x})$ as following

$$\tilde{\mu}_{IS} = \frac{1}{n} \sum_{i=1}^{n} h(\mathbf{X}_i) \tilde{w}(\mathbf{X}_i)$$

  where $\tilde{w}(\mathbf{X}_i) = \frac{f(\mathbf{X}_i)}{g(\mathbf{X}_i)}$ are *unstandardised weights*, also called *importance ratios*.

- The estimator $\tilde{\mu}_{IS}$ is preferred if it is difficult to sample from $f(\mathbf{x})$ but easy to sample from $g(\mathbf{x})$.

Importance sampling (4)

- Equation (2) suggests estimating $E[h(\mathbf{X})]$ using iid samples $\mathbf{X}_1, \cdots, \mathbf{X}_n$ generated from $g(\mathbf{x})$ as following

$$\hat{\mu}_{IS} = \sum_{i=1}^{n} h(\mathbf{X}_i) w(\mathbf{X}_i)$$

where $w(\mathbf{X}_i) = \frac{\tilde{w}(\mathbf{X}_i)}{\sum_{j=1}^{n} \tilde{w}(\mathbf{X}_j)} = \frac{\frac{f(\mathbf{X}_i)}{g(\mathbf{X}_i)}}{\sum_{j=1}^{n} \frac{f(\mathbf{X}_j)}{g(\mathbf{X}_j)}}$ are *standardised weights*.

- It is computationally more feasible to compute $w(\mathbf{X}_i)$ than $\tilde{w}(\mathbf{X}_i)$ when $f(\mathbf{x})$ is known only up to a proportionality constant.

Importance sampling (5)

- If we choose an importance sampling function $g(\mathbf{x})$ such that the weights $\tilde{w}(\mathbf{x})$ or $\hat{w}(\mathbf{x})$ are largely uniform over those influential $\mathbf{x}$ points, and do not have large values over those insignificant points, the resultant estimators $\tilde{\mu}_{IS}$ or $\hat{\mu}_{IS}$ will tend to have smaller variance and to converge more quickly than $\hat{\mu}_{MC}$.

Example 5. Estimate $\int_0^1 [\cos(20x) + \sin(50x)]^2 dx$ (1)

Example 5 Let $h(x) = [\cos(20x) + \sin(50x)]^2$. It can be found that
$\int_0^1 h(x)dx = \frac{22}{21} + \frac{1}{80}\sin(40) - \frac{1}{70}\cos(70) - \frac{1}{30}\cos(30) - \frac{1}{200}\sin(100)$
using a CAS like Maple or Mathematica. Now we see how well this integral can be approximated numerically and by a Monte Carlo estimator.

- A numerical approximation can be done using the `integrate` function in `R`, which is really good in this example.
- A Monte Carlo estimate can be obtained by generating random samples from Unif$(0, 1)$ and calculate $\hat{\mu}_{MC}$, which becomes very good when the size of the generated sample is bigger than 8000.

Example 5. Estimate $\int_0^1 [\cos(20x) + \sin(50x)]^2 dx$ (2)

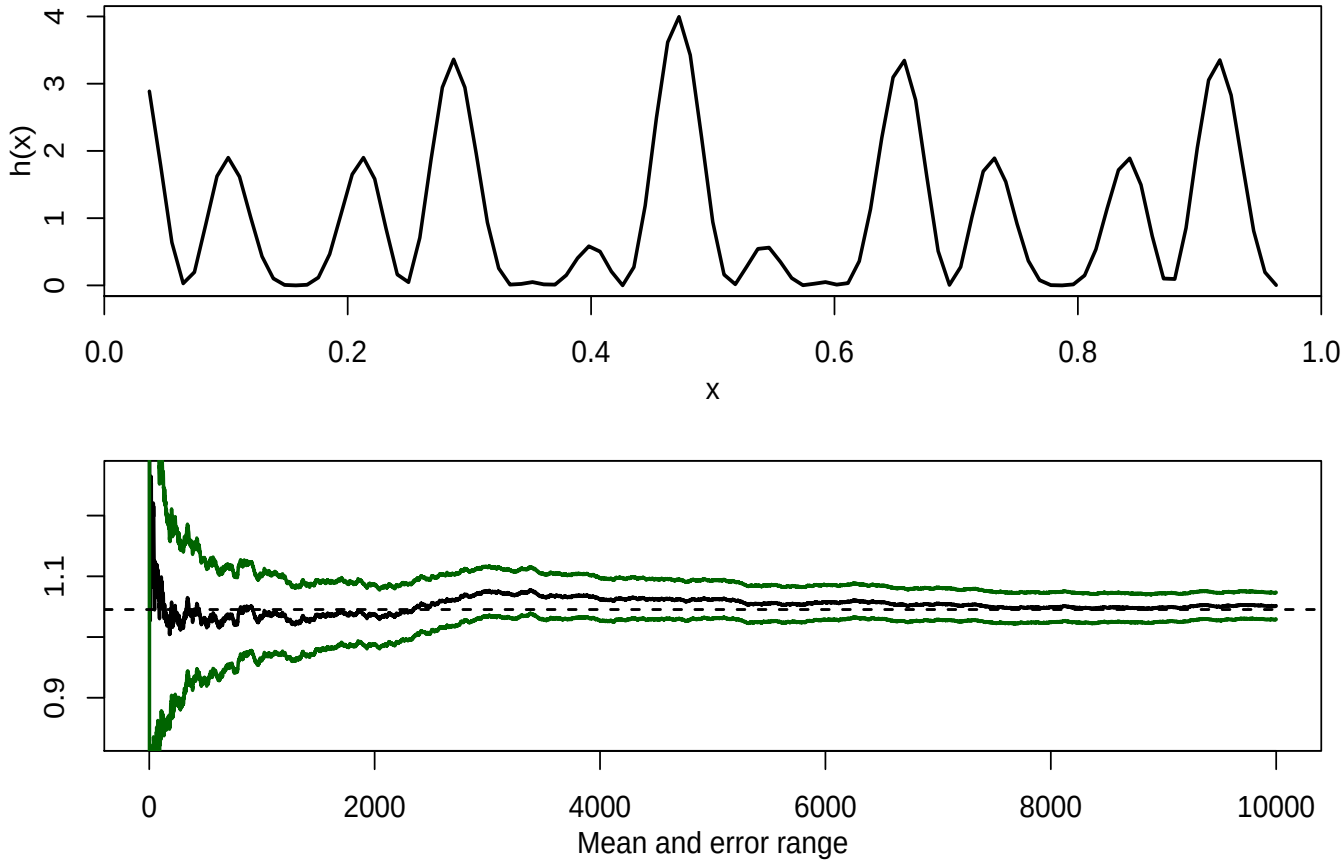


Figure: The top plot is the function $h(x)$. The bottom plot gives the sequence of Monte Carlo approximations of $\int_0^1 h(x)dx \pm$ two Monte Carlo standard errors band, with the dash line indicating the actual value of the integral.

Example 5. Estimate $\int_0^1 [\cos(20x) + \sin(50x)]^2 dx$ (3)

```
> h=function(x){(cos(20*x)+sin(50*x))^2}
> par(mgp=c(2,1,0),mar=c(3.5,3,1,1),mfrow=c(2,1))
> curve(h, xlab="Function", ylab="h(x)",lwd=2)
> integrate(h,0,1)

1.045276 with absolute error < 2.1e-10

> set.seed(123)
> x=h(runif(10^4))
> estint=cumsum(x)/(1:10^4)
> esterr=sqrt(cumsum((x-estint)^2))/(1:10^4)
> plot(estint, xlab="Mean and error range",type="l",lwd=
+ 2,ylim=mean(x)+20*c(-esterr[10^4],esterr[10^4]),ylab="")
> lines(estint+2*esterr,col="darkgreen",lwd=2)
> lines(estint-2*esterr,col="darkgreen",lwd=2)
> abline(a=1.045276,b=0, lwd=1.5, lty=2)
> estint[c(8000,9000,10000)]

#[1] 1.048807 1.048723 1.051272
```

Example 6. Monte Carlo $N(0,1)$ probabilities (1)

Example 6 Given a random sample $(x_1, \cdots, x_n)$ generated from $N(0,1)$, a Monte Carlo estimator of $\Phi(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2} dy$ is

$$\hat{\Phi}(t) = \frac{1}{n} \sum_{i=1}^{n} I(x_i \leq t),$$

with (exact) variance $n^{-1}\Phi(t)[1 - \Phi(t)]$.

The following R code gives evolution of Monte Carlo approximations of $\Phi(t)$ for $t = 0.00,\ 0.67,\ 0.84,\ 1.28,\ 1.64,\ 2.33,\ 3.09,\ 3.72$ as the simulation sample size goes from 10^2 to 10^8. It seems the approximations are very close to their corresponding exact values and have small variances when $n \geq 10^4$.

Example 6. Monte Carlo $N(0,1)$ probabilities (2)

```
set.seed(456)
x=rnorm(10^8) #whole sample
bound=qnorm(c(.5,.75,.8,.9,.95,.99,.999,.9999))
res=matrix(0,ncol=8,nrow=7)
for (i in 2:8) #lengthy loop!!
for (j in 1:8)
res[i-1,j]=mean(x[1:10^i]<bound[j])
MCnormal=matrix(as.numeric(format(res,digi=4)),ncol=8)
rownames(MCnormal)=c("10^2","10^3","10^4","10^5","10^6","10^7","10^8")
colnames(MCnormal)=format(bound,digi=2)
MCnormal
```

```
       0.00   0.67   0.84   1.28   1.64   2.33   3.09   3.72
10^2 0.4900 0.6900 0.7600 0.8400 0.9200 1.0000 1.0000 1.0000
10^3 0.4820 0.7210 0.7780 0.8950 0.9450 0.9910 1.0000 1.0000
10^4 0.4997 0.7407 0.7939 0.8968 0.9470 0.9882 0.9987 0.9997
10^5 0.4990 0.7450 0.7952 0.8970 0.9477 0.9895 0.9989 0.9998
10^6 0.4997 0.7491 0.7991 0.8996 0.9499 0.9901 0.9990 0.9999
10^7 0.5002 0.7501 0.8000 0.9000 0.9499 0.9900 0.9990 0.9999
10^8 0.5001 0.7500 0.8000 0.9000 0.9500 0.9900 0.9990 0.9999
```

Example 7. Monte Carlo $N(0,1)$ tail probability (1)

It can be seen that it requires a huge sample in order to get a good approximation of $\Phi(t)$ using naïve Monte Carlo estimator $\hat{\mu}_{MC}$. Using proper importance sampling (IS) function $g(x)$ and $\hat{\mu}_{IS}$ or $\tilde{\mu}_{IS}$ would improve the efficiency.

