Lista 1 - MAE 0399 - Análise de Dados e Simulação

Guilherme Ventura (11340293), Milton Leal (8973974), Richard Sousa (11810898)

16/05/2021

Exercício 2:

Simulando de uma $X \sim Poisson(\lambda)$

a) Verifique a relação recursiva da distribuição Poisson dada por:

$$p_{j+1} = \frac{\lambda}{j+1} p_j,$$

em que $p_j = P(X = j), j = 0, 1, 2, ...$

R: Dado que, pela definição da Poisson: $p_j = \frac{\lambda^j e^{-\lambda}}{j!}$, então:

$$p_{j+1} = \frac{\lambda^{j+1} e^{-\lambda}}{(j+1)!} = \frac{\lambda \lambda^j e^{-\lambda}}{(j+1)j!} = \frac{\lambda}{j+1} \frac{\lambda^j e^{-\lambda}}{j!} \Rightarrow \frac{\lambda}{j+1} p_j$$

b) Construa um algoritmo para simular dessa distribuição baseado na função distribuição acumulada, a partir da simulação de um número aleatório básico $u \sim U_{(0,1)}$.

R:

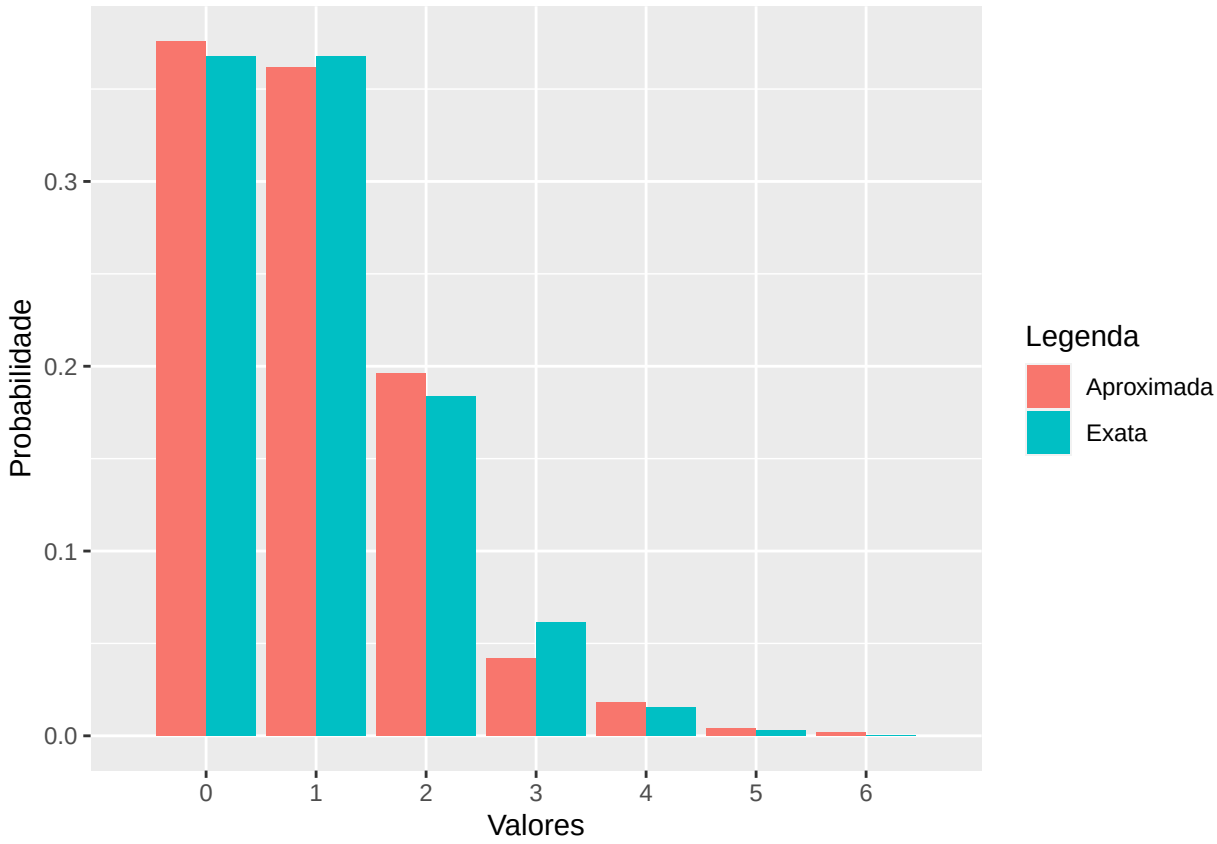
```
simula_poisson <- function(lambda){

    u <- runif(1,0,1) #gera valor aleatório de uma uniforme [0,1]
    p <- exp(-lambda) #função acumulada
    i <- 0 #contador do valor possível da variável aleatória
    cdf <- p #cópia da função acumulada

    while (TRUE) {

        if(u < cdf){ #verifica se u é menor do que a aplicação de u na acumulada
            x <- i #se for, encontramos a simulação
            break}

        else {
            #caso contrário, usa relação recursiva e atualiza a acumulada
            p <- lambda*p / (i + 1)
            cdf <- cdf + p #atualiza a cópia da acumulada
            i <- i + 1} #incrementa o contador
    }
    x #retorna o valor da variável aleatória simulada
}
```

