Simulation

Vasileios (Bill) Katsianos

September 2023

Contents

1 Random Variable Generation

Inverse Transform Method

Definition 1.1. Let F be a CDF with support S. We define the generalized inverse of F as $F^{-}:[0,1]\to S$ with $F^{-}(u)=\inf\{x\in S : F(x)\geqslant u\}$.

Note 1.1. According to the definition of the generalized inverse F^{-} and the fact that the CDF F is an increasing function, we infer that $F(x)\geqslant u \Leftrightarrow F^{-}(u)\leqslant x$.

Theorem 1.1. Let $U\sim \text{Unif}(0,1)$. Then, the random variable $X=F^{-}(U)$ has CDF F.

Proof. We know that $F_U(u)=\mathbb{P}(U\leqslant u)=u$ for $u\in[0,1]$. For $x\in S$, we calculate that:

$$F_X(x)=\mathbb{P}(X\leqslant x)=\mathbb{P}\left[F^{-}(U)\leqslant x\right]=\mathbb{P}\left[F(x)\geqslant U\right]=F_U\left(F(x)\right)=F(x).$$

□

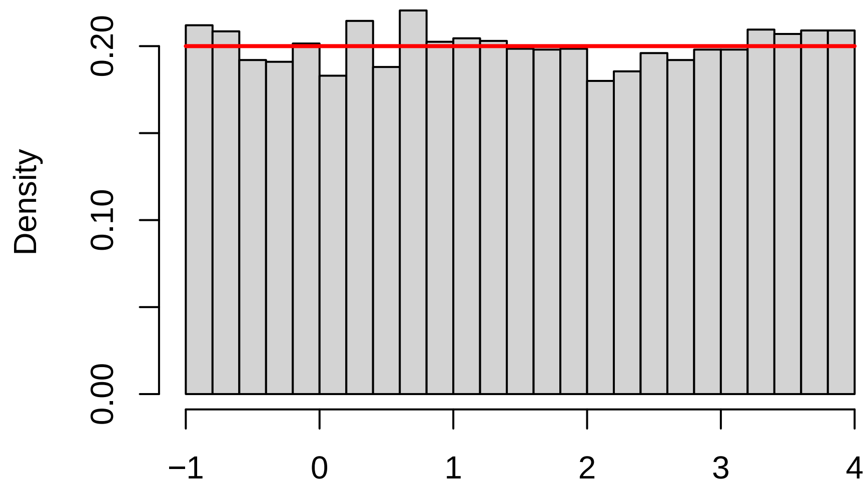
Absolutely Continuous Random Variable Genetation

Note 1.2. If the CDF F is absolutely continuous, then $F^{-}\equiv F^{-1}$.

Example 1.1. We want to generate a random sample $X_1,\ldots,X_n\sim \text{Unif}[a,b]$. For $x\in[a,b]$, we calculate that:

$$F(x)=\frac{x-a}{b-a},\quad F^{-1}(u)=(b-a)u+a.$$

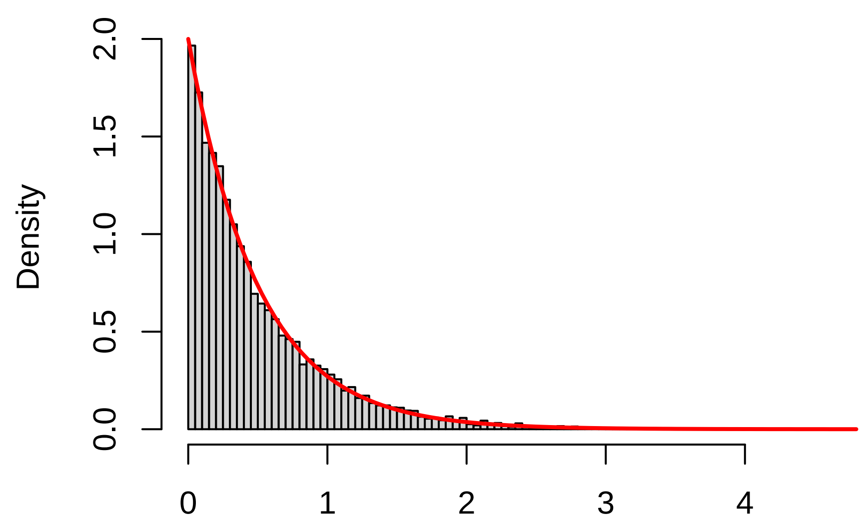
```r
n = 10000
a = -1
b = 4
U = runif(n)
X = (b - a) * U + a
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dunif(x, a, b), add = TRUE, col = "red", lwd = 2)
```



Example 1.2. We want to generate a random sample $X_1, \ldots, X_n \sim \text{Exp}(\lambda)$. For $x > 0$, we calculate that:

$$F(x) = 1 - e^{-\lambda x}, \quad F^{-1}(u) = -\frac{1}{\lambda}\log(1-u).$$

```r
n = 10000
lambda = 2
U = runif(n)
X = -log(1 - U)/lambda
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dexp(x, lambda), add = TRUE, col = "red", lwd = 2)
```



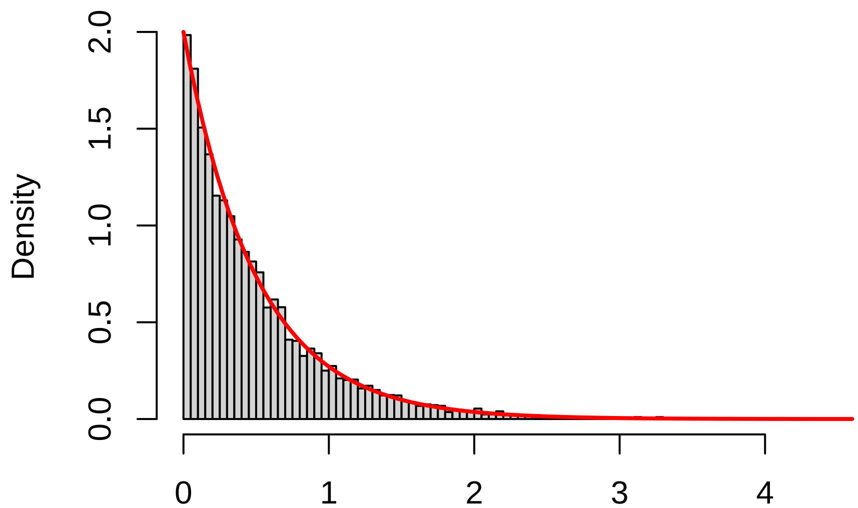
Lemma 1.1. If $U \sim \text{Unif}[0,1]$, then $V = 1 - U \sim \text{Unif}[0,1]$.

Proof. For $v \in [0,1]$, we calculate that:

$$F_V(v) = \mathbb{P}(V \leqslant v) = \mathbb{P}(1-U \leqslant v) = \mathbb{P}(U \geqslant 1-v) = 1 - \mathbb{P}(U \leqslant 1-v) = 1-(1-v) = v = F_U(v).$$

□

```
n = 10000
lambda = 2
U = runif(n)
X = -log(U)/lambda
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dexp(x, lambda), add = TRUE, col = "red", lwd = 2)
```



Lemma 1.2. If $X \sim \text{Exp}(\lambda)$ and $\mu > 0$, then the random variables $Y = (X \mid X > \mu)$ and $W = X + \mu$ are identically distributed.

Proof. First, we know that $\mathbb{P}(X > \mu) = 1 - F_X(\mu) = e^{-\lambda\mu}$. For $x > \mu$, we calculate that:

$$F_Y(x) = \mathbb{P}\left(X \leqslant x \mid X > \mu\right) = \frac{\mathbb{P}\left(X \leqslant x, X > \mu\right)}{\mathbb{P}(X > \mu)} = \frac{F_X(x) - F_X(\mu)}{F_X(\mu)} = \frac{e^{-\lambda\mu} - e^{-\lambda x}}{e^{-\lambda\mu}} = 1 - e^{-\lambda(x-\mu)},$$

$$F_W(x) = \mathbb{P}\left(W \leqslant x\right) = \mathbb{P}\left(X + \mu \leqslant x\right) = F_X(x-\mu) = 1 - e^{-\lambda(x-\mu)}.$$

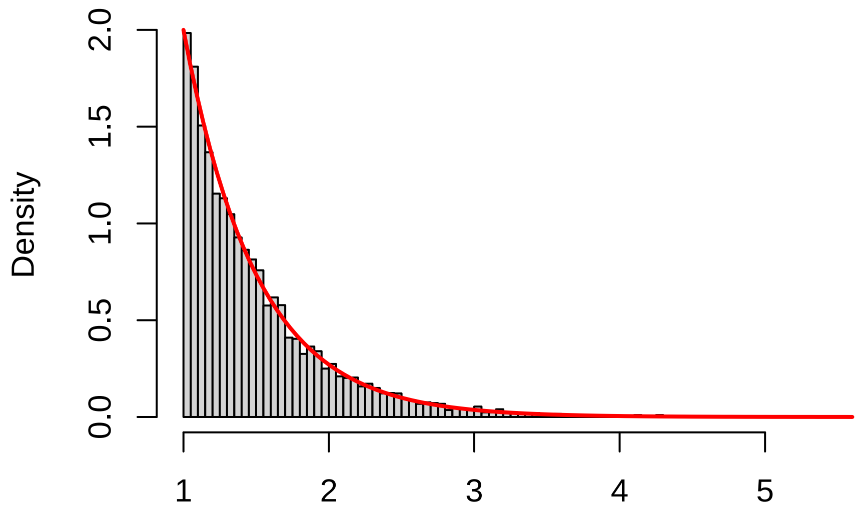
□

Note 1.3. The previous lemma is a consequence of the memoryless property of the exponential distribution.

Example 1.3. Let $X \sim \text{Exp}(\lambda)$ and $\mu > 0$. We want to generate a random sample $X_1, X_2, \ldots, X_n$ from the conditional distribution of X given that $X > \mu$.

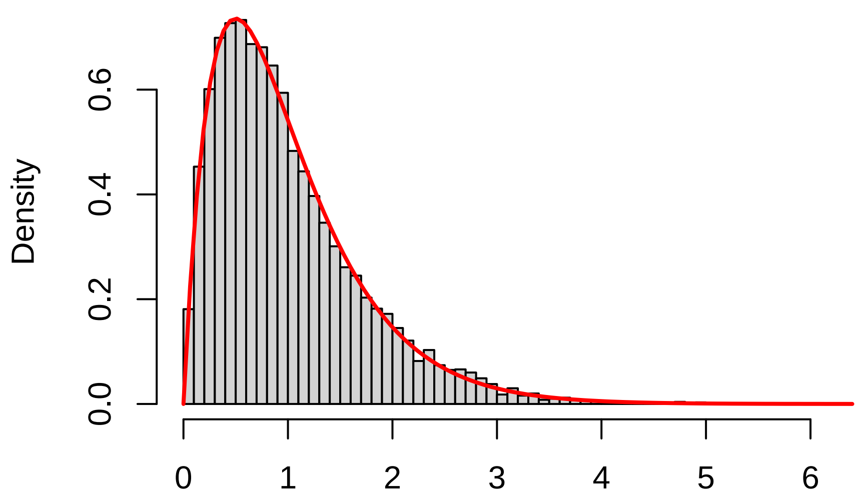
```
n = 10000
lambda = 2
mu = 1
U = runif(n)
X = mu - log(U)/lambda
```

```r
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dexp(x - mu, lambda), add = TRUE, col = "red", lwd = 2)
```



Example 1.4. We want to generate a random sample $X_1, \dots, X_n \sim \text{Gamma}(k, \lambda)$ for $k \in \mathbb{N}$. Consider independent random variables $Y_1, \dots, Y_k \sim \text{Exp}(\lambda)$. Then, we know that $Y_1 + \cdots + Y_k \sim \text{Gamma}(k, \lambda)$.

```r
n = 10000
k = 2
lambda = 2
U = matrix(runif(n * k), n)
Y = -log(U)/lambda
X = rowSums(Y)
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dgamma(x, k, lambda), add = TRUE, col = "red", lwd = 2)
```



Example 1.5. For $x \in \mathbb{R}$, we want to generate a random sample $X_1, X_2, \dots, X_n$ with the following PDF:

$$f(x) = \frac{\lambda}{2} e^{-\lambda|x-\mu|}.$$

For $x \leqslant \mu$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant x) = \int_{-\infty}^{x} f(y)dy = \int_{-\infty}^{x} \frac{\lambda}{2} e^{-\lambda(\mu-y)} dy = \frac{1}{2} e^{-\lambda(\mu-x)}.$$

For $x > \mu$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant \mu) + \mathbb{P}(\mu < X \leqslant x) = F(\mu) + \int_{\mu}^{x} \frac{\lambda}{2} e^{-\lambda(y-\mu)} dy = \frac{1}{2} - \frac{1}{2} e^{-\lambda(x-\mu)} + \frac{1}{2} = 1 - \frac{1}{2} e^{-\lambda(x-\mu)}.$$

For $u \in [0, F(\mu)] = [0, 0.5]$, we calculate that:

$$F(x) = u \Leftrightarrow x = \mu + \frac{1}{\lambda} \log(2u).$$

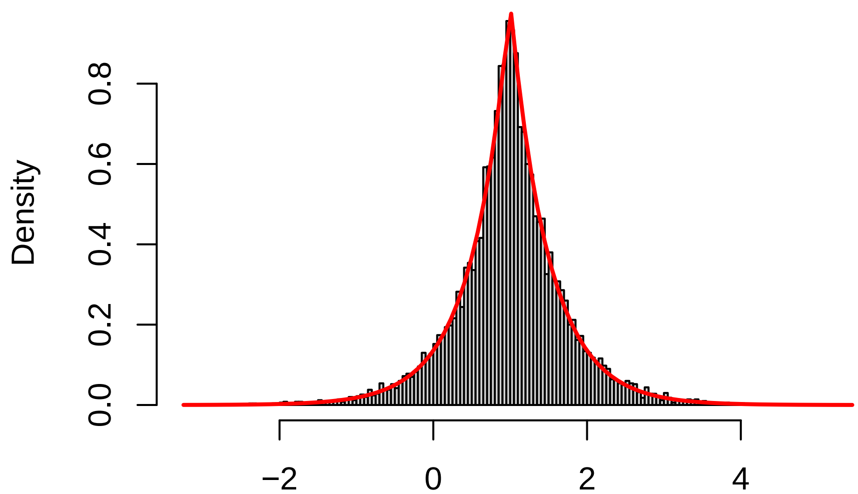
For $u \in (F(\mu), 1] = (0.5, 1]$, we calculate that:

$$F(x) = u \Leftrightarrow x = \mu - \frac{1}{\lambda} \log\left[2(1-u)\right].$$

Therefore, we infer that:

$$F^{-1}(u) = \begin{cases} \mu + \frac{1}{\lambda} \log(2u), & 0 \leqslant u \leqslant 0.5 \\ \mu - \frac{1}{\lambda} \log\left[2(1-u)\right], & 0.5 < u \leqslant 1 \end{cases}.$$

```
n = 10000
lambda = 2
mu = 1
U = runif(n)
X = ifelse(U <= 0.5, mu + log(2 * U)/lambda, mu - log(2 * (1 - U))/lambda)
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dexp(abs(x - mu), lambda)/2, add = TRUE, col = "red", lwd = 2)
```



Example 1.6. We want to generate a random sample $X_1, X_2, \ldots, X_n$ with the following PDF:

$$f(x) = \begin{cases} \frac{x-2}{2}, & 2 \leqslant x \leqslant 3 \\ \frac{6-x}{6}, & 3 < x \leqslant 6 \end{cases}.$$

For $x \in [2,3]$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant x) = \int_2^x f(y)dy = \int_2^x \frac{y-2}{2}dy = \frac{x^2}{4} - x + 1.$$

For $x \in (3,6]$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant 3) + \mathbb{P}(3 < X \leqslant x) = F(3) + \int_3^x \frac{6-y}{6}dy = \frac{1}{4} + x - \frac{x^2}{12} - \frac{9}{4} = -\frac{x^2}{12} + x - 2.$$

For $u \in [0, F(3)] = [0, 0.25]$, we calculate that:

$$F(x) = u \Leftrightarrow x^2 - 4x + 4(1-u) = 0 \Leftrightarrow x = \frac{4 \pm \sqrt{16u}}{2} = 2\left(1 \pm \sqrt{u}\right).$$

The solution $x = 2\left(1 - \sqrt{u}\right) \in [1,2]$ is rejected, so we infer that $x = 2\left(1 + \sqrt{u}\right) \in [2,3]$.

For $u \in (F(3), 1] = (0.25, 1]$, we calculate that:

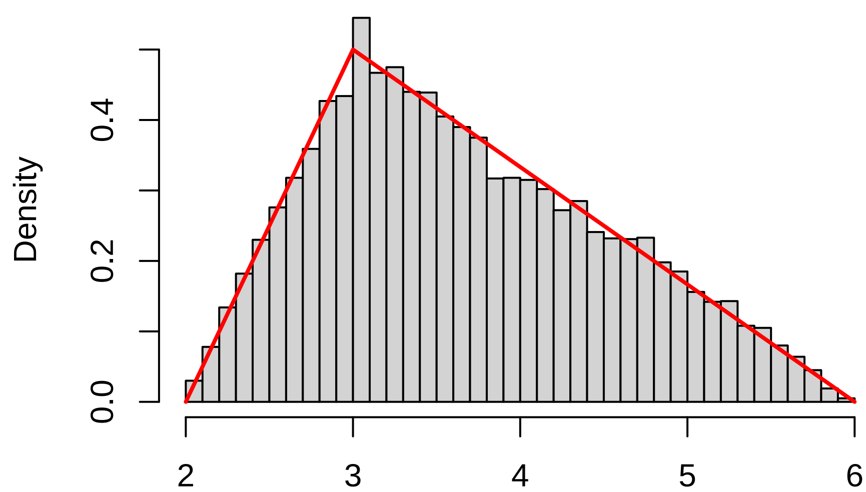
$$F(x) = u \Leftrightarrow x^2 - 12x + 12(u+2) = 0 \Leftrightarrow x = \frac{12 \pm \sqrt{48(1-u)}}{2} = 2\left[3 \pm \sqrt{3(1-u)}\right].$$

The solution $x = 2\left[3 + \sqrt{3(1-u)}\right] \in [6,9)$ is rejected, so we infer that $x = 2\left[3 - \sqrt{3(1-u)}\right] \in (3,6]$. Therefore, we conclude that:

$$F^{-1}(u) = \begin{cases} 2\left(1 + \sqrt{u}\right), & 0 \leqslant u \leqslant 0.25 \\ 2\left[3 - \sqrt{3(1-u)}\right], & 0.25 < u \leqslant 1 \end{cases}.$$

```
f = function(x) {
    ifelse(x >= 2 & x < 3, (x - 2)/2, ifelse(x >= 3 & x < 6, (6 - x)/6, 0))
}

n = 10000
U = runif(n)
X = ifelse(U <= 0.25, 2 * (1 + sqrt(U)), 2 * (3 - sqrt(3 * (1 - U))))
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(f(x), add = TRUE, col = "red", lwd = 2)
```



Example 1.7. We want to generate a random sample $X_1, X_2, \ldots, X_n$ with PDF $f(x) = 1 - |1 - x|$ for $x \in [0, 2]$. For $x \in [0, 1]$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant x) = \int_0^x f(y)dy = \int_0^x ydy = \frac{x^2}{2}.$$

For $x \in (1, 2]$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant 1) + \mathbb{P}(1 < X \leqslant x) = F(1) + \int_1^x 2 - ydy = \frac{1}{2} + 2x - \frac{x^2}{2} - \frac{3}{2} = -\frac{x^2}{2} + 2x - 1.$$

For $u \in [0, F(1)] = [0, 0.5]$, we calculate that:

$$F(x) = u \Leftrightarrow x^2 = 2u \Leftrightarrow x = \pm\sqrt{2u}.$$

The solution $x = -\sqrt{2u} \in [-1, 0]$ is rejected, so we infer that $x = \sqrt{2u} \in [0, 1]$.

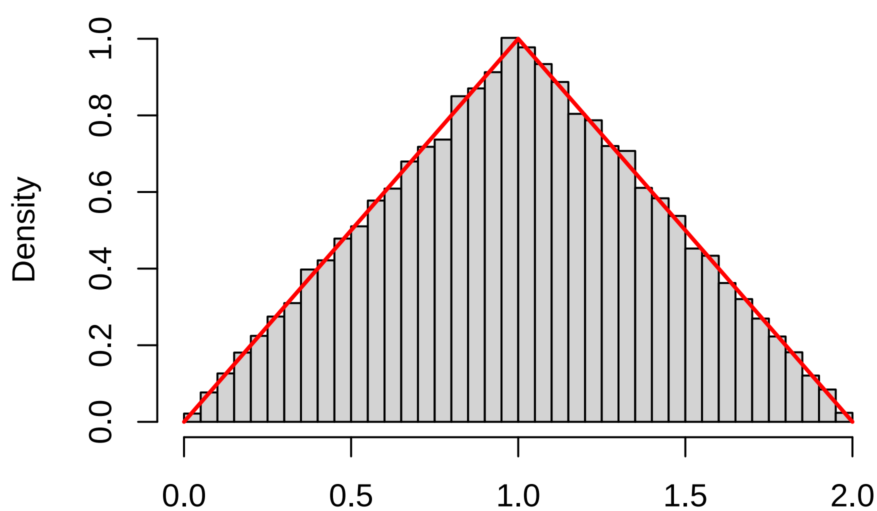
For $u \in (F(1), 1] = (0.5, 1]$, we calculate that:

$$F(x) = u \Leftrightarrow x^2 - 4x + 2(u + 1) = 0 \Leftrightarrow x = \frac{4 \pm \sqrt{8(1-u)}}{2} = 2 \pm \sqrt{2(1-u)}.$$

The solution $x = 2 + \sqrt{2(1-u)} \in [2, 3)$ is rejected, so we infer that $x = 2 - \sqrt{2(1-u)} \in (1, 2]$. Therefore, we conclude that:

$$F^{-1}(u) = \begin{cases} \sqrt{2u}, & 0 \leqslant u \leqslant 0.5 \\ 2 - \sqrt{2(1-u)}, & 0.5 < u \leqslant 1 \end{cases}.$$

```
n = 50000
U = runif(n)
X = ifelse(U <= 0.5, sqrt(2 * U), 2 - sqrt(2 * (1 - U)))
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(1 - abs(1 - x), add = TRUE, col = "red", lwd = 2)
```



Lemma 1.3. Consider the independent random variables $U, V \sim \text{Unif}[0, 1]$. Then, the random variable $X = U + V$ has PDF $f(x) = 1 - |1 - x|$ for $x \in [0, 2]$.

Proof. For $x \in [0,2]$, we calculate that:

$$F_X(x) = \mathbb{P}(U + V \leqslant x) = \int_0^1 \mathbb{P}(U \leqslant x - v) f_V(v) dv = \int_0^1 F_U(x - v) dv.$$

We observe that:

$$F_U(x - v) = \begin{cases} 1, & v \leqslant x - 1 \\ x - v, & x - 1 < v \leqslant x \\ 0, & v > x \end{cases}.$$

For $x \in [0,1]$, we infer that:

$$F_X(x) = \int_0^x x - v dv = \frac{x^2}{2}.$$

For $x \in (1,2]$, we infer that:

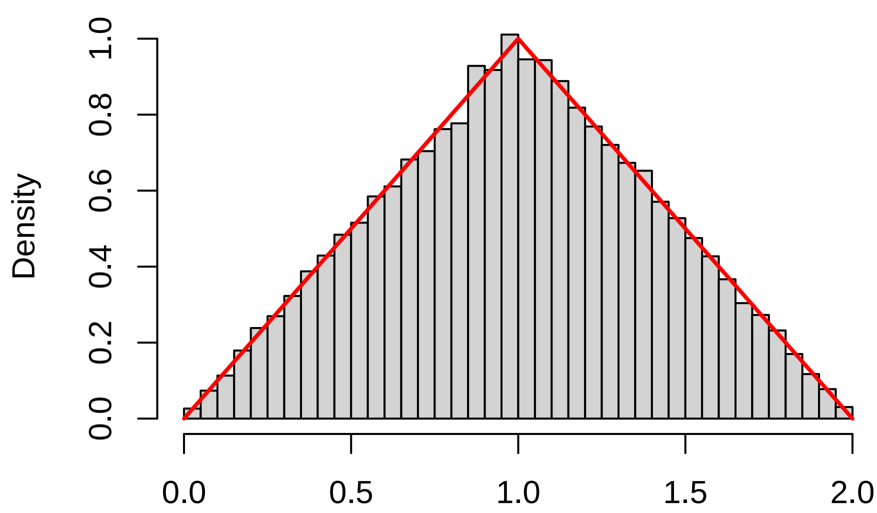
$$F_X(x) = \int_0^{x-1} 1 dv + \int_{x-1}^1 x - v dv = x - 1 + x - \frac{1}{2} - x(x-1) + \frac{(x-1)^2}{2} = -\frac{x^2}{2} + 2x - 1.$$

Therefore, we conclude that:

$$f_X(x) = \begin{cases} x, & 0 \leqslant x \leqslant 1 \\ 2 - x, & 1 < x \leqslant 2 \end{cases}.$$

We observe that $f_X(x) = 1 - |1 - x| = f(x)$ for $x \in [0,2]$. □

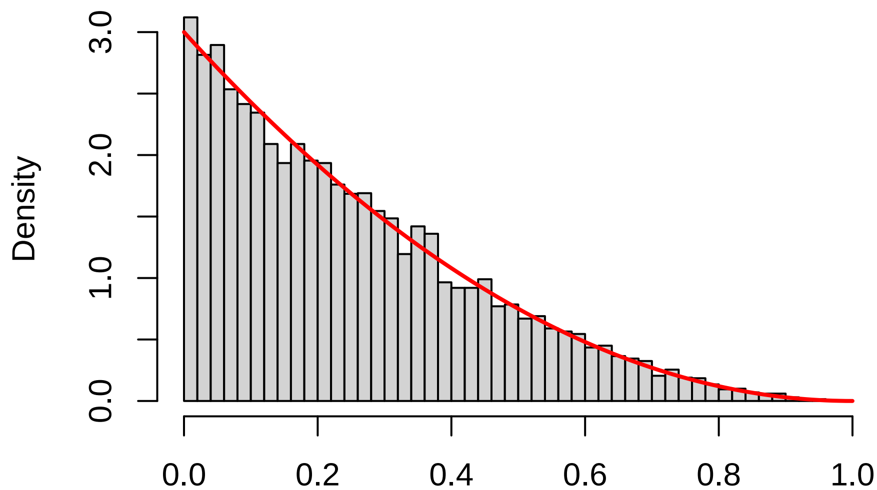
```
n = 50000
U = runif(n)
V = runif(n)
X = U + V
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(1 - abs(1 - x), add = TRUE, col = "red", lwd = 2)
```



Example 1.8. We want to generate a random sample $X_1, \ldots, X_n \sim \text{Beta}(1, k)$. For $x \in [0,1]$, we know that $f(x) = k(1-x)^{k-1}$. We calculate that:

$$F(x) = 1 - (1-x)^k, \quad F^{-1}(u) = 1 - (1-u)^{1/k}.$$

```r
n = 10000
k = 3
U = runif(n)
X = 1 - U^(1/k)
hist(X, "FD", freq = FALSE, main = NA, xlim = c(0, 1), xlab = NA)
curve(dbeta(x, 1, k), add = TRUE, col = "red", lwd = 2)
```



Lemma 1.4. Consider independent random variables Y_i with common support S and CDFs F_i for $i = 1, 2, \ldots, k$. Then,

i. The random variable $X = \max\{Y_1, \ldots, Y_k\}$ has the following CDF:

$$F(x) = \prod_{i=1}^{k} F_i(x).$$

ii. The random variable $X = \min\{Y_1, \ldots, Y_k\}$ has the following CDF:

$$F(x) = 1 - \prod_{i=1}^{k} \left[1 - F_i(x)\right].$$

Proof. i. For $x \in S$, we calculate that:

$$F(x) = \mathbb{P}\left(\max\{Y_1, \ldots, Y_k\} \leqslant x\right) = \mathbb{P}\left(Y_1 \leqslant x, \ldots, Y_k \leqslant x\right) = \prod_{i=1}^{k} \mathbb{P}(Y_i \leqslant x) = \prod_{i=1}^{k} F_i(x).$$

ii. For $x \in S$, we calculate that:

$$\begin{aligned} F(x) &= \mathbb{P}\left(\min\{Y_1, \ldots, Y_k\} \leqslant x\right) = 1 - \mathbb{P}\left(\min\{Y_1, \ldots, Y_k\} > x\right) \\ &= 1 - \mathbb{P}\left(Y_1 > x, \ldots, Y_k > x\right) = 1 - \prod_{i=1}^{k} \mathbb{P}(Y_i > x) = 1 - \prod_{i=1}^{k} \left[1 - F_i(x)\right]. \end{aligned}$$

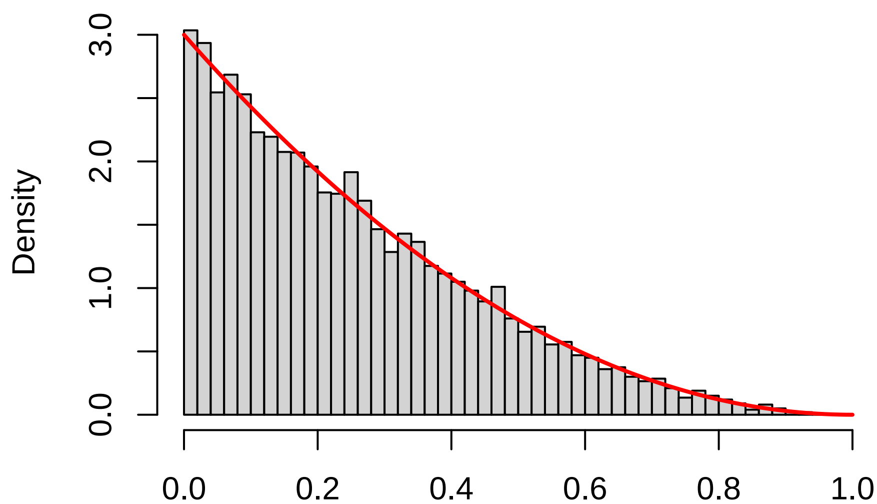
$\square$

For $x \in [0,1]$, we observe that:

$$F(x) = 1 - \prod_{i=1}^{k}(1-x).$$

Consider the independent random variables $Y_1, \ldots, Y_k \sim \text{Unif}[0,1]$ with CDF $F_Y(x) = x$ for $x \in [0,1]$. According to the previous lemma, we infer that the random variable $\min\{Y_1, \ldots, Y_k\}$ has CDF F.

```
n = 10000
k = 3
U = matrix(runif(n * k), n)
X = apply(U, 1, min)
hist(X, "FD", freq = FALSE, main = NA, xlim = c(0, 1), xlab = NA)
curve(dbeta(x, 1, k), add = TRUE, col = "red", lwd = 2)
```



Rejection Method

Consider a PDF f with bounded support $S = [0,1]$ and independent random variables $Y \sim \text{Unif}[0,1]$, $U \sim \text{Unif}[0,1]$. We define $M = \max_{x\in[0,1]} f(x)$ and $V = MU \sim \text{Unif}[0,M]$.

Note 1.4. Since f is a PDF with support $[0,1]$, it must hold that $M > 1$.

Proposition 1.1. i. The random vector (Y,V) follows the bivariate uniform distribution on the rectangle with base $[0,1]$ and height $[0,M]$.

ii. The conditional distribution of the random vector (Y,V) given that $f(Y) \geqslant V$ is the bivariate uniform distribution in the area under the curve of f, i.e. on the set $\{(y,v) \in [0,1] \times [0,M] : f(y) \geqslant v\}$.

iii. The marginal distribution of Y given that $f(Y) \geqslant V$ has PDF f.

Proof. i. For $y \in [0,1]$ and $v \in [0,M]$, we calculate that:

$$F_{Y,V}(y,v) = \mathbb{P}\left(Y \leqslant y, V \leqslant v\right) = \mathbb{P}(Y \leqslant y)\mathbb{P}(V \leqslant v) = y \cdot \frac{v}{M},$$

$$f_{Y,V}(y,v) = \frac{\partial^2 F_{Y,V}(y,v)}{\partial v \partial y} = 1 \cdot \frac{1}{M} = \frac{1}{\int_0^1 \int_0^M 1 dv dy}.$$

ii. For $y \in [0,1]$ and $v \in [0,M]$ with $f(y) \geqslant v$, we calculate that:

$$\mathbb{P}\left[f(Y) \geqslant V\right] = \int_0^1 f_Y(y)\mathbb{P}\left[V \leqslant f(y)\right] dy = \int_0^1 \frac{f(y)}{M} dy = \frac{1}{M}\int_0^1 f(y) dy = \frac{1}{M},$$

$$\mathbb{P}\left[Y \leqslant y, V \leqslant v, f(Y) \geqslant V\right] = \int_0^y f_Y(x)\mathbb{P}\left[V \leqslant v, V \leqslant f(x)\right] dx = \frac{1}{M}\int_0^y \min\{v, f(x)\} dx,$$

$$F_{Y,V|F(Y)\geqslant V}(y,v) = \mathbb{P}\left[Y \leqslant y, V \leqslant v \mid f(Y) \geqslant V\right] = \frac{\mathbb{P}\left[Y \leqslant y, V \leqslant v, f(Y) \geqslant V\right]}{\mathbb{P}\left[f(Y) \geqslant V\right]} = \int_0^y \min\{v, f(x)\} dx,$$

$$f_{Y,V|f(Y)\geqslant V}(y,v) = \frac{\partial^2 F_{Y,V|F(Y)\geqslant V}(y,v)}{\partial v \partial y} = \frac{\partial \min\{v, f(y)\}}{\partial v} = \frac{\partial v}{\partial v} = 1 = \frac{1}{\int_S \int_0^{f(y)} 1 dv dy}.$$

iii. For $y \in [0,1]$, we calculate that:

$$f_{Y|f(Y)\geqslant V}(y) = \int_0^{f(y)} f_{Y,V|f(Y)\geqslant V}(y,v) dv = \int_0^{f(y)} 1 dv = f(y).$$

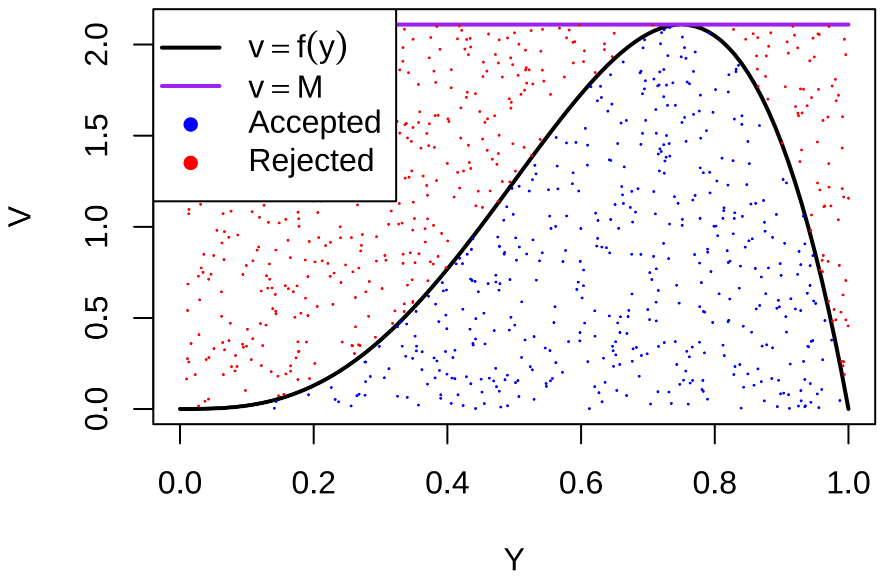
$\square$

Example 1.9. We want to generate a random sample $X_1, \ldots, X_n \sim \text{Beta}(4,2)$. For $x \in [0,1]$, we know that $f(x) = 20x^3(1-x)$. For $x \in (0,1)$, we calculate that $f'(x) = 20x^2(3-4x)$. Therefore, we infer that $f'(x) = 0 \Leftrightarrow x = 3/4$, which implies that $M = f(3/4) = 135/64$.

```
n = 1000
a = 4
b = 2
M = 135/64
print(M)
```

```
## [1] 2.109375
```

```
Y = runif(n)
U = runif(n)
V = M * U
I = which(dbeta(Y, a, b) >= V)
J = which(dbeta(Y, a, b) < V)
curve(dbeta(x, a, b), lwd = 2, xlab = "Y", ylab = "V")
curve(M * dunif(x), add = TRUE, col = "purple", lwd = 2)
points(Y[I], V[I], col = "blue", pch = 16, cex = 0.2)
points(Y[J], V[J], col = "red", pch = 16, cex = 0.2)
legend("topleft", c(expression(v == f(y)), expression(v == M), "Accepted", "Rejected"),
    col = c("black", "purple", "blue", "red"), lty = c(1, 1, NA, NA), lwd = c(2,
        2, NA, NA), pch = c(NA, NA, 16, 16), bg = "white")
```



Algorithm 1.1 Rejection Method for Bounded Support $S = [0, 1]$

Input: PDF f and sample size n.

1: We calculate $M = \max_{x \in [0,1]} f(x)$.

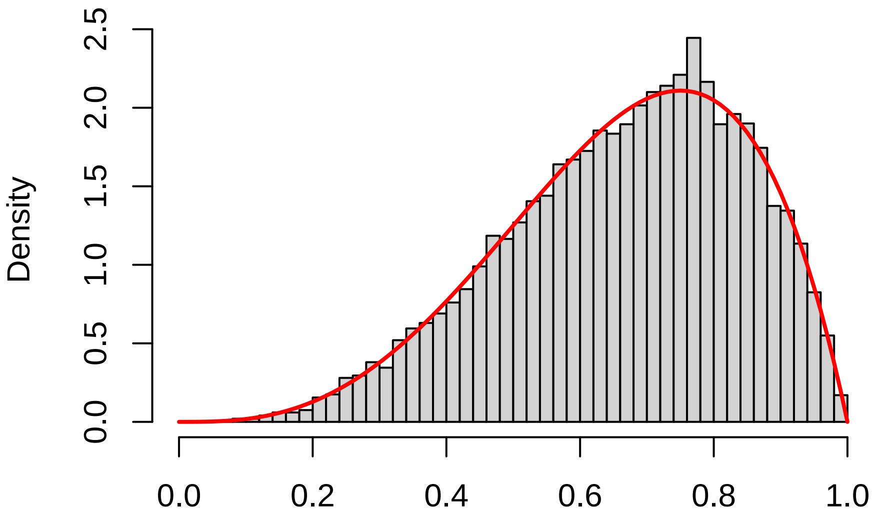
2: For $i = 1, 2, \dots, n$, we iterate the following steps:

i: We generate $Y \sim \text{Unif}[0, 1]$, $U \sim \text{Unif}[0, 1]$ and let $V = MU \sim \text{Unif}[0, 1]$.

ii: If $f(Y) \geqslant V$, we let $X_i = Y$. Otherwise, we return to step 2.

Output: Random sample $X_1, X_2, \dots, X_n$ following the PDF f.

```
n = 10000
a = 4
b = 2
M = 135/64
X = numeric(n)
for (i in 1:n) {
    Y = runif(1)
    U = runif(1)
    V = M * U
    while (dbeta(Y, a, b) < V) {
        Y = runif(1)
        U = runif(1)
        V = M * U
    }
    X[i] = Y
}
hist(X, "FD", freq = FALSE, main = NA, xlim = c(0, 1), xlab = NA)
curve(dbeta(x, a, b), add = TRUE, col = "red", lwd = 2)
```



Now, let f be a PDF with general support S and g be a proposal PDF with support $S_g \supseteq S$. Consider the independent random variables $Y \sim g$ and $U \sim \text{Unif}[0,1]$. We define $M = \max_{x\in S} \frac{f(x)}{g(x)}$ and $V = Mg(Y)U$.

Note 1.5. Since f and g are PDFs, it holds that $M > 1$.

Proposition 1.2. i. The random vector (Y, V) follows the bivariate uniform distribution in the area under the curve of Mg, i.e. on the set $\{(y, v) \in S_g \times [0, \infty] : Mg(y) \geqslant v\}$.

ii. The conditional distribution of the random vector (Y, V) given that $f(Y) \geqslant V$ is the bivariate uniform distribution in the are under the curve of f, i.e. on the set $\{(y, v) \in S \times [0, \infty] : f(y) \geqslant v\}$.

iii. The marginal distribution of Y given that $f(Y) \geqslant V$ has PDF f.

Proof. i. For $y \in S_g$ and $v \in [0, \infty]$, we calculate that:

$$F_{Y,V}(y,v) = \mathbb{P}[Y \leqslant y, V \leqslant v] = \int_{-\infty}^{y} g(x)\mathbb{P}[Mg(x)U \leqslant v]\,dx = \int_{-\infty}^{y} \cancel{g(x)} \cdot \frac{v}{M\cancel{g(x)}}dx = \frac{v}{M}\int_{-\infty}^{y} 1dx,$$

$$f_{Y,V}(y,v) = \frac{\partial^2 F_{Y,V}(y,v)}{\partial v \partial y} = \frac{\partial}{\partial v}\frac{v}{M} = \frac{1}{M} = \frac{1}{\int_{S_g}\int_0^{Mg(y)} 1dvdy}.$$

ii. For $y \in S$ and $v \in [0, \infty]$ with $f(y) \geqslant v$, we calculate that:

$$\mathbb{P}[f(Y) \geqslant V] = \int_S g(y)\mathbb{P}[Mg(y)U \leqslant f(y)]\,dy = \int_S \cancel{g(y)} \cdot \frac{f(y)}{M\cancel{g(y)}}dy = \frac{1}{M}\int_S f(y)dy = \frac{1}{M},$$

$$\mathbb{P}[Y \leqslant y, V \leqslant v, f(Y) \geqslant V] = \int_{-\infty}^{y} g(x)\mathbb{P}[Mg(x)U \leqslant v, Mg(x)U \leqslant f(x)]\,dx = \frac{1}{M}\int_{-\infty}^{y} \min\{v, f(x)\}dx,$$

$$F_{Y,V|F(Y)\geqslant V}(y,v) = \mathbb{P}[Y \leqslant y, V \leqslant v \mid f(Y) \geqslant V] = \frac{\mathbb{P}[Y \leqslant y, V \leqslant v, f(Y) \geqslant V]}{\mathbb{P}[f(Y) \geqslant V]} = \int_{-\infty}^{y} \min\{v, f(x)\}dx,$$

$$f_{Y,V|f(Y)\geqslant V}(y,v) = \frac{\partial^2 F_{Y,V|F(Y)\geqslant V}(y,v)}{\partial v \partial y} = \frac{\partial \min\{v, f(y)\}}{\partial v} = \frac{\partial v}{\partial v} = 1 = \frac{1}{\int_S \int_0^{f(y)} 1dvdy}.$$

iii. For $y \in S$, we calculate that:

$$f_{Y|f(Y)\geqslant V}(y) = \int_0^{f(y)} f_{Y,V|f(Y)\geqslant V}(y,v)dv = \int_0^{f(y)} 1dv = f(y).$$

□

Example 1.10. We want to generate a random sample $X_1, \dots, X_n \sim \text{Gamma}(2, 0.5) \equiv \chi^2_4$. If we consider a random variable $X \sim \text{Gamma}(2, 0.5)$, then we observe that $\mathbb{E}(X) = \frac{2}{0.5} = 4$. Let $Y \sim \text{Exp}(1/4)$ be a random variable with proposal PDF $g(x) = \frac{1}{4}e^{-x/4}$ for $x > 0$. We observe that $\mathbb{E}(Y) = \frac{1}{1/4} = 4$. We define:

$$h(x) = \frac{f(x)}{g(x)} = xe^{-x/4}$$

We calculate that:

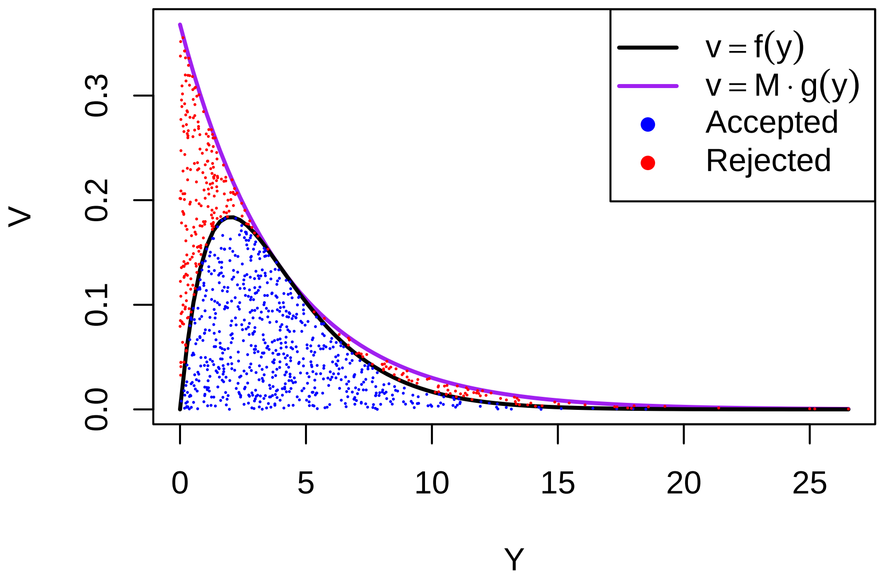
$$h'(x) = \left(1 - \frac{x}{4}\right) e^{-x/4}.$$

Therefore, we infer that $h'(x) = 0 \Leftrightarrow x = 4$, which implies that $M = h(4) = 4e^{-1}$.

```r
n = 1000
a = 2
lambda = 0.5
M = 4 * exp(-1)
print(M)
```

```
## [1] 1.471518
```

```r
W = runif(n)
Y = -log(W) * a/lambda
U = runif(n)
V = M * dexp(Y, lambda/a) * U
I = which(dgamma(Y, a, lambda) >= V)
J = which(dgamma(Y, a, lambda) < V)
curve(M * dexp(x, lambda/a), xlim = c(0, max(Y)), col = "purple", lwd = 2, xlab = "Y",
    ylab = "V")
curve(dgamma(x, a, lambda), add = TRUE, lwd = 2)
points(Y[I], V[I], col = "blue", pch = 16, cex = 0.2)
points(Y[J], V[J], col = "red", pch = 16, cex = 0.2)
legend("topright", c(expression(v == f(y)), expression(v == M %.% g(y)), "Accepted",
    "Rejected"), col = c("black", "purple", "blue", "red"), lty = c(1, 1, NA,
    NA), lwd = c(2, 2, NA, NA), pch = c(NA, NA, 16, 16))
```



Algorithm 1.2 Rejection Sampling for General Support

Input: PDF f, proposal PDF g and sample size n.

