Faculdade de Ciências e Tecnologias
Universidade NOVA de Lisboa
Estatística Numérica Computacional

PROJETO 1

Ana Breia - 61877
Gonçalo Santos - 55585
João Funenga - 61635
Mário Miranda - 62286

2021

Conteúdo

Introdução

Este projeto de Estatística Numérica Computacional recaíu sobre três grandes grupos. Estes foram a geração de variáveis aleatórias, métodos de estimação de Monte Carlo no domínio dos integrais bem como intervalos de confiança.

Na primeira parte iremos analisar e estudar como gerar observações de variáveis aleatórias com funções de densidade contínuas conhecidas através de dois métodos abordados nas aulas, *Método da Transformação Inversa* e *Método da Aceitação-Rejeição*.

De seguida, para a estimação de Monte Carlo, iremos perceber como estimar o valor de um dado integral. Para além da versão Naive, usaremos três técnicas, *Variáveis Antitéticas, Variáveis de Controlo* e *Importance Sampling*. Perceberemos que dentro destas técnicas, nem todas têm vantagens comparativamente ao método original (sendo a vantagem a percentagem de redução de variância) e tal será representado e analisado no final do trabalho.

Relativamente aos intervalos de confiança iremos discutir, para amostras aleatórias seguindo uma distribuição normal, qual o efeito de contaminações nessa dada amostra.

Geração de Variáveis Aleatórias

Método da Transformação Inversa - Variáveis Aleatórias Contínuas

O Método da Tranformação Inversa é um método muito utilizado na geração de variáveis aleatórias, e baseia-se no seguinte resultado:

Teorema 1. *Seja $U \sim U(0,1)$ uma variável aleatória, e F uma qualquer função distribuição contínua. Ora, se X é uma variável aleatória definida por*

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

então tem como função distribuição F.

Demonstração. Denotemos por F_X a função distribuição de $X = F^{-1}(U)$. Assim, queremos provar que $F_X(x) = F(x)$.
Tem-se então que

$$F_X(x) = Pr(X \leq x) = Pr(F^{-1}(U) \leq x) \tag{1}$$

Ora, uma vez que $F(x)$ é uma função monótona crescente (por ser uma função distribuição), tem-se que

$$F^{-1}(U) \leq a \iff U \leq F(a). \tag{2}$$

Conclui-se então, de (1) e de (2) que

$$F_X(x) = Pr(U \leq F(x)) = F(x)$$

□

Assim, pelo **Teorema 1**, concluímos que, se gerarmos uma observação aleatório U e definirmos $X = F^{-1}(U)$, então podemos gerar uma v.a. X, com função distribuição F.

Como exemplo, consideremos que queremos gerar amostras de $X \sim U(a,b), a < b$.

Assim, o Método da Tranformação Inversa tem o seguinte algoritmo:

1. Gerar uma observação u a partir de $U(0,1)$;
2. Fazer $x = a + (b-a)u$ (através do cálculo de $u = F(x)$ em ordem a x);
3. Repetir os passos anteriores até alcançar o número de amostras desejado.

Neste projeto apenas abordamos o caso de variáveis contínuas. Para o caso discreto ver [1].

Exercício 1

Let $X \sim TruncatedPareto(\alpha, L, H)$, which has probability density function

$$f(x) = \frac{\alpha L^{\alpha} x^{-\alpha-1}}{1 - \left(\frac{L}{H}\right)^{\alpha}}, \qquad \alpha, L > 0 \quad H > L \quad x \in [L, H].$$

a) Analytically derive the cumulative distribution function (c.d.f.) F of X as well as its inverse F^{-1}.

Para este exercício temos como objetivo chegar à função distribuição cumulativa de probabilidade (c.d.f) $F(X)$ e de seguida o cálculo da sua inversa. Começaremos por calcular a c.d.f da função p.d.f que nos

foi dada no enunciado. Para isto, é necessário calcular o integral desta função. Pelo apresentado no enunciado, os limites inferior e superior deste integral serão respetivamente L e x:

Cálculo da c.d.f.

$$\begin{aligned} F(x) &= \int_L^x f(z)dz = \int_L^x \frac{\alpha L^\alpha z^{-\alpha-1}}{1-\left(\frac{L}{H}\right)^\alpha} dz = \\ &= \frac{\alpha L^\alpha}{1-\left(\frac{L}{H}\right)^\alpha} \int_L^x z^{-\alpha-1} dz = \\ &= \frac{\alpha L^\alpha}{1-\left(\frac{L}{H}\right)^\alpha} \left[\frac{z^{-\alpha}}{-\alpha}\right|_L^x dz = \\ &= \frac{-L^\alpha x^{-\alpha}}{1-\left(\frac{L}{H}\right)^\alpha} + \frac{L^{\alpha-\alpha}}{1+\left(\frac{L}{H}\right)^\alpha} = \\ &= \frac{1-L^\alpha x^{-\alpha}}{1-\left(\frac{L}{H}\right)^\alpha} \end{aligned} \tag{3}$$

Cálculo da inversa da c.d.f.

Consideremos agora $F(x) = u$.

$$u = \frac{1-L^\alpha x^{-\alpha}}{1-\left(\frac{L}{H}\right)^\alpha} \Leftrightarrow u\left(1-\left(\frac{L}{H}\right)^\alpha\right) - 1 = -L^\alpha x^{-\alpha} \Leftrightarrow$$

$$\Leftrightarrow \frac{u\left(1-\left(\frac{L}{H}\right)^\alpha\right)-1}{-L^\alpha} = x^{-\alpha} \Leftrightarrow \frac{-L^\alpha}{u\left(1-\left(\frac{L}{H}\right)^\alpha\right)-1} = x^\alpha \Leftrightarrow$$

$$\Leftrightarrow \left(\frac{-L^\alpha}{u\left(1-\left(\frac{L}{H}\right)^\alpha\right)-1}\right)^{\frac{1}{\alpha}} = x \Leftrightarrow$$

$$F^{-1}(u) = \left(\frac{-L^\alpha}{u\left(1-\left(\frac{L}{H}\right)^\alpha\right)-1}\right)^{\frac{1}{\alpha}} = \frac{-L}{(u\left(1-\left(\frac{L}{H}\right)^\alpha\right)-1)^{\frac{1}{\alpha}}}$$

b) Implement this method in R. Call your routine sim.IT() and let it receive as input a generic sample size m as well as generic α, L and H parameters.

