Estatística Computacional

Simulação de variáveis aleatórias - Parte 1

Prof. Paulo Cerqueira Jr
Faculdade de Estatística - FAEST
Programa de Pós-graduação em Matemática e Estatística - PPGME

https://github.com/paulocerqueirajr

Introdução

Introdução

> Estatística computacional envolve transformar teoria e métodos em algoritmos e cálculos numéricos reais com dados.

- Trata-se de resolver problemas computacionais reais que surgem quando visualizamos, analisamos e modelamos dados.
- A **estatística computacional** não é um tópico único e coerente, mas sim um grande número de técnicas computacionais vagamente relacionadas que usamos em estatística.
- Em vez disso, alguns tópicos e metodologias estatísticas selecionados são tratados com algum detalhe.
- Pretende-se que boas práticas computacionais possam ser aprendidas a partir desses tópicos e transferidas para outras metodologias estatísticas conforme necessário.

Introdução

- Esta disciplina é inteiramente dedicada à exposição de métodos para simulação de variáveis aleatórias.
- A discussão dos algoritmos será fundamentada pela Teoria das Probabilidades.
- Algumas aplicações serão desenvolvidas ao final desta unidade tais como:
  - Geração de variáveis uniformes e não uniformes;
  - Integração Monte Carlo.

Introdução

> Simulação computacional significa usar um computador para recriar modelos matemáticos de sistemas ou processos em diferentes cenários.

- As simulações podem ser determinísticas ou estocásticas.

Introdução

> Por que é necessário saber gerar variáveis aleatórias?

- Gerar bancos de dados para avaliar o desempenho de modelos propostos;
- Aproximar integrais. Ex.: algoritmo EM Monte Carlo;
- Estatística Bayesiana: Gerar amostras da distribuição a posteriori quando a mesma não tem expressão fechada conhecida para sua função de densidade ou função massa de probabilidade.

Exemplo de Simulação determinística

- Considere o problema de condução de calor numa chapa metálica quadrada.
- Para simular este sistema é necessário resolver a seguinte equação diferencial parcial

$$k \frac{\partial T(x, y, t)}{\partial t}.$$

sujeita às condições de contorno nas bordas da chapa.

Exemplo de Simulação estocástica

- Suponha que clientes chegam para utilizar um certo serviço e são atendidos por k servidores onde k é um inteiro positivo.
- Suponha que os tempos entre chegadas sucessivas de clientes são variáveis aleatórias exponencialmente distribuídas com taxa λ e que o tempo de serviço prestado a cada cliente é uma variável aleatória exponencialmente distribuída com taxa μ.
- Além disso, suponha que tempos de serviço associados a clientes distintos são independentes. Quando existem mais de k clientes no sistema, o excesso forma um fila de espera até que um deles seja chamado por um dos servidores.

Eventualmente estaremos interessados em responder questões tais como:

- Quais cenários (valores de λ, μ e k) são capazes de produzir longas filas?
- Qual é o número médio de clientes no sistema?
- Qual é o tempo médio gasto por um cliente no sistema?

Simulação de Números Pseudo-Aleatórios

Simulação de Números Pseudo-Aleatórios

- Antigamente números aleatórios eram gerados mecanicamente ou manualmente, usando roletas, urnas, moedas ou dados.
- A abordagem moderna consiste em usar um computador para gerar uma sequência de números que para todos os fins práticos imita uma amostra aleatória simples de uma uniforme no intervalo $(0, 1)$.
- Na verdade, estes números são apenas pseudo-aleatórios pois, como veremos, eles são gerados de forma determinística.
- Obviamente isto pode trazer dúvidas quanto a eficiência de tais geradores como fontes de aleatoriedade. Porém a discuss˜ao deste tema é difícil e foge completamente do escopo do curso.
- Vamos assumir simplesmente que os algoritmos que serão apresentados produzem uma sequência de variáveis aleatórias i.i.d. uniformemente distribuídas no intervalo $(0, 1)$.

Operação $x \bmod n$

Operação $x \mod n$

Seja $n \in \mathbb{Z}$ e defina uma relação R em $\mathbb{Z}$ tal como segue

$$xRy \text{ se, e somente se, } x - y = kn, \ k \in \mathbb{Z}$$

A classe de equivalência de $y \in \mathbb{Z}$ é definida pelo conjunto,

$$\bar{y} = \{x \in \mathbb{Z}; xRy\} = \{x \in \mathbb{Z}; x = kn + y, k \in \mathbb{Z}\}.$$

A relação R induz uma partição em $\mathbb{Z}$ tal como segue

$$\mathbb{Z} = \bar{0} \cup \bar{1} \cup \bar{2} \cup \cdots \cup \overline{n-1}.$$

Operação $x \bmod n$

Ou seja,

$$\begin{cases} \bar{0} = \{x \in \mathbb{Z}; x = kn, k \in \mathbb{Z}\} = \{\dots, -2n, -n, 0, n, 2n, \dots\} \\ \bar{1} = \{x \in \mathbb{Z}; x = kn + 1, k \in \mathbb{Z}\} = \{\dots, -2n+1, -n+1, 1, n+1, 2n+1, \dots\} \\ \bar{2} = \{x \in \mathbb{Z}; x = kn + 2, k \in \mathbb{Z}\} = \{\dots, -2n+2, -n+2, 2, n+2, 2n+2, \dots\} \\ \vdots \\ \overline{n-1} = \{x \in \mathbb{Z}; x = kn + n, k \in \mathbb{Z}\} = \{\dots, -n, -1, -1, n-1, 2n-1, \dots\} \end{cases}$$

Note que, $\bar{n} = \bar{0}, \overline{n+1} = \bar{1}$, etc.

