STAT3003: 统计算法基础

蒙特卡洛方法（4）

曾靖

中国科学技术大学管理学院

大纲

大纲

1. 马尔可夫链和 MCMC
2. Metropolis-Hasting 算法
   - 2.1 Metropolis-Hasting
   - 2.2 Metropolis
   - 2.3 随机游走 Metropolis
   - 2.4 独立性样本生成器
3. Gibbs 算法
   - 3.1 系统扫描 Gibbs 抽样
   - 3.2 随即扫描 Gibbs 抽样

本节内容来自于《统计计算使用 R》第 9 章和《统计计算》第 3.7 节

1 马尔可夫链和 MCMC

1.1 马尔可夫链

- 状态: 对于一个随机过程 $\{X_t, t = 0, 1, \cdots\}$, 称 X_t 为系统在时刻 t 的状态
- 状态空间: X_t 的所有可能取值构成状态空间 $\mathcal{S}$。假设状态空间有限并且 $\mathcal{S} = \{1, 2, \cdots, m\}$。
- 马尔可夫链/马氏链 *(Markov Chain)*: 称一个随机过程 $\{X_t, t = 0, 1, \cdots\}$ 为马尔可夫链如果对于 $\forall t = 0, 1, \ldots$ 以及 $\forall i, j, k_0, \ldots, k_{t-1} \in \mathcal{S}$, 我们有

$$\begin{aligned} &\Pr(X_{t+1} = j \mid X_0 = k_0, \cdots, X_{t-1} = k_{t-1}, X_t = i) \\ =&\Pr(X_{t+1} = j \mid X_t = i). \end{aligned}$$

- **转移概率**: 记 $p_{ij} = \Pr(X_{t+1} = j \mid X_t = i)$ 为状态 i 到 j 的转移概率。
- **k 步转移概率**: $p_{ij}^{(k)} = \Pr(X_{t+k} = j \mid X_t = i)$; $p_{ij}^{(k)}$ 不依赖于时间 t。
- **转移概率矩阵**: 构造矩阵 $\mathbf{P} = (p_{ij})_{m\times m}$, 这里 $p_{ij} = \Pr(X_{t+1} = j \mid X_t = i)$。
- **不可约马氏链**: 对任意 $i, j \in \mathcal{S}, i \neq j$, 存在 $k \geq 1$ 使得 $p_{ij}^{(k)} > 0$。即不可约马氏链的所有状态经过若干步后总能互相转移。
- **非周期马氏链**: 状态 i 的周期为

$$d(i) = \gcd\{k \geq 1 : p_{ii}^{(k)} > 0\}.$$

若 $d(i) = 1$，则状态 i 非周期；若所有状态均非周期，则称该马氏链为非周期的。

- 平稳分布：若概率分布 π 满足

$$\boxed{\sum_{i\in\mathcal{S}}\pi_i p_{ij}=\pi_j,\quad j\in\mathcal{S},\qquad \sum_{i\in\mathcal{S}}\pi_i=1,}$$

  则称 π 为平稳分布。即若 $X_0\sim\pi$，则各时刻均有 $X_t\sim\pi$。

- 极限分布：若对任意初始状态，均有

$$\boxed{\lim_{t\to\infty}\Pr(X_t=j)=\pi_j,\quad j\in\mathcal{S},}$$

  且 π 是概率分布，则称 π 为极限分布。

- 极限分布一定是平稳分布，反之不一定。

如果状态空间 $\mathcal{S}$ 包含可数个 (countable) 元素:

- 常返: 如果从状态 i 出发总能回到状态 i，则称状态 i 是常返的。
- 正常返: 对于常返状态 i，如果从 i 出发首次返回 i 的时间的期望有限，称状态 i 是正常返的。
- 遍历马氏链: 所有状态不可约、非周期、正常返的马氏链称为遍历马氏链。若有限状态马氏链为不可约非周期的，则也必为正常返，因此也是遍历马氏链。

如果状态空间 $\mathcal{S}$ 是连续的:

- **转移密度分布**: 对于 $x, y \in \mathcal{S}$, 转移密度分布为 $F_{xy} = \Pr(X_{t+1} \leq y \mid X_t = x)$。
- **转移核**: 即转移密度函数

$$\boxed{K(x, y) = \frac{\partial F_{xy}}{\partial y}.}$$

- 细致平衡条件:
  - 离散状态空间: $\pi_i p_{ij} = \pi_j p_{ji}$.
  - 连续状态空间: $\pi_x K(x, y) = \pi_y K(y, x)$.
- 如果分布 π 满足细致平衡条件，则该分布为马氏链的平稳分布。

注：细致平衡是验证平稳分布的充分条件，可用于验证 *M–H* 算法的正确性。

1.2 Markov-Chain Monte-Carlo (MCMC)

MCMC 方法依赖于如下两个理论依据:

Theorem 1. 设 $\{X_t, t = 0, 1, \cdots\}$ 为遍历马氏链，则其极限分布为平稳分布且为唯一的平稳分布，即

$$\boxed{\pi_t \to \pi, \quad t \to \infty.}$$

这里 π_t 表示马氏链在时间 t 的边际分布, π 为极限分布。

Theorem 2 (Markov chain law of large numbers). 设 $\{X_t, t = 0, 1, \cdots\}$ 为遍历马氏链，且平稳分布为 π，则 X_t 依分布收敛到分布为 π 的随机变量 X。对任意函数 f, 当 $E_\pi|f(X)|$ 存在，则

$$\boxed{\frac{1}{n}\sum_{t=1}^{n} f(X_t) \to E_\pi f(X), \quad a.s.}$$

- 如果遍历马氏链的极限分布 π 为平稳分布，则当时间 t 很大时，随机过程样本 X_t 近似服从分布 π。
- 通过不断从马氏链中抽样，利用马尔可夫大数定律 (Theorem 2), 我们可以通过 Monte Carlo 方法研究 $E(f(X))$ 的性质。
  - Markov $-$ Chain $\rightarrow$ Generate samples from π
  - Monte $-$ Carlo $\rightarrow$
    Repeatedly generate samples from Markov Chain
- 在实际运用中，我们抛弃开始的一段序列，即对应 t 比较小的样本 X_t，这段序列称为*老化期*。

- 假设随机变量 X 服从分布 π。考虑利用 MCMC 算法估计参数 $\theta = E(h(X))$:
  - 构造具有平稳分布 π 的遍历马氏链 $\{x_t, t = 0, 1, \cdots, n\}$;
  - 抛弃前 k 个老化期样本;
  - 计算 Monte Carlo 估计量

$$\boxed{\hat{\theta} = \frac{1}{n-k} \sum_{t=k+1}^{n} h(x_t).}$$

- MCMC 主要应用在贝叶斯推断中，主要关心的问题是如何从后验分布 $f(\theta|x)$ 中抽样。然而在很多具体问题中，从后验分布 $f(\theta|x)$ 中生成独立样本非常困难，MCMC 提供了解决这类抽样问题的方法。

2 Metropolis-Hasting 算法

2.1 Metropolis-Hasting 算法

假设目标分布的密度函数为 f，Metropolis-Hasting 算法的一般性描述如下：

- 构造合适的的建议分布 (proposal distribution) $g(\cdot \mid x_t)$，这里 x_t 为 t 时刻的生成样本;
- 从建议分布 $g(\cdot \mid x_t)$ 中生成 y;
- 从均匀分布 Unif$(0,1)$ 中生成 u;
- 如果

$$u \leq \frac{f(y)g(x_t \mid y)}{f(x_t)g(y \mid x_t)},$$

则接受 y 并令 $x_{t+1} = y$, 否则令 $x_{t+1} = x_t$

则每一步中接受概率为

$$\alpha(x_t, y) = \min\left\{1, \frac{f(y)g(x_t \mid y)}{f(x_t)g(y \mid x_t)}\right\}.$$

- 建议分布的选择需要使得生成的马氏链收敛到目标分布 f，且满足不可约、正常返、非周期的正则化条件。
- 一个和目标分布具有相同的支撑集的建议分布一般会满足这些正则化条件。
- 建议分布 $g(\cdot \mid x_t)$ 的性质是影响 Metropolis-Hasting 算法的重要要素。
- 可以证明 Metropolis-Hasting 算法生成的马氏链收敛到平稳分布 f。
  - 证明 f 满足 *M-H* 算法构造的马尔可夫链的细致平衡条件 $f(x)K(x,y) = f(y)K(y,x)$（见 *p226*《统计计算使用 *R*》以及 *p149*《统计计算》）

例 2.1

使用 *Metropolis-Hasting* 算法生成服从 *Rayleigh* 分布的样本。其中 *Rayleigh* 分布的密度函数为

$$f(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)}, \quad x \geq 0, \sigma > 0.$$

- 取建议分布 $g(\cdot \mid x_t)$ 为 $\chi^2(x_t)$（密度函数为 $\frac{1}{2^{x_t/2}\Gamma(x_t/2)} x^{x_t/2-1} e^{-x/2}$）；
- 生成 $y \sim \chi^2(x_t)$, 以及 $u \sim \text{Unif}(0,1)$;
- 计算

$$\alpha(x_t, y) = \frac{f(y)g(x_t \mid y)}{f(x_t)g(y \mid x_t)};$$

- 如果 $u \leq \alpha(x_t, y)$, 则接受并令 $x_{t+1} = y$, 否则拒绝且令 $x_{t+1} = x_t$.

```r
# The pdf of Rayleigh distribution
f <- function(x, sigma) {
  if (any(x < 0)) return (0)
  stopifnot(sigma > 0)
  return((x / sigma^2) * exp(-x^2 / (2*sigma^2)))
}
```

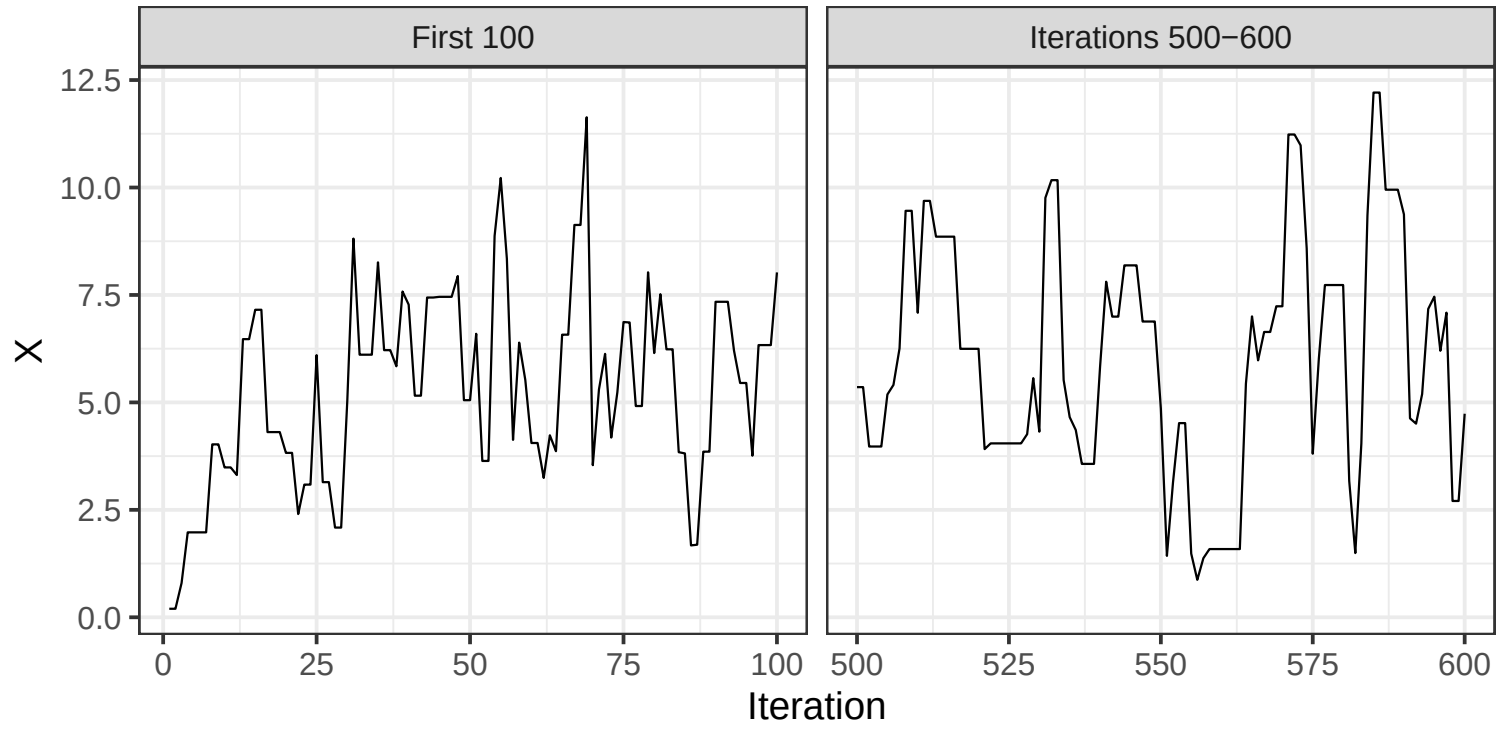
```r
m <- 10000
sigma <- 4
x <- numeric(m)
x[1] <- rchisq(1, df=1)
# The counter of the occurrence of failure.
k <- 0
u <- runif(m)
```

```r
for (i in 2:m) {
  xt <- x[i-1]
  y <- rchisq(1, df = xt)
  num <- f(y, sigma) * dchisq(xt, df = y)
  den <- f(xt, sigma) * dchisq(y, df = xt)
  if (u[i] <= num/den) x[i] <- y
  else {
    x[i] <- xt
    k <- k+1 #y is rejected
  }
}
print(k)
```

```
## [1] 3985
```