1: We calculate that $M = \max_{x \in \mathbb{R}} \frac{f(x)}{g(x)}$.

2: For $i = 1, 2, \ldots, n$, we iterate the following steps:

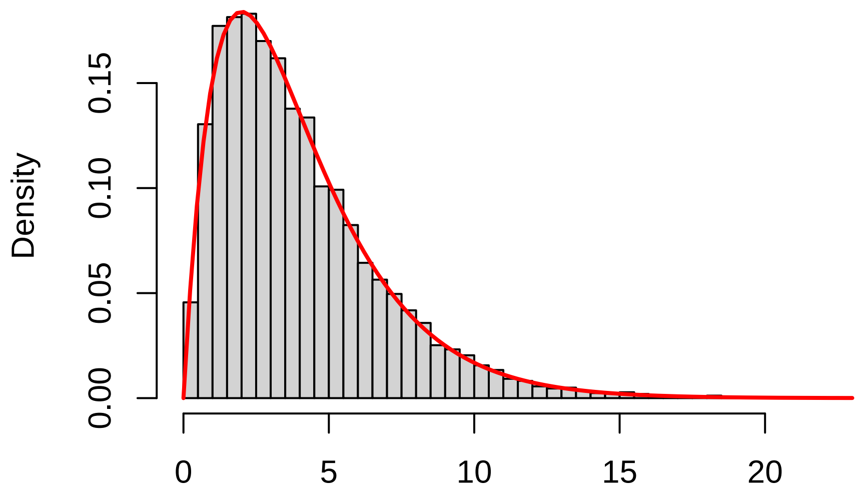
i: We generate $Y \sim g$, $U \sim \text{Unif}[0, 1]$ and let $V = Mg(Y)U$.

ii: If $f(Y) \geqslant V$, we let $X_i = Y$. Otherwise, we return to step 2.

Output: Random sample $X_1, X_2, \ldots, X_n$ following the PDF f.

```r
n = 10000
a = 2
lambda = 0.5
M = 4 * exp(-1)
X = numeric(n)
for (i in 1:n) {
    W = runif(1)
    Y = -log(W) * a/lambda
    U = runif(1)
    V = M * dexp(Y, lambda/a) * U
    while (dgamma(Y, a, lambda) < V) {
        W = runif(1)
        Y = -log(W) * a/lambda
        U = runif(1)
        V = M * dexp(Y, lambda/a) * U
    }
    X[i] = Y
}
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
```

```r
curve(dgamma(x, a, lambda), add = TRUE, col = "red", lwd = 2)
```



Theorem 1.2. i. The acceptance probability of X_i given that $Y = y$ is equal to $\frac{f(y)}{Mg(y)}$.

ii. The generation of X_i requires a finite number of iterations with probability 1. The expected number of iterations until the generation of X_i is equal to M.

Proof. i. For $y \in S_g$, we calculate that:

$$\mathbb{P}\left[f(Y) \geqslant V \mid Y = y\right] = \mathbb{P}\left[Mg(y)U \leqslant f(y)\right] = \frac{f(y)}{Mg(y)}.$$

ii. We calculated that the acceptance probability of X_i, i.e. $\mathbb{P}\left[f(Y) \geqslant V\right]$, is equal to $\frac{1}{M}$. Since every attempt at generating X_i is independent of the previous ones and each of them succeeds with probability $\frac{1}{M}$, we infer that the number of iterations until the generation of X_i follows the geometric distribution with success probability $\frac{1}{M}$. Therefore, the expected number of iterations until the generation of X_i is given by the expectation of this geometric distribution, which is equal to M. □

Note 1.6. The rejection method is more efficient when M is close to 1. In this case, only a small number of iterations is required for the generation of X_i.

Example 1.11. More generally, we want to generate a random sample $X_1, \ldots, X_n \sim \text{Gamma}\,(a, \lambda)$. Consider a random variable $Y \sim \text{Exp}(\mu)$ with proposal PDF $g_\mu(x) = \mu e^{-\mu x}$ for $x > 0$. We define:

$$h_\mu(x) = \frac{f(x)}{g_\mu(x)} = \frac{\frac{\lambda^a}{\Gamma(a)} x^{a-1} e^{-\lambda x}}{\mu e^{-\mu x}} = \frac{\lambda^a}{\mu \Gamma(a)} x^{a-1} e^{-(\lambda-\mu)x}.$$

For $a > 1$, we calculate that:

$$\frac{\partial h_\mu(x)}{\partial x} = \frac{\lambda^a}{\Gamma(a)} x^{a-2} e^{-(\lambda-\mu)x}\left[a - 1 - (\lambda - \mu)x\right],$$

$$\frac{\partial h_\mu(x)}{\partial x} = 0 \Leftrightarrow x = \frac{a-1}{\lambda - \mu},$$

$$M(\mu) = \max_{x \in \mathbb{R}} h_\mu(x) = h_\mu\left(\frac{a-1}{\lambda-\mu}\right) = \frac{\lambda^a}{\mu\Gamma(a)}\left(\frac{a-1}{\lambda-\mu}\right)^{a-1} e^{-(a-1)},$$

$$\frac{\partial M(\mu)}{\partial \mu} = \frac{\lambda^a}{\mu\Gamma(a)}\left(\frac{a-1}{\lambda-\mu}\right)^{a-1}\left(-\frac{1}{\mu}+\frac{a-1}{\lambda-\mu}\right)e^{-(a-1)},$$

$$\frac{\partial M(\mu)}{\partial \mu} = 0 \Leftrightarrow \mu = \frac{\lambda}{a},$$

$$M^* = \min_{\mu>0} M(\mu) = M\left(\frac{\lambda}{a}\right) = \frac{a^a}{\Gamma(a)}e^{-(a-1)}.$$

Therefore, the value of μ which minimizes the expected number of required iterations for the generation of a random variable from the Gamma (a,λ) distribution is equal to $\frac{\lambda}{a}$ and the minimum expected number of required iterations is equal to $\frac{a^a}{\Gamma(a)}e^{-(a-1)}$.

Note 1.7. We know that $\chi^2(\nu) \equiv \text{Gamma}\left(\frac{\nu}{2},\frac{1}{2}\right)$. Therefore, the value of μ which minimizes the expected number of required iterations for the generation of a random variable from the $\chi^2(\nu)$ distribution for $\nu > 2$ with proposal PDF $g(x) = \mu e^{-\mu x}$ for $x > 0$ is equal to $\frac{1}{\nu}$ and the minimum expected number of required iterations is equal to $\left(\frac{\nu}{2}\right)^{\nu/2}\frac{1}{\Gamma(\nu/2)}e^{-(\nu-2)/2}$.

Example 1.12. We want to generate a random sample $X_1,\ldots,X_n \sim \mathcal{N}\left(\mu,\sigma^2\right)$. For $x \in \mathbb{R}$, we consider the proposal PDF:

$$g_\lambda(x) = \frac{\lambda}{2}e^{-\lambda|x-\mu|}.$$

We define:

$$h_\lambda(x) = \frac{f(x)}{g_\lambda(x)} = \frac{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}}{\frac{\lambda}{2}e^{-\lambda|x-\mu|}} = \sqrt{\frac{2}{\pi\sigma^2}}\frac{1}{\lambda}\exp\left\{\lambda|x-\mu| - \frac{1}{2\sigma^2}(x-\mu)^2\right\}.$$

Since the function h is symmetric around $x = \mu$, we study its behavior for $x \geqslant \mu$. We calculate that:

$$\frac{\partial h_\lambda(x)}{\partial x} = \sqrt{\frac{2}{\pi\sigma^2}}\frac{1}{\lambda}\left(\lambda - \frac{x-\mu}{\sigma^2}\right)\exp\left\{\lambda(x-\mu) - \frac{1}{2\sigma^2}(x-\mu)^2\right\},$$

$$\frac{\partial h_\lambda(x)}{\partial x} = 0 \Leftrightarrow x = \mu + \sigma^2\lambda,$$

$$M(\lambda) = \max_{x\in\mathbb{R}} h_\lambda(x) = h_\lambda\left(\mu+\sigma^2\lambda\right) = \sqrt{\frac{2}{\pi\sigma^2}}\frac{1}{\lambda}e^{\sigma^2\lambda^2/2},$$

$$\frac{\partial M(\lambda)}{\partial \lambda} = \sqrt{\frac{2}{\pi\sigma^2}}\left(\sigma^2 - \frac{1}{\lambda^2}\right)e^{\sigma^2\lambda^2/2},$$

$$\frac{\partial M(\lambda)}{\partial \lambda} = 0 \Leftrightarrow \lambda = \frac{1}{\sigma},$$

$$M^* = \min_{\lambda>0} M(\lambda) = M\left(\frac{1}{\sigma}\right) = \sqrt{\frac{2e}{\pi}}.$$

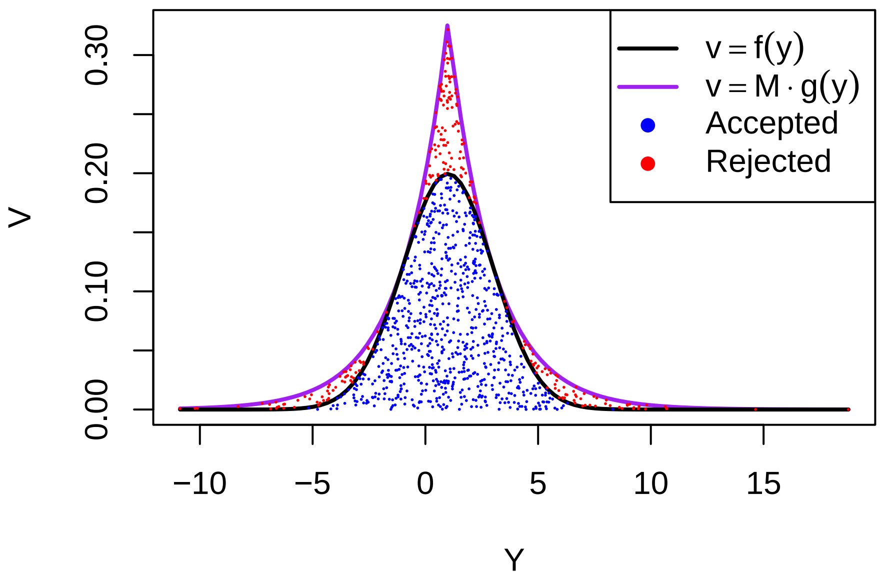
Therefore, the value of λ which minimizes the expected number of required iterations for the generation of a random variable from the $\mathcal{N}\left(\mu,\sigma^2\right)$ distribution is equal to $\frac{1}{\sigma}$ and the minimum expected number of required iterations is equal to $\sqrt{2e/\pi}$.

```
n = 1000
mu = 1
sigma = 2
lambda = 1/sigma
```

```r
M = sqrt(2 * exp(1)/pi)
print(M)
```

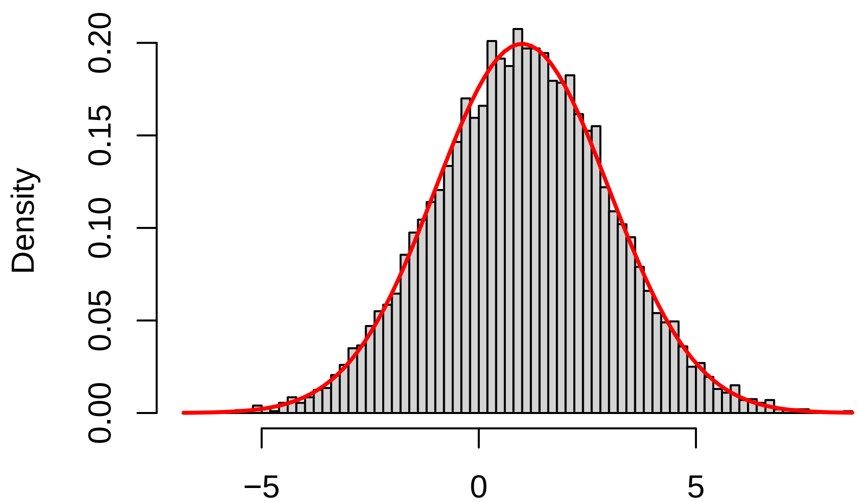
```
## [1] 1.315489
```

```r
W = runif(n)
Y = ifelse(W <= 0.5, mu + log(2 * W)/lambda, mu - log(2 * (1 - W))/lambda)
U = runif(n)
V = M * dexp(abs(Y - mu), lambda)/2 * U
I = which(dnorm(Y, mu, sigma) >= V)
J = which(dnorm(Y, mu, sigma) < V)
curve(M * dexp(abs(x - mu), lambda)/2, xlim = range(Y), col = "purple", lwd = 2,
    xlab = "Y", ylab = "V")
curve(dnorm(x, mu, sigma), add = TRUE, lwd = 2)
points(Y[I], V[I], col = "blue", pch = 16, cex = 0.2)
points(Y[J], V[J], col = "red", pch = 16, cex = 0.2)
legend("topright", c(expression(v == f(y)), expression(v == M %.% g(y)), "Accepted",
    "Rejected"), col = c("black", "purple", "blue", "red"), lty = c(1, 1, NA,
    NA), lwd = c(2, 2, NA, NA), pch = c(NA, NA, 16, 16))
```




```r
n = 10000
mu = 1
sigma = 2
lambda = 1/sigma
M = sqrt(2 * exp(1)/pi)
X = numeric(n)
for (i in 1:n) {
    W = runif(1)
    Y = ifelse(W <= 0.5, mu + log(2 * W)/lambda, mu - log(2 * (1 - W))/lambda)
    U = runif(1)
```

```
    V = M * dexp(abs(Y - mu), lambda)/2 * U
    while (dnorm(Y, mu, sigma) < V) {
        W = runif(1)
        Y = ifelse(W <= 0.5, mu + log(2 * W)/lambda, mu - log(2 * (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(Y - mu), lambda)/2 * U
    }
    X[i] = Y
}
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dnorm(x, mu, sigma), add = TRUE, col = "red", lwd = 2)
```



Let X be a random variable with absolutely continuous CDF G, PDF g, support S and $a, b \in S$ with $a < b$. We want to generate a random sample $X_1, X_2, \dots, X_n$ from the conditional distribution of X given that $a \leqslant X \leqslant b$. We know that $\mathbb{P}(a \leqslant X \leqslant b) = G(b) - G(a)$. For $x \in [a, b]$, we calculate that:

$$F_{X|a\leqslant X\leqslant b}(x) = \mathbb{P}\left(X \leqslant x \mid a \leqslant X \leqslant b\right) = \frac{\mathbb{P}\left(X \leqslant x, a \leqslant X \leqslant b\right)}{\mathbb{P}(a \leqslant X \leqslant b)} = \frac{\mathbb{P}(a \leqslant X \leqslant x)}{\mathbb{P}(a \leqslant X \leqslant b)} = \frac{G(x) - G(a)}{G(b) - G(a)},$$

$$f_{X|a\leqslant X\leqslant b}(x) = \frac{\partial F_{X|a\leqslant X\leqslant b}(x)}{\partial x} = \frac{g(x)}{G(b) - G(a)}.$$

We observe that:

$$\frac{f_{X|a\leqslant X\leqslant b}(x)}{g(x)} = \begin{cases} \frac{1}{G(b)-G(a)}, & x \in [a, b] \\ 0, & x \notin [a, b] \end{cases}, \quad M = \max_{x\in S} \frac{f_{X|a\leqslant X\leqslant b}(x)}{g(x)} = \frac{1}{G(b) - G(a)}.$$

If $Y \sim g$ and $U \sim \text{Unif}[0, 1]$, then it follows that:

$$\mathbb{P}\left[f_{X|a\leqslant X\leqslant b}(Y) \geqslant Mg(Y)U \mid Y\right] = \frac{f_{X|a\leqslant X\leqslant b}(Y)}{Mg(Y)} = \begin{cases} 1, & Y \in [a, b] \\ 0, & Y \notin [a, b] \end{cases}.$$

In other words, if the generated value Y from the PDF g lies on the given interval $[a, b]$, then it's accepted with probability 1. Otherwise, it's rejected with probability 1. Therefore, the generation of the random variable U is redundant.

Example 1.13. Let $X \sim \text{Gamma}(3, 0.5)$, $a = 2$ and $b = 10$. We want to generate a random sample $X_1, X_2, \ldots, X_n$ from the conditional distribution of X given that $a \leqslant X \leqslant b$.

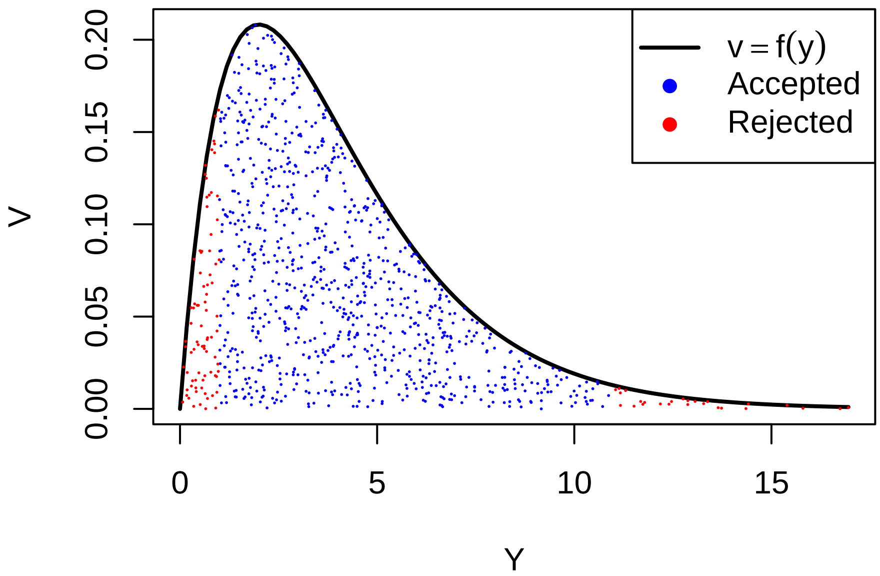
```
n = 1000
k = 2
lambda = 0.5
a = 1
b = 11
P = pgamma(b, k, lambda) - pgamma(a, k, lambda)
print(P)
```

```
## [1] 0.883232
```

```
M = 1/P
print(M)
```

```
## [1] 1.132205
```

```
W = matrix(runif(n * k), n)
R = -log(W)/lambda
Y = rowSums(R)
U = runif(n)
V = M * dgamma(Y, k, lambda) * U
I = which(Y >= a & Y <= b)
J = which(Y < a | Y > b)
curve(M * dgamma(x, k, lambda), xlim = c(0, max(Y)), lwd = 2, xlab = "Y", ylab = "V")
points(Y[I], V[I], col = "blue", pch = 16, cex = 0.2)
points(Y[J], V[J], col = "red", pch = 16, cex = 0.2)
legend("topright", c(expression(v == f(y)), "Accepted", "Rejected"), col = c("black",
    "blue", "red"), lty = c(1, NA, NA), lwd = c(2, NA, NA), pch = c(NA, 16,
    16))
```



Input: Proposal PDF g, interval $[a, b]$ and sample size n.

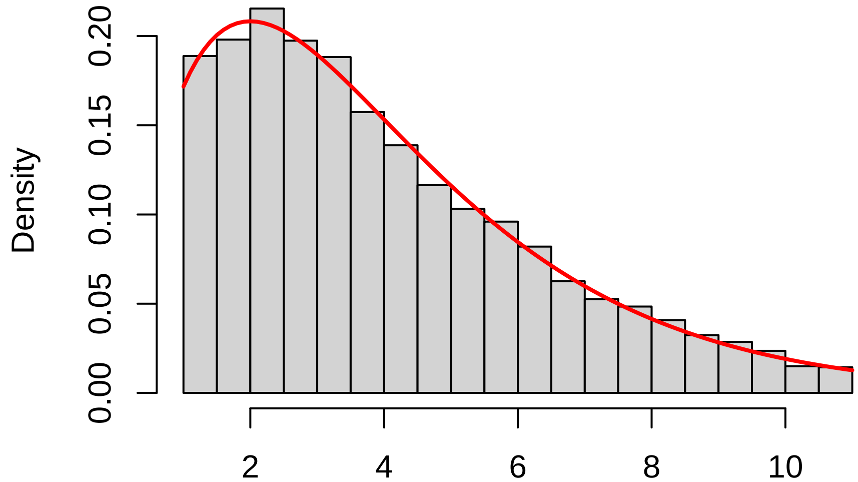
For $i = 1, 2, \ldots, n$, we iterate the following steps:

1: We generate $Y \sim g$.

2: If $Y \in [a, b]$, we let $X_i = Y$. Otherwise, we return to step 1.

Output: Random sample $X_1, X_2, \ldots, X_n$.

```r
n = 10000
k = 2
lambda = 0.5
a = 1
b = 11
M = 1/(pgamma(b, k, lambda) - pgamma(a, k, lambda))
X = numeric(n)
for (i in 1:n) {
    U = runif(k)
    R = -log(U)/lambda
    Y = sum(R)
    while (Y < a || Y > b) {
        U = runif(k)
        R = -log(U)/lambda
        Y = sum(R)
    }
    X[i] = Y
}
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(M * dgamma(x, k, lambda), add = TRUE, col = "red", lwd = 2)
```



Note 1.8. We observe that the expected number of iterations until the generation of X_i is equal to:

$$M = \frac{1}{G(b) - G(a)} = \frac{1}{\mathbb{P}(a \leqslant X \leqslant b)} > 1.$$

Therefore, the use of PDF g as a proposal is efficient when the probability $\mathbb{P}(a \leqslant X \leqslant b)$ is high. Otherwise, the use of the uniform distribution on $[a, b]$ as a proposal would be more efficient.

Example 1.14. We want to generate a random sample $(X_1, Y_1, Z_1), \ldots, (X_n, Y_n, Z_n)$ from the uniform distribution on the set $S = \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 \leqslant 2z, x \geqslant y \geqslant z\}$. We observe that:

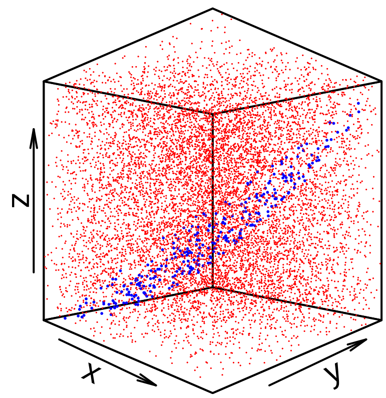
$$x^2 + y^2 \leqslant 2z \leqslant 2x \quad \Rightarrow \quad (x-1)^2 + y^2 \leqslant 1 \quad \Rightarrow \quad x \in [0, 2], \quad y \in [-1, 1],$$

$$x^2 + y^2 \leqslant 2z \leqslant 2y \quad \Rightarrow \quad x^2 + (y-1)^2 \leqslant 1 \quad \Rightarrow \quad x \in [-1, 1], \quad y \in [0, 2],$$

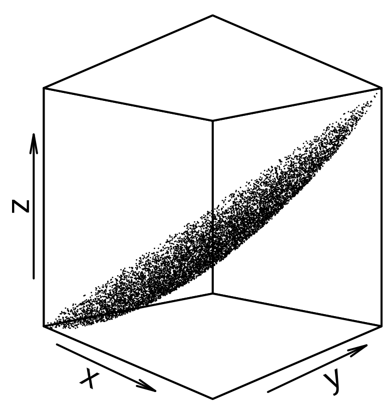
$$0 \leqslant x^2 + y^2 \leqslant 2z \quad \Rightarrow \quad z \geqslant 0.$$

Be intersecting all of the constraints, we infer that $S \subseteq [0, 1]^3$. Let (X, Y, Z) be a random vector which follows the uniform distribution on the cube $[0, 1]^3$ with PDF $g(x, y, z) = 1$ for $x, y, z \in [0, 1]$. We infer that the random variables X, Y, Z are independent and follow the Unif$[0, 1]$ distribution. Therefore, it suffices to generate a random sample from the distribution of (X, Y, Z) given that $(X, Y, Z) \in S$.

```
library(plot3D)
n = 10000
X = runif(n)
Y = runif(n)
Z = runif(n)
I = which(X >= Y & Y >= Z & X^2 + Y^2 <= 2 * Z)
J = which(X < Y | Y < Z | X^2 + Y^2 > 2 * Z)
scatter3D(X[J], Y[J], Z[J], phi = 0, theta = 45, col = "red", pch = 16, cex = 0.1)
scatter3D(X[I], Y[I], Z[I], phi = 0, theta = 45, col = "blue", add = TRUE, pch = 16,
    cex = 0.2)
```

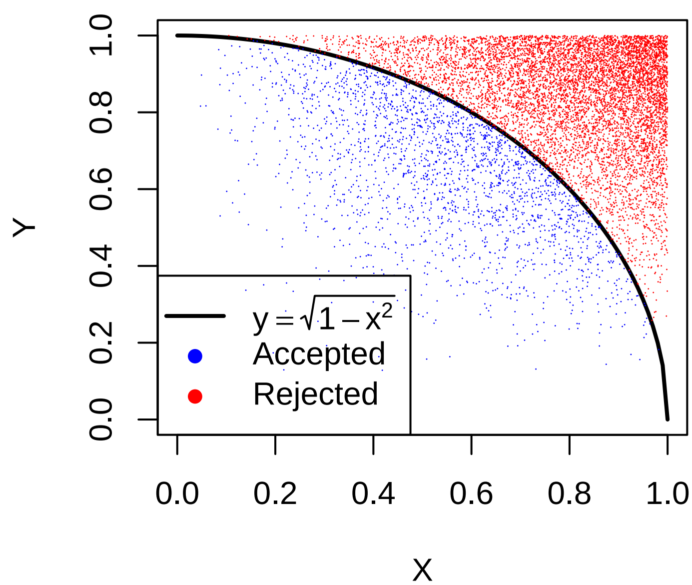



```
library(plot3D)
n = 10000
X = numeric(n)
Y = numeric(n)
Z = numeric(n)
for (i in 1:n) {
    X[i] = runif(1)
    Y[i] = runif(1)
    Z[i] = runif(1)
    while (X[i] < Y[i] || Y[i] < Z[i] || X[i]^2 + Y[i]^2 > 2 * Z[i]) {
        X[i] = runif(1)
        Y[i] = runif(1)
        Z[i] = runif(1)
    }
}
scatter3D(X, Y, Z, colvar = NA, phi = 0, theta = 45, pch = 16, cex = 0.1)
```



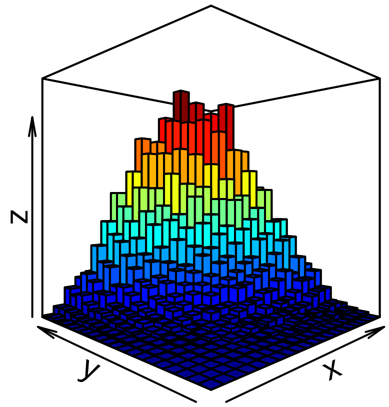
Example 1.15. We want to generate a random sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ following the PDF $f(x, y) = cx^2y^3$ for $(x, y) \in S = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leqslant 1, x, y \geqslant 0\} \subseteq [0, 1]^2$. Let $X \sim \text{Beta}(3, 1)$ and $Y \sim \text{Beta}(4, 1)$ be independent random variables with PDFs $f_X(x) = 3x^2$ and $f_Y(y) = 4y^3$ for $x, y \in [0, 1]$. We observe that $g(x, y) = f_X(x)f_Y(y) = 12x^2y^3$. Therefore, it suffices to generate a random sample from the distribution of (X, Y) given that $(X, Y) \in S$. For $x, y, u, v \in [0, 1]$, we calculate that $F_X(x) = x^3$, $F_Y(y) = y^4$, $F_X^{-1}(u) = u^{1/3}$ and $F_Y^{-1}(v) = v^{1/4}$.

```r
n = 10000
U = runif(n)
X = U^(1/3)
V = runif(n)
Y = V^(1/4)
I = which(X^2 + Y^2 <= 1)
J = which(X^2 + Y^2 > 1)
curve(sqrt(1 - x^2), xlim = c(0, 1), lwd = 2, xlab = "X", ylab = "Y")
points(X[I], Y[I], col = "blue", pch = 16, cex = 0.1)
points(X[J], Y[J], col = "red", pch = 16, cex = 0.1)
legend("bottomleft", c(expression(y == sqrt(1 - x^2)), "Accepted", "Rejected"),
    col = c("black", "blue", "red"), lty = c(1, NA, NA), lwd = c(2, NA, NA),
    pch = c(NA, 16, 16))
```




```r
library(plot3D)
n = 50000
X = numeric(n)
Y = numeric(n)
for (i in 1:n) {
    U = runif(1)
    V = runif(1)
    X[i] = U^(1/3)
    Y[i] = V^(1/4)
    while (X[i]^2 + Y[i]^2 > 1) {
        U = runif(1)
        V = runif(1)
        X[i] = U^(1/3)
        Y[i] = V^(1/4)
    }
```

```
}
hist3D(z = table(cut(X, 20), cut(Y, 20)), colkey = FALSE, phi = 0, theta = 315,
    border = 1)
```



Box-Muller Transform

Let $X, Y \sim \mathcal{N}(0,1)$ be independent random variables. Then, we calculate that:

$$f_{X,Y}(x,y) = f_X(x) f_Y(y) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \cdot \frac{1}{\sqrt{2\pi}} e^{-y^2/2} = \frac{1}{2\pi} e^{-\left(x^2+y^2\right)/2}.$$

We consider the change to polar coordinates $D = X^2 + Y^2$, $\Theta = \arctan \frac{Y}{X}$ or equivalently $X = \sqrt{D} \cos \Theta$, $Y = \sqrt{D} \sin \Theta$. For $d > 0$ and $\theta \in [0, 2\pi]$, we calculate that:

$$J_{X,Y}(d,\theta) = \frac{\partial(x,y)}{\partial(d,\theta)} = \begin{bmatrix} \frac{\partial x}{\partial d} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial d} & \frac{\partial y}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \frac{\cos\theta}{2\sqrt{d}} & -\sqrt{d}\sin\theta \\ \frac{\sin\theta}{2\sqrt{d}} & \sqrt{d}\cos\theta \end{bmatrix},$$

$$\det\left[J_{X,Y}(d,\theta)\right] = \frac{1}{2}\cos^2\theta + \frac{1}{2}\sin^2\theta = \frac{1}{2},$$

$$f_{D,\Theta}(d,\theta) = \left|\det\left[J_{X,Y}(d,\theta)\right]\right| f_{X,Y}\left(\sqrt{d}\cos\theta, \sqrt{d}\sin\theta\right) = \frac{1}{2} \cdot \frac{1}{2\pi} e^{-d/2} = \frac{1}{2} e^{-d/2} \cdot \frac{1}{2\pi}.$$

Therefore, we conclude that the random variables D and Θ are independent with $D \sim \mathrm{Exp}(1/2) \equiv \chi^2_2$ and $\Theta \sim \mathrm{Unif}[0, 2\pi]$.

Note 1.9. If $Z \sim \mathcal{N}(0,1)$, $\mu \in \mathbb{R}$ and $\sigma > 0$, then $X = \sigma Z + \mu \sim \mathcal{N}\left(\mu, \sigma^2\right)$.

Algorithm 1.3 Box-Muller Transform

Input: Expected value μ, standard deviation σ and sample size n.

For $i = 1, 2, \ldots, n/2$, we iterate the following steps:

1: We generate $U \sim \mathrm{Unif}[0,1]$ and let $D = -2\log U \sim \mathrm{Exp}(1/2)$.

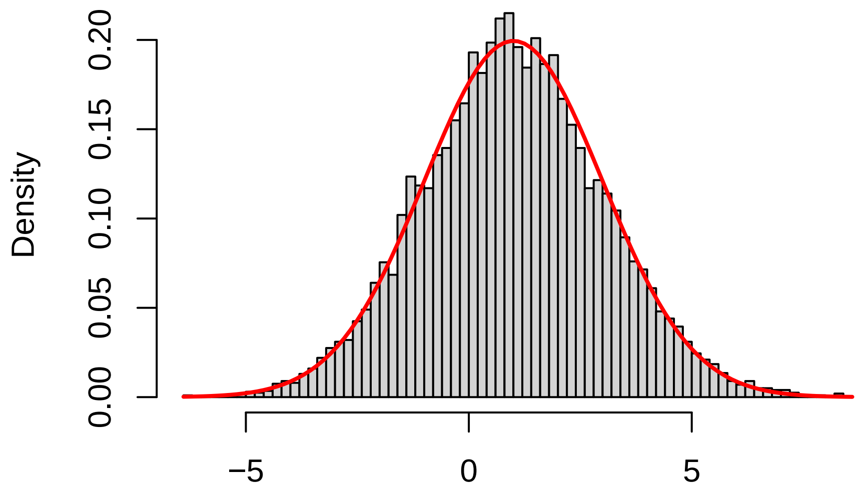
2: We generate $V \sim \mathrm{Unif}[0,1]$ and let $\Theta = 2\pi V \sim \mathrm{Unif}[0, 2\pi]$.

3: We let $Z_1 = \sqrt{D}\cos\Theta \sim \mathcal{N}(0,1)$ and $Z_2 = \sqrt{D}\sin\Theta \sim \mathcal{N}(0,1)$.

4: We let $X_{2i-1} = \sigma Z_1 + \mu \sim \mathcal{N}\left(\mu, \sigma^2\right)$ and $X_{2i} = \sigma Z_2 + \mu \sim \mathcal{N}\left(\mu, \sigma^2\right)$.

Output: Random sample $X_1, X_2, \ldots, X_n$.

```
n = 10000
mu = 1
sigma = 2
U = runif(n/2)
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
X = sigma * Z + mu
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dnorm(x, mu, sigma), add = TRUE, col = "red", lwd = 2)
```



Discrete Random Variable Generation

Let X be a discrete random variable with support $S = \mathbb{N}$ and PMF $p_j = \mathbb{P}(X = j)$ for $j = 0, 1, \ldots$. For $x \in \mathbb{N}$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant x) = \sum_{j=0}^{x} p_j,$$

$$F^{-}(u) = \inf\left\{x \in S : \sum_{j=0}^{x} p_j \geqslant u\right\} = \begin{cases} 0, & 0 \leqslant u \leqslant p_0 \\ 1, & p_0 < u \leqslant p_0 + p_1 \\ 2, & p_0 + p_1 < u \leqslant p_0 + p_1 + p_2 \\ \ldots & \end{cases}.$$

Example 1.16. We want to generate a random sample $X_1, X_2, \ldots, X_n$ following the PMF $p_1 = 0.2$, $p_2 = 0.15$, $p_3 = 0.25$, $p_4 = 0.4$. We calculate that:

$$F^{-}(u) = \begin{cases} 1, & 0 \leqslant u \leqslant 0.2 \\ 2, & 0.2 < u \leqslant 0.35 \\ 3, & 0.35 < u \leqslant 0.6 \\ 4, & 0.6 < u \leqslant 1 \end{cases}.$$

```
n = 10000
S = 1:4
pmf = c(0.2, 0.15, 0.25, 0.4)
X = numeric(n)
for (i in 1:n) {
    U = runif(1)
    j = 1
    cdf = pmf[1]
    while (U > cdf) {
        j = j + 1
        cdf = cdf + pmf[j]
    }
    X[i] = S[j]
}
table(factor(X, levels = S))/n
```

```
##
##      1      2      3      4
## 0.2010 0.1511 0.2511 0.3968
```

Note 1.10. Since the algorithm goes through the support of the random variable X from start to finish and it's more likely for X to take values with higher probability, it's more computationally efficient to sort the PMF of the random variable in decreasing order and then simulate from it.

```
n = 10000
S = 1:4
pmf = c(0.2, 0.15, 0.25, 0.4)
I = order(pmf, decreasing = TRUE)
pmf = pmf[I]
S = I[S]
X = numeric(n)
for (i in 1:n) {
    U = runif(1)
    j = 1
    cdf = pmf[1]
    while (U > cdf) {
        j = j + 1
        cdf = cdf + pmf[j]
    }
    X[i] = S[j]
}
table(factor(X, levels = S))/n
```

```
##
##      4      3      1      2
```

```
## 0.4027 0.2454 0.1946 0.1573
```

Example 1.17. We want to generate a finite path $X_1, X_2, \ldots, X_n$ from a discrete-time Markov chain with finite state-space $S = \{1, 2, \ldots, m\}$, initial distribution $a = [a_k]$ and transition probability matrix $P = [p_{k,\ell}]$. We know that:

$$\begin{aligned}\mathbb{P}\left(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\right) &= \mathbb{P}(X_1 = x_1)\mathbb{P}\left(X_2 = x_2 \mid X_1 = x_1\right) \cdots \mathbb{P}\left(X_n = x_n \mid X_{n-1} = x_{n-1}\right) \\ &= a_{x_1} p_{x_1,x_2} \cdots p_{x_{n-1},x_n}.\end{aligned}$$

Input: State-space S, initial distribution a, transition probability matrix P and path length n.

1: We generate X_1 from the initial distribution a.

2: For $i = 2, 3, \ldots, n$, we iterate the following step:

  i: We generate X_i following the PMF which is given by row X_{i-1} of the transition probability matrix P.

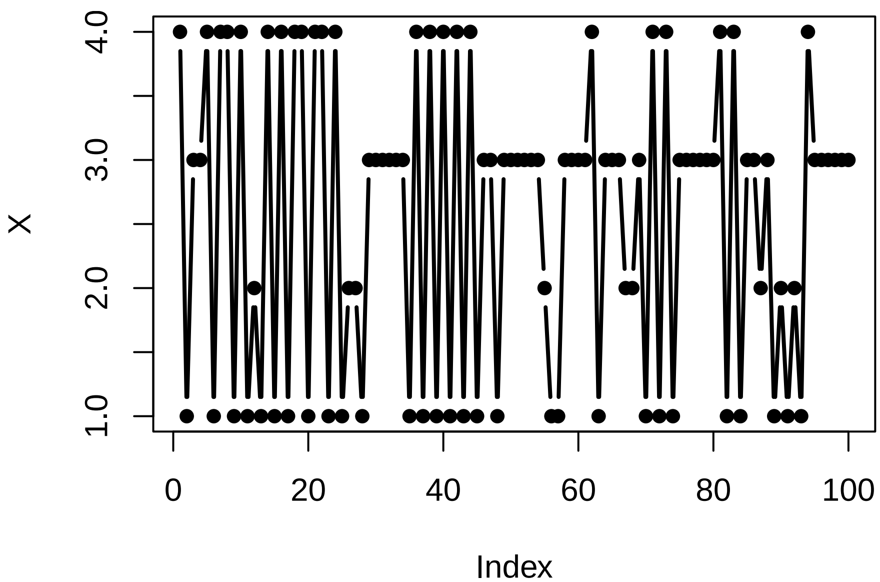
Output: Path $X_1, X_2, \ldots, X_n$.

```r
n = 100
m = 4
S = 1:m
a = c(0.2, 0.15, 0.25, 0.4)
P = rbind(c(0.1, 0.2, 0.3, 0.4), c(0.2, 0.5, 0.2, 0.1), c(0.2, 0.1, 0.6, 0.1),
    c(0.7, 0.1, 0.1, 0.1))
rownames(P) = S
colnames(P) = S
print(P)
```

```
##     1   2   3   4
## 1 0.1 0.2 0.3 0.4
## 2 0.2 0.5 0.2 0.1
## 3 0.2 0.1 0.6 0.1
## 4 0.7 0.1 0.1 0.1
```

```r
X = numeric(n)
U = runif(1)
j = 1
cdf = a[1]
while (U > cdf) {
    j = j + 1
    cdf = cdf + a[j]
}
X[1] = S[j]
for (i in 2:n) {
    pmf = P[X[i - 1], ]
    U = runif(1)
    j = 1
```

```
    cdf = pmf[1]
    while (U > cdf) {
        j = j + 1
        cdf = cdf + pmf[j]
    }
    X[i] = S[j]
}
plot(X, type = "b", pch = 16, lwd = 2)
```



Example 1.18. We want to generate a random sample $X_1, \dots, X_n \sim \text{Poisson}(\lambda)$. For $j = 0, 1, \dots$, we know that:

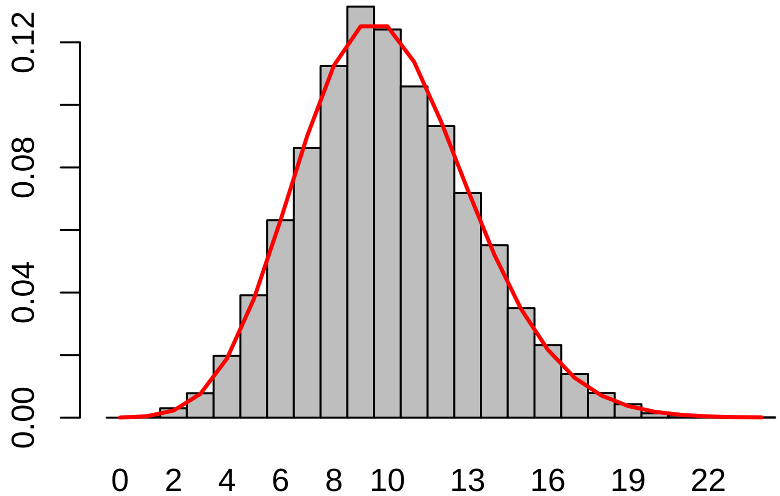
$$p_j = e^{-\lambda}\frac{\lambda^j}{j!}.$$

We observe that:

$$p_0 = e^{-\lambda}, \quad p_{j+1} = \frac{\lambda}{j+1}p_j.$$

```
n = 10000
lambda = 10
X = numeric(n)
for (i in 1:n) {
    U = runif(1)
    pmf = exp(-lambda)
    cdf = pmf
    while (U > cdf) {
        X[i] = X[i] + 1
        pmf = pmf * lambda/X[i]
        cdf = cdf + pmf
    }
}
```

```r
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0)
lines(0:max(X) + 0.5, dpois(0:max(X), lambda), col = "red", lwd = 2)
```



Let $\{N(t) : t \geqslant 0\}$ be a Poisson process with rate λ and inter-arrival times $Y_1, Y_2, \cdots \sim \text{Exp}(\lambda)$. We know that:

$$N(1) = \sup\left\{k \in \mathbb{N} : \sum_{j=1}^{k} Y_j \leqslant 1\right\} = \inf\left\{k \in \mathbb{N} : \sum_{j=1}^{k+1} Y_j > 1\right\} \sim \text{Poisson}(\lambda).$$

Input: Rate λ and sample size n.

For $i = 1, 2, \ldots, n$, we iterate the following steps:

1: We let $S \leftarrow 0$ and $k \leftarrow 0$.

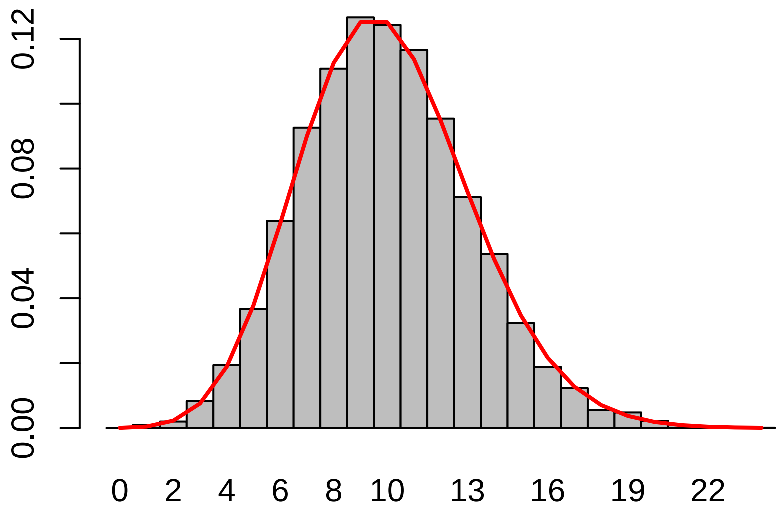
2: We generate $U \sim \text{Unif}[0, 1]$, let $Y = -\frac{1}{\lambda} \log U \sim \text{Exp}(\lambda)$ and let $S \leftarrow S + Y$.

3: If $S > 1$, then we let $X_i = k$. Otherwise, we let $k \leftarrow k + 1$ and return to step 2.

