Univesidade Federal do Ceará-UFC
Centro de Ciências - CC
Departamento de Estatística e Matemática Aplicada

SIMULAÇÃO ESTOCÁSTICA USANDO O SOFTWARE LIVRE R

Francisco Hildemar Calixto de Alencar

Thiago Oliveira da Silva

Orientador: Prof. Juvêncio Santos Nobre

Fortaleza, janeiro de 2013

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A simulação pode ser de modo geral entendida como a imitação de um processo ou operação que são executados no mundo real. O trabalho de simulação funciona em âmbito geral na geração artificial de um sistema que possua características e elementos pertencentes a um mecanismo, processo ou operação do mundo real. Deste modo artificial, podemos fazer alterações no sistema para prever as consequências no mundo real. A simulação também pode ser utilizada para estudar mecanismos em estado de projeto. Assim sendo, a simulação é uma ferramenta para prever efeitos de mudanças em mecanismos existentes e tambem uma ferramenta para prever o funcionamento de novos mecanismos.

A simulação pode ser estática ou dinâmica. Na simulação estática, também chamada de simulação de Monte Carlo, a passagem do tempo é irrelevante.
O método de Monte Carlo (MMC) é um método estatístico utilizado em simulações estocásticas com diversas aplicações em áreas como a física, matemática e biologia. O mesmo tem sido utilizado há bastante tempo como forma de obter aproximações numéricas de funções complexas. Este método tipicamente envolve a geração de observações de alguma distribuição de probabilidades e o uso da amostra obtida para aproximar a função de interesse. As aplicações mais comuns são em computação numérica para avaliar integrais. A ideia do método é escrever a integral que se deseja calcular como um valor esperado. Na simulação dinâmica os resultados variam com a passagem do tempo, como por exemplo as industrias utilizam a simulaçao dinâmica em inúmeros casos que vão desde o poder de uma determinada energia nuclear, turbinas de vapor, modelagem de veículos, motores elétricos, modelos econométricos, sistemas biológicos, braços robóticos, a massa de amortecedores de mola, sistemas hidráulicos, e até o fluxo de uma droga em um organismo humano - estes são alguns casos básicos na qual esta tecnologia esta sendo empregada.

A simulação pode ser também determinística ou estocástica. Simulações determinísticas não possuem nenhuma variável aleatória como entrada do sistema. Simulações estocásticas possuem uma ou mais variáveis aleatórias como entrada do sistema e consequentemente obtemos variáveis aleatórias como saída.

A simulação tem como umas de suas principais finalidades:

- Realizar alterações nas informações, na organização e no ambiente do mecanismo para observar efeitos;

- Experimentar novos projetos ou novos procedimentos antes de implementá-los;
- Identificar as variáveis mais importantes de um mecanismo e como elas interagem através do estudo dos sinais de entrada e saída de resultados;
- Verificar soluções analíticas;
- Adquirir maior conhecimento sobre o modelo de simulação e sobre o processo de desenvolvimento do modelo para melhorias no sistema.

Para o desenvolvimento computacional escolhemos o software livre `R` por sua grande aplicabilidade e uso no meio acadêmico, utilizamos a versão 2.11.1 para Windows.

A seguir estão listadas algumas das áreas onde a simulação tem maior aplicação.

Sistemas de manufatura

- Sistemas de manipulação e movimentação de materiais;
- Operações de montagem;
- Planejamento da inter-operação entre sistemas de estoques;
- Manufatura ágil (sistema distribuído, sistemas inteligentes, sistemas autônomos).

Sistemas de saúde

- Custo e faturamento de produtos farmacêuticos;
- Otimização do atendimento em ambulatórios;
- Gerenciamento dos recursos hospitalares.

Sistemas envolvendo recursos naturais

- Gerenciamento de sistemas de coleta de lixo;
- Operação eficiente de plantas nucleares;
- Atividades de restauração do ambiente.

Sistemas de transporte

- Transferências de cargas;
- Operações de containers em portos;
- Postos de pedágio flexíveis de acordo com a demanda.

Sistemas de construção civil

- Processo de montagem de pontes suspensas;

- Novos paradigmas do processo construtivo;
- Interface para as ferramentas de projeto e construção.

Sistemas de restaurantes e entretenimento

- Análise do fluxo de clientes em fast-foods;
- Determinação do número ideal de funcionários de empresas de serviços;
- Atividades em parques temáticos.

Reengenharia e processo de negócios

- Integração de sistemas baseado no fluxo de tarefas;
- Análise de soluções.

Processamento de alimentos

- Operações no processamento de pescados;
- Avaliação da capacidade no processamento de cereais.

Sistemas computacionais

- Sistemas com arquitetura Cliente/Servidor;
- Redes heterogêneas.

Capítulo 2

Números Aleatórios

2.1 Geração de Números Pseudo-Aleatórios

Originalmente a geração de números aleatórios era feita de modo mecânica ou manual. Hoje a geração destes números é feita com o uso de computadores. Estes números apesar de serem gerados de forma determinística, tem a aparência de serem variáveis aleatórias uniformes (0,1) independentes.

A geração de números pseudo-aleatórios começa com a ordem x_0, chamado de semente, e para geração dos demais números temos:

$$x_n, \mathrm{n} \geqslant 1,$$

sendo

$$x_n = ax_{n-1} \bmod m, \tag{2.1}$$

em que a e m são inteiros positivos. O valor de x_n é dado pelo resto da divisão entre ax_{n-1} por m, e é chamado de número pseudo-aleatório, com x_n assumindo valores no intervalo $0, 1, ..., m-1$, este meio de gerar números aleatórios é chamado de método de congruência multiplicativo.

Assim, com x_n assumindo valores neste intervalo, depois de um número finito de valores gerados, um valor tem que se repetir e assim a sucessão inteira começa a repetir. Com isto a escolha das constantes a e m deve ser de forma que antes de ocorrer a repetição dos valores, tenha ocorrido um número "grande" de geração de variáveis.

***Exemplo*$_1$:** Gerar números aleatórios utilizando o software `R`, usando o método de congruência multiplicativo, tendo os seguintes valores: $x_0 = 4$, $a = 7$ e $m = 23$.

```
#Semente
x = c(4)

#Constantes
a = 7
m = 23

#Método de congruente multiplicativo
for (i in 2:6)
```

```
{
x[i] = (a*x[i-1])%%m
}

#Resposta do software
x
4  5 12 15 13 22
```

Neste exemplo, geramos um período muito curto, mas podemos perceber que o número aleatório gerado é o resto inteiro da divisão por m, isto é, x_n assume valores no intervalo $0, 1, \ldots, m-1$.

Para aproximarmos os números gerados a valores de uma variável aleatória uniforme (0,1), devemos dividi-los por m ou $m-1$.

***Exemplo*$_2$:** Gerar números aleatórios uniformes (0,1) utilizando o software `R`, usando o método de congruência multiplicativo, tendo os seguintes valores: $x_0 = 4$, $a = 7$ e $m = 23$.

```
#Semente
x = c(4)
x_unif = c()

#Constante
a = 7
m = 23

#Método de congruente multiplicativo
for (i in 2:6)
{
x[i] = (a*x[i-1])%%m
}

for(k in 1:6)
{
x_unif[k] = x[k]/m
}

#Resposta do software
x
4  5 12 15 13 22

x_unif # Valores arredondados com 4 casas decimais.
0.1739 0.2174 0.5217 0.6522 0.5652 0.9565
```

Neste exemplo geramos um período muito curto, e segundo (Campos, 1983) pode-se utilizando o teste não-paramétrico de Kolmogorov-Smirnov para evidenciar que os valores gerados têm distribuição uniforme (0,1). A seguir temos a sintaxe do software `R` para verificarmos a uniformidade dos valores gerados.

```
#Comando
ks.test(x_unif,punif(x_unif, 0.1739130,0.9565217))

#Resposta do software
        Two-sample Kolmogorov-Smirnov test

data:  x_unif and punif(x_unif, 0.173913, 0.9565217)
D = 0.3333, p-value = 0.9307
alternative hypothesis: two-sided
```

Tendo como hipótese nula, os dados tem distribuição uniforme (0,1). Logo, pelo teste, temos evidências que os dados têm distribuição uniforme (0,1). Ao nível de 5% de significância, por exemplo.

***Exemplo*$_3$:** (Ross, Sheldon M; Simulation 4.ed.) Com $x_1 = 23$, $x_2 = 66$ e $x_n = 3x_{n-1} + 5x_{n-2} \bmod 100$, $n \geq 3$. Encontrar utilizando o software livre R os primeiros 14 valores para $u_n = x_n/100; n \geq 1$.

```
#Semente
x = c(23,66)
x_unif = c()

#Constantes
a = c(3,5)
m = 100

#Método de congruente multiplicativo
for (i in 3:14)
{
x[i] = (a[1]*x[i-1]+ a[2]*x[i-2])%%m
}

for(k in 1:14)
{
x_unif[k] = x[k]/m
}

#Resposta do software
x
23 66 13 69 72 61 43 34 17 21 48 49 87  6

x_unif
0.23 0.66 0.13 0.69 0.72 0.61 0.43 0.34 0.17 0.21 0.48 0.49 0.87 0.06
```

A seguir temos a sintaxe do software R para verificar-se a uniformidade dos valores gerados, por meio do teste não paramétrico Kolmogorov-Smirnov.

```
#Comando
ks.test(x_unif,punif(x_unif,0.06,0.87))

#Resposta do software
     Two-sample Kolmogorov-Smirnov test

data:  x_unif and punif(x_unif, 0.06, 0.87)
D = 0.2143, p-value = 0.9205
alternative hypothesis: two-sided
```

Tendo como hipótese nula, os dados tem distribuição uniforme (0,1). Logo, pelo teste, temos evidências que os dados têm distribuição uniforme (0,1). Ao nível de 5% de significância, por exemplo.

A escolha de a e m deve satisfazer três critérios:

1. Para qualquer semente, o resultado da sequência tem a aparência de uma sucessão de variáveis aleatórias uniformes (0,1) independentes.

