# Como gerar variáveis aleatórias contínuas? (parte II)

O Método da Aceitação-Rejeição

ESTAT0090 – Estatística Computacional

Prof. Dr. Sadraque E. F. Lucena
sadraquelucena@academico.ufs.br

UNIVERSIDADE FEDERAL DE SERGIPE

# Cenário

Imagine que você está trabalhando com uma nova distribuição de probabilidade contínua. Como ela é muito recente, ainda não existem funções de simulação disponíveis em bibliotecas ou pacotes de programação. Para conseguir validar seus modelos, você precisa desenvolver do zero o código para gerar dados aleatórios e realizar suas simulações.

# Objetivos da aula

Nesta aula, aprenderemos a gerar ocorrências de variáveis aleatórias contínuas usando o **método da aceitação-rejeição**.

# Introdução

- Quando se pretende gerar números aleatórios a partir de distribuições de probabilidade, o método da transformação inversa pode não ser aplicável em muitos casos. Alguns são:
  - Quando não é possível determinar a função de distribuição acumulada $F_X(x)$;
  - Quando, mesmo conhecendo $F_X(x)$, não é possível obter a sua inversa $F_X^{-1}(x)$.
- Nesses casos, uma opção é usar o método da aceitação-rejeição.

# Método da Aceitação-Rejeição

O método da aceitação-rejeição consiste em gerar ocorrências de uma distribuição $f$ usando a distribuição auxiliar $g$.

### Algoritmo

**Passo 1:** Gere um valor $x$ a partir da distribuição $g(x)$;

**Passo 2:** Gere $u \sim$ Uniforme$(0, 1)$;

**Passo 3:** O valor $x$ gerado é aceito se o número uniforme $u$ satisfizer:

$$u \leq \frac{f(x)}{Mg(x)}.$$

Caso contrário, o valor $x$ é rejeitado e o processo é repetido a partir do Passo 1.

**Passo 4:** Se o valor $x$ for aceito, ele é considerado um número aleatório da distribuição $f(x)$. Se for rejeitado, um novo valor é gerado até que seja aceito.

# Método da Aceitação-Rejeição

- $M$ é determinado de forma que

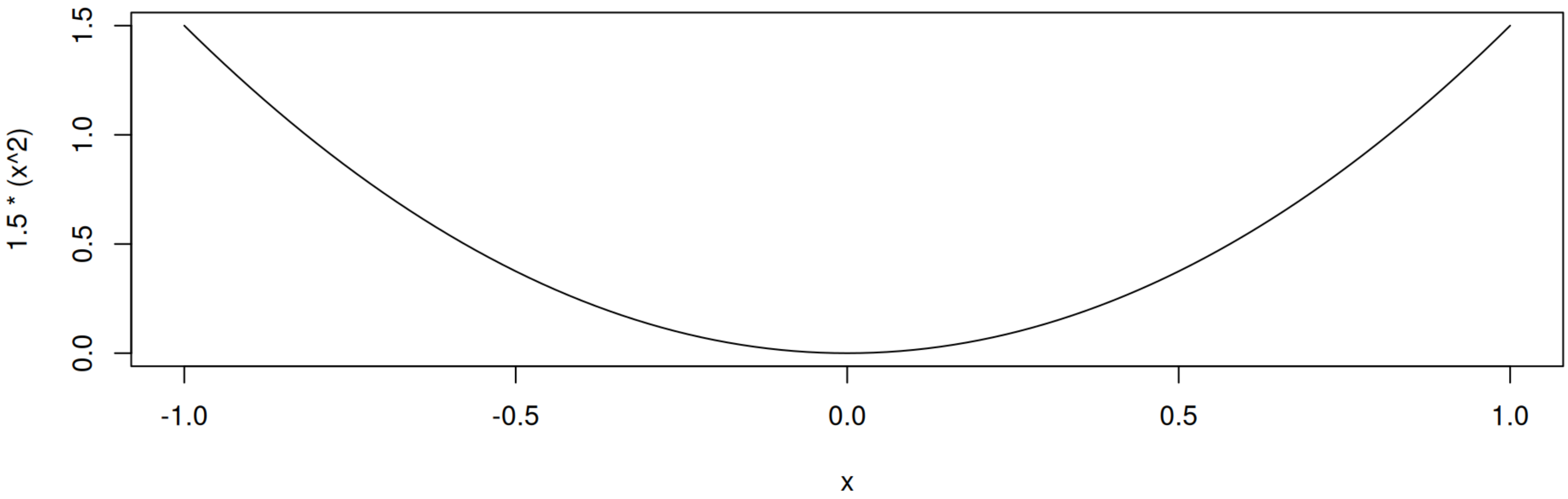
$$M \geq \max_x \frac{f(x)}{g(x)}.$$

- A probabilidade de aceitação é $1/M$ . Portanto, $M$ deve ser o menor possível para maior eficiência computacional.