Tendo em conta a explicação de como o método funciona no início desta secção iremos agora implementar o desejado em R. Para concretizar este algoritmo, criámos uma função sim.IT() que recebe como parâmetros o tamanho da amostra m, α, e os limites L e H. Abaixo temos a função *inverseCDF()* que representa a inversa da p.d.f (calculada na alínea a):

```
inverseCDF = function(u, alpha, L, H) {
    ((u * (1 - (L/H)^alpha) - 1)/(-L^alpha))^(1/-alpha)
}
```

Neste algoritmo pretendemos repetir a operação de:

1. Gerar uma observação aleatória de uma distribuição uniforme;
2. Determinar x, onde $F(x) = u$;
3. Adicionar o resultado num vetor que será depois representado no formato de um histograma para visualização dos resultados.

```r
sim.IT = function(m, alpha, L, H) {
    x = vector()
    for (i in 1:m) {
        u = runif(1, 0, 1)
        inversa = inverseCDF(u, alpha, L, H)
        # ir adicionando a inversa do resultado da integral
        x = c(x, inversa)
    }
    x
}
```

c) Use routine sim.IT() to generate a sample of size $m = 10000$ of

$$X \sim TruncatedPareto(0.5, 2, 4)$$

Report the first 10 simulated values. Explicitly derive and simplify the expression of the p.d.f.. Plot the sample histogram with the true p.d.f. superimposed.

Os 10 primeiros valores simulados a partir do método da transformada inversa são:

```r
head(sim.IT(10000, 0.5, 2, 4), 10)
```

```
##  [1] 2.359088 3.626653 2.787778 3.846843 2.215043 2.953308 3.253549 2.256881
##  [9] 2.542140 3.598445
```

Função p.d.f. simplificada com os respetivos valores para α, L e H

$$f(x) = \frac{\sqrt{2}x^{\frac{-3}{2}}\left(2+2\sqrt{\frac{1}{2}}\right)}{\left(2-2\sqrt{\frac{1}{2}}\right)\left(2+2\sqrt{\frac{1}{2}}\right)} = \frac{\sqrt{2}x^{\frac{-3}{2}}\left(2+2\sqrt{\frac{1}{2}}\right)}{4+4\sqrt{\frac{1}{2}}-4\sqrt{\frac{1}{2}}-4\left(\sqrt{\frac{1}{2}}\right)^2} = \frac{2\sqrt{2}x^{\frac{-3}{2}}+2\sqrt{\frac{1}{2}}\sqrt{2}x^{\frac{-3}{2}}}{4-4\left(\frac{1}{2}\right)} =$$

$$\frac{2\sqrt{2}x^{\frac{-3}{2}}+2\sqrt{\frac{1}{2}}\sqrt{2}x^{\frac{-3}{2}}}{2} = \frac{2\left(\sqrt{2}x^{\frac{-3}{2}}+\sqrt{\frac{1}{2}}\sqrt{2}x^{\frac{-3}{2}}\right)}{2} =$$

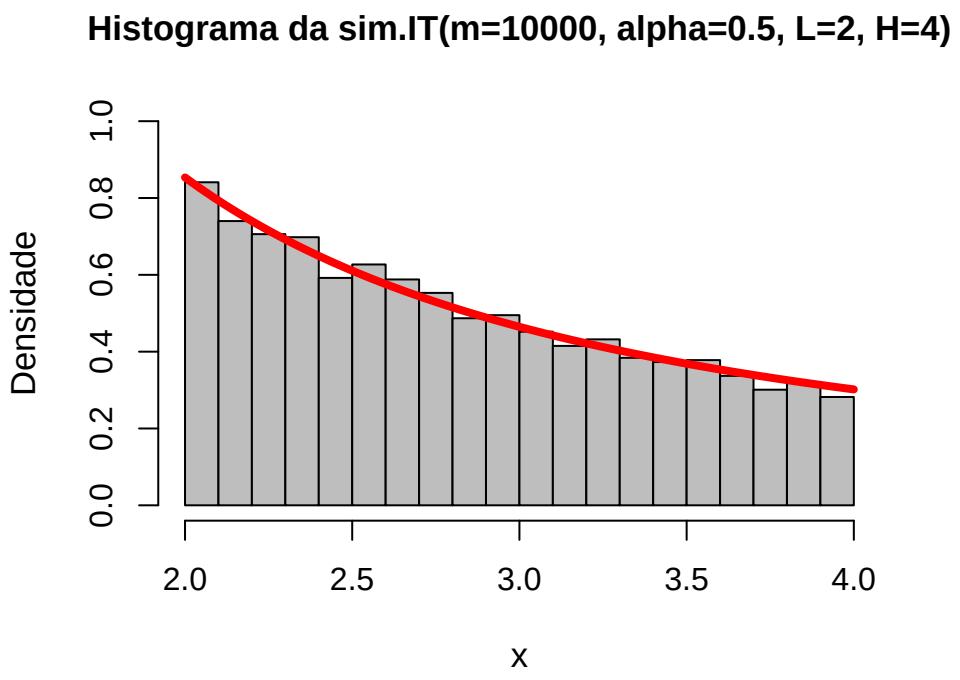
$$\frac{2\left(\sqrt{2}x^{\frac{-3}{2}}+\sqrt{\frac{1}{2}}\sqrt{2}x^{\frac{-3}{2}}\right)}{2} = \sqrt{2}x^{\frac{-3}{2}}+x^{\frac{-3}{2}} =$$

$$x^{\frac{-3}{2}}\left(1+\sqrt{2}\right)$$

Vamos agora utilizar a função sim.IT(). Para tal usamos como valores para a simulação *m=10000*, *α=0.5*, *L=2*, *H=4*. Antes de realizarmos as simulações, iremos fixar a semente para que o resultado das observações aleatórias não se altere. Sobreposto ao histograma, faremos o plot da curva da p.d.f cuja função está também declarada de seguida. Se esta seguir o mesmo formato que o do histograma, a geração das variáveis aleatórias com a distribuição dada foi bem sucedida.

```r
# Função pdf do enunciado
truncatedParetoPDF = function(x, alpha, L, H) {
    (alpha * L^(alpha) * x^(-alpha - 1))/(1 - (L/H)^alpha)
}

# Fixar a semente
set.seed(123)

# Gerar o histograma
hist(sim.IT(10000, 0.5, 2, 4), freq = F, col = "grey", ylim = c(0,
    1), cex.main = 1.1, ylab = "Densidade", xlab = "x", cex.lab = 1.1,
    main = "Histograma da sim.IT(m=10000, alpha=0.5, L=2, H=4)")
```

```r
# Para adicionar a curva da p.d.f sobreposta ao histograma
# gerado pela simulação
curve(truncatedParetoPDF(x = x, alpha = 0.5, L = 2, H = 4), add = T,
    col = "red", lwd = 4, lty = 1)
```



Método da Aceitação-Rejeição - Variáveis Aleatórias Contínuas

Suponhamos agora que não conseguimos encontrar uma forma explícita, ou simples, para $F^{-1}(x)$. Neste caso não estamos em condições de utilizar o Método da Transformação Inversa. Neste capítulo abordaremos um outro método para a geração de variáveis aleatórios, o Método da Aceitação-Rejeição.