c) Implemente o algoritmo (preferência pelo R), considerando diversos valores de λ. Use o seu algoritmo para simular uma amostra de tamanho 500 desta v.a. X. Faça um gráfico de barras para os valores simulados usando como altura as frequências relativas observadas e compare com as probabilidades exatas. Obtenha também as médias e variâncias amostrais e compare com as populacionais.

R:

- **Caso para** $\lambda = 1$

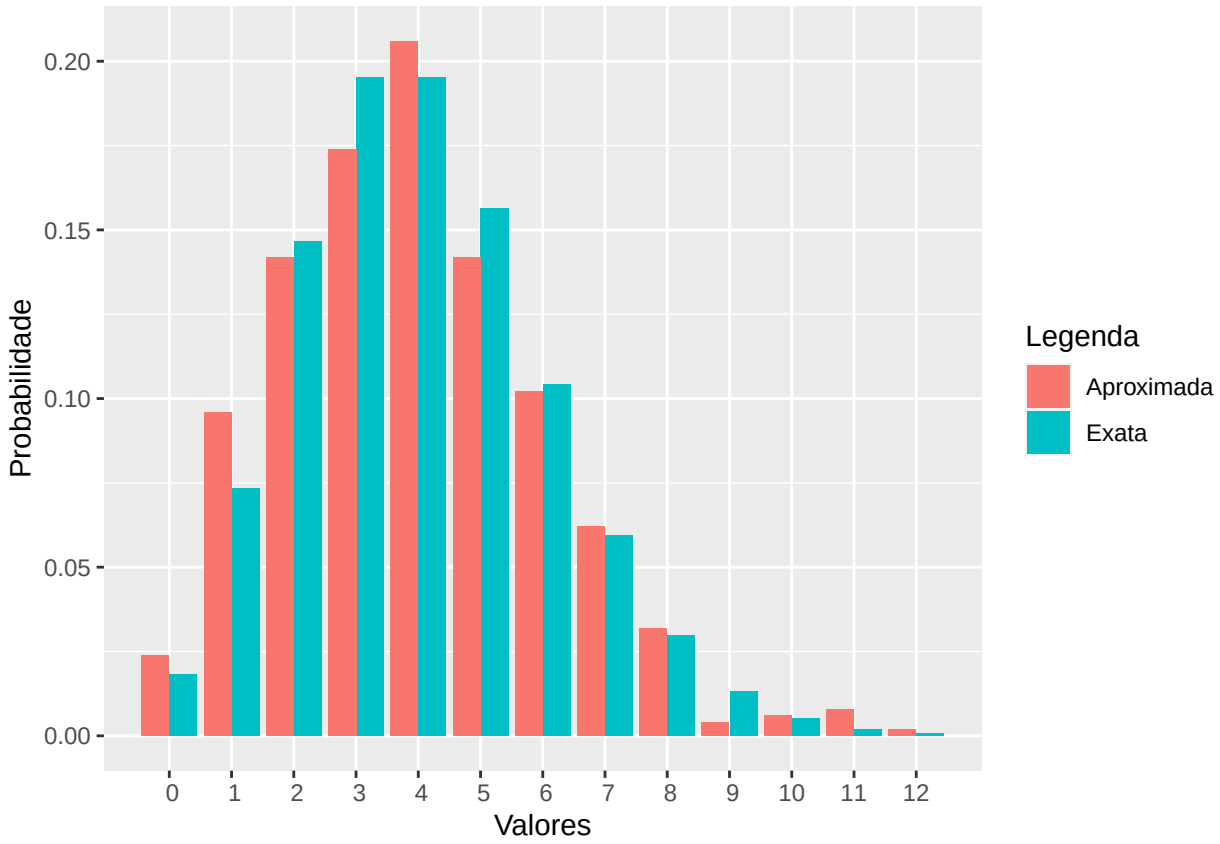
```
library(ggplot2)
lambda <- 1

set.seed(10)
simula <- replicate(500, simula_poisson(lambda))

tabela <- table(simula)

unicos <- sort(unique(simula))

prob_simula2 <- c()
for (i in 1:length(unicos)){
        prob_simula2<-c(prob_simula2,tabela[i]/length(simula))
}

prob_exata <- c()
for (i in unicos){

        prob_exata <- c(prob_exata, (lambda ** i * exp(-lambda)) / factorial(i))
}
df1 <- data.frame(Legenda = c(replicate(length(unicos), "Exata"),
                              replicate(length(unicos),"Aproximada")),
                  Valores = c(replicate(1, unicos)),
                  Probabilidade = c(prob_exata,prob_simula2))

grafico <- ggplot(data=df1, aes(x=Valores, y=Probabilidade, fill=Legenda)) +
        geom_bar(stat = "identity", position = position_dodge()) +
        scale_x_discrete("Valores", limits = (unicos))

grafico
```




```
media_simula <- c()
var_simula <-c()
media_simula <- c(media_simula, mean(simula))
var_simula <-c(var_simula, var(simula))
```