# Exemplo 12.1

Gere valores de uma variável aleatória $X$ com $f(x) = 1.5x^2, -1 < x < 1$.

```r
## Gráfico da fdp da v.a. X, f(x).
curve(1.5 * (x ^ 2), -1, 1)
```




```r
## Tem integral 1?
integrate(function(x) 1.5 * (x^2), lower = -1, upper = 1)
```

```
1 with absolute error < 1.1e-14
```

# Exemplo 12.1

Vamos considerar como $g$ (distribuição auxiliar) a densidade da $U[-1, 1]$. Ou seja,

$$g(x) = \frac{1}{1-(-1)} = \frac{1}{2} = 0.5, \qquad -1 < x < 1.$$

Agora vamos definir $M$:

$$M \geq \max_x \frac{f(x)}{g(x)} = \frac{1.5}{0.5} = 3.$$

Portanto, vamos usar $M = 3$.

# Exemplo 12.1

Então o algoritmo para gerar ocorrências da v.a. com densidade $f$ a partir de $g$ é

1. Gere $x$ a partir da distribuição $g(x)$, ou seja, $x \sim U[-1, 1]$;
2. Gere $u \sim U[0, 1]$;
3. O valor $x$ é aceito se

$$u \leq \frac{f(x)}{Mg(x)} = \frac{f(x)}{3g(x)}.$$

   Caso contrário, o valor $x$ é rejeitadoe todo o processo é repetido.

# Exemplo 12.1

Código R:

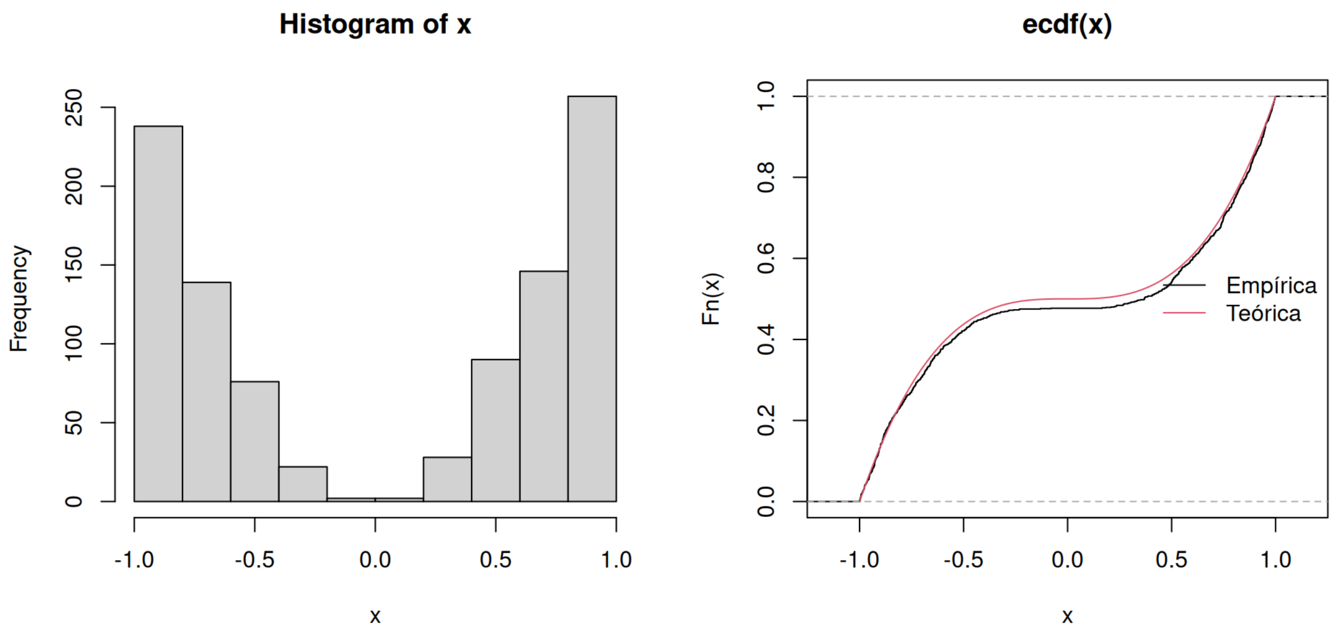
```
geraf <- function(n=1) {
  f <- function(x) 1.5 * x^2
  g <- function(x) 0.5 + 0 * x
  M <- 3
  x <- numeric(0)
  while (length(x) < n) {
    x_candidato <- runif(n = 1, min = -1, max = 1)
    u <- runif(1)
    r <- f(x_candidato)/(M * g(x_candidato))
    if(u < r) {
      x <- c(x, x_candidato)
    }
  }
  return(x)
}

geraf(1)
```

```
[1] 0.735495
```