Operação $x \bmod n$

A operação $x \bmod n$, $x \in \mathbb{Z}$ e $n \in \mathbb{Z}$, é definida por

$$x \bmod n = \begin{cases} 0 & x \in \bar{0} \\ 1 & x \in \bar{1} \\ 2 & x \in \bar{2} \\ \vdots & \vdots \\ n-1 & x \in \overline{n-1} \end{cases}$$

Exemplo $x \bmod n$

Como exemplo, vejamos a operação $x \bmod 4$:

$$x \bmod 4 = \begin{cases} 0 & x \in \bar{0} = \{\ldots, -8, -4, 0, 4, 8, \ldots\} \\ 1 & x \in \bar{1} = \{\ldots, -7, -3, 1, 5, 9, \ldots\} \\ 2 & x \in \bar{2} = \{\ldots, -6, -2, 2, 6, 10, \ldots\} \\ 3 & x \in \bar{3} = \{\ldots, -5, -1, 3, 7, 11, \ldots\} \end{cases}$$

No R

No R a operação $x \bmod n$ é realizada com o operador %%.

```
8%%4 # resto de divisão 8/4
```

```
[1] 0
```

```
11%%4 # resto de divisão 11/4
```

```
[1] 3
```

```
2%%4 # resto de divisão 2/4
```

```
[1] 2
```

```
-3%%4 # resto de divisão -3/4
```

```
[1] 1
```

Método Congruencial Linear

Método Congruencial Linear

- Sejam a, b e n inteiros positivos e seja x_1 um natural menor do que n.
- Este método propõe uma sequência de números pseudo-aleatórios gerada pela seguinte equação de recorrência:

$$x_{k+1} = (ax_k + b) \bmod n,$$

$k \geq 1$. Tal como foi definida $x_k \in \{0, 1, \ldots, n-1\}$ para todo $k \geq 1$. Portanto para obter uma sequência $(u_k)_{k\geq 1}$ com valores em $(0, 1)$, basta fazer

$$u_k = \frac{x_k}{n}, \quad k \geq 1.$$

Exemplo 1:

Faça $a = 123, b = 971, n = 1137$ e $x_1 = 27$.

```r
x <- NULL
x[1] <- 27
for(k in 1:20){
  x[k+1] <- ((123*x[k] + 971) %% 1137)
}
u <- x/1137 # Seq. uniforme.
u
```

```
 [1] 0.023746702 0.774846086 0.160070361 0.542656113 0.600703606 0.740545295
 [7] 0.941072999 0.605980651 0.389621812 0.777484609 0.484608619 0.460861917
[13] 0.540017590 0.276165347 0.822339490 0.001759015 0.070360598 0.508355321
[19] 0.381706245 0.803869833 0.729991205
```

Exemplo: Ciclos peródicos

Certas escolhas dos valores de a, b e n, podem gerar sequências com ciclos periódicos.

Faça $a = 6, b = 7, n = 5$ e $x_1 = 2$.

```r
x <- NULL
x[1] <- 2
for(k in 1:12){
  x[k+1] <- ((6*x[k] + 7) %% 5)
}
x
```

```
[1] 2 4 1 3 0 2 4 1 3 0 2 4 1
```

```r
u <- x/5
u
```

```
[1] 0.4 0.8 0.2 0.6 0.0 0.4 0.8 0.2 0.6 0.0 0.4 0.8 0.2
```

Exemplo: Ciclos perpódicos

Faça $a = 1, b = 1, n = 10$ e $x_1 = 3$.

```
x <- NULL
x[1] <- 3
for(k in 1:15){
  x[k+1] <- ((x[k] + 1) %% 10)
}
x
```

```
[1] 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
```

```
u <- x/10
u
```

```
[1] 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
```

Exemplo

Algumas boas receitas:

- $x_{k+1} = (44485709377909x_k + 375) \bmod 248;$
- $x_{k+1} = (16807x_k + 8437) \bmod (231 - 1).$

Semente

- No programa `R` podemos gerar números aleatórios uniformes contínuos utilizando uma função pré-programada.
- Os números aleatórios uniformes são gerados pelo comando `runif(n, min, max)`, em que:
  - `n` é o tamanho da sequência;
  - `min` e `max` são argumentos que delimitam o valor mínimo e máximo da sequência a ser gerada.
- O controle da semente para se gerar uma sequência reproduzível de números uniformes é dada pelo comando `set.seed(semente)`, onde o argumento `semente` deve ser um número inteiro.
- O R automaticamente determina a cada chamada uma nova semente.

Semente

Conseguimos gerar diferentes sequências em cada chamada do comando `runif(n, min, max)`, sem nos preocuparmos com a semente aleatória.

Devemos reiterar que o programa R oferece seis geradores de eventos uniformes:

Wichmann-Hill De período aproximado 7×10^{12}.

Marsaglia-Multicarry De período aproximadamente igual a 2^{60}, passa em testes rigorosos.

Super-Duper De período aproximado 5×10^{18}, não passa em alguns testes rigorosos.

Mersenne-Twister De período $2^{19937} - 1$ e boa distribuição em espaços de dimensões inferiores a 623. (*default* !)

Knuth-TAOCP Este gerador é definido pela relação $u_j = (u_{j-100} - u_{j-37}) \mod 2^{30}$ e seu periodo é de aproximadamente $2^1 29$.

Uma versão atualizada e melhorada do anterior.

Uniformes no R

Usando a função do R para gerar valores de uma uniforme (`runif`):

```r
(x <- runif(n=10, min=0, max=1))
```

```
[1] 0.14554156 0.09331545 0.41229396 0.33507436 0.22957404 0.87948652
[7] 0.37231981 0.61602233 0.50631418 0.70808067
```

```r
(x <- runif(n=10, min=1, max=2))
```

```
[1] 1.041380 1.689070 1.631495 1.556898 1.993820 1.646151 1.592357 1.164192
[9] 1.706494 1.946865
```

```r
(x <- runif(n=10, min=-1, max=1))
```

```
[1] -0.255531147  0.152504665  0.001797940 -0.810972475  0.239659859
[6] -0.799610405 -0.713856413 -0.828371328  0.001905327  0.513070924
```

Método da Transformação Inversa

Método da Transformação Inversa

Seja X uma v.a. com função de distribuição acumulada F contínua. Seja $F^{-1} : (0, 1) \to \mathbb{R}$ uma função definida por

