

1 Aproximação de integrais

1.1 Integração de Monte Carlo

1.1.1 Exemplo (exponencial)

1.1.2 Exemplo (beta)

2 Distribuições aproximadas

2.1 Amostragem por rejeição

2.1.1 Exemplo (exponencial)

2.1.2 Exemplo (beta)

2.2 Re-amostragem por importância

2.2.1 Exemplo (exponencial)

2.2.2 Exemplo (beta)

Métodos de Monte Carlo

Distribuições aproximadas baseadas em simulação de Monte Carlo

Fernando P. Mayer

1 Aproximação de integrais

Já vimos que a integração de Monte Carlo serve para aproximar integrais de alta dimensão ou que não possuem solução fechada (analítica).

Agora veremos que a integração de Monte Carlo, naturalmente serve também para o cálculo de probabilidades entre algum intervalo, e.g., $P[a < X < b]$ ou $P[X > a]$, que nada mais são do que cálculos de áreas no intervalo especificado.

A integração de Monte Carlo nos fornece uma estimativa dessa integral, e, como vimos, o erro padrão dessa estimativa pode ser calculado, e intervalos de confiança podem ser obtidos devido à suposição de que assintoticamente a estimativa possui distribuição normal.

Agora, veremos que podemos obter essa mesma estimativa, mas de uma forma um pouco diferente: simulando a própria distribuição de interesse, e obtendo valores de áreas (integrais) usando a própria amostra. A vantagem é que, dessa forma, obtemos uma descrição mais completa do problema em questão, e, com isso, podemos obter outras medidas que forem necessárias, de uma única vez. Além disso, medidas de incerteza como erros-padrões e intervalos de confiança podem ser derivados a partir da própria distribuição amostral.

A obtenção de uma amostra da distribuição de interesse será fundamental para os métodos de inferência que veremos mais adiante. Além disso, estes métodos são utilizados como o “padrão” para a resolução de problemas em inferência bayesiana.

1.1 Integração de Monte Carlo

Como vimos, a integração **simples** de Monte Carlo é um algoritmo **não sequencial** (i.e., os valores amostrados são independentes) para aproximação de integrais do tipo

$$\theta = \int_a^b h(x)dx$$

através da relação

$$\begin{aligned}\theta &= \int_a^b g(x)f(x)dx \\ &= (b-a) \int_a^b g(x) \frac{1}{b-a} dx \\ &= (b-a)E[g(X)]\end{aligned}$$

onde $X \sim U(a, b)$.

De maneira geral, para calcular $\theta = \int_a^b g(x)dx$:

1. Gere X_1, \dots, X_m de $U(a, b)$
2. Calcule $\overline{g(x)} = \frac{1}{m} \sum_{i=1}^m g(x_i)$
3. $\hat{\theta} = (b-a)\overline{g(x)}$

1.1.1 Exemplo (exponencial)

Para obter a estimativa de

$$\theta = \int_0^1 e^{-x} dx$$

fazemos:

```
## Obtem m valores da U(0,1)
m <- 1e4
x <- runif(m)
## Calcula g(x)
theta.hat <- exp(-x)
## Calcula a média
(m.theta.hat <- sum(theta.hat)/m)

# [1] 0.6307915
```

```
## Solução analítica
(theta <- 1 - exp(-1))

# [1] 0.6321206
```

Para obter a estimativa de

$$\theta = \int_2^4 e^{-x} dx$$

fazemos:

```
## Obtem m valores da U(2,4)
m <- 1e4
a <- 2; b <- 4
x <- runif(m, min = a, max = b)
## Calcula g(x)
theta.hat <- exp(-x)
## Calcula a média * (b - a)
(m.theta.hat <- (sum(theta.hat)/m) * (b - a))

# [1] 0.1164489

## Solução analítica
(theta <- exp(-2) - exp(-4))

# [1] 0.1170196
```

Vemos que, nos dois casos acima, estamos na verdade obtendo uma estimativa das probabilidades $P[0 < X < 1]$ e $P[2 < X < 4]$, respectivamente.

No entanto, se quisermos calcular a seguinte probabilidade

$$P[X > 2] = \int_2^{\infty} e^{-x} dx$$

Nesse caso, a integração simples de Monte Carlo não é útil, pois não é possível que o limite de integração seja indefinido.

1.1.2 Exemplo (beta)

Considere que uma VA X tem distribuição Beta($a = 9.39$, $b = 33.67$), e queremos obter $P[0.3 < X < 0.5]$, ou seja,

$$P[0.3 < X < 0.5] = \int_{0.3}^{0.5} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^a (1-x)^b dx$$

Pela integração simples de Monte Carlo, podemos facilmente obter esta estimativa:

```

a <- 9.39; b <- 33.67
m <- 1e4
x <- runif(m, 0.3, 0.5)
## Calcula g(x)
theta.hat <- dbeta(x, a, b)
## Calcula a média
(m.theta.hat <- (sum(theta.hat)/m) * (0.5 - 0.3))

# [1] 0.09925558

## Solução analítica
pbeta(0.5, a, b) - pbeta(0.3, a, b)

# [1] 0.1018369

```

No caso da Beta, calcular probabilidades do tipo $P[X > 0.2]$, por exemplo, é mais fácil, pois sabemos que o domínio da Beta está no intervalo $(0, 1)$. Portanto, os limites de integração são definidos

$$P[X > 0.2] = \int_{0.2}^1 \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^a (1-x)^b dx$$

Assim, podemos fazer

```

m <- 1e4
x <- runif(m, 0.2, 1)
## Calcula g(x)
theta.hat <- dbeta(x, a, b)
## Calcula a média
(m.theta.hat <- (sum(theta.hat)/m) * (1 - 0.2))

# [1] 0.5967183

## Solução analítica
pbeta(0.2, a, b, lower.tail = FALSE)

# [1] 0.5876479

```

2 Distribuições aproximadas

Note que nos dois exemplos acima, para cada problema de cálculo de probabilidades (integrais), precisamos aplicar o método de Monte Carlo.

No entanto, suponha que precisamos fazer diversos cálculos sob uma determinada VA. Ao invés de aplicar esse método para cada cálculo, podemos pensar que seria mais fácil se pudessemos obter uma **amostra** da distribuição de interesse. A partir desta **distribuição**,

qualquer cálculo necessário pode ser derivado facilmente.

Essa é a ideia dos métodos a seguir. O primeiro deles, a amostragem por **aceitação-rejeição** já foi visto no contexto de geração de números aleatórios. A mesma ideia será usada aqui, mas agora **usaremos a amostra gerada da distribuição de interesse para resolver problemas de integração**.

O segundo método, o de **re-amostragem por importância**, pode ser visto como uma extensão do método de amostragem por importância, que vimos no contexto de generalização do método simples de integração de Monte Carlo. Nesse caso, usaremos as mesmas ideias para **gerar uma amostra da distribuição de interesse**, e não apenas resolver uma integral.

2.1 Amostragem por rejeição

A ideia inicial do método é exatamente **igual ao que já foi visto anteriormente** no contexto de geração de números aleatórios de uma distribuição alvo f .

O seguinte algoritmo permite gerar números aleatórios de uma distribuição de probabilidade caracterizada pela função densidade f :

1. Gerar y como sendo uma ocorrência da variável aleatória proveniente de uma distribuição com densidade g , que “encapsula” a distribuição de f .
2. Gerar u como sendo uma ocorrência de uma uniforme padrão.
3. Se

$$u \leq \frac{f(y)}{Mg(y)}$$

considerar que $x = y$ é um valor da distribuição de probabilidade alvo cuja densidade é f , caso contrário, descartar y .