# Exemplo 12.1

Gerando 1.000 ocorrências:

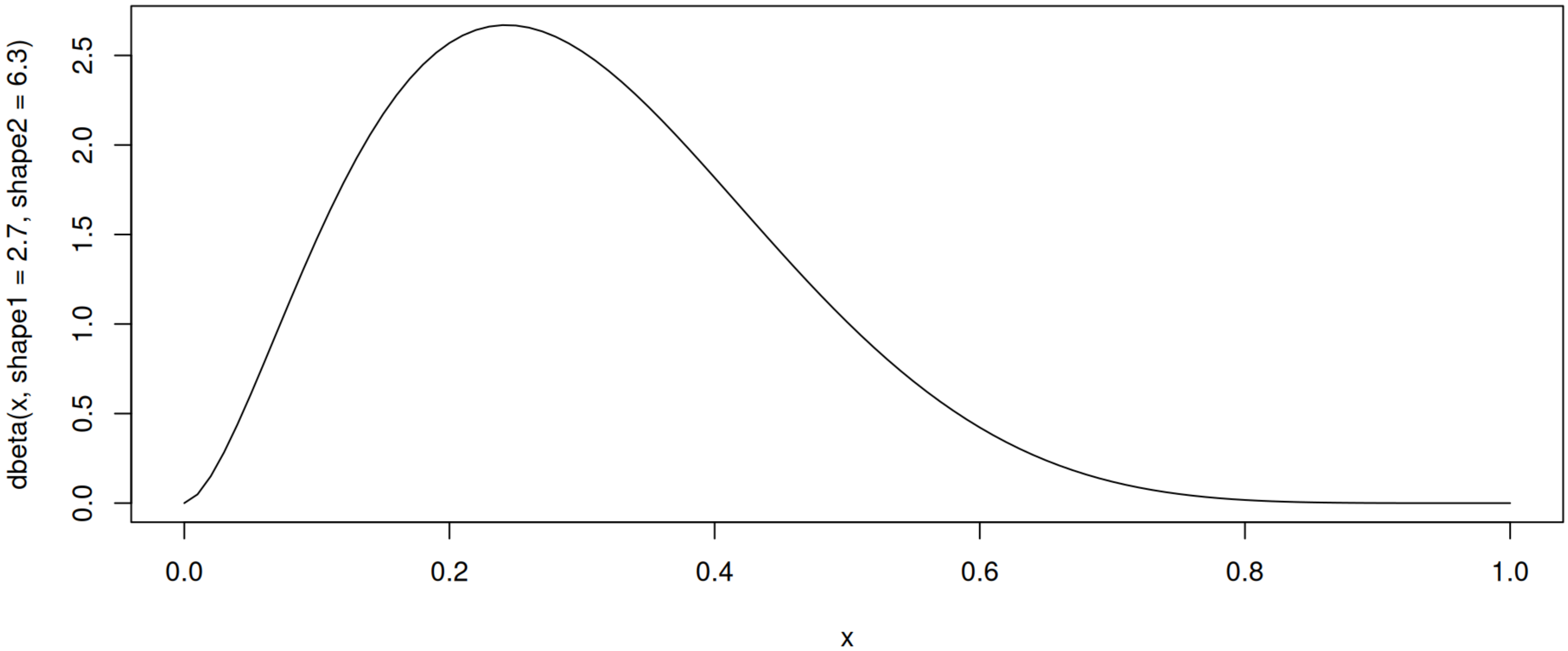
```r
x <- geraf(1000)
par(mfrow = c(1, 2))
hist(x)
Fx <- function(x) 0.5 * (x^3 + 1)
plot(ecdf(x))
curve(Fx, add = TRUE, from = -1, to = 1, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
lty = 1, col = 1:2, bty = "n")
```



# Exemplo 12.2

Gere valores de uma distribuição Beta($\alpha = 2.7, \beta = 6.3$) usando o método da aceitação-rejeição.

```r
curve(dbeta(x, shape1 = 2.7, shape2 = 6.3), from = 0, to = 1)
```