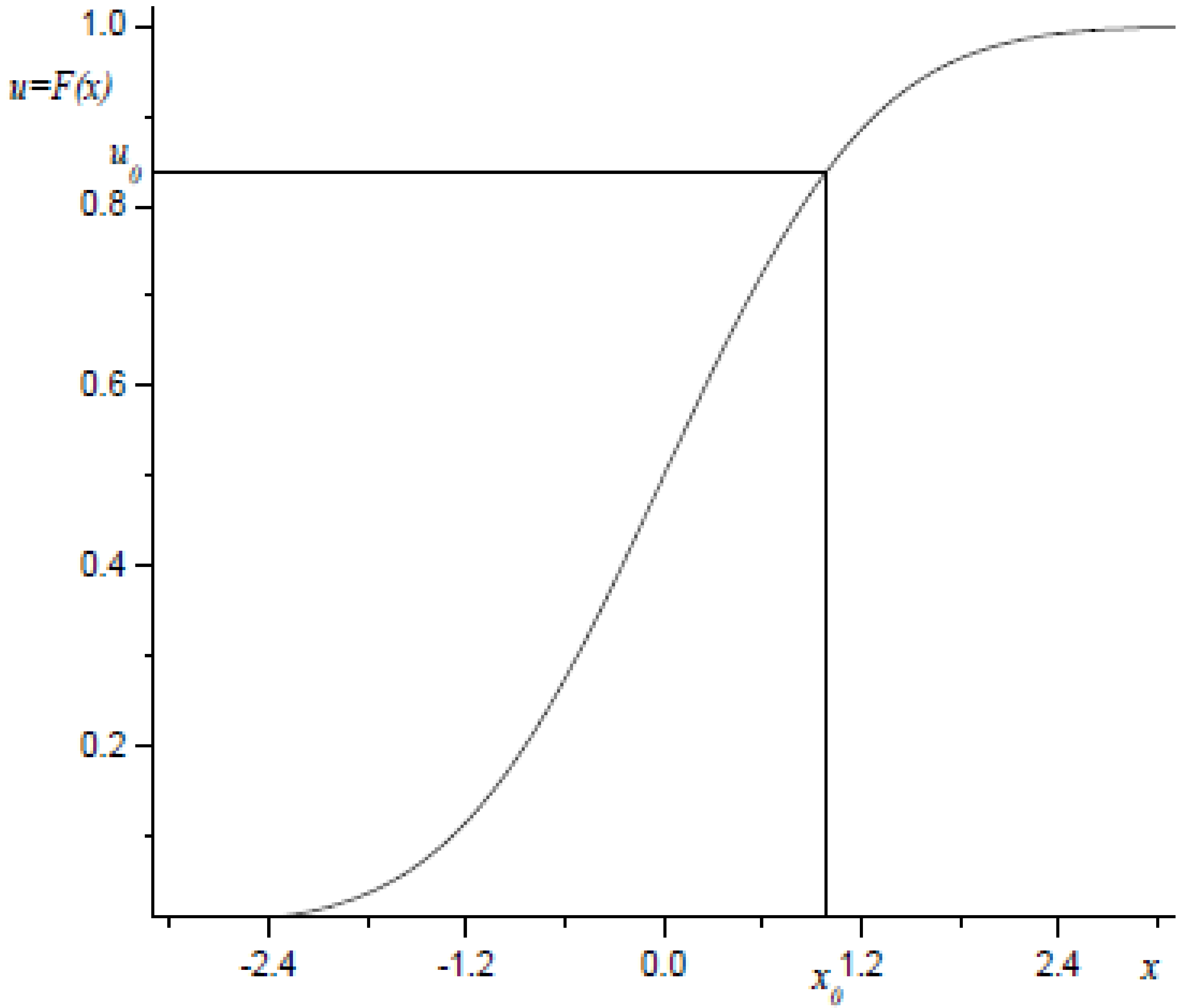
$$F^{-1}(u) = \inf\{x \in \mathbb{R} : F(x) \geq u\}.$$

A função F^{-1} é a função generalizada de F.

> Proposição 1: $U = F(X) \sim U(0, 1)$

> Proposição 2: seja $U \sim U(0, 1)$ e seja X uma v.a. com função de distribuição acumulada F contínua. Então, $F^{-1}(U) \sim F$, ou seja, X possui a mesma distribuição que $F^{-1}(U)$.

Graficamente



Simulação de $X \sim Exp(1)$.

- Temos que a função de distribuição é $F(x) = 1 - e^{-x}$.
- Logo, usando o método da transformação inversa temos que,

$$u = F(x) \Leftrightarrow u = 1 - e^{-x} \Leftrightarrow 1 - u = e^{-x}.$$

$$\log(1-u) = -x \Leftrightarrow x = -\log(1-u).$$

Assim, $X = -\log(1-U) \sim Exp(1)$, em que $U \sim Unif(0,1)$.