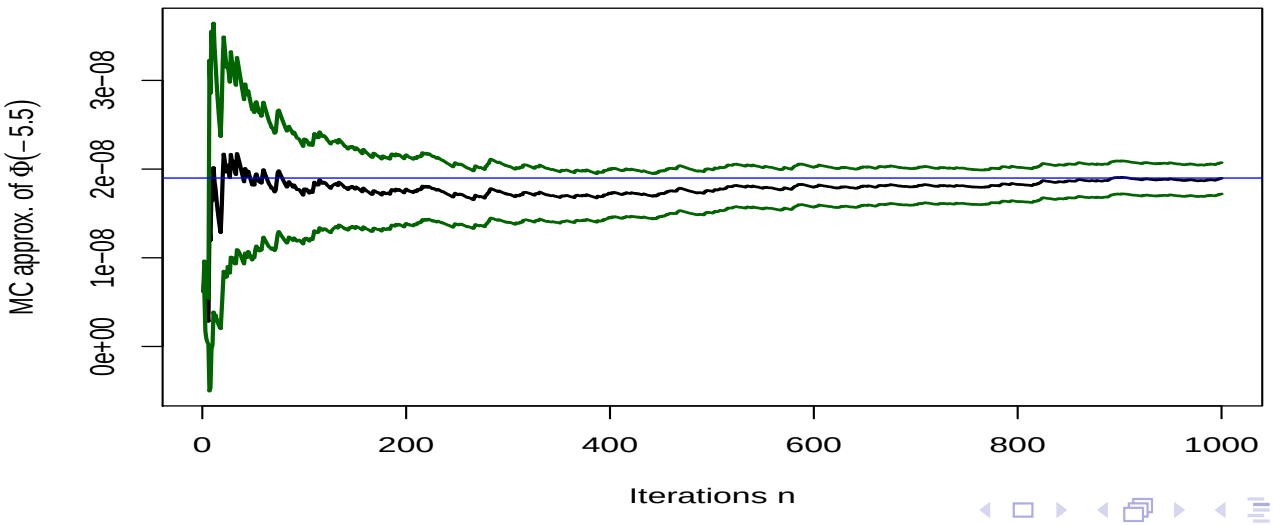
Example 7 It can be found that $\Phi(-5.5) = P(Z \leq -5.5) = P(Z \geq 5.5) = e^{-17.77938} = 1.898956 \times 10^{-8}$. If we use $\hat{\Phi}(-5.5) = \frac{1}{n}\sum_{i=1}^{n} I(x_i \geq 5.5)$, where x_i's are iid $N(0,1)$ samples, to estimate $\Phi(-5.5)$, n has to be huge to get a reasonable estimate.

Example 7. Monte Carlo $N(0,1)$ tail probability (2)

We now use importance sampling method to estimate $\Phi(-5.5) = 1 - \Phi(5.5)$. The IS function $g(x)$ is chosen to be Exp(1) shifted up by 5.5: $g(x) = e^{x-5.5}$, $x \geq 5.5$.

The IS estimator is

$$\tilde{\Phi}_{IS}(-5.5) = \frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)}{g(X_i)} = \frac{1}{n}\sum_{i=1}^{n}\frac{e^{-X_i^2/2+X_i-5.5}}{\sqrt{2\pi}}.$$



Example 7. Monte Carlo $N(0,1)$ tail probability (3)

The corresponding R code is

```
> set.seed(123456)
> Nsim=10^3
> x=rexp(Nsim)+5.5
> weit=dnorm(x)/dexp(x-5.5)
> estprob=cumsum(weit)/(1:Nsim)
> esterr=sqrt(cumsum((weit-estprob)^2))/(1:Nsim)
> plot(estprob,type="l",lwd=2,xlab="Iterations",
+ ylab=expression(paste("MC approx. of ", Phi(-5.5))))
> lines(estprob+2*esterr,col="darkgreen",lwd=2)
> lines(estprob-2*esterr,col="darkgreen",lwd=2)
> abline(a=pnorm(-5.5),b=0, col="blue")
> mean(weit)
[1] 1.896130e-08
> pnorm(-5.5)
[1] 1.898956e-08
> pnorm(-5.5, log=T)
[1] -17.77938   #exp(-17.77938)=1.898956e-08
```

Sampling importance resampling (1)

- The importance sampling technique does more than approximating integrals. It can also be used to simulate samples approximately from a complex target pdf $f(\mathbf{x})$.
- The associated technique is called **sampling importance resampling (SIR)**.
- The idea behind SIR is using simple and suitable importance sampling function $g(\mathbf{x})$ to generate samples; calculating the importance ratios or weights; then resampling samples from the generated samples according to the weights.

Sampling importance resampling (2)

Algorithm 12 [*Sampling importance resampling*]

1° Sample candidates $\mathbf{Y}_1, \cdots, \mathbf{Y}_m$ i.i.d. from $g(\mathbf{x})$.

2° Calculate the standardised importance weights $w(\mathbf{Y}_1), \cdots, w(\mathbf{Y}_m)$.

3° Resample $\mathbf{X}_1, \cdots, \mathbf{X}_n$ from $\mathbf{Y}_1, \cdots, \mathbf{Y}_m$ with replacement with probabilities $w(\mathbf{Y}_1), \cdots, w(\mathbf{Y}_m)$.

Using the strong law of large numbers and Lebesgue's dominated convergence theorem, it is not difficult to prove that the pdf for the independent samples generated by SIR converges to the target pdf $f(\mathbf{x})$ as $m \to \infty$.

Adaptive importance, bridge, and path sampling

Sometimes, one initially may be able to specify only a very poor importance sampling envelope. This may occur, e.g., when the target pdf has support nearly limited to a lower-dimensional space due to strong dependencies between variables not well understood by the analyst. In such situations, it is possible to adapt the importance sampling envelope.

Adaptive importance, bridge, and path sampling are various techniques developed for dealing with such and related problems. Details can be found in A. Gelman and X.L. Meng (*Statistical Science*, **13**:163-185, 1998).

Sequential importance sampling

Sequential importance sampling (SIS) is a strategy for constructing high-dimensional envelopes one dimension at a time.

Details can be found in J.S. Liu (*Monte Carlo strategies in scientific computing*, Springer-Verlag, 2001).

Introduction to Gibbs sampler

- In E-step of EM, in Bayes estimation, or more generally in a statistical inference task, the probability distribution of a statistic that we need to use may be multivariate and be difficult to derive analytically. Hence we want to have a Monte Carlo method to approximate such a probability distribution so that the associated inference task can be carried out based on simulations.
- Gibbs sampler is a random vector generation method which does not require the complete information of the target multivariate probability distribution. Instead, it requires only the information of a set of the associated conditional distributions. Gibbs sampler will be computationally feasible if the conditional distributions are ready to be simulated.

Introduction to Gibbs sampler

- Suppose we want to generate a vector $\mathbf{u} = (u_1, \cdots, u_K)$ from the random vector $\mathbf{U} = (U_1, \cdots, U_K)$ which has a joint pdf $f(\mathbf{u})$.
- Suppose the pdf $f(\mathbf{u})$ has a very complicated form. But for each $k = 1, \cdots, K$, the conditional pdf of U_k given $\mathbf{U}_{-k} = (U_1, \cdots, U_{k-1}, U_{k+1}, \cdots, U_K)$ is known and relatively easy to simulate. We use

$$f(u_k|\mathbf{u}_{-k}) \equiv f(u_k|u_1, \cdots, u_{k-1}, u_{k+1}, \cdots, u_K)$$

  to denote such a conditional pdf.
- Then the Gibbs sampler used to generate **one** observation of $\mathbf{U}$ can be described as following:

Gibbs sampler (1)

Algorithm 13 [*Gibbs sampler*]

1° Arbitrarily generate/assign an initial vector $\mathbf{u}^{(0)} = (u_1^{(0)}, \cdots, u_K^{(0)})$ from the support of $f(\mathbf{u})$.

2° Generate a value $u_1^{(1)}$ from $f(u_1 | u_2 = u_2^{(0)}, \cdots, u_K = u_K^{(0)})$;
then generate a value $u_2^{(1)}$ from
$f(u_2 | u_1 = u_1^{(1)}, u_3 = u_3^{(0)}, \cdots, u_K = u_K^{(0)})$;
continue until generate a value $u_{K-1}^{(1)}$ from
$f(u_{K-1} | u_1 = u_1^{(1)}, \cdots, u_{K-2} = u_{K-2}^{(1)}, u_K = u_K^{(0)})$,
and generate a value $u_K^{(1)}$ from
$f(u_K | u_1 = u_1^{(1)}, \cdots, u_{K-1} = u_{K-1}^{(1)})$.
The generated values are delivered as $\mathbf{u}^{(1)} = (u_1^{(1)}, \cdots, u_K^{(1)})$.

Gibbs sampler (2)

Algorithm 3.13 [*Gibbs sampler (continued)*]

3° For $j = 1, 2, \cdots$, do the following:
Generate a value $u_1^{(j)}$ from
$f(u_1|u_2 = u_2^{(j-1)}, \cdots, u_K = u_K^{(j-1)})$; then generate a value $u_2^{(j)}$ from $f(u_2|u_1 = u_1^{(j)}, u_3 = u_3^{(j-1)}, \cdots, u_K = u_K^{(j-1)})$;
continue until generate a value $u_{K-1}^{(j)}$ from
$f(u_{K-1}|u_1 = u_1^{(j)}, \cdots, u_{K-2} = u_{K-2}^{(j)}, u_K = u_K^{(j-1)})$,
and generate a value $u_K^{(j)}$ from
$f(u_K|u_1 = u_1^{(j)}, \cdots, u_{K-1} = u_{K-1}^{(j)})$.
The generated values are delivered as $\mathbf{u}^{(j)} = (u_1^{(j)}, \cdots, u_K^{(j)})$.

Note the conditioning is always based on the latest values of $(u_1, \cdots, u_K)$.

Remarks on Gibbs sampler

- Using the theory of Markov chain, it can be shown that

$$\mathbf{u}^{(j)} \xrightarrow{d} f(\mathbf{u}) \quad \text{as } j \to \infty$$

  under fairly general conditions. Details are not pursued here.
- This implies that $\mathbf{u}^{(j)}$ can be roughly regarded as an observation of $\mathbf{U}$ from pdf $f(\mathbf{u})$ when j is sufficiently large. Empirical methods are available (details not pursued here) to determine how large j should be.
- In practice, we usually generate a long sequence $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(m)}, \mathbf{u}^{(m+1)}, \cdots, \mathbf{u}^{(m+J)}$ using the Gibbs sampler; then ignore the **burn-in** sequence $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(m)}$ and use only the part $\mathbf{u}^{(m+1)}, \cdots, \mathbf{u}^{(m+J)}$ as a random sample from $f(\mathbf{u})$.
- $\mathbf{u}^{(m+1)}, \cdots, \mathbf{u}^{(m+J)}$ are not independent of each other but constitute a Markov chain.

Example 8(1)

Example 8 (**A Bayesian hierarchical model**) Eggs of certain species of insects hatch under appropriate conditions. Suppose there are N eggs of this species of insects in a particular area, and X eggs of them will hatch. Also let p be the probability that such an egg will hatch.

1. A Bayesian approach may be used to model (X, p, N), in which one can reasonably assume $N \overset{d}{=} \text{Poisson}(\lambda)$, $p \overset{d}{=} \text{Beta}(a, b)$ and $(X|p, N) \overset{d}{=} \text{binomial}(N, p)$. Suppose there was a previous study from which we can reasonably assume $N \overset{d}{=} \text{Poi}(16)$ and $p \overset{d}{=} \text{Beta}(2, 4)$. Find the joint pdf of (X, p, N) and the marginal pdf of X.
2. Derive a Gibbs sampler for generating (X, p, N) and X.
3. Implement the algorithm in R and check its performance.