# Exemplo 12.2

- Como os valores da distribuição beta estão definidos no intervalo $(0, 1)$, vamos usar a distribuiçao $U(0, 1)$ como distribuição auxiliar. Logo, $g = 1, 0 < x < 1$.
- Para determinar $M$ usamos $M \geq \max_x \frac{f(x)}{g(x)}$. Como $g(x) = 1$, buscamos por $M \geq \max_x f(x)$.
  - Para determinar o máximo da Beta$(2.7, 6.3)$ temos duas opções:
    1. Usar a expressão da moda da distribuição (se existir)
    2. Achar esse valor máximo por otimização numérica.
  - No caso da beta existe uma expressão em forma fechada para a moda: $\frac{\alpha-1}{\alpha+\beta-2}$.

```r
## Define parâmetros
alfa <- 2.7; beta <- 6.3
## A moda é
(moda <- (alfa - 1)/(alfa + beta - 2))
```

```
[1] 0.2428571
```

```r
## A densidade nesse ponto (ou seja, M) é então
dbeta(moda, alfa, beta)
```

```
[1] 2.669744
```

# Exemplo 12.2

Se preferirmos obter $M$ por otimização numérica, usamos a função `optimize()`. No caso da beta, usaremos a função `dbeta()`, que já implementa a função da densidade beta, mas poderíamos escrevê-la manualmente.

```
(max.beta <- optimize(f = function(x) {dbeta(x, alfa, beta)},
                      interval = c(0, 1), maximum = TRUE))
```

```
$maximum
[1] 0.2428608

$objective
[1] 2.669744
```

```
(M <- max.beta$objective/1)
```

```
[1] 2.669744
```

# Exemplo 12.2

Assim, o algoritmo para gerar ocorrências da beta$(2.7, 6.3)$ a partir da $U(0, 1)$ é:

1. Gere $x$ a partir da distribuição $g(x)$, ou seja, $x \sim U(0, 1)$;
2. Gere $u \sim U(0, 1)$;
3. O valor $x$ é aceito se

$$u \leq \frac{f(x)}{Mg(x)} = \frac{f(x)}{2.669744g(x)}.$$

Caso contrário, o valor de $x$ é redeitado e todo o processo é repetido.