2. Para qualquer semente, o número de variáveis que podem ser geradas antes da repetição é "grande".

3. Os valores devem ser calculados de forma eficaz em um computador.

Outro critério que satisfaz os anteriores, é que m seja um número primo "grande". Computacionalmente falando, para computadores que trabalham com 32bits, temos os valores

$$m = 2^{31} - 1 \text{ e } a = 7^5 = 16.807.$$

Outro gerador de números pseudo-aleatório consiste em usar

$$x_n = (ax_{n-1} + c) \bmod m. \tag{2.2}$$

Estes geradores são chamados de gerador misto côngruo, isto é, possui tanto uma multiplicação por a como uma adicão de c. Neste gerador a escolha do valor de m deve ser de modo que possamos fazer a divisão de $ax_{n-1} + c$ por m de forma computacionalmente eficiente.

2.2 Usando Números Aleatórios para Avaliar Integrais

Vamos supor que desejamos calcular o valor de θ, em que

$$\theta = \int_0^1 g(\mathrm{x})\, dx,$$

em que $g(x)$ é uma função e $g(x) : [0, 1] \to \mathbb{R}$. Se U é uma variável aleatória com distribuição uniforme (0,1), então podemos rescrever θ como sendo

$$\theta = \mathbb{E}[g(\mathrm{U})].$$

Se $U_1, U_2, ..., U_k$ são variáveis aleatórias iid com distribuição uniforme (0,1), então $g(U_1), g(U_2), ..., g(U_k)$ são variáveis aleatórias independentes e identicamente distribuídas com média θ. Pela lei forte dos grandes números, temos que

$$\sum_{i=1}^{k} \frac{g(U_i)}{k} \xrightarrow{\mathbb{P}} \mathbb{E}[g(U)] = \theta.$$

Assim sendo, gerando uma grande quantidade de números aleatórios u_i, podemos aproximar o valor θ. Esta aproximação de integrais é chamada de aproximação de Monte Carlo.

Agora se o interesse é obter

$$\theta = \int_a^b g(\mathrm{x})dx,$$

utilizaremos a seguinte transformação

$$y = \frac{(x-a)}{(b-a)}; dx = dy(b-a),$$

de forma que (2.3) fica dada por

$$\theta = \int_0^1 g[y(b-a)+a].(b-a)dy = \int_0^1 h(y)\ dy$$

Se tivermos

$$\theta = \int_0^\infty g(x)dx,$$

poderíamos fazer

$$y = \frac{1}{x+1};\ dy = \frac{-dx}{(x+1)^2}$$

assim

$$\theta = \int_0^1 h(y).$$

Para o caso multidimensional temos

$$\theta = \int_0^1 \int_0^1 \ldots g(x_1, ..., x_n)dx_1, \ldots, dx_n$$

Utilizando a aproximação de Monte Carlo para aproximar o valor de θ temos

$$\theta = \mathbb{E}[g(U_1, ..., U_n)]$$

em que $U_1, ..., U_n$ são variáveis aleatórias iid com distribuição uniforme (0,1). Se gerarmos k variáveis independentes da variável U uniforme (0,1) n-dimensional.

$$U_1^1, ..., U_n^1$$
$$U_1^2, ..., U_n^2$$
$$\vdots$$
$$U_1^k, ..., U_n^k.$$

Assim $g(U_1^i, ..., U_n^i)$, $i = 1, 2, ...k$, sendo independentes e identicamente distribuídas. Podemos agora estimar θ através do estimador consistente

$$\frac{\sum_{i=1}^{k} g(U_1^i, ..., U_n^i)}{k}.$$

Como aplicação da teoria descrita anteriormente segue exemplo da estimação de π via simulação.

***Exemplo* :** Suponha que se tenha um vetor aleatório (X,Y), e o mesmo é distribuído uniformemente dentro de um quadrado de área 4 e centrado na origem. Isto é, é um ponto aleatório especificado na figura 2.1. Considere-se a probabilidade deste ponto aleatório no quadrado esteja contido dentro de um círculo inscrito de raio 1, conforme mostra a figura 2.2. Note agora que (X,Y) é distribuído unformemente dentro de um quadrado, que se segue.

$$P\{(X,Y) \text{ está no círculo}\} = P\{X^2 + Y^2 \leq 1\} = \frac{\text{Área do Círculo}}{\text{Área do Quadrado}}$$

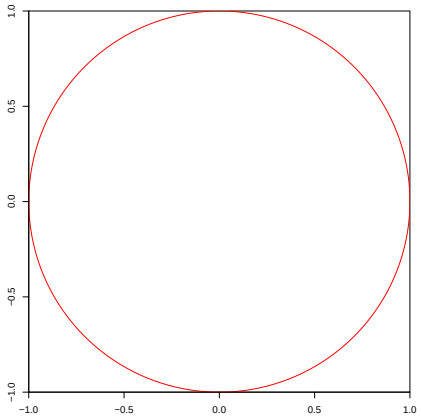

Figura 2.1: Círculo de raio 1 centrado na origem inscrito no quadrado de área 4 centrada na origem

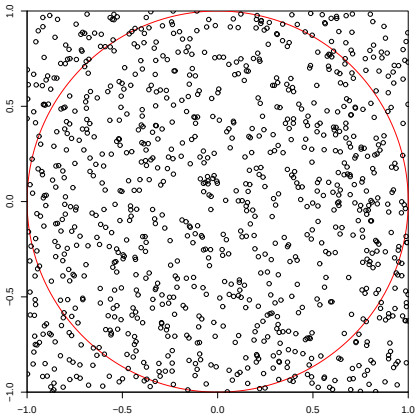

Figura 2.2: Vetor aleatório (X,Y), distribuído uniformemente dentro de um quadrado de área 4 centrado na origem

Com isso se gerarmos um grande número de pontos, a proporção de números que caem dentro do círculo será de aproximadamente $\frac{\pi}{4}$.

Conforme valores obtidos através do software R, pode-se construir a seguinte tabela, aonde o erro de estimação é dado por $|I_1 - \widehat{I}_1|$:

Tabela 2.1: Estimativa de π

n	Estimativa	Erro de Estimação
500	3,192	0,0507
1000	3,132	0,0512
10000	3,138	0,0004
100000	3,14228	$<$ 0,0001
1000000	3,140884	0,0001

```
par(mfrow=c(1,2))
r  =  1
par(pty="s",fig=c(-1,1,-1,1))
emptyplot(c(0, 0),frame.plot=r)
plotcircle(r=r, mid=c(0,0), lwd=1,lcol="red")
axis(1)
axis(2)
x = runif(1000,-1,1)
y = runif(1000,-1,1)
points(x,y)

est.pi <- function(n){
int =  0
ext = 0
pi.est =  0
x = runif(n,-1,1)
y =  runif(n,-1,1)
z = (x^2) + (y^2)
for (i in 1:n) {
if (z[i] > 1) {
ext =  ext + 1}
else {int  =  int + 1}
if (i==n) {pi.est = (int/(int+ext))*4}
}
 return(pi.est)
}
```

2.3 Exemplos de Estimação de Integrais Definidas Utilizando o Método de Monte Carlo

Para os exemplos que seguem será feito uso do software livre R. No apêndice A apresenta-se a sintaxe básica de comandos do R que serão utilizados em cada exem-

plo.

***Exemplo*$_1$:** $I_1 = \int_0^1 (1-x^2)^{3/2}dx = \mathbb{E}[(1-\mathrm{X}^2)^{3/2}]$, em que $X \sim U(0,1)$

Usando o método de Monte Carlo tem-se que a estimativa de I_1 é dada por:

$$\widehat{I_1} = \frac{\sum_{i=1}^{n}(1-x^2)^{3/2}}{n}$$

Utilizando o software `R` temos que o valor da integral é I_1 é $0,5890$.

Conforme valores obtidos através do software `R`, pode-se construir a seguinte tabela:

Tabela 2.2: Estimativa da Integral I_1, via método de Monte Carlo

n	Estimativa	Variância	Erro de Estimação
500	0,5887	0,1101	0,0002
1000	0,5894	0,1100	0,0003
10000	0,5894	0,1100	0,0004
100000	0,5888	0,1101	$<$ 0,0001
1000000	0,5889	0,1101	$<$ 0,0001

Neste exemplo, como era esperado, o método de Monte Carlo se mostra eficiente, nota-se isto quando comparamos os valores estimados, para cada valor de n com o valor calculado 0,5890, e consequentemente o erro de estimação vai decrescendo conforme n cresce.

***Exemplo*$_2$:** $I_2 = \int_0^\infty \exp\{(-x)\}dx$, fazendo a substituição $y = \frac{1}{x+1}$; $dx = -y^{-2}dy$, assim temos:

$$I_2 = \int_0^1 \exp(1-1/y).y^{-2}dy = \mathbb{E}[\exp\{(1-1/Y)\}.Y^{-2}], \text{em que } Y \sim U(0,1).$$

Usando o método de Monte Carlo tem-se que a estimativa de I_2 é dada por:

$$\widehat{I_2} = \frac{\sum_{i=1}^{n}\exp\{(1-1/y)\}.y^{-2}}{n}$$

Tem-se que $\exp\{(-x)\}$ é a f.d.p de uma variável aleatória com distribuição exponencial com parâmetro igual a 1.

De acordo com a saída do `R`, podemos construir a seguinte tabela:

Baseado nos resultados apresentados pela tabela acima, temos conclusões iguais as do exemplo anterior, ou seja, o método de aproximação de Monte Carlo é realmente eficiente. Quando comparamos os valores estimados da integral I_2 com o seu valor calculado, notamos que a diferença é mínima, isto é confirmado quando observamos os erros de estimação conforme os valores de n aumentam.

Tabela 2.3: Estimativa da Integral I_2, via método de Monte Carlo

n	Estimativa	Variância	Erro de Estimação
500	1,0001	0,2499	0,0001
1000	0,9994	0,2500	0,0005
10000	0,9998	0,2501	0,0001
100000	1	0,2500	0
1000000	1,0003	0,2495	0,0003

***Exemplo*$_3$:** $I_3 = \int_0^\infty \int_0^x \exp -\{(x+y)\} dydx$, calculando o valor desta integral obtemos $1/2$. Trabalharemos nesta integral para podemos utilizar o método de aproximação de Monte Carlo.
Fazendo $y = x.u_1; dy = x.du_1$

$$\int_0^\infty \int_0^1 \exp -\{x(1+u_1)\} x du_1 dx,$$

fazendo $x = \frac{1-u_2}{u_2}$; $dx = -u_2^{-2} du_2$

$$\int_0^1 \int_0^1 \exp -\{(1/u_2 - 1)(1+u_1)\}(u_2^{-3} - u_2^{-2}) du_1 du_2.$$

Em que $U_1, U_2 \sim U(0,1)$

Usando o método de Monte Carlo tem-se que a estimativa de I_3 é dada por:

$$\widehat{I_3} = \frac{\sum_{i=1}^n \exp -\{(1/u_2 - 1)(1+u_1)\}(u_2^{-3} - u_2^{-2})}{n}$$

Utilizando o software `R` podemos construir o seguinte quadro. Neste caso, os

Tabela 2.4: Estimativa da Integral I_3, via método de Monte Carlo

n	Estimativa	Variância	Erro de Estimação
500	0,4999	0,2383	0
1000	0,4998	0,2380	0,0001
10000	0,4998	0,2377	0,0001
100000	0,5004	0,2384	0,0004
1000000	0,4995	0,2380	0,0004

valores estimados são comparados com o valor calculado que é igual a $0,5$. Aqui, não diferente dos outros exemplos, verifica-se mais uma vez que o método de aproximação de Monte Carlo é eficiente. Percebemos isto, verificando a proximidade dos valores estimados em relação ao valor calculado.