Example 8(2)

When $N \overset{d}{=} \text{Poi}(16)$ and $p \overset{d}{=} \text{Beta}(2,4)$, the joint pdf of (X,p,N) is

$$\begin{aligned} f(x,p,N) &= f(x|p,N)\cdot f(p)\cdot f(N) \\ &= \binom{N}{x} p^x(1-p)^{N-x}\cdot\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}p^{a-1}(1-p)^{b-1}\cdot\frac{\lambda^N}{N!}e^{-\lambda} \\ &= 20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}; \\ &\quad x=0,1,\cdots,N;\ 0\le p\le 1;\ N=0,1,2,\cdots. \end{aligned}$$

The marginal pdf of X is

$$f(x)=\sum_{N=x}^{\infty}\int_0^1 f(x,p,N)dp=\sum_{N=x}^{\infty}\frac{20e^{-16}(x+1)(N-x+3)(N-x+2)(N-x+1)16^N}{(N+5)}$$

which is difficult to be further simplified.

Example 8(3)

But it is not difficult to find the full conditional pdf's:

$$(X|p,N) \stackrel{d}{=} \text{bin}(N,p), \qquad (p|x,N) \stackrel{d}{=} \text{Beta}(x+2, N-x+4),$$
$$\text{and} \qquad (N-x|x,p) \stackrel{d}{=} \text{Poi}(16(1-p)).$$

Proof:

$$\begin{aligned} f(p|x,N) &= \frac{f(x,p,N)}{f(x,N)} = \frac{20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}}{\int_0^1 20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}dp} \\ &= \frac{p^{x+1}(1-p)^{N-x+3}}{\int_0^1 p^{x+1}(1-p)^{N-x+3}dp} = \frac{\Gamma(N+6)}{\Gamma(x+2)\Gamma(N-x+4)}p^{x+1}(1-p)^{N-x+3} \end{aligned}$$

which is a pdf of $\text{Beta}(x+2, N-x+4)$.

Example 8(4)

$$
\begin{aligned}
f(N|x,p) &= \frac{f(x,p,N)}{f(x,p)} = \frac{20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}}{\sum_{N=x}^{\infty}20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}} \\
&= \frac{\binom{N}{x}[16(1-p)]^N\frac{1}{N!}}{\sum_{\tilde{N}=x}^{\infty}\binom{\tilde{N}}{x}\frac{16^{\tilde{N}}}{\tilde{N}!}(1-p)^{\tilde{N}}} = \frac{\frac{1}{(N-x)!}[16(1-p)]^N}{\sum_{\tilde{N}=x}^{\infty}\frac{1}{(\tilde{N}-x)!}[16(1-p)]^{\tilde{N}}} \\
&= \frac{[16(1-p)]^{N-x}}{(N-x)!}e^{-16(1-p)}; \quad N = x, x+1, x+2, \cdots .
\end{aligned}
$$

This implies that $(N-x|x,p) \stackrel{d}{=} \text{Poisson}(16(1-p))$. □

Now a sample of (X,p,N) can be generated using the following Gibbs sampler algorithm

Example 8(5)

Algorithm 14 [*Gibbs sampler algorithm for* (X, p, N)]

1° Arbitrarily choose an initial vector, e.g. $(x^{(0)}, p^{(0)}, N^{(0)}) = (8, 0.5, 16)$.

2° $(x^{(1)}, p^{(1)}, N^{(1)})$ is obtained by:
generating $x^{(1)}$ from $\text{Bin}(N^{(0)}, p^{(0)})$;
generating $p^{(1)}$ from $\text{Beta}(x^{(1)} + 2, N^{(0)} - x^{(1)} + 4)$; and
generating an $N^{(1)}$ from $\text{Poi}(16(1 - p^{(1)})) + x^{(1)}$.

3° $(x^{(j)}, p^{(j)}, N^{(j)})$, $j = 1, 2, \cdots$, is obtained by:
generating $x^{(j)}$ from $\text{Bin}(N^{(j-1)}, p^{(j-1)})$;
generating $p^{(j)}$ from $\text{Beta}(x^{(j)} + 2, N^{(j-1)} - x^{(j)} + 4)$; and
generating an $N^{(j)}$ from $\text{Poi}(16(1 - p^{(j)})) + x^{(j)}$.

Once a sample of (X, p, N) is obtained, a sample of X can be read off from it.

Example 8(6)

The above algorithm was implemented in R. Then a sample of 1000 (X, p, N) samples are generated, and the simulated marginal pdf's of X, p and N are obtained. The results are shown in the following.

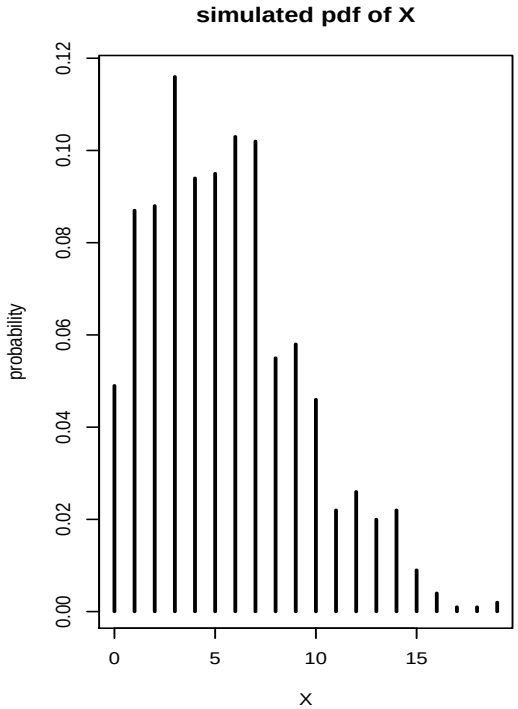
```
gibbs.f2 = function(x0, p0, N0, m, J){
# Single path method for generating m samples of (x,p,N) values by the Gibbs
# sampler. In total J+m samples are generated but the first J are discarded.
# (x0,p0,N0) is the initial value. If (x0,p0,N0) is not given, one can use
# x0=rbinom(1,16, 0.5), p0=rbeta(1,2,4) and N0=rpois(1,16) to generate it.
    x.seq <- p.seq <- N.seq <- rep(-1, J+m+1)
    x.seq[1] <- x0; p.seq[1] <- p0; N.seq[1] <- N0
    for(j in 2:(J+m+1)) {x.seq[j] <- rbinom(1, N.seq[j-1], p.seq[j-1])
       p.seq[j] <- rbeta(1, (x.seq[j] + 2), (N.seq[j-1] - x.seq[j] + 4))
       N.seq[j] <- rpois(1, 16 * (1 - p.seq[j])) + x.seq[j]}
    result <- list(X=x.seq[(J+2):(J+m+1)],p=p.seq[(J+2):(J+m+1)],
                   N=N.seq[(J+2):(J+m+1)])
    result}
```

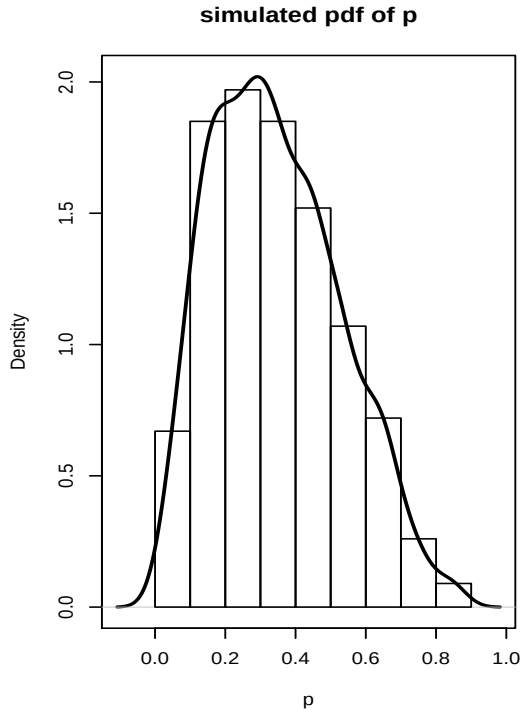
Example 8(7)

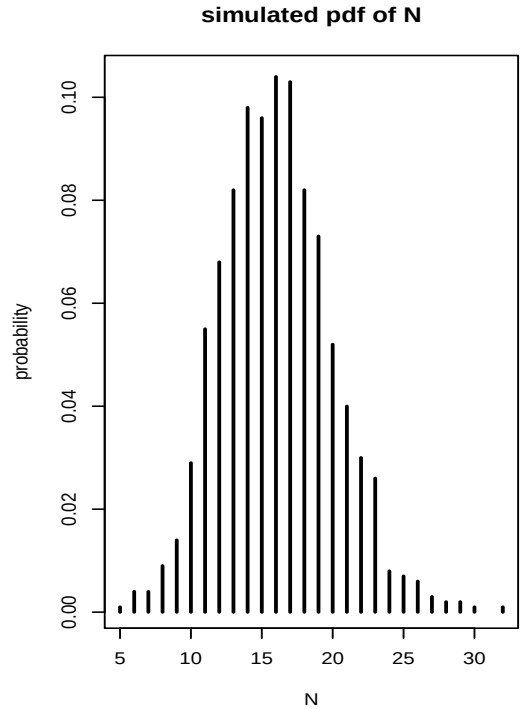
```
> set.seed(456)
> m=1000
> gibbsam2=gibbs.f2(8, 0.5, 16, m, 100)
> par(mfrow=c(1,3))
> plot(sort(unique(gibbsam2$X)), as.numeric(table(gibbsam2$X)/m),
       type='h',lwd=2, xlab="X", ylab="probability", main="simulated pdf of X")
> plot(density(gibbsam2$p), main="simulated pdf of p", xlab="p", lwd=2)
> plot(sort(unique(gibbsam2$N)), as.numeric(table(gibbsam2$N)/m),
       type='h',lwd=2, xlab="N", ylab="probability", main="simulated pdf of N")

> table(gibbsam2$N)/m    #Gives the empirical pdf of N
    5     6     7     8     9    10    11    12    13    14
0.001 0.004 0.004 0.009 0.014 0.029 0.055 0.068 0.082 0.098
   15    16    17    18    19    20    21    22    23    24
0.096 0.104 0.103 0.082 0.073 0.052 0.040 0.030 0.026 0.008
   25    26    27    28    29    30    32
0.007 0.006 0.003 0.002 0.002 0.001 0.001
```

Example 8(8)

```
> table(gibbsam2$X)/m    #Gives the empirical pdf of X
    0     1     2     3     4     5     6     7     8     9
0.049 0.087 0.088 0.116 0.094 0.095 0.103 0.102 0.055 0.058
   10    11    12    13    14    15    16    17    18    19
0.046 0.022 0.026 0.020 0.022 0.009 0.004 0.001 0.001 0.002
```