Exemplo

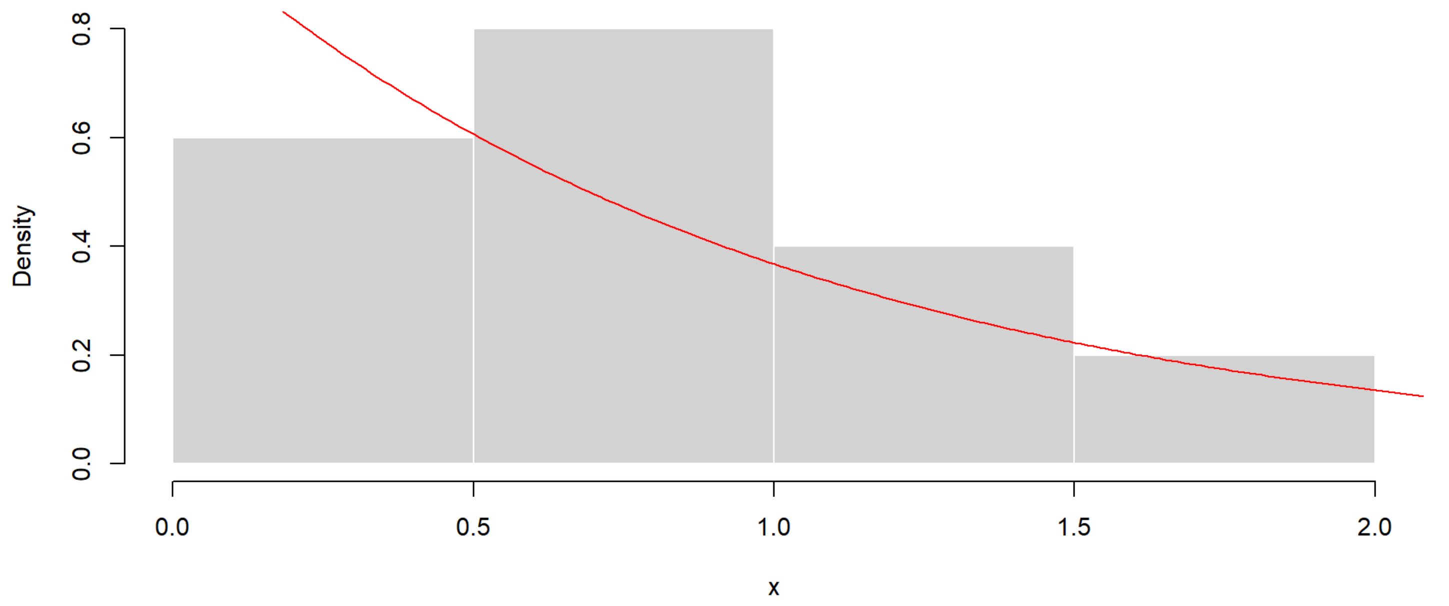
> Simulação de $X \sim Exp(1)$.

Vamos gerar 10 valores de uma distribuição exponencial com taxa 1:

```r
u <- runif(10,0,1)
x <- -log(1-u)
summary(x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu. 
Max. 
0.08763 0.41454 0.85939 0.83117 1.17131 
1.64151 
```

```r
hist(x, main="", border=FALSE,
     freq = FALSE)
aux <- seq(0,10, length.out=1000)
lines(aux, dexp(x=aux, rate = 1),
      col="red")
```



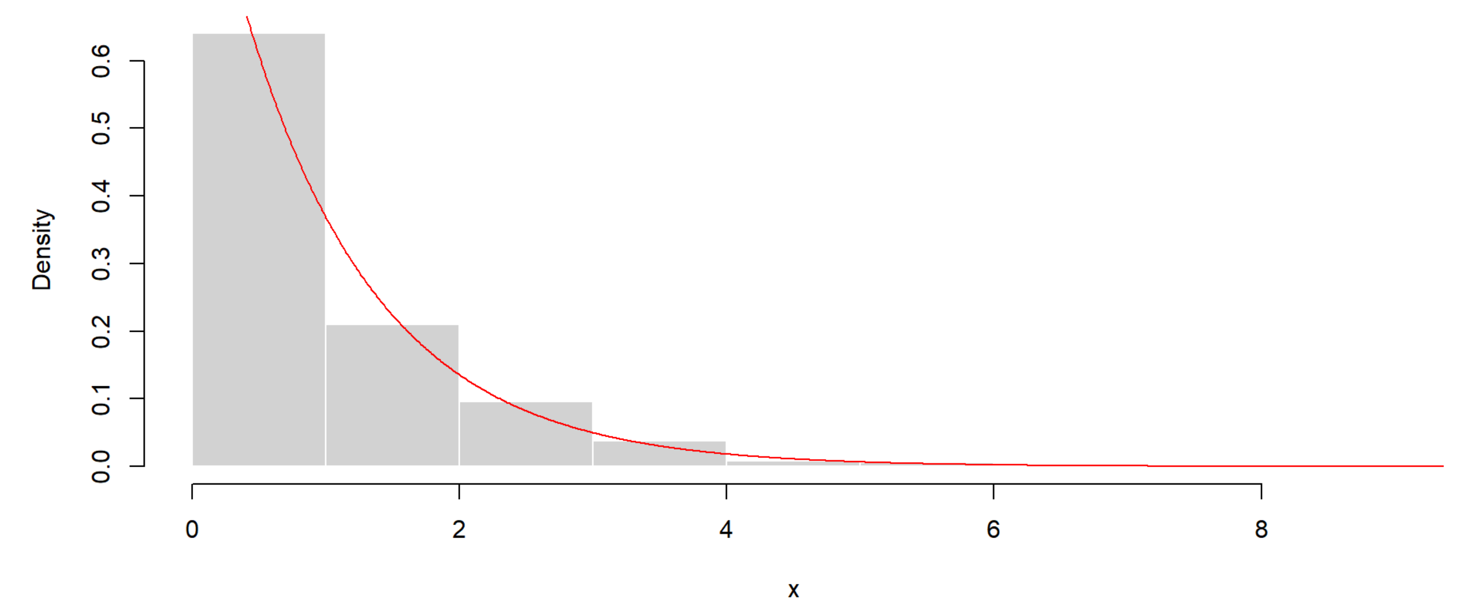
Simulação de $X \sim Exp(1)$.

Vamos gerar 500 valores de uma distribuição exponencial com taxa 1:

```r
u <- runif(500,0,1)
x <- -log(1-u)
summary(x)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.
Max.
0.007822 0.323116 0.656488 1.011301 1.380847
8.965339
```

```r
hist(x, main="", border=FALSE,
     freq = FALSE)
aux <- seq(0,10, length.out=1000)
lines(aux, dexp(x=aux, rate = 1),
      col="red")
```



Método da transformação inversa

Algoritmo genérico:

1. Obtenha a função F^{-1}, ou seja, obtenha a inversa generalizada de F (onde F representa a função de distribuição de X);
2. Simule $U \sim U(0,1)$;
3. Faça $X = F^{-1}(U)$. A proposição 2 garante que $X \sim F$.

Aviso:

- E se F não possuir função inversa de forma analítica?
- Como resolver?