Output: Random sample $X_1, X_2, \ldots, X_n$ following the Poisson(λ) distribution.

```r
n = 10000
lambda = 10
X = numeric(n)
for (i in 1:n) {
    S = 0
    while (S <= 1) {
        U = runif(1)
        Y = -log(U)/lambda
        S = S + Y
        if (S <= 1) {
            X[i] = X[i] + 1
        }
    }
}
```

```
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0)
lines(0:max(X) + 0.5, dpois(0:max(X), lambda), col = "red", lwd = 2)
```



Example 1.19. We want to generate a random sample $X_1, \ldots, X_n \sim \text{Unif}\{1, 2, \ldots, k\}$. For $x \in S$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant x) = \frac{x}{k},$$

$$F^{-}(u) = \inf \left\{x \in S : u \leqslant \frac{x}{k}\right\} = \inf \left\{x \in S : x \geqslant ku\right\} = \lfloor ku \rfloor + 1.$$

```
n = 10000
k = 5
U = runif(n)
X = floor(k * U) + 1
table(factor(X, levels = 1:k))/n
```

```
##
##      1      2      3      4      5
## 0.2010 0.2017 0.2005 0.1903 0.2065
```

Example 1.20. We want to generate a random sample $X_1, \ldots, X_n \sim \text{Unif}\{a, a+1, \ldots, b\}$. For $x \in S$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant x) = \frac{x-a+1}{b-a+1},$$

$$F^{-}(u) = \inf \left\{x \in S : u \leqslant \frac{x-a+1}{b-a+1}\right\} = \inf \left\{x \in S : x \geqslant (b-a+1)u + a - 1\right\} = \lfloor (b-a+1)u \rfloor + a.$$

```
n = 10000
a = -1
b = 3
U = runif(n)
X = floor((b - a + 1) * U) + a
table(factor(X, levels = a:b))/n
```

```
##
```

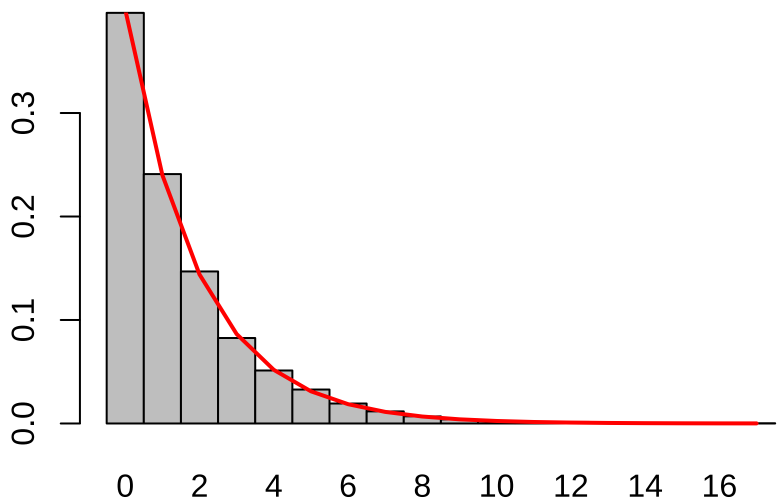
```
##     -1      0      1      2      3
## 0.2010 0.2017 0.2005 0.1903 0.2065
```

Example 1.21. We want to generate a random sample $X_1, \ldots, X_n \sim \text{Geom}(p)$ with PMF $p_j = p(1-p)^j$ for $j = 0, 1, \ldots$. For $x \in \mathbb{N}$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant x) = p \sum_{j=0}^{x} (1-p)^j = p \cdot \frac{1-(1-p)^{x+1}}{1-(1-p)} = 1-(1-p)^{x+1},$$

$$\begin{aligned} F^{-}(u) &= \min\left\{x \in \mathbb{N} : 1-(1-p)^{x+1} \geqslant u\right\} = \min\left\{x \in \mathbb{N} : (x+1)\log(1-p) \leqslant \log(1-u)\right\} \\ &= \min\left\{x \in \mathbb{N} : x \geqslant \frac{\log(1-u)}{\log(1-p)} - 1\right\} = \left\lfloor \frac{\log(1-u)}{\log(1-p)} \right\rfloor. \end{aligned}$$

```r
n = 10000
p = 0.4
U = runif(n)
X = floor(log(U)/log(1 - p))
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0)
lines(0:max(X) + 0.5, dgeom(0:max(X), p), col = "red", lwd = 2)
```



Example 1.22. We want to generate a random sample $X_1, \ldots, X_n \sim \text{Bernoulli}(p)$. We calculate that:

$$F^{-}(u) = \begin{cases} 0, & 0 \leqslant u \leqslant 1-p \\ 1, & 1-p < u \leqslant 1 \end{cases} = \begin{cases} 1, & 0 \leqslant 1-u < p \\ 0, & p \leqslant 1-u \leqslant 1 \end{cases}.$$

```r
n = 10000
p = 0.4
U = runif(n)
X = as.numeric(U < p)
table(factor(X, levels = 0:1))/n
```

```
##
##      0      1
```

```
## 0.5973 0.4027
```

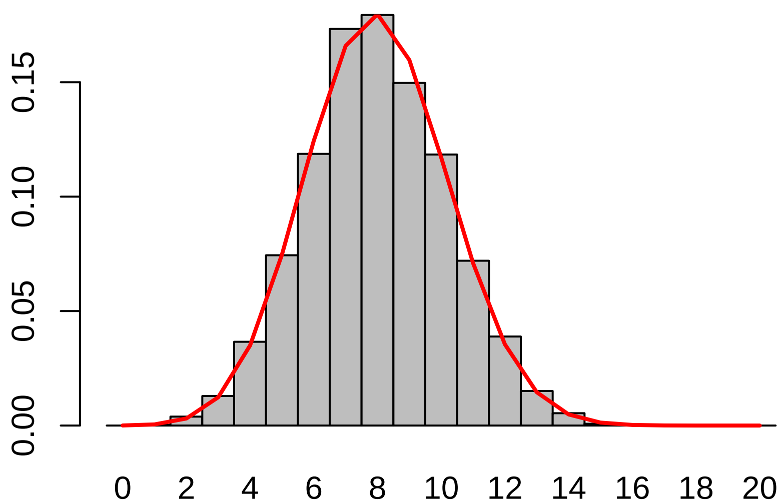
Example 1.23. We want to generate a random sample $X_1, \ldots, X_n \sim \text{Bin}(k, p)$. For $j = 0, 1, \ldots, k$, we know that:

$$p_j = \binom{k}{j} p^j (1-p)^{k-j}.$$

We observe that:

$$p_0 = (1-p)^k, \quad p_{j+1} = \frac{k-j}{j+1} \frac{p}{1-p} p_j.$$

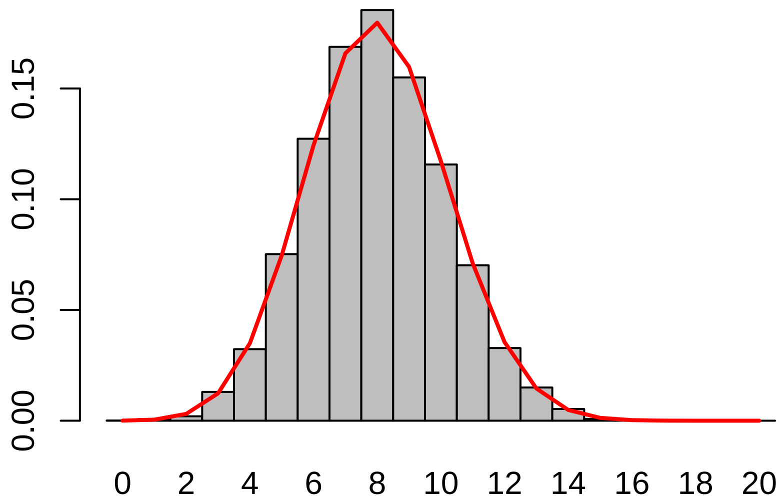
```r
n = 10000
k = 20
p = 0.4
X = numeric(n)
for (i in 1:n) {
    U = runif(1)
    pmf = (1 - p)^k
    cdf = pmf
    while (U > cdf) {
        pmf = pmf * (k - X[i])/(X[i] + 1) * p/(1 - p)
        cdf = cdf + pmf
        X[i] = X[i] + 1
    }
}
barplot(table(factor(X, levels = 0:k))/n, space = 0)
lines(0:k + 0.5, dbinom(0:k, k, p), col = "red", lwd = 2)
```



Let $Y_1, \ldots, Y_k \sim \text{Bernoulli}(p)$ be independent random variables. Then, we know that $Y_1 + \cdots + Y_k \sim \text{Bin}(k, p)$.

```r
n = 10000
k = 20
p = 0.4
U = matrix(runif(n * k), n)
Y = U < p
```

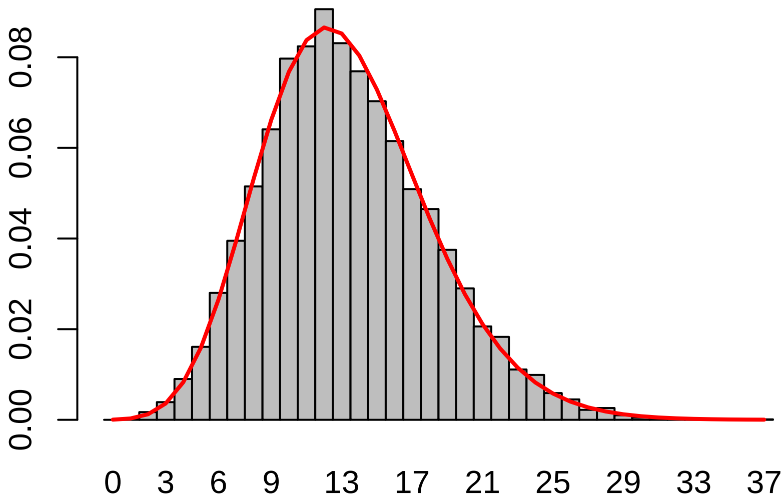
```
X = rowSums(Y)
barplot(table(factor(X, levels = 0:k))/n, space = 0)
lines(0:k + 0.5, dbinom(0:k, k, p), col = "red", lwd = 2)
```



Example 1.24. We want to generate a random sample $X_1, \ldots, X_n \sim \text{NegBin}(k, p)$ with the following PMF $p_j = \binom{j+k-1}{j} p^k (1-p)^j$ for $j = 0, 1, \ldots$. We observe that:

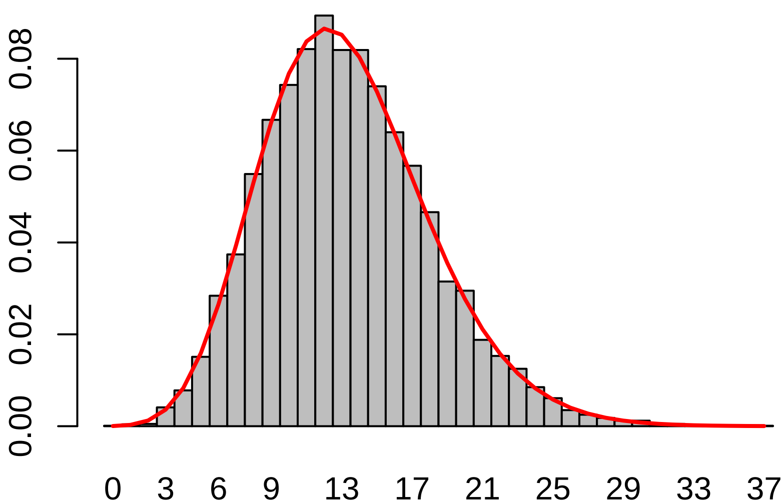
$$p_0 = p^k, \quad p_{j+1} = \frac{j+k}{j+1}(1-p)p_j.$$

```
n = 10000
k = 20
p = 0.6
X = numeric(n)
for (i in 1:n) {
    U = runif(1)
    pmf = p^k
    cdf = pmf
    while (U > cdf) {
        pmf = pmf * (1 - p) * (X[i] + k)/(X[i] + 1)
        cdf = cdf + pmf
        X[i] = X[i] + 1
    }
}
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0)
lines(0:max(X) + 0.5, dnbinom(0:max(X), k, p), col = "red", lwd = 2)
```



Let $Y_1, \ldots, Y_k \sim \text{Geom}(p)$ be independent random variables. Then, we know that $Y_1 + \cdots + Y_k \sim \text{NegBin}(k, p)$.

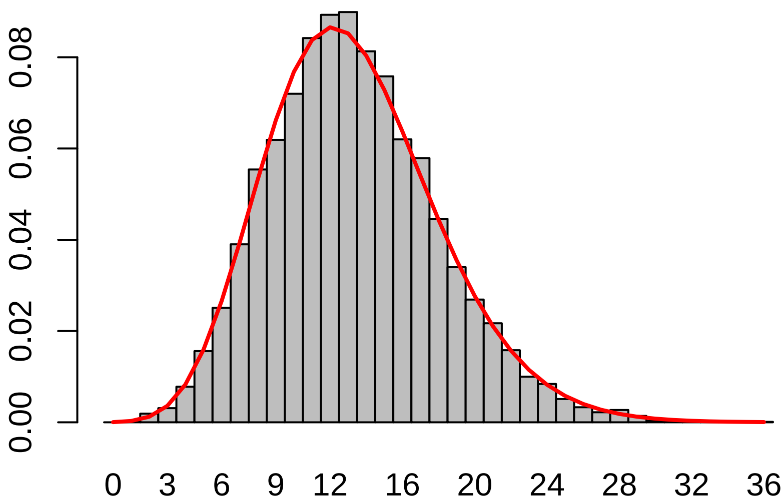
```
n = 10000
k = 20
p = 0.6
U = matrix(runif(n * k), n)
Y = floor(log(U)/log(1 - p))
X = rowSums(Y)
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0)
lines(0:max(X) + 0.5, dnbinom(0:max(X), k, p), col = "red", lwd = 2)
```



We know that the negative binomial distribution represents the number of failures until the k-th success in independent Bernoulli trials with common success probability p.

```
n = 10000
k = 20
p = 0.6
X = numeric(n)
for (i in 1:n) {
    success = 0
```

```
    while (success < k) {
        U = runif(1)
        Y = as.numeric(U < p)
        if (Y == 0) {
            X[i] = X[i] + 1
        } else {
            success = success + 1
        }
    }
}
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0)
lines(0:max(X) + 0.5, dnbinom(0:max(X), k, p), col = "red", lwd = 2)
```



Example 1.25. We want to generate a random sample $X^{(1)}, \dots, X^{(n)} \sim \text{Multinomial}(m, p_1, \dots, p_k)$. Let $Y_1, Y_2, \dots, Y_m$ be independent random variables with PMF $p = [p_j]$. Then, we know that:

$$\left(\sum_{\ell=1}^{m} \mathbb{1}_{\{Y_\ell = 1\}}, \dots, \sum_{\ell=1}^{m} \mathbb{1}_{\{Y_\ell = k\}} \right) \sim \text{Multinomial}(m, p_1, \dots, p_k).$$

```
n = 10000
m = 50
p = c(0.3, 0.1, 0.4, 0.2)
k = length(p)
X = matrix(0, n, k)
for (i in 1:n) {
    Y = numeric(m)
    for (j in 1:m) {
        U = runif(1)
        l = 1
        cdf = p[1]
        while (U > cdf) {
```

```
            l = l + 1
            cdf = cdf + p[l]
        }
        Y[j] = l
    }
    X[i, ] = table(factor(Y, levels = 1:k))
}
colMeans(X)
```

```
## [1] 15.0163  4.9851 20.0457  9.9529
```

Let $X = (X_1, X_2, \ldots, X_k) \sim \text{Multinomial}(m, p_1, \ldots, p_k)$. Then, we observe that:

$$\begin{aligned}\mathbb{P}(X = x) &= \mathbb{P}\left(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k\right) \\ &= \mathbb{P}(X_1 = x_1)\mathbb{P}\left(X_2 = x_2 \mid X_1 = x_1\right) \cdots \mathbb{P}\left(X_n = x_n \mid X_1 = x_1, \ldots, X_{n-1} = x_{n-1}\right).\end{aligned}$$

For $x_1 \in \{0, 1, \ldots, m\}$, we calculate according to the multinomial theorem that:

$$\begin{aligned}\mathbb{P}(X_1 = x_1) &= \sum_{x_2+\cdots+x_n=m-x_1} \mathbb{P}\left(X = x\right) = \sum_{x_2+\cdots+x_n=m-x_1} \frac{m!}{x_1!x_2!\cdots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k} \\ &= \frac{m!}{x_1!(m-x_1)!} p_1^{x_1} \sum_{x_2+\cdots+x_n=m-x_1} \frac{(m-x_1)!}{x_2!\cdots x_k!} p_2^{x_2} \cdots p_k^{x_k} = \binom{m}{x_1} p_1^{x_1} \left(p_2 + \cdots + p_k\right)^{m-x_1} \\ &= \binom{m}{x_1} p_1^{x_1} (1-p_1)^{m-x_1},\end{aligned}$$

i.e. $X_1 \sim \text{Bin}(m, p_1)$. For $x_2 \in \{0, 1, \ldots, m - x_1\}$, we calculate that:

$$\begin{aligned}\mathbb{P}(X_1 = x_1, X_2 = x_2) &= \sum_{x_3+\cdots+x_n=m-x_1-x_2} \mathbb{P}\left(X = x\right) = \sum_{x_3+\cdots+x_n=m-x_1-x_2} \frac{m!}{x_1!x_2!\cdots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k} \\ &= \frac{m!}{x_1!x_2!(m-x_1-x_2)!} p_1^{x_1} p_2^{x_2} \sum_{x_3+\cdots+x_n=m-x_1-x_2} \frac{(m-x_1-x_2)!}{x_3!\cdots x_k!} p_3^{x_3} \cdots p_k^{x_k} \\ &= \frac{m!}{x_1!x_2!(m-x_1-x_2)!} p_1^{x_1} p_2^{x_2} \left(p_3 + \cdots + p_k\right)^{m-x_1-x_2} \\ &= \frac{m!}{x_1!x_2!(m-x_1-x_2)!} p_1^{x_1} p_2^{x_2} (1-p_1-p_2)^{m-x_1-x_2},\end{aligned}$$

$$\begin{aligned}\mathbb{P}\left(X_2 = x_2 \mid X_1 = x_1\right) &= \frac{\mathbb{P}\left(X_1 = x_1, X_2 = x_2\right)}{\mathbb{P}(X_1 = x_1)} = \frac{\frac{m!}{x_1!x_2!(m-x_1-x_2)!} p_1^{x_1} p_2^{x_2} (1-p_1-p_2)^{m-x_1-x_2}}{\frac{m!}{x_1!(m-x_1)!} p_1^{x_1} (1-p_1)^{m-x_1}} \\ &= \binom{m-x_1}{x_2} \left(\frac{p_2}{1-p_1}\right)^{x_2} \left(1 - \frac{p_2}{1-p_1}\right)^{m-x_1-x_2},\end{aligned}$$

i.e. $(X_2 \mid X_1 = x_1) \sim \text{Bin}\left(m - x_1, \frac{p_2}{1-p_1}\right)$. For $\ell = 2, 3, \ldots, k-2$, we calculate that:

$$(X_{\ell+1} \mid X_1 = x_1, \ldots, X_\ell = x_\ell) \sim \text{Bin}\left(m - x_1 - \cdots - x_\ell, \frac{p_{\ell+1}}{1 - p_1 - \cdots - p_\ell}\right).$$

Finally, we observe that:

$$\mathbb{P}\left(X_k = x_k \mid X_1 = x_1, \ldots, X_{k-1} = x_{k-1}\right) = \begin{cases} 1, & x_k = m - x_1 - \cdots - x_{k-1} \\ 0, & x_k \neq m - x_1 - \cdots - x_{k-1} \end{cases}.$$

```
n = 10000
m = 50
p = c(0.3, 0.1, 0.4, 0.2)
k = length(p)
X = matrix(0, n, k)
for (i in 1:n) {
    trials = m
    prob = 1
    for (l in 1:(k - 1)) {
        U = runif(trials)
        X[i, l] = sum(U < p[l]/prob)
        trials = trials - X[i, l]
        prob = prob - p[l]
    }
    X[i, k] = trials
}
colMeans(X)
```

```
## [1] 15.0006  4.9823 20.0646  9.9525
```

Note 1.11. The first simulation method is more efficient when $m \ll k$, whereas the second simulation method is more efficient when $k \ll m$.

Example 1.26. We want to generate a random sample $X^{(1)}, \ldots, X^{(n)} \sim \text{Hypergeom}(m, r_1, \ldots, r_k)$ with PMF:

$$\mathbb{P}(X = x) = \mathbb{P}\left(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k\right) = \frac{\binom{r_1}{x_1}\binom{r_2}{x_2}\cdots\binom{r_k}{x_k}}{\binom{r_1+r_2+\cdots+r_k}{m}}.$$

```
n = 10000
m = 50
r = c(30, 10, 40, 20)
k = length(r)
X = matrix(0, n, k)
for (i in 1:n) {
    count = r
    total = sum(r)
    for (j in 1:m) {
        pmf = count/total
        U = runif(1)
        l = 1
```

```
        cdf = pmf[1]
        while (U > cdf) {
            l = l + 1
            cdf = cdf + pmf[l]
        }
        X[i, l] = X[i, l] + 1
        count[l] = count[l] - 1
        total = total - 1
    }
}
colMeans(X)
```

```
## [1] 15.0137  4.9887 20.0386  9.9590
```

Composition Method

Let X be a random variable with CDF $F_X(x)$ and absolutely continuous random variable Y with PDF $f_Y(y)$. We know that:

$$F_X(x) = \mathbb{P}(X \leqslant x) = \int_{\mathbb{R}} \mathbb{P}(X \leqslant x \mid Y = y) f_Y(y) dy = \int_{\mathbb{R}} F_{X|Y}(x \mid y) f_Y(y) dy.$$

Example 1.27. For $x \in [0,1]$, we want to generate a random sample $X_1, X_2, \ldots, X_n$ following the CDF:

$$F(x) = \int_0^\infty x^y e^{-y} dy.$$

Consider the PDF $g(y) = e^{-y}$ for $y > 0$, i.e. consider the random variable $Y \sim \text{Exp}(1)$. For $x \in [0,1]$, we calculate that:

$$F(x) = \mathbb{P}(X \leqslant x) = \int_0^\infty \mathbb{P}(X \leqslant x \mid Y = y)\, g(y) dy,$$

which implies that $F_{X|Y}(x \mid y) = \mathbb{P}(X \leqslant x \mid Y = y) = x^y$ and $F_{X|Y}^{-1}(u \mid y) = u^{1/y}$.

Input: CDFs F_Y, $F_{X|Y}$ and sample size n.

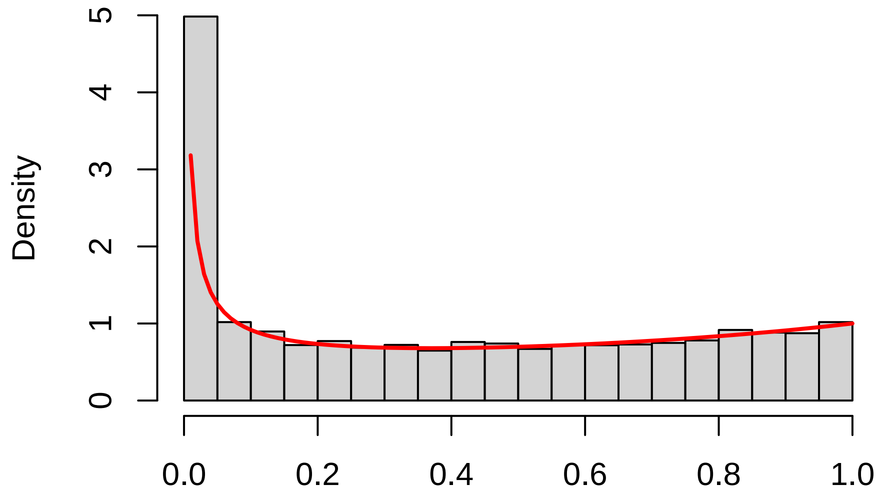
For $i = 1, 2, \ldots, n$, we iterate the following steps:

1: We generate $U \sim \text{Unif}[0,1]$ and let $Y = F_Y^{-1}(U)$.

2: We generate $V \sim \text{Unif}[0,1]$ and let $X_i = F_{X|Y}^{-}(V \mid Y)$.

Output: Random sample $X_1, X_2, \ldots, X_n$ following the CDF F_X.

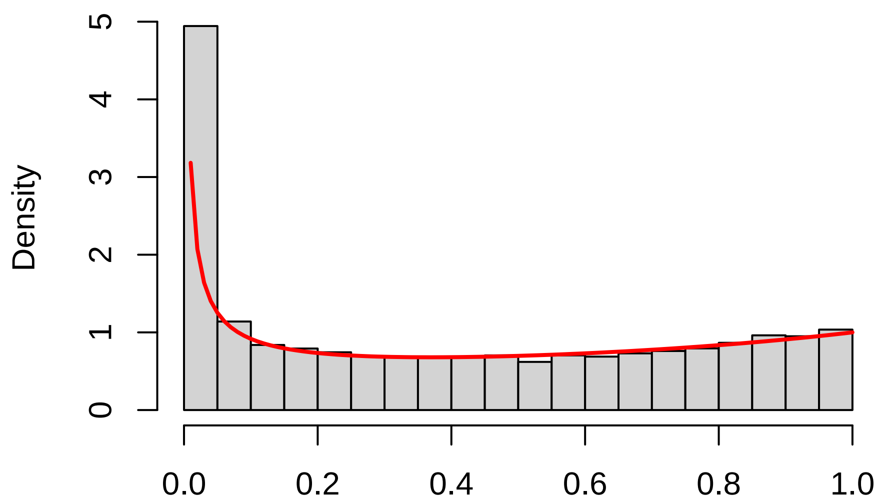
```
n = 10000
U = runif(n)
Y = -log(U)
V = runif(n)
X = V^(1/Y)
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(1/(x * (1 - log(x))^2), add = TRUE, col = "red", lwd = 2)
```



For $x \in [0, 1]$, we can directly calculate that:

$$F(x) = \int_0^\infty \left(\frac{e}{x}\right)^{-y} dy = \left[-\frac{1}{\log\frac{e}{x}}\left(\frac{e}{x}\right)^{-y}\right]_{y=0}^{\infty} = \frac{1}{1-\log x}, \quad F^{-1}(u) = e^{1-1/u}, \quad f(x) = \frac{1}{x\,(1-\log x)^2}.$$

```
n = 10000
U = runif(n)
X = exp(1 - 1/U)
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(1/(x * (1 - log(x))^2), add = TRUE, col = "red", lwd = 2)
```



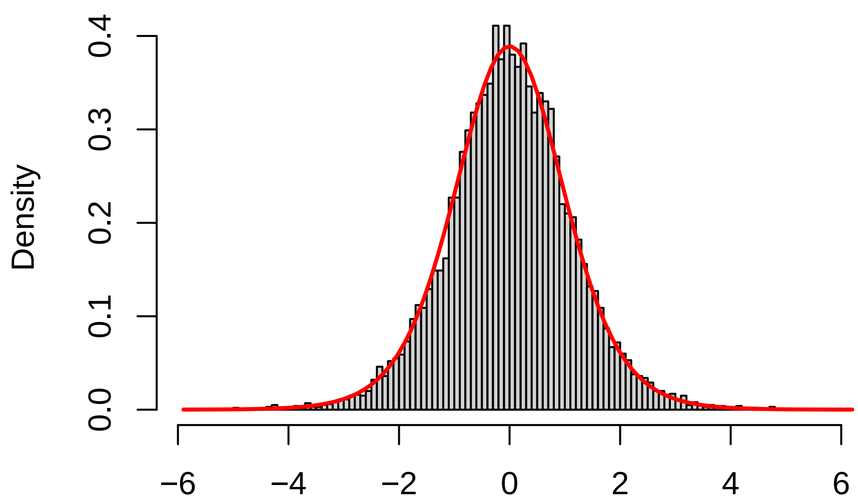
Example 1.28. We want to generate a random sample $X_1, X_2, \ldots, X_n \sim t_\nu$. Consider independent random variables $Z \sim \mathcal{N}(0, 1)$ and $Y \sim \chi^2(\nu) \equiv \text{Gamma}\left(\frac{\nu}{2}, \frac{1}{2}\right)$. Then, we know that:

$$X = \frac{Z}{\sqrt{Y/\nu}} \sim t_\nu.$$

We observe that:

$$(X \mid Y = y) = \frac{Z}{\sqrt{y/\nu}} \sim \mathcal{N}\left(0, \frac{\nu}{y}\right).$$

```
n = 10000
nu = 10
M = (nu/2)^(nu/2)/gamma(nu/2) * exp(-(nu - 2)/2)
Y = numeric(n)
for (i in 1:n) {
    W = runif(1)
    Y[i] = -nu * log(W)
    U = runif(1)
    V = M * dexp(Y[i], 1/nu) * U
    while (dchisq(Y[i], nu) < V) {
        W = runif(1)
        Y[i] = -nu * log(W)
        U = runif(1)
        V = M * dexp(Y[i], 1/nu) * U
    }
}
U = runif(n/2)
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
X = sqrt(nu/Y) * Z
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dt(x, nu), add = TRUE, col = "red", lwd = 2)
```

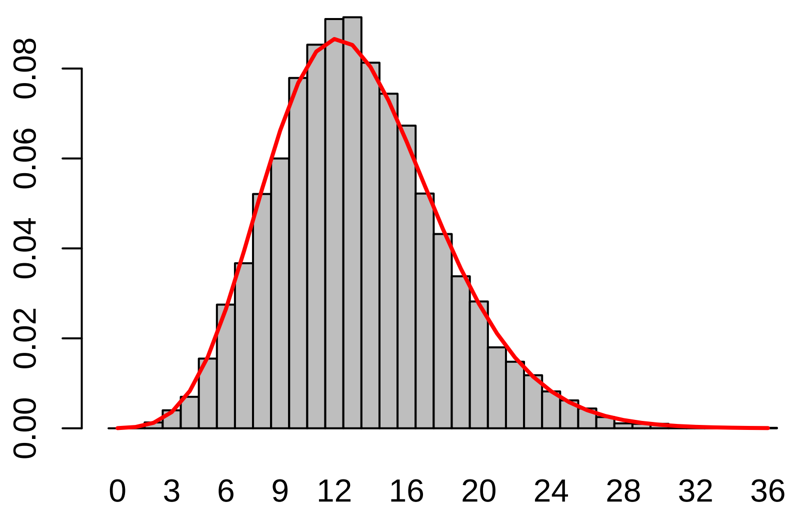


Example 1.29. Let $\{N(t) : t \geqslant 0\}$ be a Poisson process with rate 1 which counts the number of coin tosses up to time t. For $p \in [0, 1]$, the coin comes up "H" with probability p and "T" with probability $1 - p$. Consider the Poisson processes $\{N_H(t) : t \geqslant 0\}$ with rate p and $\{N_T(t) : t \geqslant 0\}$ with rate $1 - p$, which count the number of "H" and "T" respectively up to time t. We know that the processes $\{N_H(t) : t \geqslant 0\}$ and $\{N_T(t) : t \geqslant 0\}$ are independent and constitute a thinning of the process $\{N(t) : t \geqslant 0\}$. We know that the event count in the process $\{N_T(t) : t \geqslant 0\}$ up to the k-th event in the process $\{N_H(t) : t \geqslant 0\}$ follows the NegBin(k, p) distribution. However,

the event count $N_T(t)$ up to time t follows the Poisson $((1-p)t)$ distribution and the time S_k up to the k-th "H" follows the Gamma(k, p) distribution. For $j = 0, 1, \ldots$, we verify that:

$$\begin{aligned}
\mathbb{P}\left(N_T\left(S_k\right)=j\right) &= \int_0^{\infty} \mathbb{P}\left(N_T\left(S_k\right)=j \mid S_k=y\right) f_{S_k}(y) dy = \int_0^{\infty} \mathbb{P}\left(N_T\left(y\right)=j \mid S_k=y\right) f_{S_k}(y) dy \\
&= \int_0^{\infty} \mathbb{P}\left(N_T\left(y\right)=j\right) f_{S_k}(y) dy = \int_0^{\infty} e^{-(1-p)y} \frac{[(1-p)y]^j}{j!} \frac{p^k}{(k-1)!} y^{k-1} e^{-py} dy \\
&= \frac{1}{j!(k-1)!} p^k (1-p)^j \int_0^{\infty} y^{j+k-1} e^{-y} dy = \frac{1}{j!(k-1)!} p^k (1-p)^j \cdot \frac{(j+k-1)!}{1^{j+k}} \\
&= \binom{j+k-1}{j} p^k (1-p)^j .
\end{aligned}$$

```
n = 10000
k = 20
p = 0.6
U = matrix(runif(n * k), n)
R = -log(U)/p
Y = rowSums(R)
X = numeric(n)
for (i in 1:n) {
    S = 0
    while (S <= Y[i]) {
        U = runif(1)
        R = -log(U)/(1 - p)
        S = S + R
        if (S <= Y[i]) {
            X[i] = X[i] + 1
        }
    }
}
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0)
lines(0:max(X) + 0.5, dnbinom(0:max(X), k, p), col = "red", lwd = 2)
```



Let X be a random variable with CDF $F_X(x)$ and discrete random variable Y with PMF $w = [w_j]$. Additionally, we define the conditional CDFs $F_j(x) = F_{X|Y}(x \mid j)$ for $j = 0, 1, \ldots$. We know that:

$$F_X(x) = \mathbb{P}(X \leqslant x) = \sum_{j=0}^{\infty} \mathbb{P}(Y = j)\mathbb{P}(X \leqslant x \mid Y = j) = \sum_{j=0}^{\infty} w_j F_{X|Y}(x \mid j) = \sum_{j=0}^{\infty} w_j F_j(x).$$

The CDF F_X is called a mixture distribution.

Note 1.12. i. If $F_0, F_1, \ldots$ are absolutely continuous CDFs with corresponding PDFs $f_0, f_1, \ldots$, then F is an absolutely continuous CDF with corresponding PDF:

$$f(x) = \sum_{j=0}^{\infty} w_j f_j(x).$$

ii. If the CDFs $F_0, F_1, \ldots$ are step functions with corresponding PMFs $p^{(0)}, p^{(1)}, \ldots$, then the CDF F is a step function with corresponding PMF:

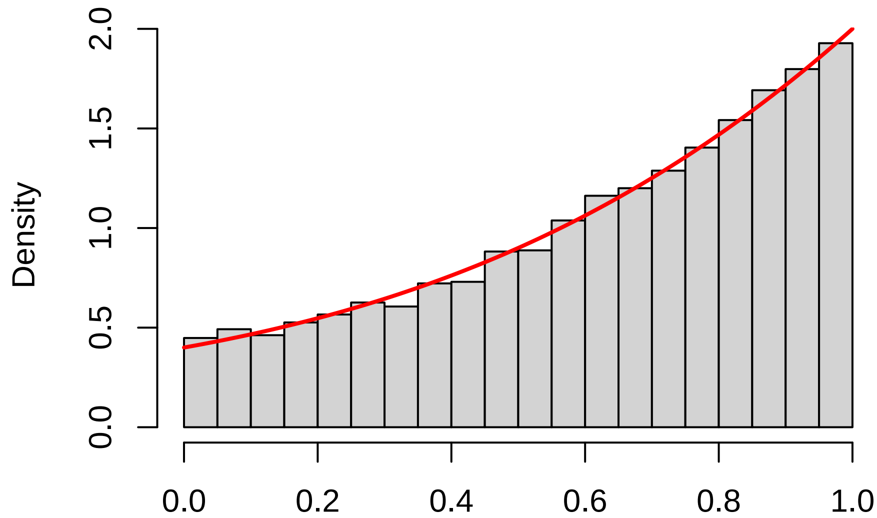
$$p_\ell = \sum_{j=0}^{\infty} w_j p_\ell^{(j)}.$$

Example 1.30. For $x \in [0, 1]$, we want to generate a random sample $X_1, X_2, \ldots, X_n$ with CDF:

$$F(x) = \sum_{j=1}^{k} w_j x^j,$$

where $w = [w_j]$ is a PMF. We observe that the CDFs $F_j(x) = x^j$ correspond to the Beta$(j, 1)$ distributions and calculate that $F_j^{-1}(u) = u^{1/j}$.

```
n = 10000
S = 1:4
w = c(0.4, 0.3, 0.2, 0.1)
Y = numeric(n)
for (i in 1:n) {
    U = runif(1)
    j = 1
    cdf = w[1]
    while (U > cdf) {
        j = j + 1
        cdf = cdf + w[j]
    }
    Y[i] = S[j]
}
U = runif(n)
X = U^(1/Y)
hist(X, "FD", freq = FALSE, main = NA, xlim = c(0, 1), xlab = NA)
curve(w[1] * dbeta(x, 1, 1) + w[2] * dbeta(x, 2, 1) + w[3] * dbeta(x, 3, 1) +
    w[4] * dbeta(x, 4, 1), add = TRUE, col = "red", lwd = 2)
```



Example 1.31. For $j = 0, 1, \ldots,$ we want to generate a random sample $X_1, X_2, \ldots, X_n$ with PMF:

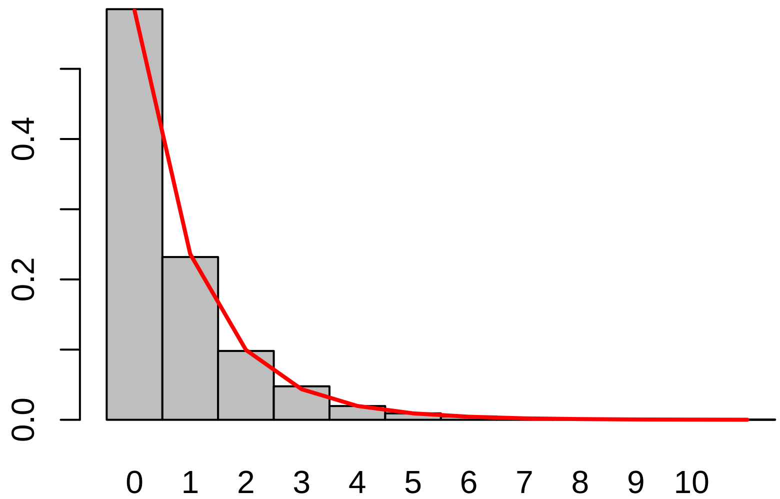
$$p_j = \frac{1}{2^{j+2}} + \frac{1}{3^{j+1}}.$$

We observe that:

$$p_j = \underbrace{\frac{1}{2}}_{w_1} \cdot \underbrace{\left(\frac{1}{2}\right)^j \frac{1}{2}}_{p_j^{(1)}} + \underbrace{\frac{1}{2}}_{w_2} \cdot \underbrace{\left(\frac{1}{3}\right)^j \frac{2}{3}}_{p_j^{(2)}}.$$

Additionally, we observe that the PMF $p^{(1)}$ corresponds to the geometric distribution with success probability $\frac{1}{2}$, whereas the PMF $p^{(2)}$ corresponds to the geometric distribution with success probability $\frac{2}{3}$.

```
n = 10000
w = c(0.5, 0.5)
p = c(0.5, 2/3)
U = runif(n)
Y = ifelse(U < w[1], 1, 2)
V = runif(n)
X = floor(log(V)/log(1 - p[Y]))
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0)
lines(0:max(X) + 0.5, w[1] * dgeom(0:max(X), p[1]) + w[2] * dgeom(0:max(X),
    p[2]), col = "red", lwd = 2)
```



Lemma 1.5. If $X \sim \text{Exp}(\lambda)$ and $\mu \in \mathbb{R}$, then the random variable $W_1 = \mu - X$ follows the PDF $f_{W_1}(x) = \lambda e^{-\lambda(\mu - x)}$ for $x < \mu$.

Proof. For $x < \mu$, we calculate that:

$$F_{W_1}(x) = \mathbb{P}(W_1 \leqslant x) = \mathbb{P}(X \geqslant \mu - x) = e^{-\lambda(\mu - x)}, \quad f_{W_1}(x) = \frac{\partial F_{W_1}(x)}{\partial x} = \lambda e^{-\lambda(\mu - x)}.$$

□

Note 1.13. We have shown that the random variable $W_2 = \mu + X$ follows the PDF $f_{W_2}(x) = \lambda e^{-\lambda(x-\mu)}$ for $x \geqslant \mu$.

Example 1.32. For $x \in \mathbb{R}$, we want to generate a random sample $X_1, X_2, \dots, X_n$ with PDF:

$$f(x) = \frac{\lambda}{2} e^{-\lambda|x - \mu|}.$$

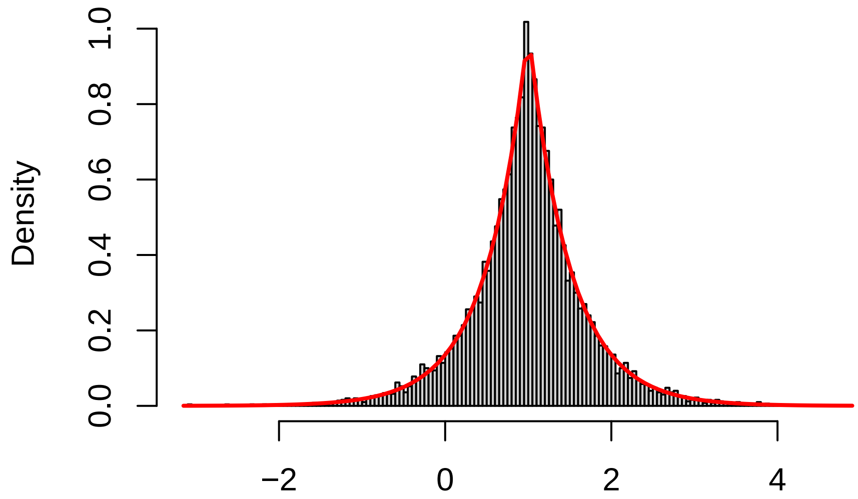
We observe that:

$$f(x) = \begin{cases} \frac{1}{2}\lambda e^{-\lambda(\mu - x)}, & x < \mu \\ \frac{1}{2}\lambda e^{-\lambda(x - \mu)}, & x \geqslant \mu \end{cases}.$$

Therefore, we infer that:

$$f(x) = \frac{1}{2} f_{W_1}(x) + \frac{1}{2} f_{W_2}(x).$$

```r
n = 10000
lambda = 2
mu = 1
U = runif(n)
Y = ifelse(U < 0.5, 1, 2)
V = runif(n)
X = ifelse(Y == 1, mu + log(V)/lambda, mu - log(V)/lambda)
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dexp(abs(x - mu), lambda)/2, add = TRUE, col = "red", lwd = 2)
```



Example 1.33. We want to generate a random sample $X_1, X_2, \ldots, X_n$ with CDF:

$$F(x) = \begin{cases} \frac{1-e^{-2x}+2x}{3}, & 0 \leqslant x \leqslant 1 \\ \frac{3-e^{-2x}}{3}, & x > 1 \end{cases}.$$

For $x \geqslant 0$, we define the following CDFs:

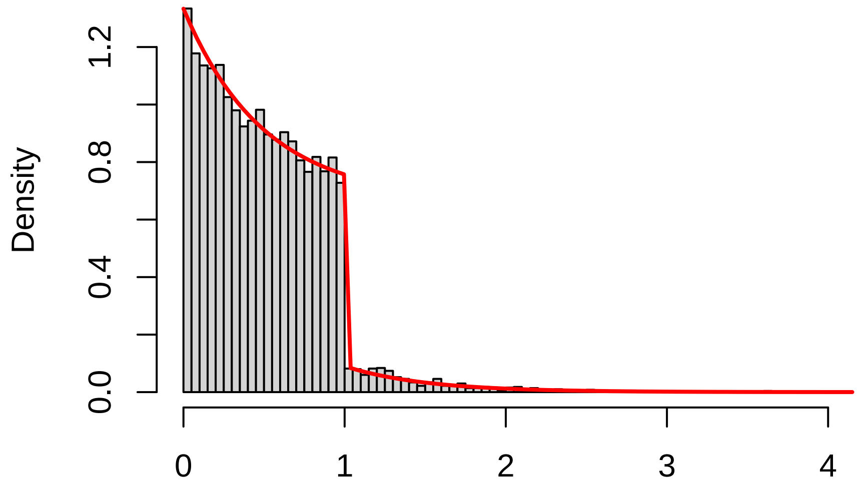
$$F_1(x) = 1 - e^{-2x}, \quad F_2(x) = \begin{cases} x, & 0 \leqslant x \leqslant 1 \\ 1, & x > 1 \end{cases}.$$

We observe that:

$$F(x) = \frac{1}{3}F_1(x) + \frac{2}{3}F_2(x).$$

Additionally, we observe that F_1 is the CDF of the exponential distribution with parameter 2, whereas F_2 is the CDF of the uniform distribution on $[0, 1]$.

```
n = 10000
w = c(1/3, 2/3)
lambda = 2
U = runif(n)
Y = ifelse(U < w[1], 1, 2)
V = runif(n)
X = ifelse(Y == 1, -log(V)/lambda, V)
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(w[1] * dexp(x, lambda) + w[2] * dunif(x), add = TRUE, col = "red", lwd = 2)
```



Example 1.34. For $x \geqslant 0$, we want to generate a random sample $X_1, X_2, \ldots, X_n$ with CDF:

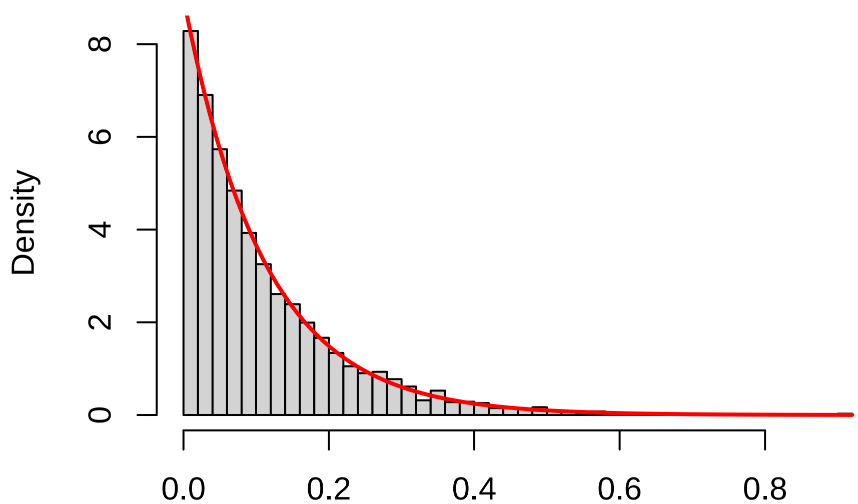
$$F(x) = \frac{2 - e^{-9x}}{2}.$$

We define the CDFs $F_1(x) = 1 - e^{-9x}$ and $F_2(x) = 1$. We observe that:

$$F(x) = \frac{1}{2}F_1(x) + \frac{1}{2}F_2(x).$$

Additionally, we observe that F_1 is the CDF of the exponential distribution with parameter 9, whereas F_2 is the CDF of the degenerate random variable $Y = 0$.

```
n = 10000
w = c(0.5, 0.5)
lambda = 9
U = runif(n)
Y = ifelse(U < w[1], 1, 2)
V = runif(n)
X = ifelse(Y == 1, -log(V)/lambda, 0)
hist(X[Y == 1], "FD", freq = FALSE, main = NA, xlab = NA)
curve(dexp(x, lambda), add = TRUE, col = "red", lwd = 2)
```



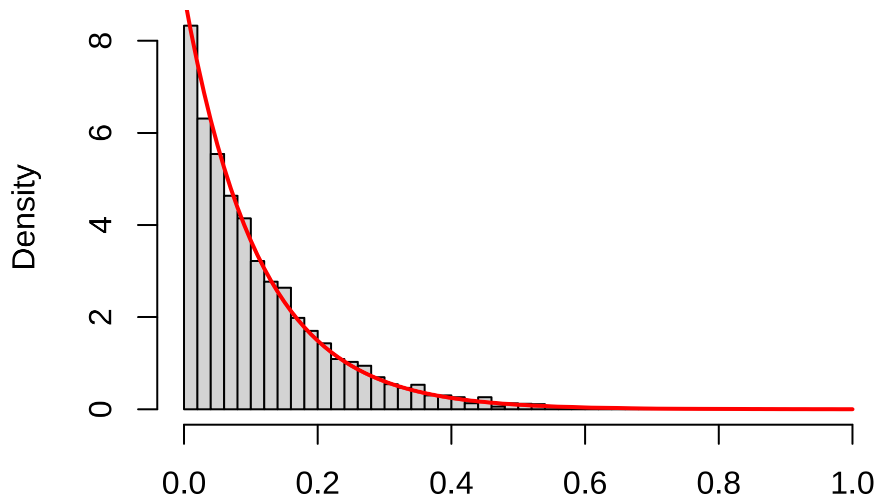
We observe that $F(0) = 0.5$. For $u \in [0, 0.5]$, it follows $F^{-}(u) = 0$. For $u \in (0.5, 1]$, we calculate that:

$$F(x) = u \Leftrightarrow x = -\frac{1}{9} \log\left[2(1-u)\right].$$

Therefore, we infer that:

$$F^{-}(u) = \begin{cases} 0, & 0 \leqslant u \leqslant 0.5 \\ -\frac{1}{9}\log\left[2(1-u)\right], & 0.5 < u \leqslant 1 \end{cases}.$$

```
n = 10000
U = runif(n)
X = ifelse(U <= 0.5, 0, -log(2 * (1 - U))/lambda)
hist(X[U > 0.5], "FD", freq = FALSE, main = NA, xlab = NA)
curve(dexp(x, lambda), add = TRUE, col = "red", lwd = 2)
```



2 Monte Carlo Method

We want to approximate the following integral:

$$I = \int_0^1 g(x)dx.$$

We know that the random variable $U \sim \text{Unif}[0,1]$ follows the PDF $f(x) = 1$ for $x \in [0,1]$. Hence, we observe that:

$$I = \int_0^1 g(x)f(x)dx = \mathbb{E}\left[g(U)\right].$$

Let $U_1, U_2, \ldots, U_n$ be a random sample from the Unif[0, 1] distribution. According to the strong law of large numbers, we know that:

$$\frac{1}{n}\sum_{i=1}^{n} g(U_i) \xrightarrow{\text{a.s.}} \mathbb{E}\left[g(U)\right] = I.$$

Example 2.1. We want to approximate the following integral:

$$I = \int_0^1 e^{e^x} dx.$$

```
n = 1e+05
U = runif(n)
I = mean(exp(exp(U)))
print(I)
```

```
## [1] 6.318484
```

Example 2.2. We want to approximate the following integral:

$$I = \int_0^1 \int_0^1 e^{(x+y)^2} dxdy.$$

Consider the independent random variables $U, V \sim \text{Unif}[0,1]$ with PDF $f_{U,V}(u,v) = f_U(u)f_V(v) = 1$ for $u, v \in [0,1]$. Then, we observe that $I = \mathbb{E}\left[e^{(U+V)^2}\right]$.

```
n = 1e+05
U = runif(n)
V = runif(n)
I = mean(exp((U + V)^2))
print(I)
```

```
## [1] 4.886297
```

More generally, we want to approximate the following interval:

$$I = \int_S g(x)dx.$$

Consider the random variable X with PDF $f(x)$ and support S. If we let $h(x) = \frac{g(x)}{f(x)}$, then we observe that:

$$I = \int_S \frac{g(x)}{f(x)} \cdot f(x) dx = \mathbb{E}\left[h(X)\right].$$

Let $X_1, X_2, \dots, X_n$ be a random sample following the PDF $f(x)$. According to the strong law of large numbers, we know that:

$$\frac{1}{n}\sum_{i=1}^{n} h(X_i) \overset{\text{a.s.}}{\to} \mathbb{E}\left[h(X)\right] = I.$$

Example 2.3. We want to approximate the following integral:

$$I = \int_{-2}^{2} e^{x+x^2} dx.$$

Consider the random variable $X \sim \text{Unif}[-2,2]$ with PDF $f(x) = \frac{1}{4}$ for $x \in [-2,2]$. Then, we observe that:

$$I = \int_{-2}^{2} 4e^{x+x^2} \frac{1}{4} dx = \mathbb{E}\left(4e^{X+X^2}\right).$$

```
n = 1e+05
U = runif(n)
X = 4 * U - 2
I = mean(4 * exp(X + X^2))
print(I)
```

```
## [1] 93.76997
```

Example 2.4. We want to approximate the following integral:

$$I = \int_{0}^{\infty} \frac{x}{(1+x^2)^2} dx.$$

Consider the random variable $X \sim \text{Exp}(1)$ with PDF $f(x) = e^{-x}$ for $x > 0$. Then, we observe that:

$$I = \int_{0}^{\infty} \frac{xe^x}{(1+x^2)^2} e^{-x} dx = \mathbb{E}\left[\frac{Xe^X}{(1+X^2)^2}\right].$$

```
n = 1e+05
U = runif(n)
X = -log(U)
I = mean(X * exp(X)/(1 + X^2)^2)
print(I)
```

```
## [1] 0.4993482
```

Example 2.5. We want to approximate the following integral:

$$I = \int_{-\infty}^{\infty} x^4 e^{-x^2} dx.$$

Consider the random variable $X \sim \mathcal{N}(0, 0.5)$ with PDF $f(x) = \frac{1}{\sqrt{\pi}} e^{-x^2}$ for $x \in \mathbb{R}$. Then, we observe that:

$$I = \int_{-\infty}^{\infty} \sqrt{\pi} x^4 \frac{1}{\sqrt{\pi}} e^{-x^2} dx = \mathbb{E}\left(\sqrt{\pi} X^4\right).$$

```
n = 1e+05
U = runif(n/2)
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
X = Z/sqrt(2)
I = mean(sqrt(pi) * X^4)
print(I)
```

```
## [1] 1.328683
```

Example 2.6. We want to approximate the following integral:

$$I = \int_{2}^{\infty} e^{-x^2/2} \sin(2\pi x) dx.$$

Consider the random variable X with PDF $f(x) = 4e^{-4(x-2)}$ for $x > 2$. Then, we observe that:

$$I = \int_{2}^{\infty} \frac{1}{4} e^{-x^2/2+4x-8} \sin(2\pi x) \cdot 4e^{-4(x-2)} dx = \mathbb{E}\left[\frac{1}{4} e^{-(X-4)^2/2} \sin(2\pi X)\right].$$

```
n = 1e+05
U = runif(n)
X = 2 - log(U)/4
I = mean(exp(-(X - 4)^2/2) * sin(2 * pi * X)/4)
print(I)
```

```
## [1] 0.01963348
```

Example 2.7. We want to approximate the following integral:

$$I = \int_{0}^{\infty} \int_{0}^{x} e^{-(x+y)} dy dx.$$

Consider the random variables X, Y with $X \sim \text{Exp}(1)$ and $(Y \mid X = x) \sim \text{Unif}[0, x]$. For $x > 0$ and $y \in [0, x]$, we observe that:

$$f_{X,Y}(x, y) = f_X(x) f_{Y|X}(y \mid x) = e^{-x} \frac{1}{x}.$$

Therefore, we calculate that

$$I = \int_{0}^{\infty} \int_{0}^{x} x e^{-y} \frac{e^{-x}}{x} dy dx = \mathbb{E}\left(X e^{-Y}\right).$$

```
n = 1e+05
U = runif(n)
```

```r
X = -log(U)
V = runif(n)
Y = X * V
I = mean(X * exp(-Y))
print(I)
```

```
## [1] 0.4984331
```

Note 2.1. We know that $\mathbb{P}(A) = \mathbb{E}(\mathbb{1}_A)$.