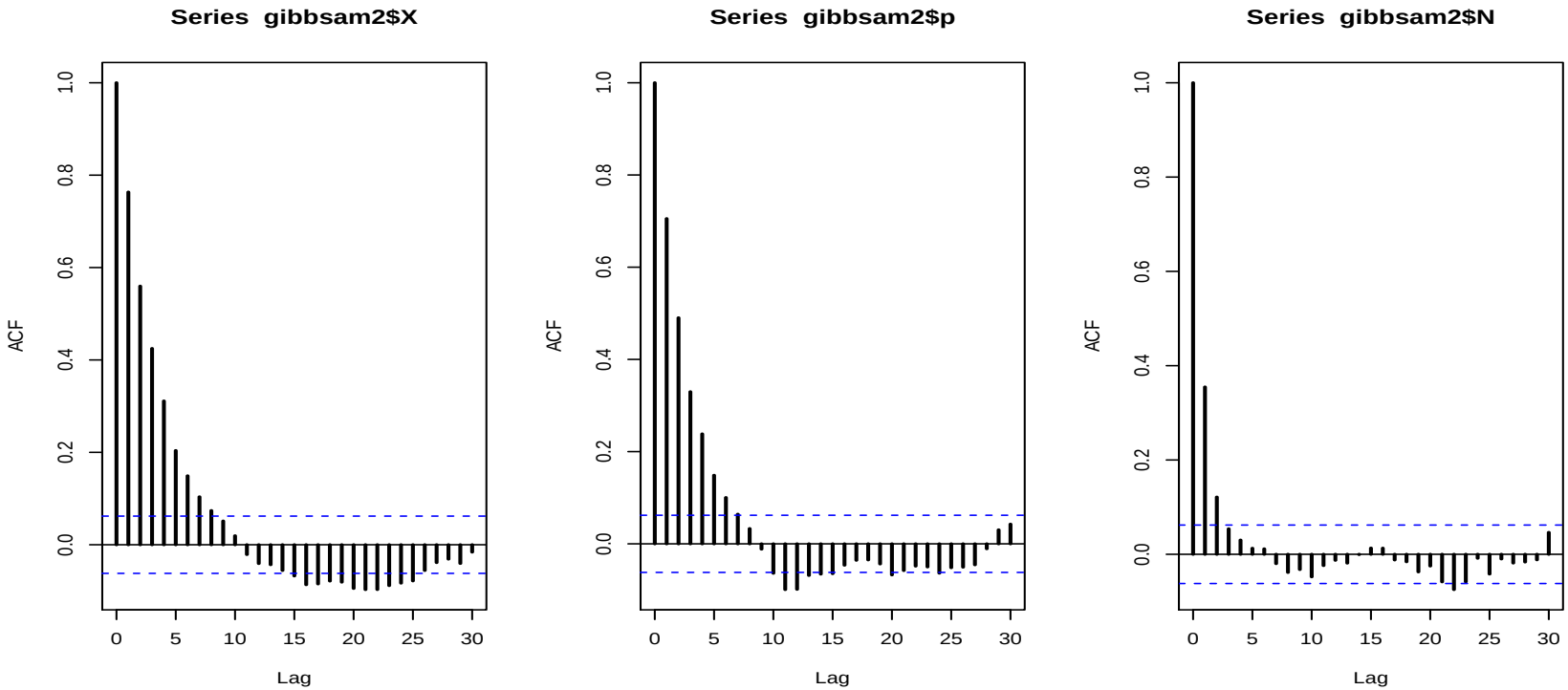




Example 8(9)

The auto-correlation functions of the generated samples of X, p and N are plotted below, showing that they are not independent samples.

acf(gibbsam2$X, lwd=2); acf(gibbsam2$p, lwd=2); acf(gibbsam2$N, lwd=2)



Monte Carlo estimator based on a Markov chain (1)

- Random samples generated by a Gibbs sampler algorithm are not independent of each other, but constitute a Markov chain. **Q**: Can we use the generated Markov chain to construct a Monte Carlo estimator? And how good is such an estimator?
- A sequence of random variables $\{X^{(t)}, t = 0, 1, 2, \cdots\}$ is called a **Markov chain** if for any nonnegative integer sequence $t_0 < t_1 < \cdots < t_m < t$, it satisfies that

$$P(X^{(t)} \in A | x^{(t_0)}, x^{(t_1)}, \cdots, x^{(t_m)}) = P(X^{(t)} \in A | x^{(t_m)})$$

  for any region A in the σ-field $\mathcal{F}$ spanned by $\mathcal{S}$, where the *state space* $\mathcal{S}$ is the set of all possible values of $X^{(t)}$.
- The Markov chain is *time-homogeneous* if the *transition (kernel) probability function* $P(X^{(t+1)} \in A | X^{(t)} = x) = P(x, A)$ does not depend on t.

Monte Carlo estimator based on a Markov chain (2)

- A region $A \in \mathcal{F}$ is called a *recurrent* region if the chain returns to it with probability 1.
- A region $A \in \mathcal{F}$ for which the expected time until recurrence is finite is called *nonnull*.
- A Markov chain is *irreducible* if any region A can be reached from any state s in a finite number of steps for all $A \in \mathcal{F}$ and $s \in \mathcal{S}$. Namely, for each A and s there must exists $m > 0$ such that $P(X^{(m+n)} \in A | X^{(n)} = s) > 0$.
- A Markov chain is *periodic* if it can visit certain regions of the state space only at certain regularly spaced time intervals. Region A has period d if the probability of going from A to A in n steps is 0 for all n not divisible by d. If every region in a Markov chain has period 1, then the chain is called *aperiodic*.

Monte Carlo estimator based on a Markov chain (3)

- A Markov chain is *ergodic* if it is irreducible, aperiodic, and all its regions are nonnull and recurrent.
- A probability distribution function π^* is said to be **stationary** or **invariant** if

$$\pi^*(dy) = \int_{\mathcal{S}} P(x, dy)\pi^*(dx); \quad P(x, dy) \text{ being the transition prob. function.}$$

Note $\pi^*(dy) = \pi(y)dy$ if Y is a continuous r.v. with pdf $\pi(y)$. Accordingly $P(x, dy) = p(x, y)dy$ with $p(x, y)$ being the transition pdf. We treat dy as a small neighbourhood of y.

- If a Markov chain is ergodic, it will have a unique stationary distribution π^* and

$$\lim_{m\to\infty} P(X^{(m+t)} \in dy | X^{(t)} = x) \equiv \lim_{m\to\infty} P^{(m)}(x, dy) = \pi^*(dy),$$

$P^{(m)}(x, dy)$ is the *m-step transition (kernel) probability*

Monte Carlo estimator based on a Markov chain (4)

Theorem (Ergodic Theorem)

If $\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(n)}, \cdots$ *are realisations from an ergodic Markov chain with stationary distribution* π^**, then*

1. $\mathbf{X}^{(n)}$ *converges in distribution to* π^* *as* $n \to \infty$*;*
2. *for any function* h*,*

$$\frac{1}{n}\sum_{t=1}^{n} h(\mathbf{X}^{(t)}) \to E_{\pi^*}[h(\mathbf{X})] \quad \text{almost surely as } n \to \infty.$$

This is one form of the *ergodic theorem*, a generalisation of the strong law of large numbers. It implies that Monte Carlo estimation can be done based on Markov chain realisations with good approximation results.

Rao-Blackwellised Monte Carlo estimator

- Suppose n realisations of a random vector $\mathbf{X} = (X_1, \cdots, X_k)$ with joint pdf $f(x_1, \cdots, x_k)$ have been generated, which are denoted as $\mathbf{x}_1 = (x_{11}, \cdots, x_{k1}), \cdots, \mathbf{x}_n = (x_{1n}, \cdots, x_{kn})$. Suppose we want to estimate $\theta = E_f[h(\mathbf{X})]$.
- Naturally, θ can be estimated by $\hat{\theta}_{MC} = \frac{1}{n}\sum_{j=1}^{n} h(\mathbf{x}_j)$.
- Suppose $\mathbf{S} = \mathbf{s}(\mathbf{X})$ is a sufficient statistic for θ based on the sample $\mathbf{X}$, and it is easy to compute $E_f[h(\mathbf{X})|\mathbf{S}] \equiv h^*(\mathbf{S})$. Then we can use the following to estimate θ:

$$\hat{\theta}_{RB} = \frac{1}{n}\sum_{j=1}^{n} h^*(\mathbf{s}_j) \quad \text{where } \mathbf{s}_j = \mathbf{s}(\mathbf{x}_j).$$

- By Rao-Blackwell Theorem, $\text{var}(\hat{\theta}_{RB}) \leq \text{var}(\hat{\theta}_{MC})$. We call $\hat{\theta}_{RB}$ the *Rao-Blackwellised* Monte Carlo estimator of θ.

Rao-Blackwell Theorem

Theorem (Rao-Blackwell)

Suppose $\mathbf{S} = (S_1, \cdots, S_k)$ *is jointly sufficient for* θ*. If* T *is an unbiased estimator of* $\tau(\theta)$*, then*

1. $T^* = E(T|\mathbf{S})$ *is also an unbiased estimator of* $\tau(\theta)$*;*
2. T^* *is a function of* $\mathbf{S}$*; and*
3. $Var(T^*) \leq Var(T)$ *for every* θ*, and* $Var(T^*) < Var(T)$ *for some* θ *unless* $T^* = T$ *with probability 1.*

Example 9. (1)

Example 9 (**Example 8 continued**) Use the $m = 1000$ samples $(X_1, p_1, N_1), \cdots (X_m, p_m, N_m)$ generated from Example 8 (refer to the R outputs there and following) to

1. estimate $\mu = E(X)$;
2. estimate $f(x)$, the marginal pdf of X.

1.
   - The naïve MC estimate of $E(X)$ is $\hat{\mu}_{MC} = \frac{1}{m}\sum_{j=1}^{m} x_j = 5.61$, and $\widehat{\text{var}}(\hat{\mu}_{MC}) = 0.0139$.
   - As $E(X|p, N) = Np$, the Rao-Blackwellised MC estimate of $E(X)$ is

$$\hat{\mu}_{RB} = \frac{1}{m}\sum_{i=1}^{m} E(X|p_j, N_j) = \frac{1}{m}\sum_{j=1}^{m} N_j p_j = 5.544,$$

   and $\widehat{\text{var}}(\hat{\mu}_{RB}) = 0.0106$.