***Exemplo*$_4$:** Calcular a probabilidade do valor de X de uma distribuição normal com média μ e desvio padrão σ esta no intervalo (a,b).

$$I_4 = \int_a^b \frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left\{-(x-\mu)^2/2\sigma^2\right\} dx.$$

Fazendo a transformação:

$$y = \frac{(x-a)}{(b-a)}$$

Obteremos:

$$dx = (b-a)dy$$

e assim:

$$\int_0^1 \frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left\{-[y(b-a)-\mu]^2/2\sigma^2\right\} \times (b-a)dy.$$

Escolhendo os valores de μ, σ, a e b. Assim utilizando o Método de Monte Carlo temos que a estimativa da integral é dada por:

$$\widehat{I_4} = \frac{\sum_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \times \exp\left\{-[y(b-a)-\mu]^2/2\sigma^2\right\} \times (b-a)}{n}$$

Escolhendo $\mu = 0$ e $\sigma = 1$, temos:

$$\int_0^1 \frac{1}{\sqrt{2\pi}} \times \exp\left\{-[y(b-a)]^2/2\right\} \times (b-a)dy$$

e

$$\widehat{I_4} = \frac{\sum_{i=1}^n \frac{1}{\sqrt{2\pi}} \times \exp\left\{-[y(b-a)^2/2\right\} \times (b-a)}{n}.$$

A distribuição acima é chamada de normal padrão. Para valores de $a = 0$, $b = 1$ e $n = 500$, obtem-se fazendo uso do software `R` o valor $\widehat{I_4} = 0.34135$. E o valor tabelado para a e b é $0,34134$. Assim, observamos que a aproximação é bastante satisfatória.

Capítulo 3

Geração de Variáveis Aleatórias

3.1 Método da Transformação Inversa para Variável Aleatória Discreta

Para gerar uma variável aleatória discreta X com função de probabilidade dada por :

$$P\{X = x_j\} = p_j \ \mathbb{1}(j)_{\{0,1,\ldots\}};\ \sum_{j=0}^{\infty} p_j = 1$$

Vamos gerar um número aleatório U$\sim U(0,1)$ e definimos:

$$X = \begin{cases} x_0, & se \quad U < p_0 \\ x_1, & se \quad p_0 \leq U < p_0 + p_1 \\ \vdots & \\ x_j, & se \quad \sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^{j} p_i \\ \vdots & \\ x_{j+k}, & se \quad \sum_{i=0}^{j+k-1} p_i \leq U < \sum_{i=0}^{j+k} p_i \end{cases}$$

Uma vez que, para $0 < a < b < 1$, $P\{a \leq U < b\} = b - a$, temos que:

$$P\{X = x_j\} = P\left\{\sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^{j} p_i\right\} = p_j.$$

E assim X tem a distribuição desejada.

3.1.1 Observações

Função de Distribuição

Seja x_i ordenado da forma $x_{(0)} < x_{(1)} < x_{(2)} < \ldots < x_{(j-1)}$, e seja F a função de distribuição de X tal que: $F(x_k) = \sum_{i=0}^{k} p_i$, assim: $X = x_j$ se $F(x_{j-1}) \leq U < F(x_j)$. Assim encontramos o valor de X achando o intervalo $[F(x_{j-1}), F(x_j)]$ (ou, equivalentemente, achando o inverso de F(U)). É por isto que chamamos de método da transfomação inversa.

Algoritmo

A obtenção dos valores de X pode ser descrita como no algoritmo abaixo:

1. Geração do número aleatório $U \sim U(0,1)$;
2. Se $U < p_0$, então $X = x_0$;
3. Se $p_0 < U < p_0 + p_1$, então $X = x_1$;
4. Se $p_0 + p_1 < U < p_0 + p_1 + p_2$, então $X = x_2$;
   $\vdots$

3.1.2 Exemplos da Geração de Variáveis Aleatórias Discretas

1. Para simular uma variável aleatória X tal que:

$$p_1 = 0,20; p_2 = 0,15; p_3 = 0,25; p_4 = 0,40 \text{ em que } P\{X = x_j\} = p_j.$$

Faremos uso do algoritmo descrito anteriormente, e assim teremos:

$$\text{Se } U < 0,20 \ , \ X = 1$$
$$\text{Se } 0,20 < U < 0,35 \ , \ X = 2$$
$$\text{Se } 0,35 < U < 0,60 \ , \ X = 3$$
$$\text{Caso contrário } X = 4$$

Abaixo temos o algoritmo descrito na sintaxe do software `R` para obtermos os valores da variável discreta X, para determinar valores gerados da variável $U \sim U(0,1)$.

```
P1 = 0.20;P2 = 0.15;P3 = 0.25;P4 = 0.40
X = 0
set.seed(3)
U = runif(10000)
for(i in 1:length(U))
{
if (U[i] < P1) X[i] = 1
if (P1< U[i] & U[i] < P1 + P2) X[i] = 2
if (P1 + P2 < U[i] & U[i] < P1 + P2 + P3)  X[i] =  3
if (U[i] > P1 + P2 + P3) X[i] =4
}
```

A programação acima esta gerando 10.000 valores de uma variável $U \sim U(0,1)$, e para cada valor gerado, verifica as condições do algoritmo descrito, e assim, atribui os valores estabelecidos $(1, 2, 3, 4)$ à variável X. A distribuição de X para os valores aleatórios da variável U, esta descrita na Figura 3.1.

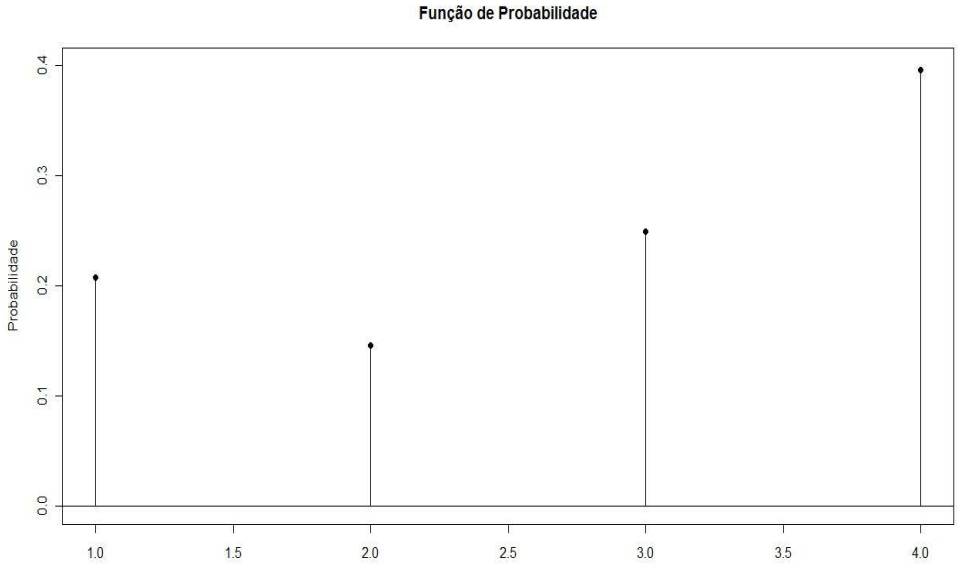

Figura 3.1: Distribuição da variável aleatória discreta X para valores gerados de U.

Também podemos utilizarmos o algoritmo descrito da seguinte forma:

$$\text{Se } U < 0,40 \ , \ X = 4$$

$$\text{Se } 0,40 < U < 0,65 \ , \ X = 3$$

$$\text{Se } 0,65 < U < 0,85 \ , \ X = 1$$

$$\text{Caso contrário } X = 2$$

Abaixo temos o algoritmo descrito na sintaxe do software R.

```
P1 = 0.20;P2 = 0.15;P3 = 0.25;P4 = 0.40
B = 0
set.seed(3)
U = runif(10000)
for(i in 1:lenght(U))
{
if (U[i] < P4) B[i] = 4
if (P4< U[i] & U[i] < P4 + P3) B[i] = 3
if (P4 + P3 < U[i] & U[i] < P4 + P3 + P1)  B[i] =  1
if (U[i] > P4 + P3 + P1) B[i] =2
}
```

Podemos verificar pela Figura 3.2 que as modificações no algoritmo inicial praticamente não resultam em alterações na distribuição de X.

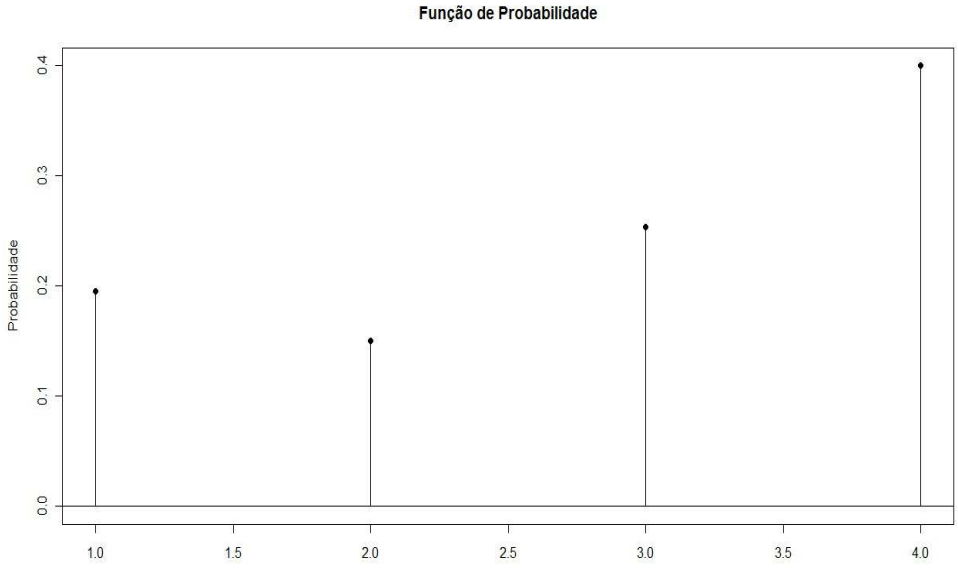

Figura 3.2: Distribuição da variável aleatória discreta X para valores gerados de U.