Example 2.8. We want to approximate the value of the constant π. Consider the independent random variables $U, V \sim \text{Unif}[0,1]$. Then, we calculate that:

$$
\begin{aligned}
\mathbb{P}\left(U^2 + V^2 \leqslant 1\right) = \mathbb{P}\left(U \leqslant \sqrt{1 - V^2}\right) &= \int_0^1 \int_0^{\sqrt{1-v^2}} 1 du dv \\
&= \int_0^1 \sqrt{1 - v^2} dv \overset{v = \sin x}{=} \int_0^{\pi/2} \sqrt{1 - \sin^2 x} \cos x dx = \int_0^{\pi/2} \cos^2 x dx \\
&= \int_0^{\pi/2} \frac{1 + \cos(2x)}{2} dx = \left[\frac{x}{2} + \frac{\sin(2x)}{4}\right]_{x=0}^{\pi/2} = \frac{\pi}{4}.
\end{aligned}
$$

Therefore, we observe that:

$$
\pi = 4\mathbb{P}\left(U^2 + V^2 \leqslant 1\right) = \mathbb{E}\left(4 \cdot \mathbb{1}_{\{U^2 + V^2 \leqslant 1\}}\right).
$$

```r
n = 1e+06
U = runif(n)
V = runif(n)
mean(4 * (U^2 + V^2 <= 1))
```

```
## [1] 3.141728
```

Example 2.9. Let $X_1, \ldots, X_k \sim \mathcal{N}\left(\mu, \sigma^2\right)$ be a random sample. We define:

$$
\overline{X} = \frac{1}{k} \sum_{j=1}^{k} X_j, \quad S^2 = \frac{1}{k-1} \sum_{j=1}^{k} \left(X_j - \overline{X}\right)^2.
$$

We know that the $100(1-\alpha)\%$ equal-tailed confidence interval for the parameter μ is equal to:

$$
I(X) = \left[\overline{X} - t_{k-1;\alpha/2} \frac{S}{\sqrt{k}}, \overline{X} + t_{k-1;\alpha/2} \frac{S}{\sqrt{k}}\right].
$$

Additionally, we know by construction that $\mathbb{P}[\mu \in I(X)] = 1 - \alpha$. We want to verify that the interval $I(X)$ has coverage $100(1-\alpha)\%$ for the parameter μ. We consider the random samples $X^{(1)}, \ldots, X^{(n)}$ following the $\mathcal{N}\left(\mu, \sigma^2\right)$ distribution and construct the corresponding confidence intervals $I^{(1)}, \ldots, I^{(n)}$. According to the strong law of large numbers, it must hold that:

$$
\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{\mu \in I^{(i)}\}} \overset{\text{a.s}}{\to} \mathbb{E}\left[\mathbb{1}_{\{\mu \in I(X)\}}\right] = \mathbb{P}[\mu \in I(X)] = 1 - \alpha.
$$

```r
n = 10000
k = 10
mu = 1
sigma = 2
alpha = 0.05
U = matrix(runif(k * n/2), n/2)
D = -2 * log(U)
V = matrix(runif(k * n/2), n/2)
Theta = 2 * pi * V
Z = rbind(sqrt(D) * cos(Theta), sqrt(D) * sin(Theta))
X = sigma * Z + mu
Xbar = rowMeans(X)
S = apply(X, 1, sd)
I = cbind(Xbar - qt(alpha/2, k - 1, lower.tail = FALSE) * S/sqrt(k), Xbar +
    qt(alpha/2, k - 1, lower.tail = FALSE) * S/sqrt(k))
100 * mean(I[, 1] <= mu & mu <= I[, 2])
```

```
## [1] 94.78
```

Example 2.10. Let $X_1, \ldots, X_k \sim \mathcal{N}\left(\mu, \sigma^2\right)$ be a random sample. We know that the statistic of the one-sided test of the hypotheses $H_0 : \mu = \mu_0$ vs. $H_1 : \mu < \mu_0$ is equal to:

$$T(X) = \frac{\overline{X} - \mu_0}{S/\sqrt{k}}.$$

Additionally, we know that $T(X) \sim t_{k-1}$ under the null hypothesis H_0. We define the p-value $p(X) = F_{t_{k-1}}(T(X))$. We reject H_0 at statistical significance level α if $T(X) < -t_{k-1;\alpha}$ or $p(X) < \alpha$. The type I error probability is equal to $\mathbb{P}_{\mu_0}\left[T(X) < -t_{k-1;\alpha}\right] = \alpha$ and the power is equal to $\beta(\mu) = \mathbb{P}_{\mu}\left[T(X) < -t_{k-1;\alpha}\right]$. We want to study the distribution of the statistic $T(X)$ and the p-value $p(X)$ under the hypotheses H_0 and H_1. We consider the random samples $X^{(1)}, \ldots, X^{(n)}$ following the distributions $\mathcal{N}\left(\mu, \sigma^2\right)$ and conduct the corresponding hypothesis tests. Furthermore, we want to verify that the test has type I error probability equal to α independent of n, k, μ and σ. Finally, we want to study the change in power of the test for different values of k, μ, σ and α.

Proposition 2.1. If the random variable X has absolutely continuous CDF F, then it holds that:

$$U = F(X) \sim \text{Unif}[0, 1].$$

Proof. We calculate that:

$$F_U(u) = \mathbb{P}\left[F(X) \leqslant u\right] = \mathbb{P}\left[X \leqslant F^{-1}(u)\right] = F\left(F^{-1}(u)\right) = u.$$

□

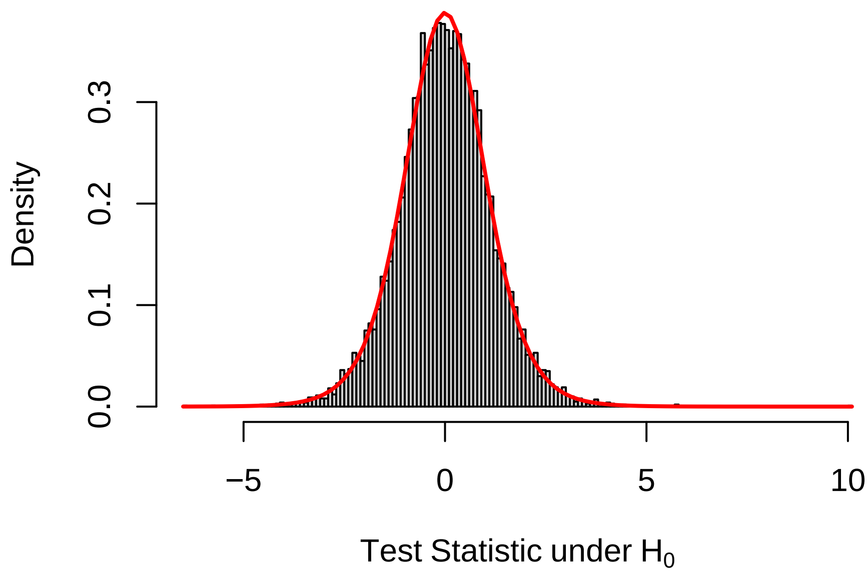
Corollary 2.1. It holds that $p(X) \sim \text{Unif}[0, 1]$ under the null hypothesis H_0.

Proof. The statistic $T(X)$ follows the t_{k-1} distribution under H_0. Therefore, we infer that:

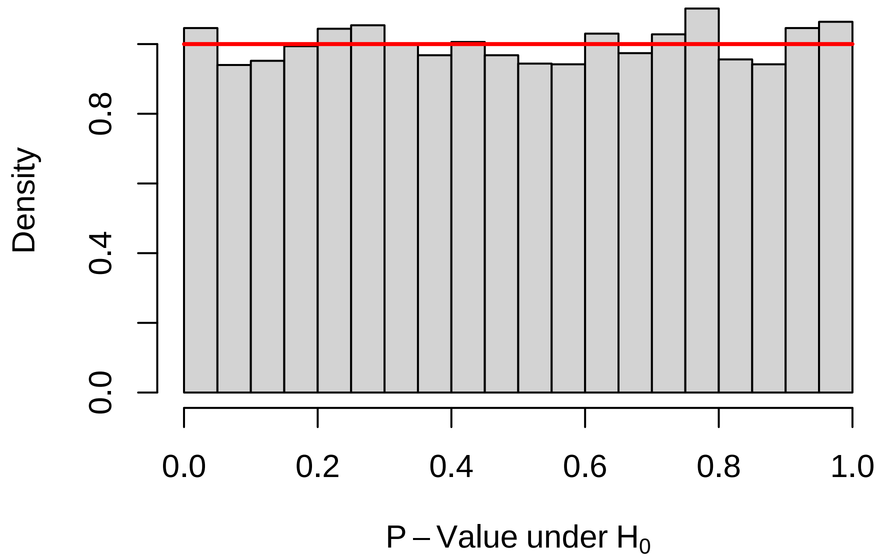
$$p(X) = F_{t_{k-1}}(T(X)) \sim \text{Unif}[0,1].$$

$\square$

```
n = 10000
k = 10
mu = 1
sigma = 2
mu0 = 1
alpha = 0.05
U = matrix(runif(k * n/2), n/2)
D = -2 * log(U)
V = matrix(runif(k * n/2), n/2)
Theta = 2 * pi * V
Z = rbind(sqrt(D) * cos(Theta), sqrt(D) * sin(Theta))
X = sigma * Z + mu
Xbar = rowMeans(X)
S = apply(X, 1, sd)
t = (Xbar - mu0) * sqrt(k)/S
hist(t, "FD", freq = FALSE, main = NA, xlab = expression(Test ~ Statistic ~
    under ~ H[0]))
curve(dt(x, k - 1), add = TRUE, col = "red", lwd = 2)
```



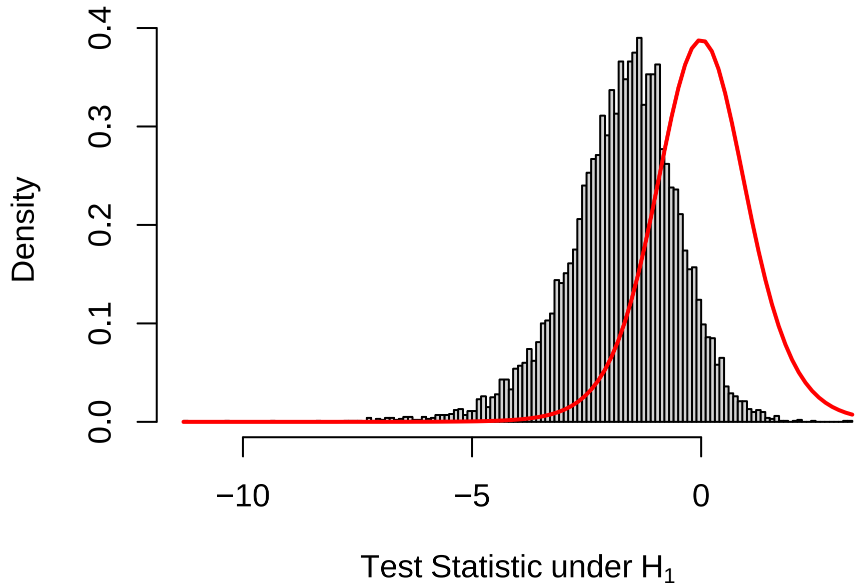

```
p = pt(t, k - 1)
hist(p, "FD", freq = FALSE, main = NA, xlim = c(0, 1), xlab = expression(P -
    Value ~ under ~ H[0]))
curve(dunif(x), add = TRUE, col = "red", lwd = 2)
```



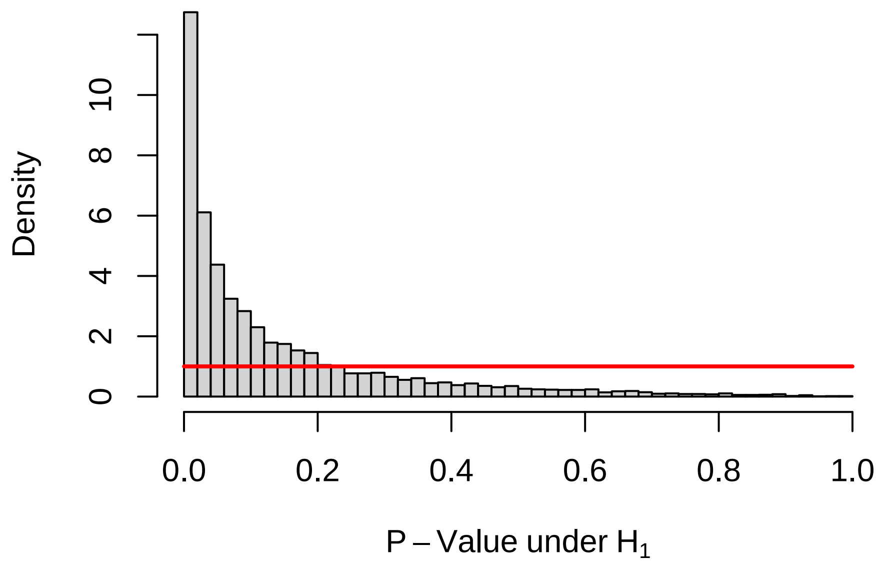

```
mean(t < qt(alpha, k - 1))
```

```
## [1] 0.0523
```

```
n = 10000
k = 10
mu = 0
sigma = 2
mu0 = 1
alpha = 0.05
U = matrix(runif(k * n/2), n/2)
D = -2 * log(U)
V = matrix(runif(k * n/2), n/2)
Theta = 2 * pi * V
Z = rbind(sqrt(D) * cos(Theta), sqrt(D) * sin(Theta))
X = sigma * Z + mu
Xbar = rowMeans(X)
S = apply(X, 1, sd)
t = (Xbar - mu0) * sqrt(k)/S
hist(t, "FD", freq = FALSE, main = NA, xlab = expression(Test ~ Statistic ~
    under ~ H[1]))
curve(dt(x, k - 1), add = TRUE, col = "red", lwd = 2)
```




```
p = pt(t, k - 1)
hist(p, "FD", freq = FALSE, main = NA, xlim = c(0, 1), xlab = expression(P -
    Value ~ under ~ H[1]))
curve(dunif(x), add = TRUE, col = "red", lwd = 2)
```

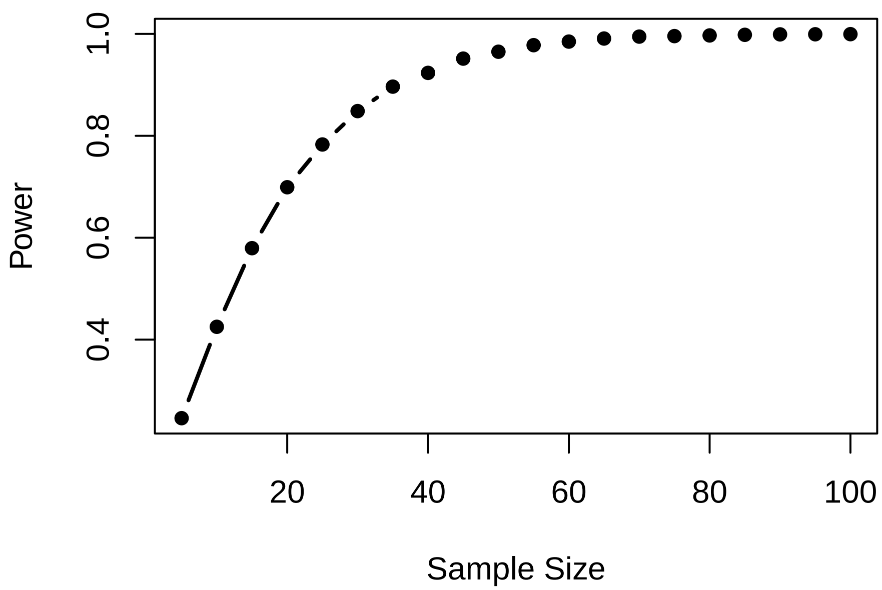



```
beta = mean(t < qt(alpha, k - 1))
print(beta)
```

```
## [1] 0.4231
```

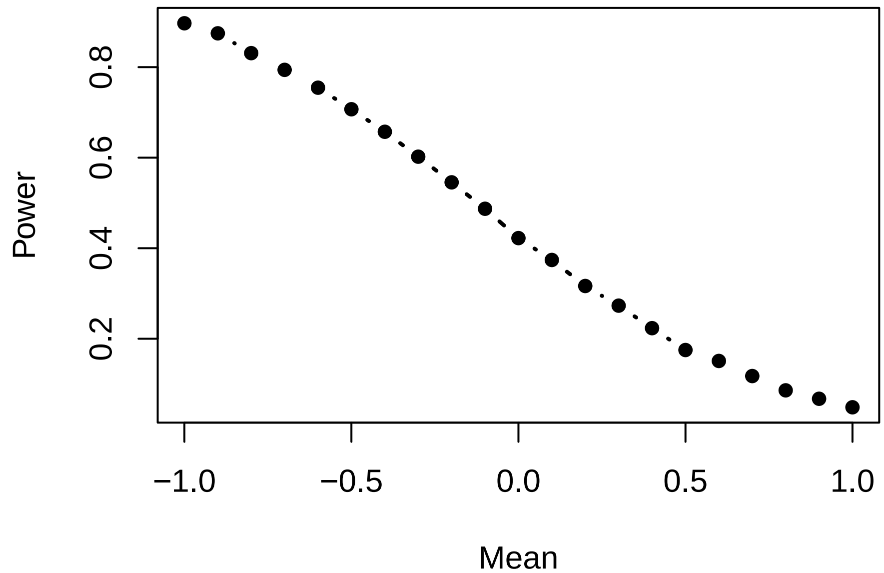
```
n = 10000
k = seq(5, 100, 5)
mu = 0
sigma = 2
mu0 = 1
alpha = 0.05
```

```
beta = numeric(length(k))
for (j in 1:length(k)) {
    U = matrix(runif(k[j] * n/2), n/2)
    D = -2 * log(U)
    V = matrix(runif(k[j] * n/2), n/2)
    Theta = 2 * pi * V
    Z = rbind(sqrt(D) * cos(Theta), sqrt(D) * sin(Theta))
    X = sigma * Z + mu
    Xbar = rowMeans(X)
    S = apply(X, 1, sd)
    t = (Xbar - mu0) * sqrt(k[j])/S
    beta[j] = mean(t < qt(alpha, k[j] - 1))
}
plot(k, beta, "b", xlab = "Sample Size", ylab = "Power", pch = 16, lwd = 2)
```

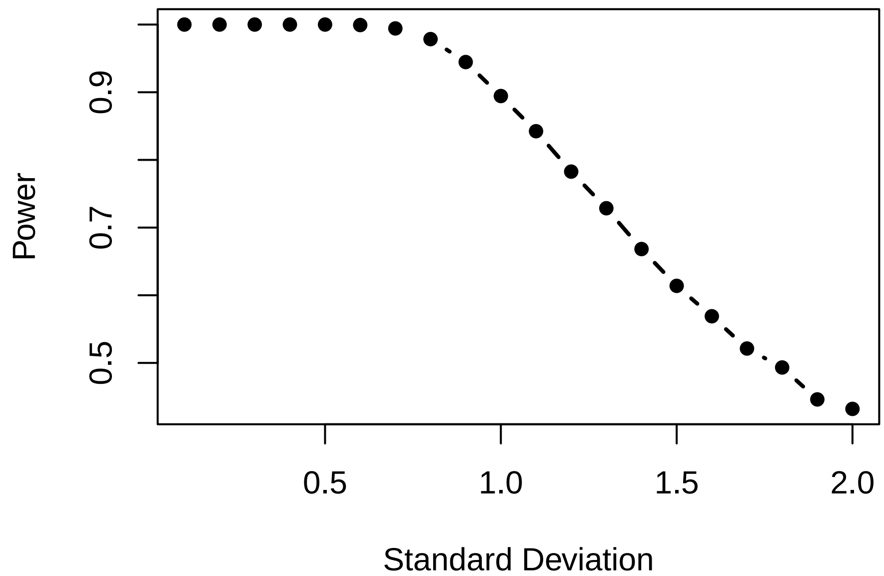



```
n = 10000
k = 10
mu = seq(-1, 1, 0.1)
sigma = 2
mu0 = 1
alpha = 0.05
beta = numeric(length(mu))
for (j in 1:length(mu)) {
    U = matrix(runif(k * n/2), n/2)
    D = -2 * log(U)
    V = matrix(runif(k * n/2), n/2)
    Theta = 2 * pi * V
    Z = rbind(sqrt(D) * cos(Theta), sqrt(D) * sin(Theta))
    X = sigma * Z + mu[j]
```

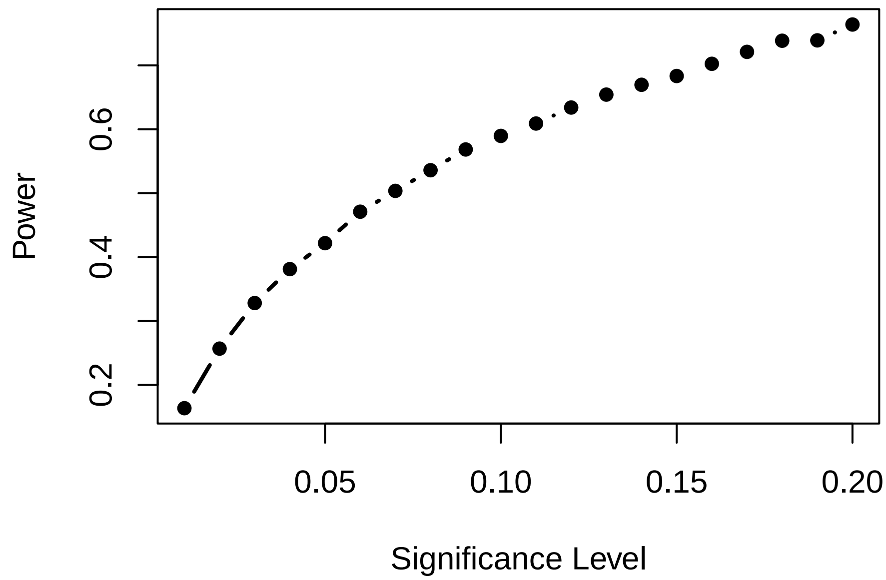
```
    Xbar = rowMeans(X)
    S = apply(X, 1, sd)
    t = (Xbar - mu0) * sqrt(k)/S
    beta[j] = mean(t < qt(alpha, k - 1))
}
plot(mu, beta, "b", xlab = "Mean", ylab = "Power", pch = 16, lwd = 2)
```




```
n = 10000
k = 10
mu = 0
sigma = seq(0.1, 2, 0.1)
mu0 = 1
alpha = 0.05
beta = numeric(length(sigma))
for (j in 1:length(sigma)) {
    U = matrix(runif(k * n/2), n/2)
    D = -2 * log(U)
    V = matrix(runif(k * n/2), n/2)
    Theta = 2 * pi * V
    Z = rbind(sqrt(D) * cos(Theta), sqrt(D) * sin(Theta))
    X = sigma[j] * Z + mu
    Xbar = rowMeans(X)
    S = apply(X, 1, sd)
    t = (Xbar - mu0) * sqrt(k)/S
    beta[j] = mean(t < qt(alpha, k - 1))
}
plot(sigma, beta, "b", xlab = "Standard Deviation", ylab = "Power", pch = 16,
    lwd = 2)
```




```
n = 10000
k = 10
mu = 0
sigma = 2
mu0 = 1
alpha = seq(0.01, 0.2, 0.01)
beta = numeric(length(alpha))
for (j in 1:length(alpha)) {
    U = matrix(runif(k * n/2), n/2)
    D = -2 * log(U)
    V = matrix(runif(k * n/2), n/2)
    Theta = 2 * pi * V
    Z = rbind(sqrt(D) * cos(Theta), sqrt(D) * sin(Theta))
    X = sigma * Z + mu
    Xbar = rowMeans(X)
    S = apply(X, 1, sd)
    t = (Xbar - mu0) * sqrt(k)/S
    beta[j] = mean(t < qt(alpha[j], k - 1))
}
plot(alpha, beta, "b", xlab = "Significance Level", ylab = "Power", pch = 16,
    lwd = 2)
```



We want to approximate the sum of the following series:

$$S = \sum_{j=0}^{\infty} a_j.$$

Consider the random variable X with PMF $p = [p_j]$. If we let $b_j = \frac{a_j}{p_j}$, then we observe that:

$$S = \sum_{j=0}^{\infty} p_j \frac{a_j}{p_j} = \mathbb{E}\left(b_X\right).$$

Let $X_1, X_2, \ldots, X_n$ be a random sample following the PMF $p = [p_j]$. According to the strong law of large numbers, we know that:

$$\frac{1}{n}\sum_{i=1}^{n} b_{X_i} \xrightarrow{\text{a.s.}} \mathbb{E}\left(b_X\right) = S.$$

Example 2.11. We want to approximate the sum of the following series:

$$S = \sum_{j=0}^{\infty} \frac{e^{-j^2}}{j!}.$$

Consider the random variable $X \sim \text{Poisson}(1)$ with PMF $p_j = e^{-1}/j!$ for $j = 0, 1, \ldots$. Then, we observe that:

$$S = \sum_{j=0}^{\infty} \frac{e^{-1}}{j!} e^{1-j^2} = \mathbb{E}\left(e^{1-X^2}\right).$$

```
n = 1e+05
X = numeric(n)
for (i in 1:n) {
    U = runif(1)
    pmf = exp(-1)
    cdf = pmf
```

```r
    while (U > cdf) {
        X[i] = X[i] + 1
        pmf = pmf/X[i]
        cdf = cdf + pmf
    }
}
I = mean(exp(1 - X^2))
print(I)
```

```
## [1] 1.37738
```

Lemma 2.1. Let X be a non-negative and discrete random variable. Then, it holds that:

$$\mathbb{E}(X) = \sum_{k=0}^{\infty} \mathbb{P}(X > k).$$

Proof. We observe that:

$$\mathbb{E}(X) = \sum_{j=0}^{\infty} j\mathbb{P}(X = j) = \sum_{j=0}^{\infty}\sum_{k=0}^{j-1} \mathbb{P}(X = j) = \sum_{k=0}^{\infty}\sum_{j=k+1}^{\infty} \mathbb{P}(X = j) = \sum_{k=0}^{\infty} \mathbb{P}(X > k).$$

□

Note 2.2. We know that $\text{Var}(X) = \mathbb{E}\left[(X - \mathbb{E}(X))^2\right]$.

Example 2.12. Let $X_1, X_2, \ldots$ be a sequence of non-negative and discrete random variables. We want to approximate the expected value and the variance of the following random variable:

$$N = \sup\left\{k \in \mathbb{N} : X_1 < X_2 < \cdots < X_{k-1}\right\}.$$

For $k \in \mathbb{N}$, we observe that:

$$\mathbb{P}(N > k) = \mathbb{P}(X_1 < X_2 < \cdots < X_k) = \frac{1}{k!}.$$

Therefore, we calculate that:

$$\mathbb{E}(N) = \sum_{k=0}^{\infty} \mathbb{P}(N > k) = \sum_{k=0}^{\infty} \frac{1}{k!} = e.$$

```r
n = 1e+05
N = numeric(n)
for (i in 1:n) {
    Uold = runif(1)
    Unew = runif(1)
    N[i] = 2
    while (Uold < Unew) {
        Uold = Unew
        Unew = runif(1)
        N[i] = N[i] + 1
```

```
    }
}
I = mean(N)
print(I)
```

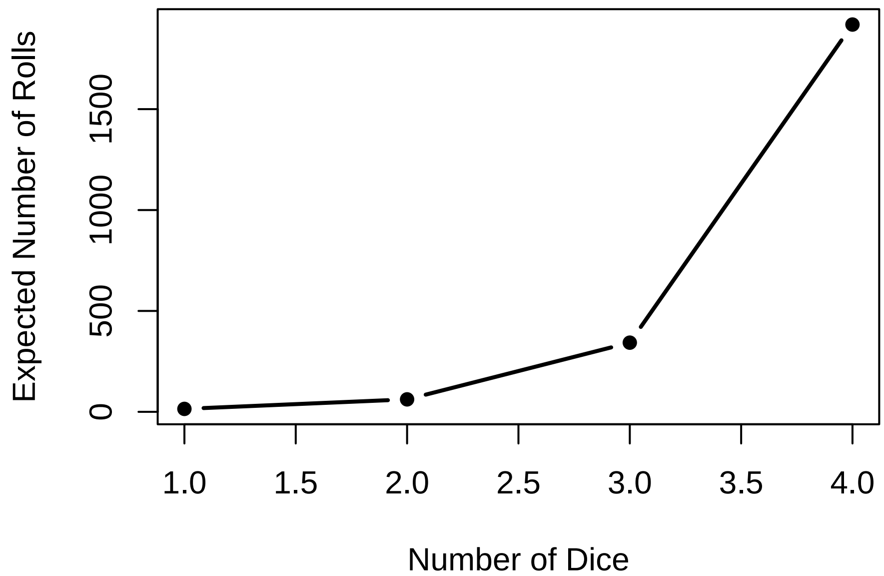
```
## [1] 2.71821
```

```
mean((N - I)^2)
```

```
## [1] 0.7690844
```

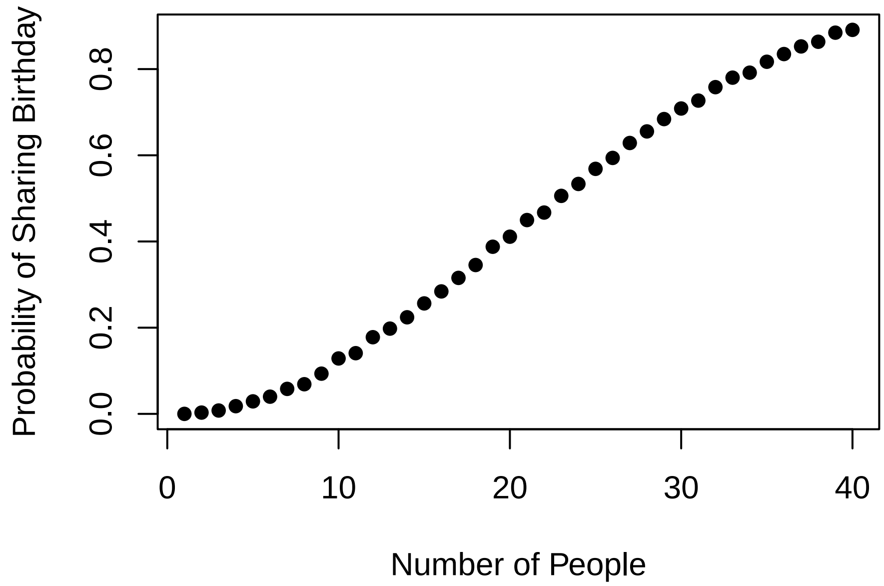
Example 2.13. We roll k fair dice and we want to estimate the expected minimum number of rolls until all of the possible sums of their faces appear as a function of k.

```
n = 1000
k = 1:4
I = numeric(length(k))
for (j in 1:length(k)) {
    N = numeric(n)
    for (i in 1:n) {
        S = numeric(6 * k[j])
        while (sum(S == 0) > k[j] - 1) {
            U = runif(k[j])
            Y = floor(6 * U) + 1
            X = sum(Y)
            S[X] = S[X] + 1
            N[i] = N[i] + 1
        }
    }
    I[j] = mean(N)
}
plot(k, I, "b", xlab = "Number of Dice", ylab = "Expected Number of Rolls",
    pch = 16, lwd = 2)
```



Example 2.14. We want to estimate the probability that at least 2 out of k people have their birthday on the same day of the year as a function of k.

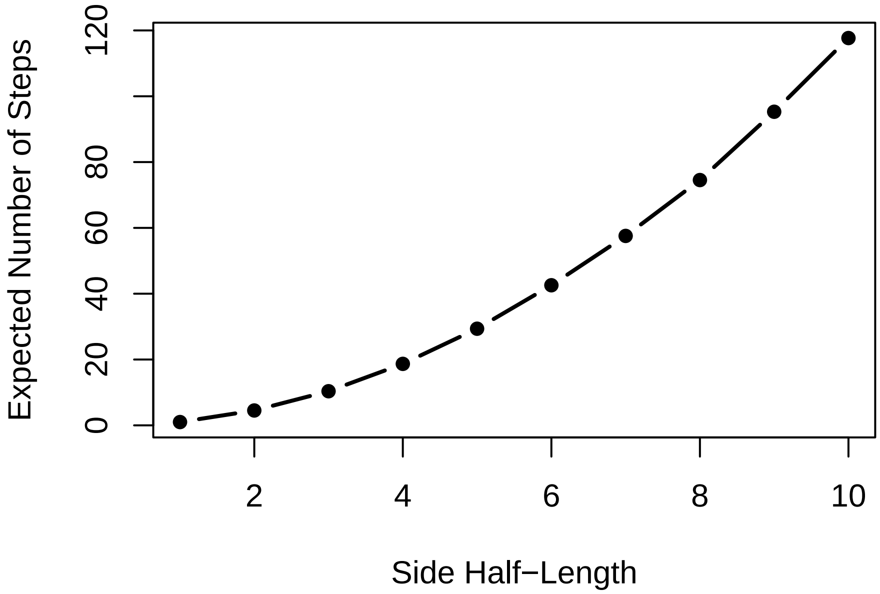
```
n = 10000
k = 1:40
I = numeric(length(k))
for (j in 1:length(k)) {
    found = logical(n)
    for (i in 1:n) {
        D = numeric(365)
        l = 0
        while (!found[i] && l < k[j]) {
            U = runif(1)
            X = floor(365 * U) + 1
            if (D[X] == 1) {
                found[i] = TRUE
            } else {
                D[X] = D[X] + 1
                l = l + 1
            }
        }
    }
    I[j] = mean(found)
}
plot(k, I, "b", xlab = "Number of People", ylab = "Probability of Sharing Birthday",
    pch = 16, lwd = 2)
```



Example 2.15. Consider a square of side $2k$ for $k \in \mathbb{N}$ centered on the axis origin. A body performs a random walk on the pairs of integers starting from the axis origin until it arrives at the border of the square. We want to estimate the expected number of steps it will take as a function of k.

```
n = 10000
k = 1:10
I = numeric(length(k))
for (j in 1:length(k)) {
    N = numeric(n)
    for (i in 1:n) {
        X = 0
        Y = 0
        while (abs(X) < k[j] && abs(Y) < k[j]) {
            U = runif(1)
            if (U <= 0.25) {
                X = X + 1
            } else if (U <= 0.5) {
                Y = Y + 1
            } else if (U <= 0.75) {
                X = X - 1
            } else {
                Y = Y - 1
            }
            N[i] = N[i] + 1
        }
    }
    I[j] = mean(N)
}
plot(k, I, "b", xlab = "Side Half-Length", ylab = "Expected Number of Steps",
```

```
    pch = 16, lwd = 2)
```



Expected Number of Steps
0
20
40
60
80
120
2
4
6
8
10
Side Half−Length

3 Discrete-Event Simulation

Consider a $M/M/1$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the $\text{Exp}(\mu)$ distribution. We want to simulate the state of the system over time.

Input: Arrival rate λ and service rate μ.

We let $Q \leftarrow 0$, $D \leftarrow \infty$, we simulate $A \sim \text{Exp}(\lambda)$ and we iterate the following steps:

1: We let $t \leftarrow \min\{A, D\}$.

2: If $t = A$, then:

i: We let $Q \leftarrow Q + 1$, we simulate $R \sim \text{Exp}(\lambda)$ and we let $A \leftarrow t + R$.

ii: If $Q = 1$, then we simulate $X \sim \text{Exp}(\mu)$ and we let $D \leftarrow t + X$.

If $t = D$, then:

i: We let $Q \leftarrow Q - 1$.

ii: If $Q > 0$, then we simulate $X \sim \text{Exp}(\mu)$ and we let $D \leftarrow t + X$. Otherwise, We let $D \leftarrow \infty$.