Example 9. (2)

2. Estimate $f(x)$, the marginal pdf of X.

- The naïve MC estimator is $\hat{f}_{MC}(x) = \frac{1}{m}\sum_{j=1}^{m} I(x_j = x)$.

```
> table(gibbsam2$X)/m
```

0	1	2	3	4	5	6	7	8	9
0.049	0.087	0.088	0.116	0.094	0.095	0.103	0.102	0.055	0.058
10	11	12	13	14	15	16	17	18	19
0.046	0.022	0.026	0.020	0.022	0.009	0.004	0.001	0.001	0.002

- As $(X|p, N) \overset{d}{=} \text{Bin}(N, p)$, the RB MC estimator of $f(X)$ is

$$\hat{f}_{RB}(x) = \frac{1}{m}\sum_{i=1}^{m} f(x|p_j, N_j) = \frac{1}{m}\sum_{j=1}^{m} \binom{N_j}{x} p_j^x (1-p_j)^{N_j - x}, \quad x = 0, 1, \cdots \max N_j$$

```
> round(fx,digit=3)
```

0	1	2	3	4	5	6	7	8	9	10
0.047	0.082	0.102	0.110	0.109	0.103	0.093	0.081	0.068	0.055	0.043
11	12	13	14	15	16	17	18	19	20	21
0.033	0.025	0.018	0.013	0.008	0.005	0.003	0.002	0.001	0.000	0.000

Example 9. (3)

```
> mean(gibbsam2$X); var(gibbsam2$X)/m
[1] 5.61;          [1] 0.01390981
> mean(gibbsam2$N*gibbsam2$p); var(gibbsam2$N*gibbsam2$p)/m
[1] 5.544295;    [1] 0.01061962
fx.RB=function(x){px=rep(-1,length(x))
for(i in 1:length(x))px[i]=mean(dbinom(x[i],size=gibbsam2$N,prob=gibbsam2$p))
return(px)}
> fx=fx.RB(0:max(gibbsam2$N)); names(fx)=0:32; round(fx,dig=3)
    0     1     2     3     4     5     6     7     8     9    10
0.047 0.082 0.102 0.110 0.109 0.103 0.093 0.081 0.068 0.055 0.043
   11    12    13    14    15    16    17    18    19    20    21
0.033 0.025 0.018 0.013 0.008 0.005 0.003 0.002 0.001 0.000 0.000
   22    23    24    25    26    27    28    29    30    31    32
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
> table(gibbsam2$X)/m
    0     1     2     3     4     5     6     7     8     9    10
0.049 0.087 0.088 0.116 0.094 0.095 0.103 0.102 0.055 0.058 0.046
   11    12    13    14    15    16    17    18    19
0.022 0.026 0.020 0.022 0.009 0.004 0.001 0.001 0.002
```

Example 9. (4)

```
> plot(0:32,fx,type="n",ylim=c(0,0.12),xlab="X", ylab="probability",
      lwd=2,main="simulated pdf of X")
> points(0:max(gibbsam2$N), fx, pch=19)
> lines(0:max(gibbsam2$N), fx)
> points(sort(unique(gibbsam2$X)), table(gibbsam2$X)/m, type="h", lwd=1)
> points(sort(unique(gibbsam2$X)), as.numeric(table(gibbsam2$X)/m), pch=17)
> legend("topright", legend = c("naive MC estimator",
     "Rao-Blackwellised MC estimator"), pch=c(17,19),bty = "o")
```

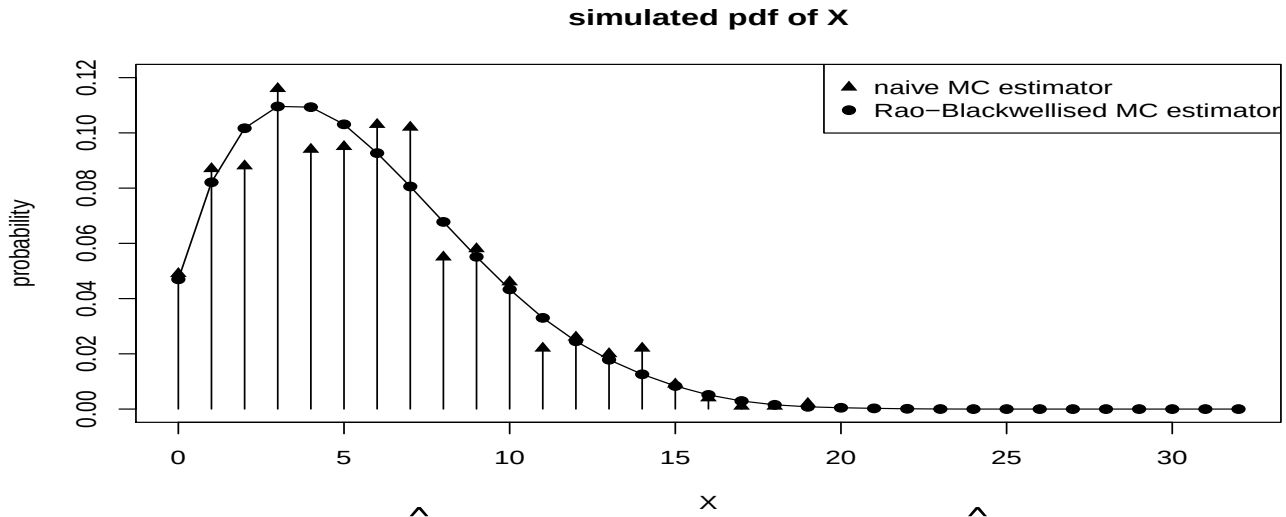


Figure: $\hat{f}_{RB}(x)$ is smoother than $\hat{f}_{MC}(x)$.

Explaining Gibbs sampler (1)

Theorem (Fundamental Theorem of Simulation (1D case))

1. *Simulating X from a pdf/pmf $f(x)$ is equivalent to simulating (X, U) from a 2D uniform distribution on the 2D set*

$$A = \{(x, u) : 0 \leq u \leq f(x)\}.$$

2. *If $f(x) = cf_1(x)$ with $c > 0$ a constant, then simulating X from a pdf/pmf $f(x)$ is equivalent to simulating (X, U') from a 2D uniform distribution on the 2D set*

$$A_1 = \{(x, u') : 0 \leq u' \leq f_1(x)\}.$$

Explaining Gibbs sampler (2)

Proof: Let $g(x,u)$ be the uniform pdf on A. Then $g(x,u)$ must be a constant κ on A and $\int\int_A g(x,u)dudx = 1$. That is

$$1 = \int\int_A g(x,u)dudx = \int\int_A \kappa \cdot dudx = \kappa \cdot \text{area}(A) = \kappa \int_{-\infty}^{\infty} f(x)dx = \kappa. \tag{3}$$

Thus $g(x,u) = 1$ over A and $= 0$ outside A.

When $(X,U) \stackrel{d}{=} \text{Uniform}(A)$, it follows that the marginal pdf of X is

$$\int_{-\infty}^{\infty} g(x,u)du = \int_0^{f(x)} 1 \cdot du = f(x).$$

This proves part 1 of the theorem.

Explaining Gibbs sampler (3)

Proof: (*continued*) Let $g_1(x, u')$ be the uniform pdf on A_1. Similar to using (3) it can be verified that $g_1(x, u') = [\text{area}(A_1)]^{-1} = [\int_{-\infty}^{\infty} f_1(x)dx]^{-1} = [c^{-1} \int_{-\infty}^{\infty} f(x)dx]^{-1} = c$ over A_1 and $= 0$ outside A_1.
When $(X, U') \stackrel{d}{=} \text{Uniform}(A_1)$, it follows that the marginal pdf of X is

$$\int_{-\infty}^{\infty} g_1(x, u')du' = \int_0^{f_1(x)} c \cdot du' = cf_1(x) = f(x).$$

This proves part 2 of the theorem.

Explaining Gibbs sampler (4)

Theorem (Fundamental Theorem of Simulation (2D case))

Simulating (X, Y) from a pdf/pmf $f(x, y)$ is equivalent to simulating (X, Y, U) from a 3D uniform distribution on the 3D set

$$A = \{(x, y, u) : 0 \leq u \leq f(x, y)\}.$$

Explaining Gibbs sampler (5)

We can apply a so-called random walk principle to implement the above theorem. That is, generate a random sample of (X, Y, U) from Uniform(A) by generating one component at each time, conditional on the latest values of the other components. Namely, starting at a point (x, y, u) in A, generate

(i) X along the x-axis from the uniform distribution on $\{x\colon\ u \le f(x, y)\}$, obtaining an x';

(ii) Y along the y-axis from the uniform distribution on $\{y\colon\ u \le f(x', y)\}$, obtaining a y';

(iii) U along the u-axis from the uniform distribution on $\{u\colon\ 0 \le u \le f(x', y')\}$, obtaining a u'. Repeat (i) to (iii).

By the random walk principle the generated samples of (X, Y, U) form a Markov chain with its stationary distribution being Uniform(A).

Explaining Gibbs sampler (6)

- By case (2) of the Fundamental Theorem of Simulation (1D case), (i) $\Leftrightarrow$ simulating X from the uniform distribution on $\{x\colon\ u \leq f_{X|Y}(x|y)\}$; and (ii) $\Leftrightarrow$ simulating Y from the uniform distribution on $\{y\colon\ u \leq f_{Y|X}(y|x')\}$.
- Also by the random walk principle, simulating the sequence of uniform generations along the 3 axes need not follow the same order $x \to y \to u \to x$ all the time for the Markov chain to remain having Uniform(A) as its stationary distribution.
- This implies, theoretically we could repeat (i) and (iii) ∞ times then repeating (ii) and (iii) ∞ times.
- Repeat (i) and (iii) ∞times $\Leftrightarrow$ generating X from $f_{X|Y}(x|y)$.
- Repeat (ii) and (iii) ∞ times $\Leftrightarrow$ generating Y from $f_{Y|X}(y|x')$.

Explaining Gibbs sampler (7)

- Hence, repeating (i) to (iii) in the way as aforementioned is equivalent to the following 2-stage Gibbs sampler:
  Take $X_0 = x_0$. Then for $t = 1, 2, \cdots$, generate
  1° $Y_t \sim f_{Y|X}(\cdot|x_{t-1})$.
  2° $X_t \sim f_{X|Y}(\cdot|y_t)$.
- Therefore, the Markov chain generated by the 2-stage Gibbs sampler has the stationary distribution $f(x, y)$ which is the target pdf/pmf. Accordingly, the Markov chain generated (after a burn-in period) can be deemed as a random sample from the target pdf/pmf.
- Similarly, it can be shown that a p-dimensional target pdf/pmf $f(\mathbf{x})$ can be simulated using a p-stage Gibbs sampler which produces a Markov chain with $f(\mathbf{x})$ as the stationary distribution.

Markov chain Monte Carlo (1)

- Let $\{\mathbf{X}^{(t)}, t = 0, 1, 2, \cdots\}$ be a time-homogeneous Markov chain where $\mathbf{X}^{(t)}$ can be multivariate, has state space $\mathcal{S}$ and σ-field $\mathcal{F}$ spanned by $\mathcal{S}$.
- The one-step transition probability function is defined as $P(\mathbf{x}, A) = P(\mathbf{X}^{(t+1)} \in A | \mathbf{X}^{(t)} = \mathbf{x})$ for any $A \in \mathcal{F}$, which is the probability that the Markov chain moves to a state in A at the next step from the current state $\mathbf{x}$. Note that
  - $P(\mathbf{x}, \mathcal{S}) = 1$ for any $\mathbf{x} \in \mathcal{S}$;
  - but $P(\mathbf{x}, \{\mathbf{x}\})$ is not necessarily equal to 0.
  - $P(\mathbf{x}, \{\mathbf{x}\})$ represents the probability that the chain stays at $\mathbf{x}$ after one step of move.
- If there exists a function $p(\mathbf{x}, \mathbf{y})$ such that $P(\mathbf{x}, A) = \int_A p(\mathbf{x}, \mathbf{y}) d\mathbf{y}$ for any $A \in \mathcal{F}$, we call $p(\mathbf{x}, \mathbf{y})$ the *transition density* of $\{\mathbf{X}^{(t)}, t = 0, 1, \cdots\}$.