- **Caso para $\lambda = 4$**

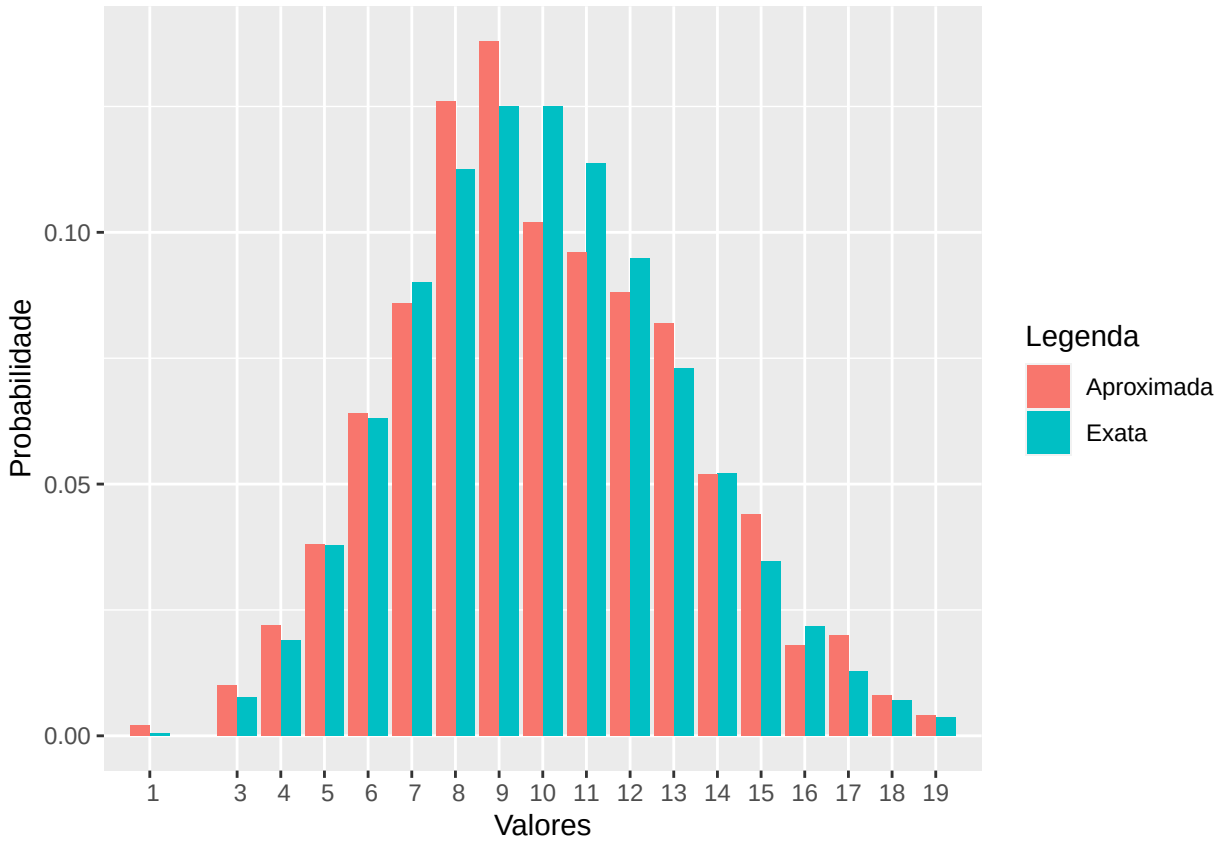
```
library(ggplot2)
lambda <- 4

set.seed(13)
simula <- replicate(500, simula_poisson(lambda))

tabela <- table(simula)

unicos <- sort(unique(simula))

prob_simula2 <- c()
for (i in 1:length(unicos)){
        prob_simula2<-c(prob_simula2,tabela[i]/length(simula))
}

prob_exata <- c()
for (i in unicos){

        prob_exata <- c(prob_exata, (lambda ** i * exp(-lambda)) / factorial(i))
}
df1 <- data.frame(Legenda = c(replicate(length(unicos), "Exata"),
                              replicate(length(unicos),"Aproximada")),
                  Valores = c(replicate(1, unicos)),
                  Probabilidade = c(prob_exata,prob_simula2))

grafico <- ggplot(data=df1, aes(x=Valores, y=Probabilidade, fill=Legenda)) +
        geom_bar(stat = "identity", position = position_dodge()) +
        scale_x_discrete("Valores", limits = (unicos))

grafico
```




```
media_simula <- c(media_simula, mean(simula))
var_simula <-c(var_simula, var(simula))
```

- **Caso para $\lambda = 10$**

```
library(ggplot2)
lambda <- 10

set.seed(5)
simula <- replicate(500, simula_poisson(lambda))

tabela <- table(simula)

unicos <- sort(unique(simula))

prob_simula2 <- c()
for (i in 1:length(unicos)){
        prob_simula2<-c(prob_simula2,tabela[i]/length(simula))
}

prob_exata <- c()
for (i in unicos){

        prob_exata <- c(prob_exata, (lambda ** i * exp(-lambda)) / factorial(i))
}
df1 <- data.frame(Legenda = c(replicate(length(unicos), "Exata"),
                              replicate(length(unicos),"Aproximada")),
                  Valores = c(replicate(1, unicos)),
                  Probabilidade = c(prob_exata,prob_simula2))

grafico <- ggplot(data=df1, aes(x=Valores, y=Probabilidade, fill=Legenda)) +
        geom_bar(stat = "identity", position = position_dodge()) +
        scale_x_discrete("Valores", limits = (unicos))

grafico
```




```
media_simula <- c(media_simula, mean(simula))
var_simula <-c(var_simula, var(simula))
```