4. Repetir até atingir o número de valores desejado n .

Para determinar o valor de M , basta seguir a seguinte relação

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

2.1.1 Exemplo (exponencial)

Suponha novamente que queremos obter a integral

$$\theta = \int_2^4 e^{-x} dx$$

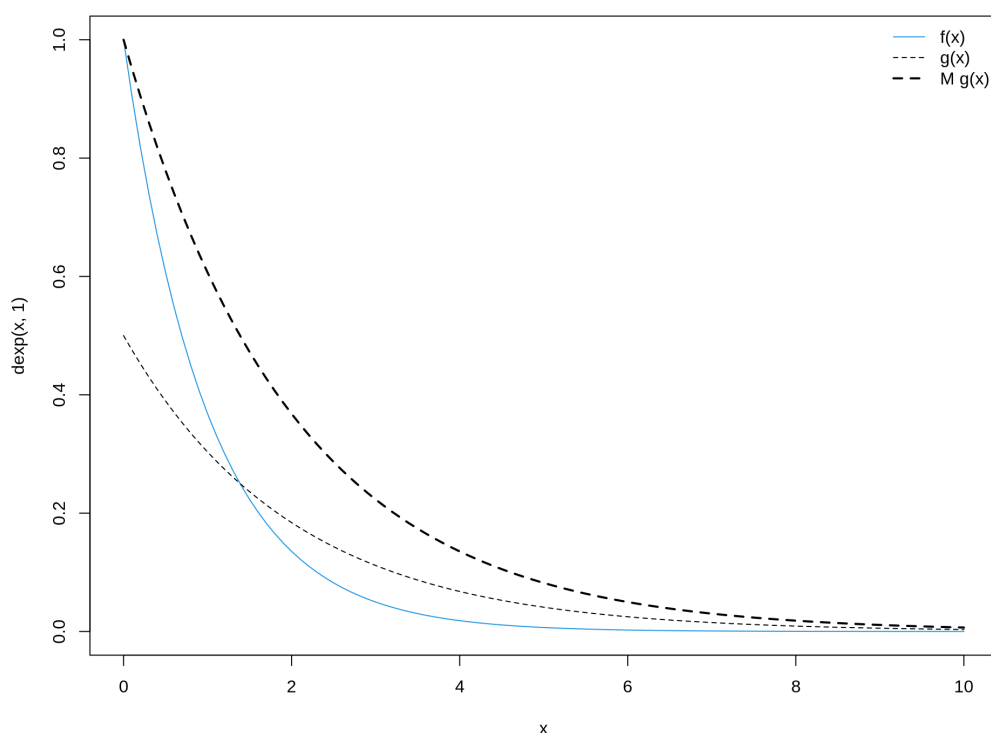
Sabemos que a função acima é uma $\text{Exp}(1)$. Ao invés de usar o método de integração de Monte Carlo, vamos **obter uma amostra desta distribuição**, e, com ela, calcularemos a integral desejada.

Por simplificação, usaremos como distribuição proposta uma $\text{Exp}(0.5)$. Nesse caso, obtemos o valor de M como

```
(M <- optimize(f = function(x) {dexp(x, 1)/dexp(x, .5)},
               interval = c(0, 50), maximum = TRUE)$objective)

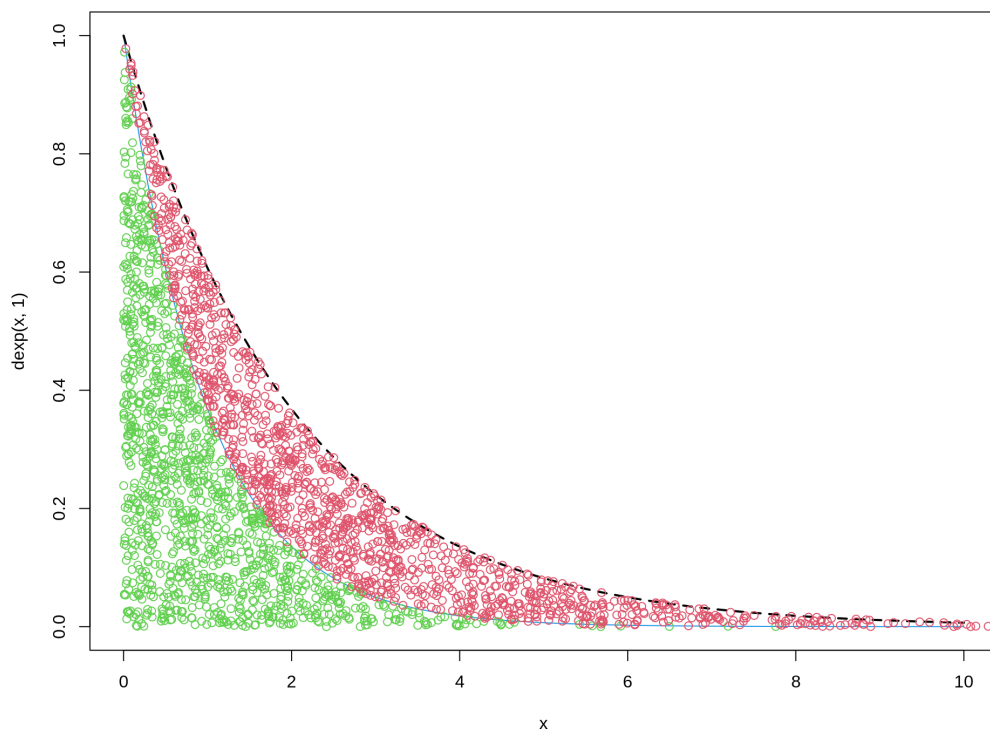
# [1] 1.99993

curve(dexp(x, 1), from = 0, to = 10, col = 4, ylim = c(0, 1))
curve(dexp(x, 0.5), from = 0, to = 10, add = TRUE, lty = 2)
curve(M * dexp(x, .5), add = TRUE, lty = 2, lwd = 2)
legend("topright", legend = c("f(x)", "g(x)", "M g(x)"),
      lty = c(1, 2, 2), col = c(4, 1, 1), lwd = c(1, 1, 2), bty =
      "n")
```



Usando um valor fixo apenas para ver o gráfico

```
## Criando os elementos necessários
f <- function(x) dexp(x, 1)
g <- function(x) dexp(x, 0.5)
## Simula
Nsim <- 2500
## Amostra da proposta
y <- rexp(Nsim, 0.5)
## Amostra da U(0,1)
u <- runif(Nsim)
r <- f(y)/(M * g(y))
x <- y[u < r]
ua <- u[u < r]
ur <- u[u >= r]
## Gráfico de aceitação
curve(dexp(x, 1), from = 0, to = 10, col = 4)
curve(M * dexp(x, 0.5), add = TRUE, lty = 2, lwd = 2)
points(x, ua * M * g(x), col = 3)
points(y[u >= r], ur * M * g(y[u >= r])), col = 2)
```