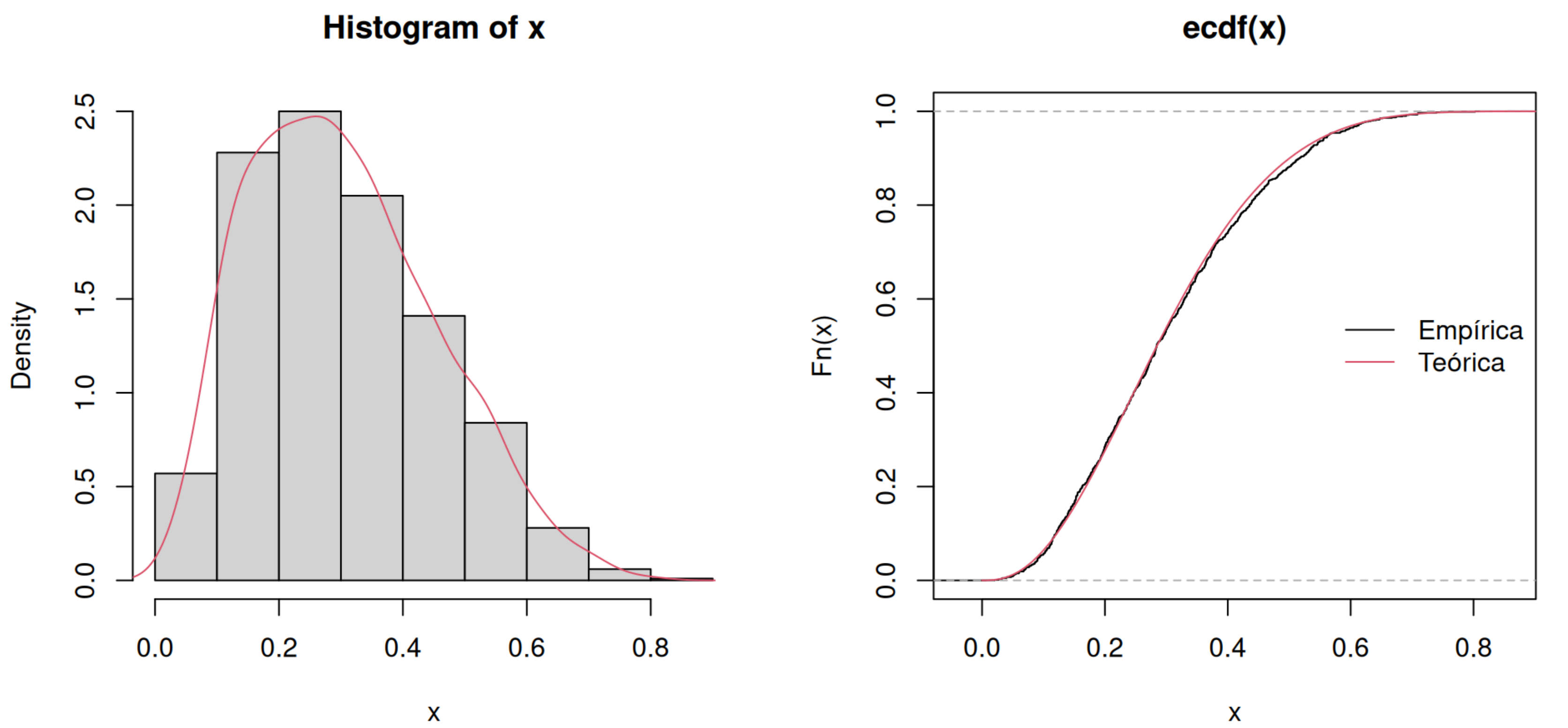
# Exemplo 12.2

Código R:

```
beta2.7_6.3 <- function(n = 1) {
  ## Define funções
  f <- function(x) dbeta(x, alfa, beta)
  g <- function(x) 1 + 0 * x
  M <- 2.669744
  x <- numeric(0)
  while (length(x) < n) {
    x_candidato <- runif(n = 1, min = 0, max = 1)
    u <- runif(1)
    r <- f(x_candidato)/(M * g(x_candidato))
    if(u < r) {
      x <- c(x, x_candidato)
    }
  }
  return(x)
}
```

# Exemplo 12.2

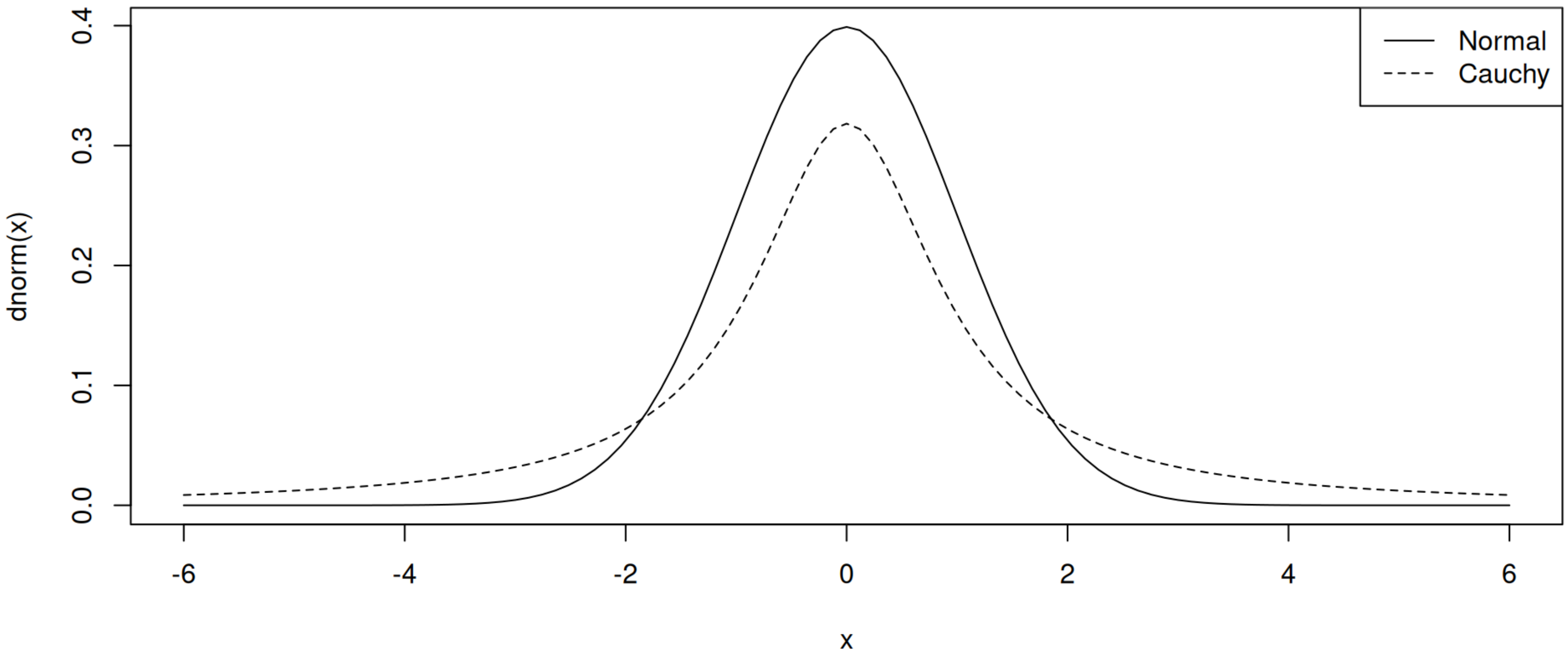
```
x <- beta2.7_6.3(1000)
par(mfrow = c(1, 2))
hist(x, freq = FALSE); lines(density(x), col = 2)
plot(ecdf(x))
curve(pbeta(x, 2.7, 6.3), add = TRUE, from = 0, to = 1, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
lty = 1, col = 1:2, bty = "n")
```