2. Para gerar valores de X, tal que X seja uma variável aleatória Uniforme discreta, isto é, $\mathrm{P}\{X = x_j\} = 1/n$, $\mathbb{1}(x_j)_{\{1,\ldots,n\}}$. Fazemos:

$$X = x_j, \text{ se } \frac{x_j - 1}{n} \leq U < \frac{x_j}{n}$$

ou

$$X = x_j, \text{ se } x_j - 1 \leq nU < x_j$$

Deste modo, temos que:

$$X = \lceil nU \rceil + 1$$

Em que $\mathrm{Int}(nU)$ é o maior número inteiro menor ou igual a nU.

Abaixo temos a sintaxe do software livre `R` para gerarmos diversos tamanhos de amostras de uma variável aleatória Uniforme discreta, conforme a teoria apresentada, e a também uma comparação entre os tamanhos de amostras geradas, através de seus histogramas (Figura 3.3).

```
n = # tamanho da amostra
set.seed(5)
U = runif(n)
D = 0
for(i in 1:n)
{
D[i] = round(n*U[i]) + 1
}
```

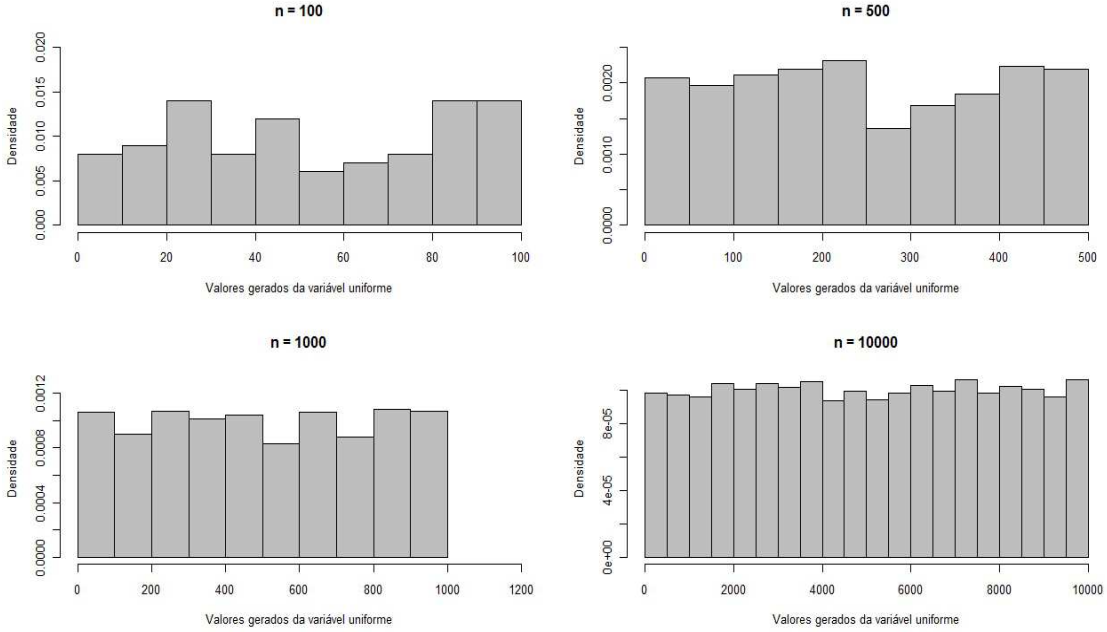

Figura 3.3: Histogramas das amostras para vários valores de n

3. Para gerar valores de uma variável aleatória X, definida como o número de repetições necessárias para obter a primeira ocorrência de sucesso em um experimento, nele se incluíndo esta última, estamos definindo X como uma variável aleatória com distribuição geométrica tendo função de densidade dada por:

$$P\{X = x\} = (1-p)^{x-1}p \;\; \mathbb{1}(x)_{\{1,2,\ldots\}} \;\; \text{em que}$$

p é a probabilidade de ocorrêcia do sucesso.
Temos que

$$\sum_{x=1}^{j-1} P\{X = x\} = 1 - P\{X > j-1\} = 1 - (1-p)^{j-1}, \quad j \geq 1$$

Nós podemos obter o valores de X gerando valores de uma variável uniforme U e fixando X igual ao valor j. Pelo método da transformação inversa temos:

$$1 - (1-p)^{j-1} \leq U < 1 - (1-p)^{j}$$

$$(1-p)^{j} < 1 - U \leq (1-p)^{j-1}$$

Assim podemos definir X como

$$X = \text{Min}\{j : (1-p)^{j} < 1 - U\}.$$

Das propriedades da função logarítmica temos que se a $<$ b então $\ln(a) < \ln(b)$, logo podemos esvrever

$$X = \text{Min}\{j : j.\ln(q) < \ln(1-U)\}$$

$$X = \text{Min}\left\{j : j > \frac{\ln(1-U)}{\ln(1-p)}\right\}.$$

Usando a notação Int() podemos expressar

$$X = \left\lceil \frac{\ln(1-U)}{\ln(1-p)} \right\rceil + 1.$$

Como $1 - U$ é Uniformemente distribuida em (0,1), temos que

$$X = \left\lceil \frac{\ln(U)}{\ln(1-p)} \right\rceil + 1.$$

Abaixo temos a sintaxe do software livre `R` para gerarmos diversos tamanhos de amostras de uma variável aleatória geométrica, conforme a teoria apresentada, e a também uma comparação entre os tamanhos de amostras geradas, através da Figura 3.4 .

```
# valor de p fixo 0,5
n = # tamanho da amostra
set.seed(5)
U = runif(n)
G = 0
for(i in 1:n)
{
G[i] = round(log(U[i])/log(0.5)) + 1
}
```

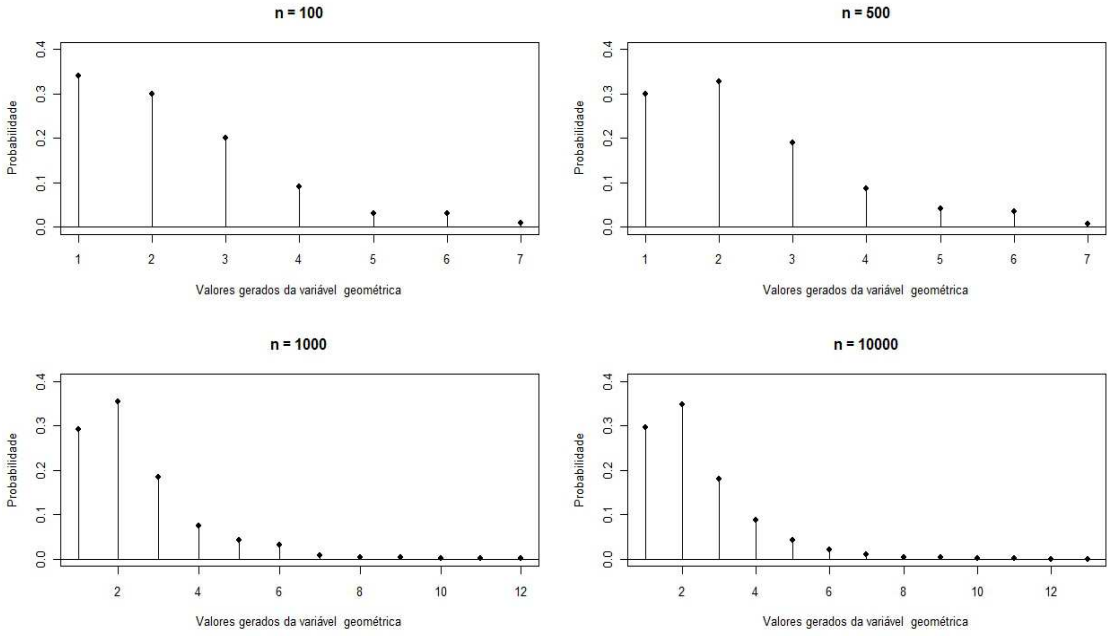

Figura 3.4: Distribuição da variável aleatória geométrica para valores gerados de U, com $p = 0,5$

4. Seja X uma variável aleatória discreta, com distribuição de probabilidade dada por:

$$P(X = x) = \frac{e^{-\lambda}.\lambda^x}{x!} \quad \mathbb{1}(x)_{\{0,1,2,\ldots\}}$$

diremos que X tem distribuição de Poisson, com parâmetro $\lambda > 0$.
A distribuição de Poisson está intimamente relacionada com a distribuição exponencial e é usada em muitos problemas de simulação que envolvem chegadas e partidas.
Para gerarmos valores de uma variável discreta com distribuição de Poisson, utilizando o método da transformação inversa, usamos a seguinte identidade:

$$p_{x+1} = \frac{\lambda}{x+1} p_x, \quad x \geq 0$$

e o seguinte algoritmo para gerarmos os valores da variável.