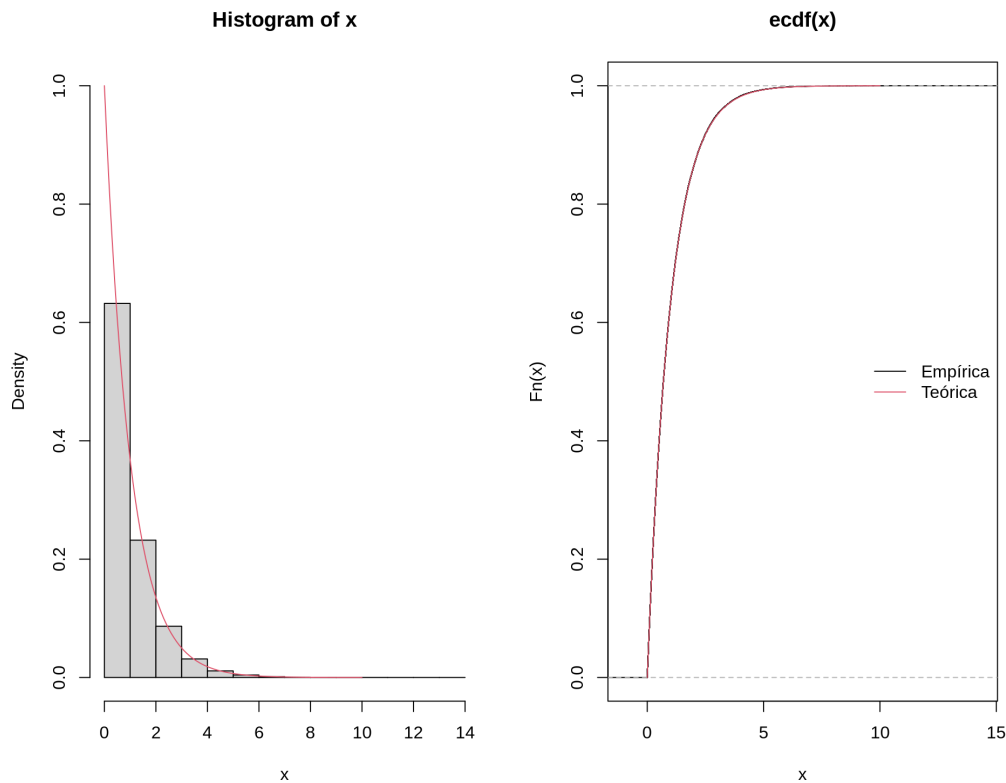


Agora podemos prosseguir com o algoritmo mais geral da seguinte forma:

```
## Criando os elementos necessários
f <- function(x) dexp(x, 1)
g <- function(x) dexp(x, 0.5)
## Simula valores de f
N <- 1e5
x <- numeric(N)
i <- 1
while(i <= N) {
  y <- rexp(1, 0.5)
  u <- runif(1)
  r <- f(y)/(M * g(y))
  if(u < r) {
    x[i] <- y
    i <- i + 1
  }
}
## Tamanho da amostra
length(x)

# [1] 100000

## Visualização da amostra
par(mfrow = c(1, 2))
hist(x, freq = FALSE, ylim = c(0, 1))
curve(dexp(x, 1), add = TRUE, from = 0, to = 10, col = 2)
plot(ecdf(x))
curve(pexp(x, 1), add = TRUE, from = 0, to = 10, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
      lty = 1, col = 1:2, bty = "n")
par(mfrow = c(1, 1))
```

Como agora temos a distribuição completa da $\text{Exp}(1)$, podemos obter qualquer probabilidade (ou integral) aproximada, simplesmente através das contagens de valores no intervalo especificado. Note que a integral da distribuição, ou seja, $\int_0^\infty f(x)dx$ deve fechar em **aproximadamente 1**:

```
sum(x >= 0)/N
```

```
# [1] 1
```

```
mean(x >= 0)
```

```
# [1] 1
```

Agora, por exemplo, a $P[2 < X < 4]$ pode ser calculada como

```
## Probabilidade empírica
```

```
sum(x > 2 & x < 4)/N
```

```
# [1] 0.11805
```

```
## Teórica
```

```
pexp(4, 1) - pexp(2, 1)
```

```
# [1] 0.1170196
```

Agora se quisermos calcular $P[X > 2]$, temos condições de fazer através de

```
## Probabilidade empírica
sum(x >= 2)/N

# [1] 0.13568

## Teórica
pexp(2, 1, lower.tail = FALSE)

# [1] 0.1353353
```

De maneira geral, dessa forma, qualquer probabilidade pode ser calculada como mostrado acima. Matematicamente, podemos representar esse processo por

$$P[X \leq x] = \frac{1}{m} \sum_{i=1}^m I(X \leq x)$$

onde

$$I(X \leq x) = \begin{cases} 1, & \text{se } X \leq x \\ 0, & \text{caso contrário} \end{cases}$$

é uma função indicadora. Note que isso se estende para qualquer outro tipo de probabilidade, e.g. $P[X > x]$, etc.

2.1.2 Exemplo (beta)

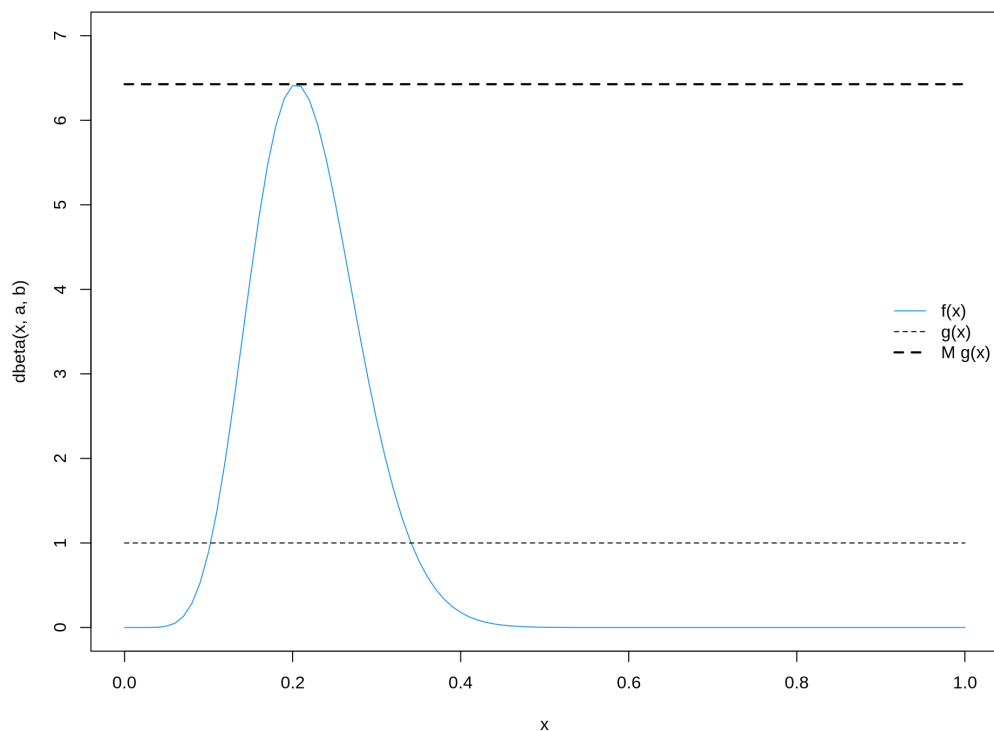
Voltando ao exemplo de $X \sim \text{Beta}(a = 9.39, b = 33.67)$, queremos calcular $P[0.3 < X < 0.5]$, mas agora através da obtenção de uma **amostra completa desta distribuição**.

Usando o método de aceitação-rejeição, vamos usar como distribuição proposta uma $U(0, 1)$. Portanto:

```
a <- 9.39; b <- 33.67
(M <- optimize(f = function(x) {dbeta(x, a, b)/dunif(x)},
               interval = c(0, 1), maximum = TRUE)$objective)

# [1] 6.426808

curve(dbeta(x, a, b), from = 0, to = 1, col = 4, ylim = c(0, 7))
curve(dunif(x), from = 0, to = 1, add = TRUE, lty = 2)
curve(M * dunif(x), add = TRUE, lty = 2, lwd = 2)
legend("right", legend = c("f(x)", "g(x)", "M g(x)"),
      lty = c(1, 2, 2), col = c(4, 1, 1), lwd = c(1, 1, 2), bty =
"n")
```



Seguindo o mesmo algoritmo anterior, temos então:

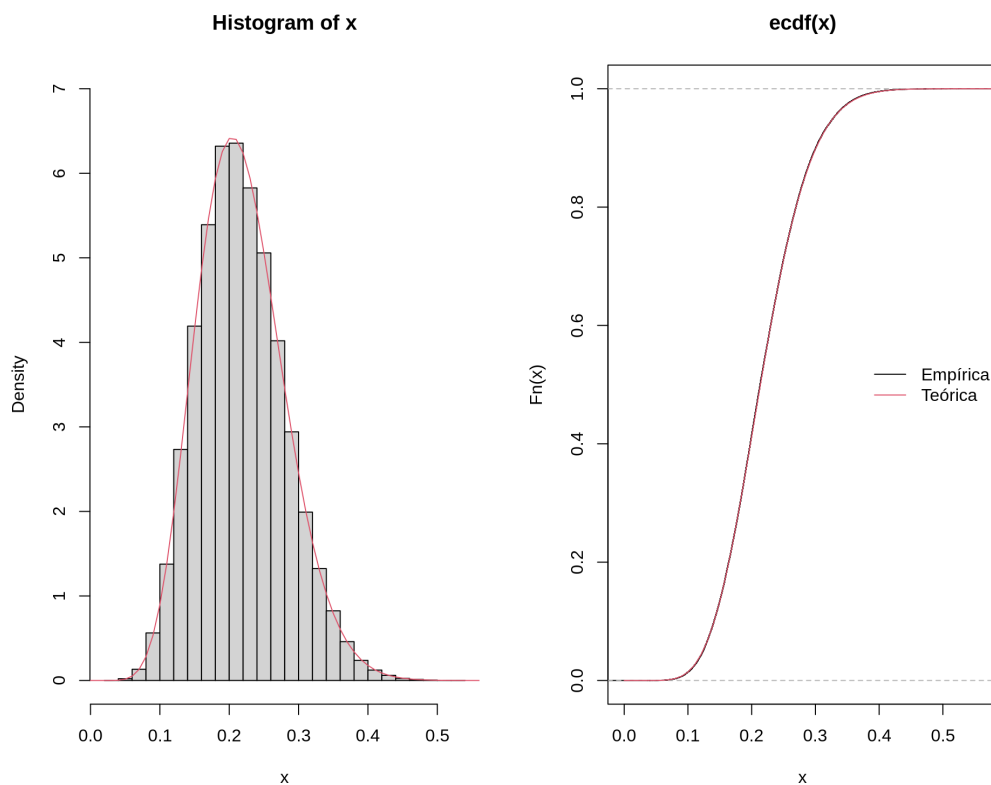
```
## Criando os elementos necessários
f <- function(x, a, b) dbeta(x, a, b)
g <- function(x) dunif(x)
## Simula valores de f
N <- 1e5
x <- numeric(N)
i <- 1
while(i <= N) {
  y <- runif(1)
  u <- runif(1)
  r <- f(y, a, b)/(M * g(y))
  if(u < r) {
    x[i] <- y
    i <- i + 1
  }
}
## Tamanho da amostra
length(x)

# [1] 100000

## Integral
sum(x >= 0 & x <= 1)/N

# [1] 1
```

```
## Visualização da amostra
par(mfrow = c(1, 2))
hist(x, freq = FALSE, ylim = c(0, 7))
curve(dbeta(x, a, b), add = TRUE, from = 0, to = 1, col = 2)
plot(ecdf(x))
curve(pbeta(x, a, b), add = TRUE, from = 0, to = 1, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
      lty = 1, col = 1:2, bty = "n")
par(mfrow = c(1, 1))
```



Agora, podemos calcular qualquer probabilidade. Por exemplo:

```
## P[0.3 < X < 0.5]
sum(x > 0.3 & x < 0.5)/N

# [1] 0.1013

pbeta(0.5, a, b) - pbeta(0.3, a, b)

# [1] 0.1018369

## P[X > 0.2]
sum(x > 0.2)/N

# [1] 0.58538
```

```
pbeta(0.2, a, b, lower.tail = FALSE)
```

```
# [1] 0.5876479
```

2.2 Re-amostragem por importância

O método de re-amostragem por importância (SIR - *Sampling Importance Resampling*) é uma alternativa ao método de aceitação-rejeição **quando não é possível determinar um valor razoável para M** . A essência do algoritmo é amostrar valores segundo pesos que são computados para cada um deles.

Veremos que o método nada mais é do que uma **extensão do método de amostragem por importância para aproximação de integrais**. No entanto, o objetivo aqui também é obter a distribuição completa da densidade alvo, e, com isso, calcular quantidades de interesse.

O método de re-amostragem por importância é também muito utilizado no contexto de **inferência bayesiana**, na obtenção de distribuições posteriores de parâmetros.

Lembrando que, na **amostragem por importância**, temos o seguinte cenário. Suponha que X seja uma VA com densidade $f(x)$. Seja $Y = g(X)/f(X)$, então

$$\int g(x)dx = \int \frac{g(x)}{f(x)} f(x)dx = E[Y]$$

Com isso, podemos estimar $E[Y]$ como uma integração simples de Monte Carlo, ou seja,

$$E[Y] = \frac{1}{m} \sum_{i=1}^m Y_i = \frac{1}{m} \sum_{i=1}^m \frac{g(X_i)}{f(X_i)}$$

onde as VAs X_1, \dots, X_m são geradas a partir da distribuição de $f(x)$. A função $f(x)$ é chamada de **função de importância**.

O algoritmo para o método de **amostragem por importância** é o seguinte:

1. Gerar uma amostra de m observações independentes da função de importância $f(x)$, $\{x_1, \dots, x_m\}$.
2. Calcular a razão entre $g(x)$ e $f(x)$ e determinar a média $\frac{1}{m} \sum_{i=1}^m \frac{g(x_i)}{f(x_i)}$

A média fornece diretamente uma estimativa de integral requerida.

Aproximando uma integral indefinida pela amostragem por importância

Voltando brevemente ao caso da exponencial, pela amostragem por importância, podemos agora obter uma estimativa de $P[X > 2]$, caso que não conseguimos usando a integração simples de Monte Carlo.

```
## Define funções
g <- function(x) exp(-x) * (x >= 2)
f <- function(x) dexp(x, 0.5) # Proposta é Exp(0.5)
## 1. Amostra da proposta
m <- 1e5
x <- rexp(m, 0.5)
## 2. Calcula a média da razão
mean(g(x)/f(x))

# [1] 0.1351572

## Teórico
pexp(2, 1, lower.tail = FALSE)

# [1] 0.1353353
```

O algoritmo do método de **re-amostragem por importância** é uma extensão do algoritmo anterior e consiste nos seguintes passos:

1. **(Amostragem)** Gerar uma amostra de m observações independentes da função de importância $f(x)$, $\{x_1, \dots, x_m\}$.
2. Calcular os **pesos**

$$w(x_i) = \frac{g(x_i)}{f(x_i)}$$

Os pesos dão uma ideia da probabilidade de aceitação de um determinado x_i . Como estes pesos podem ser maiores do que 1, eles precisam ser padronizados por

$$w^*(x_i) = \frac{w(x_i)}{\sum_{i=1}^m w(x_i)}$$

3. **(Re-amostragem por importância)** Amostrar **com reposição** $k < m$ valores de $\{x_1, \dots, x_m\}$, utilizando os pesos $\{w^*(x_1), \dots, w^*(x_m)\}$, como uma **distribuição discreta de probabilidades**.

Dessa forma, $\mathcal{X} = \{x_1^*, \dots, x_k^*\}$ será uma amostra de $g(X)$.