Markov chain Monte Carlo (2)

- Similarly the m-step transition probability function is

$$P^{(m)}(\mathbf{x}, A) = P(\mathbf{X}^{(t+m)} \in A | \mathbf{X}^{(t)} = \mathbf{x}) = \int_{\mathcal{S}} P^{(m-1)}(\mathbf{x}, d\mathbf{y}) P(\mathbf{y}, A), \quad A \in \mathcal{F},$$

  which is the probability that the Markov chain moves to a state in A after m steps from the current state $\mathbf{x}$. Note that
  - $P^{(1)}(\mathbf{x}, A) = P((\mathbf{x}, A)$;
  - $d\mathbf{y}$ is understood as a small neighbourhood of $\mathbf{y}$.
- In classic Markov chain study the transition probability $P(\mathbf{x}, A)$ is often provided and the major interest is to determine whether the chain converges to a stationary distribution, and if so, find the stationary distribution.

Markov chain Monte Carlo (3)

- In Markov chain Monte Carlo (MCMC), we require the existence of stationary distribution, which is set to be the target pdf $\pi(\mathbf{x})$. The target pdf often has the form $\pi(\mathbf{x}) = \frac{f(\mathbf{x})}{\kappa}$ where the standardising constant κ may be intractable.
- A major objective in MCMC is to find a transition probability function $P(\mathbf{x}, A)$ so that the Markov chain generated by this $P(\mathbf{x}, A)$ converges to the stationary distribution $\pi^*(d\mathbf{x}) = \pi(\mathbf{x})d\mathbf{x}$.
- Once a simple such $P(\mathbf{x}, A)$ is found, we can use it to generate new moves from the current moves. The resultant sequence will be a Markov chain with $\pi(\mathbf{x})$ being the stationary distribution, so can be used to estimate various attributes of $\pi(\mathbf{x})$ by the ergodic theorem.

Markov chain Monte Carlo (4)

- To find a useful transition probability function $P(\mathbf{x}, A)$, we restrict it to have the following form:

$$P(\mathbf{x}, d\mathbf{y}) = p(\mathbf{x}, \mathbf{y})d\mathbf{y} + r(\mathbf{x})\delta_{\mathbf{x}}(d\mathbf{y}) \tag{4}$$

where
$\delta_{\mathbf{x}}(d\mathbf{y}) = \begin{cases} 1 & \text{if } d\mathbf{y} \supset \{\mathbf{x}\}, \\ 0 & \text{otherwise;} \end{cases}$ and $r(\mathbf{x}) = 1 - \int_{\mathcal{S}} p(\mathbf{x}, \mathbf{y})d\mathbf{y}$.

Note $\int_{\mathcal{S}} p(\mathbf{x}, \mathbf{y})d\mathbf{y} \leq 1$; and $P(\mathbf{x}, d\mathbf{y})$ says that the process $\mathbf{X}^{(t)}$ either moves to $\mathbf{y}$ from $\mathbf{x}$ according to the non-standardised transition pdf $p(\mathbf{x}, \mathbf{y})$, or stays at $\mathbf{x}$ with probability $r(\mathbf{x})$.

Markov chain Monte Carlo (5)

- It is found that a pdf $\pi(\mathbf{y})$ would be the stationary pdf associated with the transition probability function $P(\mathbf{x}, d\mathbf{y})$ if the following **detailed balance** condition is satisfied:

$$\pi(\mathbf{x})p(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})p(\mathbf{y},\mathbf{x}) \quad \text{for all } \mathbf{x},\mathbf{y} \in \mathcal{S}. \tag{5}$$

  The Markov chain is called **reversible** if (5) holds.

Proof. For $P(\mathbf{x}, d\mathbf{y})$ given by (4),

$$\int_{\mathcal{S}} P(\mathbf{x}, d\mathbf{y})\pi(\mathbf{x})d\mathbf{x} = \int_{\mathcal{S}} [p(\mathbf{x},\mathbf{y})d\mathbf{y} + r(\mathbf{x})\delta_{\mathbf{x}}(d\mathbf{y})]\pi(\mathbf{x})d\mathbf{x}$$

$$= \int_{\mathcal{S}} \pi(\mathbf{x})p(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} + \int_{\mathcal{S}} r(\mathbf{x})\delta_{\mathbf{x}}(d\mathbf{y})\pi(\mathbf{x})d\mathbf{x}$$

$$= \int_{\mathcal{S}} \pi(\mathbf{y})p(\mathbf{y},\mathbf{x})d\mathbf{x}d\mathbf{y} + r(\mathbf{y})\pi(\mathbf{y})d\mathbf{y} = \pi(\mathbf{y})d\mathbf{y}[\int_{\mathcal{S}} p(\mathbf{y},\mathbf{x})d\mathbf{x} + r(\mathbf{y})] = \pi(\mathbf{y})d\mathbf{y}.$$

Hence $\pi(\mathbf{x})$ is indeed the stationary pdf for the transition $P(\mathbf{x}, d\mathbf{y})$.

Metropolis-Hastings algorithm (1)

Now we are ready to derive the Metropolis-Hastings (MH) algorithm.

- The target pdf is $\pi(\mathbf{x}) = \frac{f(\mathbf{x})}{\kappa}$.
- Suppose a *proposal transition pdf* $q(\mathbf{x}, \mathbf{y})$ is given such that $\int_{\mathcal{S}} q(\mathbf{x}, \mathbf{y}) d\mathbf{y} = 1$.
- If $\pi(\mathbf{x})q(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y})q(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{S}$, the detailed balance condition is satisfied, and $q(\mathbf{x}, \mathbf{y})$ is the desired transition pdf by which a Markov chain can be generated to achieve the stationary pdf $\pi(\mathbf{x})$.

Metropolis-Hastings algorithm (2)

- If $\pi(\mathbf{x})q(\mathbf{x},\mathbf{y}) \neq \pi(\mathbf{y})q(\mathbf{y},\mathbf{x})$, we define

$$p_{MH}(\mathbf{x},\mathbf{y}) = q(\mathbf{x},\mathbf{y})\alpha(\mathbf{x},\mathbf{y})$$

  where $\alpha(\mathbf{x},\mathbf{y}) \leq 1$ is a probability of a move from $\mathbf{x}$ to $\mathbf{y}$.
- We want to find a proper $\alpha(\mathbf{x},\mathbf{y})$ so that $\pi(\mathbf{x})p_{MH}(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})p_{MH}(\mathbf{y},\mathbf{x})$ for all $\mathbf{x},\mathbf{y} \in \mathcal{S}$.
- Namely, we want $\pi(\mathbf{x})q(\mathbf{x},\mathbf{y})\alpha(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})q(\mathbf{y},\mathbf{x})\alpha(\mathbf{y},\mathbf{x})$, or equivalently

$$\frac{\alpha(\mathbf{x},\mathbf{y})}{\alpha(\mathbf{y},\mathbf{x})} = \frac{\pi(\mathbf{y})q(\mathbf{y},\mathbf{x})}{\pi(\mathbf{x})q(\mathbf{x},\mathbf{y})} = \frac{f(\mathbf{y})q(\mathbf{y},\mathbf{x})}{f(\mathbf{x})q(\mathbf{x},\mathbf{y})}. \tag{6}$$

- The solution is $\alpha(\mathbf{x},\mathbf{y}) = \min\left\{\frac{f(\mathbf{y})q(\mathbf{y},\mathbf{x})}{f(\mathbf{x})q(\mathbf{x},\mathbf{y})}, 1\right\}$.

Metropolis-Hastings algorithm (3)

We call $\alpha(\mathbf{x}, \mathbf{y})$ the **acceptance probability**, having the following interpretations:

- If $\pi(\mathbf{x})q(\mathbf{x}, \mathbf{y}) < \pi(\mathbf{y})q(\mathbf{y}, \mathbf{x})$, meaning the process moves from $\mathbf{y}$ to $\mathbf{x}$ more often than from $\mathbf{x}$ to $\mathbf{y}$, we need to reduce the probability of moving from $\mathbf{y}$ to $\mathbf{x}$ to satisfy the detailed balance condition. This is done by multiplying $\alpha(\mathbf{y}, \mathbf{x})$ with $q(\mathbf{y}, \mathbf{x})$. Note $\alpha(\mathbf{y}, \mathbf{x}) < 1$ and $\alpha(\mathbf{x}, \mathbf{y}) = 1$ now.
- If $\pi(\mathbf{x})q(\mathbf{x}, \mathbf{y}) > \pi(\mathbf{y})q(\mathbf{y}, \mathbf{x})$, we need to reduce the probability of moving from $\mathbf{x}$ to $\mathbf{y}$ by multiplying $\alpha(\mathbf{x}, \mathbf{y})$ with $q(\mathbf{x}, \mathbf{y})$. Note $\alpha(\mathbf{x}, \mathbf{y}) < 1$ and $\alpha(\mathbf{y}, \mathbf{x}) = 1$ now.
- The desired transition probability function producing the stationary pdf $\pi(\mathbf{x})$ is
$P_{MH}(\mathbf{x}, d\mathbf{y}) = q(\mathbf{x}, \mathbf{y})\alpha(\mathbf{x}, \mathbf{y})d\mathbf{y} + [1 - \int_{\mathcal{S}} q(\mathbf{x}, \mathbf{y})\alpha(\mathbf{x}, \mathbf{y})d\mathbf{y}]\delta_{\mathbf{x}}(d\mathbf{y})$

Metropolis-Hastings algorithm (4)

The transition probability function $P_{MH}(\mathbf{x}, d\mathbf{y})$ implies the following algorithm for generating an approximate sample (a Markov chain) from the target pdf $\pi(\mathbf{x})$:

Algorithm 15 [*Metropolis-Hastings algorithm*]

- Start from an initial (arbitrarily) sample point $\mathbf{x}^{(0)}$.
- Repeat for $j = 0, 1, 2, \cdots, N$.
- Generate $\mathbf{y}$ from $q(\mathbf{x}^{(j)}, \mathbf{y})$ and u from Uniform(0,1).
- If $u \leq \alpha(\mathbf{x}^{(j)}, \mathbf{y})$, set $\mathbf{x}^{(j+1)} = \mathbf{y}$.
  Else, set $\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)}$.
- Return the samples $\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\}$.

The pdf of $\mathbf{X}^{(j)}$ converges to the target pdf $\pi(\mathbf{x})$ as $j \to \infty$.

Remarks on Metropolis-Hastings algorithm

1. *How to choose a proposal transition pdf* $q(\mathbf{x}, \mathbf{y})$*?* In general, $q(\mathbf{x}, \mathbf{y})$ can be any pdf in terms of $\mathbf{y}$ as long as its $\mathbf{y}$ support covers that of the target pdf $\pi(\mathbf{x})$. For example, one could choose Exp(1) pdf as $q(\mathbf{x}, \mathbf{y})$ if $\pi(\mathbf{x})$ is the pdf of a positive continuous r.v.; choose some discrete pdf as $q(\mathbf{x}, \mathbf{y})$ if $\pi(\mathbf{x})$ is discrete. However, the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$ can be very small most of the time if $q(\mathbf{x}, \mathbf{y})$ is not properly chosen. Then the resultant Markov chain can be very slow in becoming stationary. Choosing $q(\mathbf{x}, \mathbf{y})$ depends on the problem under consideration.
2. Note $\alpha(\mathbf{x}, \mathbf{y})$ does not involve κ in $\pi(\mathbf{x}) = \frac{f(\mathbf{x})}{\kappa}$.
3. It is called the *Metropolis algorithm* if $q(\mathbf{x}, \mathbf{y})$ is symmetric, i.e. $q(\mathbf{x}, \mathbf{y}) = q(\mathbf{y}, \mathbf{x})$. Then $\alpha(\mathbf{x}, \mathbf{y}) = \min\{\frac{f(\mathbf{y})}{f(\mathbf{x})}, 1\}$.