Example 3.1. Consider a $M/M/1$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the $\text{Exp}(\mu)$ distribution. The partial busy period of the system is defined as the time period between an arrival which finds the system empty until a departure which leaves the system again empty. We want to estimate the average duration of a partial busy period and the expected maximum number of customers present in the system during a partial busy period.

```
n = 10000
lambda = 4
mu = 6
Y = numeric(n)
M = numeric(n)
for (i in 1:n) {
    U = runif(1)
    A = -log(U)/lambda
    t = A
    Q = 1
    Y[i] = t
    M[i] = 1
    U = runif(1)
    A = t - log(U)/lambda
    V = runif(1)
    D = t - log(V)/mu
    while (Q > 0) {
        t = min(A, D)
        if (t == A) {
            Q = Q + 1
            U = runif(1)
```

```
            A = t - log(U)/lambda
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/mu
            }
            M[i] = max(M[i], Q)
        } else {
            Q = Q - 1
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/mu
            } else {
                D = Inf
            }
        }
    }
    Y[i] = t - Y[i]
}
mean(Y)
```

```
## [1] 0.4986261
```

```
mean(M)
```

```
## [1] 1.9508
```

Example 3.2. Consider a $M/E_s/1$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the Gamma(s, μ) distribution. Suppose that there exists some time point T^* after which no new arrivals in the system are allowed, but the server continues servicing all customers who were already present in the system before time T^*. We want to estimate the expected overtime put in by the server, the average sojourn time (total time a customer spends in the system) and the expected total idle time of the server.

```
n = 1000
lambda = 10
mu = 40
s = 3
Tstar = 100
S = numeric(n)
I = numeric(n)
O = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    t = A
    N = 0
```

```
    temp = 0
    arrivals = numeric(0)
    while (t < Tstar || Q > 0) {
        if (t == A) {
            Q = Q + 1
            U = runif(1)
            A = t - log(U)/lambda
            if (A > Tstar) {
                A = Inf
            }
            if (Q == 1) {
                V = runif(s)
                D = t - log(prod(V))/mu
                I[i] = I[i] + t - temp
            }
            N = N + 1
            arrivals = c(arrivals, t)
        } else {
            Q = Q - 1
            if (Q > 0) {
                V = runif(s)
                D = t - log(prod(V))/mu
            } else {
                D = Inf
                temp = t
            }
            S[i] = S[i] + t - arrivals[1]
            arrivals = arrivals[-1]
        }
        t = min(A, D)
    }
    S[i] = S[i]/N
    O[i] = max(temp - Tstar, 0)
}
mean(S)
```

```
## [1] 0.2221672
```

```
mean(O)
```

```
## [1] 0.1529004
```

```
mean(I)
```

```
## [1] 25.13479
```

Example 3.3. Consider a $M/M/1$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the Exp(μ) distribution. When the system becomes empty, the server goes on working vacation and returns back to the normal working period only if there are s customers in the system. We want to estimate the expected percentage of time until time point T^* when there are at least m customers in the system.

```
n = 1000
lambda = 4
mu = 6
s = 10
m = 12
Tstar = 100
P = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    D = Inf
    t = A
    vacation = TRUE
    while (t < Tstar) {
        if (t == A) {
            Q = Q + 1
            U = runif(1)
            A = t - log(U)/lambda
            if (Q == s && vacation) {
                V = runif(1)
                D = t - log(V)/mu
                vacation = FALSE
            }
            if (Q == m) {
                temp = t
            }
        } else {
            Q = Q - 1
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/mu
            } else {
                D = Inf
                vacation = TRUE
            }
            if (Q == m - 1) {
                P[i] = P[i] + t - temp
            }
```

```
        }
        t = min(A, D)
    }
    if (Q >= m) {
        P[i] = P[i] + Tstar - temp
    }
    P[i] = 100 * P[i]/Tstar
}
mean(P)
```

```
## [1] 8.75385
```

Example 3.4. Consider a $M/M/1$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the $\text{Exp}(\mu)$ distribution. Suppose that each customer waits in the queue for a time period which follows the $\text{Unif}[0, \nu]$ distribution before departing without getting serviced. We want to estimate the average number of lost customers until time T^*.

```
n = 1000
lambda = 5
mu = 4
nu = 5
Tstar = 100
L = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    t = A
    R = numeric(0)
    while (t < Tstar) {
        if (t == A) {
            Q = Q + 1
            U = runif(1)
            A = t - log(U)/lambda
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/mu
            } else {
                W = runif(1)
                R = c(R, t + nu * W)
            }
        } else if (t == D) {
            Q = Q - 1
            if (Q > 0) {
```

```
                V = runif(1)
                D = t - log(V)/mu
                R = R[-1]
            } else {
                D = Inf
            }
        } else {
            Q = Q - 1
            R = R[-which.min(R)]
            L[i] = L[i] + 1
        }
        t = min(A, D, R)
    }
}
mean(L)
```

```
## [1] 118.119
```

Example 3.5. Consider a $M/M/1$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the $\text{Exp}(\mu)$ distribution. Suppose that each customer waits in the queue for a time period which follows the $\text{Unif}[0,\nu]$ distribution before departing without getting serviced. Suppose also that, every time a service is completed, the customer with the shortest departure time from the system is chosen to be served next. We want to compare the average number of lost customers until time T^* with that of the previous example.

```
n = 1000
lambda = 5
mu = 4
nu = 5
Tstar = 100
L = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    t = A
    R = numeric(0)
    while (t < Tstar) {
        if (t == A) {
            Q = Q + 1
            U = runif(1)
            A = t - log(U)/lambda
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/mu
```

```
            } else {
                W = runif(1)
                R = c(R, t + nu * W)
            }
        } else if (t == D) {
            Q = Q - 1
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/mu
                R = R[-which.min(R)]
            } else {
                D = Inf
            }
        } else {
            Q = Q - 1
            R = R[-which.min(R)]
            L[i] = L[i] + 1
        }
        t = min(A, D, R)
    }
}
mean(L)
```

```
## [1] 100.265
```

Example 3.6. Consider a queuing network constituting of two serial $M/M/1$ queuing systems, where the arrival process at the first system is Poisson with rate λ and the services times at system j follow the $\text{Exp}(\mu_j)$ distribution. Suppose that there exists some time point T^* after which no new arrivals in the system are allowed, but the servers continue servicing all customers already present in the network before time T^*. We want to estimate the average sojourn time of a customer within each of the two systems.

```
n = 1000
lambda = 4
mu1 = 5
mu2 = 6
Tstar = 100
S1 = numeric(n)
S2 = numeric(n)
for (i in 1:n) {
    Q1 = 0
    Q2 = 0
    U = runif(1)
    A = -log(U)/lambda
    D2 = Inf
```

```r
    t = A
    N = 0
    A1 = numeric(0)
    A2 = numeric(0)
    while (t < Tstar || Q1 > 0 || Q2 > 0) {
        if (t == A) {
            Q1 = Q1 + 1
            U = runif(1)
            A = t - log(U)/lambda
            if (A > Tstar) {
                A = Inf
            }
            if (Q1 == 1) {
                V = runif(1)
                D1 = t - log(V)/mu1
            }
            N = N + 1
            A1 = c(A1, t)
        } else if (t == D1) {
            Q1 = Q1 - 1
            Q2 = Q2 + 1
            if (Q1 > 0) {
                V = runif(1)
                D1 = t - log(V)/mu1
            } else {
                D1 = Inf
            }
            if (Q2 == 1) {
                W = runif(1)
                D2 = t - log(W)/mu2
            }
            A2 = c(A2, t)
            S1[i] = S1[i] + t - A1[1]
            A1 = A1[-1]
        } else {
            Q2 = Q2 - 1
            if (Q2 > 0) {
                W = runif(1)
                D2 = t - log(W)/mu2
            } else {
                D2 = Inf
            }
            S2[i] = S2[i] + t - A2[1]
```

```
            A2 = A2[-1]
        }
        t = min(A, D1, D2)
    }
    S1[i] = S1[i]/N
    S2[i] = S2[i]/N
}
mean(S1)
```

```
## [1] 0.9374927
```

```
mean(S2)
```

```
## [1] 0.4806224
```

Example 3.7. Consider a $M/M/c$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the Exp(μ) distribution. We want to estimate the average sojourn time of a customer in the system and the expected percentage of the first N^* services which are performed by each of the servers.

```
n = 1000
c = 2
lambda = 6
mu = c(4, 3)
Nstar = 1000
S = numeric(n)
P = matrix(0, n, c)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    D = rep(Inf, c)
    N = 0
    arrivals = numeric(0)
    while (N < Nstar || Q > 0) {
        t = min(A, D)
        if (t == A) {
            Q = Q + 1
            N = N + 1
            if (N < Nstar) {
                U = runif(1)
                A = t - log(U)/lambda
            } else {
                A = Inf
            }
            if (Q <= c) {
                I = match(Inf, D)
```

```
                V = runif(1)
                D[I] = t - log(V)/mu[I]
                S[i] = S[i] + D[I] - t
            } else {
                arrivals = c(arrivals, t)
            }
        } else {
            Q = Q - 1
            I = which.min(D)
            if (Q >= c) {
                V = runif(1)
                D[I] = t - log(V)/mu[I]
                S[i] = S[i] + D[I] - arrivals[1]
                arrivals = arrivals[-1]
            } else {
                D[I] = Inf
            }
            P[i, I] = P[i, I] + 1
        }
    }
    S[i] = S[i]/Nstar
    P[i, ] = P[i, ]/Nstar
}
mean(S)
```

```
## [1] 1.021141
```

```
colMeans(P)
```

```
## [1] 0.58169 0.41831
```

Example 3.8. Consider a $M/M/c$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the $\text{Exp}(\mu)$ distribution. Suppose that there exists some time point T^* when the system breaks down and the customers who haven't already finished getting serviced are lost. We want to estimate the average sojourn time in the system of a customer who has finished getting serviced before time point T^*, the average number of lost customers and the probability of losing more than 2 customers.

```
n = 1000
c = 2
lambda = 6
mu = c(4, 3)
Tstar = 100
S = numeric(n)
Q = numeric(n)
for (i in 1:n) {
    U = runif(1)
```

```
    A = -log(U)/lambda
    D = rep(Inf, c)
    t = A
    N = 0
    arrivals = numeric(0)
    while (t < Tstar) {
        if (t == A) {
            Q[i] = Q[i] + 1
            U = runif(1)
            A = t - log(U)/lambda
            if (Q[i] <= c) {
                I = match(Inf, D)
                V = runif(1)
                D[I] = t - log(V)/mu[I]
                if (D[I] < Tstar) {
                  S[i] = S[i] + D[I] - t
                }
            } else {
                arrivals = c(arrivals, t)
            }
        } else {
            Q[i] = Q[i] - 1
            I = which.min(D)
            if (Q[i] >= c) {
                V = runif(1)
                D[I] = t - log(V)/mu[I]
                if (D[I] < Tstar) {
                  S[i] = S[i] + D[I] - arrivals[1]
                }
                arrivals = arrivals[-1]
            } else {
                D[I] = Inf
            }
            N = N + 1
        }
        t = min(A, D)
    }
    S[i] = S[i]/N
}
mean(S)
```

```
## [1] 0.9902442
```

```r
mean(Q)
```

```
## [1] 6.151
```

```r
mean(Q > 2)
```

```
## [1] 0.655
```

Example 3.9. Consider a queuing network consisting of two parallel $M/M/1$ queuing systems, where the arrival process at the network is Poisson with rate λ and the service times at system j follow the $\text{Exp}(\mu_j)$ distribution. Suppose that an arriving customer enters the system with the shortest queue or the first system if both have the same number of customers. We want to estimate the average sojourn time of a customer in the system and the expected percentage of the first N^* customers who enter the first system.

```r
n = 1000
lambda = 6
mu1 = 4
mu2 = 3
Nstar = 1000
S = numeric(n)
P = numeric(n)
for (i in 1:n) {
    Q1 = 0
    Q2 = 0
    U = runif(1)
    A = -log(U)/lambda
    D1 = Inf
    D2 = Inf
    N = 0
    A1 = numeric(0)
    A2 = numeric(0)
    while (N < Nstar || Q1 > 0 || Q2 > 0) {
        t = min(A, D1, D2)
        if (t == A) {
            if (Q1 <= Q2) {
                Q1 = Q1 + 1
                if (Q1 == 1) {
                  V = runif(1)
                  D1 = t - log(V)/mu1
                }
                A1 = c(A1, t)
            } else {
                Q2 = Q2 + 1
                if (Q2 == 1) {
                  W = runif(1)
                  D2 = t - log(W)/mu2
```

```
            }
            A2 = c(A2, t)
        }
        N = N + 1
        if (N < Nstar) {
            U = runif(1)
            A = t - log(U)/lambda
        } else {
            A = Inf
        }
    } else if (t == D1) {
        Q1 = Q1 - 1
        if (Q1 > 0) {
            V = runif(1)
            D1 = t - log(V)/mu1
        } else {
            D1 = Inf
        }
        P[i] = P[i] + 1
        S[i] = S[i] + t - A1[1]
        A1 = A1[-1]
    } else {
        Q2 = Q2 - 1
        if (Q2 > 0) {
            W = runif(1)
            D2 = t - log(W)/mu2
        } else {
            D2 = Inf
        }
        S[i] = S[i] + t - A2[1]
        A2 = A2[-1]
    }
  }
  S[i] = S[i]/Nstar
  P[i] = P[i]/Nstar
}
mean(S)
```

```
## [1] 1.072401
```

```
mean(P)
```

```
## [1] 0.583024
```

Example 3.10. Consider a queuing network consisting of two parallel $M/M/1$ queuing systems, where the arrival process at the network is Poisson with rate λ and the service times at system j follow the $\text{Exp}(\mu_j)$ distribution. Suppose that an arriving customer enters the first system with probability p, where p is the estimated percentage of the first N^* customers who enter the first system of the previous example. We want to compare the average sojourn time of a customer in the system with that of the previous example.

```
n = 1000
lambda = 6
mu1 = 4
mu2 = 3
Nstar = 1000
p = mean(P)
S = numeric(n)
for (i in 1:n) {
    Q1 = 0
    Q2 = 0
    U = runif(1)
    A = -log(U)/lambda
    D1 = Inf
    D2 = Inf
    N = 0
    A1 = numeric(0)
    A2 = numeric(0)
    while (N < Nstar || Q1 > 0 || Q2 > 0) {
        t = min(A, D1, D2)
        if (t == A) {
            U = runif(1)
            if (U < p) {
                Q1 = Q1 + 1
                if (Q1 == 1) {
                  V = runif(1)
                  D1 = t - log(V)/mu1
                }
                A1 = c(A1, t)
            } else {
                Q2 = Q2 + 1
                if (Q2 == 1) {
                  W = runif(1)
                  D2 = t - log(W)/mu2
                }
                A2 = c(A2, t)
            }
            N = N + 1
            if (N < Nstar) {
                U = runif(1)
```

```
            A = t - log(U)/lambda
        } else {
            A = Inf
        }
    } else if (t == D1) {
        Q1 = Q1 - 1
        if (Q1 > 0) {
            V = runif(1)
            D1 = t - log(V)/mu1
        } else {
            D1 = Inf
        }
        S[i] = S[i] + t - A1[1]
        A1 = A1[-1]
    } else {
        Q2 = Q2 - 1
        if (Q2 > 0) {
            W = runif(1)
            D2 = t - log(W)/mu2
        } else {
            D2 = Inf
        }
        S[i] = S[i] + t - A2[1]
        A2 = A2[-1]
    }
  }
  S[i] = S[i]/Nstar
}
mean(S)
```

```
## [1] 1.827888
```

Example 3.11. Consider a system which requires N machines for it to function. Suppose that there exist s backup machines in the system. Every machine functions for a time period which follows the $\text{Exp}(\lambda)$ distribution before breaking down. Whenever a machine breaks down, it's immediately replaced by a backup and it's sent to the repair shop. A repairman fixes the machines in time which follows the $\text{Exp}(\mu)$ distribution. As soon as a machine gets fixed, it becomes available as a backup for whenever it might be needed. The system stops working when a machine breaks down and there's no backup to replace it. We want to estimate the expected amount of time until the system stops working.

```
n = 1e+05
lambda = 1
mu = 2
s = 3
```

```r
N = 4
C = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(N)
    A = -log(U)/lambda
    D = Inf
    while (Q <= s) {
        t = min(A, D)
        if (t == D) {
            Q = Q - 1
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/mu
            } else {
                D = Inf
            }
        } else {
            Q = Q + 1
            U = runif(1)
            A[which.min(A)] = t - log(U)/lambda
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/mu
            }
        }
    }
    C[i] = t
}
mean(C)
```

```
## [1] 1.536251
```

Lemma 3.1. Let $X \sim \text{Exp}(\lambda)$ and $Y \sim \text{Exp}(\mu)$ be independent random variables. Then, we infer that:

$$W = \min\{X, Y\} \sim \text{Exp}(\lambda + \mu).$$

Proof. For $w > 0$, we calculate that:

$$\begin{aligned} F_W(w) &= \mathbb{P}(\min\{X, Y\} \leqslant w) = 1 - \mathbb{P}(\min\{X, Y\} > w) = 1 - \mathbb{P}(X > w, Y > w) \\ &= 1 - \mathbb{P}(X > w)\mathbb{P}(Y > w) = 1 - [1 - F_X(w)]\,[1 - F_Y(w)] = 1 - e^{-\lambda w}e^{-\mu w} = 1 - e^{-(\lambda+\mu)w}. \end{aligned}$$

□

Corollary 3.1. The time until 1 out of the N functioning machines breaks down follows the $\text{Exp}(N\lambda)$ distribution.

```r
C = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/(N * lambda)
    D = Inf
    while (Q <= s) {
        t = min(A, D)
        if (t == A) {
            Q = Q + 1
            U = runif(1)
            A = t - log(U)/(N * lambda)
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/mu
            }
        } else {
            Q = Q - 1
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/mu
            } else {
                D = Inf
            }
        }
    }
    C[i] = t
}
mean(C)
```

```
## [1] 1.531271
```

Example 3.12. Suppose that messages arrive at a communications facility according to a Poisson process with rate λ. The facility has c communication channels. If all of the channels are busy at the arrival time of a new message, then the message is lost. The weather is initially nice and alternates between nice and bad periods lasting s_1 and s_2 hours respectively. If the weather is nice at the arrival time of a new message, then the time required for its decoding follows the Beta$(\mu_1, 1)$ distribution, otherwise it follows the Beta$(\mu_2, 1)$ distribution. We want to estimate the average number of lost messages up to time T^*.

```r
n = 1000
c = 3
lambda = 2
mu = c(1, 3)
s = c(2, 1)
```

```r
Tstar = 100
L = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    D = rep(Inf, c)
    t = A
    while (t < Tstar) {
        if (t == A) {
            U = runif(1)
            A = t - log(U)/lambda
            if (Q < c) {
                Q = Q + 1
                V = runif(1)
                if (t%%sum(s) <= s[1]) {
                  D[match(Inf, D)] = t + V^(1/mu[1])
                } else {
                  D[match(Inf, D)] = t + V^(1/mu[2])
                }
            } else {
                L[i] = L[i] + 1
            }
        } else {
            Q = Q - 1
            D[which.min(D)] = Inf
        }
        t = min(A, D)
    }
}
mean(L)
```

```
## [1] 17.229
```

4 Variance Reduction Techniques

Antithetic Variables

Let X_1, X_2, X, Y be identically distributed random variables with expected value μ and variance σ^2. Suppose that the random variables X_1, X_2 are independent, whereas the random variables X, Y have covariance $\mathrm{Cov}(X, Y) = \sigma_{XY}$ and Pearson correlation coefficient $\mathrm{Corr}(X, Y) = \rho_{XY}$. Then, we observe that:

$$\mathbb{E}\left(\frac{X_1 + X_2}{2}\right) = \mathbb{E}\left(\frac{X + Y}{2}\right) = \mu,$$

$$\mathrm{Var}\left(\frac{X_1 + X_2}{2}\right) = \frac{\mathrm{Var}(X_1) + \mathrm{Var}(X_2)}{4} = \frac{\sigma^2}{2},$$

$$\mathrm{Var}\left(\frac{X + Y}{2}\right) = \frac{\mathrm{Var}(X) + \mathrm{Var}(Y) + 2\mathrm{Cov}(X, Y)}{4} = \frac{\sigma^2 + \sigma_{XY}}{2},$$

$$\mathrm{Var}\left(\frac{X + Y}{2}\right) < \mathrm{Var}\left(\frac{X_1 + X_2}{2}\right) \quad \Leftrightarrow \quad \sigma_{XY} < 0.$$

The percentage of variance reduction for an estimator of μ by use of the antithetic variables method is equal to:

$$100 \cdot \frac{\sigma^2 - \left(\sigma^2 + \sigma_{XY}\right)}{\sigma^2} = 100 \cdot \frac{|\sigma_{XY}|}{\sigma^2} = 100 \cdot |\rho_{XY}|\,.$$

Proposition 4.1. Let $U_1, U_2, \ldots, U_k \sim \mathrm{Unif}[0,1]$ be independent random variables. Suppose that $h : [0,1]^k \to \mathbb{R}$ is a monotone function of each of its arguments. Then, it holds that:

$$\mathrm{Cov}\left[h\left(U_1, \ldots, U_k\right), h\left(1 - U_1, \ldots, 1 - U_k\right)\right] \leqslant 0.$$

Corollary 4.1. Let $U, U_1, U_2 \sim \mathrm{Unif}[0,1]$ be independent random variables. Consider a monotone function $h : [0,1] \to \mathbb{R}$. Then, it holds that:

$$\mathbb{E}\left[\frac{h(U) + h(1 - U)}{2}\right] = \mathbb{E}\left[\frac{h(U_1) + h(U_2)}{2}\right],$$

$$\mathrm{Var}\left[\frac{h(U) + h(1 - U)}{2}\right] \leqslant \mathrm{Var}\left[\frac{h(U_1) + h(U_2)}{2}\right].$$

Example 4.1. Let $U \sim \mathrm{Unif}[0,1]$ be a random variable. We want to estimate the expected value $\mathbb{E}\left(e^U\right)$. We observe that the function $h(x) = e^x$ is increasing for $x \in [0,1]$.

```r
n = 1e+05
U = runif(n)
X = exp(U)
I = mean(X)
print(I)
```

```
## [1] 1.718251
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.2427706
```

```
U = runif(n/2)
X = exp(U)
Y = exp(1 - U)
W = (X + Y)/2
I = mean(W)
print(I)
```

```
## [1] 1.718183
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.003916482
```

```
rho = (2 * VarW - VarX)/VarX
print(rho)
```

```
## [1] -0.9677351
```

```
100 * abs(rho)
```

```
## [1] 96.77351
```

Example 4.2. Let $U, V \sim \mathrm{Unif}[0,1]$ be independent random variables. We want to estimate the expected value $\mathbb{E}\left[e^{(U+V)^2}\right]$. We observe that $h(x,y) = e^{(x+y)^2}$ is an increasing function of each of its arguments for $x, y \in [0,1]$.

```
n = 1e+05
U = runif(n)
V = runif(n)
X = exp((U + V)^2)
I = mean(X)
print(I)
```

```
## [1] 4.886297
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 35.24747
```

```
U = runif(n/2)
V = runif(n/2)
X = exp((U + V)^2)
Y = exp((2 - U - V)^2)
W = (X + Y)/2
```

```
I = mean(W)
print(I)
```

```
## [1] 4.897734
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 11.51644
```

```
rho = (2 * VarW - VarX)/VarX
print(rho)
```

```
## [1] -0.3465383
```

```
100 * abs(rho)
```

```
## [1] 34.65383
```

Example 4.3. Let $U_1, U_2, \cdots \sim \text{Unif}[0,1]$ be a sequence of independent random variables. We want to estimate the expected value of the following random variable:

$$X = \sup\left\{k \in \mathbb{N} : U_1 < U_2 < \cdots < U_{k-1}\right\}.$$

We define the following random variable:

$$Y = \sup\left\{k \in \mathbb{N} : 1 - U_1 < 1 - U_2 < \cdots < 1 - U_{k-1}\right\} = \sup\left\{k \in \mathbb{N} : U_1 > U_2 > \cdots > U_{k-1}\right\}.$$

```
n = 1e+06
X = numeric(n)
for (i in 1:n) {
    Uold = runif(1)
    Unew = runif(1)
    X[i] = 2
    while (Uold < Unew) {
        Uold = Unew
        Unew = runif(1)
        X[i] = X[i] + 1
    }
}
I = mean(X)
print(I)
```

```
## [1] 2.717766
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.765044
```

```r
X = numeric(n/2)
Y = numeric(n/2)
for (i in 1:(n/2)) {
    Uold = runif(1)
    Unew = runif(1)
    X[i] = 2
    Y[i] = 2
    if (Uold < Unew) {
        while (Uold < Unew) {
            Uold = Unew
            Unew = runif(1)
            X[i] = X[i] + 1
        }
    } else {
        while (Uold > Unew) {
            Uold = Unew
            Unew = runif(1)
            Y[i] = Y[i] + 1
        }
    }
}
W = (X + Y)/2
I = mean(W)
print(I)
```

```
## [1] 2.718524
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.1254453
```

```r
rho = (2 * VarW - VarX)/VarX
print(rho)
```

```
## [1] -0.6720574
```

```r
100 * abs(rho)
```

```
## [1] 67.20574
```

Note 4.1. Let $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$ be a random variable. Then, the random variable $Y = 2\mu - X$ is identically distributed and negatively correlated with X.

Example 4.4. Let $Z \sim \mathcal{N}(0, 1)$ be a random variable. We want to estimate the random variable $\mathbb{E}\left(e^Z\right)$.

```r
n = 1e+05
U = runif(n/2)
```

```r
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
X = exp(Z)
I = mean(X)
print(I)
```

```
## [1] 1.651902
```

```r
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 4.685197
```

```r
U = runif(n/4)
D = -2 * log(U)
V = runif(n/4)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
X = exp(Z)
Y = exp(-Z)
W = (X + Y)/2
I = mean(W)
print(I)
```

```
## [1] 1.654055
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 1.504112
```

```r
rho = (2 * VarW - VarX)/VarX
print(rho)
```

```
## [1] -0.3579303
```

```r
100 * abs(rho)
```

```
## [1] 35.79303
```

Control Variables

Consider 2 random variables X and Y with $\mathbb{E}(X) = \mu$, $\mathbb{E}(Y) = \mu_Y$, $\mathrm{Var}(X) = \sigma_X^2$, $\mathrm{Var}(Y) = \sigma_Y^2$, $\mathrm{Cov}(X, Y) = \sigma_{XY}$ and $\mathrm{Corr}(X, Y) = \rho_{XY}$. We observe that the random variable $W_c = X + c\,(Y - \mu_Y)$ also has expected value μ for every $c \in \mathbb{R}$. We calculate that:

$$\mathrm{Var}\,(W_c) = \mathrm{Var}(X) + c^2 \mathrm{Var}\,(Y - \mu_Y) + 2c\mathrm{Cov}\,(X, Y - \mu_Y) = \sigma_Y^2 c^2 + 2\sigma_{XY} c + \sigma_X^2.$$

Since $\sigma_Y^2 > 0$, we know that the function $\operatorname{Var}(W_c)$ has a unique global minimum at:

$$c^* = -\frac{\sigma_{XY}}{\sigma_Y^2}.$$

We observe that:

$$\operatorname{Var}(W_{c^*}) = \frac{\sigma_{XY}^2}{\sigma_Y^2} - 2\frac{\sigma_{XY}^2}{\sigma_Y^2} + \sigma_X^2 = \sigma_X^2 - \frac{\sigma_{XY}^2}{\sigma_Y^2} = \sigma_X^2\left(1 - \frac{\sigma_{XY}^2}{\sigma_X^2\sigma_Y^2}\right) = \sigma_X^2\left(1 - \rho_{XY}^2\right) \leqslant \sigma_X^2.$$

Therefore, the percentage of variance reduction for an estimator of μ by use of the control variable method is equal to:

$$100 \cdot \frac{\sigma_X^2 - \sigma_X^2\left(1 - \rho_{XY}^2\right)}{\sigma_X^2} = 100 \cdot \rho_{XY}^2.$$

If X and Y are uncorrelated, then $\operatorname{Var}(W_{c^*}) = \sigma_X^2$, i.e. no variance reduction may be achieved for this specific choice of control variable Y. It's usually not possible to directly calculate the quantities σ_{XY} and σ_Y^2, so they're estimated from the simulated sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ as follows:

$$\widehat{\sigma}_{XY} = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right), \quad \widehat{\sigma}_Y^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2.$$

Example 4.5. Let $U \sim \text{Unif}[0,1]$ be a random variable. We want to estimate the expected value $\mathbb{E}\left(\sqrt{1-U^2}\right)$. We will first use $Y = U$ as a control variable with $\mathbb{E}(Y) = 0.5$.

```
n = 1e+05
U = runif(n)
X = sqrt(1 - U^2)
I = mean(X)
print(I)
```

```
## [1] 0.78518
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.05000468
```

```
Y = U
muY = 0.5
VarY = var(Y)
Cov = cov(X, Y)
c = -Cov/VarY
W = X + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.7850612
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.007591752
```

```
rho = Cov/sqrt(VarX * VarY)
print(rho)
```

```
## [1] -0.920971
```

```
100 * rho^2
```

```
## [1] 84.81877
```

We will then use $Y = U^2$ as a control variable with:

$$\mathbb{E}(Y) = \text{Var}(U) + [\mathbb{E}(U)]^2 = \frac{1}{12} + \frac{1}{4} = \frac{1}{3}.$$

```
Y = U^2
muY = 1/3
VarY = var(Y)
Cov = cov(X, Y)
c = -Cov/VarY
W = X + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.7852937
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.001647797
```

```
rho = Cov/sqrt(VarX * VarY)
print(rho)
```

```
## [1] -0.9833905
```

```
100 * rho^2
```

```
## [1] 96.70568
```

Example 4.6. Let $S \sim \text{Gamma}(2, 1)$ be a random variable. We want to estimate the probability $\mathbb{P}(S^2 \leqslant 4)$. We will first use $Y = S$ as a control variable with $\mathbb{E}(Y) = 2$.

```
n = 1e+05
U = matrix(runif(2 * n), n)
R = -log(U)
S = rowSums(R)
X = S^2 <= 4
I = mean(X)
print(I)
```

```
## [1] 0.59344
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.241269
```

```
Y = S
muY = 2
VarY = var(Y)
Cov = cov(X, Y)
c = -Cov/VarY
W = X + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.5937954
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.09494908
```

```
rho = Cov/sqrt(VarX * VarY)
print(rho)
```

```
## [1] -0.7787591
```

```
100 * rho^2
```

```
## [1] 60.64657
```

We will then use $Y = S^2$ as a control variable with:

$$\mathbb{E}(Y) = \mathrm{Var}(S) + [\mathbb{E}(S)]^2 = 2 + 4 = 6.$$

```
Y = S^2
muY = 6
VarY = var(Y)
Cov = cov(X, Y)
c = -Cov/VarY
W = X + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.5936546
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.1553603
```

```
rho = Cov/sqrt(VarX * VarY)
print(rho)
```

```
## [1] -0.5967191
```

```
100 * rho^2
```

```
## [1] 35.60737
```

Conditioning Method

Let X, Y be two random variables with $\mathbb{E}(X) = \mu$ and $\text{Var}(X) = \sigma^2$. According to the law of iterated expectations, we know that $\mathbb{E}(X) = \mathbb{E}\left[\mathbb{E}\left(X \mid Y\right)\right]$, i.e. the random variable $W = \mathbb{E}\left(X \mid Y\right)$ also has expected value μ. According to the law of total variance, we know that:

$$\text{Var}(X) = \text{Var}\left[\mathbb{E}\left(X \mid Y\right)\right] + \mathbb{E}\left[\text{Var}\left(X \mid Y\right)\right] \quad \Rightarrow \quad \text{Var}\left[\mathbb{E}\left(X \mid Y\right)\right] \leqslant \sigma^2.$$

The percentage of variance reduction for an estimator of μ by use of the conditioning method is equal to:

$$100 \cdot \frac{\text{Var}(X) - \text{Var}\left[\mathbb{E}\left(X \mid Y\right)\right]}{\text{Var}(X)} = 100 \cdot \frac{\mathbb{E}\left[\text{Var}\left(X \mid Y\right)\right]}{\sigma^2}.$$

Note 4.2. We know that $\text{Var}\left(X \mid Y\right) \equiv 0$ if and only if $X = g(Y)$ for some measurable function g. In this case, we observe that $\text{Var}\left[\mathbb{E}\left(X \mid Y\right)\right] = \sigma^2$, i.e. no variance reduction may be achieved for this specific choice of conditioning variable Y.

Example 4.7. Let $Y \sim \text{Exp}(1)$ and $(S \mid Y) \sim \mathcal{N}(Y, 4)$ be two random variables. We want to estimate the probability $\mathbb{P}(S > 1)$. For $y > 0$, we calculate that:

$$\mathbb{P}(S > 1 \mid Y = y) = \mathbb{P}\left(\frac{S-y}{2} > \frac{1-y}{2} \middle| Y = y\right) = 1 - \Phi\left(\frac{1-y}{2}\right).$$

According to the law of iterated expectations, we infer that:

$$\mathbb{P}(S > 1) = \mathbb{E}\left(\mathbb{1}_{\{S>1\}}\right) = \mathbb{E}\left[\mathbb{E}\left(\mathbb{1}_{\{S>1\}} \middle| Y\right)\right] = \mathbb{E}\left[\mathbb{P}\left(S > 1 \mid Y\right)\right] = \mathbb{E}\left[1 - \Phi\left(\frac{1-Y}{2}\right)\right].$$

```
n = 1e+05
U = runif(n)
Y = -log(U)
U = runif(n/2)
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
S = 2 * Z + Y
X = S > 1
I = mean(X)
```

```
print(I)
```

```
## [1] 0.49067
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.249913
```

```
W = 1 - pnorm((1 - Y)/2)
I = mean(W)
print(I)
```

```
## [1] 0.4905546
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.02691248
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 89.23126
```

We will then use Y as a control variable with $\mathbb{E}(Y) = 1$.

```
muY = 1
VarY = var(Y)
Cov = cov(W, Y)
c = -Cov/VarY
W = W + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.4900535
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.001347862
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 99.46067
```

Example 4.8. We want to approximate the value of the constant π. Consider two independent random variables $U, V \sim \mathrm{Unif}[0,1]$. Then, we know that:

$$\pi = 4\mathbb{P}\left(U^2 + V^2 \leqslant 1\right) = \mathbb{E}\left(4 \cdot \mathbb{1}_{\{U^2+V^2 \leqslant 1\}}\right).$$

For $u \in [0,1]$, we calculate that:

$$\mathbb{P}\left(U^2+V^2 \leqslant 1 \middle| U=u\right) = \mathbb{P}\left(V \leqslant \sqrt{1-u^2} \middle| U=u\right) = \mathbb{P}\left(V \leqslant \sqrt{1-u^2}\right) = \sqrt{1-u^2}.$$

According to the law of iterated expectations, we infer that:

$$\pi = \mathbb{E}\left[\mathbb{E}\left(4 \cdot \mathbb{1}_{\{U^2+V^2 \leqslant 1\}} \middle| U\right)\right] = \mathbb{E}\left[4\mathbb{P}\left(U^2+V^2 \leqslant 1 \middle| U\right)\right] = \mathbb{E}\left(4\sqrt{1-U^2}\right).$$

```
n = 1e+06
U = runif(n)
V = runif(n)
X = 4 * (U^2 + V^2 <= 1)
I = mean(X)
print(I)
```

```
## [1] 3.141728
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 2.696457
```

```
W = 4 * sqrt(1 - U^2)
I = mean(W)
print(I)
```

```
## [1] 3.141884
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.7960046
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 70.47961
```

We will then use $Y = U$ as a control variable with $\mathbb{E}(Y) = 0.5$.

```
Y = U
muY = 0.5
VarY = var(Y)
Cov = cov(W, Y)
c = -Cov/VarY
W = W + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 3.141869
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.1205411
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 95.52965
```

Example 4.9. Let $R \sim \text{Exp}(1)$ and $S \sim \text{Exp}(0.5)$ be two independent random variables. We want to estimate the probability $\mathbb{P}(R + S > 4)$. For $s > 0$, we calculate that:

$$\mathbb{P}(R + S > 4 \mid S = s) = \mathbb{P}(R > 4 - s \mid S = s) = \mathbb{P}(R > 4 - s) = \begin{cases} e^{-(4-s)}, & s \leqslant 4 \\ 1, & s > 4 \end{cases}.$$

According to the law of iterated expectations, we infer that:

$$\mathbb{P}(R + S > 4) = \mathbb{E}\left(\mathbb{1}_{\{R+S>4\}}\right) = \mathbb{E}\left[\mathbb{E}\left(\mathbb{1}_{\{R+S>4\}} \middle| S\right)\right] = \mathbb{E}\left[\mathbb{P}\left(R + S > 4 \mid S\right)\right] = \mathbb{E}\left[\min\left\{e^{-(4-S)}, 1\right\}\right].$$

```
n = 1e+05
U = runif(n)
R = -log(U)
V = runif(n)
S = -2 * log(V)
X = R + S > 4
I = mean(X)
print(I)
```

```
## [1] 0.253
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.188991
```

```
W = pmin(exp(-(4 - S)), 1)
I = mean(W)
print(I)
```

```
## [1] 0.2518087
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.11645
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 38.38331
```

We will then use $Y = S$ as a control variable with $\mathbb{E}(Y) = 2$.

```
Y = S
muY = 2
VarY = var(Y)
Cov = cov(W, Y)
c = -Cov/VarY
W = W + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.2523751
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.02201785
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 88.34979
```

Alternatively, we calculate that $\mathbb{P}(R+S>4) = \mathbb{E}\left[\mathbb{P}\left(R+S>4 \mid R\right)\right] = \mathbb{E}\left[\min\left\{e^{-(4-R)/2}, 1\right\}\right]$.

```
W = pmin(exp(-(4 - R)/2), 1)
I = mean(W)
print(I)
```

```
## [1] 0.2530393
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.02831303
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 85.01885
```

We will then use $Y = R$ as a control variable with $\mathbb{E}(Y) = 1$.

```
Y = R
muY = 1
VarY = var(Y)
Cov = cov(W, Y)
c = -Cov/VarY
W = W + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.2525319
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.002096755
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 98.89055
```

Example 4.10. Let $R, S \sim \mathrm{Bin}(k, p)$ be two independent random variables. We want to estimate the expected value $\mathbb{E}\left(e^{RS}\right)$. For $r \in \{0, 1, \ldots, k\}$, we calculate that:

$$\mathbb{E}\left(e^{RS}\middle|\, R = r\right) = \mathbb{E}\left(e^{rS}\middle|\, R = r\right) = \mathbb{E}\left(e^{rS}\right) = \sum_{s=0}^{k} \binom{k}{s} p^s (1-p)^{k-s} e^{rs}$$
$$= \sum_{s=0}^{k} \binom{k}{s} \left(pe^r\right)^s (1-p)^{k-s} = \left(pe^r + 1 - p\right)^k .$$

According to the law of iterated expectations, we infer that:

$$\mathbb{E}\left(e^{RS}\right) = \mathbb{E}\left[\mathbb{E}\left(e^{RS}\middle|\, R\right)\right] = \mathbb{E}\left[\left(pe^R + 1 - p\right)^k\right].$$

```r
n = 1e+05
k = 2
p = 0.1
U = matrix(runif(n * k), n)
R = rowSums(U < p)
V = matrix(runif(n * k), n)
S = rowSums(V < p)
X = exp(R * S)
I = mean(X)
print(I)
```

```
## [1] 1.081712
```

```r
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.5129571
```

```r
W = (p * exp(R) + 1 - p)^k
I = mean(W)
print(I)
```

```
## [1] 1.084086
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.04611129
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 91.01069
```

We will then use $Y = R$ as a control variable with $\mathbb{E}(Y) = kp$.

```r
Y = R
muY = k * p
VarY = var(Y)
Cov = cov(W, Y)
c = -Cov/VarY
W = W + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 1.083844
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.007070317
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 98.62166
```

Lemma 4.1. Let $S \sim \text{Gamma}(k, \lambda)$ be a random variable with $k \in \mathbb{N}$. For $s > 0$, we know that:

$$F_S(s) = 1 - \sum_{j=0}^{k-1} e^{-\lambda s} \frac{(\lambda s)^j}{j!}.$$

Proof. Let $\{N(t) : t \geqslant 0\}$ be a Poisson process with rate λ and arrival times $S_1, S_2, \ldots$. Then, we know that $S_k \sim \text{Gamma}(k, \lambda)$. We observe that:

$$F_{S_k}(s) = \mathbb{P}\left(S_k \leqslant s\right) = \mathbb{P}\left[N(s) \geqslant k\right] = 1 - \mathbb{P}\left[N(s) \leqslant k-1\right] = 1 - \sum_{j=0}^{k-1} \mathbb{P}\left[N(s) = j\right] = 1 - \sum_{j=0}^{k-1} e^{-\lambda s} \frac{(\lambda s)^j}{j!}.$$

Example 4.11. Let $K \sim \text{Poisson}(\lambda)$ be a random variable. Consider a sequence of independent random variables $R_1, R_2, \cdots \sim \text{Exp}(\mu)$ which is independent of K. We want to estimate the probability $\mathbb{P}\left(S_K > s\right)$, where:

$$S_K = \sum_{\ell=1}^{K} R_\ell.$$

For $k \in \mathbb{N}$, we observe that $S_k \sim \text{Gamma}(k, \mu)$. Therefore, we calculate that:

$$\mathbb{P}\left(S_K > s \mid K = k\right) = \mathbb{P}\left(S_k > s \mid K = k\right) = \mathbb{P}\left(S_k > s\right) = \sum_{j=0}^{k-1} e^{-\mu s} \frac{(\mu s)^j}{j!}.$$

According to the law of iterated expectations, we infer that:

$$\mathbb{P}(S_K > s) = \mathbb{E}\left(\mathbb{1}_{\{S_K>s\}}\right) = \mathbb{E}\left[\mathbb{E}\left(\mathbb{1}_{\{S_K>s\}}\middle| K\right)\right] = \mathbb{E}\left[\mathbb{P}(S_K > s \mid K)\right] = \mathbb{E}\left[\sum_{j=0}^{K-1} e^{-\mu s}\frac{(\mu s)^j}{j!}\right].$$

```
n = 1e+05
lambda = 4
mu = 6
s = 1
K = numeric(n)
S = numeric(n)
for (i in 1:n) {
    U = runif(1)
    pmf = exp(-lambda)
    cdf = pmf
    while (U > cdf) {
        K[i] = K[i] + 1
        pmf = pmf * lambda/K[i]
        cdf = cdf + pmf
    }
    V = runif(K[i])
    R = -log(V)/mu
    S[i] = sum(R)
}
X = S > s
I = mean(X)
print(I)
```

```
## [1] 0.21204
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.167079
```

```
W = numeric(n)
for (i in 1:n) {
    if (K[i] > 0) {
        pmf = exp(-mu * s)
        for (j in 0:(K[i] - 1)) {
            W[i] = W[i] + pmf
            pmf = pmf * mu * s/(j + 1)
        }
    }
}
```

```
I = mean(W)
print(I)
```

```
## [1] 0.2116497
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.04840946
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 71.02601
```

We will then use $Y = K$ as a control variable with $\mathbb{E}(Y) = \lambda$.

```
Y = K
muY = lambda
VarY = var(Y)
Cov = cov(W, Y)
c = -Cov/VarY
W = W + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.2125292
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.003967733
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 97.62524
```

Alternatively, we define $M = \min\{m \in \mathbb{N} : S_m > s\}$. For $m \in \mathbb{N}$, we calculate that:

$$\begin{aligned}\mathbb{P}(S_K > s \mid M = m) &= \mathbb{P}(K \geqslant M \mid M = m) = \mathbb{P}(K \geqslant m \mid M = m) \\ &= 1 - \mathbb{P}(K \leqslant m-1) = 1 - \sum_{j=0}^{m-1} e^{-\lambda}\frac{\lambda^j}{j!}.\end{aligned}$$

According to the law of iterated expectations, we infer that:

$$\mathbb{P}(S_K > s) = \mathbb{E}\left(\mathbb{1}_{\{S_K>s\}}\right) = \mathbb{E}\left[\mathbb{E}\left(\mathbb{1}_{\{S_K>s\}}\middle| M\right)\right] = \mathbb{E}\left[\mathbb{P}(S_K > s \mid M)\right] = \mathbb{E}\left(1 - \sum_{j=0}^{M-1} e^{-\lambda}\frac{\lambda^j}{j!}\right).$$

```
n = 1e+05
lambda = 4
mu = 6
```

```
s = 1
S = numeric(n)
M = numeric(n)
W = numeric(n)
for (i in 1:n) {
    while (S[i] <= s) {
        U = runif(1)
        R = -log(U)/mu
        S[i] = S[i] + R
        M[i] = M[i] + 1
    }
    W[i] = 1
    pmf = exp(-lambda)
    for (j in 0:(M[i] - 1)) {
        W[i] = W[i] - pmf
        pmf = pmf * lambda/(j + 1)
    }
}
I = mean(W)
print(I)
```

```
## [1] 0.2129754
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.05250765
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 68.57317
```

We will then use $Y = S_M - M/\mu$ as a control variable. For $m \in \mathbb{N}$, we calculate that:

$$\mathbb{E}\left(Y \mid M = m\right) = \mathbb{E}\left(S_m - \frac{m}{\mu} \middle| M = m\right) = \mathbb{E}\left(S_m\right) - \frac{m}{\mu} = 0.$$

According to the law of iterated expectations, we infer that $\mathbb{E}(Y) = \mathbb{E}\left[\mathbb{E}\left(Y \mid M\right)\right] = 0$.

```
Y = S - M/mu
muY = 0
VarY = var(Y)
Cov = cov(W, Y)
c = -Cov/VarY
W = W + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 0.2126705
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.01807493
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 89.18181
```

Example 4.12. Consider a $M/M/1/k$ queuing system, where the arrival process $\{N(t) : t \geqslant 0\}$ is Poisson with rate λ and the service times follow the $\text{Exp}(\mu)$ distribution. We want to estimate the average number X of lost customers up to time T^*. We let S be the total time that the system is full up to time point T^*. Then, we observe that $X \stackrel{d}{=} N(S)$. According to the law of iterated expectations, we infer that:

$$\mathbb{E}(X) = \mathbb{E}\left[N(S)\right] = \mathbb{E}\left[\mathbb{E}\left(N(S) \mid S\right)\right] = \mathbb{E}\left(\lambda S\right).$$

```
n = 1000
lambda = 4
mu = 6
k = 10
Tstar = 100
X = numeric(n)
S = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    t = A
    while (t < Tstar) {
        if (t == A) {
            if (Q < k) {
                Q = Q + 1
                if (Q == k) {
                  S[i] = S[i] - t
                }
            } else {
                X[i] = X[i] + 1
            }
            U = runif(1)
            A = t - log(U)/lambda
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/mu
            }
```

```
        } else {
            Q = Q - 1
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/mu
            } else {
                D = Inf
            }
            if (Q == k - 1) {
                S[i] = S[i] + t
            }
        }
        t = min(A, D)
    }
    if (Q == k) {
        S[i] = S[i] + Tstar
    }
}
I = mean(X)
print(I)
```

```
## [1] 2.221
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 9.140159
```

```
W = lambda * S
I = mean(W)
print(I)
```

```
## [1] 2.231415
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 7.261257
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 20.55656
```

Example 4.13. Consider a $M/M/1$ queuing system, where the arrival process is Poisson with rate λ and the service times follow the Exp(μ) distribution. We want to estimate the expected total sojourn time of the first N^* customers in the system. Let S_j be the sojourn time of the j-th customer, R_j be the service time of the j-th customer and M_j be the number of customers present in the system at the arrival moment of the j-th customer.

Then, we define:

$$X = \sum_{j=1}^{N^*} S_j, \quad Y = \sum_{j=1}^{N^*} R_j, \quad K = \sum_{j=1}^{N^*} M_j.$$

We will first use Y as a control variable with $\mathbb{E}(Y) = N^*/\mu$.

```
n = 10000
lambda = 4
mu = 6
Nstar = 10
X = numeric(n)
Y = numeric(n)
K = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    D = Inf
    N = 0
    arrivals = numeric(0)
    while (N < Nstar || Q > 0) {
        t = min(A, D)
        if (t == A) {
            K[i] = K[i] + Q
            Q = Q + 1
            N = N + 1
            if (N < Nstar) {
                U = runif(1)
                A = t - log(U)/lambda
            } else {
                A = Inf
            }
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/mu
                Y[i] = Y[i] + D - t
            }
            arrivals = c(arrivals, t)
        } else {
            Q = Q - 1
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/mu
                Y[i] = Y[i] + D - t
```

```
            } else {
                D = Inf
            }
            X[i] = X[i] + t - arrivals[1]
            arrivals = arrivals[-1]
        }
    }
}
I = mean(X)
print(I)
```

```
## [1] 3.204967
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 4.15866
```

```
muY = Nstar/mu
VarY = var(Y)
Cov = cov(X, Y)
c = -Cov/VarY
W = X + c * (Y - muY)
I = mean(W)
print(I)
```

```
## [1] 3.212926
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 1.587515
```

```
rho = Cov/sqrt(VarX * VarY)
print(rho)
```

```
## [1] 0.7863362
```

```
100 * rho^2
```

```
## [1] 61.83246
```

According to the law of iterated expectations, we alternatively calculate that:

$$\mathbb{E}(X) = \sum_{j=1}^{N^*} \mathbb{E}\left(S_j\right) = \sum_{j=1}^{N^*} \mathbb{E}\left[\mathbb{E}\left(S_j \mid M_j\right)\right] = \sum_{j=1}^{N^*} \mathbb{E}\left(\frac{M_j+1}{\mu}\right) = \mathbb{E}\left(\frac{1}{\mu}\sum_{j=1}^{N^*} M_j + \frac{N^*}{\mu}\right) = \mathbb{E}\left(\frac{K+N^*}{\mu}\right).$$

```
W = (K + Nstar)/mu
I = mean(W)
```

```r
print(I)
```

```
## [1] 3.2149
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 1.452951
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 65.06203
```

Importance Sampling

Let R, Y be random variables with PDFs $f(x)$, $g(x)$ and supports S_f, S_g respectively. Consider a function $h : S_f \to \mathbb{R}$. Suppose it holds that either $S_g \subseteq S_f$ and $h(x) = 0$ for $x \in S_f \setminus S_g$ or $S_f \subseteq S_g$. We want to estimate the expected value $\mathbb{E}[h(R)]$. If we let $\phi(x) = \frac{h(x)f(x)}{g(x)}$, then we observe that:

$$\mathbb{E}[h(R)] = \int_{S_f} h(x)f(x)dx = \int_{S_g} \frac{h(x)f(x)}{g(x)} g(x)dx = \int_{S_g} \phi(x)g(x)dx = \mathbb{E}[\phi(Y)].$$

The aim of the importance sampling method is to select a random variable Y such that $f(x) \gg g(x)$ if and only if $|h(x)| \approx 0$ and $f(x) \ll g(x)$ if and only if $|h(x)| \gg 0$. In this way, we manage to minimize the variance of the random variable $\phi(Y)$.