2.2.1 Exemplo (exponencial)

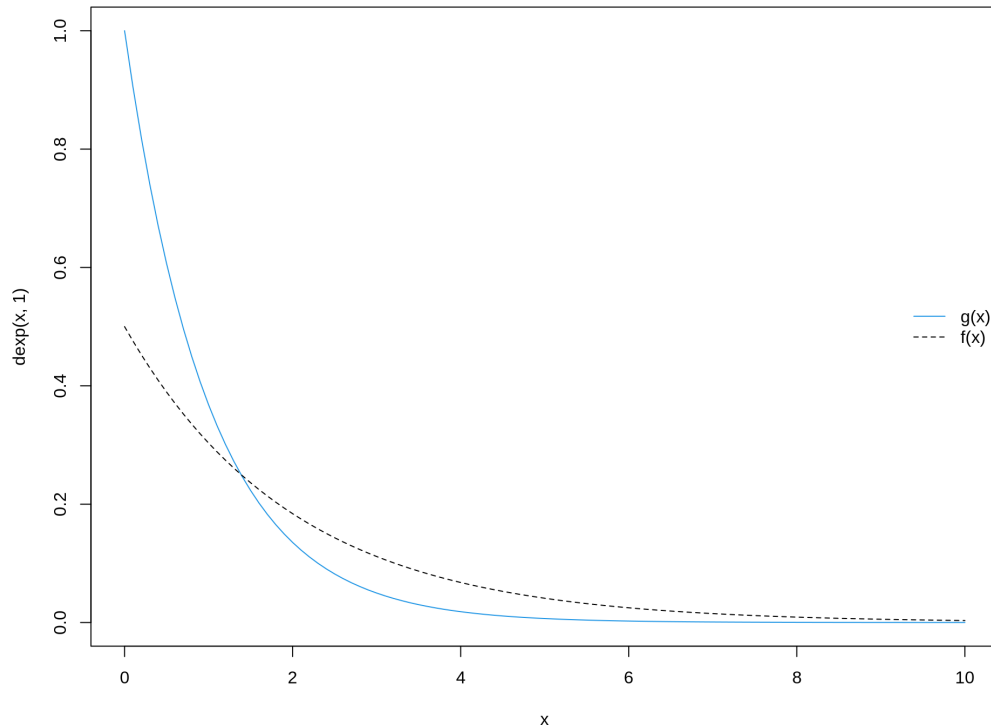
Seguindo com o exemplo da $\text{Exp}(1)$, o código abaixo serve para gerar uma amostra por importância desta distribuição.

A função de importância considerada aqui é uma $\text{Exp}(0.5)$. Lembre-se que nesse caso, **não é necessário que a função proposta seja maior do que a função alvo**.

```

curve(dexp(x, 1), from = 0, to = 10, col = 4)
curve(dexp(x, 0.5), from = 0, to = 10, add = TRUE, lty = 2)
legend("right", legend = c("g(x)", "f(x)"),
      lty = c(1, 2), col = c(4, 1), bty = "n")

```



```

## Define funções
g <- function(x) exp(-x) * (x >= 0)
f <- function(x) dexp(x, 0.5)
## 1. Amostragem
m <- 1e5
x <- rexp(m, 0.5)
## 2. Calcula pesos
w <- g(x)/f(x)
w.norm <- w/sum(w)
## 3. Re-amostragem por importância
k <- 1e4
## Amostra considerando os pesos normalizados para cada X
g.sample <- sample(x, size = k, replace = TRUE, prob = w.norm)
## Tamanho da amostra
length(g.sample)

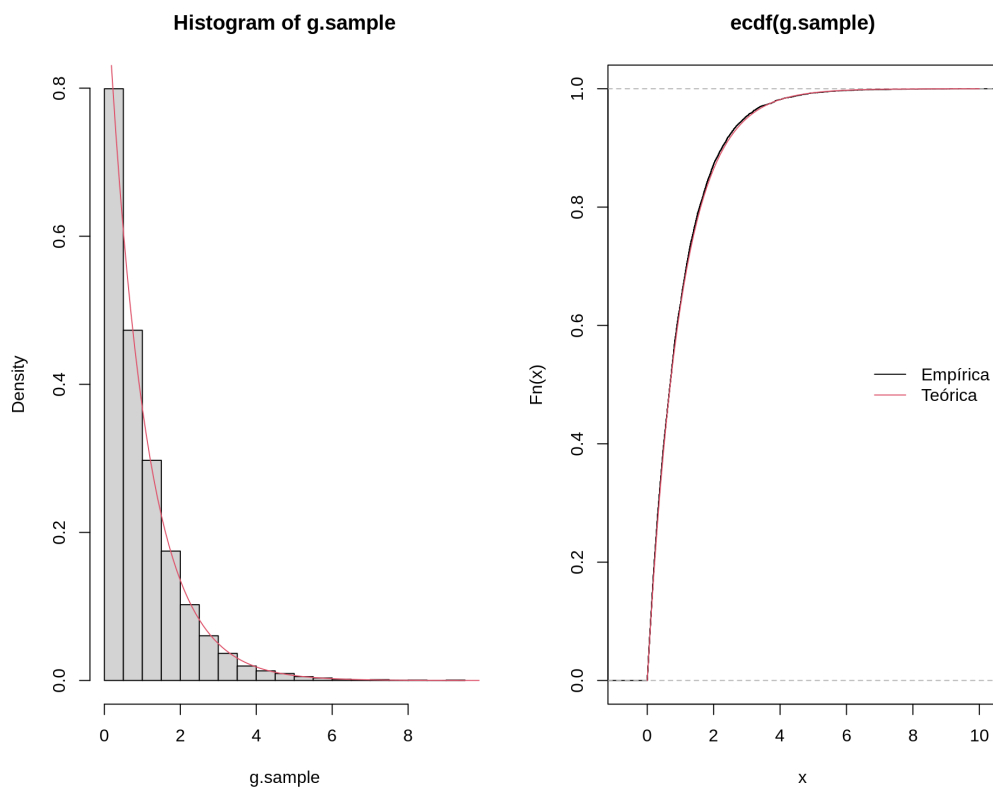
# [1] 10000

## Integral
sum(g.sample > 0)/k

```

```
# [1] 1
```

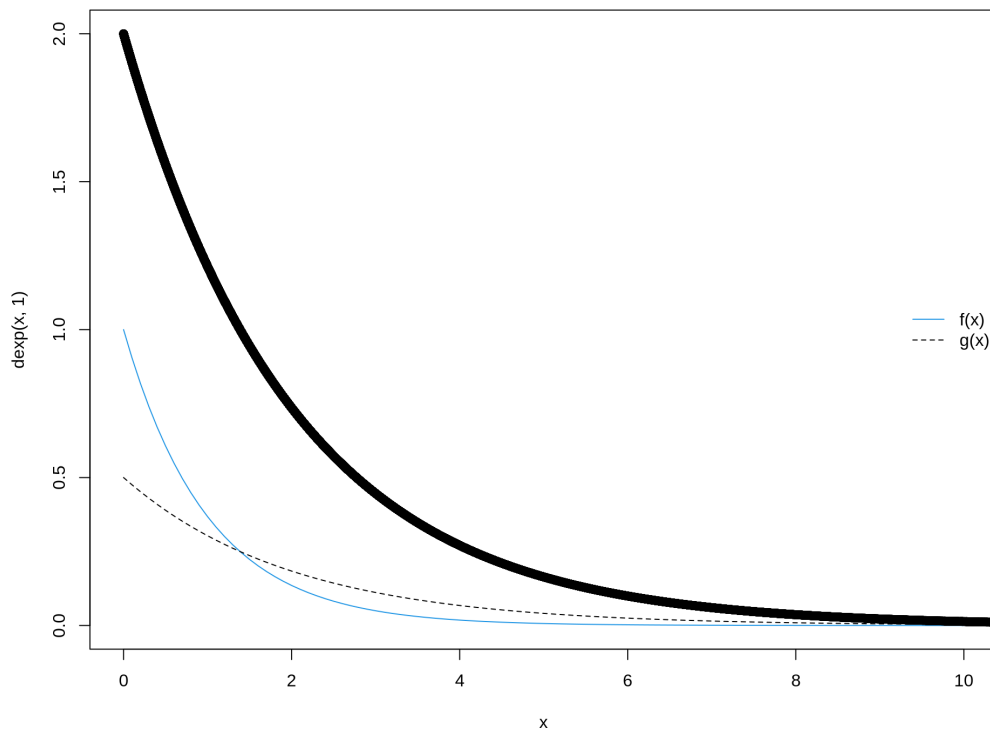
```
## Visualização da amostra
par(mfrow = c(1, 2))
hist(g.sample, freq = FALSE)
curve(dexp(x, 1), add = TRUE, from = 0, to = 10, col = 2)
plot(ecdf(g.sample))
curve(pexp(x, 1), add = TRUE, from = 0, to = 10, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
      lty = 1, col = 1:2, bty = "n")
```



```
par(mfrow = c(1, 1))
```