- **Comparação**

```
df_compara <- data.frame(Lambda = c(1,4,10), media_simulacao = media_simula,
                         media_real = c(1,4,10), var_simulacao = var_simula,
                         var_real = c(1,4,10))

df_compara
```

```
##   Lambda media_simulacao media_real var_simulacao var_real
## 1      1           0.984          1      1.017780        1
## 2      4           3.946          4      4.451988        4
## 3     10           9.980         10     10.436473       10
```

A média e a variância populacional da Poisson são dadas por $E(x) = \lambda$ e $Var(x) = \lambda$. Em cada um dos casos acima, a média e a variância ficaram relativamente próximas do valor de λ, como era esperado pelas propriedades da distribuição Poisson, mas para aumentar a precisão precisaríamos aumentar o número de simulações.

Exercício 8:

Um baralho possui 100 cartas numeradas $1, 2, ..., 100$. As cartas são embaralhadas e então retiradas, uma a uma. Ocorre uma coincidência quando a i-ésima carta aparece na i-ésima retirada, $i = 1, 2, ..., 100$.

Escreva um programa para simular o processo e estimar a média e variância do total de coincidências. Obtenha os valores verdadeiros de média e variância e compare os valores simulados.

R:

```
#inicializa vetor que guarda número de coincidência em cada simulação
lista_coincid <- c()

set.seed(4)

for (k in 1:10000){

#gera números aleatórios de 1 a 100, sem reposição
embaralhadas <- sample(seq(1, 100), 100)

#verifica quantos números coincidiram com a ordem de 1 a 100
coincid <- sum(embaralhadas == seq(1, 100))

lista_coincid[k] <- coincid

}
```

Avaliando analiticamente o problema:

Seja $P(x_1)$ a probabilidade de obtermos 1 carta cujo número corresponde a sua ordem de retirada dentre 100 cartas.

Como o número de ordenações do deck de cartas para um dado número de coincidências é simplesmente o número de permutações das cartas restantes não coincidentes, temos:

$$P(x_1) = \frac{(100-1)!}{100!} = \frac{1}{100}$$

Vamos definir X como sendo a variável aleatória com distribuição $Bernoulli(\frac{1}{100})$ que representa a ocorrência de uma coincidência ou não de acordo com a ordem de retirada das cartas.

Ao somarmos todas essas Bernoullis, obteremos uma variável aleatória Y que terá distribuição $Binomial(100, \frac{1}{100})$

Calculemos agora a $E(Y)$:

$$\begin{aligned} E(Y) &= E(x_1 + \cdots + x_{100}) \\ &= E(x_1) + \cdots + E(x_{100}) \\ &= n \cdot E(x_1) \\ &= n \cdot \frac{1}{n} \\ &= 1 \end{aligned}$$

Para calcularmos $Var(Y)$, vamos primeiramente encontrar a $Var(x_i)$ e a $Cov(x_i, x_j)$, já que os eventos x_i não são independentes.

Como x_i é uma variável aleatória Bernoulli, de maneira geral temos:

$$Var(x_i) = \frac{1}{n}(1 - \frac{1}{n}) = \frac{n-1}{n^2}$$

Por sua vez, o único caso de covariância que nos interessa é quando $x_i = 1$ e $x_j = 1$:

$$\begin{aligned} Cov(x_i, x_j) &= E(x_i, x_j) - E(x_i)E(x_j) \\ &= (P(x_i = 1, x_j = 1) - E(x_i)E(x_j)) \\ &= \frac{1}{n(n-1)} - \frac{1}{n^2} \\ &= \frac{1}{n^2(n-1)} \end{aligned}$$

Agora estamos em condição de calcular $Var(Y)$ para o caso do nosso baralho:

$$Var(Y) = \sum_{i=1}^{100} Var(x_i) + 2\sum_{i \neq j} Cov(x_i, x_j)$$

$$= 100 \cdot \frac{100-1}{100^2} + 2\binom{100}{2}\frac{1}{100^2(100-1)} = 1$$

Portanto, analiticamente obtivemos $E(Y) = 1$ e $Var(Y) = 1$.