Example 4.14. Let $Z \sim \mathcal{N}(0, 1)$ be a random variable. We want to estimate the probability $\mathbb{P}(Z > 3)$. We will use $g(x) = e^{-(x-3)}$ as an importance variable for $x > 3$. Then, we calculate that:

$$\phi(x) = \frac{h(x)f(x)}{g(x)} = \frac{1}{e^{-(x-3)}} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} = \frac{e^{-x^2/2+x-3}}{\sqrt{2\pi}}.$$

```r
n = 1e+05
U = runif(n/2)
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
X = Z > 3
I = mean(X)
print(I)
```

```
## [1] 0.0014
```

```r
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.00139804
```

```r
U = runif(n)
Y = 3 - log(U)
W = exp(-Y^2/2 + Y - 3)/sqrt(2 * pi)
I = mean(W)
print(I)
```

```
## [1] 0.001350963
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 1.850184e-06
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 99.86766
```

Example 4.15. Let $S \sim \text{Gamma}(3,1)$ be a random variable. We want to estimate the expected value $\mathbb{E}\left(\max\{X-8,0\}\right)$. We will use $g(x) = e^{-(x-8)}$ as an importance variable for $x > 8$. Then, we calculate that:

$$\phi(x) = \frac{h(x)f(x)}{g(x)} = \frac{x-8}{e^{-(x-8)}}\frac{1}{\Gamma(3)}x^2e^{-x} = \frac{x^2(x-8)}{2e^8}.$$

```r
n = 1e+05
U = matrix(runif(3 * n), n)
R = -log(U)
S = rowSums(R)
X = pmax(S - 8, 0)
I = mean(X)
print(I)
```

```
## [1] 0.01686379
```

```r
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.04255399
```

```r
V = runif(n)
Y = 8 - log(V)
W = Y^2 * (Y - 8)/(2 * exp(8))
I = mean(W)
print(I)
```

```
## [1] 0.01721288
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.0006926254
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 98.37236
```

Example 4.16. Let $U \sim \text{Unif}[0,1]$ be a random variable. We want to estimate the expected value $\mathbb{E}\left[\left(1-U^2\right)e^U\right]$. We will first use $Y \sim \text{Beta}(2,1)$ as an importance variable. For $x \in [0,1]$, we calculate that:

$$\phi(x) = \frac{h(x)f(x)}{g(x)} = \frac{\left(1-x^2\right)e^x}{2x}.$$

```
n = 1e+05
U = runif(n)
X = (1 - U^2) * exp(U)
I = mean(X)
print(I)
```

```
## [1] 0.9993458
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.09743515
```

```
V = runif(n)
Y = V^(1/2)
W = (1 - Y^2) * exp(Y)/(2 * Y)
I = mean(W)
print(I)
```

```
## [1] 1.007208
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 3.706268
```

```
100 * (VarW - VarX)/VarX
```

```
## [1] 3703.831
```

We observe that the variance of the estimator of $\mathbb{E}\left[\left(1-U^2\right)e^U\right]$ is hugely increased for this specific choice of importance variable. For $x \in [0,1]$, we calculate that $h'(x) = \left(1-2x-x^2\right)e^x$ and $h''(x) = -\left(x^2+4x+1\right)e^x < 0$. In other words, the function h is maximized at $x^* = \sqrt{2}-1$. We know that the PDF of the $\text{Beta}(a,b)$ distribution for $a, b > 1$ is maximized at:

$$x^* = \frac{a-1}{a+b-2}.$$

If we select $a = \sqrt{2}$ and $b = 3-\sqrt{2}$, then the functions $h(x)$ and $g(x)$ are maximized at the same point. We will then use $Y \sim \text{Beta}\left(\sqrt{2}, 3-\sqrt{2}\right)$ as an importance variable. For $x \in [0,1]$, we calculate that:

$$\phi(x) = \frac{h(x)f(x)}{g(x)} = \left(1-x^2\right)e^x \frac{\Gamma\left(\sqrt{2}\right)\Gamma\left(3-\sqrt{2}\right)}{2x^{\sqrt{2}-1}(1-x)^{2-\sqrt{2}}}.$$

```r
M = dbeta(sqrt(2) - 1, sqrt(2), 3 - sqrt(2))
Y = numeric(n)
for (i in 1:n) {
    Y[i] = runif(1)
    U = runif(1)
    V = M * U
    while (dbeta(Y[i], sqrt(2), 3 - sqrt(2)) < V) {
        Y[i] = runif(1)
        U = runif(1)
        V = M * U
    }
}
W = (1 - Y^2) * exp(Y)/dbeta(Y, sqrt(2), 3 - sqrt(2))
I = mean(W)
print(I)
```

```
## [1] 1.000158
```

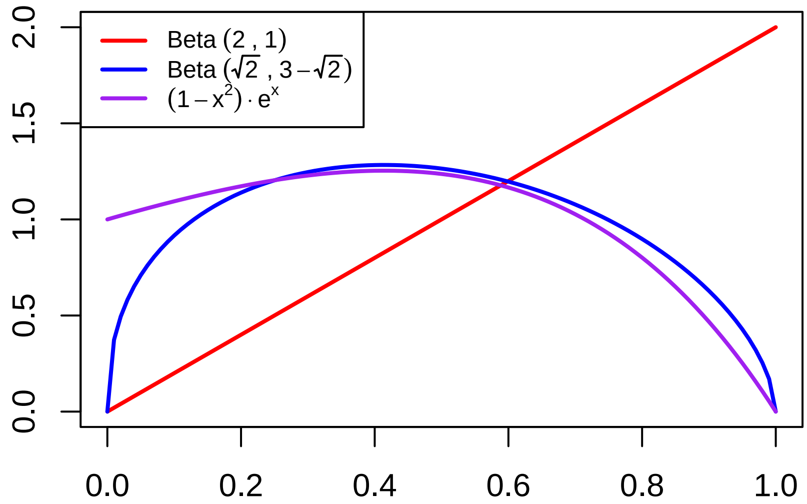
```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.05403956
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 44.53792
```

```r
curve(dbeta(x, 2, 1), col = "red", lwd = 2, xlab = NA, ylab = NA, xlim = c(0,
    1))
curve(dbeta(x, sqrt(2), 3 - sqrt(2)), add = TRUE, col = "blue", lwd = 2)
curve((1 - x^2) * exp(x), add = TRUE, col = "purple", lwd = 2)
legend("topleft", c(expression("Beta" ~ (2 ~ "," ~ 1)), expression("Beta" ~
    (sqrt(2) ~ "," ~ 3 - sqrt(2))), expression((1 - x^2) %.% e^x)), col = c("red",
    "blue", "purple"), lty = c(1, 1, 1), lwd = c(2, 2, 2), cex = 0.75)
```



Note 4.3. Let $X \sim t_\nu$ be a random variable. For $x \in \mathbb{R}$, we know that:

$$f_X(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}.$$

For $k \in \mathbb{N}$, we know that:

$$\Gamma\left(k + \frac{1}{2}\right) = \frac{(2k)!}{4^k k!}\sqrt{\pi}.$$

For $\nu = 1$, we know that $X \sim t_1 \equiv \text{Cauchy}(0, 1)$. Then, we infer that:

$$f_X(x) = \frac{1}{\pi\left(1 + x^2\right)}, \quad F_X(x) = \frac{1}{\pi}\arctan x + \frac{1}{2}, \quad F_X^{-1}(u) = \tan\left[\pi\left(u - \frac{1}{2}\right)\right].$$

Example 4.17. Let $S \sim t_3$ be a random variable. We want to estimate the expected value $\mathbb{E}\left(|S|\right)$. We will first use $Y \sim \text{Cauchy}(0, 1)$ as an importance variable. For $x \in \mathbb{R}$, we calculate that:

$$\phi(x) = \frac{h(x)f(x)}{g(x)} = |x|\pi\left(1 + x^2\right)\frac{1}{\sqrt{3\pi}\sqrt{\pi}/2}\left(1 + \frac{x^2}{3}\right)^{-2} = \frac{2|x|\left(1 + x^2\right)}{\sqrt{3}}\left(1 + \frac{x^2}{3}\right)^{-2}.$$

```
n = 1e+05
M = (3/2)^(3/2) * 2/sqrt(pi) * exp(-0.5)
Y = numeric(n)
for (i in 1:n) {
    W = runif(1)
    Y[i] = -3 * log(W)
    U = runif(1)
    V = M * dexp(Y[i], 1/3) * U
    while (dchisq(Y[i], 3) < V) {
        W = runif(1)
        Y[i] = -3 * log(W)
        U = runif(1)
        V = M * dexp(Y[i], 1/3) * U
    }
```

```
}
U = runif(n/2)
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
S = sqrt(3/Y) * Z
X = abs(S)
I = mean(X)
print(I)
```

```
## [1] 1.099443
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 1.727892
```

```
U = runif(n)
Y = tan(pi * (U - 0.5))
W = 2 * abs(Y) * (1 + Y^2)/sqrt(3) * (1 + Y^2/3)^(-2)
I = mean(W)
print(I)
```

```
## [1] 1.102958
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.5165322
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 70.10622
```

We will then use $Y \sim \mathcal{N}(0,1)$ as an importance variable. For $x \in \mathbb{R}$, we calculate that:

$$\phi(x) = \frac{h(x)f(x)}{g(x)} = |x|\sqrt{2\pi}e^{x^2/2}\frac{1}{\sqrt{3\pi}\sqrt{\pi}/2}\left(1+\frac{x^2}{3}\right)^{-2} = \frac{4|x|e^{x^2/2}}{\sqrt{6\pi}}\left(1+\frac{x^2}{3}\right)^{-2}.$$

```
U = runif(n/2)
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Y = sqrt(D) * c(cos(Theta), sin(Theta))
W = 4 * abs(Y) * exp(Y^2/2)/sqrt(6 * pi) * (1 + Y^2/3)^(-2)
I = mean(W)
print(I)
```

```
## [1] 1.010371
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

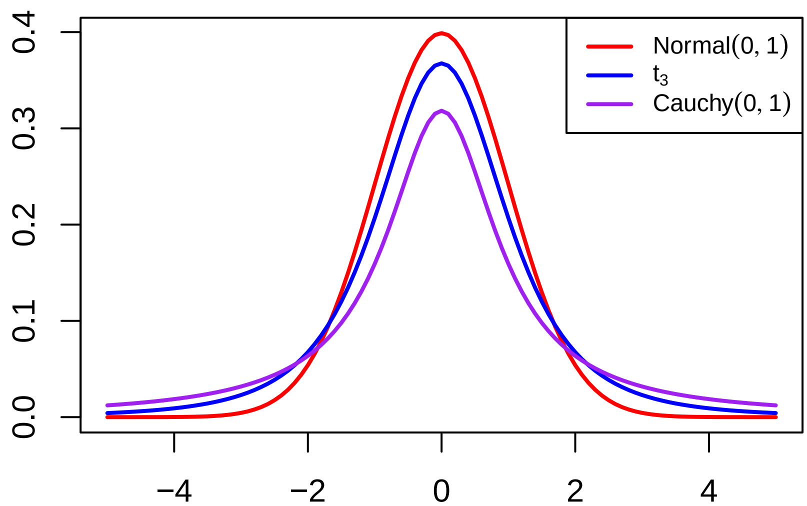
```
## [1] 304.6457
```

```r
100 * (VarW - VarX)/VarX
```

```
## [1] 17531.06
```

We observe that the variance of the estimator of $\mathbb{E}(|S|)$ is hugely increased for this specific choice of importance variable. As far as the function $h(x) = |x|$ is concerned, we observe that $h(x) \approx 0$ for $x \approx 0$ and $h(x) \gg 0$ for $|x| \gg 0$. If $Y \sim \text{Cauchy}(0, 1)$, then $f(x) \gg g(x)$ for $x \approx 0$ and $f(x) \ll g(x)$ for $|x| \gg 0$, i.e. this choice of importance variable is suitable. If $Y \sim \mathcal{N}(0, 1)$, then $f(x) \ll g(x)$ for $x \approx 0$ and $f(x) \gg g(x)$ for $|x| \gg 0$, i.e. the importance sampling method leads to an estimator with much higher variance than the original.

```r
curve(dnorm(x), col = "red", lwd = 2, xlab = NA, ylab = NA, xlim = c(-5, 5))
curve(dt(x, 3), add = TRUE, col = "blue", lwd = 2)
curve(dcauchy(x), add = TRUE, col = "purple", lwd = 2)
legend("topright", c(expression(Normal(0, 1)), expression(t[3]), expression(Cauchy(0,
    1))), col = c("red", "blue", "purple"), lty = c(1, 1, 1), lwd = c(2, 2,
    2), cex = 0.75)
```



Example 4.18. Let $S \sim \text{Exp}(2)$ and $R \sim \text{Exp}(1)$ be independent random variables. We want to estimate the expected value $\mathbb{E}(\max\{S + R - 7, 0\})$. We will use $g(x, y) = 2e^{-2x}e^{-(y-\max\{7-x,0\})}$ as an importance density for $x > 0$ and $y > \max\{7 - x, 0\}$. Then, we calculate that:

$$\phi(x, y) = \frac{h(x, y)f(x, y)}{g(x, y)} = (x + y - 7)\frac{2e^{-2x}e^{-y}}{2e^{-2x}e^{-(y-\max\{7-x,0\})}} = \frac{x + y - 7}{e^{\max\{7-x,0\}}}.$$

```r
n = 1e+05
U = runif(n)
S = -log(U)/2
V = runif(n)
```

```r
R = -log(V)
X = pmax(S + R - 7, 0)
I = mean(X)
print(I)
```

```
## [1] 0.001809774
```

```r
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.003704638
```

```r
U = runif(n)
S = -log(U)/2
V = runif(n)
Y = pmax(7 - S, 0) - log(V)
W = (S + Y - 7)/exp(pmax(7 - S, 0))
I = mean(W)
print(I)
```

```
## [1] 0.001837356
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 2.769405e-05
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 99.25245
```

Example 4.19. Let $R_1, R_2, \cdots \sim \mathcal{N}\left(\mu, \sigma^2\right)$ be a sequence of independent random variables with $\mu < 0$ and $A, B > 0$. We define the random variables:

$$S_m = \sum_{j=1}^{m} R_j, \quad M = \min\left\{m \in \mathbb{N} : S_m < -A \text{ or } S_m > B\right\}.$$

We want to estimate the probability $\mathbb{P}\left(S_M > B\right)$. We will use $Y_1, Y_2, \cdots \sim \mathcal{N}\left(-\mu, \sigma^2\right)$ as importance variables. Then, we infer that:

$$\begin{aligned}\phi(x) = \frac{h(x)f(x)}{g(x)} &= \mathbb{1}_{\{S_M > B\}} \prod_{j=1}^{M} \frac{f_{R_j}(x_j)}{f_{Y_j}(x_j)} = \mathbb{1}_{\{S_M > B\}} \prod_{j=1}^{M} \exp\left\{-\frac{1}{2\sigma^2}\left(x_j - \mu\right)^2\right\} \exp\left\{\frac{1}{2\sigma^2}\left(x_j + \mu\right)^2\right\} \\ &= \mathbb{1}_{\{S_M > B\}} \exp\left\{\sum_{j=1}^{M} 2\mu x_j\right\} = \mathbb{1}_{\{S_M > B\}} \exp\left\{\frac{2\mu S_M}{\sigma^2}\right\}.\end{aligned}$$

```r
n = 10000
mu = -3
sigma = 2
```

```r
A = 6
B = 3
lambda = 1/sigma
M = sqrt(2 * exp(1)/pi)
S = numeric(n)
for (i in 1:n) {
    while (S[i] >= -A && S[i] <= B) {
        W = runif(1)
        Y = ifelse(W <= 0.5, mu + log(2 * W)/lambda, mu - log(2 * (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(Y - mu), lambda)/2 * U
        while (dnorm(Y, mu, sigma) < V) {
            W = runif(1)
            Y = ifelse(W <= 0.5, mu + log(2 * W)/lambda, mu - log(2 * (1 - W))/lambda)
            U = runif(1)
            V = M * dexp(abs(Y - mu), lambda)/2 * U
        }
        S[i] = S[i] + Y
    }
}
X = S > B
I = mean(X)
print(I)
```

```
## [1] 0.0022
```

```r
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.00219516
```

```r
S = numeric(n)
for (i in 1:n) {
    while (S[i] >= -A && S[i] <= B) {
        W = runif(1)
        Y = ifelse(W <= 0.5, -mu + log(2 * W)/lambda, -mu - log(2 * (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(Y + mu), lambda)/2 * U
        while (dnorm(Y, -mu, sigma) < V) {
            W = runif(1)
            Y = ifelse(W <= 0.5, -mu + log(2 * W)/lambda, -mu - log(2 * (1 -
                W))/lambda)
            U = runif(1)
            V = M * dexp(abs(Y + mu), lambda)/2 * U
        }
```

```
        S[i] = S[i] + Y
    }
}
W = (S > B) * exp(2 * mu * S/sigma^2)
I = mean(W)
print(I)
```

```
## [1] 0.002104925
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 7.516815e-06
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 99.65757
```

Definition 4.1. Let X be a random variable with support S, PDF f and MGF $M(t) = \mathbb{E}\left[e^{tX}\right]$ for $t \in \mathbb{R}$. For $x \in S$, we define the tilted PDF of f as follows:

$$f_t(x) = \frac{e^{tx} f(x)}{M(t)}.$$

Note 4.4. Let $X \sim f$ and $Y \sim f_t$ be independent random variables. If $t > 0$, then $\mathbb{E}(Y) > \mathbb{E}(X)$. Otherwise, $\mathbb{E}(Y) < \mathbb{E}(X)$.

Example 4.20. Let $R_1, \ldots, R_k \sim \text{Exp}(1)$ be independent random variables. We define the random variable $S = R_1 + \cdots + R_k$. We want to estimate the probability $\mathbb{P}(S > a)$ with $a > k$. For $t < 1$, we know that $M_j(t) = \mathbb{E}\left(e^{tR_j}\right) = \frac{1}{1-t}$. For $x > 0$, we define the following tilted importance densities:

$$f_t^{(j)}(x) = \frac{e^{tx} f_{R_j}(x)}{M_j(t)} = (1-t)e^{tx}e^{-x} = (1-t)e^{-(1-t)x}.$$

In other words, we are led to the tilted importance variables $Y_1, \ldots, Y_k \sim \text{Exp}(1-t)$. Then, we infer that:

$$\begin{aligned}
\phi(x) = \frac{h(x)f(x)}{g(x)} &= \mathbb{1}_{\{S>a\}} \prod_{j=1}^{k} \frac{f_{R_j}(x_j)}{f_t^{(j)}(x_j)} = \mathbb{1}_{\{S>a\}} \prod_{j=1}^{k} \frac{e^{-x_j}}{(1-t)e^{-(1-t)x_j}} \\
&= \frac{\mathbb{1}_{\{S>a\}}}{(1-t)^k} \exp\left\{-\sum_{j=1}^{k} t x_j\right\} = \frac{\mathbb{1}_{\{S>a\}} e^{-tS}}{(1-t)^k}.
\end{aligned}$$

A suitable value for the parameter t is given by letting $\mathbb{E}_t(S) = a$. Then, we conclude that:

$$\sum_{j=1}^{k} \frac{1}{1-t} = a \quad \Rightarrow \quad t = 1 - \frac{k}{a}.$$

```
n = 1e+05
k = 4
```

```r
a = 10
U = matrix(runif(k * n), n)
R = -log(U)
S = rowSums(R)
X = S > a
I = mean(X)
print(I)
```

```
## [1] 0.00991
```

```r
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.009811792
```

```r
t = 1 - k/a
print(t)
```

```
## [1] 0.6
```

```r
V = matrix(runif(k * n), n)
Y = -log(V)/(1 - t)
S = rowSums(Y)
W = (S > a) * exp(-t * S)/(1 - t)^k
I = mean(W)
print(I)
```

```
## [1] 0.0103581
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.0004474145
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 95.44003
```

Example 4.21. Let $R_1, \dots, R_k \sim \text{Bernoulli}(p)$ be independent random variables. we define the random variables $S = R_1 + \cdots + R_k$. We want to estimate the probability $\mathbb{P}(S \geqslant a)$ with $a < k$. For $t \in \mathbb{R}$, we know that $M_j(t) = \mathbb{E}\left(e^{tR_j}\right) = 1 - p + pe^t$. For $x \in \{0, 1\}$, we define the following tilted importance densities:

$$f_t^{(j)}(x) = \frac{e^{tx} f_{R_j}(x)}{M_j(t)} = \frac{e^{tx} p^x (1-p)^{1-x}}{1 - p + pe^t} = \left(\frac{pe^t}{1 - p + pe^t}\right)^x \left(\frac{1-p}{1 - p + pe^t}\right)^{1-x}.$$

In other words, we are led to the tilted importance variables $Y_1, \dots, Y_k \sim \text{Bernoulli}\left(\frac{pe^t}{1-p+pe^t}\right)$. Then, we infer that:

$$\phi(x) = \frac{h(x)f(x)}{g(x)} = \mathbb{1}_{\{S\geqslant a\}} \prod_{j=1}^{k} \frac{f_{R_j}(x_j)}{f_t^{(j)}(x_j)} = \mathbb{1}_{\{S\geqslant a\}} \prod_{j=1}^{k} p^{x_j}(1-p)^{1-x_j} \frac{1-p+pe^t}{e^{tx_j}p^{x_j}(1-p)^{1-x_j}}$$

$$= \mathbb{1}_{\{S\geqslant a\}} \left(1-p+pe^t\right)^k \exp\left\{-\sum_{j=1}^{k} tx_j\right\} = \mathbb{1}_{\{S\geqslant a\}} \left(1-p+pe^t\right)^k e^{-tS}.$$

A suitable value for the parameter t is given by letting $\mathbb{E}_t(S) = a$. Then, we conclude that:

$$\sum_{j=1}^{k} \frac{pe^t}{1-p+pe^t} = a \quad \Rightarrow \quad t = \log \frac{a(1-p)}{p(k-a)}.$$

```
n = 1e+05
p = 0.4
k = 20
a = 16
U = matrix(runif(k * n), n)
S = rowSums(U < p)
X = S >= a
I = mean(X)
print(I)
```

```
## [1] 0.00032
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.0003198976
```

```
t = log(a * (1 - p)/(p * (k - a)))
print(t)
```

```
## [1] 1.791759
```

```
V = matrix(runif(k * n), n)
S = rowSums(V < p * exp(t)/(1 - p + p * exp(t)))
W = (S >= a) * (1 - p + p * exp(t))^k * exp(-t * S)
I = mean(W)
print(I)
```

```
## [1] 0.0003173146
```

```
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 2.419507e-07
```

```
100 * (VarX - VarW)/VarX
```

```
## [1] 99.92437
```

Example 4.22. Consider a $M/M/1$ queuing system, where the arrival process is Poisson with rate λ, inter-arrival times $P_1, P_2, \ldots$ and the services times $R_1, R_2, \ldots$ follow the $\text{Exp}(\mu)$ distribution with $\mu > \lambda$. We let W_j be the waiting time of the j-th customer in the queue. Additionally, we define:

$$S_W = \sum_{j=1}^{N^*} W_j, \quad S_P = \sum_{j=1}^{N^*} P_j, \quad S_R = \sum_{j=1}^{N*} R_j.$$

We want to estimate the probability $\mathbb{P}(S_W > a)$. We will use the importance variables $\widetilde{P}_1, \ldots, \widetilde{P}_{N^*} \sim \text{Exp}(\mu)$ and $\widetilde{R}_1, \ldots, \widetilde{R}_{N^*} \sim \text{Exp}(\lambda)$. Then, we infer that:

$$\begin{aligned}\phi(x) = \frac{h(x)f(x)}{g(x)} &= \mathbb{1}_{\{S_W > a\}} \prod_{j=1}^{N^*} \frac{f_{P_j}(x_j) f_{R_j}(y_j)}{f_{\widetilde{P}_j}(x_j) f_{\widetilde{R}_j}(y_j)} = \mathbb{1}_{\{S_W > a\}} \prod_{j=1}^{N^*} \frac{\lambda e^{-\lambda x_j} \mu e^{-\mu y_j}}{\mu e^{-\mu x_j} \lambda e^{-\lambda y_j}} \\ &= \mathbb{1}_{\{S_W > a\}} \exp\left\{ (\mu - \lambda) \sum_{j=1}^{N^*} x_j + (\lambda - \mu) \sum_{j=1}^{N^*} y_j \right\} = \mathbb{1}_{\{S_W > a\}} e^{(\mu - \lambda)(S_P - S_R)}.\end{aligned}$$

```
n = 10000
lambda = 4
mu = 6
Nstar = 5
a = 4
SW = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/lambda
    D = Inf
    N = 0
    arrivals = numeric(0)
    while (N < Nstar || Q > 0) {
        t = min(A, D)
        if (t == A) {
            Q = Q + 1
            N = N + 1
            if (N < Nstar) {
                U = runif(1)
                A = t - log(U)/lambda
            } else {
                A = Inf
            }
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/mu
            }
```

```
            arrivals = c(arrivals, t)
        } else {
            Q = Q - 1
            arrivals = arrivals[-1]
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/mu
                SW[i] = SW[i] + t - arrivals[1]
            } else {
                D = Inf
            }
        }
    }
}
X = SW > a
I = mean(X)
print(I)
```

```
## [1] 0.0019
```

```
VarX = mean((X - I)^2)
print(VarX)
```

```
## [1] 0.00189639
```

```
SW = numeric(n)
SP = numeric(n)
SR = numeric(n)
for (i in 1:n) {
    Q = 0
    U = runif(1)
    A = -log(U)/mu
    SP[i] = SP[i] - log(U)/mu
    D = Inf
    N = 0
    arrivals = numeric(0)
    while (N < Nstar || Q > 0) {
        t = min(A, D)
        if (t == A) {
            Q = Q + 1
            N = N + 1
            if (N < Nstar) {
                U = runif(1)
                A = t - log(U)/mu
                SP[i] = SP[i] - log(U)/mu
```

```r
            } else {
                A = Inf
            }
            if (Q == 1) {
                V = runif(1)
                D = t - log(V)/lambda
                SR[i] = SR[i] - log(V)/lambda
            }
            arrivals = c(arrivals, t)
        } else {
            Q = Q - 1
            arrivals = arrivals[-1]
            if (Q > 0) {
                V = runif(1)
                D = t - log(V)/lambda
                SR[i] = SR[i] - log(V)/lambda
                SW[i] = SW[i] + t - arrivals[1]
            } else {
                D = Inf
            }
        }
    }
}
W = (SW > a) * exp((mu - lambda) * (SP - SR))
I = mean(W)
print(I)
```

```
## [1] 0.002036983
```

```r
VarW = mean((W - I)^2)
print(VarW)
```

```
## [1] 0.0001958506
```

```r
100 * (VarX - VarW)/VarX
```

```
## [1] 89.67245
```

5 Markov Chain Monte Carlo Methods

Gibbs Sampler

We want to generate a sample $X^{(1)}, X^{(2)}, \dots, X^{(n)}$ following the joint PDF $f_{X_1,X_2,\dots,X_k}$. Suppose that either the marginal PDFs f_{X_j} are intractable or it's difficult to simulate from them, but it's easy to simulate from the conditional PDFs $f_{X_j|X_{-j}} \equiv f_{X_j|X_1,\dots,X_{j-1},X_{j+1},\dots,X_k}$.

Algorithm 5.1 Gibbs Sampler

Input: Conditional CDFs, burn-in size b and sample size n.

1: We consider the initial values $X_1^{(1)}, X_2^{(1)}, \dots, X_k^{(1)}$.

2: For $i = 2, 3, \dots, b+n$, we iterate the following step:

i: For $j = 1, 2, \dots, k$, we generate $X_j^{(i)}$ according to the conditional CDF:

$$F_{X_j|X_1,\dots,X_{j-1},X_{j+1},\dots,X_k}\left(x \,\middle|\, X_1^{(i)}, \dots, X_{j-1}^{(i)}, X_{j+1}^{(i-1)}, \dots, X_k^{(i-1)}\right).$$

Output: Random sample $X^{(b+1)}, X^{(b+2)}, \dots, X^{(b+n)}$ following the joint CDF.

The sequence of random variables $\left\{X^{(i)}\right\}$ constitutes a discrete-time Markov process with the following transition kernel:

$$K\left(x^{(i)} \,\middle|\, x^{(i-1)}\right) = \prod_{j=1}^{k} f_{X_j|X_{-j}}\left(x_j^{(i)} \,\middle|\, x_1^{(i)}, \dots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \dots, x_k^{(i-1)}\right).$$

Theorem 5.1. If the Markov process $\left\{X^{(i)}\right\}$ with transition kernel $K\left(x^{(i)} \,\middle|\, x^{(i-1)}\right)$ and state-space S is irreducible, then $f_{X_1,X_2,\dots,X_k}$ is its unique stationary distribution.

Proof. We define the reverse transition kernel:

$$L\left(x^{(i-1)} \,\middle|\, x^{(i)}\right) = \prod_{j=1}^{k} f_{X_j|X_{-j}}\left(x_j^{(i-1)} \,\middle|\, x_1^{(i)}, \dots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \dots, x_k^{(i-1)}\right).$$

Then, we observe that:

$$\begin{aligned}
f_{X_1,\dots,X_k}\left(x^{(i-1)}\right) K\left(x^{(i)} \,\middle|\, x^{(i-1)}\right) &= f_{X_1,\dots,X_k}\left(x^{(i-1)}\right) \prod_{j=1}^{k} f_{X_j|X_{-j}}\left(x_j^{(i)} \,\middle|\, x_1^{(i)}, \dots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \dots, x_k^{(i-1)}\right) \\
&= f_{X_1,\dots,X_k}\left(x^{(i-1)}\right) \prod_{j=1}^{k} \frac{f_{X_1,\dots,X_k}\left(x_1^{(i)}, \dots, x_j^{(i)}, x_{j+1}^{(i-1)}, \dots, x_k^{(i-1)}\right)}{f_{X_{-j}}\left(x_1^{(i)}, \dots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \dots, x_k^{(i-1)}\right)} \\
&= f_{X_1,\dots,X_k}\left(x^{(i)}\right) \prod_{j=1}^{k} \frac{f_{X_1,\dots,X_k}\left(x_1^{(i)}, \dots, x_{j-1}^{(i)}, x_j^{(i-1)}, \dots, x_k^{(i-1)}\right)}{f_{X_{-j}}\left(x_1^{(i)}, \dots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \dots, x_k^{(i-1)}\right)} \\
&= f_{X_1,\dots,X_k}\left(x^{(i)}\right) \prod_{j=1}^{k} f_{X_j|X_{-j}}\left(x_j^{(i-1)} \,\middle|\, x_1^{(i)}, \dots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \dots, x_k^{(i-1)}\right) \\
&= f_{X_1,\dots,X_k}\left(x^{(i)}\right) L\left(x^{(i-1)} \,\middle|\, x^{(i)}\right).
\end{aligned}$$

Therefore, we infer that:

$$\begin{aligned}\int_S f_{X_1,\dots,X_k}\left(x^{(i-1)}\right) K\left(x^{(i)}\middle| x^{(i-1)}\right) dx^{(i-1)} &= \int_S f_{X_1,\dots,X_k}\left(x^{(i)}\right) L\left(x^{(i-1)}\middle| x^{(i)}\right) dx^{(i-1)} \\ &= f_{X_1,\dots,X_k}\left(x^{(i)}\right) \int_S L\left(x^{(i-1)}\middle| x^{(i)}\right) dx^{(i-1)} = f_{X_1,\dots,X_k}\left(x^{(i)}\right).\end{aligned}$$

Since the function $f_{X_1,X_2,\dots,X_k}$ satisfies the balance equations for the Markov process $\left\{X^{(i)}\right\}$, we conclude that it is the unique stationary distribution of the Markov process. □

Note 5.1. Let S_j be the support of the marginal CDF F_{X_j} for $j = 1, 2, \dots, k$. If $S = S_1 \times S_2 \times \cdots \times S_k$, then the Markov process $\left\{X^{(i)}\right\}$ with state-space S and transition kernel $K\left(x^{(i)}\middle| x^{(i-1)}\right)$ is irreducible.

Note 5.2.

i. The Markov Chain Monte Carlo algorithms require an initial number of iterations until the Markov process which they produce converges to its stationary distribution, i.e. the given joint PDF. This initial number of iterations is called the burn-in period of the algorithm and the sample which has been produced during this period is thrown away since it doesn't follow the given distribution.

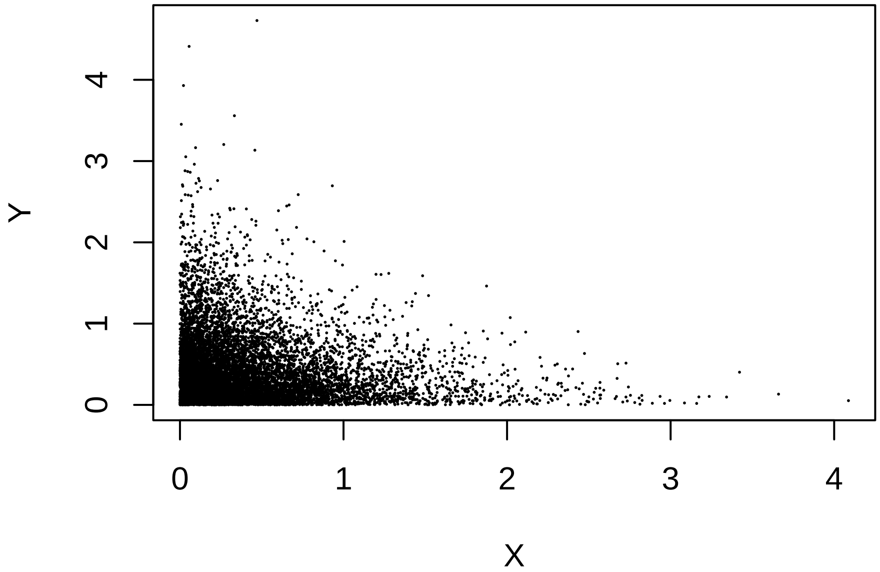
ii. The observations which are produced by a Markov Chain Monte Carlo algorithm are not independent. On the contrary, they display a dependence pattern which is determined by the properties of the Markov process that the algorithm produces. The lack of independence of the observations doesn't influence the approximation of expected values via the Monte Carlo method. In cases where the use of random samples is required, we can perform a thinning of the sample which is produced by the algorithm. If we calculate that up to T^* consecutive observations which are produced by the algorithm display statistically significant autocorrelation, then we accept only one out of every T^* observations produced by the algorithm and reject all the rest.

Example 5.1. We want to generate a sample $(X_1, Y_1), \dots, (X_n, Y_n)$ with PDF $f(x, y) \propto e^{-x-y-axy}$ for $x, y > 0$ and $a > 0$. We observe that:

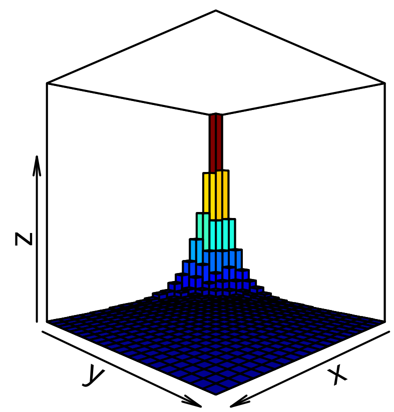
$$f_{X|Y}(x \mid y) \propto e^{-x-axy} = e^{-(y+a)x}, \quad f_{Y|X}(y \mid x) \propto e^{-y-axy} = e^{-(x+a)y}.$$

In other words, $(X \mid Y = y) \sim \mathrm{Exp}(y + a)$ and $(Y \mid X = x) \sim \mathrm{Exp}(x + a)$.

```
library(plot3D)
b = 10000
n = 10000
a = 2
X = numeric(b + n)
Y = numeric(b + n)
for (i in 2:(b + n)) {
    U = runif(1)
    X[i] = -log(U)/(Y[i - 1] + a)
    V = runif(1)
    Y[i] = -log(V)/(X[i] + a)
}
X = X[-(1:b)]
Y = Y[-(1:b)]
plot(X, Y, pch = 16, cex = 0.2)
```




```r
hist3D(z = table(cut(X, 20), cut(Y, 20)), colkey = FALSE, phi = 0, theta = 135,
    border = 1)
```



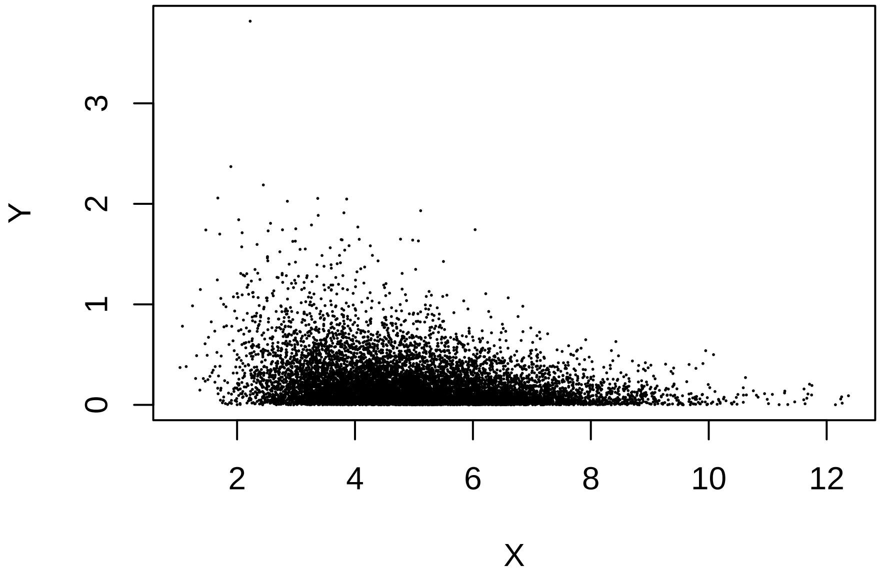
Example 5.2. We want to generate a sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ with PDF $f(x, y) \propto x^k e^{-(\lambda+y)x}$ for $x, y > 0$ and $k, \lambda > 0$. We observe that:

$$f_{X|Y}(x \mid y) \propto x^k e^{-(\lambda+y)x}, \quad f_{Y|X}(y \mid x) \propto e^{-xy}.$$

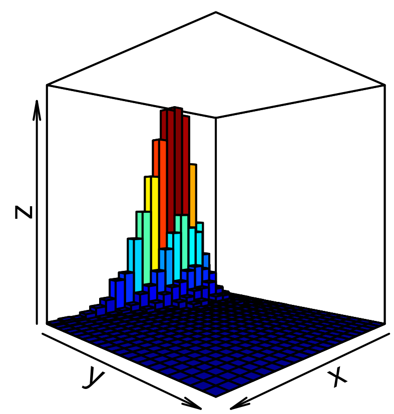
In other words, $(X \mid Y = y) \sim \text{Gamma}(k + 1, \lambda + y)$ and $(Y \mid X = x) \sim \text{Exp}(x)$.

```r
library(plot3D)
b = 10000
n = 10000
k = 10
lambda = 2
X = numeric(b + n)
Y = numeric(b + n)
for (i in 2:(b + n)) {
    U = runif(k + 1)
    R = -log(U)/(lambda + Y[i - 1])
```

```
    X[i] = sum(R)
    V = runif(1)
    Y[i] = -log(V)/X[i]
}
X = X[-(1:b)]
Y = Y[-(1:b)]
plot(X, Y, pch = 16, cex = 0.2)
```



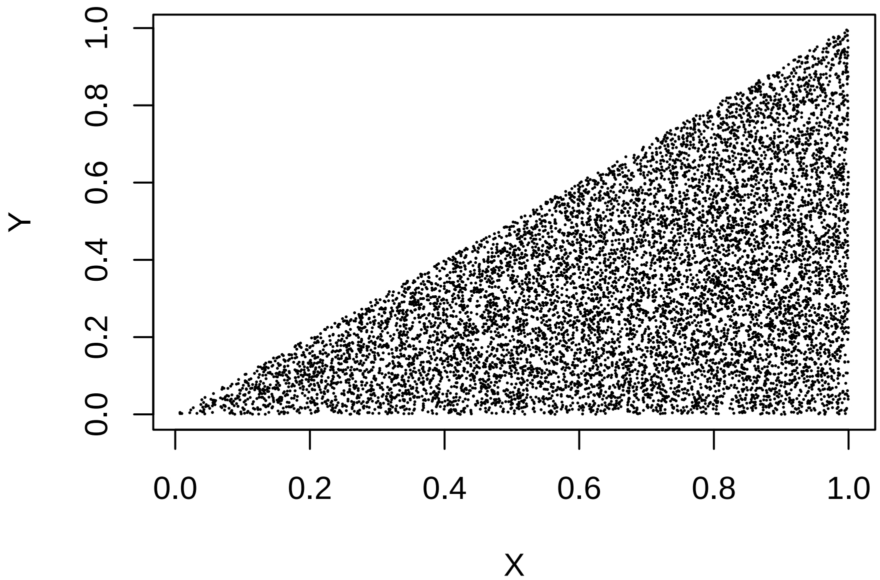

```
hist3D(z = table(cut(X, 20), cut(Y, 20)), colkey = FALSE, phi = 0, theta = 135,
    border = 1)
```



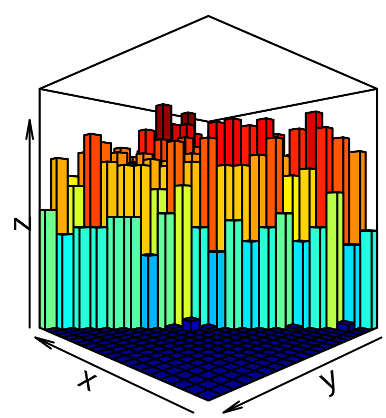
Example 5.3. We want to generate a sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ with PDF $f(x, y) \propto 1$ for $0 \leqslant y \leqslant x \leqslant 1$. We observe that $(X \mid Y = y) \sim \text{Unif}[y, 1]$ for $y \in [0, 1]$ and $(Y \mid X = x) \sim \text{Unif}[0, x]$ for $x \in [0, 1]$.

```
library(plot3D)
b = 10000
n = 10000
X = numeric(b + n)
Y = numeric(b + n)
for (i in 2:(b + n)) {
```

```
    U = runif(1)
    X[i] = (1 - Y[i - 1]) * U + Y[i - 1]
    V = runif(1)
    Y[i] = X[i] * V
}
X = X[-(1:b)]
Y = Y[-(1:b)]
plot(X, Y, pch = 16, cex = 0.2)
```



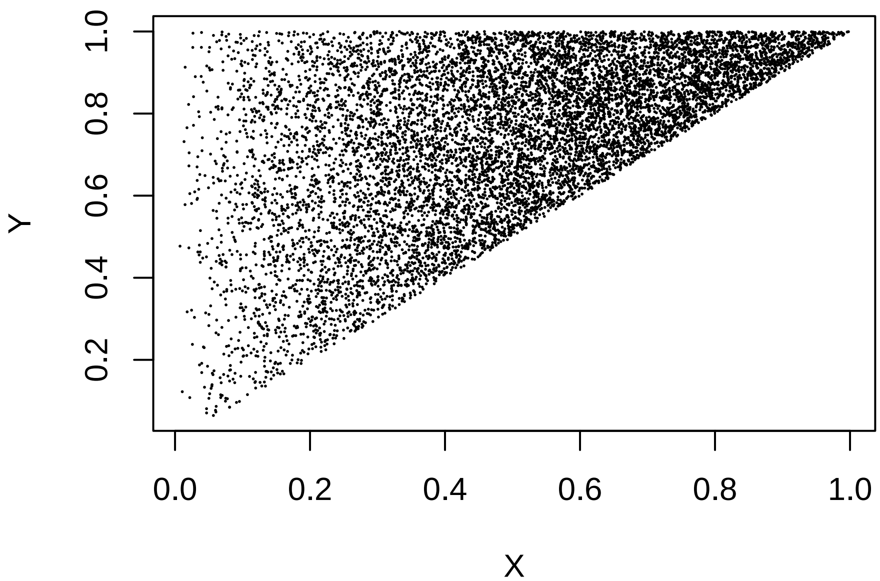

```
hist3D(z = table(cut(X, 20), cut(Y, 20)), colkey = FALSE, phi = 0, theta = 225,
    border = 1)
```



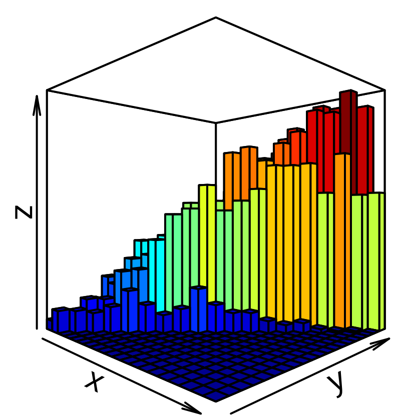
Example 5.4. We want to generate a sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ with PDF $f(x, y) \propto x$ for $0 \leqslant x \leqslant y \leqslant 1$. For $x \in [0, 1]$, we observe that $f_{Y|X}(y \mid x) \propto 1$, i.e. $(Y \mid X = x) \sim \text{Unif}[x, 1]$. For $y \in [0, 1]$, we observe that $f_{X|Y}(x \mid y) \propto x$. For $x \in [0, y]$, we calculate that:

$$f_{X|Y}(x \mid y) = \frac{2x}{y^2}, \quad F_{X|Y}(x \mid y) = \frac{x^2}{y^2}, \quad F^{-1}_{X|Y}(u \mid y) = y\sqrt{u}.$$

```
library(plot3D)
b = 10000
n = 10000
X = numeric(b + n)
Y = numeric(b + n)
for (i in 2:(b + n)) {
    U = runif(1)
    X[i] = Y[i - 1] * sqrt(U)
    V = runif(1)
    Y[i] = (1 - X[i]) * V + X[i]
}
X = X[-(1:b)]
Y = Y[-(1:b)]
plot(X, Y, pch = 16, cex = 0.2)
```




```
hist3D(z = table(cut(X, 20), cut(Y, 20)), colkey = FALSE, phi = 0, theta = 45,
    border = 1)
```



Example 5.5. For $\rho \in (-1, 1)$, we want to generate a sample:

$$(X_1, Y_1), \ldots, (X_n, Y_n) \sim \mathcal{N}_2\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}\right).$$

For $x, y \in \mathbb{R}$, we calculate that:

$$f_{X,Y}(x, y) = \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left\{-\frac{x^2 - 2\rho xy + y^2}{2\left(1-\rho^2\right)}\right\},$$

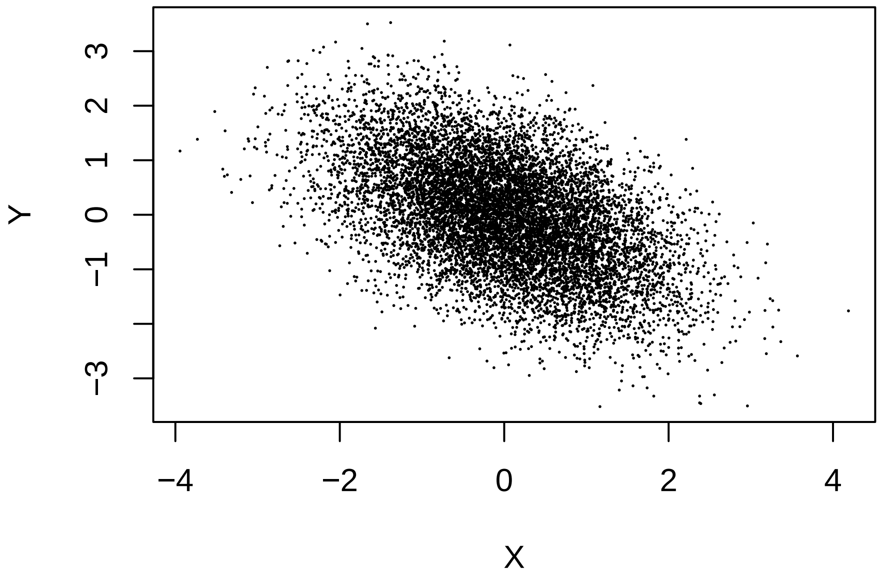
$$f_{X|Y}(x \mid y) \propto \exp\left\{-\frac{x^2 - 2\rho yx}{2\left(1-\rho^2\right)}\right\} \propto \exp\left\{-\frac{(x-\rho y)^2}{2\left(1-\rho^2\right)}\right\},$$

$$f_{Y|X}(y \mid x) \propto \exp\left\{-\frac{y^2 - 2\rho xy}{2\left(1-\rho^2\right)}\right\} \propto \exp\left\{-\frac{(y-\rho x)^2}{2\left(1-\rho^2\right)}\right\}.$$

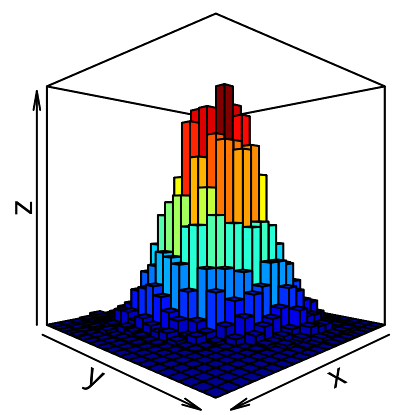
In other words, $(X \mid Y = y) \sim \mathcal{N}\left(\rho y, 1-\rho^2\right)$ and $(Y \mid X = x) \sim \mathcal{N}\left(\rho x, 1-\rho^2\right)$.