# Exemplo 12.3

Gere ocorrências da $N(0, 1)$ a partir da Cauchy$(0, 1)$.

```
## Verifica as distribuições
curve(dnorm(x), -6, 6)
curve(dcauchy(x), add = TRUE, lty = 2)
legend("topright", legend = c("Normal", "Cauchy"),
       lty = c(1, 2))
```



# Exemplo 12.3

Vamos obter $M$ por otimização.

```
## Obtém valor de M por otimização
(M <- optimize(f = function(x) {dnorm(x)/dcauchy(x)},
               interval = c(-6, 6), maximum = TRUE)$objective)
```

```
[1] 1.520347
```

Algoritmo:

1. Gere $x \sim$ Cauchy$(0, 1)$;
2. Gere $u \sim U(0, 1)$;
3. O valor $x$ é aceito se

$$u \leq \frac{f(x)}{Mg(x)} = \frac{f(x)}{1.520347g(x)}.$$

Caso contrário, o valor $x$ é rejeitado e todo o processo é repetido.

# Exemplo 12.3

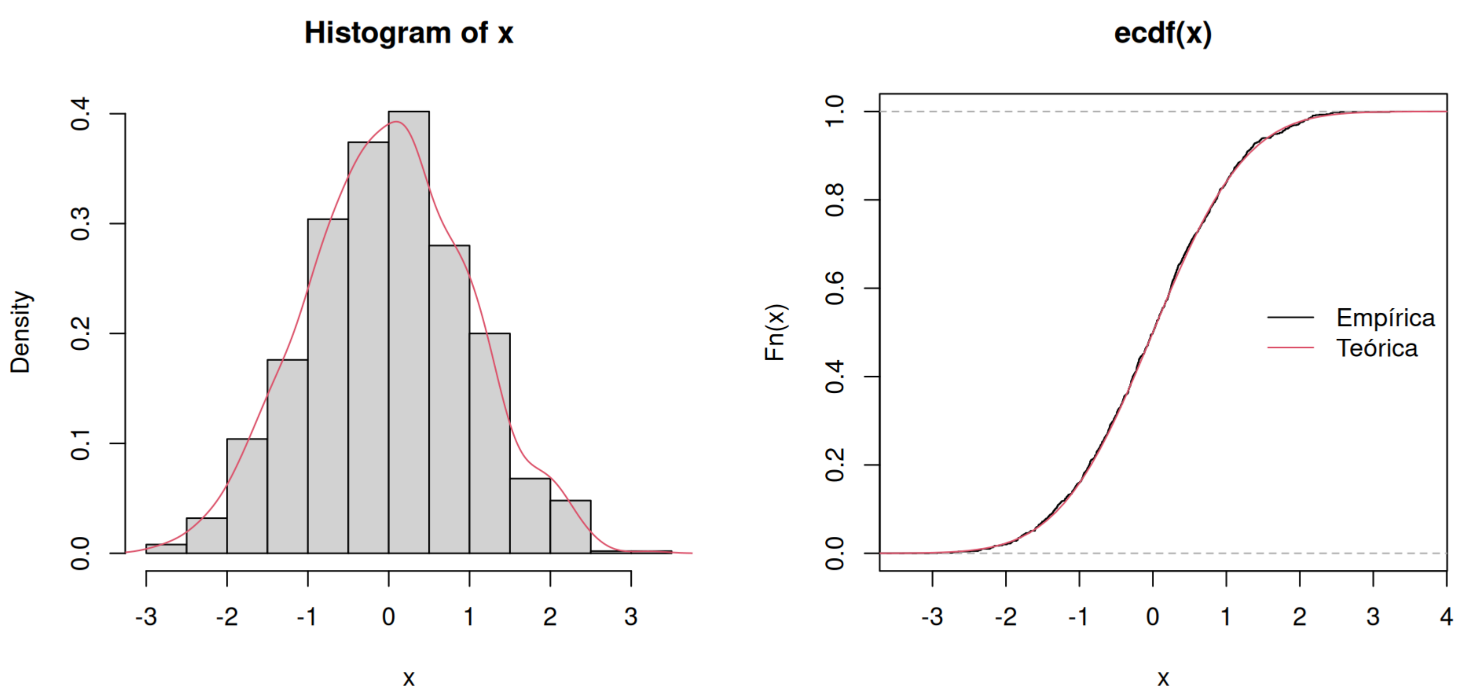
Código R:

```
norm01 <- function(n = 1) {
  ## Define funções
  f <- function(x) dnorm(x, 0, 1)
  g <- function(x) dcauchy(x, 0, 1)
  M <- optimize(f = function(x) {dnorm(x)/dcauchy(x)},
                interval = c(-6, 6), maximum = TRUE)$objective
  x <- numeric(0)
  while (length(x) < n) {
    x_candidato <- rcauchy(n = 1, location = 0, scale = 1)
    u <- runif(1)
    r <- f(x_candidato)/(M * g(x_candidato))
    if(u < r) {
      x <- c(x, x_candidato)
    }
  }
  return(x)
}
```