Para tal, começamos por considerar X, uma variável aleatória contínua com função densidade $f(x)$. Consideremos agora uma função densidade $g(x)$, da qual já conhecemos um algoritmo eficiente para a geração de novas variáveis, e tal que

$$\exists M > 0 : f(x) \leq Mg(x), \forall x$$

A ideia principal deste método é encontrar a função $g(x)$, denominada função candidata, que satisfaça estas condições e de seguida rejeitar todas as observações que são prováveis de "calhar" abaixo de $f(x)$, denominada função objetivo.

Assim, o Método da Aceitação-Rejeição tem o seguinte algoritmo:

1. Gerar uma observação y a partir de g.
2. Gerar uma observação u a partir de $U(0,1)$.
3. Determinar $\alpha = \frac{1}{M}\frac{f(y)}{g(y)}$.
4. Se $u \leq \alpha$, fazer $x = y$. Se não, voltar ao passo 1.
5. Repetir todos os passos até obter o número de amostras desejado.

Com isto é importante provar que a variável gerada pelo método tem de facto função densidade $f(x)$:

Teorema 2. *A variável aleatória X gerada pelo Método da Aceitação-Rejeição tem função densidade $f(x)$.*

Demonstração. Seja Y a variável aleatória com função densidade $g(x)$ e N o número total de iterações necessárias até obter a amostra com o tamanho desejado. Então, tem-se que

$$\begin{aligned} P\{X \leq x\} &= P\{Y_N \leq x\} \\ &= P\{Y \leq x | u \leq \frac{f(Y)}{Mg(Y)}\} \\ &= \frac{P\{Y \leq x, u \leq \frac{f(Y)}{Mg(Y)}\}}{K}, \end{aligned} \tag{4}$$

onde $K = P\{u \leq \frac{f(Y)}{Mg(Y)}\}$. Ora, uma vez que Y e U são independentes, a sua função densidade conjuta é dada por

$$f(y,u) = g(y), 0 < u < 1$$

Assim, tem-se que

$$\begin{aligned} P\{X \leq x\} &= \frac{1}{K}\int_{y \leq x}\int_{0 \leq u \leq f(y)/Mg(y)} g(y)\,du\,dy \\ &= \frac{1}{K}\int_{-\infty}^{x}\int_{0}^{f(y)/Mg(y)} du g(y)\,dy \\ &= \frac{1}{MK}\int_{-\infty}^{x} f(y)\,dy \end{aligned} \tag{5}$$

Ora, se considerarmos X a tender para ∞, temos que

$$1 = \frac{1}{MK}\int_{-\infty}^{\infty} f(y)\,dy = \frac{1}{MK}, \tag{6}$$

uma vez que f é uma função densidade.

Desta forma, de (2) e de (3) obtemos

$$P\{X \leq x\} = \int_{-\infty}^{x} f(y)\,dy,$$

donde se conclui que f é de facto função densidade de X. □

Exercício 1

d) Identify the candidate density function for the particular case of the $X \sim TruncatedPareto(0.5, 2, 4)$ and compute by hand the constants of the AR method (namely, M and the acceptance probability α). Use the R function optimize() or other to confirm the result you obtained for M.

Visto que o suporte da nossa variável aleatória X é o intervalo $[L, H]$, a escolha natural para a função candidata é a distribuição uniforme $U(0, 1)$, isto é, $g(x) = 1, L \leq x \leq H$.

Neste método de geração de amostras aleatórios queremos que a nossa função candidata esteja "acima" da função que segue a distribuição da qual queremos realizar observações. Para isto, teremos de multiplicar a função candidata pelo valor de M (calculado a partir do máximo da função $h(x) = f(x)/g(x)$, no intervalo de L a H, neste caso, 2 a 4). Como a função candidata é $g(x) = 1$, a função $h(x)$ será igual a $f(x)$.

```
# funcao h = f/g nossa funcao g vai ser a funcao uniforme
# y=1
hFunc = function(x, alpha, L, H) {
    ((alpha * (L^alpha) * x^(-alpha - 1))/(1 - (L/H)^alpha))/1
}
```

Tendo em conta que a função é estritamente decrescente e estamos a trabalhar no intervalo de L a H, sendo estes valores neste caso respetivamente 2 e 4, o máximo da função $h(x)$, M, será em $x = 2$. Para calcular o valor da ordenada deste ponto máximo podemos substituir na nossa função $x = 2$. Substituindo $x = 2$ na função:

$$y = 2^{\frac{-3}{2}}(1 + \sqrt{2}) = 0.8535$$

Assim, temos que o ponto máximo desta função delimitada de L a H tem como coordenadas

$$(x = 2, y = 0.8535).$$

Para verificarmos que o valor máximo da função calculado analiticamente está correto podemos utilizar a funcao optimize() do R no mesmo intervalo.

```
maximoH = optimize(hFunc, c(2, 4), maximum = T, alpha = 0.5,
    L = 2, H = 4)$objective
maximoH
```

```
## [1] 0.8535271
```

Com isto já podemos criar a função que será usada como candidata (já multiplicada por M) para poder ficar "acima" da p.d.f $f(x)$.

```
# multiplicar por x e dividir por x, senao dá erro a dizer
# que nao é uma funcao
candidataVezesM = function(x, M) {
    M * 1 * x/x
}
```