```
library(plot3D)
b = 10000
n = 10000
rho = -0.5
lambda = 1/sqrt(1 - rho^2)
M = sqrt(2 * exp(1)/pi)
X = numeric(b + n)
Y = numeric(b + n)
for (i in 2:(b + n)) {
    W = runif(1)
    X[i] = ifelse(W <= 0.5, rho * Y[i - 1] + log(2 * W)/lambda, rho * Y[i -
        1] - log(2 * (1 - W))/lambda)
    U = runif(1)
    V = M * dexp(abs(X[i] - rho * Y[i - 1]), lambda)/2 * U
    while (dnorm(X[i], rho * Y[i - 1], sqrt(1 - rho^2)) < V) {
        W = runif(1)
        X[i] = ifelse(W <= 0.5, rho * Y[i - 1] + log(2 * W)/lambda, rho * Y[i -
            1] - log(2 * (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(X[i] - rho * Y[i - 1]), lambda)/2 * U
    }
    W = runif(1)
    Y[i] = ifelse(W <= 0.5, rho * X[i] + log(2 * W)/lambda, rho * X[i] - log(2 *
        (1 - W))/lambda)
    U = runif(1)
    V = M * dexp(abs(Y[i] - rho * X[i]), lambda)/2 * U
    while (dnorm(Y[i], rho * X[i], sqrt(1 - rho^2)) < V) {
        W = runif(1)
```

```
        Y[i] = ifelse(W <= 0.5, rho * X[i] + log(2 * W)/lambda, rho * X[i] -
            log(2 * (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(Y[i] - rho * X[i]), lambda)/2 * U
    }
}
X = X[-(1:b)]
Y = Y[-(1:b)]
plot(X, Y, pch = 16, cex = 0.2)
```




```
hist3D(z = table(cut(X, 20), cut(Y, 20)), colkey = FALSE, phi = 0, theta = 135,
    border = 1)
```



Example 5.6. We want to generate a sample $(X_1, Y_1, Z_1), \ldots, (X_n, Y_n, Z_n)$ with the following PDF:

$$f(x, y, z) \propto \binom{z}{x} y^{x+a-1}(1-y)^{zx+b-1}\frac{\lambda^z}{z!}, \quad x \in \{0, 1, \ldots, z\}, \quad y \in [0, 1], \quad z \in \mathbb{N}.$$

We observe that:

$$f_{X|Y,Z}(x \mid y,z) \propto \binom{z}{x} y^x (1-y)^{zx} = \binom{z}{x} [y(1-y)^z]^x \propto \binom{z}{x} \left[\frac{y(1-y)^z}{y(1-y)^z+1}\right]^x \left[\frac{1}{y(1-y)^z+1}\right]^{z-x},$$

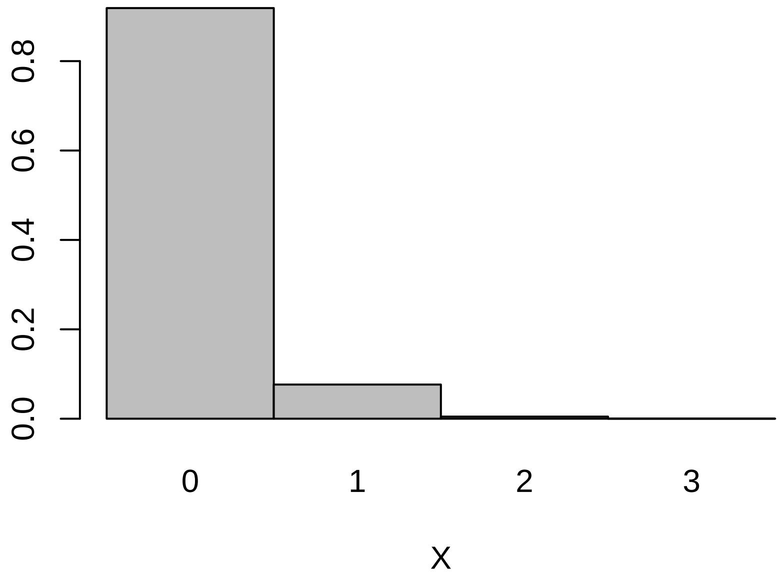
$$f_{Y|X,Z}(y \mid x,z) \propto y^{x+a-1}(1-y)^{zx+b-1},$$

$$f_{Z|X,Y}(z \mid x,y) \propto \binom{z}{x}(1-y)^{zx}\frac{\lambda^z}{z!} \propto \frac{[\lambda(1-y)^x]^z}{(z-x)!} \propto \frac{[\lambda(1-y)^x]^{z-x}}{(z-x)!}.$$

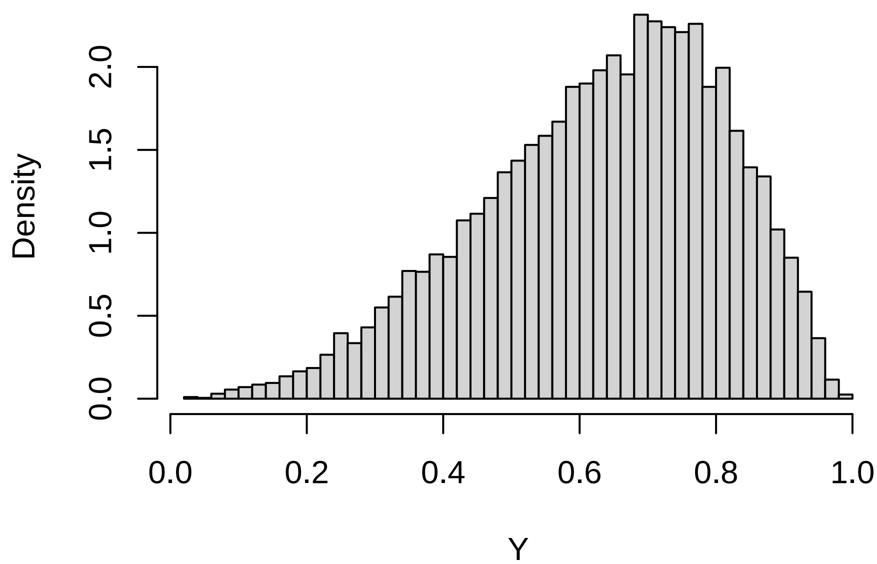
In other words, $(X \mid Y=y, Z=z) \sim \text{Binom}\left(z, \frac{y(1-y)^z}{y(1-y)^z+1}\right)$, $(Y \mid X=x, Z=z) \sim \text{Beta}(z+a, zx+b)$ and $(Z-X \mid X=x, Y=y) \sim \text{Poisson}\left(\lambda(1-y)^x\right)$.

```
b = 10000
n = 10000
a = 2
b = 3
lambda = 4
X = numeric(b + n)
Y = numeric(b + n)
Z = numeric(b + n)
for (i in 2:(b + n)) {
    U = runif(Z[i - 1])
    X[i] = sum(U < Y[i - 1] * (1 - Y[i - 1])^Z[i - 1]/(Y[i - 1] * (1 - Y[i -
        1])^Z[i - 1] + 1))
    M = dbeta((Z[i - 1] + a - 1)/(Z[i - 1] * (1 + X[i]) + a + b - 2), Z[i -
        1] + a, Z[i - 1] * X[i] + b)
    Y[i] = runif(1)
    U = runif(1)
    V = M * U
    while (dbeta(Y[i], Z[i - 1] + a, Z[i - 1] * X[i] + b) < V) {
        Y[i] = runif(1)
        U = runif(1)
        V = M * U
    }
    Z[i] = X[i]
    U = runif(1)
    pmf = exp(-(lambda * (1 - Y[i])^X[i]))
    cdf = pmf
    while (U > cdf) {
        Z[i] = Z[i] + 1
        pmf = pmf * lambda * (1 - Y[i])^X[i]/(Z[i] - X[i])
        cdf = cdf + pmf
    }
}
X = X[-(1:b)]
```

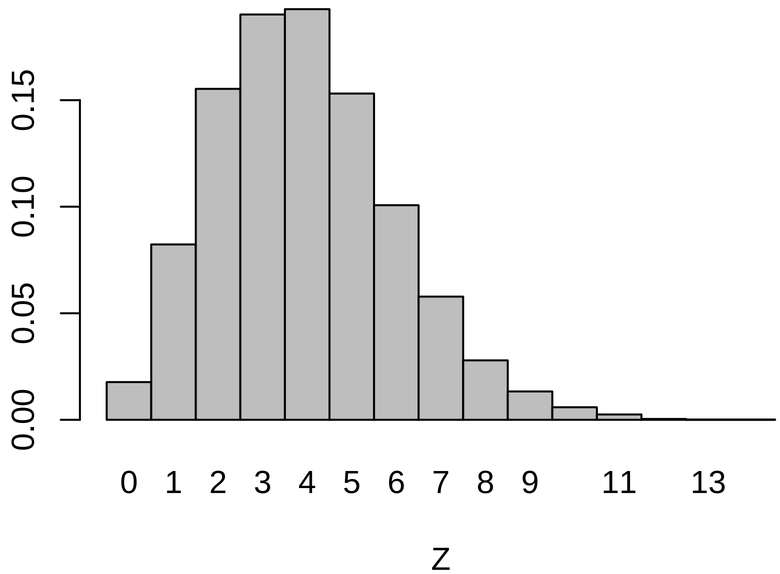
```r
Y = Y[-(1:b)]
Z = Z[-(1:b)]
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0, xlab = "X")
```




```r
hist(Y, "FD", freq = FALSE, main = NA)
```




```r
barplot(table(factor(Z, levels = 0:max(Z)))/n, space = 0, xlab = "Z")
```



Example 5.7. Let $X \sim \text{Poisson}(4)$ and $Y \sim \text{Poisson}(7)$ be independent random variables. We want to generate a sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ following the conditional distribution of (X, Y) given that $X + Y \leqslant 10$. For $x, y \in \mathbb{N}$ with $x + y \leqslant 10$, we calculate that:

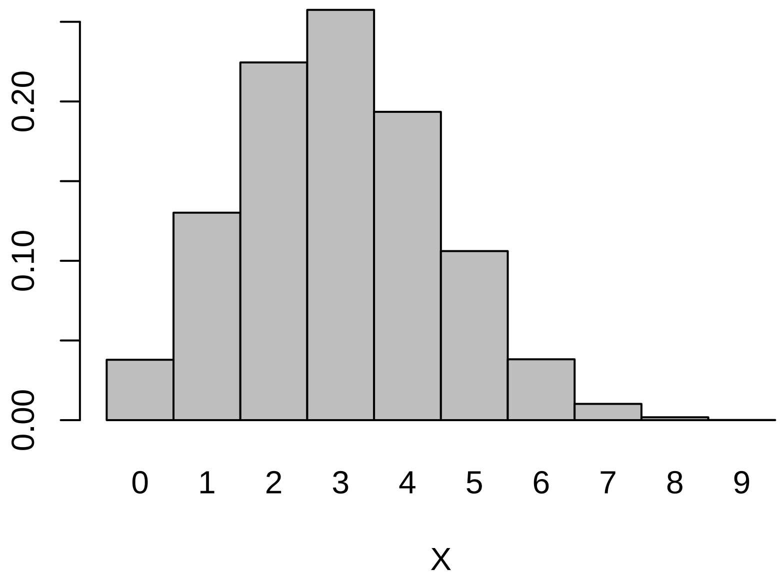
$$f_{X,Y|X+Y\leqslant 10}(x, y) = \frac{f_{X,Y}(x, y)}{\mathbb{P}(X + Y \leqslant 10)} \propto \frac{4^x 7^y}{x!y!},$$

$$f_{X|Y,X+Y\leqslant 10}(x \mid y) \propto \frac{4^x}{x!}, \quad f_{Y|X,X+Y\leqslant 10}(y \mid x) \propto \frac{7^y}{y!}.$$

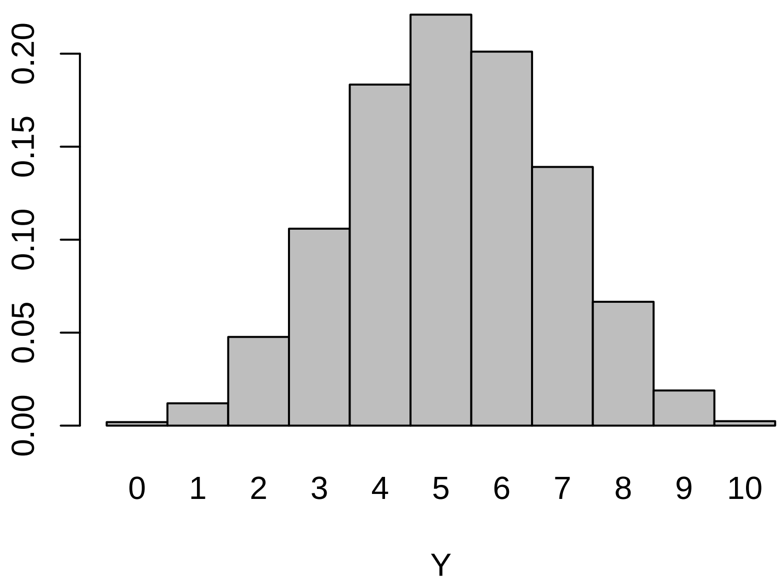
In other words, $(X \mid Y = y, X + Y \leqslant 10) \overset{d}{=} (X \mid X \leqslant 10 - y)$ and $(Y \mid X = x, X + Y \leqslant 10) \overset{d}{=} (Y \mid Y \leqslant 10 - x)$.

```
b = 10000
n = 10000
X = numeric(b + n)
Y = numeric(b + n)
for (i in 2:(b + n)) {
    U = runif(1)
    pmf = exp(-4)/ppois(10 - Y[i - 1], 4)
    cdf = pmf
    while (U > cdf) {
        X[i] = X[i] + 1
        pmf = pmf * 4/X[i]
        cdf = cdf + pmf
    }
    V = runif(1)
    pmf = exp(-7)/ppois(10 - X[i], 7)
    cdf = pmf
    while (V > cdf) {
        Y[i] = Y[i] + 1
        pmf = pmf * 7/Y[i]
```

```
        cdf = cdf + pmf
    }
}
X = X[-(1:b)]
Y = Y[-(1:b)]
barplot(table(factor(X, levels = 0:max(X)))/n, space = 0, xlab = "X")
```




```
barplot(table(factor(Y, levels = 0:max(Y)))/n, space = 0, xlab = "Y")
```



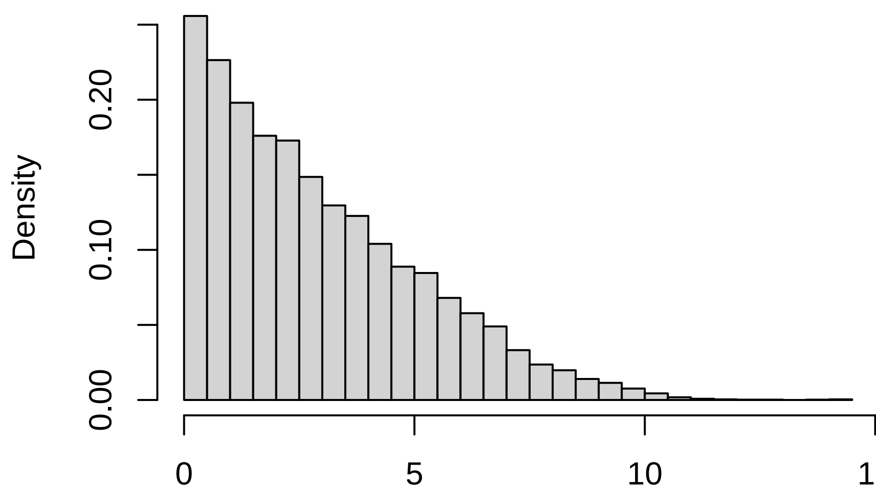
Example 5.8. Let $X_j \sim \text{Exp}(\lambda_j)$ be independent random variables for $j = 1, 2, \dots, k$. We define the random variable $S = X_1 + \cdots + X_k$. We want to generate a sample $X^{(1)}, X^{(2)}, \dots, X^{(n)}$ following the conditional distribution of $(X_1, X_2, \dots, X_k)$ given that $S > c$. For $x_1, \dots, x_k > 0$ with $x_1 + \cdots + x_k > c$, we calculate that:

$$f_{X_1,\dots,X_k|S>c}(x_1, \dots, x_k) = \frac{f_{X_1,\dots,X_k}(x_1, \dots, x_k)}{\mathbb{P}(X_1 + \cdots + X_k > c)} \propto \prod_{j=1}^{k} e^{-\lambda_j x_j} = \exp\left\{-\sum_{j=1}^{k} \lambda_j x_j\right\},$$

$$f_{X_j|X_{-j},S>c}(x_j \mid x_{-j}) \propto e^{-\lambda_j x_j}.$$

We let $s_{-j} = x_1 + \cdots + x_{j-1} + x_{j+1} + \cdots + x_k$. Then, $(X_j \mid X_{-j} = x_{-j}, S > c) \stackrel{d}{=} (X_j \mid X_j > c - s_{-j})$. In other words, we infer that $(X_j \mid X_{-j} = x_{-j}, S > c) \stackrel{d}{=} X_j + \max\{c - s_{-j}, 0\}$.

```
b = 10000
n = 10000
c = 10
k = 4
lambda = rep(1, k)
X = matrix(0, b + n, k)
for (i in 2:(b + n)) {
    for (j in 1:k) {
        if (j == 1) {
            s = sum(X[i - 1, -1])
        } else if (j == k) {
            s = sum(X[i, -k])
        } else {
            s = sum(c(X[i, 1:(j - 1)], X[i - 1, (j + 1):k]))
        }
        U = runif(1)
        X[i, j] = max(c - s, 0) - log(U)/lambda[j]
    }
}
X = X[-(1:b), ]
hist(X[, 1], "FD", freq = FALSE, main = NA, xlab = NA)
```



Example 5.9. Let $Y \sim \text{Unif}[0.02, 0.1]$ be a random variable. Consider the conditionally independent random variables $(W_1 \mid Y = y), (W_2 \mid Y = y) \sim \text{Exp}(y)$. Given that $W_1 = w_1$ and $W_2 = w_2$, we define the conditionally independent Poisson processes $\{N_1(t) : t \geqslant 0\}$ with rate w_1 and $\{N_2(t) : t \geqslant 0\}$ with rate w_2. We want to estimate the following conditional expected values:

$$\mathbb{E}\left[N_1(1) \mid N_1(0.5) = 25, N_2(0.5) = 18\right], \quad \mathbb{E}\left[N_2(1) \mid N_1(0.5) = 25, N_2(0.5) = 18\right].$$

According to the law of iterated expectations, we calculate that:

$$\begin{aligned}\mathbb{E}\left[N_1(1) \mid N_1(0.5)=25, N_2(0.5)=18\right] &= \mathbb{E}\left[\mathbb{E}\left(N_1(1) \mid W_1, N_1(0.5)=25, N_2(0.5)=18\right)\right] \\ &= \mathbb{E}\left[25+0.5W_1 \mid N_1(0.5)=25, N_2(0.5)=18\right],\end{aligned}$$

$$\begin{aligned}\mathbb{E}\left[N_2(1) \mid N_1(0.5)=25, N_2(0.5)\right] &= \mathbb{E}\left[\mathbb{E}\left(N_2(1) \mid W_2, N_1(0.5)=25, N_2(0.5)=18\right)\right] \\ &= \mathbb{E}\left[18+0.5W_2 \mid N_1(0.5)=25, N_2(0.5)=18\right].\end{aligned}$$

We define the events $A_1=[N_1(0.5)=25]$, $A_2=[N_2(0.5)=18]$. For $y \in [0.02, 0.1]$ and $w_1, w_2 > 0$, we calculate that:

$$\begin{aligned}f_{Y,W_1,W_2 \mid A_1,A_2}(y,w_1,w_2) &= \frac{f_{Y,W_1,W_2}(y,w_1,w_2)\mathbb{P}\left[A_1,A_2 \mid Y=y, W_1=w_1, W_2=w_2\right]}{\mathbb{P}\left(A_1,A_2\right)} \\ &\propto f_Y(y) f_{W_1,W_2\mid Y}(w_1,w_2 \mid y)\mathbb{P}\left[A_1 \mid W_1=w_1\right]\mathbb{P}\left[A_2 \mid W_2=w_2\right] \\ &\propto f_{W_1\mid Y}(w_1 \mid y) f_{W_2\mid Y}(w_2 \mid y) e^{-0.5w_1} w_1^{25} e^{-0.5w_2} w_2^{18} \\ &= y e^{-yw_1} y e^{-yw_2} w_1^{25} w_2^{18} e^{-0.5(w_1+w_2)} = y^2 w_1^{25} w_2^{18} e^{-(y+0.5)(w_1+w_2)},\end{aligned}$$

$$f_{Y\mid W_1,W_2,A_1,A_2}(y \mid w_1,w_2) \propto y^2 e^{-(w_1+w_2)y},$$

$$f_{W_1\mid Y,W_2,A_1,A_2}(w_1 \mid y,w_2) \propto w_1^{25} e^{-(y+0.5)w_1},$$

$$f_{W_2\mid Y,W_1,A_1,A_2}(w_2 \mid y,w_1) \propto w_2^{18} e^{-(y+0.5)w_2}.$$

If $(X \mid W_1=w_1, W_2=w_2) \sim \text{Gamma}(3, w_1+w_2)$, then:

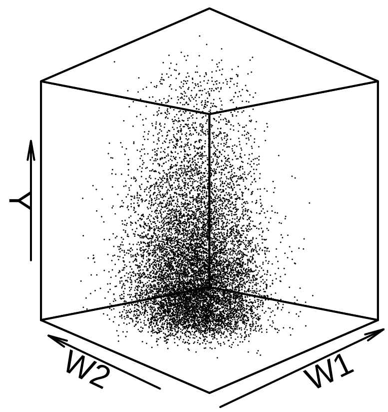
$$[Y \mid W_1=w_2, W_2=w_2, A_1, A_2] \stackrel{d}{=} \left(X \mid W_1=w_1, W_2=w_2, 0.02 \leqslant X \leqslant 0.1\right),$$

$$[W_1 \mid Y=y, W_2=w_2, A_1, A_2] \sim \text{Gamma}(26, y+0.5),$$

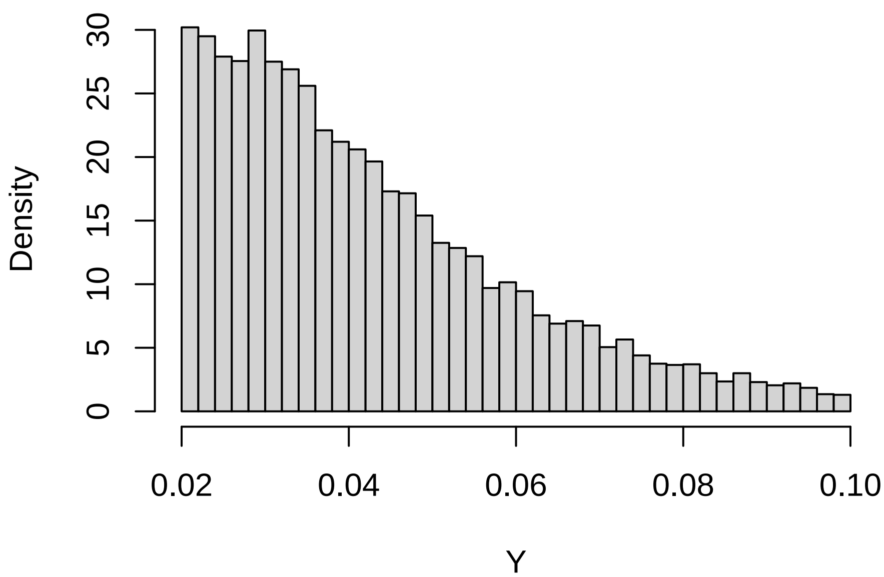
$$[W_2 \mid Y=y, W_1=w_1, A_1, A_2] \sim \text{Gamma}(19, y+0.5).$$

```
library(plot3D)
b = 10000
n = 10000
Y = numeric(b + n)
W1 = numeric(b + n)
W2 = numeric(b + n)
W1[1] = 1
W2[1] = 1
for (i in 2:(b + n)) {
    U = runif(3)
    R = -log(U)/(W1[i - 1] + W2[i - 1])
    Y[i] = sum(R)
    while (Y[i] < 0.02 || Y[i] > 0.1) {
        U = runif(3)
        R = -log(U)/(W1[i - 1] + W2[i - 1])
        Y[i] = sum(R)
    }
```

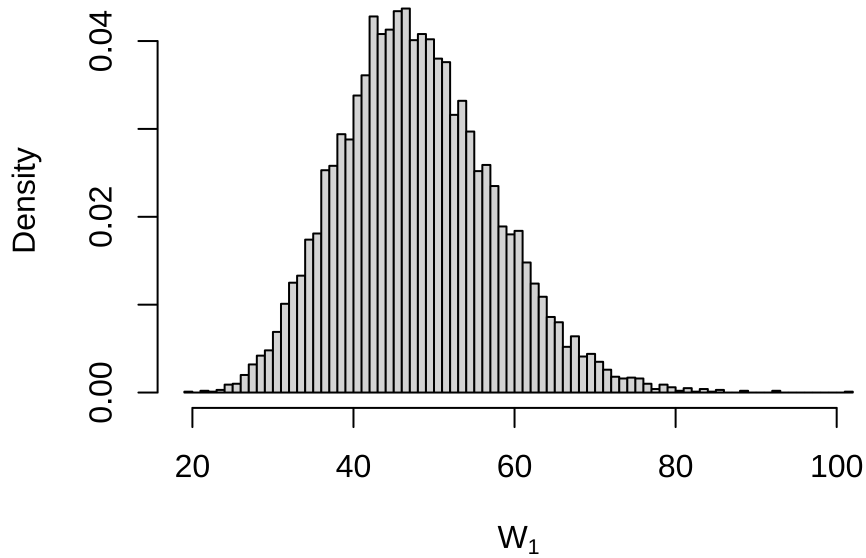
```
    U = runif(26)
    R = -log(U)/(Y[i] + 0.5)
    W1[i] = sum(R)
    U = runif(19)
    R = -log(U)/(Y[i] + 0.5)
    W2[i] = sum(R)
}
Y = Y[-(1:b)]
W1 = W1[-(1:b)]
W2 = W2[-(1:b)]
scatter3D(W1, W2, Y, colvar = NA, phi = 0, theta = 315, xlab = "W1", ylab = "W2",
    zlab = "Y", pch = 16, cex = 0.1)
```



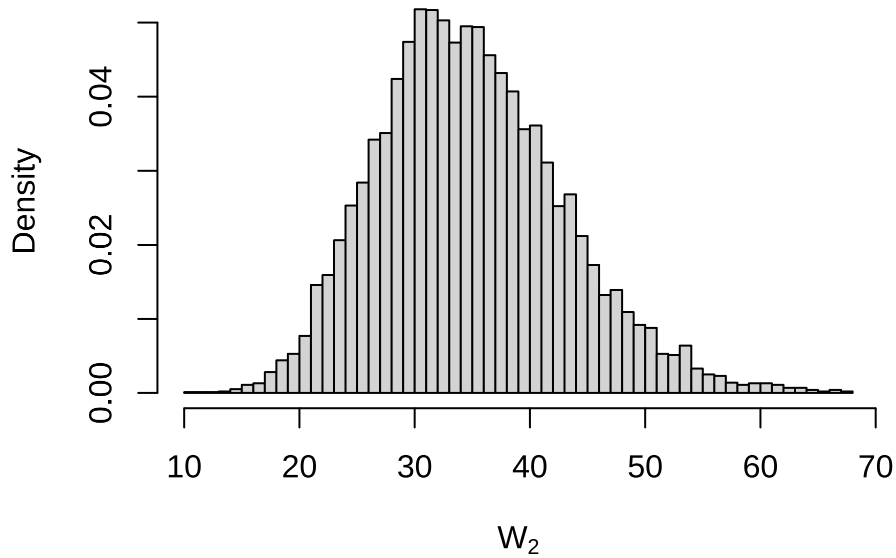

```
hist(Y, "FD", freq = FALSE, main = NA)
```




```
hist(W1, "FD", freq = FALSE, main = NA, xlab = expression(W[1]))
```




```
hist(W2, "FD", freq = FALSE, main = NA, xlab = expression(W[2]))
```




```
mean(25 + 0.5 * W1)
```

```
## [1] 48.99764
```

```
mean(18 + 0.5 * W2)
```

```
## [1] 35.43046
```

Slice Sampling

We want to generate a sample $X_1, X_2, \ldots, X_n$ following the PDF f with support S and $M = \max_{x \in S} f(x)$. Suppose that the application of the inverse transform method is impossible. Consider the random variables X, Y with joint PDF $f_{X,Y}(x,y) = \mathbb{1}_{\{y \leqslant f(x)\}}$ for $x \in S$ and $y \in [0, M]$. Then, we observe that:

$$f_X(x) = \int_0^M f_{X,Y}(x,y)dy = \int_0^{f(x)} 1dy = f(x), \quad f_{X|Y}(x \mid y) \propto \mathbb{1}_{\{x \in f^{-1}[y,\infty)\}},$$

where $f^{-1}[y, \infty) = \{x \in S : f(x) \geqslant y\}$. In other words, $X \sim f$, $(Y \mid X = x) \sim \text{Unif}\,[0, f(x)]$ and the conditional distribution of X given that $Y = y$ is uniform on the set $f^{-1}[y, \infty)$.

Algorithm 5.2 Slice Sampling

Input: PDF f, burn-in size b and sample size n.

1: We consider the initial value X_1.

2: For $i = 2, 3, \ldots, b + n$, we iterate the following steps:

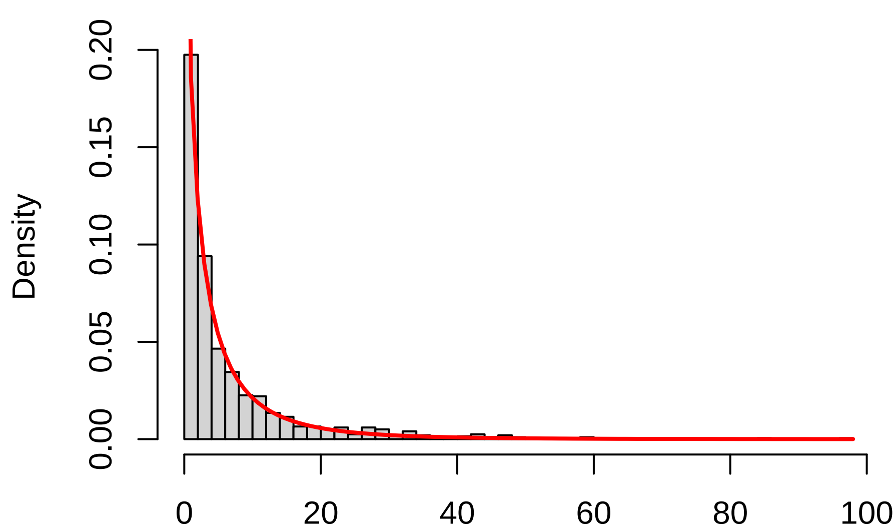
i: We generate $U \sim \text{Unif}[0, 1]$ and let $Y = f(X_{i-1})\, U \sim \text{Unif}\,[0, f(X_{i-1})]$.

ii: We generate X_i following the uniform distribution on $f^{-1}[Y, \infty)$.

Output: Random sample $X_{b+1}, X_{b+2}, \ldots, X_{b+n}$ following the PDF f.

Example 5.10. We want to generate a sample $X_1, X_2, \ldots, X_n$ following the PDF $f(x) = e^{-\sqrt{x}}/2$ for $x > 0$. For $y \in [0, 0.5]$, we observe that $f(x) \geqslant y \Leftrightarrow x \leqslant \log^2(2y)$. In other words, $(X \mid Y = y) \sim \text{Unif}\left[0, \log^2(2y)\right]$.

```
b = 1000
n = 1000
X = numeric(b + n)
for (i in 2:(b + n)) {
    U = runif(1)
    Y = exp(-sqrt(X[i - 1]))/2 * U
    V = runif(1)
    X[i] = log(2 * Y)^2 * V
}
X = X[-(1:b)]
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(exp(-sqrt(x))/2, add = TRUE, col = "red", lwd = 2)
```



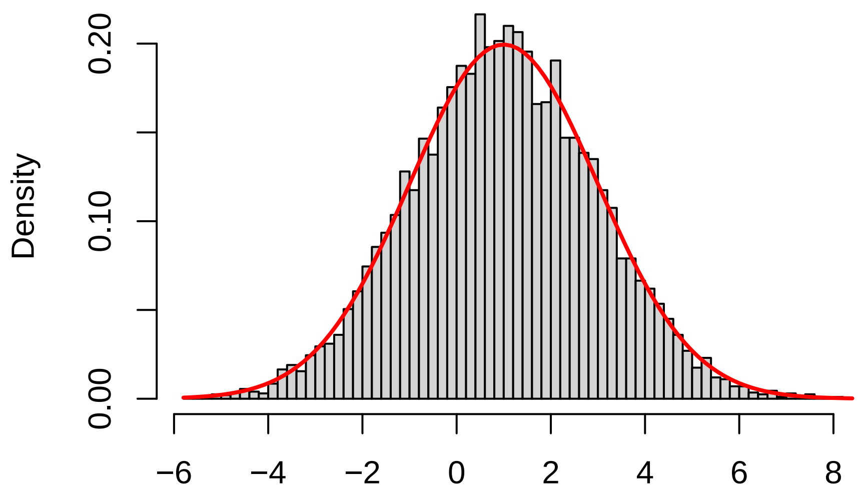
Example 5.11. We want to generate a sample $X_1, \ldots, X_n \sim \mathcal{N}\left(\mu, \sigma^2\right)$. For $y \in \left[0, \frac{1}{\sqrt{2\pi}\sigma}\right]$, we observe that:

$$f(x) \geqslant y \quad \Leftrightarrow \quad \mu - \sigma\sqrt{\log \frac{1}{2\pi\sigma^2 y^2}} \leqslant x \leqslant \mu + \sigma\sqrt{\log \frac{1}{2\pi\sigma^2 y^2}}.$$

In other words,

$$(X \mid Y=y) \sim \text{Unif}\left[\mu-\sigma\sqrt{\log\frac{1}{2\pi\sigma^2 y^2}}, \mu+\sigma\sqrt{\log\frac{1}{2\pi\sigma^2 y^2}}\right].$$

```
b = 10000
n = 10000
mu = 1
sigma = 2
X = numeric(b + n)
for (i in 2:(b + n)) {
    U = runif(1)
    Y = exp(-(X[i - 1] - mu)^2/(2 * sigma^2))/(sqrt(2 * pi) * sigma) * U
    V = runif(1)
    X[i] = 2 * sigma * sqrt(log(1/(2 * pi * sigma^2 * Y^2))) * V + mu - sigma *
        sqrt(log(1/(2 * pi * sigma^2 * Y^2)))
}
X = X[-(1:b)]
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(dnorm(x, mu, sigma), add = TRUE, col = "red", lwd = 2)
```



Metropolis-Hastings Algorithm

We want to generate a sample $X_1, X_2, \ldots, X_n$ following the PDF f. Suppose that the application of the inverse transform method or the slice sampling method is impossible.

The sequence of random variables $\{X_i\}$ constitutes a discrete-time Markov process with transition kernel $K(x_i \mid x_{i-1}) = g(x_i \mid x_{i-1})\, A(x_{i-1}, x_i)$.

Theorem 5.2. The Markov process $\{X_i\}$ with transition kernel $K(x_i \mid x_{i-1})$ is reversible and f is its unique stationary distribution.

Proof. If $\frac{f(x_i)}{f(x_{i-1})} < \frac{g(x_i|x_{i-1})}{g(x_{i-1}|x_i)}$, then it holds that:

$$A(x_{i-1}, x_i) = \frac{f(x_i) g(x_{i-1} \mid x_i)}{f(x_{i-1})\, g(x_i \mid x_{i-1})}, \quad A(x_i, x_{i-1}) = 1.$$

Algorithm 5.3 Metropolis-Hastings

Input: PDF f, conditional proposal PDF g, burn-in size b and sample size n.

1: We consider an initial value X_1.

2: For $i = 2, 3, \ldots, b+n$, we iterate the following steps:

i: We generate Y following the conditional PDF $g(y \mid X_{i-1})$.

ii: We generate $U \sim \text{Unif}[0,1]$ and calculate the acceptance probability:

$$A(X_{i-1}, Y) = \min\left\{\frac{f(Y) g(X_{i-1} \mid Y)}{f(X_{i-1}) g(Y \mid X_{i-1})}, 1\right\}.$$

iii: If $U < A(X_{i-1}, Y)$, then we let $X_i = Y$. Otherwise, we let $X_i = X_{i-1}$.

Output: Random sample $X_{b+1}, X_{b+2}, \ldots, X_{b+n}$ following the PDF f.

If $\frac{f(x_i)}{f(x_{i-1})} > \frac{g(x_i|x_{i-1})}{g(x_{i-1}|x_i)}$, then it holds that:

$$A(x_{i-1}, x_i) = 1, \quad A(x_i, x_{i-1}) = \frac{f(x_{i-1}) g(x_i \mid x_{i-1})}{f(x_i) g(x_{i-1} \mid x_i)}.$$

Therefore, we infer that:

$$f(x_{i-1}) g(x_i \mid x_{i-1}) \min\left\{\frac{f(x_i) g(x_{i-1} \mid x_i)}{f(x_{i-1}) g(x_i \mid x_{i-1})}, 1\right\} = f(x_i) g(x_{i-1} \mid x_i) \min\left\{\frac{f(x_{i-1}) g(x_i \mid x_{i-1})}{f(x_i) g(x_{i-1} \mid x_i)}, 1\right\},$$

$$\begin{aligned} f(x_{i-1}) K(x_i \mid x_{i-1}) &= f(x_{i-1}) g(x_i \mid x_{i-1}) A(x_{i-1}, x_i) \\ &= f(x_i) g(x_{i-1} \mid x_i) A(x_i, x_{i-1}) = f(x_i) K(x_{i-1} \mid x_i). \end{aligned}$$

Since the function f satisfies the detailed balance equations for the Markov process $\{X_i\}$, we conclude that the process $\{X_i\}$ is reversible with unique stationary distribution f. $\square$

Example 5.12. We want to generate a sample $X_1, \ldots, X_n \sim \text{Bin}(k, p)$. We consider the following conditional proposal PMF:

$$g(y \mid x) = \begin{cases} 0.5, & x \in \{1, 2, \ldots, k-1\}, \quad y = x+1 \\ 0.5, & x \in \{1, 2, \ldots, k-1\}, \quad y = x-1 \\ 1, & x = 0, \quad y = 1 \\ 1, & x = k, \quad y = k-1 \end{cases},$$

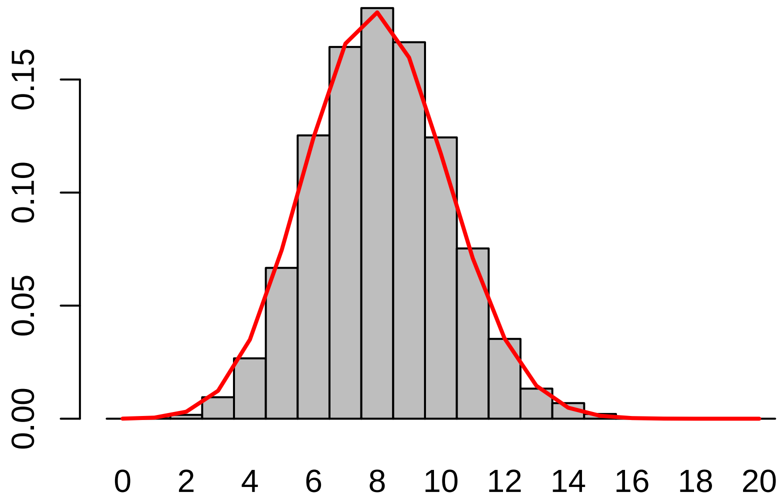
i.e. the symmetric random walk on the state-space $S = \{0, 1, \ldots, k\}$ with reflecting barriers at states $\{0\}$ and $\{k\}$.

Then, we calculate that:

$$\frac{f(y)g\left(x \mid y\right)}{f\left(x\right)g\left(y \mid x\right)} = \begin{cases} \frac{k-x}{x+1}\frac{p}{1-p}, & x \in \{1,2,\dots,k-2\}, \quad y = x+1 \\ \frac{x}{k-x+1}\frac{1-p}{p}, & x \in \{2,3,\dots,k-1\}, \quad y = x-1 \\ \frac{k}{2}\frac{p}{1-p}, & x = 0, \quad y = 1 \\ \frac{2}{k}\frac{1-p}{p}, & x = 1, \quad y = 0 \\ \frac{2}{k}\frac{p}{1-p}, & x = k-1, \quad y = k \\ \frac{k}{2}\frac{1-p}{p}, & x = k, \quad y = k-1 \end{cases}.$$

```r
b = 10000
n = 10000
k = 20
p = 0.4
X = numeric(b + n)
for (i in 2:(b + n)) {
    if (X[i - 1] == 0) {
        Y = 1
        A = k/2 * p/(1 - p)
    } else if (X[i - 1] == k) {
        Y = k - 1
        A = k/2 * (1 - p)/p
    } else {
        V = runif(1)
        if (V < 0.5) {
            Y = X[i - 1] + 1
            if (Y == k) {
                A = 2/k * p/(1 - p)
            } else {
                A = (k - X[i - 1])/(X[i - 1] + 1) * p/(1 - p)
            }
        } else {
            Y = X[i - 1] - 1
            if (Y == 0) {
                A = 2/k * (1 - p)/p
            } else {
                A = X[i - 1]/(k - X[i - 1] + 1) * (1 - p)/p
            }
        }
    }
    U = runif(1)
    X[i] = ifelse(U < A, Y, X[i - 1])
}
```

```r
X = X[-(1:b)]
barplot(table(factor(X, levels = 0:k))/n, space = 0)
lines(0:k + 0.5, dbinom(0:k, k, p), col = "red", lwd = 2)
```



Example 5.13. Let $X \sim \text{Poisson}(\lambda)$ be a random variable. We want to generate a sample $X_1, X_2, \ldots, X_n$ following the conditional distribution of X given that $X \leqslant k$. For $x \in \{0, 1, \ldots, k\}$, we observe that $f_{X|X\leqslant k}(x) \propto \frac{\lambda^x}{x!}$. We consider the following conditional proposal PMF:

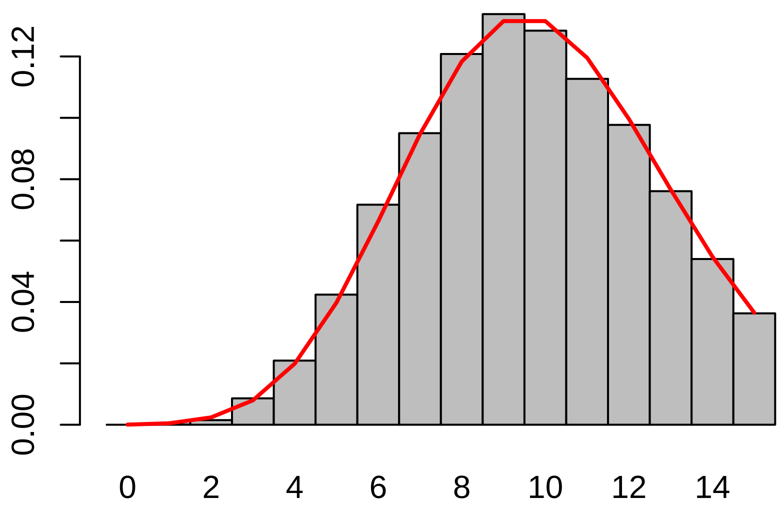
$$g(y \mid x) = \begin{cases} 0.5, & x \in \{0, 1, \ldots, k-1\}, \quad y = x+1 \\ 0.5, & x \in \{1, 2, \ldots, k\}, \quad y = x-1 \\ 0.5, & x = 0, \quad y = 0 \\ 0.5, & x = k, \quad y = k \end{cases},$$

i.e. the symmetric random walk on the state-space $S = \{0, 1, \ldots, k\}$ with elastic barriers at states $\{0\}$ and $\{k\}$. Then, we calculate that:

$$\frac{f(y)g(x \mid y)}{f(x)\,g(y \mid x)} = \begin{cases} \frac{\lambda}{x+1}, & x \in \{0, 1, \ldots, k-1\}, \quad y = x+1 \\ \frac{x}{\lambda}, & x \in \{1, 2, \ldots, k\}, \quad y = x-1 \\ 1, & x = y = 0 \\ 1, & x = y = k \end{cases}.$$

```r
b = 10000
n = 10000
lambda = 10
k = 15
X = numeric(b + n)
for (i in 2:(b + n)) {
    V = runif(1)
    if (X[i - 1] == 0) {
        if (V < 0.5) {
```

```
            Y = 0
            A = 1
        } else {
            Y = 1
            A = lambda
        }
    } else if (X[i - 1] == k) {
        if (V < 0.5) {
            Y = k
            A = 1
        } else {
            Y = k - 1
            A = k/lambda
        }
    } else {
        if (V < 0.5) {
            Y = X[i - 1] + 1
            A = lambda/(X[i - 1] + 1)
        } else {
            Y = X[i - 1] - 1
            A = X[i - 1]/lambda
        }
    }
    U = runif(1)
    X[i] = ifelse(U < A, Y, X[i - 1])
}
X = X[-(1:b)]
barplot(table(factor(X, levels = 0:k))/n, space = 0)
lines(0:k + 0.5, dpois(0:k, lambda)/ppois(k, lambda), col = "red", lwd = 2)
```



Example 5.14. We want to generate a sample $X_1, X_2, \ldots, X_n$ following the PDF:

$$f(x) = \frac{1}{2\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} + \frac{1}{2\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x+\mu)^2\right\}.$$

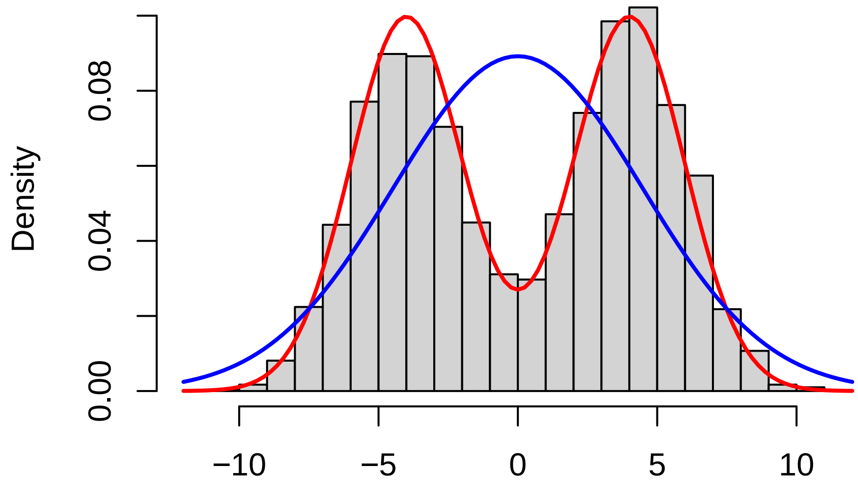
We consider the proposal random variable $Y \sim \mathcal{N}\left(0, \sigma^2 + \mu^2\right)$ with PDF:

$$g(y) = \frac{1}{\sqrt{2\pi\left(\sigma^2+\mu^2\right)}} \exp\left\{-\frac{1}{2\left(\sigma^2+\mu^2\right)} y^2\right\}.$$

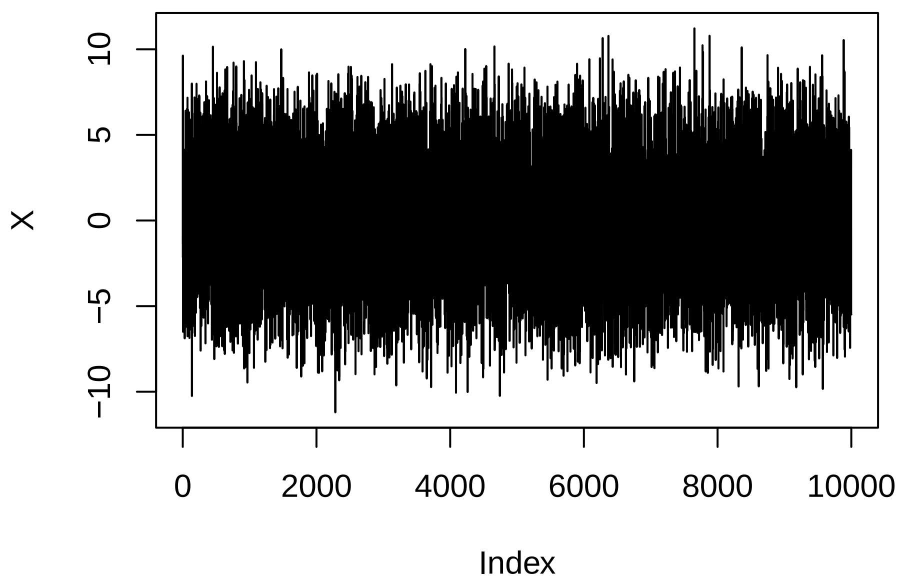
We observe that the proposal PDF doesn't depend on the current state of the Markov process $\{X_i\}$.