Método da transformação inversa

- Precisamos lembrar da seguinte relação do método:

$$u = F(x) \;\Rightarrow\; F(x) - u = 0$$

- Observe que $F(x) - u = 0$ é uma função em que a raiz é x.
- Dessa forma, podemos usar métodos de otimização numérica no caso univariado.

Exemplo - Modelo $N(0, 1)$

- Sabemos por definção do modelo que

$$F(x) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\{-0.5X\} dx.$$

- Neste caso, só conseguiremos resolver, mudando as coordenadas ou através de algum método numérico de integração.
- No R:

```r
## Função para otimizar
f <- function(x, u){
  pnorm(x, 0, 1) - u
}
```

Exemplo - Modelo $N(0, 1)$

Função de otimização:

- No R dispomos de diversar funções para encontrar raiz ded funções univariadas.
- Usaremos a função `uniroot` como proposta.

- Logo, para gerar um valor da variável X:

```
uniroot(f, interval = c(-5, 5),
        u=runif(1))
```

```
$root
[1] 0.9666498

$f.root
[1] -2.174406e-07

$iter
[1] 9

$init.it
[1] NA
```

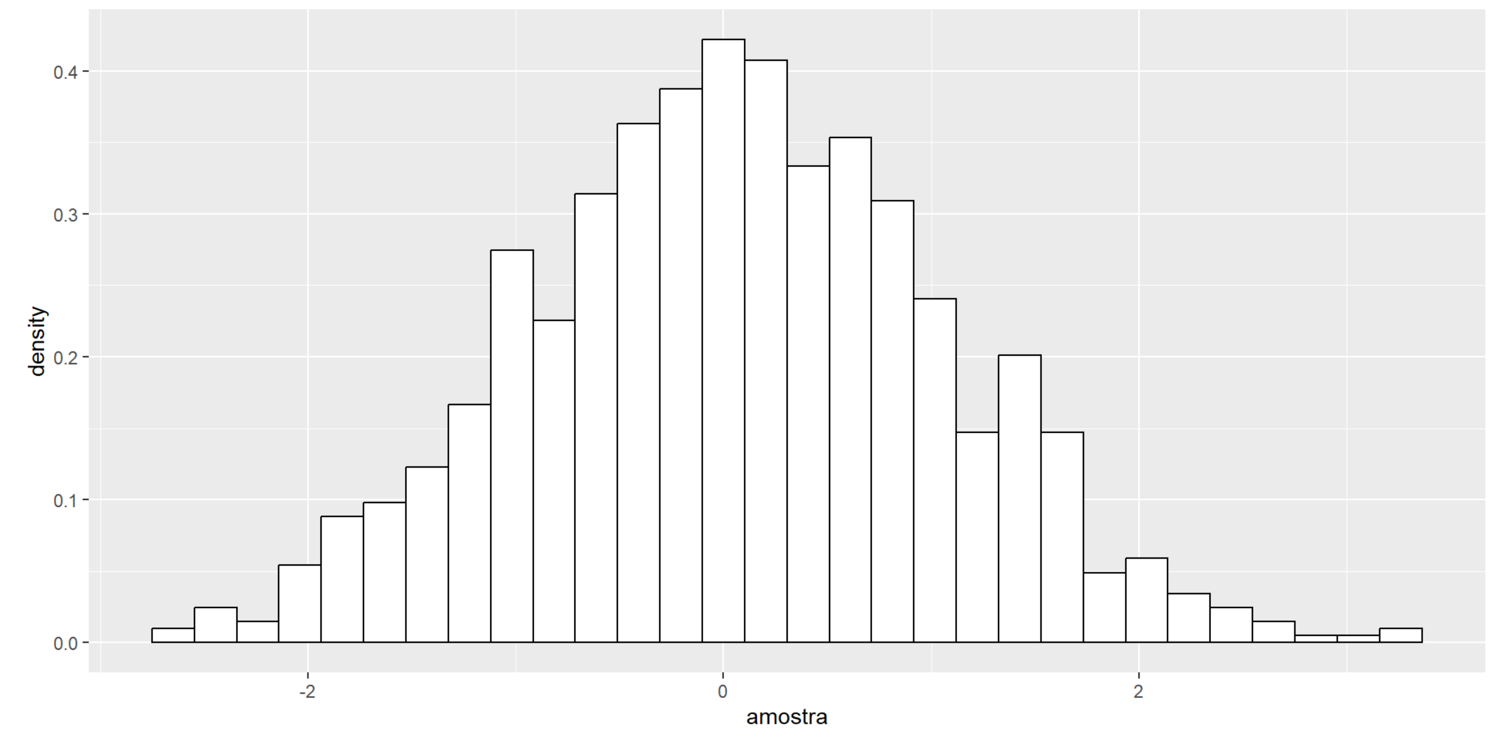
Exemplo - Modelo $N(0, 1)$

- Para gerar uma amostra de tamanho n basta fazer:

```
n <- 1000
amostra <- vector(mode = "numeric"
                  length = n)

for(i in 1:n){
  amostra[i]  <- uniroot(f,
                   interval = c(-
                        u=runif(1
}
```

- Avaliando o histograma com os dados gerados:

```
require(ggplot2)
dados <- as.data.frame(amostra)
ggplot(dados, aes(x=amostra))+
  geom_histogram(aes(y=..density..
                 colour="black",
                 fill="white")
```



Método da Aceitação/Rejeição

Método da Aceitação/Rejeição

- Seja X uma v.a. contínua com densidade g e assuma que existe um método para simular de g.
- É possível usar este método como base para simular de uma outra v.a. contínua Y com densidade f.
- A idéia fundamental é inicialmente gerar X de g e então aceitar o valor gerado, com probabilidade proporcional a $f(X)/g(X)$, com sendo uma valor gerado de f.
- Este método é chamado de aceitação/rejeição (A-R).