Como explicado no início, precisamos agora de calcular α, dado por:

$$\alpha = \frac{1}{M}\frac{f(y)}{g(y)},$$

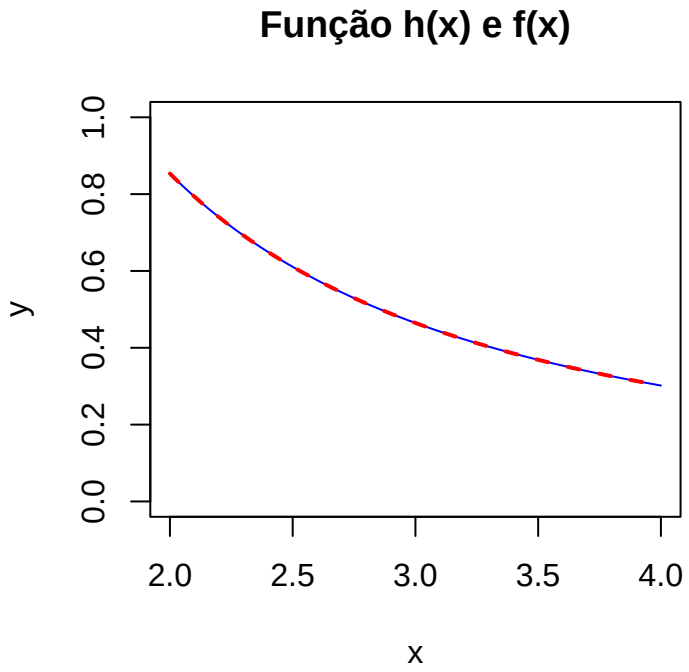
onde y representa uma observação da nossa função candidata.

Neste caso em específico ficaríamos com:

$$\alpha = \frac{1}{0.8535} x^{\frac{-3}{2}} \left(1 + \sqrt{2}\right)$$

Como a função candidata usada é a uniforme, fazendo o plot de $h(x)$ (que resulta da divisão de $f(x)/g(x)$) a curva resultante sobrepõe-se à função $f(x)$.

```
curve(hFunc(x = x, alpha = 0.5, L = 2, H = 4), col = "blue",
    xlim = c(2, 4), ylim = c(0, 1), main = "Função h(x) e f(x)",
    ylab = "y")

curve(truncatedParetoPDF(x = x, alpha = 0.5, L = 2, H = 4), add = T,
    col = "red", lwd = 2, lty = 2)
```



e) Implement the AR method in R. Call your routine sim.AR() and let it receive as input a generic sample size m as well as generic α, L and H parameters. Besides returning the simulated values of the target distribution, the sim.AR() routine should also return the simulated values that were rejected.

Agora iremos implementar o método de aceitação-rejeição *sim.AR()* que tem como argumentos o número de observações a serem feitas, para além dos que a função p.d.f leva originalmente. Neste caso em específico a abordagem foi a seguinte:

1. Gerar uma observação candidata $y \sim U(L, H)$;
2. Calcular a probabilidade de aceitar y, cujo valor corresponde ao α falado anteriormente;
3. Gerar uma observação u de uma distribuição uniforme U(0,1);
4. Caso $u > \alpha$, a observação é considerada "falhada"e adicionamos as coordenadas desse ponto a um vetor para posterior demonstração num gráfico. Caso contrário, adicionar esta observação às bem sucedidas;

5. Repetir até alcançar o número de amostras desejado (das aceites).

Abaixo encontra-se o algoritmo explicado em R:

```
sim.AR = function(n, alpha, L, H) {
    x <- rej_x <- yx <- yrej_x <- vector()
    M <- optimize(hFunc, c(2, 4), maximum = T, alpha = alpha,
        L = L, H = H)$objective
    for (i in 1:n) {
        u = 1
        a = 0
        while (u > a) {
            # Gerar uma observação da distrib candidata no
            # intervalo desejado
            x.c <- runif(1, L, H)
            a <- (1/M) * truncatedParetoPDF(x.c, alpha, L, H)
            u <- runif(1, 0, 1)
            # Caso a observaçao seja superior ao alpha
            # calculado adicionar às obsv de rejeição
            if (u > a) {
                rej_x <- c(rej_x, x.c)
                yrej_x <- c(yrej_x, u * candidataVezesM(x.c,
                  M))
            }
        }
        # Adicionar no caso das observações bem sucedidas
        x <- c(x, x.c)
        yx <- c(yx, u * candidataVezesM(x.c, M))
    }
    # Retornar as listas para os pontos x e y tanto
    # rejeitados como aceites
    return(list(x = x, rej_x = rej_x, yx = yx, yrej_x = yrej_x))
}
```

f) Use routine sim.AR() to generate a sample of size m = 10000 of

$$X \sim TruncatedPareto(0.5, 2, 4).$$

Report the first 10 simulated values of the $TruncatedPareto(0.5, 2, 4)$ and the rejection rate. Plot the sample histogram with the true p.d.f. superimposed.

Graphically display, the candidate and p.d.f. functions against the hits and misses of the AR method (as done in class – week 2) when used to simulate just $m = 15$ sample values.

Utilizando agora o método sim.AR() para 10000 observações (m=10000), $\alpha = 0.5$, L=2, H=4, iremos verificar os 10 primeiros valores observados.

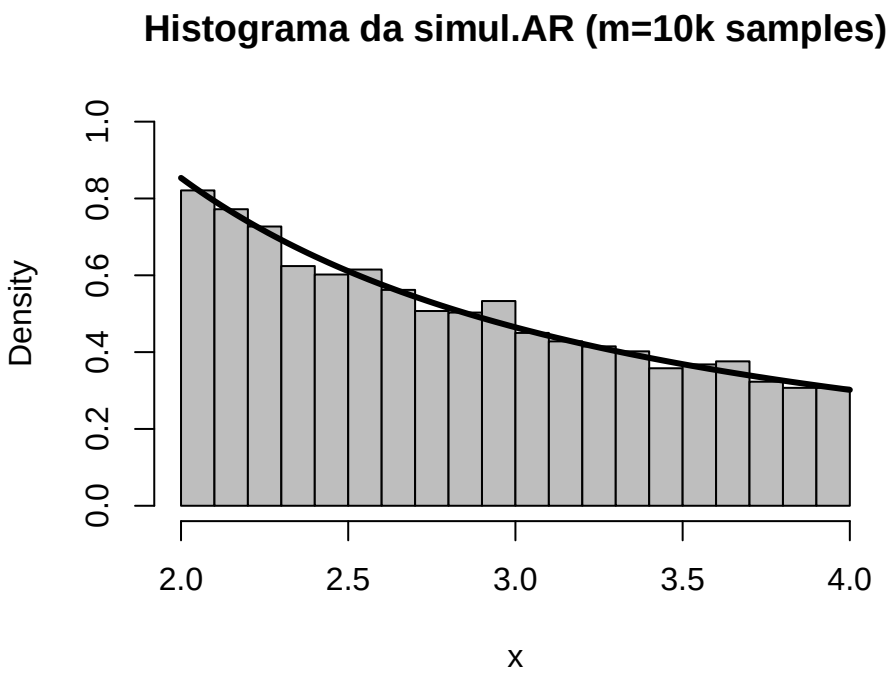
```
set.seed(123)

simulAR10000 = sim.AR(10000, alpha = 0.5, L = 2, H = 4)
first10.AR = (simulAR10000$x)[1:10]
first10.AR
```

```
##  [1] 3.880935 3.102870 2.492175 2.578319 2.049227 3.516919 2.636362 2.285600
##  [9] 2.827449 2.304889
```