```
b = 10000
n = 10000
mu = 4
sigma = 2
lambda = 1/sqrt(sigma^2 + mu^2)
M = sqrt(2 * exp(1)/pi)
accept = 0
X = numeric(b + n)
for (i in 2:(b + n)) {
    W = runif(1)
    Y = ifelse(W <= 0.5, log(2 * W)/lambda, -log(2 * (1 - W))/lambda)
    U = runif(1)
    V = M * dexp(abs(Y), lambda)/2 * U
    while (dnorm(Y, 0, sqrt(sigma^2 + mu^2)) < V) {
        W = runif(1)
        Y = ifelse(W <= 0.5, log(2 * W)/lambda, -log(2 * (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(Y), lambda)/2 * U
    }
    A = (0.5 * dnorm(Y, mu, sigma) + 0.5 * dnorm(Y, -mu, sigma)) * dnorm(X[i -
        1], 0, sqrt(sigma^2 + mu^2))/((0.5 * dnorm(X[i - 1], mu, sigma) + 0.5 *
        dnorm(X[i - 1], -mu, sigma)) * dnorm(Y, 0, sqrt(sigma^2 + mu^2)))
    U = runif(1)
    if (U < A) {
        X[i] = Y
        if (i > b) {
            accept = accept + 1
        }
    } else {
        X[i] = X[i - 1]
    }
}
X = X[-(1:b)]
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
```

```r
curve(0.5 * dnorm(x, mu, sigma) + 0.5 * dnorm(x, -mu, sigma), add = TRUE, col = "red",
    lwd = 2)
curve(dnorm(x, 0, sqrt(sigma^2 + mu^2)), add = TRUE, col = "blue", lwd = 2)
```




```r
plot(X, type = "l")
```




```r
print(accept/n)
```

```
## [1] 0.6781
```

For the correct implementation of the Metropolis-Hastings algorithm with proposal PDF which doesn't depend on the current state of the Markov process $\{X_i\}$, we must select a suitable proposal density, so that the percentage of accepted proposal states of the algorithm is close to 100%.

If it holds that $g(y \mid x_{i-1}) = g(x_{i-1} \mid y)$ for the conditional proposal PDF, then the algorithm is called **Random Walk Metropolis-Hastings**. We observe that:

$$A(x_{i-1}, y) = \min\left\{\frac{f(y)}{f(x_{i-1})}, 1\right\}.$$

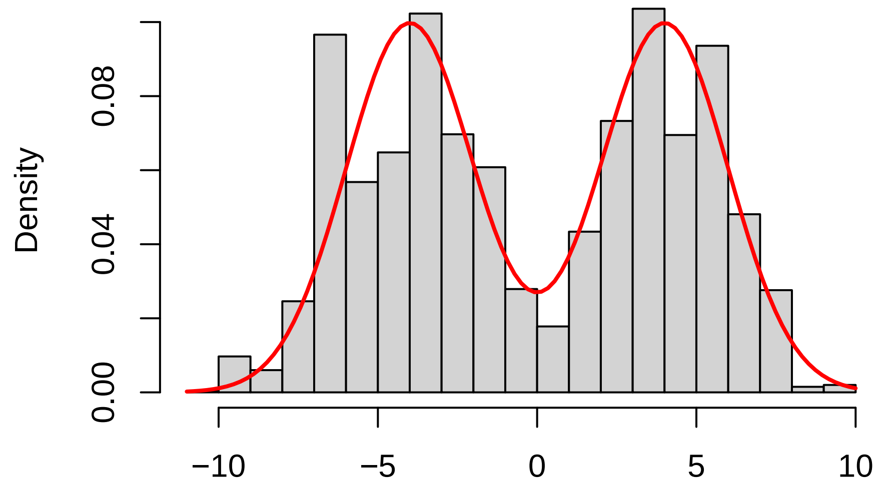
If $f(y) > f(x_{i-1})$, then we infer that $A(x_{i-1}, y) = 1$. In other words, the Markov process transitions to a proposal state with higher density than the current one with probability 1. If the proposal state has lower density than the current one, then the Markov process transitions to it with probability $A(x_{i-1}, y) \in (0, 1)$.

We consider the proposal random variable $(Y \mid X_{i-1} = x_{i-1}) \sim \mathcal{N}\left(x_{i-1}, \sigma_p^2\right)$ with conditional PDF:

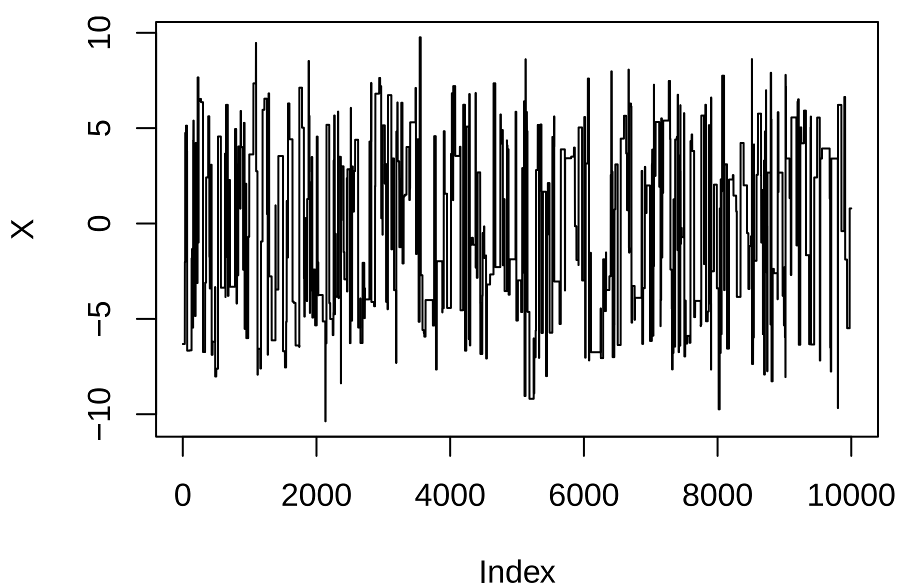
$$g(y \mid x_{i-1}) = \frac{1}{\sqrt{2\pi}\sigma_p} \exp\left\{-\frac{1}{2\sigma_p^2}(y - x_{i-1})^2\right\}.$$

```
b = 10000
n = 10000
mu = 4
sigma = 2
sigmap = 100
lambda = 1/sigmap
M = sqrt(2 * exp(1)/pi)
accept = 0
X = numeric(b + n)
for (i in 2:(b + n)) {
    W = runif(1)
    Y = ifelse(W <= 0.5, X[i - 1] + log(2 * W)/lambda, X[i - 1] - log(2 * (1 -
        W))/lambda)
    U = runif(1)
    V = M * dexp(abs(Y - X[i - 1]), lambda)/2 * U
    while (dnorm(Y, X[i - 1], sigmap) < V) {
        W = runif(1)
        Y = ifelse(W <= 0.5, X[i - 1] + log(2 * W)/lambda, X[i - 1] - log(2 *
            (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(Y - X[i - 1]), lambda)/2 * U
    }
    A = (0.5 * dnorm(Y, mu, sigma) + 0.5 * dnorm(Y, -mu, sigma))/(0.5 * dnorm(X[i -
        1], mu, sigma) + 0.5 * dnorm(X[i - 1], -mu, sigma))
    U = runif(1)
    if (U < A) {
        X[i] = Y
        if (i > b) {
            accept = accept + 1
        }
    } else {
        X[i] = X[i - 1]
    }
}
X = X[-(1:b)]
```

```r
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(0.5 * dnorm(x, mu, sigma) + 0.5 * dnorm(x, -mu, sigma), add = TRUE, col = "red",
    lwd = 2)
```




```r
plot(X, type = "l")
```



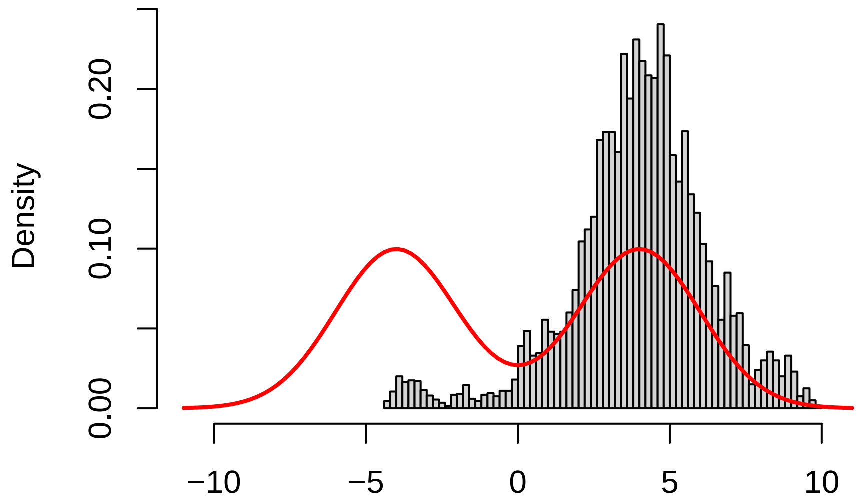

```r
print(accept/n)
```

```
## [1] 0.0502
```

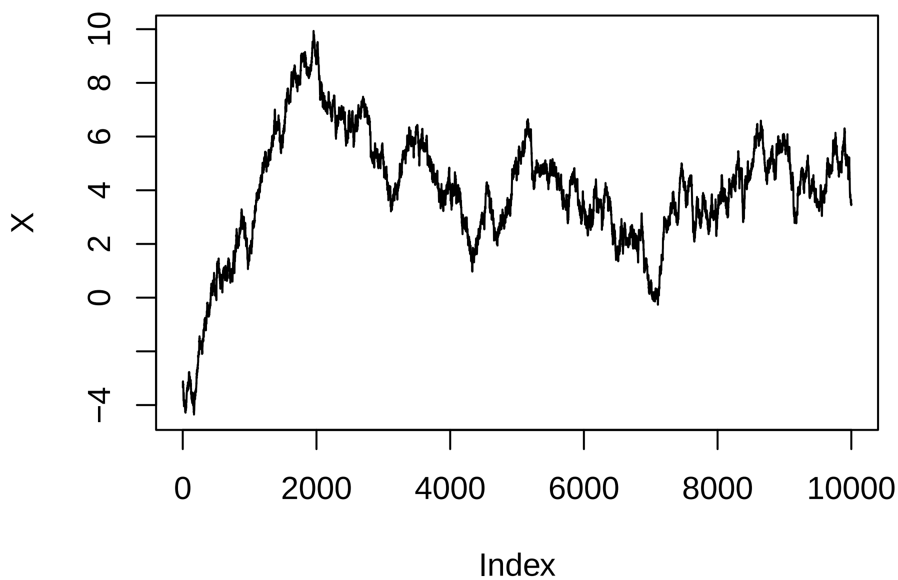
When the variance σ_p^2 of the proposal density is too high, then we observe that the algorithm proposes states too far away from the current state of the Markov process. These states have really low density, so there's a very small probability that the Markov process transitions to them. Consequently, the Markov process is trapped in the same state for large periods of time and doesn't adequately explore the entire support of the PDF f.

```r
b = 10000
n = 10000
mu = 4
sigma = 2
```

```r
sigmap = 0.1
lambda = 1/sigmap
M = sqrt(2 * exp(1)/pi)
accept = 0
X = numeric(b + n)
for (i in 2:(b + n)) {
    W = runif(1)
    Y = ifelse(W <= 0.5, X[i - 1] + log(2 * W)/lambda, X[i - 1] - log(2 * (1 -
        W))/lambda)
    U = runif(1)
    V = M * dexp(abs(Y - X[i - 1]), lambda)/2 * U
    while (dnorm(Y, X[i - 1], sigmap) < V) {
        W = runif(1)
        Y = ifelse(W <= 0.5, X[i - 1] + log(2 * W)/lambda, X[i - 1] - log(2 *
            (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(Y - X[i - 1]), lambda)/2 * U
    }
    A = (0.5 * dnorm(Y, mu, sigma) + 0.5 * dnorm(Y, -mu, sigma))/(0.5 * dnorm(X[i -
        1], mu, sigma) + 0.5 * dnorm(X[i - 1], -mu, sigma))
    U = runif(1)
    if (U < A) {
        X[i] = Y
        if (i > b) {
            accept = accept + 1
        }
    } else {
        X[i] = X[i - 1]
    }
}
X = X[-(1:b)]
hist(X, "FD", freq = FALSE, main = NA, xlim = c(-11, 11), xlab = NA)
curve(0.5 * dnorm(x, mu, sigma) + 0.5 * dnorm(x, -mu, sigma), add = TRUE, col = "red",
    lwd = 2)
```




```
plot(X, type = "l")
```



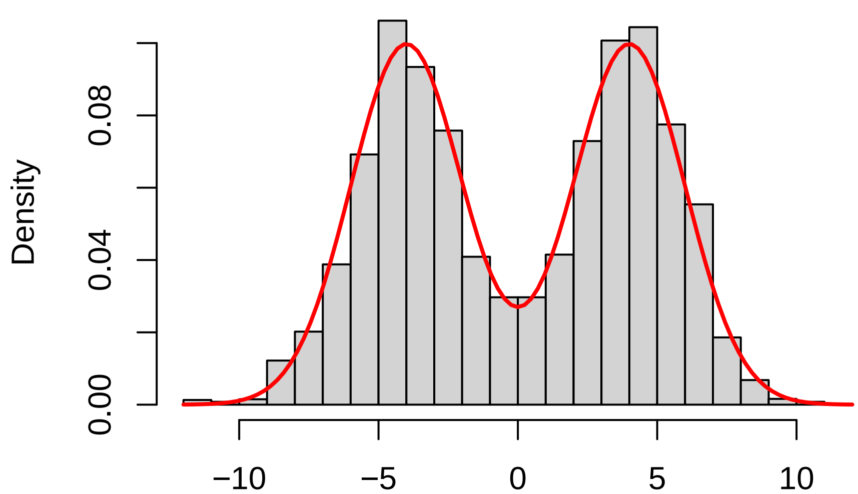

```
print(accept/n)
```

```
## [1] 0.9874
```

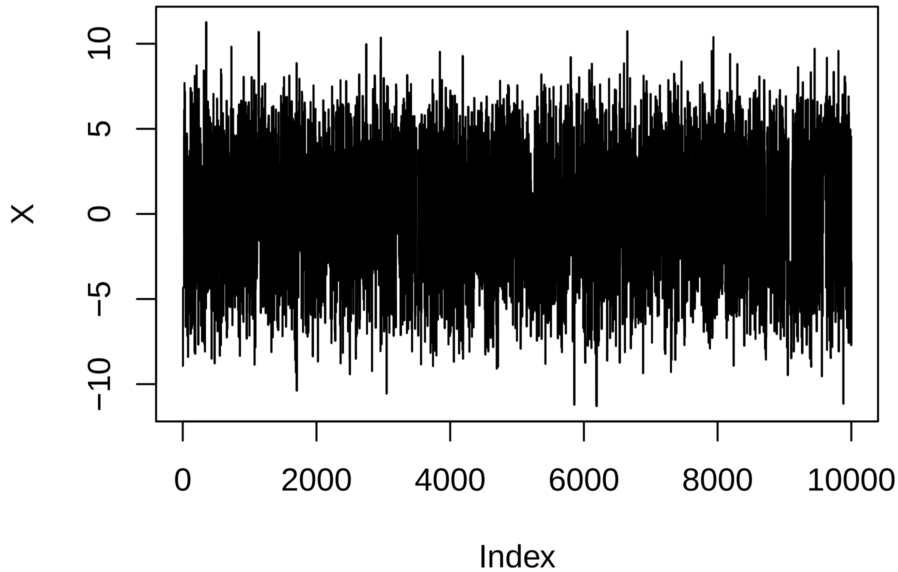
When the variance σ_p^2 of the proposal PDF is too low, then we observe that the algorithm proposes states very close to the current state of the Markov process. Since the Markov process transitions to proposal states with higher density than the current state with probability 1, it tends to transition towards the closest mode of the PDF f. Consequently, the Markov process is trapped around a mode of f and doesn't adequately explore the entire support of f.

```
b = 10000
n = 10000
mu = 4
sigma = 2
sigmap = 10
lambda = 1/sigmap
M = sqrt(2 * exp(1)/pi)
```

```r
accept = 0
X = numeric(b + n)
for (i in 2:(b + n)) {
    W = runif(1)
    Y = ifelse(W <= 0.5, X[i - 1] + log(2 * W)/lambda, X[i - 1] - log(2 * (1 -
        W))/lambda)
    U = runif(1)
    V = M * dexp(abs(Y - X[i - 1]), lambda)/2 * U
    while (dnorm(Y, X[i - 1], sigmap) < V) {
        W = runif(1)
        Y = ifelse(W <= 0.5, X[i - 1] + log(2 * W)/lambda, X[i - 1] - log(2 *
            (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(Y - X[i - 1]), lambda)/2 * U
    }
    A = (0.5 * dnorm(Y, mu, sigma) + 0.5 * dnorm(Y, -mu, sigma))/(0.5 * dnorm(X[i -
        1], mu, sigma) + 0.5 * dnorm(X[i - 1], -mu, sigma))
    U = runif(1)
    if (U < A) {
        X[i] = Y
        if (i > b) {
            accept = accept + 1
        }
    } else {
        X[i] = X[i - 1]
    }
}
X = X[-(1:b)]
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(0.5 * dnorm(x, mu, sigma) + 0.5 * dnorm(x, -mu, sigma), add = TRUE, col = "red",
    lwd = 2)
```



```r
plot(X, type = "l")
```




```r
print(accept/n)
```

```
## [1] 0.4041
```

For the correct implementation of the Random Walk Metropolis-Hastings algorithm, we must select a suitable value for the variance σ_p^2 of the proposal PDF, so that the percentage of accepted proposal states of the algorithm is close to 50%.

Data Augmentation

We want to generate a sample $X_1, X_2, \ldots, X_n$ following the PDF f. Suppose that the application of the inverse transform method is impossible. Consider two random variables $X \sim f$ and Y. First, suppose that it's easy to simulate from the conditional PDF $f_{X|Y}$. Then, we calculate that:

$$f_{Y|X}(y \mid x) = \frac{f_Y(y) f_{X|Y}(x \mid y)}{f(x)} \propto f_Y(y) f_{X|Y}(x \mid y).$$

Alternatively, suppose that it's easy to simulate from the conditional PDF $f_{Y|X}$. Then, we calculate that:

$$f_{X|Y}(x \mid y) = \frac{f_X(x) f_{Y|X}(y \mid x)}{f(y)} \propto f_X(x) f_{Y|X}(y \mid x).$$

In either of these two cases, we can then proceed with applying a Gibbs sampler to alternatively simulate from the conditional distributions of X given Y and Y given X.

Example 5.15. We want to generate a sample $X_1, X_2, \ldots, X_n$ following the PDF:

$$f(x) = \frac{1}{2\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} + \frac{1}{2\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x+\mu)^2\right\}.$$

Consider two random variables $X \sim f$ and $Y \sim \text{Bernoulli}(0.5)$. Suppose that $(X \mid Y = 0) \sim \mathcal{N}\left(\mu, \sigma^2\right)$ and

Algorithm 5.4 Data Augmentation

Input: Conditional PDFs $f_{X|Y}$, $f_{Y|X}$, burn-in size b and sample size n.

1: We consider an initial value X_1.

2: For $i = 2, 3, \ldots, b+n$, we iterate the following steps:

i: We generate Y following the conditional PDF $f_{Y|X}(y \mid X_{i-1})$.

ii: We generate X_i following the conditional PDF $f_{X|Y}(x \mid Y)$.

Output: Random sample $X_{b+1}, X_{b+2}, \ldots, X_{b+n}$ following the PDF f.

$(X \mid Y = 1) \sim \mathcal{N}\left(-\mu, \sigma^2\right)$. Then, we calculate that:

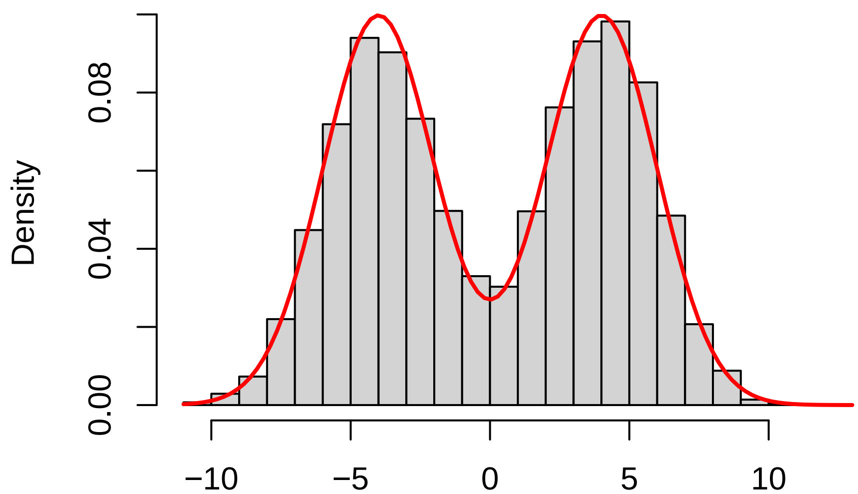
$$\mathbb{P}(Y = 0 \mid X = x) \propto \mathbb{P}(Y = 0) f_{X|Y}(x \mid 0) \propto \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\},$$

$$\mathbb{P}(Y = 1 \mid X = x) \propto \mathbb{P}(Y = 1) f_{X|Y}(x \mid 1) \propto \exp\left\{-\frac{1}{2\sigma^2}(x + \mu)^2\right\},$$

$$(Y \mid X = x) \sim \text{Bernoulli}\left(\frac{\exp\left\{-\frac{1}{2\sigma^2}(x+\mu)^2\right\}}{\exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} + \exp\left\{-\frac{1}{2\sigma^2}(x+\mu)^2\right\}}\right) \equiv \text{Bernoulli}\left(\frac{1}{e^{2\mu x/\sigma^2} + 1}\right).$$

```
b = 10000
n = 10000
mu = 4
sigma = 2
lambda = 1/sigma
M = sqrt(2 * exp(1)/pi)
X = numeric(b + n)
for (i in 2:(b + n)) {
    U = runif(1)
    Y = U < 1/(exp(2 * mu * X[i - 1]/sigma^2) + 1)
    if (Y == 0) {
        W = runif(1)
        X[i] = ifelse(W <= 0.5, mu + log(2 * W)/lambda, mu - log(2 * (1 - W))/lambda)
        U = runif(1)
        V = M * dexp(abs(X[i] - mu), lambda)/2 * U
        while (dnorm(X[i], mu, sigma) < V) {
            W = runif(1)
            X[i] = ifelse(W <= 0.5, mu + log(2 * W)/lambda, mu - log(2 * (1 -
                W))/lambda)
            U = runif(1)
            V = M * dexp(abs(X[i] - mu), lambda)/2 * U
        }
    } else {
        W = runif(1)
```

```
        X[i] = ifelse(W <= 0.5, -mu + log(2 * W)/lambda, -mu - log(2 * (1 -
            W))/lambda)
        U = runif(1)
        V = M * dexp(abs(X[i] + mu), lambda)/2 * U
        while (dnorm(X[i], -mu, sigma) < V) {
            W = runif(1)
            X[i] = ifelse(W <= 0.5, -mu + log(2 * W)/lambda, -mu - log(2 * (1 -
                W))/lambda)
            U = runif(1)
            V = M * dexp(abs(X[i] + mu), lambda)/2 * U
        }
    }
}
X = X[-(1:b)]
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
curve(0.5 * dnorm(x, mu, sigma) + 0.5 * dnorm(x, -mu, sigma), add = TRUE, col = "red",
    lwd = 2)
```



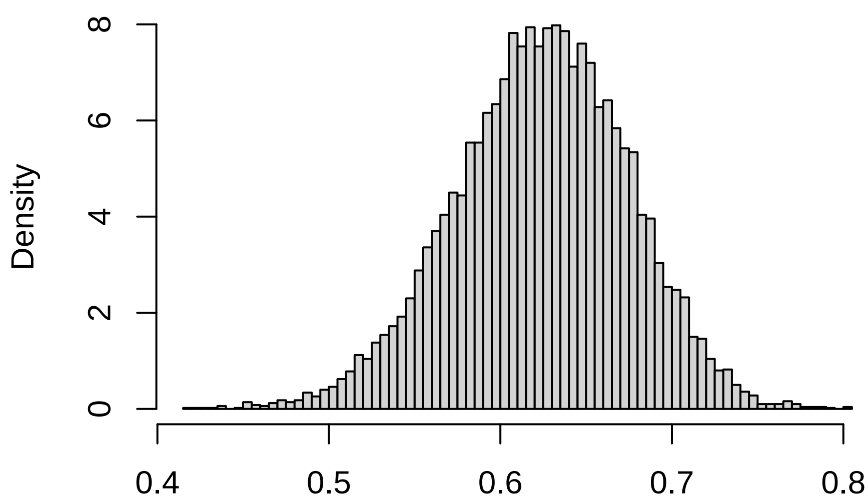
Example 5.16. We want to generate a sample $X_1, X_2, \ldots, X_n$ following the PDF $f(x) \propto x^{34}(1-x)^{38}(2+x)^{125}$ for $x \in [0,1]$. Consider two random variables $X \sim f$ and Y. Suppose that $(Y \mid X = x) \sim \text{Bin}\left(125, \frac{x}{2+x}\right)$, For $y \in \{0, 1, \ldots, 125\}$, we calculate that:

$$f_{X|Y}(x \mid y) \propto f(x) f_{Y|X}(y \mid x)$$
$$\propto x^{34}(1-x)^{38}(2+x)^{125} \binom{125}{y} \left(\frac{x}{2+x}\right)^y \left(1 - \frac{x}{2+x}\right)^{125-y} \propto x^{y+34}(1-x)^{38}.$$

In other words, $(X \mid Y = y) \sim \text{Beta}(y + 35, 39)$.

```
b = 10000
n = 10000
X = numeric(b + n)
for (i in 2:(b + n)) {
```

```
    U = runif(125)
    Y = sum(U < X[i - 1]/(2 + X[i - 1]))
    M = dbeta((Y + 34)/(Y + 72), Y + 35, 39)
    X[i] = runif(1)
    U = runif(1)
    V = M * U
    while (dbeta(X[i], Y + 35, 39) < V) {
        X[i] = runif(1)
        U = runif(1)
        V = M * U
    }
}
X = X[-(1:b)]
hist(X, "FD", freq = FALSE, main = NA, xlab = NA)
```



Simulated Annealing

We want to maximize the function $h : S \to \mathbb{R}$ with $\int_S e^{h(x)}dx < \infty$. We define the maximum value $h^* = \max_{x \in S} h(x)$ and the set of maxima $M = \{x \in S : h(x) = h^*\}$. For $i \in \mathbb{N}$, we consider the following PDF:

$$f_i(x) \propto e^{\lambda_i h(x)} \propto e^{\lambda_i [h(x) - h^*]}.$$

We observe that $h(x) - h^* < 0$ for $x \notin M$ and $h(x) - h^* = 0$ for $x \in M$. If $\lambda_i \to \infty$, then we infer that:

$$\lim_{i \to \infty} f_i(x) \propto \mathbb{1}_{\{x \in M\}}.$$

If $X_i \sim f_i$ for $i \in \mathbb{N}$, the sequence of random variables $\{X_i\}$ converges in distribution to a random variable which follows the uniform distribution on the set M. In order to simulate from the PDF f_i we usually apply a Random Walk Metropolis-Hastings algorithm with $\lambda_i = \lambda_1 \log i$ or $\lambda_i = \lambda_1 r^{i-1}$ for $\lambda_1 > 0$ and $r > 1$.

Example 5.17. We want to maximize the function $h(x) = [\cos(50x) + \sin(20x)]^2$ for $x \in [0, 1]$. We consider the

Algorithm 5.5 Simulated Annealing

Input: Function h, conditional proposal PDF g, sequence λ_i and sample size n.

1: We consider an initial value X_1.

2: For $i = 2, 3, \dots, n$, we iterate the following steps:

i: We generate Y following the conditional PDF $g\left(y \mid X_{i-1}\right)$.

ii: We generate $U \sim \text{Unif}[0, 1]$ and calculate the acceptance probability:

$$A\left(X_{i-1}, Y\right) = \min\left\{e^{\lambda_i\left[h(Y) - h\left(X_{i-1}\right)\right]}, 1\right\}.$$

iii: If $U < A\left(X_{i-1}, Y\right)$, then we let $X_i = Y$. Otherwise, we let $X_i = X_{i-1}$.

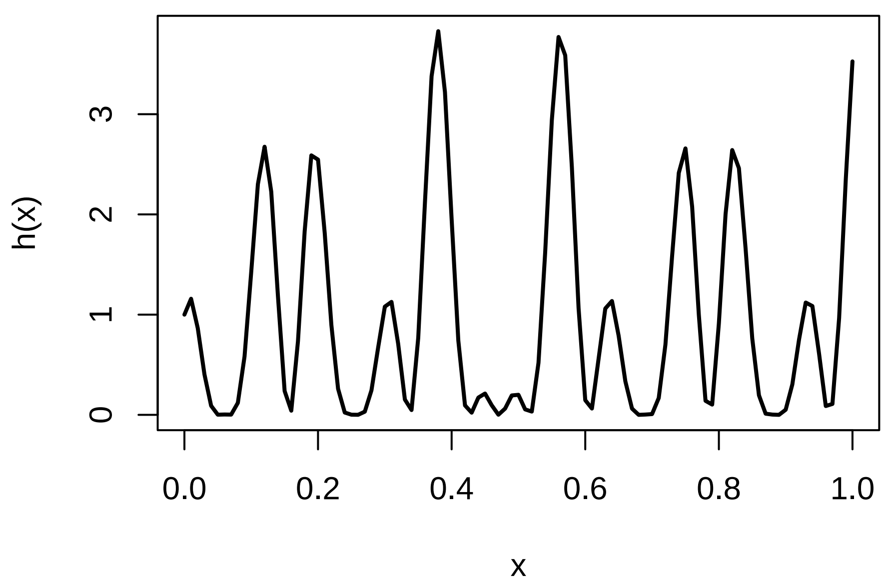
Output: Maximum value $h^* = \max_i h(X_i)$.

proposal random variable $\left(Y \mid X_{i-1} = x_{i-1}\right) \sim \text{Unif}\left[x_{i-1} - s_i, x_{i-1} + s_i\right]$ with the following conditional PDF:

$$g\left(y \mid x_{i-1}\right) = \frac{1}{2s_i}.$$

```
h = function(x) {
    ifelse(x >= 0 & x <= 1, (cos(50 * x) + sin(20 * x))^2, 0)
}

curve(h(x), lwd = 2)
```

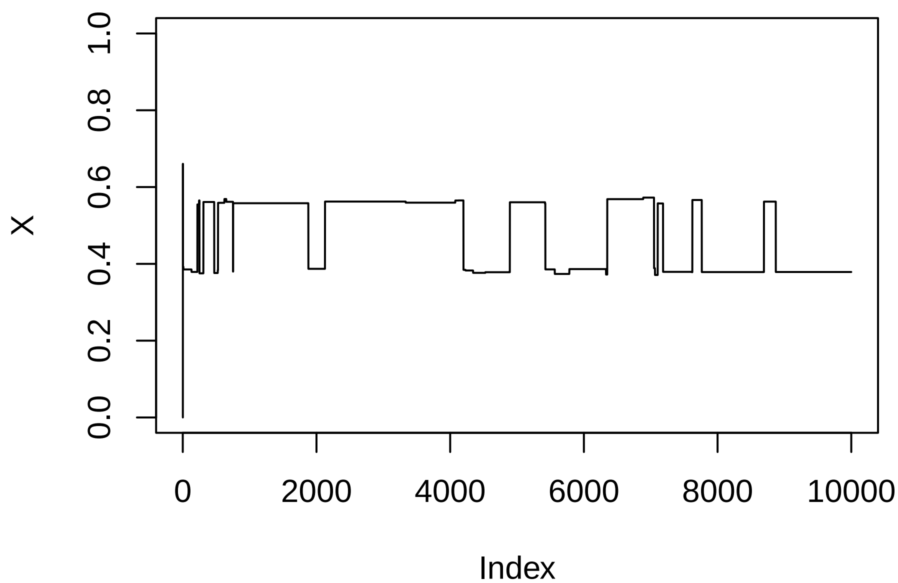



```
optimize(h, c(0, 1), maximum = TRUE)
```

```
## $maximum
## [1] 0.379125
##
## $objective
```

```
## [1] 3.832543
```

```r
n = 10000
lambda1 = 1
X = numeric(n)
for (i in 2:n) {
    lambda = lambda1 * log(i)
    s = sqrt(lambda)
    V = runif(1)
    Y = 2 * s * V + X[i - 1] - s
    logA = lambda * (h(Y) - h(X[i - 1]))
    U = runif(1)
    X[i] = ifelse(log(U) < logA, Y, X[i - 1])
}
plot(X, type = "l", ylim = c(0, 1))
```




```r
hstar = max(h(X))
print(hstar)
```

```
## [1] 3.832543
```

```r
I = which(h(X) == hstar)
print(unique(X[I]))
```

```
## [1] 0.3791546
```

6 Bootstrap Method

Definition 6.1. Let $X_1, X_2, \ldots, X_n$ be a random sample. For $x \in \mathbb{R}$, we define the empirical CDF of the random sample as follows:

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{X_i \leqslant x\}}.$$

Let $X_1, X_2, \ldots, X_n$ be a random sample following the CDF F with parameter θ and an estimator $T(X)$ of θ. We want to estimate the expected value $\mathbb{E}[h(T)]$. Suppose that it's not efficient to simulate from the CDF F. On the contrary, we can easily generate random samples $X_*^{(1)}, X_*^{(2)}, \ldots, X_*^{(B)}$ following the empirical CDF F_n of the random sample $X_1, X_2, \ldots, X_n$. If $T_j^* = T\left(X_*^{(j)}\right)$ for $j = 1, 2, \ldots, B$, then we can estimate the expected value $\mathbb{E}[h(T)]$ as follows:

$$\frac{1}{B} \sum_{j=1}^{B} h\left(T_j^*\right).$$

Algorithm 6.1 Non-Parametric Bootstrap

Input: Random sample X, statistic $T(X)$ and bootstrap sample size B.

For $j = 1, 2, \ldots, B$, we iterate the following steps:

i: We generate $U_1, \ldots, U_n \sim \text{Unif}[0, 1]$ and let $I_i = \lfloor nU_i \rfloor + 1$ for $i = 1, 2, \ldots, n$.

ii: We let $X_*^{(j)} = (X_{I_1}, X_{I_2}, \ldots, X_{I_n})$ and $T_j^* = T\left(X_*^{(j)}\right)$.

Output: Bootstrap statistics $T_1^*, T_2^*, \ldots, T_B^*$.

Example 6.1. Let $U_1, U_2, \cdots \sim \text{Unif}[0, 1]$ be a sequence of independent random variables. We define the random variable:

$$X = \sup\{k \in \mathbb{N} : U_1 < U_2 < \cdots < U_{k-1}\}.$$

Consider the parameter $\theta = \sqrt{\mathbb{E}(X^2)}$ and a random sample $X_1, X_2, \ldots, X_n$ from the distribution of X. We define an estimator of θ as follows:

$$T(X) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} X_i^2}.$$

We want to estimate the expected value, the variance, the bias and the mean squared error of the estimator $T(X)$. Additionally, we want to construct a $100(1-\alpha)\%$ confidence interval for θ.

```
X = c(2, 4, 3, 2, 2, 2, 2, 3, 3, 2)
n = length(X)
t = sqrt(mean(X^2))
B = 10000
Tstar = numeric(B)
for (j in 1:B) {
    U = runif(n)
    I = floor(n * U) + 1
    Xstar = X[I]
    Tstar[j] = sqrt(mean(Xstar^2))
```

```r
}
Tbar = mean(Tstar)
print(Tbar)
```

```
## [1] 2.577908
```

```r
VarT = mean((Tstar - Tbar)^2)
print(VarT)
```

```
## [1] 0.05363952
```

```r
bias = mean(Tstar - t)
print(bias)
```

```
## [1] -0.01052766
```

```r
MSE = mean((Tstar - t)^2)
print(MSE)
```

```
## [1] 0.05375035
```

We define:

$$\overline{T}^* = \frac{1}{B}\sum_{j=1}^{b} T_j^*, \quad S_*^2 = \frac{1}{B}\sum_{j=1}^{B}\left(T_j^* - \overline{T}^*\right)^2.$$

Then, we get the following bootstrap normal confidence interval for θ:

$$\left[\overline{T}^* - z_{1-\alpha/2}S_*, \overline{T}^* + z_{1-\alpha/2}S_*\right].$$

Alternatively, we can use the following bootstrap percentile confidence interval for θ:

$$\left[T^*_{(\lfloor B\alpha/2\rfloor)}, T^*_{(\lceil B(1-\alpha/2)\rceil)}\right].$$

Finally, we can use the following bootstrap pivotal confidence interval for θ:

$$\left[2T - T^*_{(\lceil B(1-\alpha/2)\rceil)}, 2T - T^*_{(\lfloor B\alpha/2\rfloor)}\right].$$

```r
alpha = 0.05
I = c(Tbar - qnorm(1 - alpha/2) * sqrt(VarT), Tbar + qnorm(1 - alpha/2) * sqrt(VarT))
print(I)
```

```
## [1] 2.123976 3.031840
```

```r
Tstar = sort(Tstar)
I = c(Tstar[floor(B * alpha/2)], Tstar[ceiling(B * (1 - alpha/2))])
print(I)
```

```
## [1] 2.121320 3.016621
```

```r
I = c(2 * t - Tstar[ceiling(B * (1 - alpha/2))], 2 * t - Tstar[floor(B * alpha/2)])
print(I)
```

```
## [1] 2.160251 3.055551
```

Example 6.2. Let $X_1, \ldots, X_n \sim \mathcal{N}\left(\mu, \sigma^2\right)$ be a random sample. We know that the statistic of the one-sided test of the hypotheses $H_0 : \mu = \mu_0$ vs. $H_1 : \mu < \mu_0$ is equal to:

$$T(X) = \frac{\overline{X} - \mu_0}{S/\sqrt{n}}.$$

```r
n = 20
mu = 0
sigma = 2
U = runif(n/2)
D = -2 * log(U)
V = runif(n/2)
Theta = 2 * pi * V
Z = sqrt(D) * c(cos(Theta), sin(Theta))
X = sigma * Z + mu
```

Algorithm 6.2 Normal Hypothesis Test by Use of Non-Parametric Bootstrap

Input: Random sample X, statistic $T(X)$ and bootstrap sample size B.

1: For $j = 1, 2, \ldots, B$, we iterate the following steps:

i: We generate $U_1, \ldots, U_n \sim \text{Unif}[0, 1]$ and let $I_i = \lfloor nU_i \rfloor + 1$ for $i = 1, 2, \ldots, n$.

ii: We let $X_*^{(j)} = (X_{I_1}, X_{I_2}, \ldots, X_{I_n})$.

iii: We calculate the sample average $\overline{X}_j^*$ and the square root S_j^* of the sample variance of $X_*^{(j)}$.

iv: We calculate the statistic:

$$T_j^* = \frac{\overline{X}_j^* - \overline{X}}{S_j^*/\sqrt{n}}.$$

2: We calculate:

$$N^* = \sum_{j=1}^{B} \mathbb{1}_{\{T_j^* \leqslant T\}}, \quad p^* = \frac{N^* + 1}{B + 1}.$$

Output: Estimated p-value p^* of the hypothesis test.

```r
mu0 = 1
alpha = 0.05
Xbar = mean(X)
S = sd(X)
t = (Xbar - mu0) * sqrt(n)/S
pval = pt(t, n - 1)
print(pval)
```

```
## [1] 0.01179182
```

```
B = 10000
Tstar = numeric(B)
for (j in 1:B) {
    U = runif(n)
    I = floor(n * U) + 1
    Xstar = X[I]
    Xbarstar = mean(Xstar)
    Sstar = sd(Xstar)
    Tstar[j] = (Xbarstar - Xbar) * sqrt(n)/Sstar
}
pval = (sum(Tstar <= t) + 1)/(B + 1)
print(pval)
```

```
## [1] 0.00909909
```

Algorithm 6.3 Normal Hypothesis Test by Use of Parametric Bootstrap

Input: Random sample X, statistic $T(X)$ and bootstrap sample size B.

1: We calculate the MLE of σ^2 under $H_0 : \mu = \mu_0$ as follows:

$$\widehat{\sigma}_0^2 = \frac{1}{n} \sum_{i=1}^{n} (X_i - \mu_0)^2 .$$

2: For $j = 1, 2, \ldots, B$, we iterate the following steps:

i: We generate a random sample $X_*^{(j)}$ of size n following the $N\left(\mu_0, \widehat{\sigma}_0^2\right)$ distribution.

ii: We calculate the sample average $\overline{X}_j^*$ and the square root S_j^* of the sample variance of $X_*^{(j)}$.

iii: We calculate the statistic:

$$T_j^* = \frac{\overline{X}_j^* - \mu_0}{S_j^*/\sqrt{n}}.$$

3: We calculate:

$$N^* = \sum_{j=1}^{B} \mathbb{1}_{\{T_j^* \leqslant T\}}, \quad p^* = \frac{N^* + 1}{B + 1}.$$

Output: Estimated p-value p^* of the hypothesis test.

```
sigma0 = sqrt(mean((X - mu0)^2))
for (j in 1:B) {
    U = runif(n/2)
    D = -2 * log(U)
    V = runif(n/2)
    Theta = 2 * pi * V
    Z = sqrt(D) * c(cos(Theta), sin(Theta))
```

```
    Xstar = sigma0 * Z + mu0
    Xbarstar = mean(Xstar)
    Sstar = sd(Xstar)
    Tstar[j] = (Xbarstar - mu0) * sqrt(n)/Sstar
}
pval = (sum(Tstar <= t) + 1)/(B + 1)
print(pval)
```

```
## [1] 0.01129887
```