- **Comparação**

```
df_compara <- data.frame(media_simulacao = mean(lista_coincid),
                         media_real = 1, var_simulacao = var(lista_coincid),
                         var_real = 1)
df_compara
```

```
##   media_simulacao media_real var_simulacao var_real
## 1          0.9894          1     0.9589835        1
```

A média e a variância dos valores simulados ficaram relativamente próximos dos valores exatos.

Exercício 11:

Usando o algoritmo de rejeição com uma proposta $Exp(\lambda)$, simular de uma distribuição $Gamma(3,1)$ com fdp dada por:

$$f(x) = \frac{1}{2}x^2e^{-x} \quad x > 0.$$

a) Encontre o valor de λ que minimiza o número esperado de iterações do algoritmo. Para este valor de λ, qual é o número esperado de iterações do algoritmo? Qual é a probabilidade média de rejeição do algoritmo?

R:

Sejam $f(x) = \frac{1}{2}x^2e^{-x} \quad x > 0$ a função densidade da $Gamma(3,1)$ e $g(x) = \lambda e^{-\lambda x}$ a função densidade da $Exp(\lambda)$. Como vamos usar o método da aceitação/rejeição, temos que o número esperado de iterações é dado por:

$$c(\lambda) = Max_x \frac{f(x)}{g(x)} = \frac{\frac{x^2}{2}e^{-x}}{\lambda e^{-\lambda x}} = \frac{x^2e^{x(\lambda-1)}}{2\lambda}$$

Para encontrarmos o λ que minimiza o número de iterações, vamos primeiramente encontrar o valor máximo que x assume na equação acima derivando e igualando a zero.

$$\begin{aligned}\frac{d}{dx}\frac{x^2e^{x(\lambda-1)}}{2\lambda} = \frac{1}{2\lambda}(2xe^{x(\lambda-1)} + \lambda x^2e^{x(\lambda-1)} - x^2e^{x(\lambda-1)}) = 0 \\ \Rightarrow x^2e^{x(\lambda-1)} = e^{x(\lambda-1)}(2x + \lambda x^2) \\ \Rightarrow x^2 = 2x + \lambda x^2 \\ \Rightarrow x = \frac{2}{1-\lambda}\end{aligned}$$

Como a segunda derivada é negativa, $x = \frac{2}{1-\lambda}$ é ponto de máximo. Substituindo o valor encontrado em $c(\lambda)$, encontramos:

$$c(\frac{2}{1-\lambda}) = \frac{(\frac{2}{1-\lambda})^2 \cdot e^{\frac{2}{1-\lambda}(\lambda-1)}}{2\lambda} = \frac{2e^{-2}}{\lambda(-\lambda+1)^2}$$

Para encontrarmos o λ que minimiza o número de iterações, vamos derivar e igualar a zero.

$$\begin{aligned}\frac{d}{d\lambda}\frac{2e^{-2}}{\lambda(-\lambda+1)^2} = -\frac{2(3\lambda-1)}{e^2\lambda^2(\lambda-1)^3} = 0 \\ \Rightarrow \lambda = \frac{1}{3}\end{aligned}$$

Como a segunda derivada é negativa, temos que $\lambda = \frac{1}{3}$ é o valor que minimiza $c(\lambda)$.

Para $\lambda = \frac{1}{3}$, o número esperado de iterações é dado por:

$$c(\frac{2}{1-\frac{1}{3}}) = \frac{2e^{-2}}{\frac{1}{3}(-\frac{1}{3}+1)^2} \approx 1.827$$

A probabilidade média de rejeição do algoritmo é dada por $1 - \frac{1}{c(\frac{1}{3})} \approx \frac{1}{1.827} \approx 0.452$

b) Compare os valores esperados da distribuição proposta e da distribuição de interesse.