É possível ver porque o método funciona quando vemos um gráfico dos valores amostrados de X com seus respectivos pesos:

```
curve(dexp(x, 1), from = 0, to = 10, col = 4, ylim = c(0, 2))
curve(dexp(x, 0.5), from = 0, to = 10, add = TRUE, lty = 2)
legend("right", legend = c("f(x)", "g(x)"),
      lty = c(1, 2), col = c(4, 1), bty = "n")
points(x, w)
```

Com uma amostra completa da distribuição, podemos então calcular qualquer tipo de probabilidades:

```
## P[2 < X < 4]
sum(g.sample > 2 & g.sample < 4)/k

# [1] 0.1095
```

```
## Teórica
pexp(4, 1) - pexp(2, 1)

# [1] 0.1170196
```

```
## P[X > 2]
sum(g.sample > 2)/k

# [1] 0.1278
```

```
pexp(2, 1, lower.tail = FALSE)

# [1] 0.1353353
```

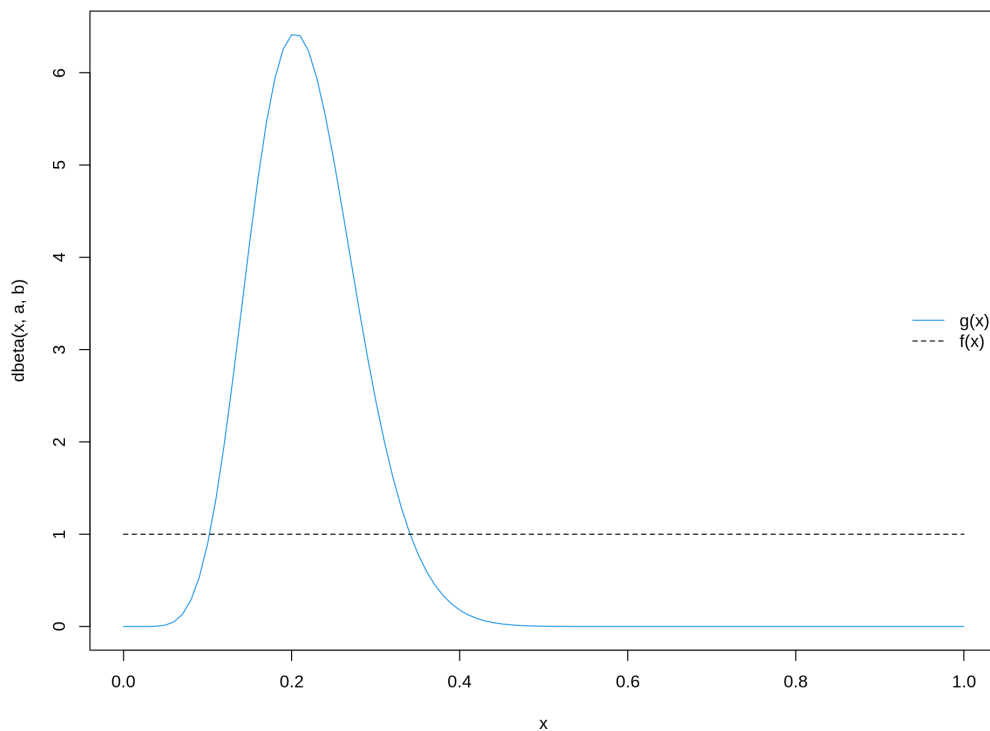
2.2.2 Exemplo (beta)

Voltando ao exemplo de $X \sim \text{Beta}(a = 9.39, b = 33.67)$, vamos obter uma amostra desta distribuição pelo método de re-amostragem por importância. Aqui também utilizaremos uma $U(0, 1)$ como função de importância.

```

a <- 9.39; b <- 33.67
curve(dbeta(x, a, b), from = 0, to = 1, col = 4)
curve(dunif(x, 0, 1), from = 0, to = 1, add = TRUE, lty = 2)
legend("right", legend = c("g(x)", "f(x)"),
      lty = c(1, 2), col = c(4, 1), bty = "n")

```



```

## Define funções
g <- function(x, a, b) dbeta(x, a, b)
f <- function(x) dunif(x) # usando U(0,1)
## 1. Amostragem
m <- 1e5
x <- runif(m)
## 2. Calcula pesos
w <- g(x, a, b)/f(x)
w.norm <- w/sum(w)
## 3. Re-amostragem por importância
k <- 1e4
## Amostra considerando os pesos normalizados para cada X
g.sample <- sample(x, size = k, replace = TRUE, prob = w.norm)
## Tamanho da amostra
length(g.sample)

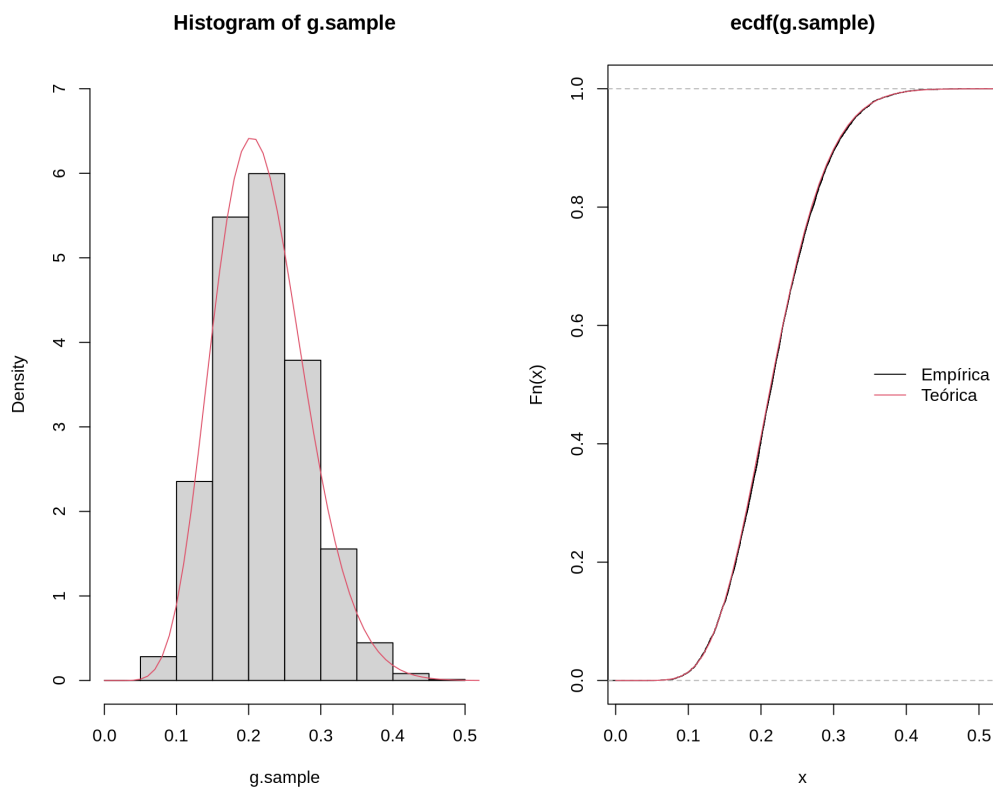
# [1] 10000

## Integral
sum(g.sample >= 0)/k