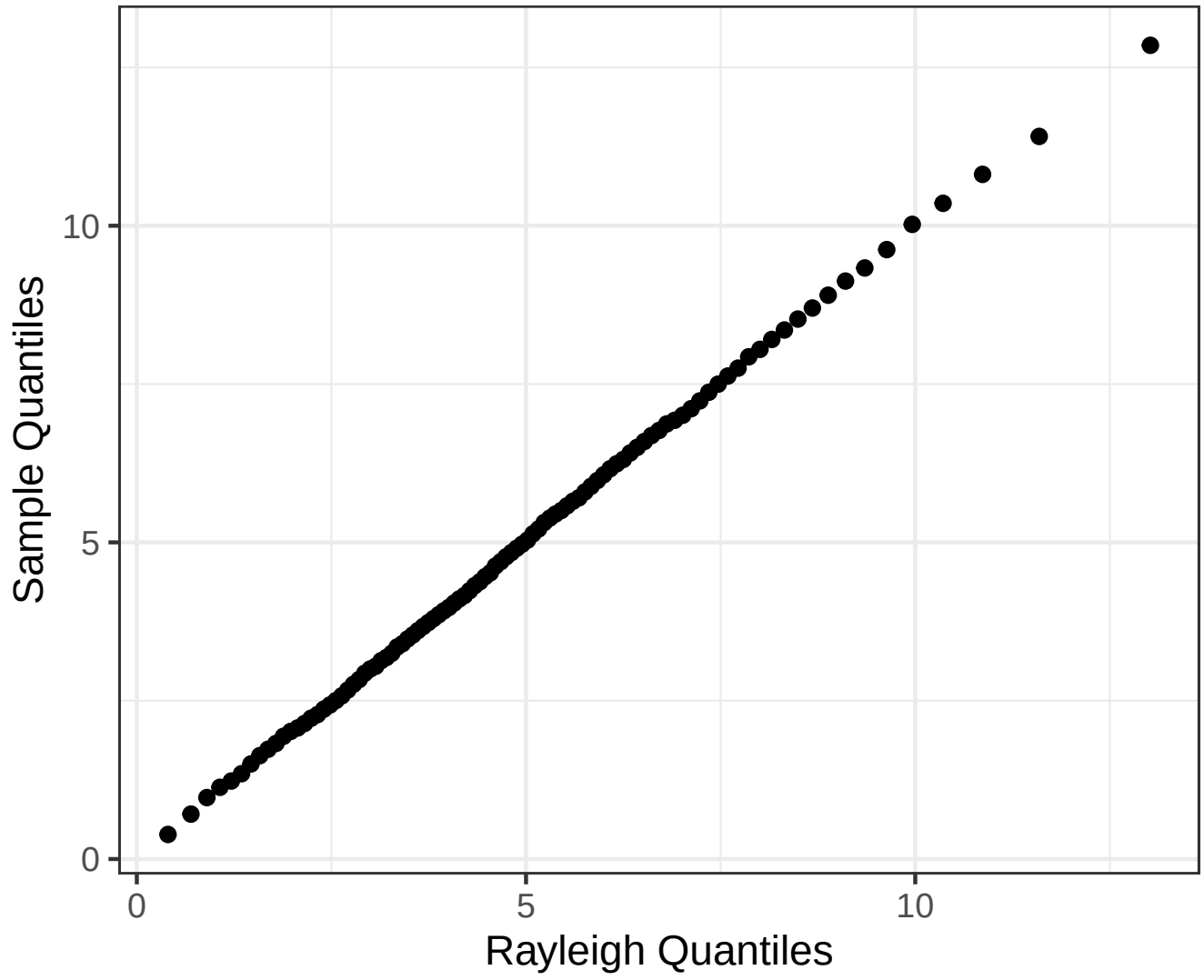
- 折线图：比较初期与后期的样本轨迹

```
index <- c(1:100, 500:600)
trace_data <- data.frame(
  iteration = index, value = x[index],
  period = factor(c(rep("First 100", 100),
                    rep("Iterations 500-600", 101)),
                  levels = c("First 100",
                             "Iterations 500-600")))
ggplot(trace_data, aes(iteration, value)) +
  geom_line(linewidth = 0.3) +
  facet_wrap(~period, nrow = 1, scales = "free_x") +
  labs(x = "Iteration", y = "X") +
  theme_bw()
```


First 100
Iterations 500−600
X
12.5
10.0
7.5
5.0
2.5
0.0
0
25
50
75
100
500
525
550
575
600
Iteration

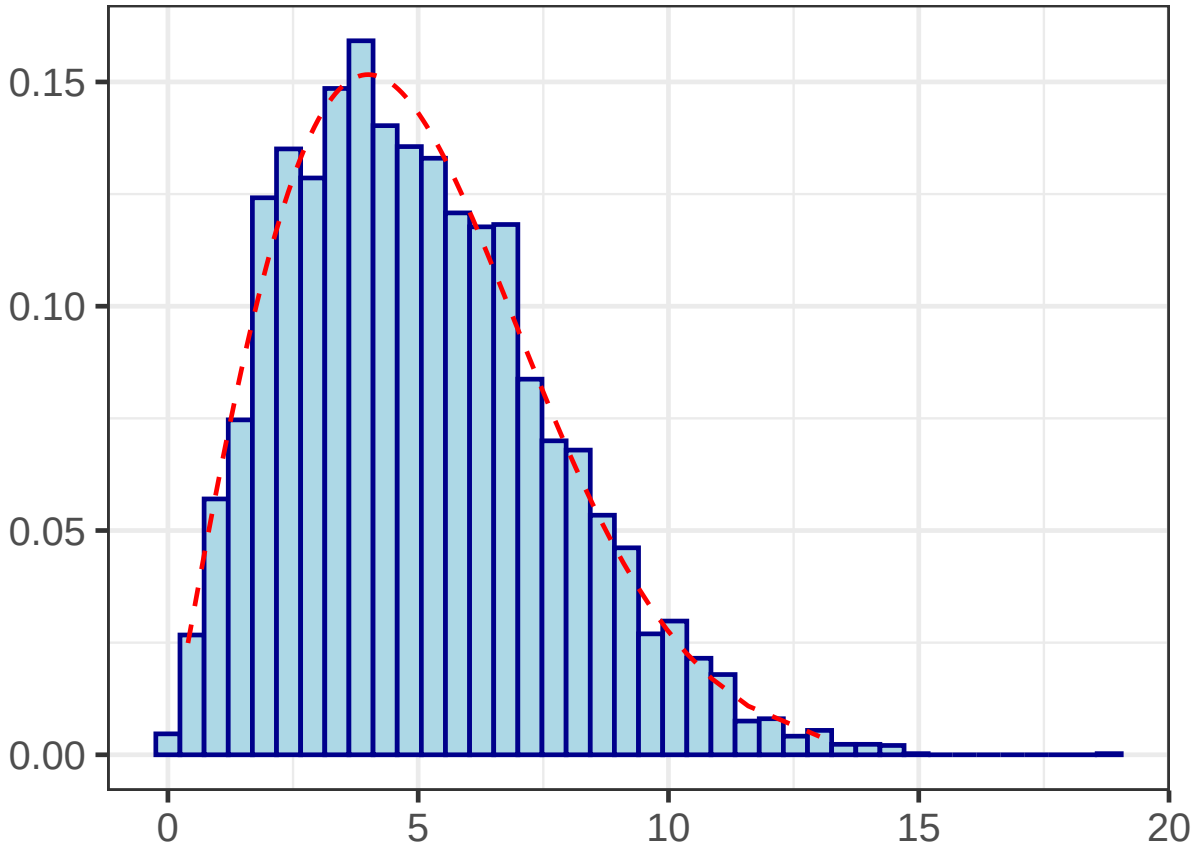
- QQ-plot

```
b <- 2001 #discard the burnin sample
y <- x[b:m]
a <- ppoints(100) # cut points
#quantiles of Rayleigh
QR <- sigma * sqrt(-2 * log(1 - a))
Q <- quantile(y, a)
qq_data <- data.frame(
  rayleigh = QR, sample = as.numeric(Q))
ggplot(qq_data, aes(x = rayleigh, y = sample)) +
  geom_point() +
  labs(x = "Rayleigh Quantiles",
       y = "Sample Quantiles") +
  theme_bw()
```


Sample Quantiles
10
5
0
0
5
10
Rayleigh Quantiles

- 直方图

```
data <- data.frame(x = y)
data_curve <- data.frame(x = QR, y = f(QR, sigma))
ggplot(data, aes(x = x))+
  geom_histogram(aes(y=after_stat(density)),
                 color="darkblue", fill="lightblue",
                 bins = 40)+
  geom_line(data = data_curve, aes(x = x, y = y),
            color = "red", linetype = "dashed") +
  labs(x = "", y = "")+
  theme_bw()
```


0.15
0.10
0.05
0.00
0
5
10
15
20

2.2 Metropolis 算法

- 在关于接受概率的不同假设下，我们进一步可以得到特殊的 Metropolis-Hasting 算法。
- 在 Metropolis 算法中，假设提议分布 $g(\cdot \mid x_t)$ 是对称的，即 $g(x \mid y) = g(y \mid x)$。则接受概率为
  $\alpha(x_t, y) = \min\left\{1, \frac{f(y)}{f(x_t)}\right\}$.
- Metropolis 算法是 Metropolis-Hasting 算法的一个特例。

2.2 Metropolis 算法：随机游走 Metropolis 算法

- 不同的对称提议分布 $g(\cdot \mid x_t)$ 会得到不同的 Metropolis 算法。
- 随机游走 Metropolis 算法是 Metropolis 算法的一个特例，假设 $g(y \mid x_t)$ 是 x_t 和 y 之间距离的函数，即

$$g(y \mid x_t) = h(|x_t - y|).$$

- 给定 $X_t = x_t$，候选变量 Y 的密度为 $g(y \mid x_t)$。定义增量 $Z = Y - x_t$，由变量变换 $y = x_t + z$（雅可比为 1），有

$$f_{Z|X_t}(z \mid x_t) = g(x_t + z \mid x_t) = h(|z|).$$

  该密度不依赖 x_t，且关于零对称。
- 因此上述提议等价于从对称分布 $q(z) = h(|z|)$ 中生成 Z，并令

$$Y = x_t + Z.$$

- 如果假设随机增量 $z \sim N(0, \sigma^2)$, 那么生成的马尔可夫链为高斯随机游走且 $y \sim N(x_t, \sigma^2)$.
- 随机游走 Metropolis 算法的收敛性对尺度参数非常敏感。
  - 当增量方差非常大时，大多数点都被拒绝。
  - 当增量方差非常小时，大多数点都被接受。
- 可以通过尝试具有不同尺度参数的建议函数，观察样本的拒绝率。拒绝率在$[0.15, 0.5]$范围内则马氏链有较好的性质。

例 2.2

使用随机游走 *Metropolis* 算法以及建议分布
$g(\cdot \mid x_t) = N(x_t, \sigma^2)$ 产生自由度为 ν 的 t 分布的随机样本。

- 计算接受概率为

$$\alpha(x_t, y) = \min\left\{1, \frac{f(y)}{f(x_t)}\right\} = \min\left\{1, \frac{(1 + y^2/\nu)^{-(\nu+1)/2}}{(1 + x_t^2/\nu)^{-(\nu+1)/2}}\right\}.$$

```r
rw.Metropolis <- function(nu, sigma, x0, N){
  x <- numeric(N)
  x[1] <- x0
  u <- runif(N)
  k <- 0
  for (i in 2:N) {
    y <- rnorm(1, x[i-1], sigma)
    if (u[i] <= (dt(y, nu) / dt(x[i-1], nu))) x[i]
    <- y
    else {
      x[i] <- x[i-1]
      k <- k + 1
    }
  }
return(list(x=x, k=k))
}
```