Algoritmo

1. Geração do número aleatório $U \sim U(0, 1)$;

2. $x = 0$, $p_x = \exp^{-\lambda}$, $F(x) = p_x$;

3. Se $U < F(x)$, então $X = x$ e para;

4. $p_{x+1} = \frac{\lambda}{x+1}.p_x$, $F(x+1) = F(x) + p_{x+1}$;

5. Volta para o terceiro passo.

Se $U < F(x)$ não ocorrer passamos para o quarto passo e calculamos p_1 usando o método da transformação inversa. E depois verifica-se $U < F(1)$(novo valor de $F(x)$) e definimos $x = 1$, assim o algoritmo continua.O algoritmo acima sucessivamente verifica se o valor de Poisson é 0, então se é 1, então 2, e assim por diante.
Segue a sintaxe do software livre `R` para gerarmos diversos tamanhos de amostras de uma variável aleatória Poisson e a Figura 3.6 com valores gerados de uma variável Poisson para $\lambda = 2$.

```
u = runif(n)
lambda = # valor de Lambda
exp(-lambda)
px = 1
k = c() # vetor que receberá os valores da variável poisson.
n = 0 # valor inicial da variável poisson.
for(i in 1:length(u))
{
if(px*u[i] < exp(-lambda)) k[i] = n else ((n = n + 1) && (px = px*u[i]))
}
```

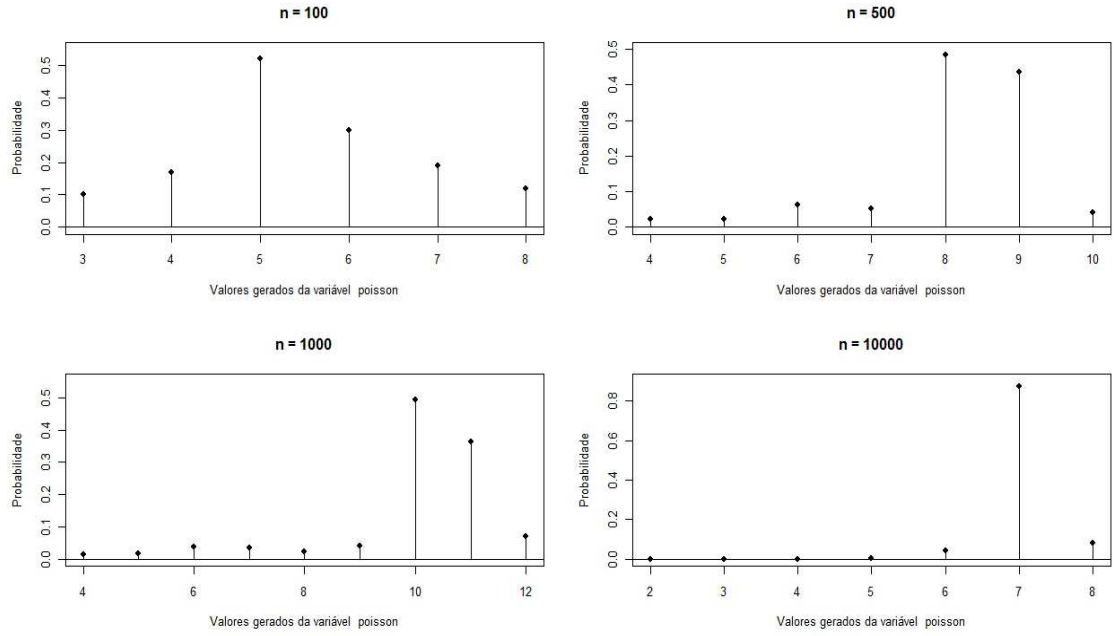

Figura 3.5: Distribuição de uma variável aleatória poisson para valores gerados de U, com $\lambda = 2$

5. Seja X uma variável aleatória discreta, tomando valores dentro do interlavo $0, 1, 2, \cdots, n$. E com distribuição de probabilidade dada por:

$$P(X = x) = \frac{n!}{x!.(n-x)!} p^x (1-p)^{n-x} \quad \mathbb{1}(x)_{\{0,1,2,\ldots,n\}}.$$

diremos que X tem distribuição Binomial, com parâmetros (n, p).

Para gerarmos valores de uma variável aleatória Binomial utilizando o método da transformação inversa, usaremos a seguinte identidade e algoritmo.

$$P(X = x+1) = \frac{n-1}{x+1} \frac{p}{1-p} P(X = x)$$

Algoritmo

1. Geração do número aleatório $U \sim U(0, 1)$;

2. $c = p/(1-p), i = 0, pr = (1-p)^n, F(x) = pr$;

3. Se $U < F(x)$, então $X = x$ e para;

4. $pr = [c.(n-x)/(x+1)].pr^*$, $F(x+1) = F(x) + pr, x = x+1$, em que pr^* é pr do passo anterior;

5. Ir para o passo 3.

Observação

Outra forma de gerar uma variável $X \sim$ Benomial (n, p), é utilizar sua interpretação como o número de sucessos em n tentativas independentes de uma variável Bernoulli , onde a probabilidade de sucesso é p.

Para gerar $Y \sim$ Bernoulli(θ), $\theta \in (0, 1)$, fazemos:

Algoritmo

1. Gerar $U \sim U(0, 1)$;

2. Faça

$$Y = \begin{cases} 0, & se \quad u_i > \theta \\ 1, & se \quad u_i \leq \theta \end{cases}$$

Segue a sintaxe do software livre `R` para gerarmos amostras de uma variável aleatória Bernoulli.

```
set.seed(1)
U = runif(n)
b = c()
teta = # valor de teta
for(i in 1 : length(U))
{
if (U[i] > teta) b[i] = 0 else (b[i] = 1)
}
```

Para gerar $X \sim$ Benomial (n, θ), em que $X = \sum_{i=1}^{m} Y_i$, e Y_i é iid Bernoulli(θ).

Repetimos n vezes o experimento da Bernoulli e fazemos $x_i = \sum_{i=1}^{m} y_i$.

Segue sintaxe do software `R`.

```
b = c() # guarda valores da bernoulli
bn = c() # guarda valores da binomial
teta = # valor de teta
for(i in 1 : 20)
{
U = runif(20)
for (a in 1 : 20)
{
if (U[a] > teta) b[a] = 0 else (b[a] = 1)
}
bn[i]= sum(b)
}
```

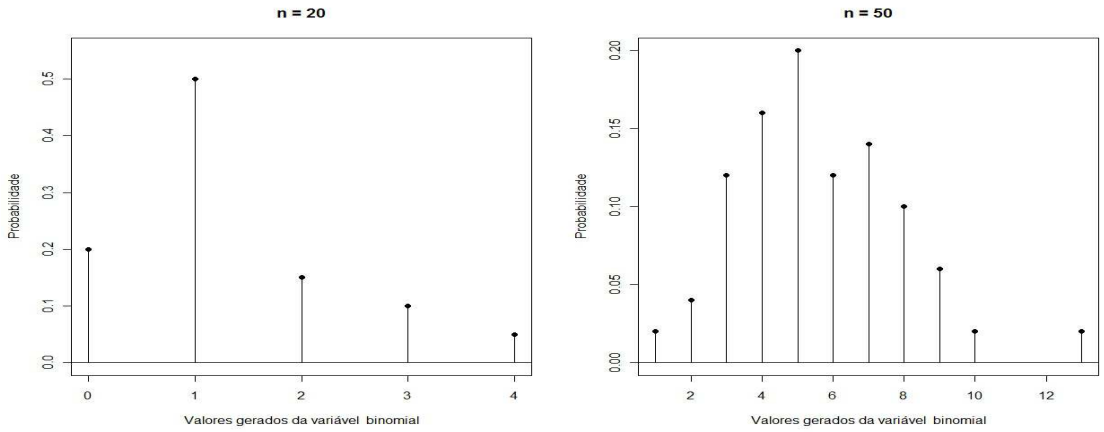

Figura 3.6: Distribuição de uma variável aleatória binomial para valores gerados de U, com $\theta = 0,2$

6. Se X é uma variável aleatória discreta com distribuição de probabilidade dada por:

$$P(X = x) = \frac{\binom{A}{x}\binom{N-A}{n-x}}{\binom{N}{n}} \mathbb{1}(x)_{\{max(0,n-N+A),\ldots,min(n,A)\}}.$$

Dizemos que X é modelada pela distribuição hipergeométrica, que descreve a probabilidade de se retirar x elementos do tipo A numa sequência de n extrações de uma população finita de tamanho N e $A - N$ elementos do tipo B, sem reposição.

Deste modo a função de distribuição acumulada X e da forma:

$$F_X(x) = P(X \leq x) = \begin{cases} 0, & \text{se } x < 0 \\ \sum_{j=0}^{x} \frac{\binom{A}{j}\binom{N-A}{n-j}}{\binom{N}{n}}, & \text{se } 0 \leq x < n \\ 1, & \text{se } x \geq n \end{cases}$$

Para simularmos valores de X, utilizando o método da transfomação inversa, e de modo semelhante a simulação de valores de uma variável poisson, fazemos uso da seguinte identidade:

$$p_x = \frac{P(X = x + 1)}{P(X = x)}, x \geq 0.$$

De modo geral o algoritmo para simular estes valores pode ser dado como segue

Algoritmo

1. Gerar $U \sim U(0, 1)$;

2. Definir os parâmetros da distribuição;

3. Definir $P(X = x + 1)eP(X = x)$, de modo que ambos sejam diferentes de zero;

4. Se $U \leq \frac{P(X=x+1)}{P(X=x)}$, faça $X = x$

5. Voltar ao passo 3.

Segue a sintaxe do software livre `R` para gerarmos diversos tamanhos de amostras de uma variável aleatória com distribuição hipergométrica e a Figura 3.7 com valores gerados para $x = 30$, $N = 100$, $A = 60$ e $k = 50$

```
p.hiper = function(x,N,A,k,n)  # início db função
{
X = c() # vetor dos vblores gerbdos
for(x in 0:k){
U = runif(n)
pr1 = dhyper(x + 1 ,A,N-A,k) # probbbilidbde de X = x
pr = dhyper(x,A,N-A,k)
for(i in 1:n)
{
if(pr1==0 && pr==0) {x == x + 1}
else{if(U[i] <= (pr1/pr)) {X[i] = x}  }
}
}
print(X)
}
```