# Exemplo 12.3

Código R:

```
x <- norm01(1000)
par(mfrow = c(1, 2))
hist(x, freq = FALSE); lines(density(x), col = 2)
plot(ecdf(x))
curve(pnorm(x), add = TRUE, from = -6, to = 6, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
       lty = 1, col = 1:2, bty = "n")
```



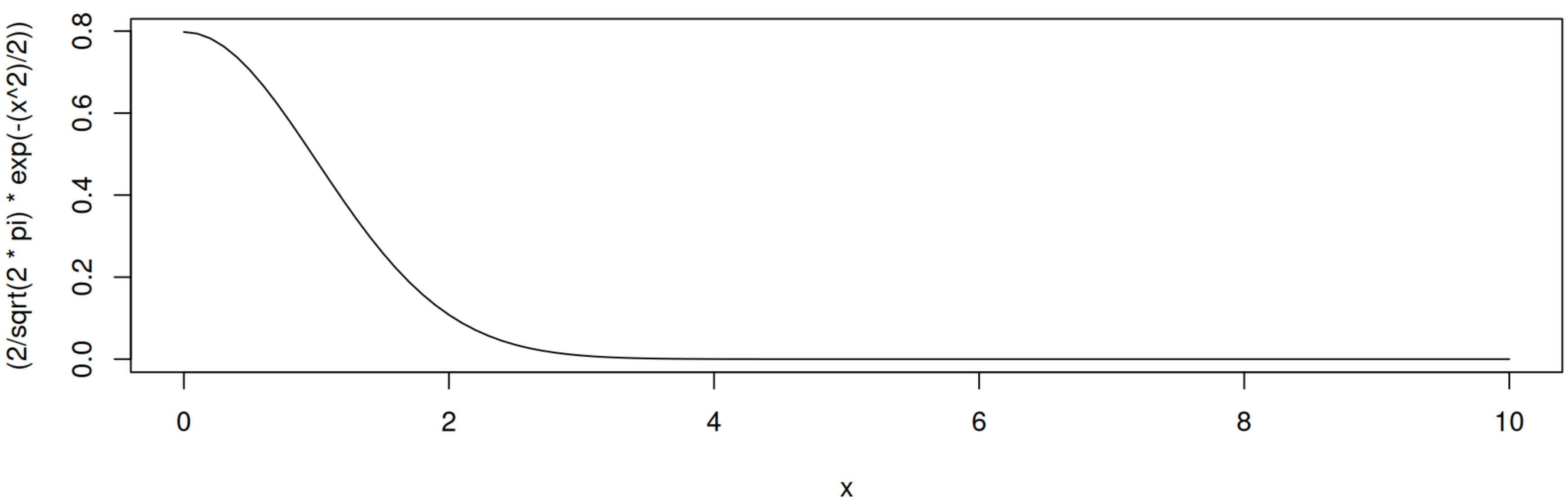
# Exemplo 12.4

Gere ocorrências da densidade

$$f(x) = \frac{2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, x \geq 0,$$

a partir da Exponencial(1).

```
## Gráfico da fdp da v.a. X, f(x).
curve((2/sqrt(2*pi) * exp(-(x^2)/2)), 0, 10)
```



# Exemplo 12.4

Vamos obter $M$ por otimização.

```r
# Obtém valor de M por otimização
(M <- optimize(function(x) {((2/sqrt(2*pi)*exp(-(x^2)/2)))/dexp(x)},
               interval = c(0, 20), maximum = TRUE)$objective)
```

```
[1] 1.315489
```

Algoritmo: 1. Gere $x \sim$ Exponencial(1); 2. Gere $u \sim U(0, 1)$; 3. O valor $x$ é aceito se

$$u \leq \frac{f(x)}{Mg(x)} = \frac{f(x)}{1.315489g(x)}.$$

Caso contrário, o valor $x$ é rejeitado e todo o processo é repetido.