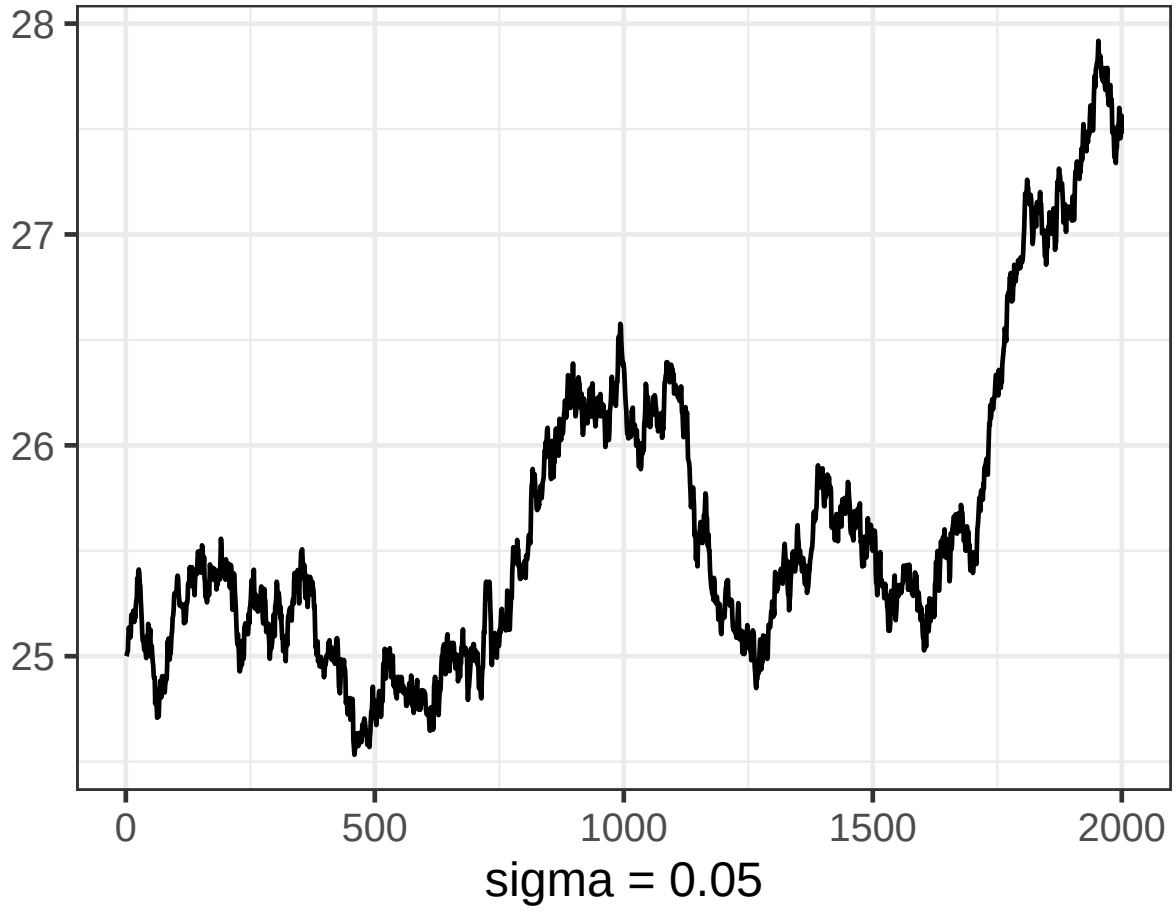
```r
# degrees of freedom for target Student t
# dist.
nu <- 4
N <- 2000
sigma <- c(.05, .5, 2, 16)
x0 <- 25
rw1 <- rw.Metropolis(nu, sigma[1], x0, N)
rw2 <- rw.Metropolis(nu, sigma[2], x0, N)
rw3 <- rw.Metropolis(nu, sigma[3], x0, N)
rw4 <- rw.Metropolis(nu, sigma[4], x0, N)
print(c(rw1$k, rw2$k, rw3$k, rw4$k)/N)
```

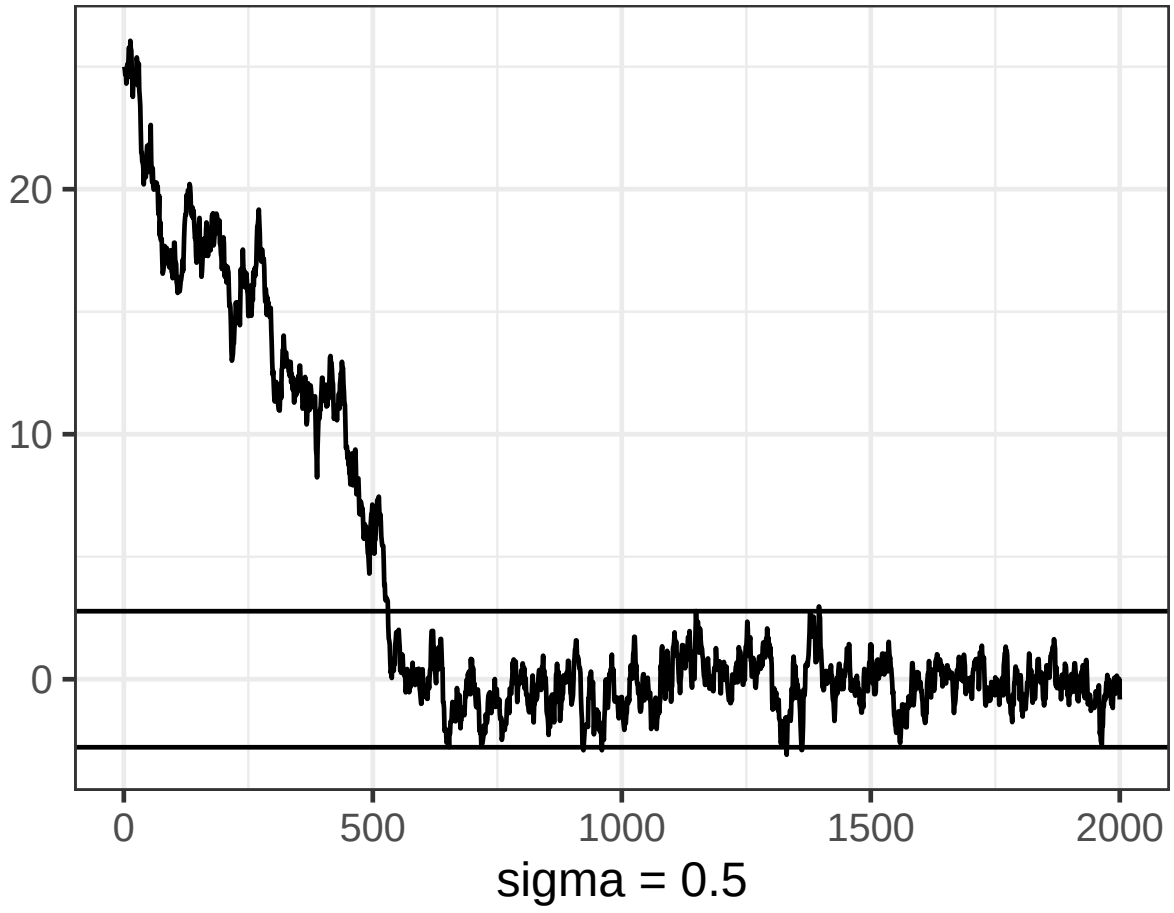
```
## [1] 0.0040 0.1295 0.4480 0.8980
```

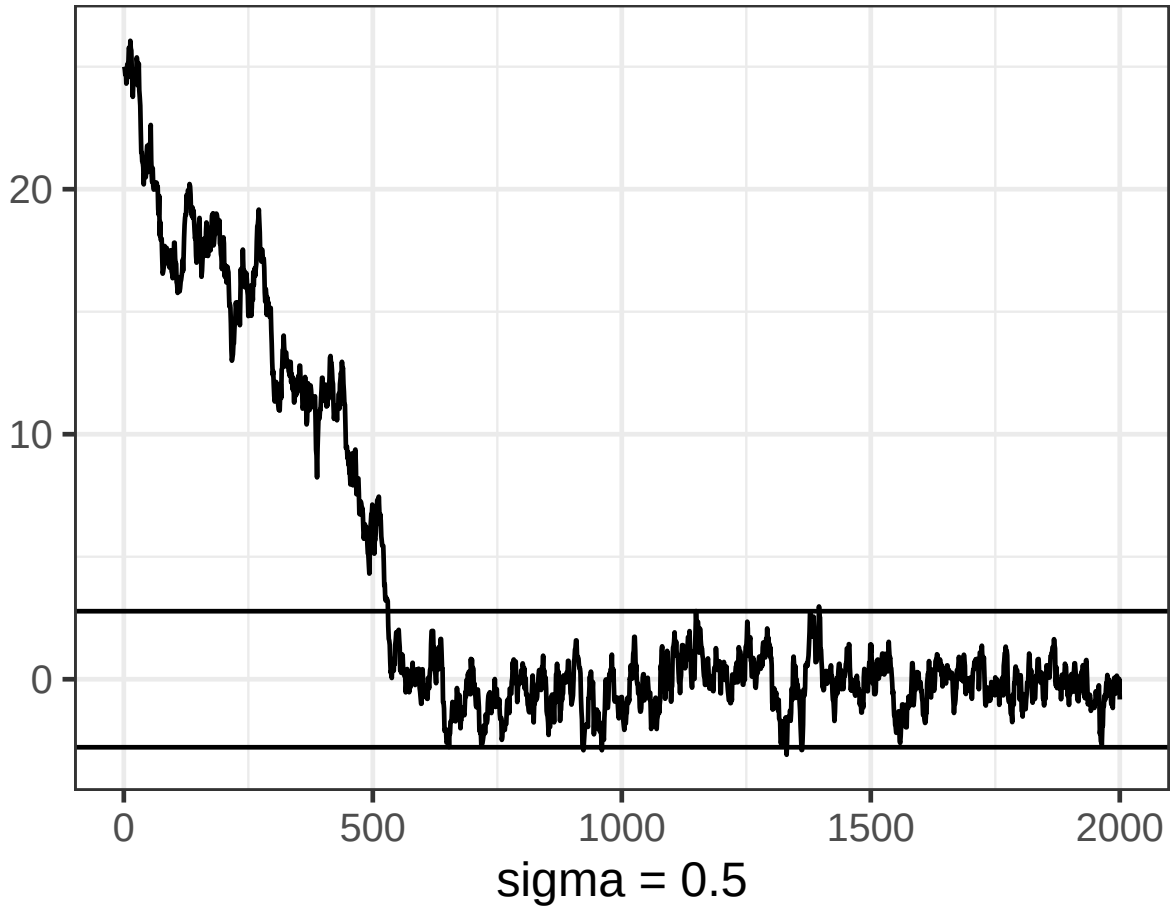
```r
rw <- cbind(rw1$x, rw2$x, rw3$x, rw4$x)

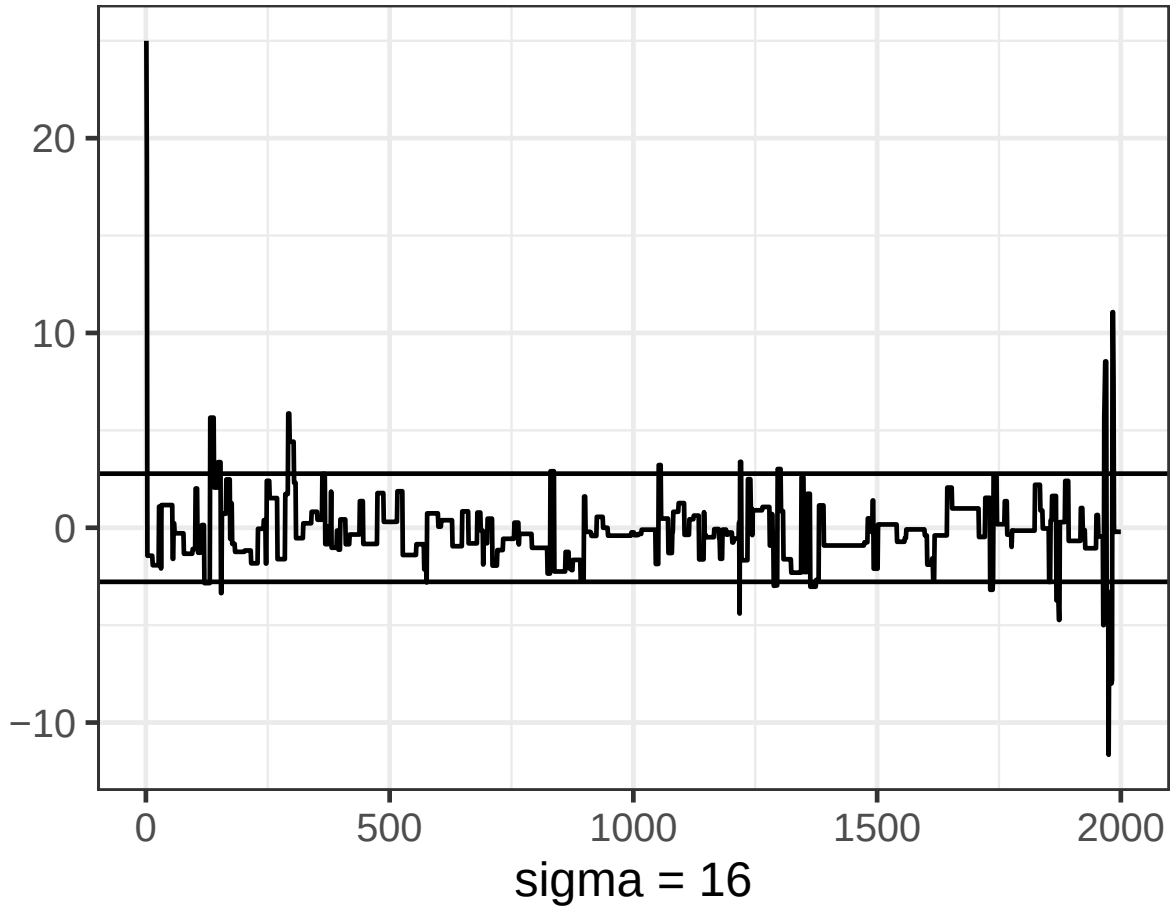
# j = 1
j <- 1
data <- data.frame(x = seq_along(rw[,j]), y = rw[,j])
g <- ggplot(data, aes(x = x, y = y))+
  geom_line()+
  labs(x = sprintf("sigma = %s",sigma[j]), y = "")+
  ylim(range(rw[,j]))+
  theme_bw()
print(g)
```

```r
# j = 2,3,4
for (j in 2:4) {
  data <- data.frame(x = seq_along(rw[,j]), y =
rw[,j])
  g <- ggplot(data, aes(x = x, y = y))+
    geom_line()+
    geom_hline(yintercept = qt(c(.025, .975),
    df=nu))+
    labs(x = sprintf("sigma = %s",sigma[j]), y = "")+
    ylim(range(rw[,j]))+
    theme_bw()
  print(g)
}
```