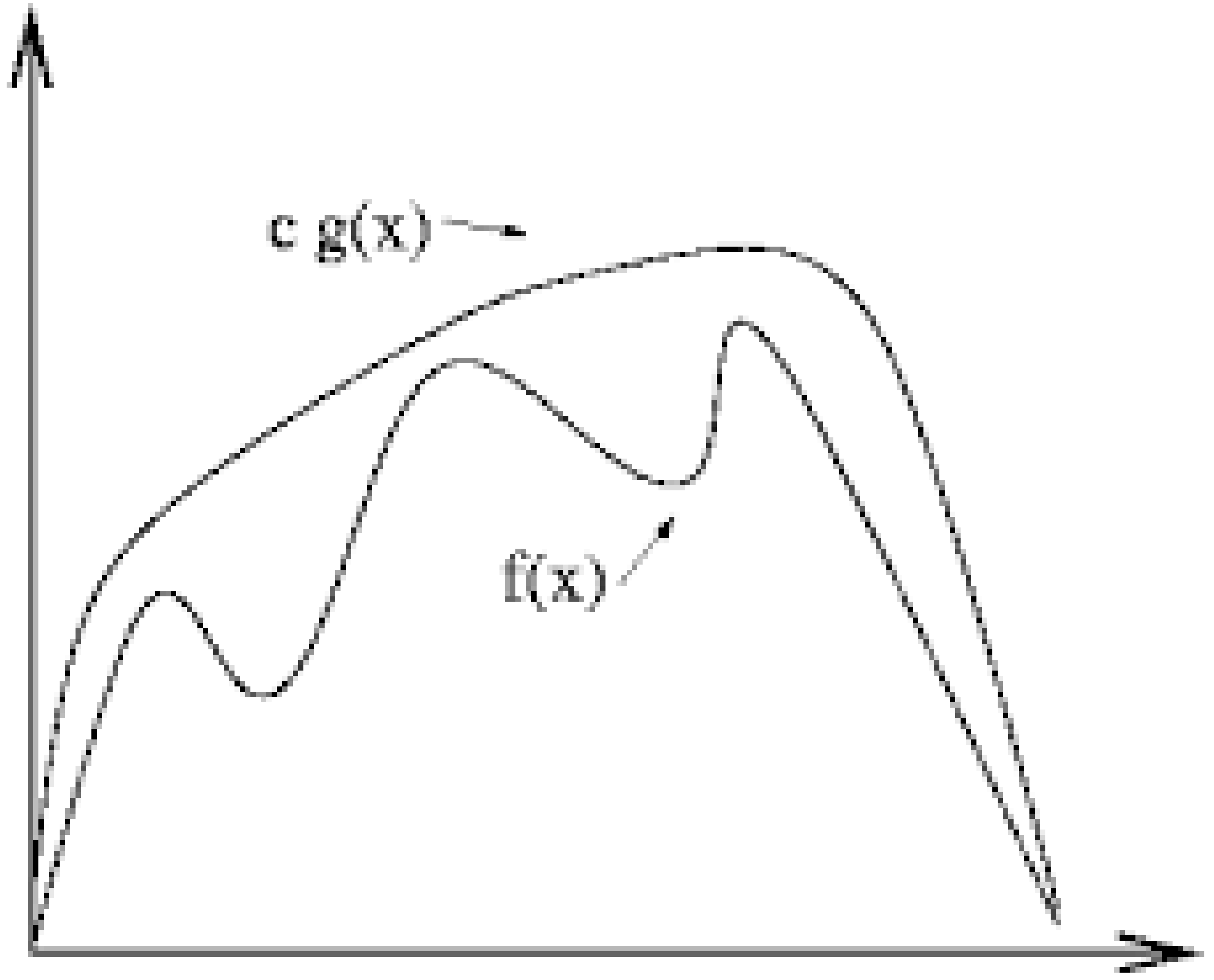
Método da Aceitação/Rejeição

Algorítmo genérico:

1. Simule $X \sim g$ e $U \sim U(0,1)$;
2. Faça $Y = X$ se $U \leq f(X)/cg(X)$. Caso contrário, volte ao passo anterior.

- A constante c é tal que $f(X) \leq cg(X)$ para todo x no suporte de f e g.
- A função cg é chamada de envelope.

Graficamente



Método da Aceitação/Rejeição

Proposições

1. O algoritmo acima produz uma v.a. com distribuição f.
2. Seja τ o número de iterações do algoritmo necessárias para produzir um valor Y com distribuição f. Então

$$\tau \sim Geo(1/c), \text{ com } E(\tau) = c.$$

Método da aceitação/rejeição

Exercício

Seja $Y \sim f(x) = 12x^2(1-x),\ 0 < x < 1$. Utilize o método da aceitação-rejeição para simular 1000 valores de f.

> Solução:

- Note que $X \sim U(0,1)$. Como esta variável pertence ao intervalo (0,1), podemos considerar o método com $g(x) = 1$, para $x \in (0,1)$, i.e, uma distribuição uniforme.
- Para determinar o menor valor de c tal que $f(x)/g(x) \leq c$, maximizamos

$$\frac{f(x)}{g(x)} = 12x^2(1-x)$$

- Temos que a derivada da expressão anterior e igualada a zero é $2x - 3x^2 = 0$. Temos que o valor de x que maximiza é $x = 2/3$.

Exemplo

Assim, $f(x)/g(x) \leq 12(2/3)^2(1/3) = 16/9 = c$. Logo, temos que

$$\frac{f(x)}{c\, g(x)} = \frac{9}{16} 12x^2(1-x) = \frac{27}{9} x^2(1-x).$$

Assim, para este exemplo o algoritmo fica:

1. Gere x e u com distribuição $U(0,1)$;
2. Se $u \leq \dfrac{27}{4} x^2(1-x)$, aceite o valor $Y = x$. Caso contrário, retorne o passo 1.