De seguida, fazemos o plot do histograma gerado a partir do método de aceitação-rejeição com a curva da p.d.f sobreposta para verificarmos que, de facto, este método de geração de números aleatórios seguindo a distribuição desejada está correto e aproximado da distribuição desejada.

```
# fazer o histograma com os valores gerados a partir do
# algoritmo AR
hist(simulAR10000$x, freq = F, col = "grey", xlab = "x", ylim = c(0,
```

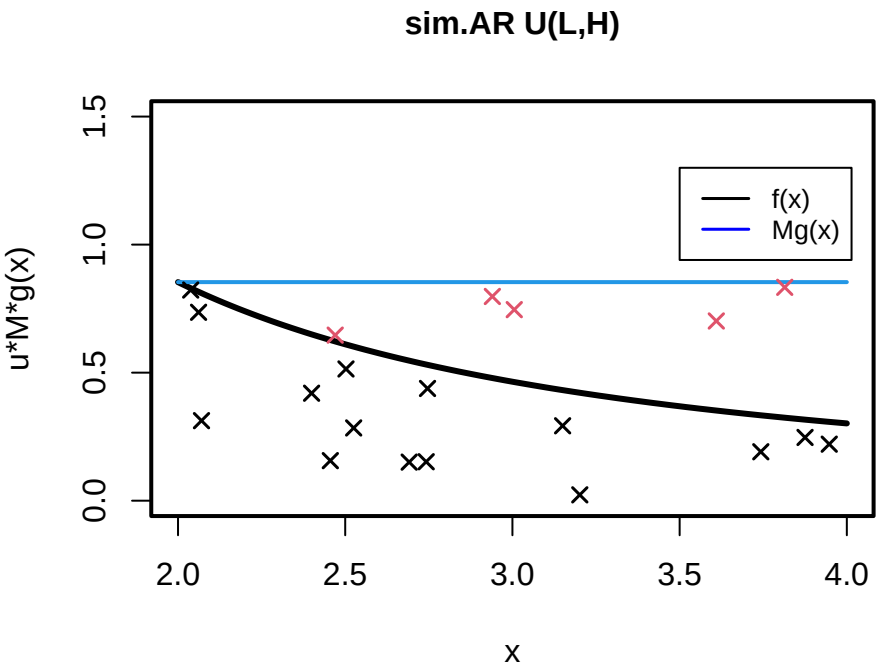
```
    1), main = "Histograma da simul.AR (m=10k samples)")

curve(truncatedParetoPDF(x, alpha = 0.5, L = 2, H = 4), lwd = 3,
    lty = 1, ylab = "u*M*g(x)", main = "trunc N(0,1)", cex.axis = 1,
    col = "black", cex.main = 1, cex.lab = 1, ylim = c(0, 1.5),
    xlim = c(2, 4), add = T, xlab = "x")
```



Finalmente faremos o plot da função p.d.f juntamente com a candidata bem como os pontos que correspondem às observacoes geradas (podendo estes ser aceites ou rejeitados). Para não sobrecarregar o gráfico iremos fazer apenas 15 observações como recomendado.

```
# fazer a simulacao para 15 pontos
simulAR = sim.AR(15, alpha = 0.5, L = 2, H = 4)

# primeiros 10 valores registados pela simulacao Cada ponto
# tem as coordenadas em x e y separadas nos vetores

# fazer o plot da curva candidata ja multiplicada pelo M
# para ficar acima da nossa funcao f
curve(truncatedParetoPDF(x, alpha = 0.5, L = 2, H = 4), lwd = 3,
    lty = 1, ylab = "u*M*g(x)", main = "sim.AR U(L,H)", cex.axis = 1,
    col = "black", cex.main = 1, cex.lab = 1, ylim = c(0, 1.5),
    xlim = c(2, 4))

M <- 0.85352
curve(candidataVezesM(x = x, M), add = T, lwd = 2, col = 4)
# fazer o plot dos pontos que foram aceites (debaixo da
# curva da nossa funcao)
points(simulAR$x, simulAR$yx, pch = 4, cex = 1, lwd = 1.5)

# fazer o plot dos pontos que foram rejeitados (Acima da
# curva da nossa funcao)
points(simulAR$rej_x, simulAR$yrej_x, col = 2, pch = 4, cex = 1,
    lwd = 1.5)
```

```r
# por uma borda mais bonita no grafico
box(lwd = 2)

legend(3.5, 1.3, legend = c("f(x)", "Mg(x)"), col = c("black",
    "blue"), lty = 1, cex = 0.8, lwd = 1.5)
```



Relativamente à taxa de rejeição das observações feitas, ou seja, o rácio de pontos que calharam acima da curva desejada (mas abaixo da função candidata) este é calculado a partir do número de pontos rejeitados sobre o número total de pontos:

```r
# Calcular a taxa de rejeicao, taxa de pontos que 'falharam
# o alvo', ou seja, ficaram fora da nossa zona de aceitao
# (acertaram acima da curva da funcao)

# nr de pontos rejeitados a divir pelo nr total de pontos
rej.rate <- length(simulAR10000$rej_x)/(length(simulAR10000$rej_x) +
    length(simulAR10000$x))
rej.rate
```

```
## [1] 0.408284
```

```r
# para 100%
percentagemErro = rej.rate * 100
cat(round(percentagemErro, 4), "%")
```

```
## 40.8284 %
```

g) Compare the computational times of routines sim.IT() and sim.AR() for generating a sample of size $m = 50000$ from the truncated Pareto distribution (you can use the R routine proc.time() as done in class – week 2).