28
27
26
25
0
500
1000
1500
2000
sigma = 0.05


20
10
0
0
500
1000
1500
2000
sigma = 0.5


20
10
0
0
500
1000
1500
2000
sigma = 0.5


20
10
0
−10
0
500
1000
1500
2000
sigma = 16

```r
a <- c(.05, seq(.1, .9, .1), .95)
Q <- qt(a, nu)
rw <- cbind(rw1$x, rw2$x, rw3$x, rw4$x)
burn <- 500
mc <- rw[(burn+1):N, ]
Qrw <- apply(mc, 2, function(x) quantile(x, a))
colnames(Qrw) <- c("rw1", "rw2", "rw3", "rw4")
print(round(cbind(Q, Qrw), 3))
```

```
##           Q    rw1    rw2    rw3    rw4
## 5%  -2.132 24.834 -1.963 -2.506 -2.306
## 10% -1.533 24.950 -1.580 -1.631 -1.943
## 20% -0.941 25.114 -1.006 -1.045 -1.052
## 30% -0.569 25.300 -0.695 -0.619 -0.912
## 40% -0.271 25.439 -0.406 -0.375 -0.488
## 50%  0.000 25.591 -0.142 -0.124 -0.354
## 60%  0.271 25.803  0.107  0.159 -0.100
## 70%  0.569 26.120  0.359  0.379  0.182
## 80%  0.941 26.292  0.651  0.735  0.737
## 90%  1.533 27.101  1.188  1.355  1.067
## 95%  2.132 27.395  1.718  1.995  2.063
```

例 2.3

贝叶斯推断（《统计计算》例 *3.7.2*）考虑一个贝叶斯推断问题。在金融投资中，投资者经常把若干种证券组合在一起来减少风险。假设有 *5* 支股票的 $n = 250$ 个交易日的收益率记录，每个交易日都找出这 *5* 支股票收益率最高的一个，设 X_i 表示第 i 支股票在 n 个交易日中收益率为最高的次数 $(i = 1, \ldots, 5)$。设 $(X_1, \ldots, X_5)$ 服从多项分布，相应的概率假设为

$$p = \left(\frac{1}{3}, \frac{1-\beta}{3}, \frac{1-2\beta}{3}, \frac{2\beta}{3}, \frac{\beta}{3}\right),$$

其中 $\beta \in (0, 0.5)$ 为未知参数。假设 β 有先验分布 $\pi \sim \text{Unif}(0, 0.5)$。设 $(x_1, \ldots, x_5)$ 为 $(X_1, \ldots, X_5)$ 的观测值，则 β 的后验分布为

$$\begin{aligned} f(\beta \mid x_1, \ldots, x_5) &\propto p(x_1, \ldots, x_5 \mid \beta)\pi(\beta) \\ &\propto (1-\beta)^{x_2}(1-2\beta)^{x_3}(\beta)^{x_4+x_5} I_{(0,0.5)}(\beta). \end{aligned}$$

我们通过对后验分布抽样来估计 β。

- Metropolis 算法的接受概率为

$$
\begin{aligned}
&\alpha(\beta_t, y) \\
&= \min\left\{1, \frac{f(y \mid x_1, \ldots, x_5)}{f(\beta_t \mid x_1, \ldots, x_5)}\right\} \\
&= \min\left\{1, \left(\frac{1-y}{1-\beta_t}\right)^{x_2} \left(\frac{1-2y}{1-2\beta_t}\right)^{x_3} \left(\frac{y}{\beta_t}\right)^{x_4+x_5} I_{(0,0.5)}(y)\right\}
\end{aligned}
$$

- 建议分布 1: $g(y \mid \beta_t) \sim \text{Unif}(0, 0.5)$
- 建议分布 2: $g(y \mid \beta_t) \sim \text{Unif}(\beta_t - w, \beta_t + w)$
- 建议分布 3: $g(y \mid \beta_t) \sim N(\beta_t, \sigma^2)$

```r
set.seed(3)
beta <- .2
m <- 5000
burn <- 1000
days <- 250
x <- numeric(m)

i <- sample(1:5, size=days, replace=TRUE,
            prob=c(1, 1-beta, 1-2*beta, 2*beta,
            beta)/3)
win <- tabulate(i)
print(win)
```

```
## [1] 90 65 49 39  7
```

- 建议分布 1: 均匀分布

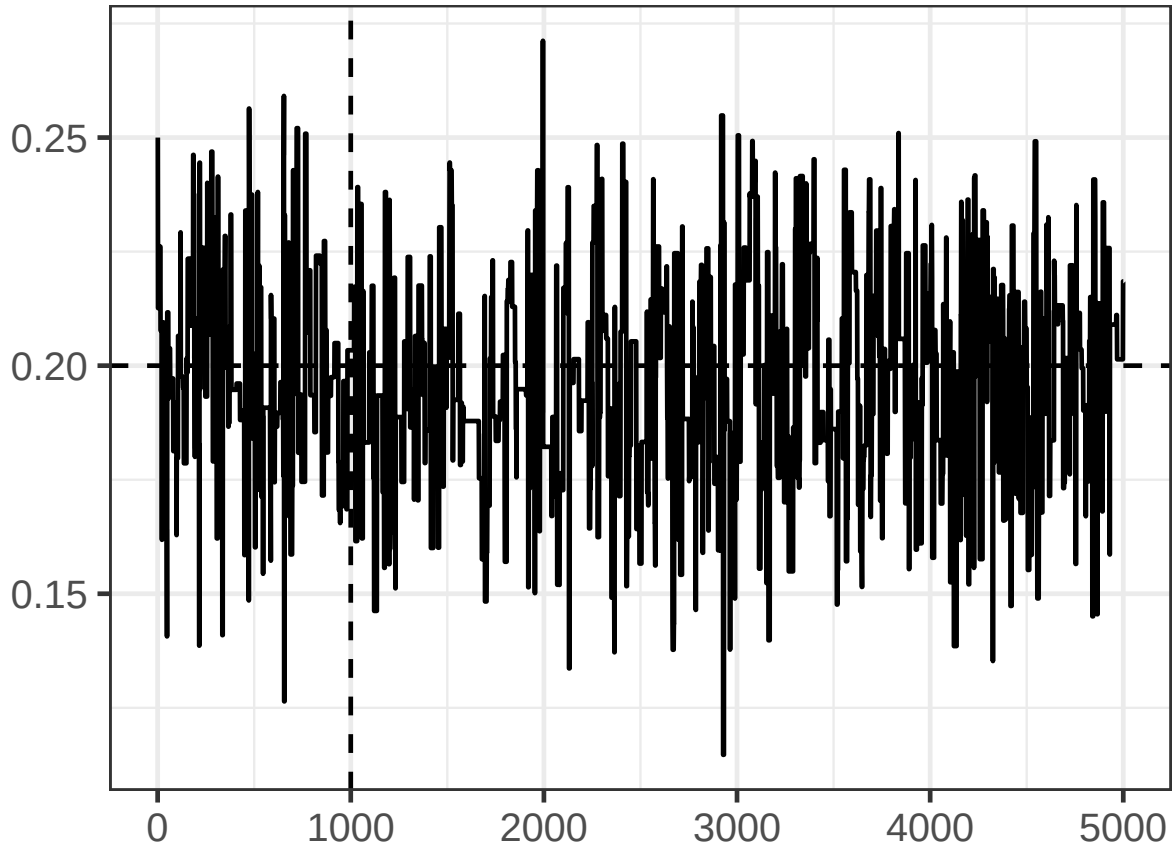
```
set.seed(3)
prob <- function(y, x, win) {
  if (y < 0 || y >= 0.5) 0
  else ((1-y)/(1-x))^win[2] *
  ((1-2*y)/(1-2*x))^win[3] *
    (y/x)^(win[4] + win[5])
}
```

```r
u <- runif(m) #for accept/reject step
v <- runif(m, 0, 0.5) #proposal distribution
x[1] <- .25
k <- 0
for (i in 2:m) {
  y <- v[i]
  if (u[i] <= prob(y, x[i-1], win)) x[i] <- y
  else {
    x[i] <- x[i-1]
    k <- k + 1
  }
}
k/m
```

```
## [1] 0.8582
```

- 拒绝率较高。

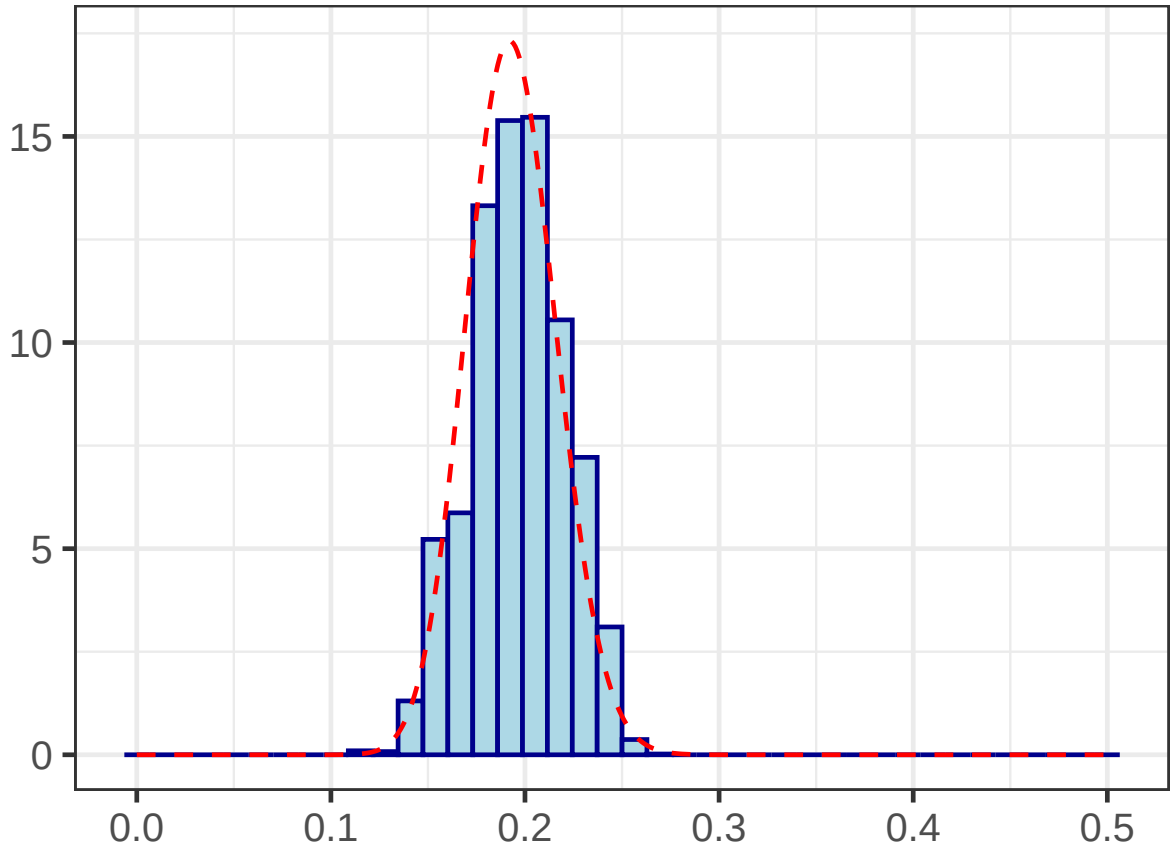
```r
data <- data.frame(x = seq_along(x), y = x)
g <- ggplot(data, aes(x = x, y = y))+
  geom_line()+
  geom_vline(xintercept = burn, linetype = "dashed")+
  geom_hline(yintercept = beta, linetype = "dashed")+
  labs(x = "", y = "")+
  ylim(range(x))+
  theme_bw()
print(g)
```


0.25
0.20
0.15
0
1000
2000
3000
4000
5000

理论后验密度

```r
log_beta_posterior <- function(b) {
  win[2] * log1p(-b) + win[3] * log1p(-2 * b) +
    (win[4] + win[5]) * log(b)
}
# Rescale to avoid underflow; density unchanged.
peak_beta <- optimize(log_beta_posterior,
                      c(0, 0.5), maximum=TRUE)$objective
beta_kernel <- function(b) exp(log_beta_posterior(b) - peak_beta)
beta_norm <- integrate(beta_kernel, 0, 0.5,
                       rel.tol=1e-10)$value
f_beta <- function(b) beta_kernel(b) / beta_norm
stopifnot(abs(integrate(f_beta, 0, 0.5)$value - 1) < 1e-8)
```

```r
xb <- x[-(1:burn)]
data <- data.frame(x = xb)
z <- seq(0, 0.5, length.out=501)
data_curve <- data.frame(x = z, y = f_beta(z))
ggplot(data, aes(x = x))+
  geom_histogram(aes(y=after_stat(density)),
                 color="darkblue", fill="lightblue",
                 bins = 40)+
  geom_line(data = data_curve, aes(x = x, y = y),
            color = "red", linetype = "dashed") +
  labs(x = "", y = "")+
  theme_bw()
```


15
10
5
0
0.0
0.1
0.2
0.3
0.4
0.5

```r
print(mean(xb))
```

```
## [1] 0.1961795
```

```r
print(sd(xb))
```

```
## [1] 0.02389386
```