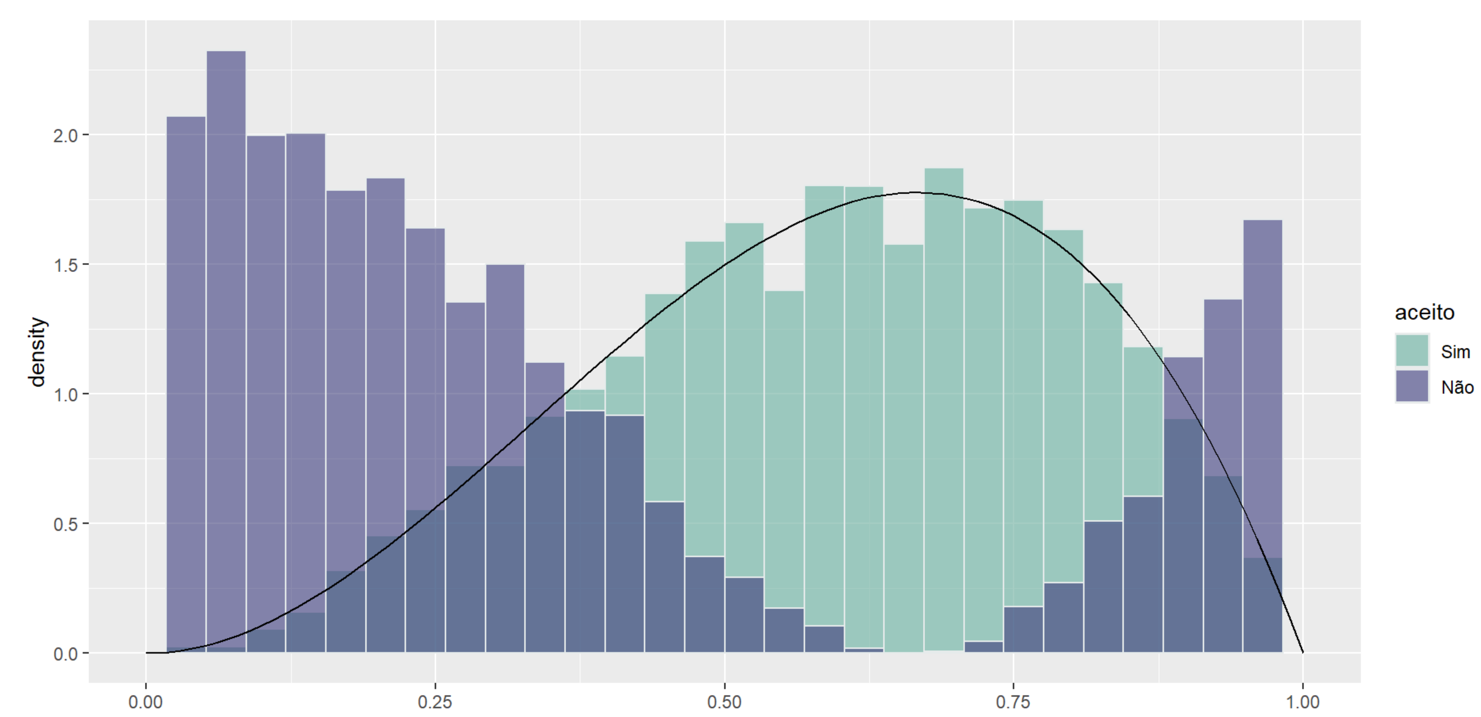
Exemplo

```r
#gerando os valores que podem ser
x = runif(10000,0,1)
aceito.val = c()

for(i in 1:length(x)){
  U = runif(1)
  if(U <= (27/4)*(x[i]^2)*(1-x[i])
    aceito.val[i] = 'Sim'
  }
  else{
    aceito.val[i] = 'Não'
  }
}
T = data.frame(x,
  aceito = factor(aceito.val
```

```
          x aceito
1 0.2094598    Não
2 0.9152412    Não
3 0.5431052    Sim
4 0.5613739    Sim
5 0.8326165    Não
6 0.5052088    Sim
```

```r
require(ggplot2)
require(dplyr)
require(hrbrthemes)

fun.1 <- function(x) 12*(x^2)*(1-x
```

```r
ggplot(T, aes(x=x, fill=aceito))+
  geom_histogram(aes(y = ..density
                 color="#e9ecef",
               alpha=0.6,
               position = 'identity')
  scale_fill_manual(values=c("#69b
                             "#404
  stat_function(fun = fun.1) +
  xlim(0,1)
```



Reamostragem ponderada

Reamostragem ponderada

- O algoritmo sampling importance resampling (SIR) simula realizações aproximadas de alguma distribuição alvo.
- Informalmente temos que:
  - Amostras são geradas de uma função de amostragem por importância g.
  - Cada ponto na amostra é ponderado para corrigir a probabilidade de amostragem de tal forma que a amostra ponderada seja relacionada da densidade alvo f.

Reamostragem ponderada

- Para a densidade alvo f, os pesos usados para corrigir as probabilidades de amostragem são chamados pesos de importância padronizados e são definidos por

$$\omega(x_i) = \frac{f(x_i)/g(x_i)}{\sum_{i=1}^{m} f(x_i)/g(x_i)},$$

para uma coleção de valores $x_1, \ldots, x_m$ gerados de g.

- Podemos ver este método como uma aproximação de f por uma distribuição discreta tendo massa $\omega(x_i)$ em cada ponto observado x_i para $i = 1, \ldots, m$.

Reamostragem ponderada

Algoritmo genérico:

1. Gere candidatos $Y_1, \ldots, Y_m$ iid de g.
2. Calcule os pesos de importância padronizados , $\omega(Y_1), \ldots, \omega(Y_m)$.
3. Reamostre, com reposição, $X_1, \ldots, X_n$ de $Y_1, \ldots, Y_m$ com probabilidades $\omega(Y_1), \ldots, \omega(Y_m)$.

A variável aleatória X amostrada com o algoritmo SIR tem distribuição que converge para f quando $m \to \infty$.

Considerações:

- $n/m \to 0$ para convergência em distribuição da amostra e, para n fixo, a convergência ocorre quando $m \to 1$.
- A tolerância máxima para a razão n/m depende da qualidade da densidade g. Algumas vezes $n/m \leq 1/10$ é suficiente para resultar em uma reamostra que não contenha muitas replicações de valores.
- O suporte de g deve incluir todo o suporte de f e g deve ter caudas mais pesadas que f, ou de forma mais geral, g ser escolhida garantindo que $f(x)/g(x)4 nunca cresça demais.
- Se $g(x)$ está próximo de zero em um local onde $f(x)$ é positivo, então um valor gerado desta região acontecerá muito raramente, mas quando isso acontecer receberá um enorme peso.