Monitoring MCMC convergence (1)

- The Markov chain generated by an MCMC algorithm such as the MH algorithm or Gibbs sampler may take some time to become stationary. Therefore a **burn-in period** is required for the Markov chain to be used as an approximate sample for the target pdf $\pi(\mathbf{x})$.
- Normally one generates a long sequence $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}, \mathbf{x}^{(m+1)}, \cdots, \mathbf{x}^{(m+n)}\}$, then drop the first m observations and use only the last n as the approximate samples.
- To reduce the dependence among the samples to be used, sub-sampling may also be applied. Namely, systematically or randomly take a sub-sequence from the Markov chain generated.

Monitoring MCMC convergence (2)

- The key question here is how large m should be and how to assess the convergence, or stationarity, or equilibrium of the Markov chain generated.
- A popular procedure to answer this question is to generate multiple, say k Markov chains, each of size n, using different initial values. The multiple chains will behave similarly when they become stationary. This can be tested by e.g. **Gelman and Rubin statistic** (Gelman & Rubin (1992). *Statistical Science*, **7**, 452-472.), which focus on univariate chains.
- The GR statistic is related to the ratio of between-chain (across k chains) variation to within-chain (up to iteration n) variation. If this ratio becomes close to 1 (e.g. 1.01, 1.05 or 1.1), one can accept that the chains have become stationary.

Monitoring MCMC convergence (3)

The GR statistic is formally defined as

$$\hat{R} = \left(\frac{n-1}{n} + \frac{(k+1)}{kn} \frac{B}{W} \right) \frac{\text{df}}{\text{df}-2} = \frac{\hat{V}}{W} \frac{\text{df}}{\text{df}-2},$$

where $\frac{B}{n} = \frac{1}{k-1} \sum_{i=1}^{k} (\bar{x}_{i\bullet} - \bar{x}_{\bullet\bullet})^2$; $W = \frac{1}{k} \sum_{i=1}^{k} s_i^2$ with $s_i^2 = \frac{1}{n-1} \sum_{j=1}^{n} (x_{ij} - \bar{x}_{i\bullet})^2$ being the sample variance of chain i; and $\text{df} = \frac{2\hat{V}^2}{\widehat{\text{var}}(\hat{V})}$ with

$$\hat{V} = \frac{n-1}{n} W + \frac{k+1}{kn} B \quad \text{and}$$

$$\widehat{\text{var}}(\hat{V}) = \left(\frac{n-1}{n} \right)^2 \frac{1}{k} \widehat{\text{var}}(s_i^2) + \left(\frac{k+1}{kn} \right)^2 \frac{2}{k-1} B^2$$
$$+2 \frac{(k-1)(n-1)}{k^2 n} \left[\widehat{cov}(s_i^2, \bar{x}_{i\bullet}^2) - 2\bar{x}_{\bullet\bullet} \widehat{cov}(s_i^2, \bar{x}_{i\bullet}) \right].$$

Monitoring MCMC convergence (4)

- Many research papers were published at the end of 1990s concerning developing various diagnostic criteria for MCMC convergence (c.f. Robert & Casella (2004) *Monte Carlo Statistical Methods*, 2nd Ed., Chapter 13), concluding sadly that no criterion is absolute. Randomness involved in the problem prevents any categorical guarantee of performance.
- Many such diagnostic tools have been implemented in the `R` package `coda` by Plummer et al. (2006, CODA: convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7-11).
- Finally note that there are two separate notions of MCMC convergence: one is the convergence to stationary distribution, the other is convergence of ergodic average (i.e. Monte Carlo estimator based on Markov chains).

Example 10. Simulate Beta(2, 8) by MH algorithm (1)

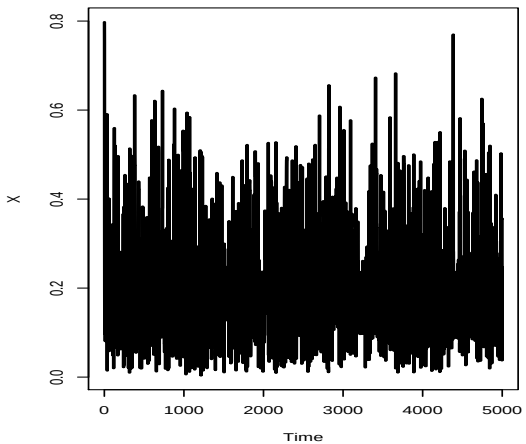
Example 10 Recall Example 1, where an AR algorithm is used to simulate a Beta(2, 8) distribution. An MH algorithm is now used to simulate Beta(2,8) where we use Unif(0,1) as the proposal density (pdf). We will not do formal MCMC convergence diagnosis here. The following code will generate a Markov chain of size 5000:

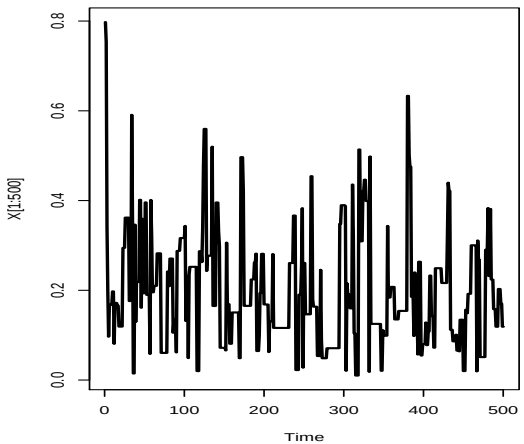
```
set.seed(123456)
a=2.0; b=8.0    #initial values
Nsim=5000;  X=rep(runif(1),Nsim)    #initialize the chain
for (i in 2:Nsim){
   Y=runif(1)      #generate a number from the proposal pdf
   alpha=dbeta(Y,a,b)/dbeta(X[i-1],a,b)      #acceptance probability
   X[i]=X[i-1] + (Y-X[i-1])*(runif(1)<alpha)  #accept Y with probability alpha.
}
> mean(X); var(X); 2/(2+8); 16/(100*11)
[1] 0.1949034; [1] 0.01472183; [1] 0.2; [1] 0.01454545
```

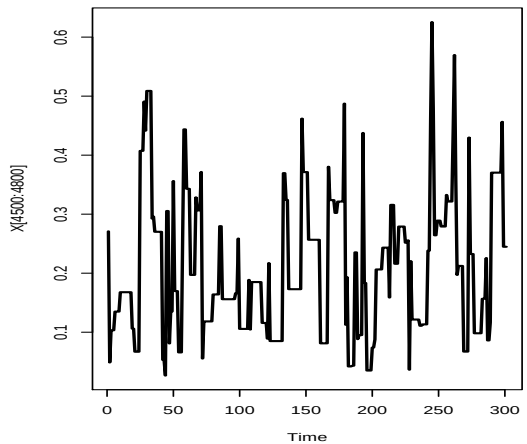
Example 10. (2)

Time series plots of the whole chain and the parts where $t \leq 500$, and $4500 \leq t \leq 4800$ are displayed below. It seems the chain becomes stationary after possibly just 100 generations. Also the plots contain many time intervals where the chain does not change because all corresponding proposed generations are rejected.

```
par(mfrow=c(1,3))
ts.plot(X, type="l",lwd=2)
ts.plot(X[1:500], type="l",lwd=2)
ts.plot(X[4500:4800], type="l",lwd=2)
```



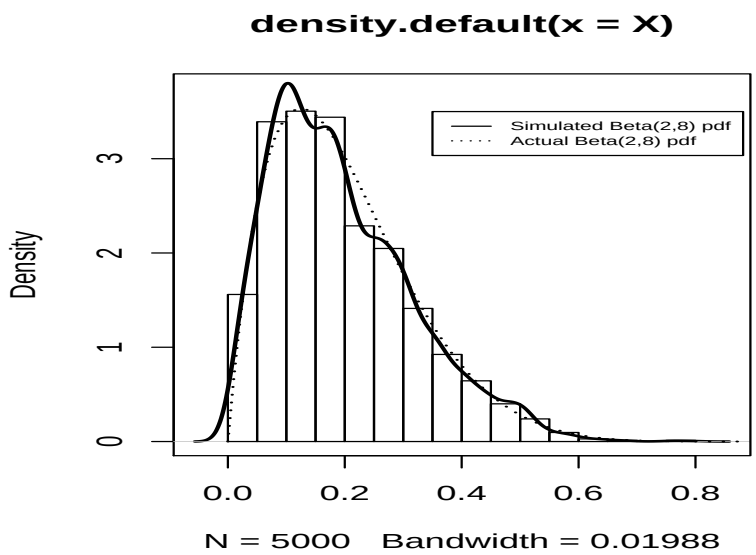


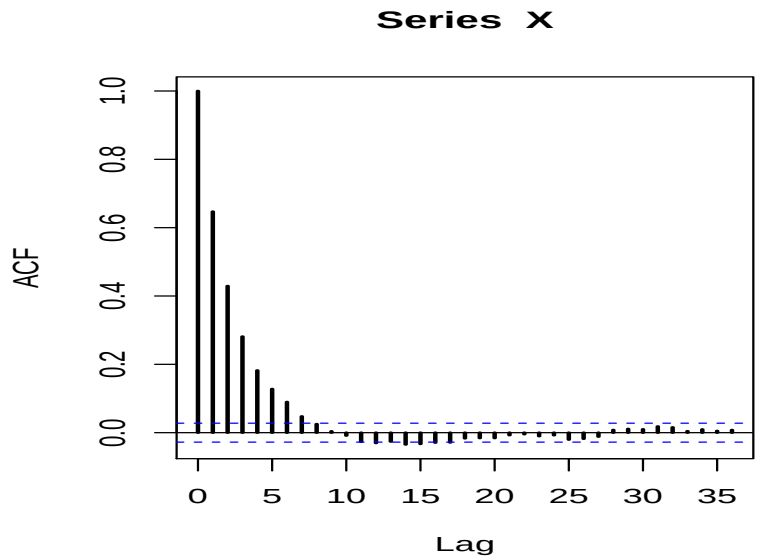


Example 10. (3)

The plots show the generated samples aren't independent, and provide a good approximation to Beta(2,8), but not as good as that provided by AR sampling.

```
par(mfrow=c(1,2));  plot(density(X),ylim=c(0.0,3.75),lwd=2)
hist(X, freq=F, add=T)
curve(dbeta(x,2,8),from=0,to=1, add=T,lwd=1.5, lty=3)
legend(x=0.35, y=3.5, legend=c("Simulated Beta(2,8) pdf",
          "Actual Beta(2,8) pdf"), lty=c(1,3),cex=0.6)
acf(X, lwd=2)
```





Example 11. Simulate Cauchy(1, 0) by MH algorithm (1)

Example 11 (Robert and Casella, 2010). Cauchy(1,0), or $t(1)$, has the pdf $g(x) = \frac{1}{\pi(1+x^2)}$, $-\infty < x < \infty$, which does not have mean and variance. The R function `rcauchy()` or `rt(n, df=1)` can be used to generate Cauchy(0,1) random numbers. Here we use MH algorithm to do the generation. The objective is to illustrate the importance of choosing a proper proposal density.