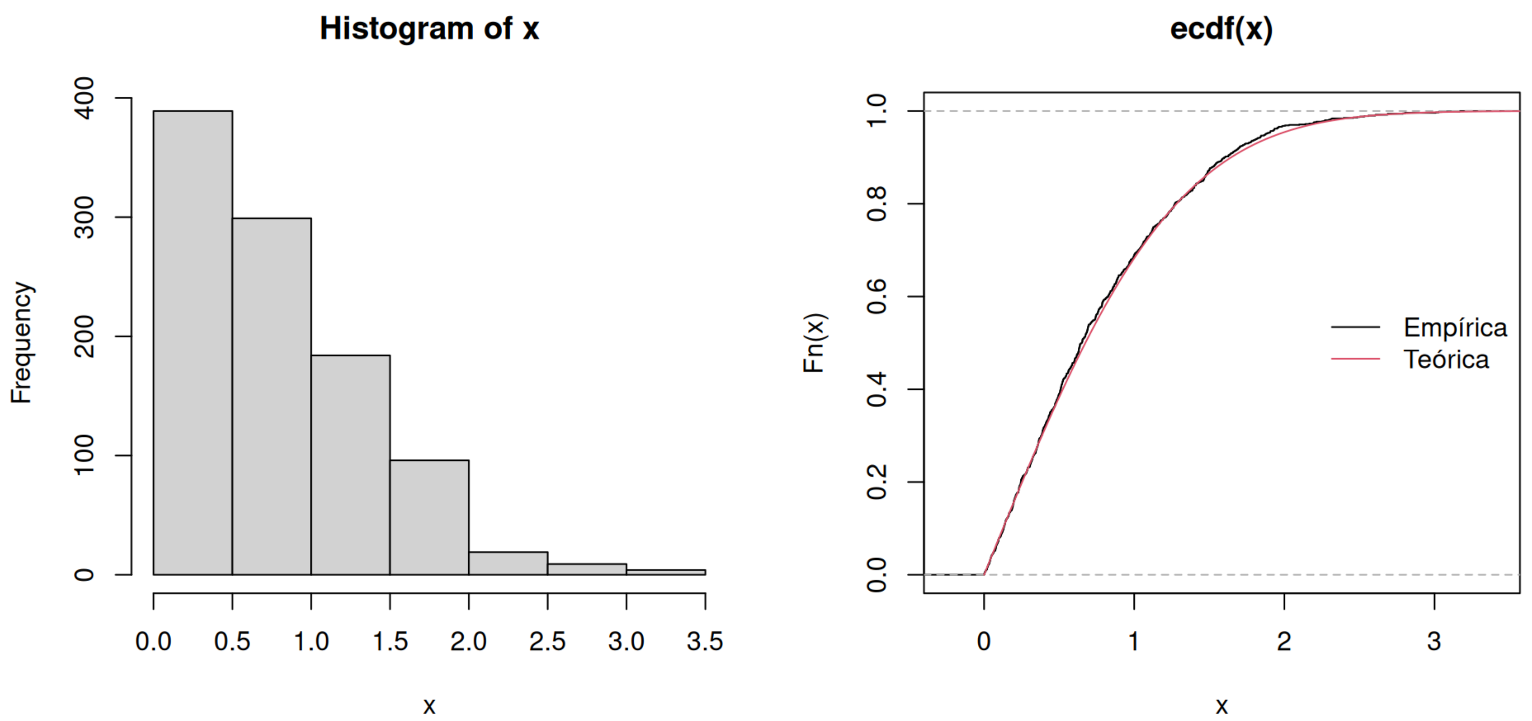
# Exemplo 12.4

Código R:

```
geraEx4 <- function(n = 1) {
  ## Define funções
  f <- function(x) (2/sqrt(2*pi) * exp(-(x^2)/2))
  g <- function(x) dexp(x, 1)
  M <- optimize(f = function(x) {f(x)/g(x)},
                interval = c(0, 10), maximum = TRUE)$objective
  x <- numeric(0)
  while (length(x) < n) {
    x_candidato <- rexp(n = 1, rate = 1)
    u <- runif(1)
    r <- f(x_candidato)/(M * g(x_candidato))
    if(u < r) {
      x <- c(x, x_candidato)
    }
  }
  return(x)
}
```

# Exemplo 12.4

```r
x <- geraEx4(1000)
par(mfrow = c(1, 2))
hist(x)
Fx <- function(x) pracma::erf(x/sqrt(2))
plot(ecdf(x))
curve(Fx, add = TRUE, from = 0, to = 4, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
       lty = 1, col = 1:2, bty = "n")
```



# Atividade

Gere ocorrências da $N(0, 1)$ a partir da $U(-10, 10)$ pelo método da aceitação-rejeição.

# Ganho da aula

- Compreensão da geração de números aleatórios contínuos usando o método da aceitação-rejeição.

# Fim

Aula baseada no material de Walmes M. Zeviani e Fernando P. Mayer.