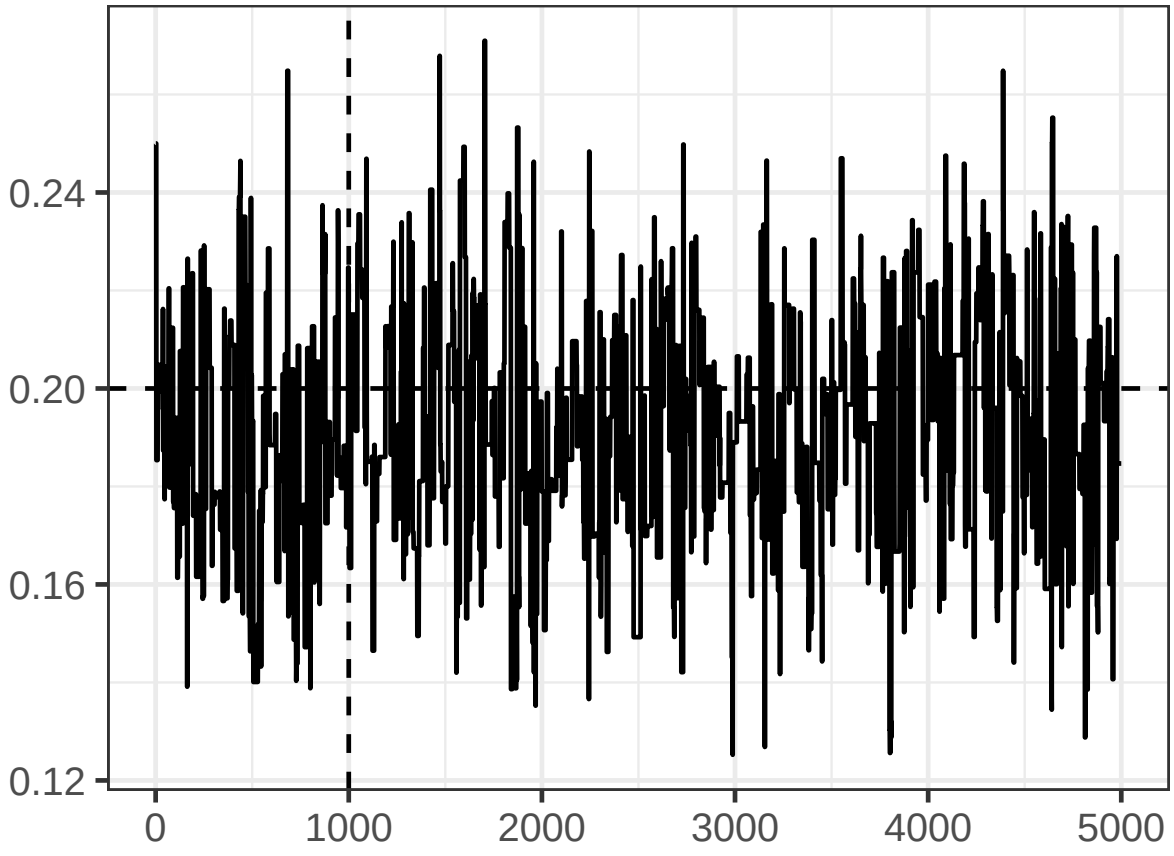
- 建议分布 2: 均匀随机游走

```
w <- .25
u <- runif(m) #for accept/reject step
v <- runif(m, -w, w) #proposal distribution
x[1] <- .25
k <- 0
for (i in 2:m) {
  y <- x[i-1] + v[i]
  if (u[i] <= prob(y, x[i-1], win)) x[i] <- y
  else {
    x[i] <- x[i-1]
    k <- k + 1
  }
}
k/m
```

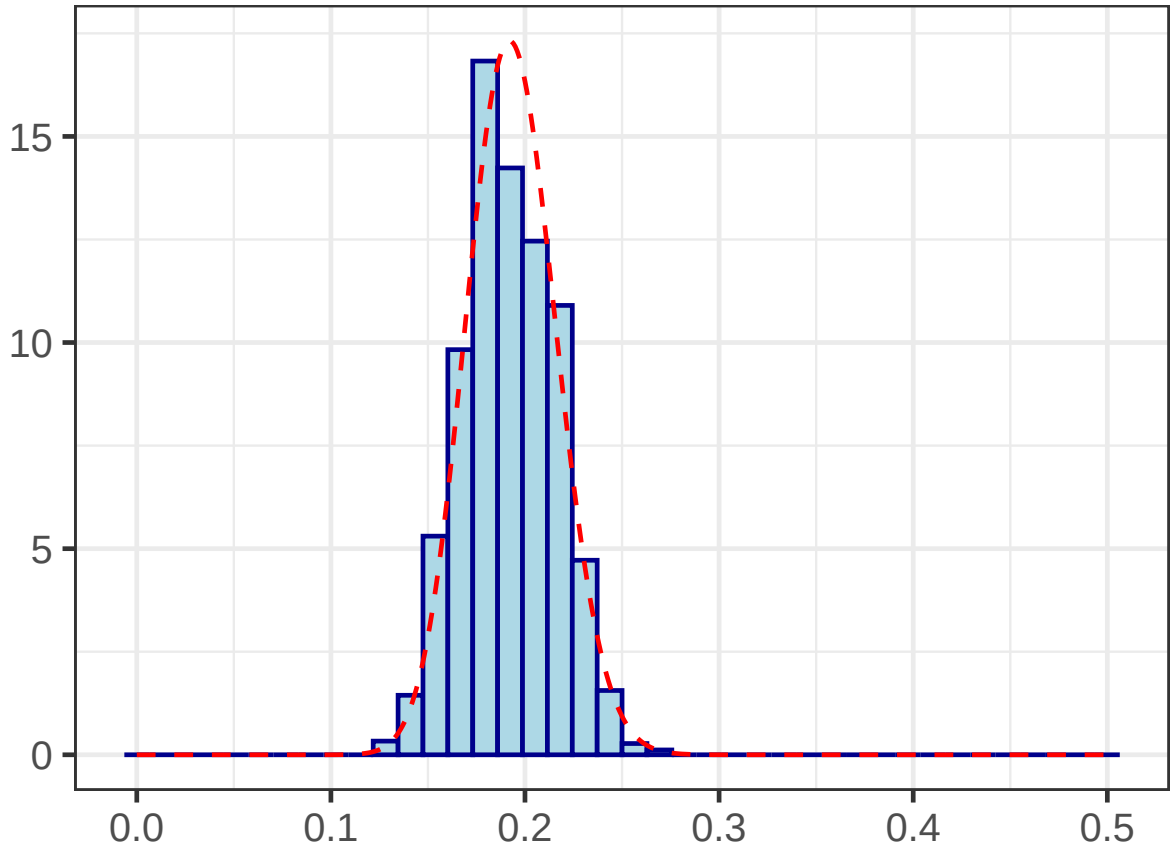
```
## [1] 0.8528
```

- 拒绝率较大，可以通过减小均匀随机游走的范围（即 w）来降低拒绝率

```r
data <- data.frame(x = seq_along(x), y = x)
g <- ggplot(data, aes(x = x, y = y))+
  geom_line()+
  geom_vline(xintercept = burn, linetype = "dashed")+
  geom_hline(yintercept = beta, linetype = "dashed")+
  labs(x = "", y = "")+
  ylim(range(x))+
  theme_bw()
print(g)
```


0.24
0.20
0.16
0.12
0
1000
2000
3000
4000
5000

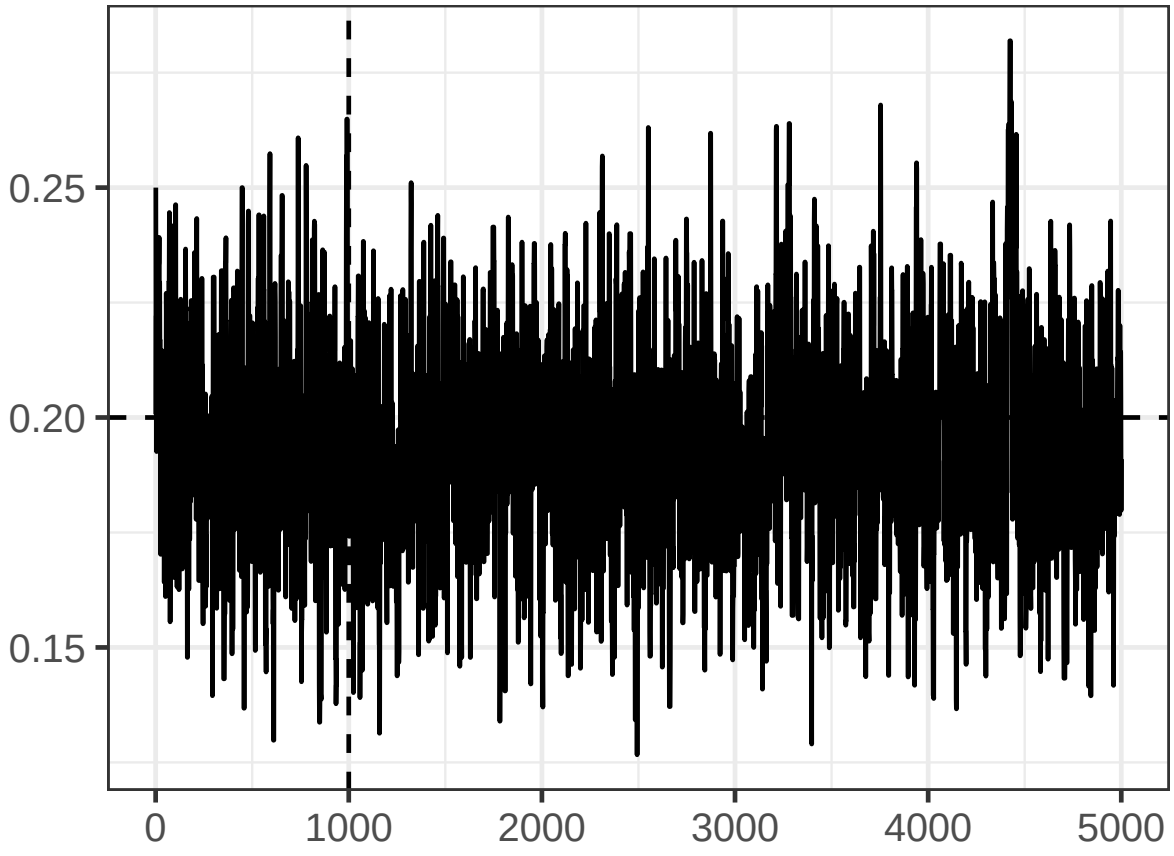
```r
xb <- x[- (1:burn)]
data <- data.frame(x = xb)
z <- seq(0, 0.5, length.out=501)
data_curve <- data.frame(x = z, y = f_beta(z))
ggplot(data, aes(x = x))+
  geom_histogram(aes(y=after_stat(density)),
                 color="darkblue", fill="lightblue",
                 bins = 40)+
  geom_line(data = data_curve, aes(x = x, y = y),
            color = "red", linetype = "dashed") +
  labs(x = "", y = "")+
  theme_bw()
```


15
10
5
0
0.0
0.1
0.2
0.3
0.4
0.5

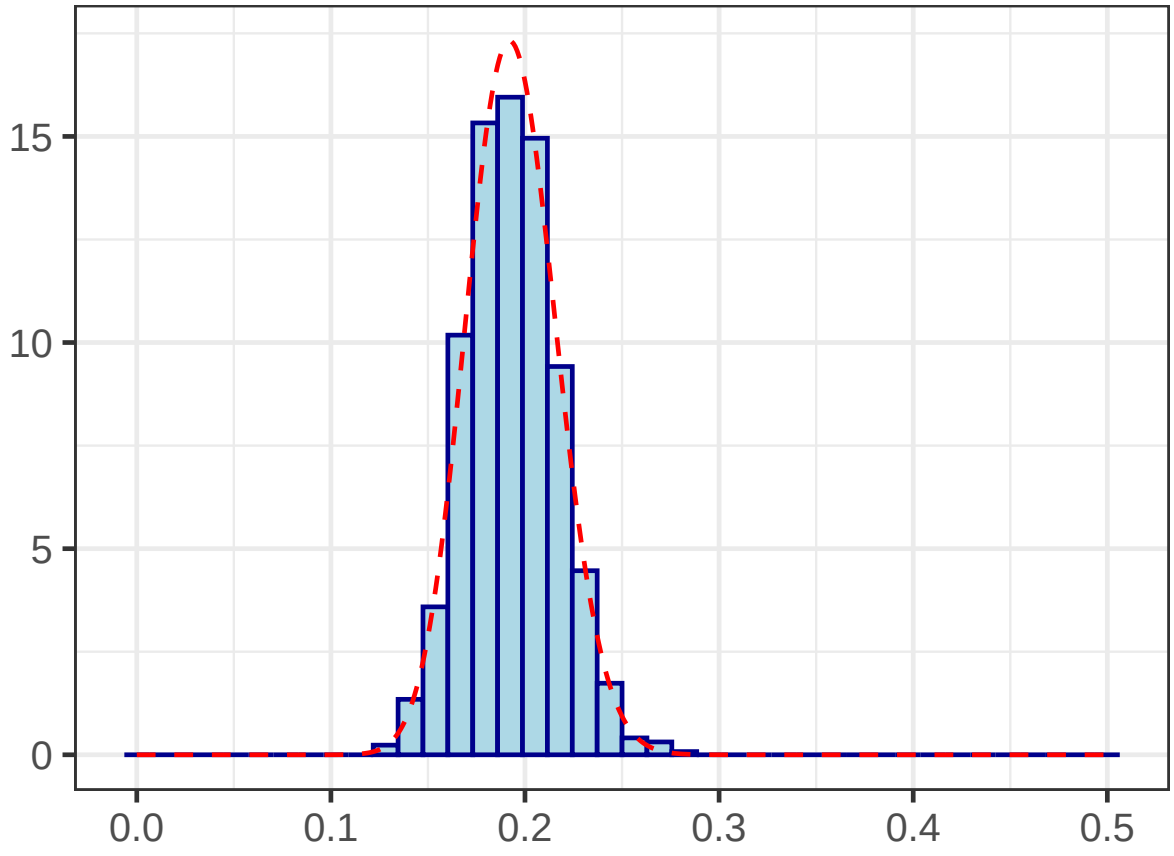
```r
w <- .05
u <- runif(m) #for accept/reject step
v <- runif(m, -w, w) #proposal distribution
x[1] <- .25
k <- 0
for (i in 2:m) {
  y <- x[i-1] + v[i]
  if (u[i] <= prob(y, x[i-1], win)) x[i] <- y
  else {
    x[i] <- x[i-1]
    k <- k + 1
  }
}
k/m
```

```
## [1] 0.3978
```

```r
data <- data.frame(x = seq_along(x), y = x)
g <- ggplot(data, aes(x = x, y = y))+
  geom_line()+
  geom_vline(xintercept = burn, linetype = "dashed")+
  geom_hline(yintercept = beta, linetype = "dashed")+
  labs(x = "", y = "")+
  ylim(range(x))+
  theme_bw()
print(g)
```