Exemplo

Sejam X e U variáveis aleatórias independentes, X com distribuição $N(0,1)$ e U com distribuição $Uniforme(0,1)$. Dizemos que $Y = X/U$ tem distribuição de Slash e sua função densidade de probabilidade é dada por

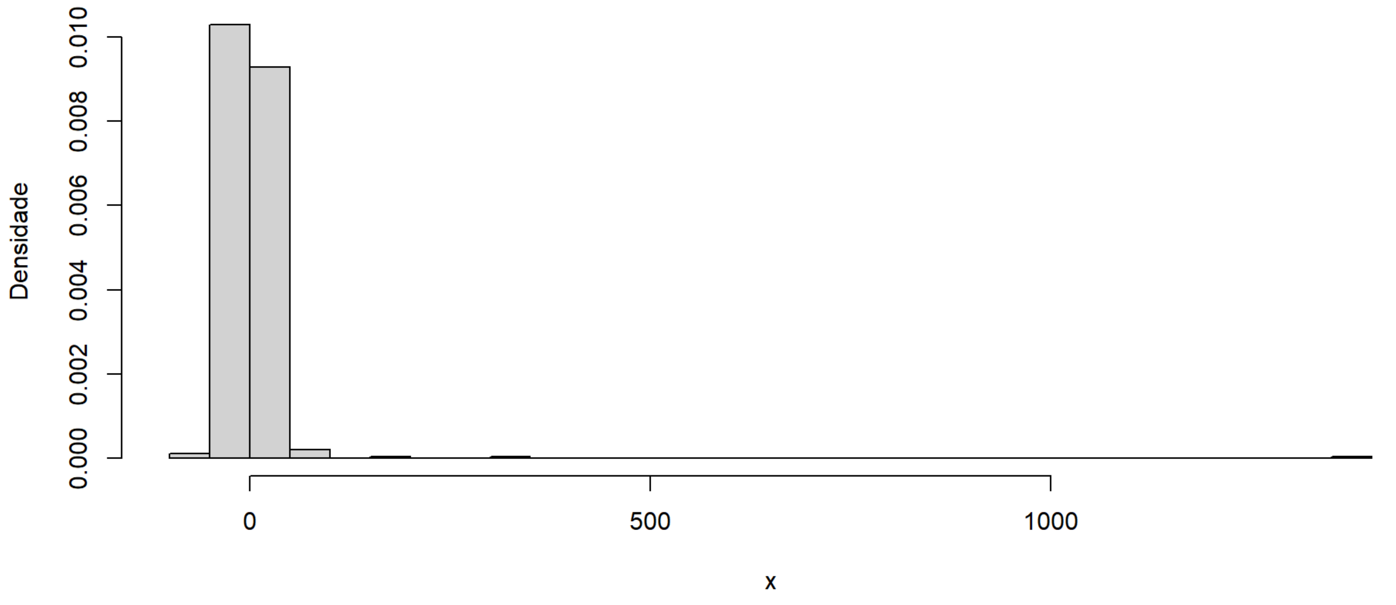
$$f(y) = \begin{cases} \dfrac{1-\exp{-y^2/2}}{y^2\sqrt{2\pi}}, & \text{se } y \neq 0 \\ \dfrac{1}{2\sqrt{2\pi}}, & \text{se } y = 0 \end{cases}$$

Exemplo

- Esta densidade tem caudas muito pesadas.
- Para entendermos melhor a vantagem do SIR, veja abaixo a geração de amostra da distribuição Slash diretamente por transformação. Note que são gerados alguns valores muito atípicos.

```
dslash <- function(y){
(y!=0)*(1-exp(-y^2/2))/(y^2*sqrt(2
    (y==0)*(1-exp(-y^2/2))/(y^2*sq
}

rslash <- function(n){
u <- runif(n)
x <- rnorm(n)
y <- x/u
return(y)
}
```

```
x <- rslash(500)
hist(x, ylab = "Densidade", xlab =
     main = "", breaks = 30,
     prob = TRUE)
```



Exemplo

- Agora ilustramos o uso do SIR, considerando:
  - Uso do SIR com proposta Slash para gerar da Normal padrão.
  - Uso do SIR com proposta Normal padrão para gerar da Slash.
- No R, temos:

```r
## Para gerar de uma normal padrã
Pesos1 <- function(y){
df <- dnorm(y); dg <- dslash(y)
p <- df/dg
pesos <- p/sum(p)
}
rnorm_sir <- function(n, m){
y <- rslash(m)
pesos <- Pesos1(y)
amostra <- sample(x = y, size = n,
                  replace = TRUE,
                  prob = pesos)
return(amostra)
}
```

```r
## Para gerar de uma slash com pro
Pesos2 <- function(y){
df <- dslash(y); dg <- dnorm(y)
p <- df/dg
pesos <- p/sum(p)
}
rslash_sir <- function(n, m){
y <- rnorm(m)
pesos <- Pesos2(y)
amostra <- sample(x = y, size = n,
                  replace = TRUE,
                  prob = pesos)
return(amostra)
}
```

Exemplo

```r
n <- 5000
m <- 100000
amostra1 <- rnorm_sir(n, m)
amostra2 <- rslash_sir(n, m)
```

```r
par(mar = c(4, 4, 3, 2), mfrow = c(1, 2))
hist(amostra1, prob = TRUE, main = "", xlab = "x",
col = "darkgoldenrod", breaks = 50, ylab = "Densidade" )
grid <- seq(-5, 5, by = 0.01)
densidade <- dnorm(grid)
lines(grid, densidade, col = 2, lty = 1, lwd = 2)
hist(amostra2, prob = TRUE, main = "", xlab = "x",
col = "aquamarine1", breaks = 50, xlim = c(-7,7), ylab = "Densidade")
grid <- seq(-7, 7, by = 0.01)
densidade <- dslash(grid)
lines(grid, densidade, col = 2, lty = 1, lwd = 2)
```