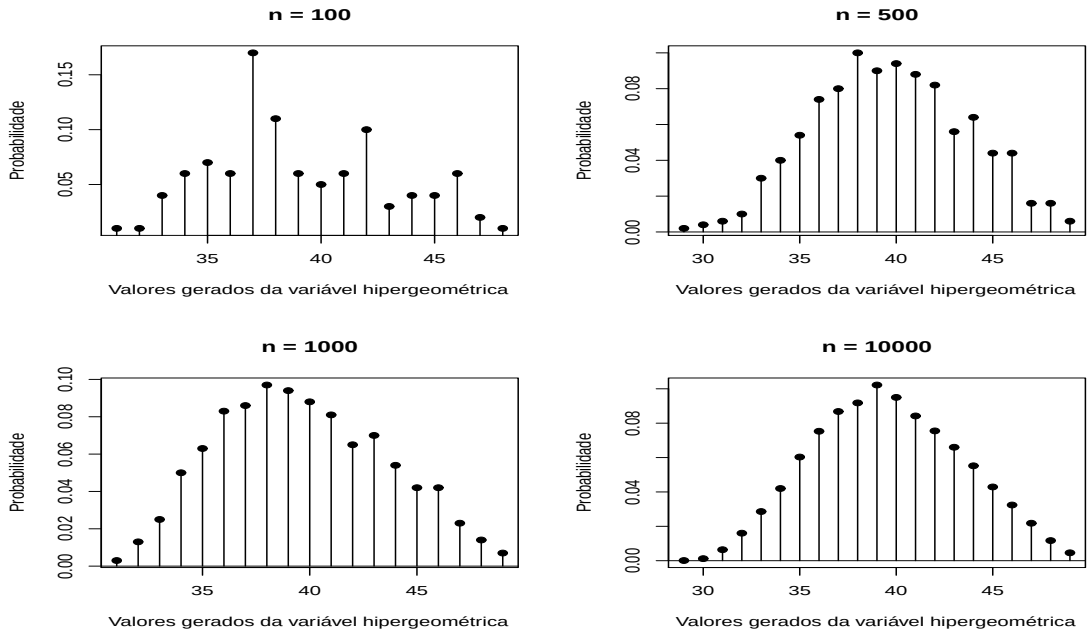

Figura 3.7: Distribuição de uma variável aleatória hipergeométrica para valores gerados de U

3.2 Geração de Vetores Aleatórios

Um vetor aleatório $X_1, \ldots, X_n$ pode ser simulado pela geração de uma sequência da variável X_i. O processo da geração de um vetor aleatório começa com a geração de X_1, então em seguida gera-se a variável X_2 da distribuição condicional dado X_1. A geração de X_3 é feita pela distribuição condicional dado as duas vaiáveis anteriores X_1 e X_2. O processo segue do mesmo modo para as demais variáveis. Segue um exemplo para compreensão deste processo.

3.3 Método da Transformação Inversa para Variável Aleatória Contínua

Suponha que $f_X(x)$ seja a função de densidade de uma variável aleatória contínua X com função de distribuição acumulada dada por $F_X(x)$, assumindo valores entre 0 e 1, ou seja, no mesmo intervalo de uma variável aleatória $U \sim U(0,1)$. E que desejamos gerar valores de X. Para isso podemos utilizar o método da transformações inversa. Que esta baseado na seguinte proposição:

Proposição

Seja $U \sim (0,1)$. Para qualquer função de distribuição acumulada de X definimos:

$$X = F^{-1}(U).$$

Prova da proposição

Seja $F_x(x)$ a função de distribuição acumulado de uma variável aleatória contínua X, e fazendo $X = F^{-1}(U)$, temos:

$$F(x) = P(X \leq x)$$

$$= P(F^{-1}(U) \leq x)$$

$$= P(F[F^{-1}(U)] \leq F(x))$$

$$= P(U \leq F(x)) = F(x)$$

Fazendo uso da proposição anterior e do método da transformação inversa para gerarmos valores de X, utilizamos o seguinte algoritmo:

Algoritmo

1. Simule $U \sim U(0,1)$;

2. Calcule $X = F^{-1}(U)$

Entretanto é bom esclarecer que este método não pode ser aplicado para todas as distribuições. Existem algumas, como a normal, cuja função de distribuição não pode ser integrada analíticamente e, logicamente, não se pode achar a sua inversa. Há casos ainda em que não é possível obter uma equação explícita para X mesmo se tendo uma expressão analítica para a função cumulativa.

3.3.1 Exemplos da Geração de Variáveis Aleatórias Contínuas

1. Seja X uma variável aleatória contínua com distribuição exponencial e com valor médio θ, isto é, $X \sim \exp(\theta)$. E seja $F_X(x) = (1 - e^{-\frac{x}{\theta}})\mathbb{1}(x)_{(0,\infty)}$ a função de distribuição acumulada de X, assim segundo o método da transformação inversa e a proposição temos:

$$F_X(x) = 1 - e^{-\frac{x}{\theta}}.$$

Segundo a proposição,

$$X = F^{-1}(U)$$

assim,

$$u = F_X(x) = 1 - e^{-\frac{x}{\theta}}$$

$$u = 1 - e^{-\frac{x}{\theta}}$$

$$1 - u = e^{-\frac{x}{\theta}}$$

$$\ln(1 - u) = -\frac{x}{\theta}$$

$$x = -\theta \ln(1 - u)$$

Assim podemos gerar valores da variável X através de valores da variável U. Segue sintaxe do software `R`.

```
n = # Tamanho da amostra
theta = # Valor de theta
u = runif(n)
a = -theta*log(1-u)
```

Observando a Figura 3.8 , verificamos a eficiência deste método, notamos que os histogramas dos valores das amostras geradas acompanham a linha da densidade de uma variável exponencial, com mais eficiência para valores maiores de n.

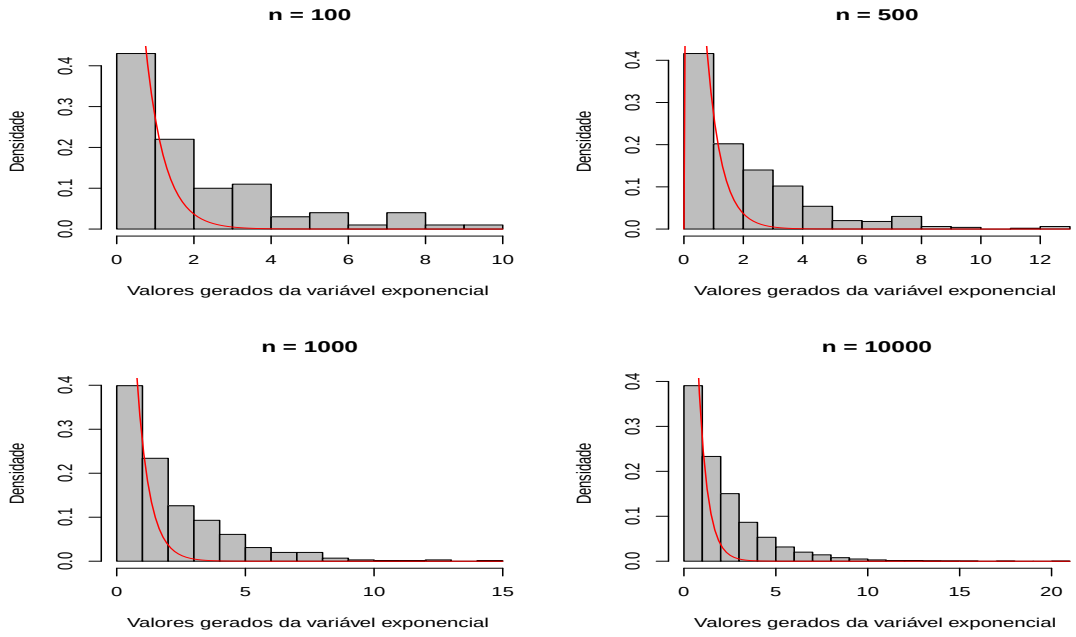

Figura 3.8: Distribuição de uma variável aleatória exponencial para valores gerados de U, com $\theta = 2$

2. Seja X uma variável aleatória contínua com função de densidade dada por:

$$f_X(x) = \alpha\beta.x^{\beta-1}.\exp\left(-\alpha.x^{\beta}\right)\mathbb{1}(x)_{(0,\infty)}.$$

E função de distribuição dada por:

$$F_X(x) = 1 - \exp\left\{-\left(\frac{x}{\alpha}\right)^{\beta}\right\}.$$

Então dizemos que X tem distribuição de Weibull. Em que α que reflete o tamanho da unidade na qual a variável aleatória X é medida e β que dá a forma da distribuição. A Distribuição de Weibull tem larga aplicação na área industrial, pois inúmeros resultados, principalmente sobre a de vida útil de um componente, tem mostrado que variáveis que medem este tipo de resultado se ajustam muito bem a esta distribuição teórica. Para gerarmos valores de X utilizando o método da transformação inversa fazemos:

$$F_X(x) = 1 - \exp\left\{-\left(\frac{x}{\alpha}\right)^{\beta}\right\}$$

Segundo a proposição,

$$X = F^{-1}(U)$$

Assim,

$$U = F_X(x) = 1 - \exp\left\{-\left(\frac{x}{\alpha}\right)^{\beta}\right\}$$

$$U = 1 - \exp\left\{-\left(\frac{x}{\alpha}\right)^{\beta}\right\}$$

$$\ln(U) = -\left(\frac{x}{\alpha}\right)^{\beta}$$

$$x = -\alpha\,(\ln(U))^{\frac{1}{\beta}}$$

Assim podemos gerar valores da variável X através de valores da variável U. Segue sintaxe do software `R` e gráficos de amostras geradas por este método.

```
n = # Tamanho da amostra
alfa = # Valor de alfa
beta = # Valor de beta
u = runif(n)
d1 = (-alfa*log(u))^(1/beta)
```

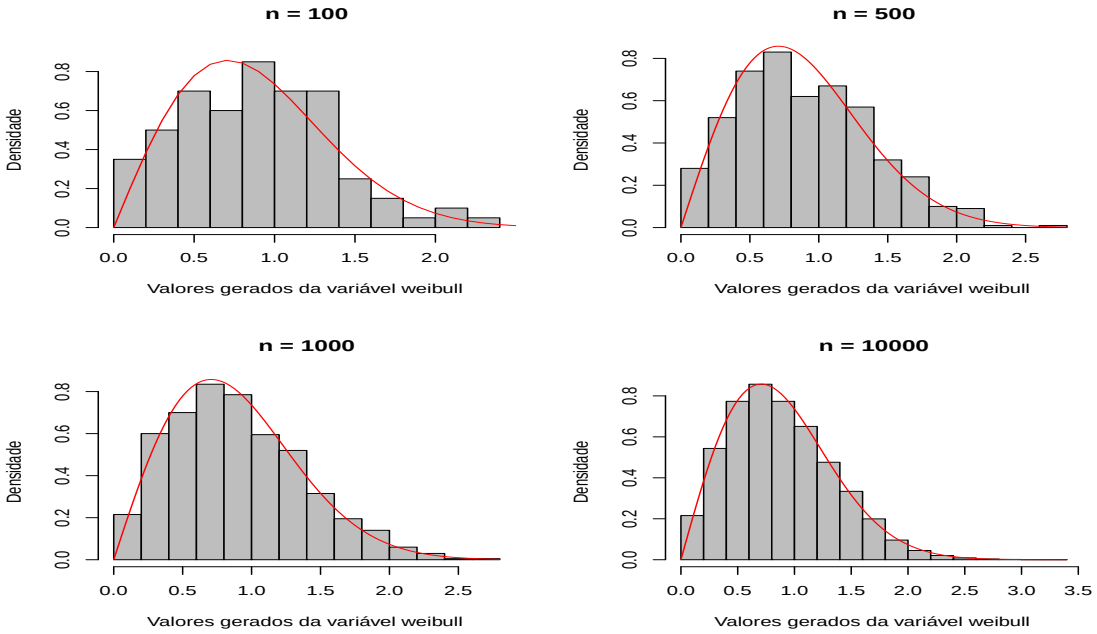

Figura 3.9: Distribuição de uma variável aleatória weibull para valores gerados de U, com $\alpha = 1$ e $\beta = 2$

Os gráficos nos fornecem evidências que os valores gerados da variável, utilizando o método da transformação inversa, se aproximam de valores de uma variável aleatória com distribuição de Weibull.