0.25
0.20
0.15
0
1000
2000
3000
4000
5000

```r
xb <- x[- (1:burn)]
data <- data.frame(x = xb)
z <- seq(0, 0.5, length.out=501)
data_curve <- data.frame(x = z, y = f_beta(z))
ggplot(data, aes(x = x))+
  geom_histogram(aes(y=after_stat(density)),
                 color="darkblue", fill="lightblue",
                 bins = 40)+
  geom_line(data = data_curve, aes(x = x, y = y),
            color = "red", linetype = "dashed") +
  labs(x = "", y = "")+
  theme_bw()
```


15
10
5
0
0.0
0.1
0.2
0.3
0.4
0.5

```r
print(mean(xb))
```

```
## [1] 0.1931318
```

```r
print(sd(xb))
```

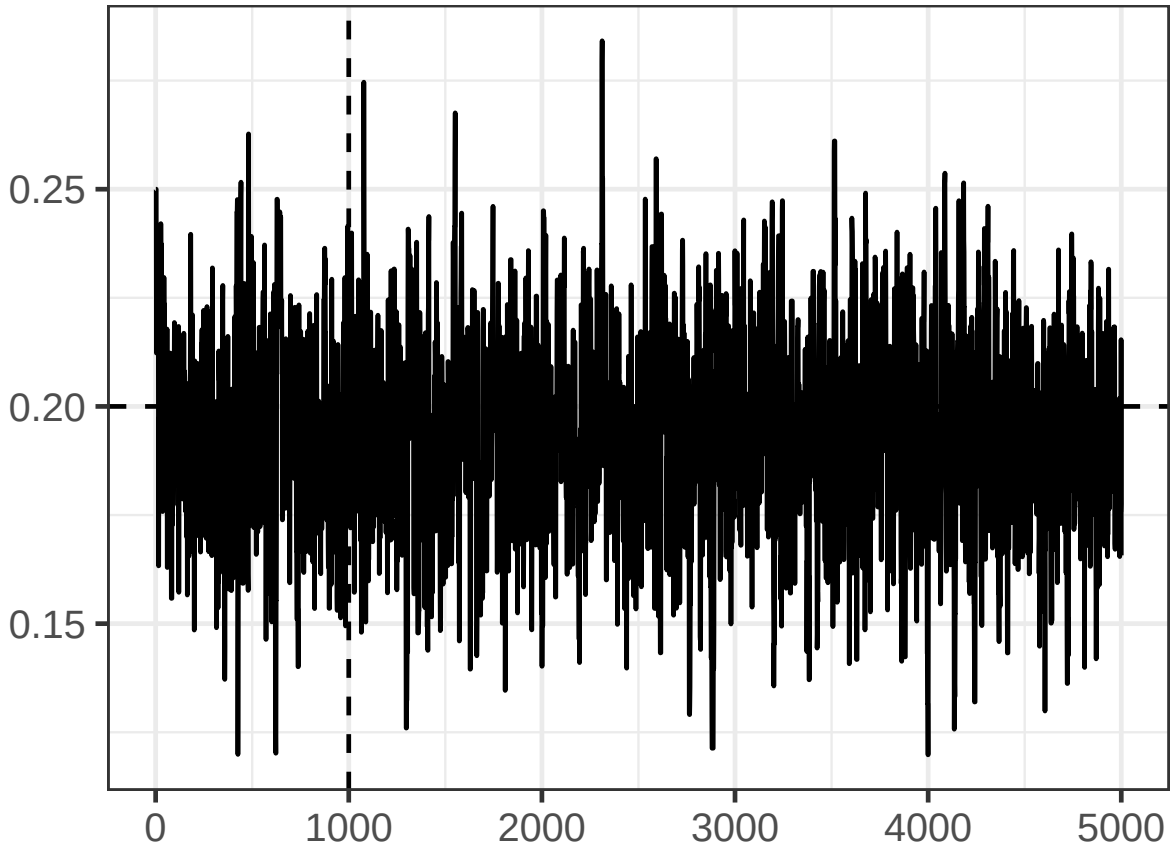
```
## [1] 0.02300047
```

- 建议分布 3: 高斯随机游走

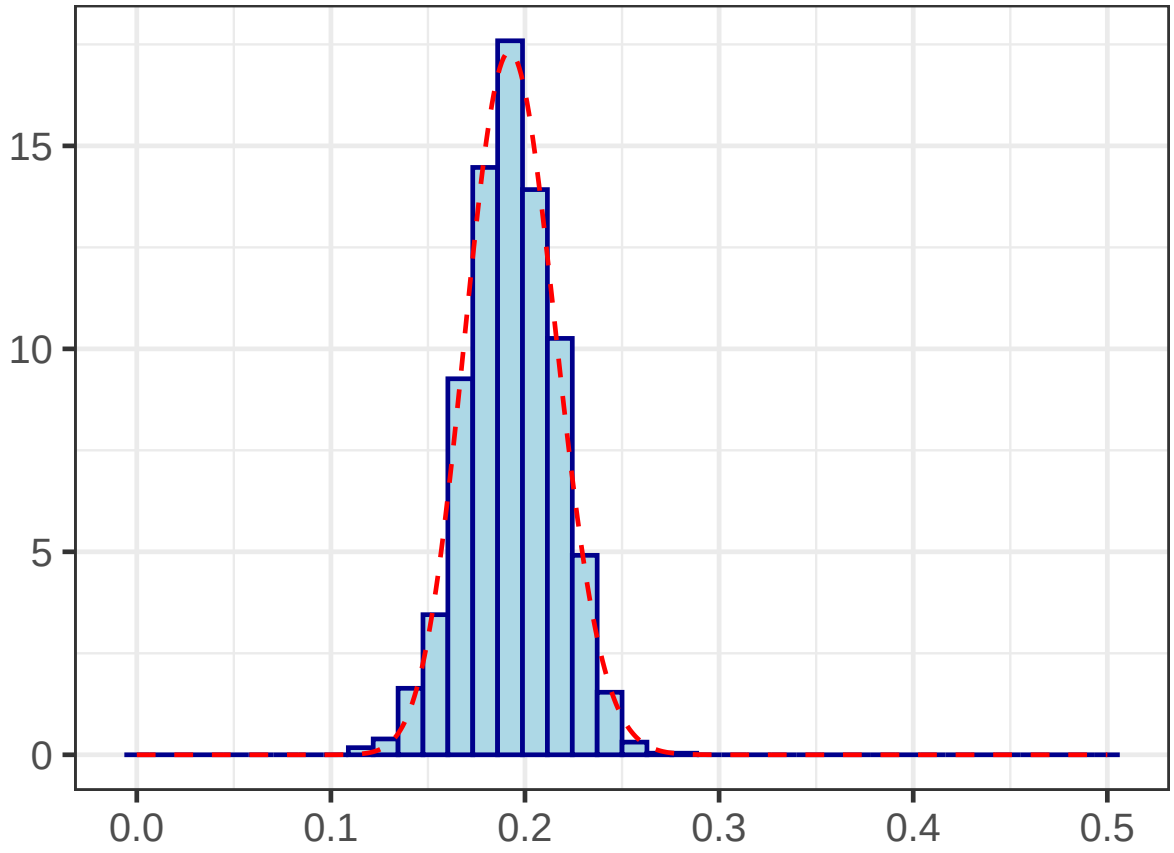
```
u <- runif(m) #for accept/reject step
sigma <- 0.03
v <- rnorm(m, 0, sigma) #proposal distribution
x[1] <- .25
k <- 0
for (i in 2:m) {
  y <- x[i-1] + v[i]
  if (u[i] <= prob(y, x[i-1], win)) x[i] <- y
  else {
    x[i] <- x[i-1]
    k <- k + 1
  }
}
k/m
```

```
## [1] 0.3708
```

```r
data <- data.frame(x = seq_along(x), y = x)
g <- ggplot(data, aes(x = x, y = y))+
  geom_line()+
  geom_vline(xintercept = burn, linetype = "dashed")+
  geom_hline(yintercept = beta, linetype = "dashed")+
  labs(x = "", y = "")+
  ylim(range(x))+
  theme_bw()
print(g)
```


0.25
0.20
0.15
0
1000
2000
3000
4000
5000

```r
xb <- x[- (1:burn)]
data <- data.frame(x = xb)
z <- seq(0, 0.5, length.out=501)
data_curve <- data.frame(x = z, y = f_beta(z))
ggplot(data, aes(x = x))+
  geom_histogram(aes(y=after_stat(density)),
                 color="darkblue", fill="lightblue",
                 bins = 40)+
  geom_line(data = data_curve, aes(x = x, y = y),
            color = "red", linetype = "dashed") +
  labs(x = "", y = "")+
  theme_bw()
```


15
10
5
0
0.0
0.1
0.2
0.3
0.4
0.5

```r
print(mean(xb))
```

```
## [1] 0.192901
```

```r
print(sd(xb))
```

```
## [1] 0.02301409
```

2.3 独立性样本生成器

- Metropolis-Hasting 算法的另一个特例是独立性样本生成器，这里假设建议分布独立于马氏链的前一步状态，即

$$g(y \mid x_t) = g(y).$$

- 在独立性样本生成器中，接受概率

$$\alpha(x_t, y) = \min\left\{1, \frac{f(y)g(x_t)}{f(x_t)g(y)}\right\}.$$

- 当建议分布 g 和目标分布 f 接近时，独立性样本生成器表现较好。但是当两者差异很大时，则表现较差。
- 例如，目标分布为 $f = N(0,1)$:
  - 若建议分布 $g = N(0,1) = f$，则接受概率恒为 1，得到独立的目标分布样本。
  - 若建议分布 $g = N(5,1)$，候选点多在 5 附近，偏离目标分布的主要区域。若当前 $x = 0$、候选 $y = 5$，则

$$\alpha(0,5) = \min\left\{1, \frac{f(5)g(0)}{f(0)g(5)}\right\} = e^{-25} \approx 1.4 \times 10^{-11}.$$

    几乎必然拒绝，链容易长时间停在原地。

例 2.4

假设从如下的正态混合分布中抽取样本 $(z_1, \dots, z_n)$,

$$pN(\mu_1, \sigma_1^2) + (1-p)N(\mu_2, \sigma_2^2).$$

假设参数 $\mu_1, \mu_2, \sigma_1, \sigma_2$ 已知, p 的先验分布为 $\pi \sim \text{Unif}(0,1)$ 如何利用独立性样本生成器从参数 p 的后验分布中生成样本，并估计 p。

- 令 f_1 和 f_2 为 $N(\mu_1, \sigma_1^2)$ 和 $N(\mu_2, \sigma_2^2)$ 的密度函数，则样本 $(z_1, \dots, z_n)$ 的似然函数为

$$p(z_1, \dots, z_n \mid p) = \Pi_{j=1}^n \{pf_1(z_j) + (1-p)f_2(z_j)\}.$$

- 目标分布为 p 的后验分布:

$$\begin{aligned} f(p \mid z_1, \dots, z_n) &\propto p(z_1, \dots, z_n \mid p)\pi(p) \\ &\propto \Pi_{j=1}^n \{pf_1(z_j) + (1-p)f_2(z_j)\} I_{(0,1)}(p) \end{aligned}$$

- 考虑建议分布为 Beta(a, b), 即

$$g(y) \propto y^{a-1}(1-y)^{b-1}.$$

- 接受概率为

$$\begin{aligned}
&\alpha(x_t, Y) \\
&= \min\left\{1, \frac{f(Y)g(x_t)}{f(x_t)g(Y)}\right\} \\
&= \min\Bigg\{1, \\
&\quad \frac{x_t^{a-1}(1-x_t)^{b-1}\Pi_{j=1}^{n}\{yf_1(z_j)+(1-y)f_2(z_j)\}}{y^{a-1}(1-y)^{b-1}\Pi_{j=1}^{n}\{x_t f_1(z_j)+(1-x_t)f_2(z_j)\}} I_{(0,1)}(y)\Bigg\}
\end{aligned}$$

- 模拟试验中，已知 $p = 0.2, \mu_1 = 0, \mu_2 = 5, \sigma_1 = \sigma_2 = 1.$
- 考虑建议分布为 $\text{Beta}(1, 1)$

```
set.seed(3)
m <- 5000 #length of chain
xt <- numeric(m)
a <- 1 #parameter of Beta(a,b) proposal dist.
b <- 1 #parameter of Beta(a,b) proposal dist.
p <- .2 #mixing parameter
n <- 30 #sample size
mu <- c(0, 5) #parameters of the normal densities
sigma <- c(1, 1)
k <- 0 # the number of rejections.
```