Neste exercício pretendemos comparar os tempos de execução dos dois métodos (transformada inversa e aceitação-rejeição) para ver qual deles demora menos tempo a gerar m observações corretas. Para isto, utilizaremos a funcao *proc.time()* para contar os segundos entre o início e o final da execução da função que englobam. Como esperado, o método de aceitação-rejeição demora sempre mais tempo, principalmente

devido à função candidata escolhida. Isto é, como existe bastante "espaço" entre a p.d.f e a função candidata, uma boa parte das observações cairão na área em que estas serão rejeitadas. Deste modo, tem de realizar um maior número de tentativas para preencher o número de amostras m com observações válidas levando assim mais tempo a findar.

```
# Compare times of sim.IT and sim.AR
timeToProcessIT <- proc.time()
simIT50000 = sim.IT(50000, 0.5, 2, 4)
print(proc.time() - timeToProcessIT)
```

```
##    user  system elapsed
##    1.69    0.02    1.70
```

```
timeToProcessAR <- proc.time()
simAR50000 = sim.AR(50000, 0.5, 2, 4)
print(proc.time() - timeToProcessAR)
```

```
##    user  system elapsed
##    5.04    0.03    5.08
```

Exercício 2

Some random variables can be generated from the exponential distribution (exponential-based method), which we know how to obtain from the $U(0,1)$ distribution. Such is the case of the random variable $X \sim Gamma(\alpha, \theta)$:

$$\text{If } Y_i \overset{\sim}{iid} Exp(1) \text{ then } X = \theta \sum_{i=1}^{\alpha} Y_i \sim Gamma(\alpha, \theta), \alpha = 1, 2, ...$$

a) Implement the exponential-based method in R for generating a sample from $X \sim Gamma(\alpha, \theta)$, which has probability density function (p.d.f.)

$$f(x) = \frac{\theta^{-\alpha}}{\Gamma(\alpha)} e^{-\frac{x}{\theta}} x^{\alpha-1}, \qquad \alpha, \theta > 0, x \geq 0,$$

where Γ is the gamma function.

Call your routine sim.gam() and let it receive as input a generic sample size m and the Gamma distribution parameters α (note that here $\alpha \in \mathbb{N}$) and θ. Provide both algorithm and R code.

Seja X uma variável contínua com uma função densidade de probabilidade:

$$f(x) = \frac{\theta^{-\alpha}}{\Gamma(\alpha)} e^{-\frac{x}{\theta}} x^{\alpha-1}, \qquad \alpha, \theta > 0, x \geq 0,$$

Para gerar uma amostra aleatória de $X \sim Gamma(\alpha, \theta)$ através de uma distribuição normal $Y \sim Exp(1)$ é necessário obter observações aleatórias recorrendo à transformação inversa da mesma. Para isso, vamos gerar observações de uma distribuição uniforme $U \sim U(0,1)$.

Ora, a função densidade de probabilidade de $Y \sim Exp(1)$ é

$$f(x) = \lambda e^{-\lambda x}$$

Pelo que

$$F(x) : \int_0^x \lambda e^{-\lambda z} dz = (\frac{-\lambda e^{-\lambda z}}{\lambda}]_0^x = 1 - e^{-\lambda x}$$

Entâo, temos que:

$$u = 1 - e^{-\lambda x} \Rightarrow x = -\frac{1}{\lambda} log(1-u) = F^{-1}(u)$$

Desta forma, gerar m observações aleatórias da distribuição exponecial através da inversa transformada é dado pelo seguinte código R:

```
set.seed(654)

# Código para gerar m observações aleatórias da distribuição
# Normal
sim.exp = function(m, lambda) {
    x <- vector()
    for (i in 1:m) {
        u = runif(1, 0, 1)
        x = c(x, -log(u)/lambda)
    }
    x
}
```

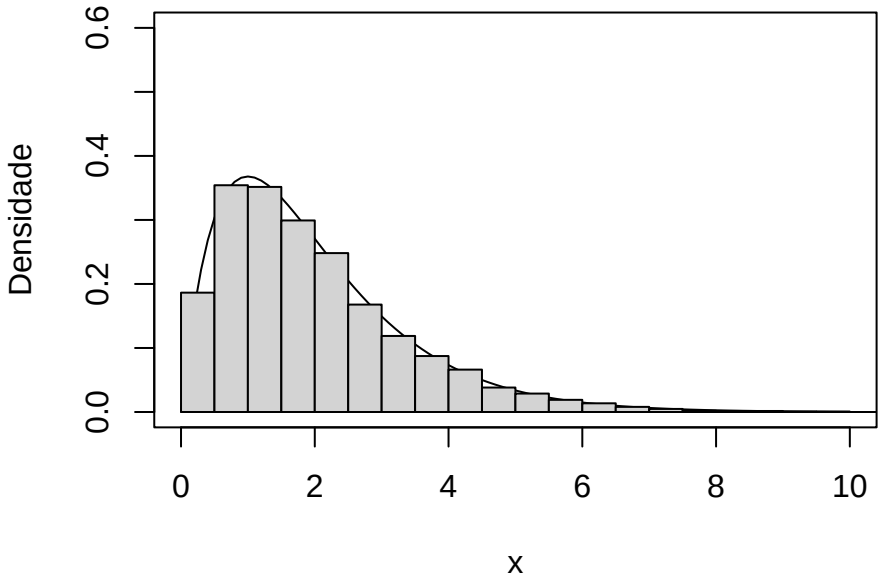
Para obter uma amostra de m observações de $X \sim Gamma(\alpha, \theta)$, utilizou-se o seguinte código a partir da seguinte expressão #(enuciado) (quem escreveu isto?)

```
# Código para gerar amostra aleatória de m observações de
# uma distribuição Gammma
sim.gam = function(m, alpha, theta) {
    x = vector()

    for (i in 1:m) {
        somatorio = 0
        for (j in 1:alpha) {
            obsExp = sim.exp(1, 1)
            somatorio = somatorio + obsExp
        }
        res = theta * somatorio
        x = c(x, res)
    }
    x
}
```

b) Use routine sim.gam() to generate a sample of size m = 10000 from $X \sim Gamma(2, 1)$. Plot the histogram with the pdf superimposed.