R:

O valor esperado de uma variável aleatória X com distribuição $Gamma(3,1)$ é dado por:

$$E(X) = 3 \cdot 1 = 3$$

O valor esperado de uma variável aleatória Y com distribuição $Exp(\frac{1}{3})$ é dada por:

$$E(Y) = \frac{1}{\frac{1}{3}} = 3$$

Portanto, os valores esperados de ambas as distribuições são iguais.

c)Implemente o algoritmo (preferencialmente no R). Apresente os resultados desenhando um gráfico do tipo Box-plot dos valores simulados.

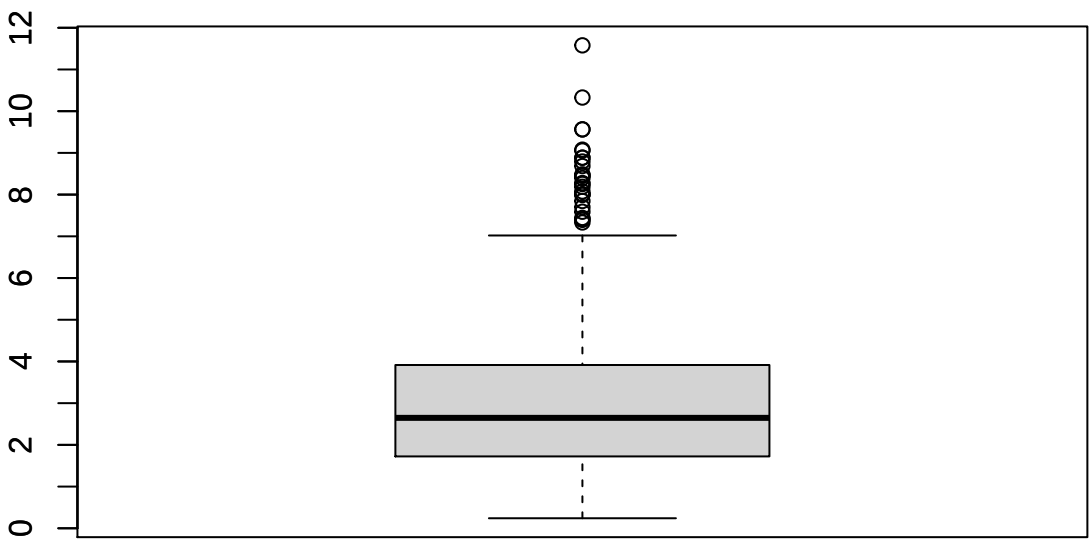
R:

```
simula_gamma <- function(lambda){

        f <- function (y){1/2 * (y^2 * exp(-y))} #densidade da Gama(3,1)

        g <- function (y){1/3 * (exp(-(1/3) * y))} #densidade da Exp (1/3)

        c = 1/1.827

        while (TRUE) {

          u1 <- runif(1,0,1) #gera valor aleatório de uma uniforme [0,1]
          u2 <- runif(1,0,1) #gera outro valor aleatório da uniforme [0,1]

          y = -3*log(u1) #simula da Exp pela Transformada Inversa

          alpha = c*(f(y)/g(y)) #calcula alpha

          if(u2 < alpha) { #compara se u2 é menor que alpha

              x <- y #se for, encontramos o valor simulado da Gama
              break

                }
        }
        x
}

x <- replicate(1000, simula_gamma(1/3))
```

```
mean(x) #média dos valores simulados
```

```
## [1] 3.002329
```

```
boxplot(x, main="Boxplot dos valores simulados da Gama(3,1)")
axis(2, seq(0,12, 1))
```


Boxplot dos valores simulados da Gama(3,1)
0
2
4
6
8
10
12