```

```
# [1] 1
```

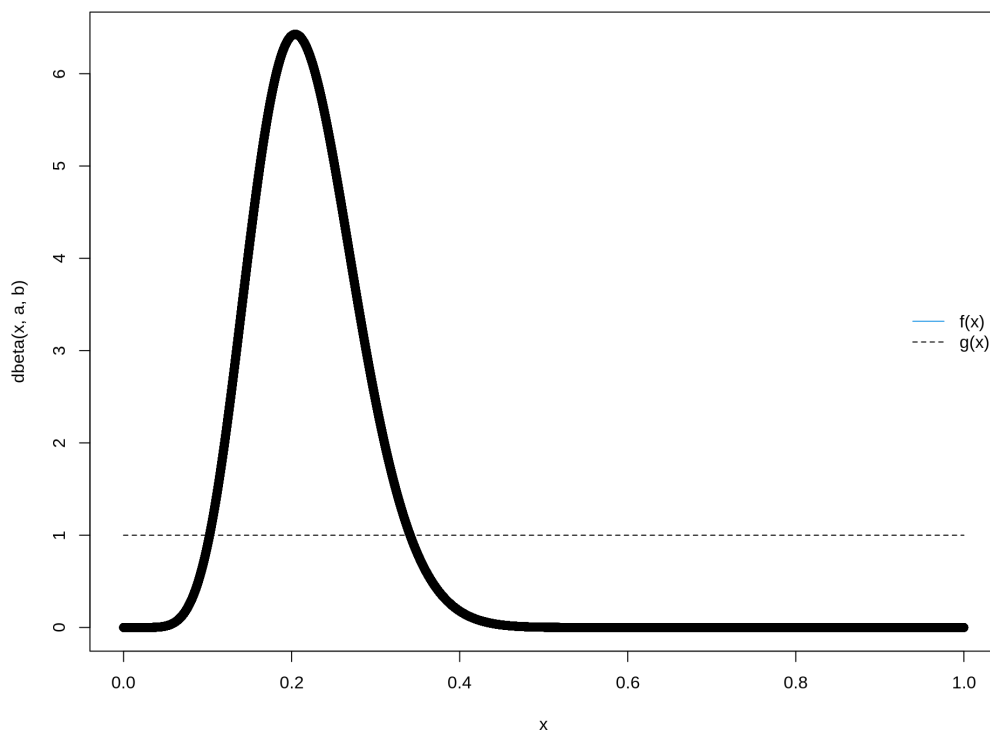
```
## Visualização da amostra
par(mfrow = c(1, 2))
hist(g.sample, freq = FALSE, ylim = c(0, 7))
curve(dbeta(x, a, b), add = TRUE, from = 0, to = 1, col = 2)
plot(ecdf(g.sample))
curve(pbeta(x, a, b), add = TRUE, from = 0, to = 1, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
      lty = 1, col = 1:2, bty = "n")
```



```
par(mfrow = c(1, 1))
```

Novamente podemos conferir a relação entre os valores amostrados de X e seus respectivos pesos calculados:

```
a <- 9.39; b <- 33.67
curve(dbeta(x, a, b), from = 0, to = 1, col = 4)
curve(dunif(x, 0, 1), from = 0, to = 1, add = TRUE, lty = 2)
legend("right", legend = c("f(x)", "g(x)"),
      lty = c(1, 2), col = c(4, 1), bty = "n")
points(x, w)
```



Calcula probabilidades:

```
## P[0.3 < X < 0.5]
sum(g.sample > 0.3 & g.sample < 0.5)/k
```

```
# [1] 0.1048
```

```
pbeta(0.5, a, b) - pbeta(0.3, a, b)
```

```
# [1] 0.1018369
```

```
## P[X > 0.2]
sum(g.sample > 0.2)/k
```

```
# [1] 0.594
```

```
pbeta(0.2, a, b, lower.tail = FALSE)
```

```
# [1] 0.5876479
```

A avaliação da proximidade entre a distribuição alvo f , e a função geradora (ou de importância) g ainda é um tema de estudo em aberto. West (1993) propôs a medida chamada *Entropia Relativa à Uniformidade* (ERU), definida por

$$\text{ERU} = -\frac{\sum w^*(x_i) \log w^*(x_i)}{\log m}$$

Quando a distribuição alvo e a proposta são idênticas, então os pesos serão constantes e iguais a $1/m$, fazendo com que $ERU = 1$ seja o valor máximo. À medida que a entropia entre os $w^*(x_i)$ aumenta, $ERU < 1$.

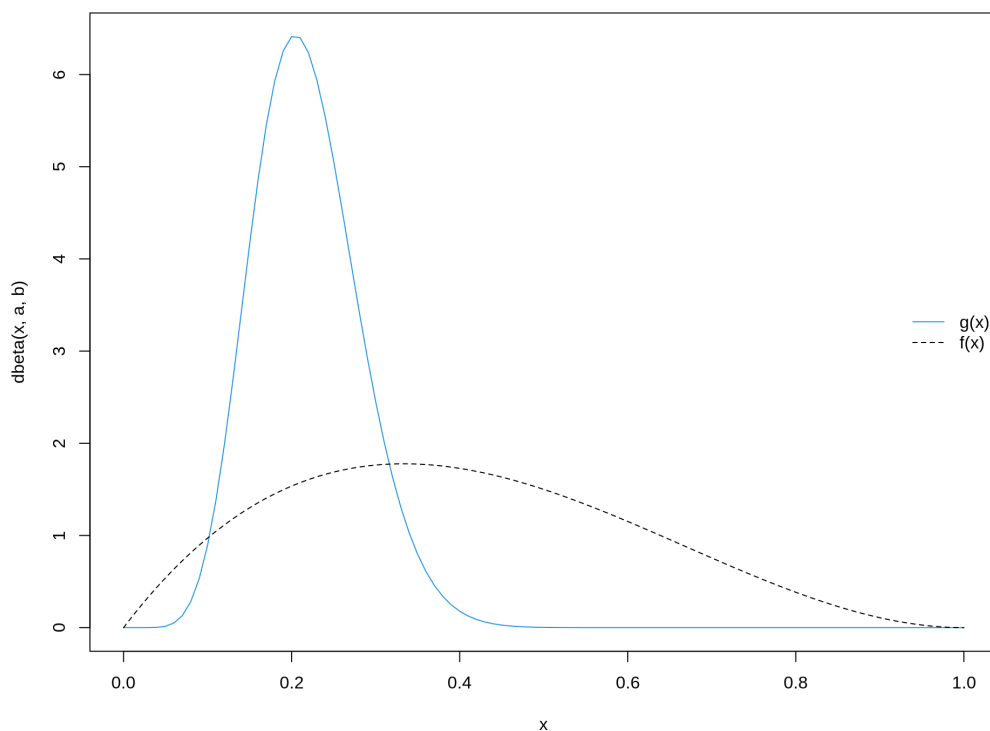
Para distribuições unidimensionais, valores de $ERU > 0.9$ são recomendados. Em casos multidimensionais, valores mais baixos são aceitáveis. Nesse caso, usando a Uniforme, esse valor foi

```
-sum(w.norm * log(w.norm))/log(m)
```

```
# [1] 0.8806501
```