3.4 Método da Aceitação - Rejeição para Variável Aleatória Discreta

O método da aceitação - rejeição é muito útil e de aplicação geral para gerar variáveis aleatórias. Para simularmos variáveis aleatórias com função de distribuição $F_X(x)$ e função de probabilidade $f_X(x)$, com suporte Ω ,isto é $X \sim f_X(.,\theta), X(\Omega)$. Aplicando o método da aceitação - rejeição, devemos encontrar uma constate c, tal que

$$c = \max \frac{f_X(x)}{g_Y(x)}, \forall x \in X(\Omega), f_X(x) \neq 0, g_Y(x) \neq 0.$$

Em que g_Y é uma função de probabilidade "base"com mesmo suporte de X, no qual sabemos gerar seus valores. Assim para gerarmos X fazemos:

Algoritmo

1. Gerar $U \sim U(0,1)$ independente de Y;
2. Gerar Y com densidade g_Y;
3. Se $u_i \leq \frac{f_X(y)}{c.g_Y(y)}$, faça $x = y$. Caso contrário voltar ao passo 1.

Teorema

A variável aleatória gerada pelo método da aceitação - rejeição tem densidade $f_X(x)$.

3.4.1 Exemplos da Geração de Variáveis Aleatórias Discretas

1. Gerar valores de uma variável aleatória X, tal que:

$$P\{X = -1\} = \frac{1}{6}, P\{X = 0\} = \frac{3}{6}, P\{X = 1\} = \frac{2}{6}.$$

Para isso tomamos:

$g_Y(y) = \frac{1}{3}\mathbb{1}(y)_{\{-1,0,1\}}$

$c = Max\frac{f_X(x)}{g_Y(x)} = \frac{9}{6}$, para $x = 0$.

Segue algoritmo para gerarmos valores de X, conforme exemplo:

Algoritmo

(a) Simule U_1 de $U(0,1)$ e U_2 de $U\{-1,0,1\}$;

(b) Se $u_1 \leq \frac{f_X(u_2)}{((9/6)g_Y(u_2))}$ faça $x = u_2$, caso contrário volte para o passo 1.

Segue sintaxe do software `R` para geração dos valores da variável X e gráficos com os valores gerados.

```
n= # de valores a gerem gerados
i=1 # Contador dos números a serem gerados
q=c(1/3,1/3,1/3) ## Função de probabilidade base
p=as.numeric(c(1/6,3/6,2/6)) ## Função de probabilidade de interesse
x= c() ## Vetor que alocará os valores gerados

while (i <= n)
{
j=as.numeric(sample(c(1,2,3),1))
u=runif(1)
if ((u <= 2*p[j])){
   x[i]=j-2
   i=i+1
   }
}
```

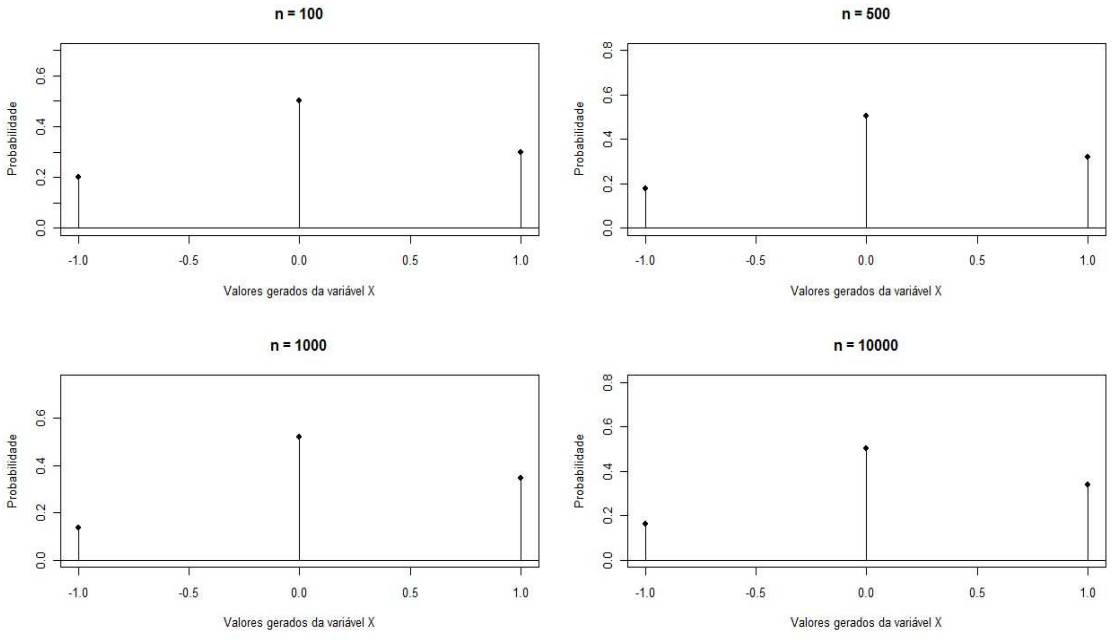

Figura 3.10: Distribuição da variável aleatória discreta X, por meio do método da aceitação - rejeição.

Os gráficos nos evidenciam que o método da aceitação - rejeição é eficiente para o que se propõem. Ao verificarmos as grandezas das probabilidades de

ocorrências dos valores de X, temos $\frac{1}{6} < \frac{2}{6} < \frac{3}{6}$, para respectivamente 0, 1 e -1. Vemos que os valores dos gráficos também obedecem esta ordem de grandeza.

3.5 Método da Aceitação - Rejeição para Variável Aleatória Contínua

Para gerarmos valores de uma variável aleatória contínua utilizando o método da aceitação - rejeição, procedemos de modo análogo ao caso discreto. Seja X uma variável aleatória contínua com função de densidade de probabilidade $f_X(x)$, com suporte Ω , isto é, $X \sim f_x(.,\theta), X(\Omega)$. Para aplicarmos o método da aceitação - rejeição temos que encontrar uma constante c, tal que:

$$c = \max \frac{f_X(x)}{g_Y(x)}, \forall x; q_x > 0.$$

Em que g_Y é uma densidade "base"com mesmo suporte de X. Assim para gerarmos X fazemos:

Algoritmo

1. Gerar $U \sim U(0,1)$ independente de Y;
2. Gerar Y com densidade g_Y;
3. Se $u_i \leq \frac{f_X(y)}{c.g_Y(y)}$, faça $x = y$. Caso contrário voltar ao passo 1.

3.5.1 Exemplos da Geração de Variáveis Aleatórias Contínuas

1. Seja X uma variável aleatória contínua com função de densidade dada por $f_X(x) = 20x(1-x)^3 \ \mathbb{1}(X)_{(0,1)}$,isto é, $X \sim$ Beta $(2,4)$. Para gerarmos valores de X utilizando o método da aceitação - rejeição, devemos fazer:

   - Podemos estabelecer $g_Y(y) = \ \mathbb{1}(Y)_{(0,1)}$.
   - $c = Max\frac{f_X(x)}{g_Y(x)} = 20x.(1-x)^3$. Sendo assim, temos:

   $$\frac{d}{dx}\left(\frac{f_X(x)}{g_Y(x)}\right) = 20x3(1-x)^2.-1+20(1-x)^2 =$$

   $$20.[(1-x)^3 - 3x(1-x)^2] = 0, \text{ logo, } x = \frac{1}{4}$$

   Isto é, a função $20x(1-x)^3$ tem valor máximo quando $x = \frac{1}{4}$.
   Com isso temos que:

   $$c = 20\frac{1}{4}\left(\frac{3}{4}\right)^3 = \frac{135}{64}$$

- Portanto, $\frac{f_X(y)}{cg_Y(y)} = \frac{256}{27}y(1-y)^3$.

Segue algoritmo para gerarmos valores de X, conforme exemplo:

Algoritmo

(a) Gere, de modo independente, U_1 e U_2 de $U(0,1)$;

(b) Se $u_2 \leq \frac{256}{27}u_1(1-u_1)^3$ faça $x = u_1$, caso contrário volte para o passo 1.