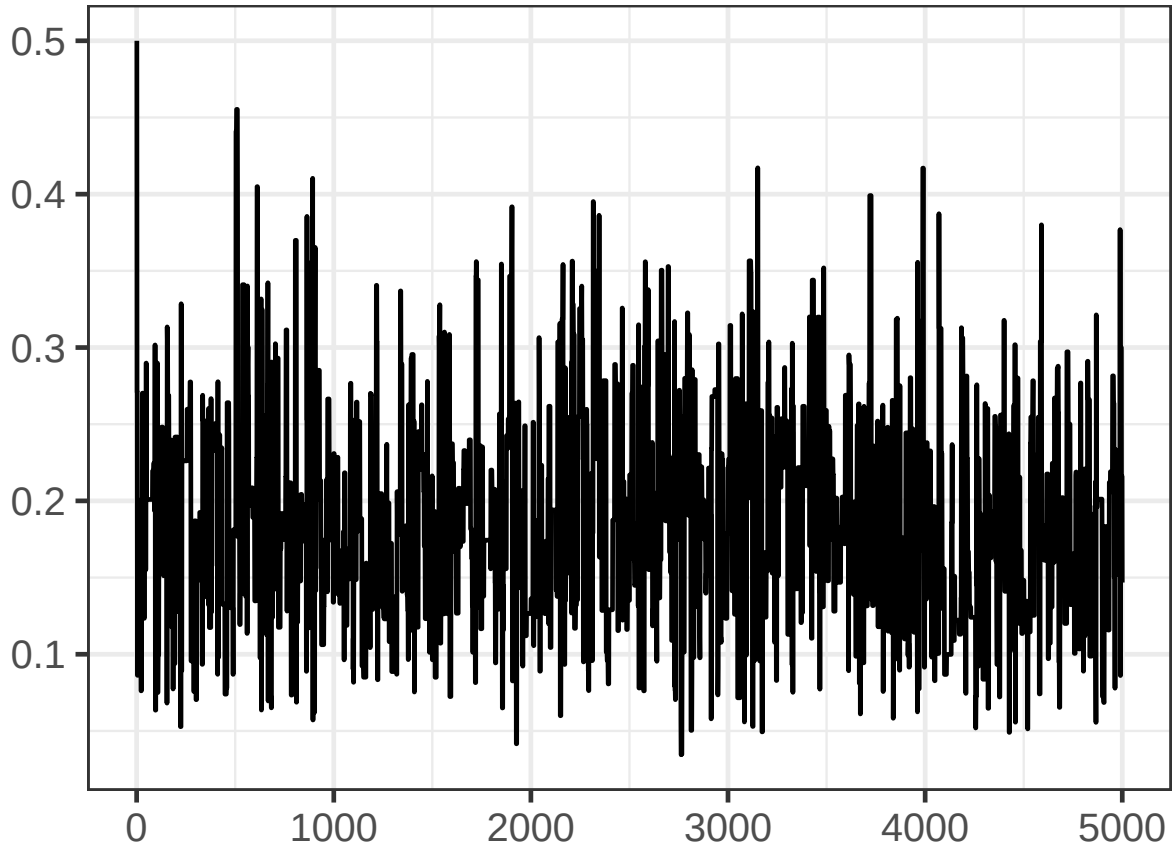
```r
# generate the observed sample
i <- sample(1:2, size=n, replace=TRUE, prob=c(p,
1-p))
x <- rnorm(n, mu[i], sigma[i])

# generate the independence sampler chain
u <- runif(m)
y <- rbeta(m, a, b) #proposal distribution
xt[1] <- .5
```

```
for (i in 2:m) {
  fy <- y[i] * dnorm(x, mu[1], sigma[1]) +
    (1-y[i]) * dnorm(x, mu[2], sigma[2])
  fx <- xt[i-1] * dnorm(x, mu[1], sigma[1]) +
    (1-xt[i-1]) * dnorm(x, mu[2], sigma[2])
  r <- prod(fy / fx) *
    (xt[i-1]^(a-1) * (1-xt[i-1])^(b-1)) /
    (y[i]^(a-1) * (1-y[i])^(b-1))
  if (u[i] <= r) xt[i] <- y[i]
  else {
    xt[i] <- xt[i-1]
    k <- k + 1
  }
}
k/m
```

```
## [1] 0.7882
```

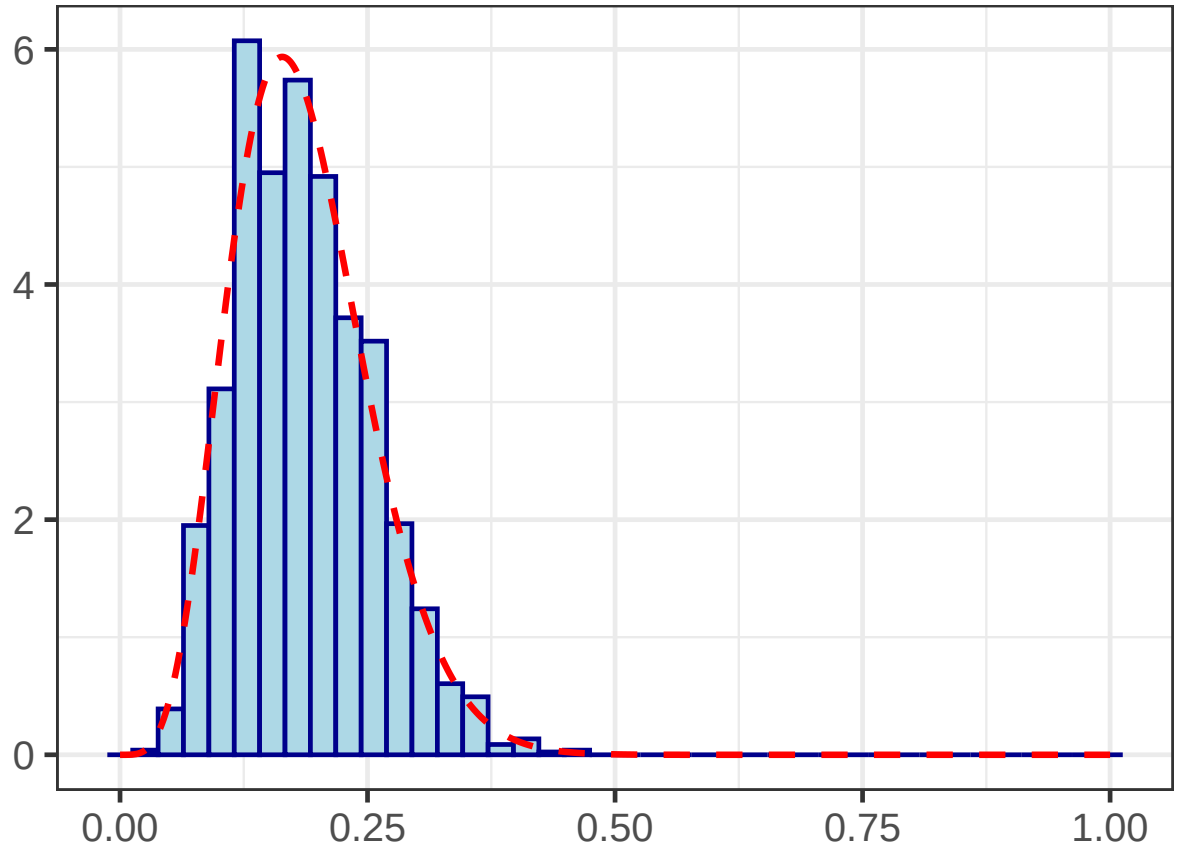
```r
data <- data.frame(x = seq_along(xt), y = xt)
g <- ggplot(data, aes(x = x, y = y))+
  geom_line()+
  labs(x = "", y = "")+
  theme_bw()
print(g)
```


0.5
0.4
0.3
0.2
0.1
0
1000
2000
3000
4000
5000

实际后验密度的计算

```
f1 <- dnorm(x, mu[1], sigma[1])
f2 <- dnorm(x, mu[2], sigma[2])
logpost <- function(p) {
  sum(log(p * f1 + (1 - p) * f2))
}
peak <- optimize(logpost, c(0, 1), maximum=TRUE)$objective
kernel <- function(p) {
  sapply(p, function(v) exp(logpost(v) - peak))
}
normalizer <- integrate(kernel, 0, 1, rel.tol=1e-10)$value
p_grid <- seq(0, 1, length.out=1001)
posterior <- data.frame(
  p = p_grid, density = kernel(p_grid) / normalizer)
stopifnot(abs(integrate(function(p) kernel(p) / normalizer,
                        0, 1)$value - 1) < 1e-7)
```

```r
xb <- xt[101:m]
data <- data.frame(x = xb)
ggplot(data, aes(x = x))+
  geom_histogram(aes(y=after_stat(density)),
                 color="darkblue", fill="lightblue",
                 bins = 40)+
  geom_line(data = posterior,
            aes(x = p, y = density),
            color = "red", linetype = "dashed",
            linewidth = 0.7) +
  labs(x = "", y = "")+
  theme_bw()
```


6
4
2
0
0.00
0.25
0.50
0.75
1.00

- 考虑建议分布为 Beta$(5, 2)$

```
m <- 5000 #length of chain
xt <- numeric(m)
a <- 5 #parameter of Beta(a,b) proposal dist.
b <- 2 #parameter of Beta(a,b) proposal dist.
p <- .2 #mixing parameter
n <- 30 #sample size
mu <- c(0, 5) #parameters of the normal densities
sigma <- c(1, 1)
k <- 0 # the number of rejections.
```

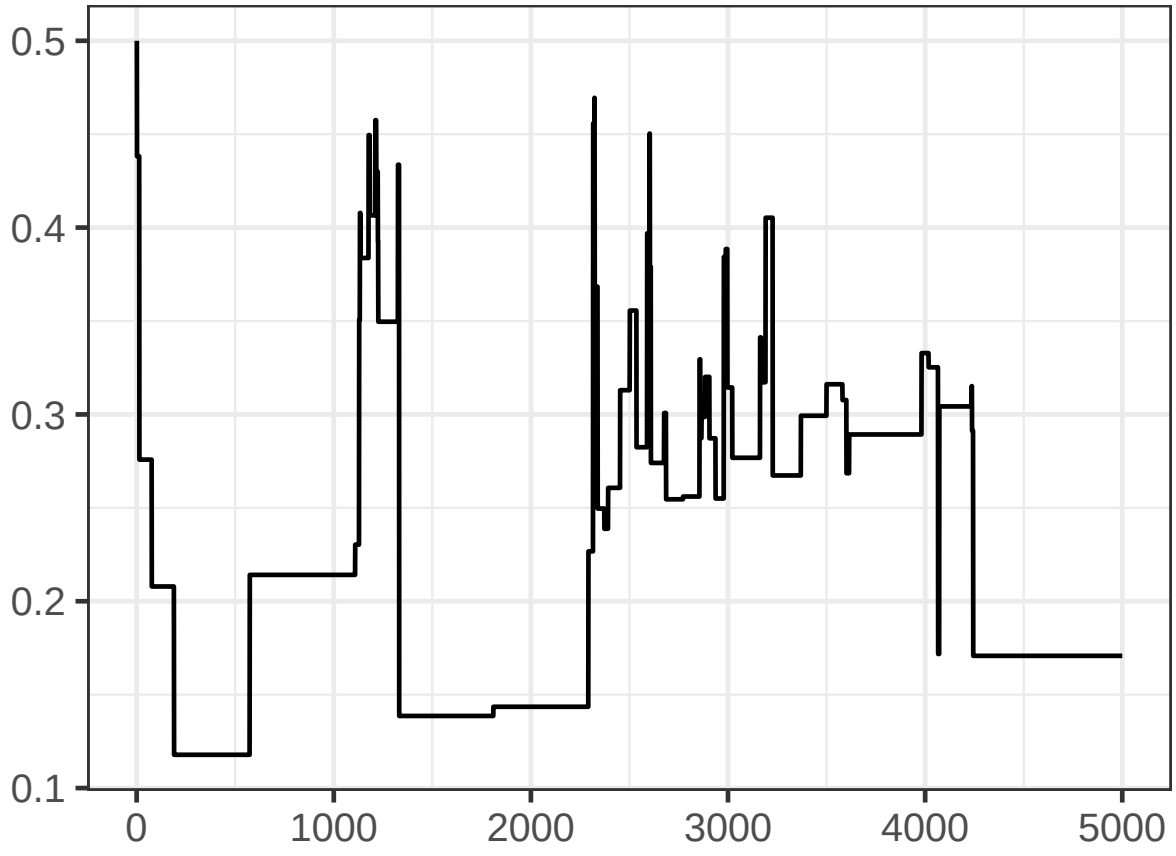
```r
# generate the observed sample
i <- sample(1:2, size=n, replace=TRUE, prob=c(p,
1-p))
x <- rnorm(n, mu[i], sigma[i])

# generate the independence sampler chain
u <- runif(m)
y <- rbeta(m, a, b) #proposal distribution
xt[1] <- .5
```

```
for (i in 2:m) {
  fy <- y[i] * dnorm(x, mu[1], sigma[1]) +
    (1-y[i]) * dnorm(x, mu[2], sigma[2])
  fx <- xt[i-1] * dnorm(x, mu[1], sigma[1]) +
    (1-xt[i-1]) * dnorm(x, mu[2], sigma[2])
  r <- prod(fy / fx) *
    (xt[i-1]^(a-1) * (1-xt[i-1])^(b-1)) /
    (y[i]^(a-1) * (1-y[i])^(b-1))
  if (u[i] <= r) xt[i] <- y[i]
  else {
    xt[i] <- xt[i-1]
    k <- k + 1
  }
}
k/m
```

```
## [1] 0.9876
```