- Naturally $N(0, 1)$ can be used as a proposal. With this proposal, the following code generates 10,000 numbers:

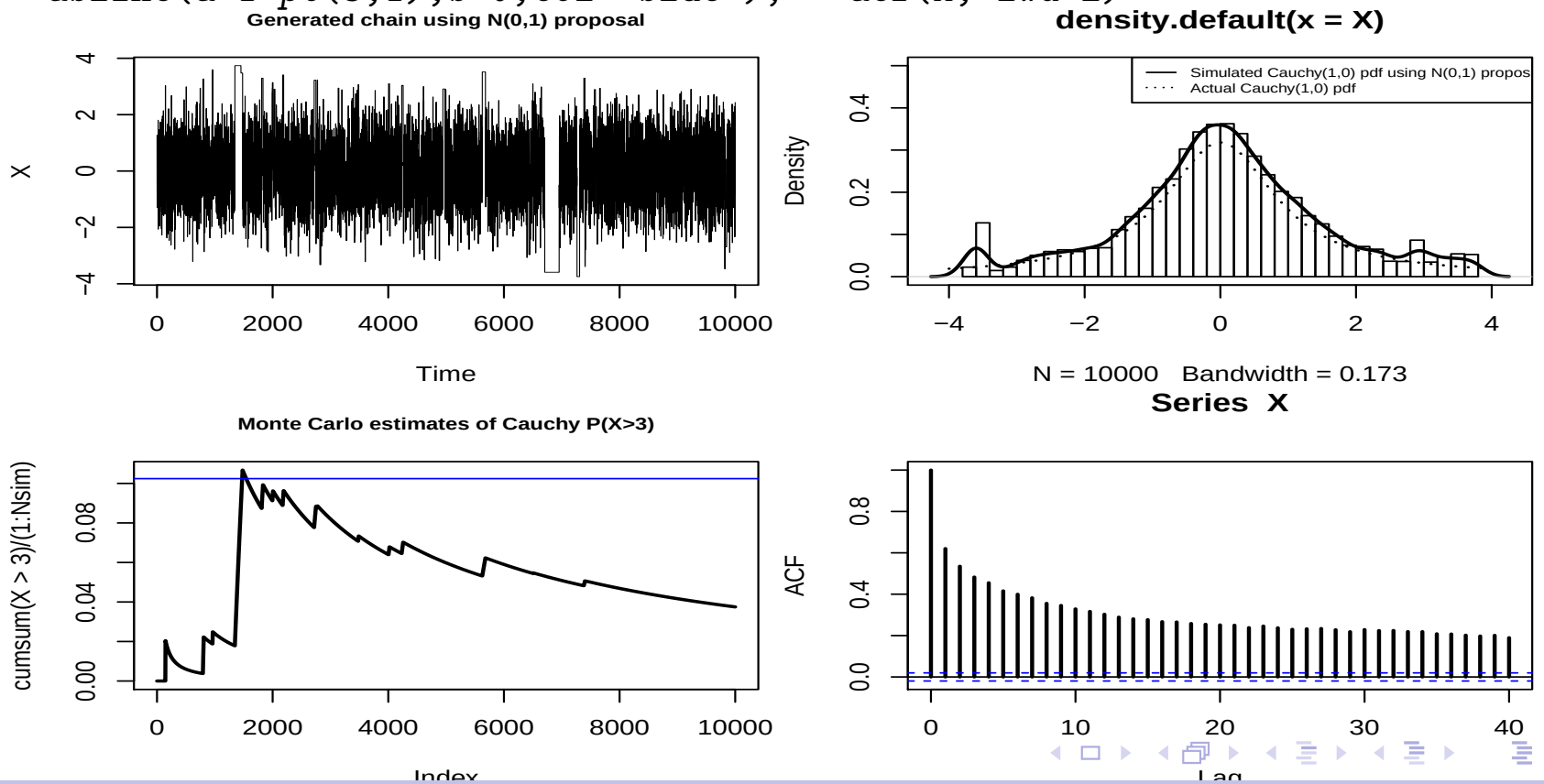
```
set.seed(123456)
Nsim=10^4
X=c(rt(1,1)) # initialize the chain
for (t in 2:Nsim){
Y=rnorm(1) # normal proposal
alpha=dt(Y,1)*dnorm(X[t-1])/(dt(X[t-1],1)*dnorm(Y))
X[t]=X[t-1] + (Y-X[t-1])*(runif(1)<alpha)
}
```

Example 11. Simulate Cauchy(1, 0) by MH algorithm (2)

- The acceptance probability for the $N(0,1)$ proposal is $\alpha(x^{(t-1)}, y) = \min\{\frac{g(y)\phi(x^{(t-1)})}{g(x^{(t-1)})\phi(y)}, 1\}$; $\phi(\cdot)$ is the $N(0,1)$ pdf.
- If a generated $x^{(t-1)}$ is pretty small or large at certain $t-1$, meaning $\phi(x^{(t-1)})$ is very very small (eg. $\phi(\pm 5) = 1.49 \times 10^{-6}$), then $\alpha(x^{(t-1)}, y)$ will be very close to 0, meaning the chain will get stuck at $x^{(t-1)}$ for very long following time $t-1$.
- If the chain starts from a more central value, the generated chain will resemble a normal sample much more than a Cauchy sample as the generated chain is more likely to stay centrally than move outward.
- Monte Carlo estimates of $P(X > 3)$ are poor based on the generated chain. Actual value of $P(X > 3) = 0.1024$.

Example 11. Simulate Cauchy(1, 0) by MH algorithm (3)

```
> par(mfrow=c(2,2),mar=c(4,4,3,0.6))
> ts.plot(X, type="l"); plot(density(X)); hist(X, freq=F, add=T)
> curve(dt(x,1),from=-4,to=4, add=T,lwd=1.5, lty=3);
> plot(cumsum(X>3)/(1:Nsim),lwd=2,ty="l")
> abline(a=1-pt(3,1),b=0,col="blue");   acf(X, lwd=2)
```



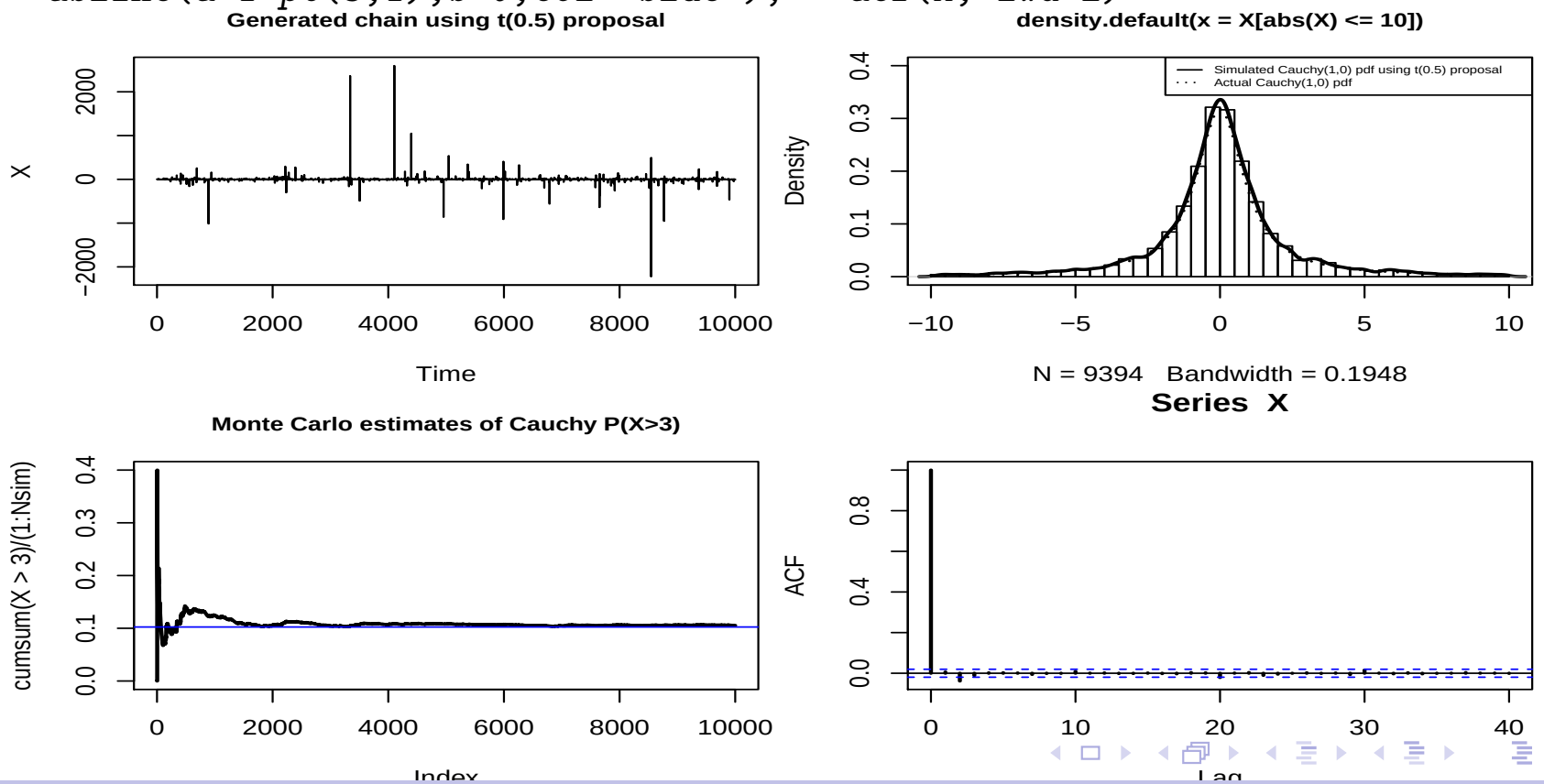
Example 11. Simulate Cauchy(1, 0) by MH algorithm (4)

Using $t(\text{df} = 0.5)$ as the proposal density will give much better results on approximating Cauchy(1,0) pdf and $P(X > 3)$.

- We only need to replace `Y=rnorm(1)` with `Y=rt(1, df=0.5)` in the simulation code.
- The acceptance probability for the $t(\text{df} = 0.5)$ proposal is $\alpha(x^{(t-1)}, y) = \min\{\frac{g(y)t_{0.5}(x^{(t-1)})}{g(x^{(t-1)})t_{0.5}(y)}, 1\}$.
- The $t(\text{df} = 0.5)$ pdf is more spread out than Cauchy(1,0). The acceptance probability is less likely to be very small now.
- Very large or small values may be generated occasionally, but the chain will not stay there for long time.
- The generated chain has much less autocorrelation than that using the $N(0, 1)$ proposal.

Example 11. Simulate Cauchy(1, 0) by MH algorithm (5)

```
> par(mfrow=c(2,2),mar=c(4,4,3,0.6)); ts.plot(X, type="l")
> plot(density(X[abs(X)<=10])); hist(X[abs(X)<=10], freq=F, add=T)
> curve(dt(x,1),from=-4,to=4, add=T,lwd=1.5, lty=3);
> plot(cumsum(X>3)/(1:Nsim),lwd=2,ty="l")
> abline(a=1-pt(3,1),b=0,col="blue");   acf(X, lwd=2)
```



Questions?