Usando agora uma $Beta(2, 3)$ como função de importância.

```
a <- 9.39; b <- 33.67
curve(dbeta(x, a, b), from = 0, to = 1, col = 4)
curve(dbeta(x, 2, 3), from = 0, to = 1, add = TRUE, lty = 2)
legend("right", legend = c("g(x)", "f(x)"),
      lty = c(1, 2), col = c(4, 1), bty = "n")
```



```
## Define funções
g <- function(x, a, b) dbeta(x, a, b)
f <- function(x) dbeta(x, 2, 3) # usando Beta(2, 3)

## 1. Amostragem
m <- 1e5
x <- rbeta(m, 2, 3)

## 2. Calcula pesos
w <- g(x, a, b)/f(x)
w.norm <- w/sum(w)

## 3. Re-amostragem por importância
k <- 1e4

## Amostra considerando os pesos normalizados para cada X
g.sample <- sample(x, size = k, replace = TRUE, prob = w.norm)

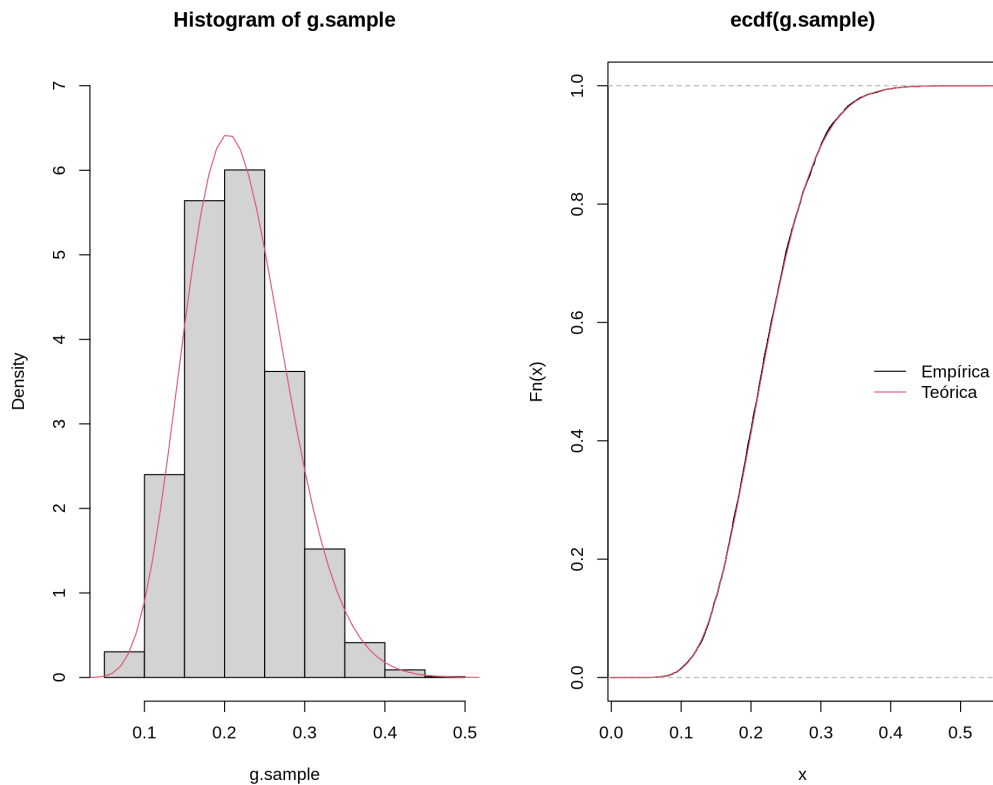
## Tamanho da amostra
length(g.sample)

# [1] 10000

## Integral
sum(g.sample >= 0)/k

# [1] 1

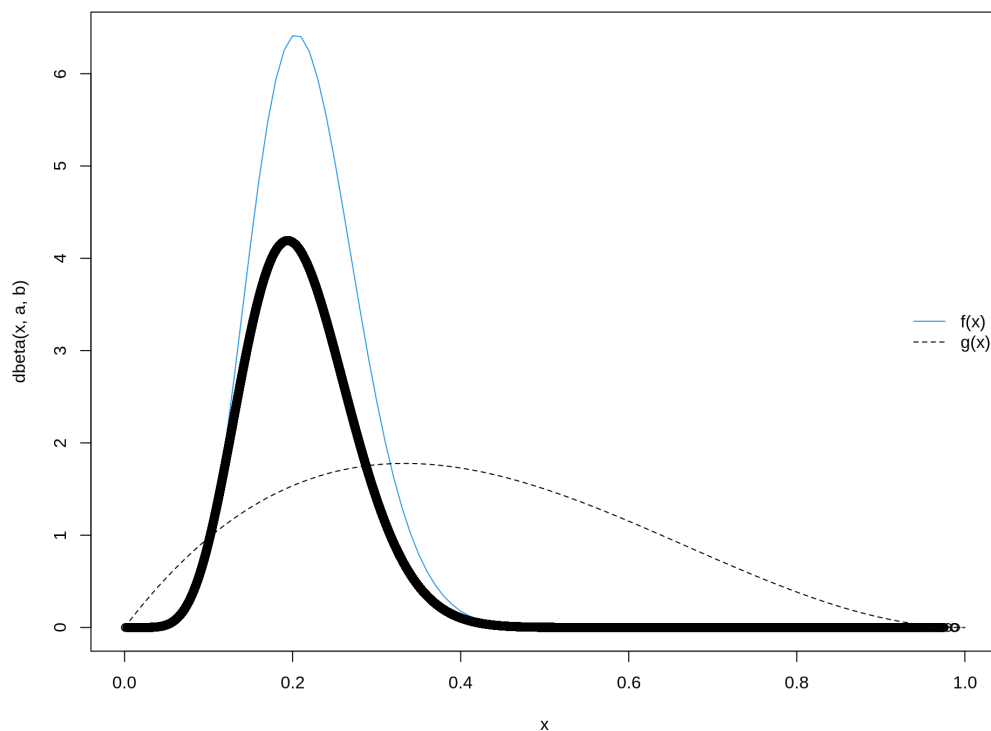
## Visualização da amostra
par(mfrow = c(1, 2))
hist(g.sample, freq = FALSE, ylim = c(0, 7))
curve(dbeta(x, a, b), add = TRUE, from = 0, to = 1, col = 2)
plot(ecdf(g.sample))
curve(pbeta(x, a, b), add = TRUE, from = 0, to = 1, col = 2)
legend("right", legend = c("Empírica", "Teórica"),
      lty = 1, col = 1:2, bty = "n")
```



```
par(mfrow = c(1, 1))
```

Novamente podemos conferir a relação entre os valores amostrados de X e seus respectivos pesos calculados:

```
a <- 9.39; b <- 33.67
curve(dbeta(x, a, b), from = 0, to = 1, col = 4)
curve(dbeta(x, 2, 3), from = 0, to = 1, add = TRUE, lty = 2)
legend("right", legend = c("f(x)", "g(x)"),
      lty = c(1, 2), col = c(4, 1), bty = "n")
points(x, w)
```



O ERU nesse caso foi

```
-sum(w.norm * log(w.norm))/log(m)
```

```
# [1] 0.9170381
```

A obtenção de uma amostra da distribuição completa da VA de interesse possui diversas finalidades. Por exemplo, no caso da beta, podemos obter um resumo

```
summary(g.sample)
```

```
#      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.0515  0.1730  0.2130  0.2176  0.2574  0.4965
```

ou, por exemplo, saber o intervalo onde se encontram 95% dos dados:

```
quantile(g.sample, probs = c(.025, .975))
```

```
#      2.5%      97.5%
# 0.1087736 0.3505009
```

Esse tipo de resumo será utilizado nos métodos de inferência que veremos mais adiante.

No contexto de inferência bayesiana, as distribuições geradas por estes métodos se referem às **distribuições posteriores dos parâmetros** objetos da inferência. Dessa forma, o intervalo acima, por exemplo, nos forneceria diretamente um “intervalo de credibilidade” de 95% para os valores do parâmetro.



(https://creativecommons.org/licenses/by-nc-sa/4.0/deed.pt_BR)

Este conteúdo está disponível por meio da Licença Creative Commons 4.0