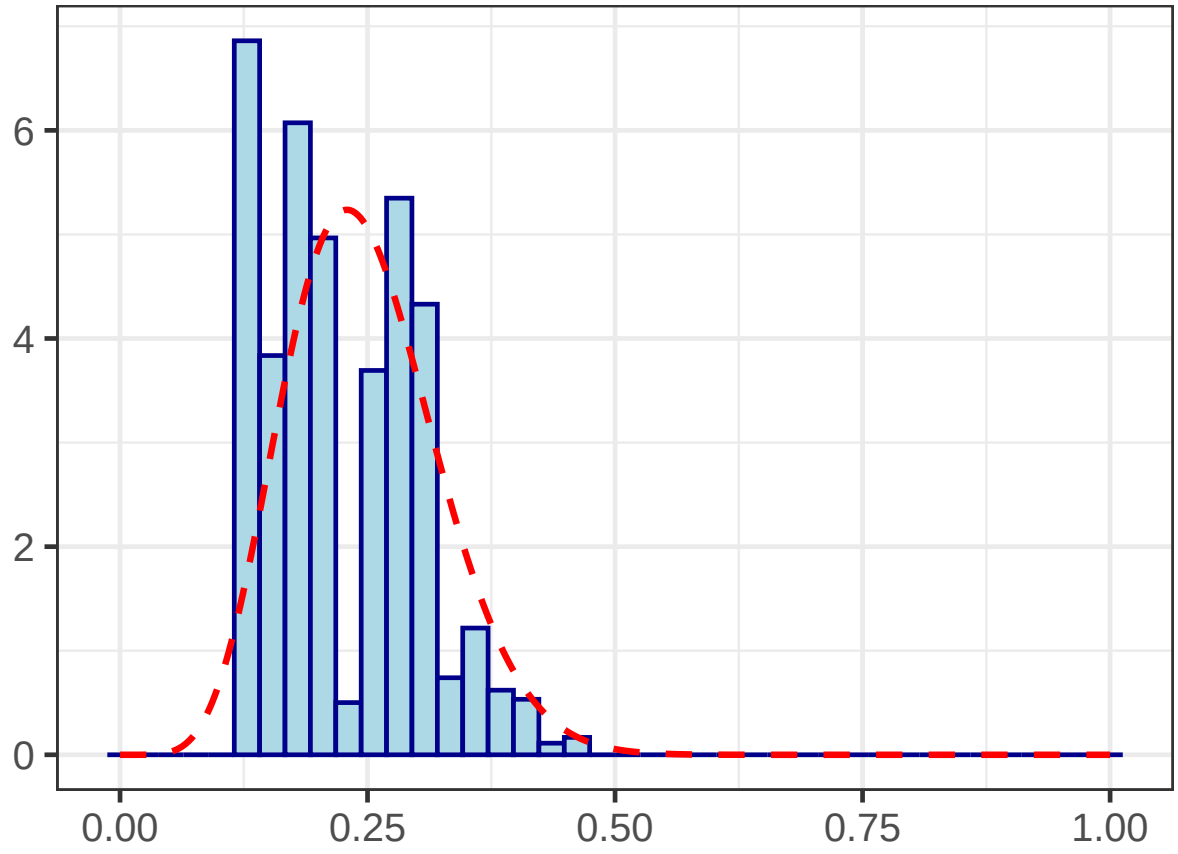
```r
data <- data.frame(x = seq_along(xt), y = xt)
g <- ggplot(data, aes(x = x, y = y))+
  geom_line()+
  labs(x = "", y = "")+
  theme_bw()
print(g)
```


0.5
0.4
0.3
0.2
0.1
0
1000
2000
3000
4000
5000

理论后验密度

```
f1 <- dnorm(x, mu[1], sigma[1])
f2 <- dnorm(x, mu[2], sigma[2])
logpost <- function(p) {
  sum(log(p * f1 + (1 - p) * f2))
}
peak <- optimize(logpost, c(0, 1), maximum=TRUE)$objective
kernel <- function(p) {
  sapply(p, function(v) exp(logpost(v) - peak))
}
normalizer <- integrate(kernel, 0, 1, rel.tol=1e-10)$value
p_grid <- seq(0, 1, length.out=1001)
posterior <- data.frame(
  p = p_grid, density = kernel(p_grid) / normalizer)
stopifnot(abs(integrate(function(p) kernel(p) / normalizer,
                        0, 1)$value - 1) < 1e-7)
```

```r
xb <- xt[101:m]
data <- data.frame(x = xb)
ggplot(data, aes(x = x))+
  geom_histogram(aes(y=after_stat(density)),
                 color="darkblue", fill="lightblue",
                 bins = 40)+
  geom_line(data = posterior,
            aes(x = p, y = density),
            color = "red", linetype = "dashed",
            linewidth = 0.7) +
  labs(x = "", y = "")+
  theme_bw()
```


6
4
2
0
0.00
0.25
0.50
0.75
1.00

2.4 Gibbs 抽样

- 一般的 MCMC 每次尝试抽样会有拒绝的概率，也即存在很多无效样本。
- Gibbs 抽样是一类常用于多元分布抽样的 Metropolis-Hasting 方法。考虑多元向量 $X = (X_1, \dots, X_d)^\top \in \mathbb{R}^d$。Gibbs 抽样每次只更新向量 X 的一个变量 X_j。
- 记 t 时刻的多元向量样本为 $x_t = (x_{t,1}, \dots, x_{t,d})^\top \in \mathbb{R}^d$，并且记 $y = (y_j, x_{t,-j}) \in \mathbb{R}^d$ 是来自建议分布 $g(\cdot \mid x_t)$ 的生产样本，这里 $x_{t,-j} = (x_{t,1}, \dots, x_{t,j-1}, x_{t,j+1}, \dots, x_{t,d})^\top \in \mathbb{R}^{d-1}$。在 Gibbs 抽样中，取建议分布为边际条件分布 $X_j \mid (X_{-j} = x_{t,-j})$，即

$$g(y \mid x_t) = f_{X_j \mid X_{-j}}(y_j \mid x_{t,-j}).$$

- 思考：如何证明 *Gibbs* 抽样对于每次尝试抽样都接受。

2.5 系统扫描 Gibbs 抽样

每一轮对每一个分量都抽样

- 初始化 x_0;
- 在第 t 步，按照如下过程更新 $x_t \to x_{t+1}$:
  1. 对于 $j = 1, \ldots, p$,
     - 从边际条件分布 $X_j \mid (X_{-j} = x_{t,-j})$ 中抽样得到分量 $x_{t+1,j}$;
     - 更新 $x_{t,j} \leftarrow x_{t+1,j}$;
  2. 得到 $x_{t+1} = (x_{t+1,1}, \cdots, x_{t+1,d})^\top$;
  3. $t \leftarrow t + 1$.

2.6 随机扫描 Gibbs 抽样

每一轮随机对某一个分量抽样

- 初始化 x_0;
- 在第 t 步，按照如下过程更新 $x_t \to x_{t+1}$：
  1. 从事先给定概率 $(\alpha_1, \dots, \alpha_d)$ 中抽取抽样下标 j;
  2. 对于抽取下标 j, 从边际条件分布 $X_j \mid (X_{-j} = x_{t,-j})$ 中抽样得到分量 $x_{t+1,j}$;
  3. 令 $x_{t+1} = (x_{t,1}, \dots, x_{t,j-1}, x_{t+1,j}, x_{t,j+1}, x_{t,d})^\top$;
  4. $t \leftarrow t + 1$.

例 2.5

利用 *Gibbs* 抽样生成二元正态分布 $N(\mu, \Sigma)$, 这里 $\mu = (\mu_1, \mu_2)$,

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}.$$

- 易知

$$f(x_1 \mid x_2) \sim N\left(\mu_1 + \rho\frac{\sigma_1}{\sigma_2}(x_2 - \mu_2), (1-\rho^2)\sigma_1^2\right)$$

$$f(x_2 \mid x_1) \sim N\left(\mu_2 + \rho\frac{\sigma_2}{\sigma_1}(x_1 - \mu_1), (1-\rho^2)\sigma_2^2\right)$$

```r
#initialize constants and parameters
m <- 5000 #length of chain
burn <- 1000 #burn-in length
# the chain, a bivariate sample
X <- matrix(0, m, 2)

rho <- -.75 #correlation
mu1 <- 0
mu2 <- 2
sigma1 <- 1
sigma2 <- .5
s1 <- sqrt(1-rho^2)*sigma1
s2 <- sqrt(1-rho^2)*sigma2
```

```
###### generate the chain #####

X[1, ] <- c(mu1, mu2) #initialize

for (i in 2:m) {
  x2 <- X[i-1, 2]
  m1 <- mu1 + rho * (x2 - mu2) * sigma1/sigma2
  X[i, 1] <- rnorm(1, m1, s1)
  x1 <- X[i, 1]
  m2 <- mu2 + rho * (x1 - mu1) * sigma2/sigma1
  X[i, 2] <- rnorm(1, m2, s2)
}

b <- burn + 1
x <- X[b:m, ]
```

```r
# compare sample statistics to parameters
colMeans(x)
```

```
## [1] -0.06791174  2.02962101
```

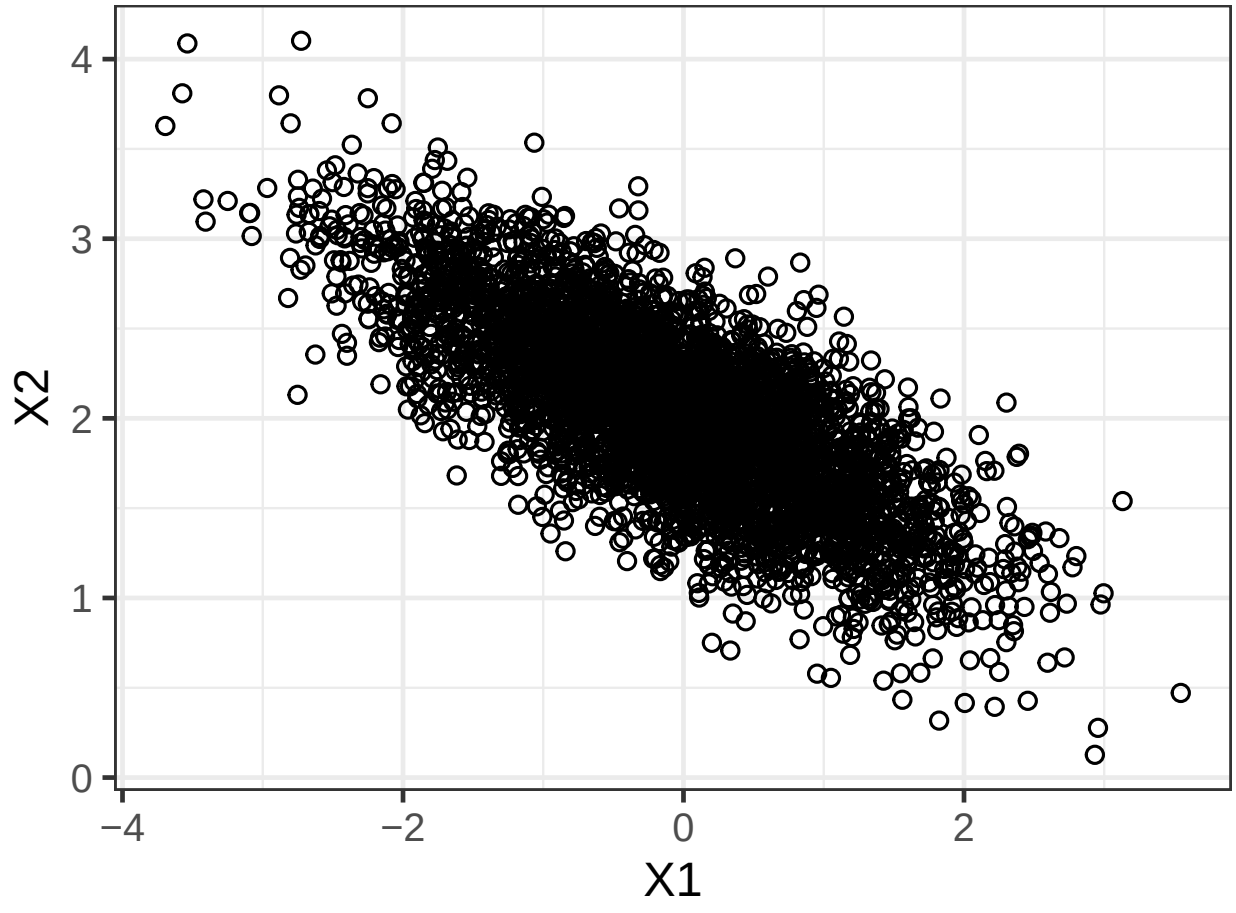
```r
cov(x)
```

```
##             [,1]       [,2]
## [1,]  1.0330993 -0.3872264
## [2,] -0.3872264  0.2597620
```

```r
cor(x)
```

```
##             [,1]       [,2]
## [1,]  1.0000000 -0.7474913
## [2,] -0.7474913  1.0000000
```

```r
data <- data.frame(x = x[,1], y = x[,2])
g <- ggplot(data, aes(x = x, y = y))+
  geom_point(shape = 1)+
  labs(x = "X1", y = "X2")+
  theme_bw()
print(g)
```


4
3
2
1
0
X2
−4
−2
0
2
X1