Para gerar uma amostra de $m = 10000$ a partir de $X \sim Gamma(2, 1)$ basta utilizar a função anteriormente descrita com os respetivos parâmetros, isto é:

```
# mostra-nos as 20 primeiras observações
simulacoesGamma10k = head(sim.gam(10000, 2, 1), 20)

# definição da função gamma(p.d.f)
functionExpGammaPDF = function(x, alpha, theta) {
    ((theta^(-alpha))/gamma(alpha)) * exp(-(x/theta)) * x^(alpha -
        1)
}

# curva
curve(functionExpGammaPDF(x = x, alpha = 2, theta = 1), ylim = c(0,
    0.6), xlim = c(0, 10), add = F, ylab = "Densidade")


# histograma
hist(sim.gam(10000, 2, 1), main = "10000 Observations - Gamma(2,1)",
    freq = F, add = T, breaks = 50)
```



Acima está representado o respetivo histograma.

Métodos de Monte Carlo

Os métodos de Monte Carlo são uma ampla classe de algoritmos computacionais que nos permitem realizar amostras aleatórias. Para este projecto utilizaremos os métodos de Integração e de Intervalos de Confiança.

Integração

Integração com Monte Carlo é uma técnica de integração numérica que usa números aleatórios. É um método de Monte Carlo especifico que calcula numericamente um integral definido. É utilizada em estatística para a estimação de integrais que se referem a caracteristicas de variáveis aleatórias.

Se tivermos que X é uma variável aleatória continua com uma função p.d.f $f(x)$ e $g(x)$: $\mathbb{R} \to \mathbb{R}$ é continua então $Y = g(X)$ é uma variável aleatória e

$$E[g(X)] = \int_a^b g(x)dx$$

Suponhamos, agora, que queremos integrar uma função unidimensional $g(x)$ no intervalo $[a, b]$.

$$I = \int_a^b g(x)$$

No entanto, suponhamos também que esta integração não apresenta uma solução analitica ou apresenta alto grau de dificuldade. Então, podemos estimar este integral através de amostragem aleatória utilizando o método de Monte Carlo. Para isto, basta-nos escrever o integral como expectativa de uma variável aleatória.

Assim, e admitindo que $b - a \neq 0$ reescrevemos o integral como

$$I = \int_a^b (b-a)\frac{1}{b-a}g(x)dx = (b-a)\int_a^b g(x)f(x) = (b-a)E[g(X)] \tag{7}$$

Considerando uma amosta aleatória $X_1..., X_m$ de uma população $X \sim U(a, b)$ e $g(X_1, ..., g(X_m))$ uma amostra aleatória de $g(x)$, temos que o estimador de Monte Carlo de I é :

$$I_{MC} = (b-a)\frac{1}{m}\sum_{i=1}^{m} g(Xi) \tag{8}$$

O que é uma estimativa não enviasada de I:

$$E[I_{MC}] = \frac{b-a}{m}\sum_{i=1}^{m} g(Xi) = E[g(X)] = \frac{(b-a)}{m}\sum_{i=1}^{m} E[g(X_i)] = (b-a)E[g(X)]$$

$$= (b-a)\int_a^b \frac{1}{b-a} g(x)dx$$

$$\int_a^b g(x)dx = I$$

Temos que a variância do estimador de Monte Carlo é:

$$\begin{aligned} V[I_{MC}] &= \left(\frac{(b-a)}{m}\right)^2 \sum_{i=1}^{m} V[g(Xi)] \\ &= \left(\frac{(b-a)}{m}\right)^2 mV[g(X)] \\ &= (b-a)^2\frac{V(g(X))}{m} \end{aligned} \tag{9}$$

Esta variância pode ser estimada retirando a variância de $g(x)$ da fórmula acima:

$$\widehat{V[I_{MC}]} = \frac{(b-a)^2}{m(m-1)}\sum_{i=1}^{m}(g(X_i) - \overline{g(X)})^2 \tag{10}$$

Da mesma forma:

$$SE(I_{MC}) = (b-a)\sqrt{\frac{V(g(X))}{m}} \tag{11}$$

Assim, o método de integração de Monte Carlo tem o seguinte algoritmo:

1. Gerar observações $X_1, ..., X_m$ a partir de $X \sim U(a, b)$.
2. Calcular $g(X_1), ..., g(X_m)$.
3. Calcular $\overline{g} = \frac{1}{m}\sum_{i=1}^{m} g(X_i)$.
4. Usar $(b-a)\overline{g}$ e estimar $V(I_{MC})$ ou $SE(I_{MC})$.

Técnicas de redução de variância

Quando reportamos estimadores de Monte Carlo de $I = E(g(X))$ é também importante reportar o tamanho da simulação m e um estimador do **erro padrão** do estimador de Monte Carlo, $\widehat{SE(I_{MC})}$.

Muitas vezes podemos correr simulações até que a variabilidade de um estimador seja menor que algum valor limite. No entanto, o custo computacional de o executar é muito alto.

Em geral, queremos que $SE(I_{MC})$ seja no máximo e, tal que $V(g(X)) = \sigma^2$, então $m \geq \lceil \sigma^2/e^2 \rceil$

Existem algumas técnicas que nos ajudam a reduzir a variância do estimador de Monte Carlo I_{MC} sem a necessidade de fazer muitas simulações.

Percentagem da redução da variância:

Sejam I_1, I_2 dois estimadores de I tal que $V(I_2) < V(I_1)$, então a redução da variância (em %) ao usarmos I_2 em vez de I_1 é:

$$100 \times \frac{V(I_1) - V(I_2)}{V(I_1)} = 100 \times \left(1 - \frac{V(I_2)}{V(I_1)}\right)$$

Variáveis Antitéticas

Duas variáveis dizem-se **antitéticas** se $X_1, X_2 \overset{\sim}{iid} X$ tal que $cov(X_1, X_2) < 0$.

Neste caso, considerando a nova variável $Y = \frac{X_1+X_2}{2}$ teremos:

$$V(Y) = \frac{1}{4}V(X_1 + X_2) = \frac{1}{4}(V(X_1) + V(X_2) + 2cov(X_1, X_2)) =$$
$$= \frac{1}{4}(2V(X) + 2cov(X_1, X_2)) = \frac{1}{2}(V(X) + cov(X_1, X_2)) \leq \frac{V(X)}{2}$$

Isto significa que a variância da nova variável Y é reduzida em pelo menos metade comparando com a variância de X_1 ou X_2.