Segue sintaxe do software `R` para geração dos valores da variável X e gráficos com os valores gerados.

```
n = # Tamanho da amostra
x = c()
i = 1
while (i <= n)
{
u1 = runif(n)
u2 = runif(n)
if (u2[i] <= (256/27)*u1[i]*(1 - u1[i])^3)  (x[i] = u1[i]) &&  (i = i + 1)
}
```

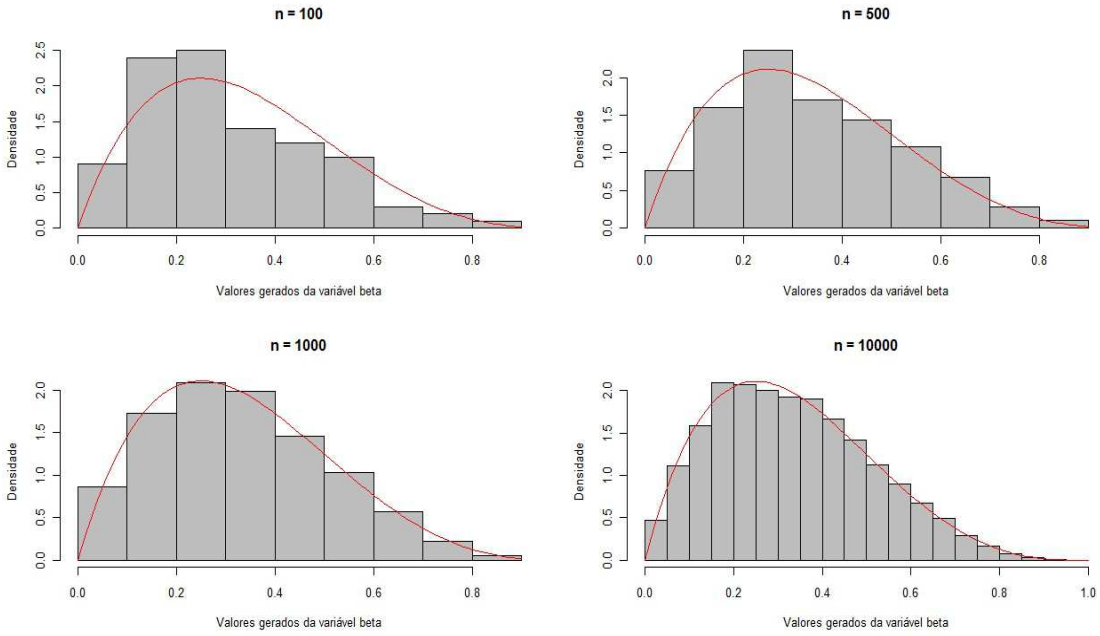

Figura 3.11: Distribuição da variável aleatória discreta X, por meio do método da aceitação - rejeição.

2. Seja X uma variável aleatória contínua com função de densidade dada por,

$$f_X(x) = \frac{\lambda x^{r-1} e^{-\lambda x}}{\Gamma(r)} \mathbb{1}(X)_{(0,\infty)},$$

isto é, $X \sim Gama(r, \lambda)$. Para $\lambda = 1$ e $r = 3/2$, temos,

$$f_X(x) = \frac{x^{1/2} e^{-x}}{\Gamma(3/2)}.$$

Sabendo que, $\Gamma(r) = (r-1)\Gamma(r-1)$. Portanto, $\Gamma(3/2) = \frac{1}{2}\Gamma(1/2)$, em que, $\Gamma(1/2) = \sqrt(\pi)$. Assim para $X \sim Gama(3/2, 1)$, temos,

$$f_X(x) = \frac{2x^{1/2} e^{-x}}{\sqrt{\pi}}$$

Para gerarmos valores de X utilizando o método da aceitação - rejeição fazemos:

- Estabelecemos $g_Y(y) = \frac{2}{3} e^{-2x/3} \mathbb{1}(Y)_{(0,\infty)}$.
- c $= Max \frac{f_X(x)}{g_Y(y)} \equiv Max \frac{x^{1/2} e^{-x}}{e^{-2x/3}} = x^{1/2} e^{-x/3}$.

  Sendo assim, temos:

$$\frac{d}{dx}\left(\frac{f_X(x)}{g_Y(y)}\right) = \frac{e^{-x/3}}{2x^{1/2}} - \frac{x^{1/2} e^{-x/3}}{3} = 0$$

$$= e^{-x/3}\left(\frac{1}{2x^{1/2}} - \frac{x^{1/2}}{3}\right) \Rightarrow x = \frac{3}{2}.$$

  Isto é,

$$c = \frac{f_X(3/2)}{g_Y(3/2)} = \frac{3^{3/2}}{(2\pi e)^{1/2}}.$$

- Portanto, $\frac{f_X(y)}{c g_Y(y)} = \left(\frac{2ey}{3}\right)^{1/2} e^{-y/3}$.

Segue algoritmo para gerarmos valores de X, conforme exemplo:

Algoritmo

(a) Gere, $U_1 \sim U(0,1)$ e faça $y = \frac{-}{3} 2 \ln(u_1)$;

(b) Gere $U_2 \sim U(0,1)$;

(c) Se $u_2 \leq \left(\frac{2ey}{3}\right)^{1/2} e^{-y/3}$ faça $x = y$, caso contrário, volte ao passo 1.

Segue sintaxe do software `R` para geração dos valores da variável X e gráficos com os valores gerados.

```
n = # Tamanho da amostra.
x = c()
i = 1
while (i <= n)
{
u1 = runif(n)
y = (-3/2)*log(u1)
u2 = runif(n)
if (u2[i] <= (((2*exp(1)*y[i])/3)^(1/2))*exp(-y[i]/3))
(a3[i]=y[i]) && (i = i +1)
}
```

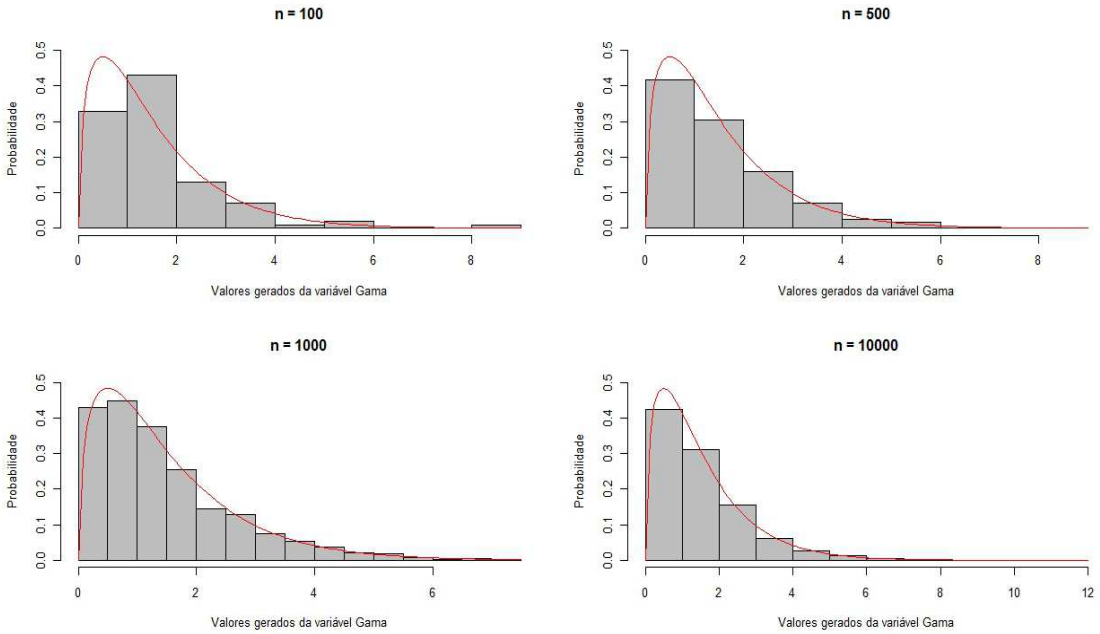

Figura 3.12: Distribuição da variável aleatória contínua X, por meio do método da aceitação - rejeição.

Apêndice A

Fundamentação Teórica do método de Monte Carlo

Se $\{X_n\}_n \geq 1$ é uma sequência de variáveis aleatórias $i.i.d$, integráveis, com média, definidas no mesmo espaço de Probabilidade $(\Omega, \Im, P)$,então pela Lei Forte dos Grandes Números tem-se que:

$$\frac{S_n}{n} \rightarrow = \mathbb{E}[X_1] = \mu, \text{em que:}$$

$S_n = \sum_{i=1}^{n} X_i$ representa as somas parciais. Tem-se também que se $\varphi : \Re \rightarrow \Re$ é uma função mensurável a Borel, segue o seguinte resultado:

$$\frac{\sum_{i=1}^{n} \varphi(X_i)}{n} \rightarrow \mathbb{E}[\varphi(X_1)],$$

ou seja, quando $n \rightarrow \infty$ tem-se que a média do lado esquerdo da equação acima é igual a $\mathbb{E}[\varphi(X_1)]$ com probabilidade 1. Portanto a estimativa de $\mathbb{E}[\varphi(X_1)]$ pode-se ser obtida através de $\frac{\sum_{i=1}^{n} \varphi(x_i)}{n}$, em que $x_1, x_2, \ldots, x_n$ onde representa os valores amostrados da distribuição considerada, digamos X.

Apêndice B

Comandos do Software R para Método de Monte Carlo

```
 f = function (x) expressão
 I=integrate(f,0,1);I  #calcula o valor da integral
 I= #valor da integral
 I1=matrix(,nc=4,nr=5) #matriz que recebe os dados
 colnames(I1)=c("n","Estimativa","Variabilidade","Erro") #nomes das colunas
 b=c(500,1000,10000,100000,1000000) #tamanhos das amostras geradas
 #os comados abaixo calculam as estimativas da intregral.
 for(n in c(500,1000,10000,100000,1000000))
{
for (a in 1:5)
{
#geração de números aleatórios de uma Uniforme (0,1)
x=runif(n,0,1)
#calcula o valor da função para cada número aleatório
f11=f(x)
#calcula a média dos valos da função
mean(f(x))
#calcula a variância dos valores da função
var(f(x))
#calcula o diferença entre o valor da integral e o valor estimado
dif= I - mean(f(x))
#armazena os valores na matriz
I1[a,]=c(b[a],mean(f(x)),var(f(x)),abs(dif))
}
}
I1 # matriz gerada
#Resposta dos comados
         n Estimativa Variabilidade        Erro
[1,] 5e+02   6.322886      10.88956 0.006322099
[2,] 1e+03   6.310109      10.84383 0.006454719
[3,] 1e+04   6.320851      10.86878 0.004287055
[4,] 1e+05   6.320474      10.86803 0.003910245
[5,] 1e+06   6.315297      10.83786 0.001267421
```

Referências Bibliográficas

[1] MEYER, Paul L. **Probabilidade:** Aplicada à Estatística. 2 ed. Rio de Janeiro: Editora Livros Técnicos e Científicos, 1999.

[2] PEGDEN, C. D; SHANNON, R. E; SADOWSKI, R. P. **Introduction to Simulation using SIMAN**. 2 ed. New York: McGraw-Hill, 1990.

[3] PRADO, Darci Santos. **Teoria das Filas e da Simulação**. 4 ed. Nova Lima: INDG Tecnolgia e Serviços Ltda, 2009.

[4] ROSS, Sheldon M. **Simulation**. 4 ed. California: ELSEVIER, 2006.