Além disso,

$$E(Y) = E\left(\frac{X_1 + X_2}{2}\right) = \frac{1}{2}E(X_1 + X_2) = \frac{1}{2}(E(X_1) + E(X_2)) = E(X)$$

Seja $U \overset{\sim}{\texttt{iid}} U(0, 1)$.

Uma vez que,

- $U, 1 - U \overset{\sim}{iid} U(0, 1)$
- $\rho(U, 1 - U) = -1$

U e $1 - U$ são **variáveis antitéticas**.

Seja $U \sim U(0, 1)$ e $X \sim F(.)$ tal que $X = F^{-1}(U)$ (o X é gerado pelo ITM).

Se g é uma função **monótona contínua** então $g(F^{-1}(U))$ tem a mesma distribuição que $g(F^{-1}(1-U))$ e $\rho(g(F^{-1}(U)), g(F^{-1}(1-U))) < 0$.

Como $g(F^{-1}(U))$ e $g(F^{-1}(1-U))$ são igualmente distribuidas, então

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

Numa simulação Monte Carlo, vamos precisar de m simulações (m par) tal que:

$\rightarrow$ gere uma amostra $U_1, ..., U_{M/2}$ de $U(0, 1)$

$\rightarrow$ determine $g(F^{-1}(U_i))$ e $g(F^{-1}(1 - U_i))$ para todos $i = 1, ..., m/2$

$\rightarrow$ e que calcule

$$I_1 = \frac{1}{m/2}\sum_{i=1}^{m/2} g(F^{-1}(U_i)), \qquad \frac{1}{m/2}\sum_{i=1}^{m/2} g(F^{-1}(1 - U_i))$$

que são estimadores **não enviesados** de $E(g(X)) = I$, precisando assim apenas de metade das simulações para ter redução da variância.

Logo, o estimador Monte Carlo obtido é

$$I_{ant} = \frac{I_1 + I_2}{2} = \frac{2}{m}\sum_{i=1}^{m/2} \frac{g(F^{-1}(U_i)) + g(F^{-1}(1 - U_i))}{2}$$

que também é um estimador **não enviesado** de $I = E(g(X))$.

O algoritmo para estimar E(G(X)) é o seguinte:

1. Gerar uma observação y a partir de g.
2. Gerar uma observação $u_1, ..., u_{m/2}$ a partir de $U(0, 1)$, com m par.
3. Determinar $I_1 = \frac{1}{m/2}\sum_{i=1}^{m/2} g(F^{-1}(U_i))$ e $\frac{1}{m/2}\sum_{i=1}^{m/2} g(F^{-1}(1 - U_i))$.

4. Determinar $\hat{I}_{ant} = \frac{\hat{I}_1 + \hat{I}_2}{2}$, e reportar um estimador de $SE(I_{ant})$.

Quando $X = F^{-1}(U) \sim U(0,1)$ então o segundo passo do algoritmo passa a ser:

Determinar $I_1 = \frac{1}{m/2} \sum_{i=1}^{m/2} g(U_i)$ e $\frac{1}{m/2} \sum_{i=1}^{m/2} g(1 - U_i)$.

Variáveis de Controlo

Outra técnica para reduzir a variância do estimador Monte Carlo é usando variáveis de controlo. A ideia por detrás deste método é usar a mesma simulação Monte Carlo para estimar a expetativa "conhecida" de uma função g que é semelhante a f, cujo valor é o que estamos a tentar estimar. Quando g e f estão fortemente correlacionadas, podemos usar os erros aquando a estimação de g para corrigir a estimativa de f. Assim, temos dois critérios importantes na escolha da variável de controlo:

1. g deve ser uma função cuja expetativa seja conhecida (ou que possamos estimar com uma elevada precisão);
2. Deve estar correlacionada com a função f que estamos a tentar estimar.

Desta forma, para qualquer $c \in \mathbb{R}$

$$I_C = g(X) + c(h(X) - \mu) \tag{12}$$

Assim, conclui-se que a equação (12) é um estimador imparcial de I.

A variância do estimador escreve-se:

$$\begin{aligned} V(I_C) &= V(g(X)) + c(h(X)) - \mu \\ &= V(g(X) + c * h(X) - c * \mu) \\ &= V(g(X)) + c^2 * V(h(X)) + 2c * Cov(g(X), h(X)) \end{aligned}$$

que é a função quadrática em c que tem mínimo em $c = c*$, com:

$$c* = -\frac{Cov(g(X), h(X))}{V(h(X))}$$

Para este valor de c a variância do estimador é :

$$V(I_{C*}) = V(g(X)) - \frac{Cov(g(X), h(X))}{V(h(X))} = V(g(X))(1 - \rho^2)$$

sendo $\rho = cor(g(X), h(X))$

O estimador de Monte Carlo, I, utilizando o método de variáveis de controlo, é dado por:

$$I_{cont} = \frac{1}{m} \sum_{i=1}^{m} (g(X_i) + c * (h(X_i) - \mu)) \tag{13}$$

Assim, tem-se que

$$E(I_{cont}) = I$$

e

$$V(I_{cont}) = \frac{V(l_{C*})}{m} = \frac{V(g(X))(1 - \rho^2)}{m}. \tag{14}$$

A percentagem de redução da variância alcançada ao usar o estimador MC com variáveis de controlo (I_{cont}), ao invés do estimador Naive MC (I_{mc}), é dado por:

$$100 * \frac{V(I_{mc}) - V(I_{cont})}{V(I_{mc})} = 100\rho^2 \tag{15}$$

Importance Sampling

Vamos assumir que queremos estimar o integral

$$I = \int g(x)f(x)dx = E(g(X)),$$

onde $X \sim f$.

Se ϕ for uma função de densidade de probabilidade positiva podemos reescrever o integral como:

$$I = \int_0^1 g(x)\frac{f(x)}{\phi(x)}\phi(x)dx = \int_0^1 h(x)\phi(x)dx = E_\phi(h(X)),\ com\ X \sim \phi$$

ϕ é chamado de função de importância e é uma função positiva, pelo menos quando $g(x)f(x) \neq 0$. Se $\phi(x)$ for escolhida para que o rácio de $h(x)$ tenha uma variância baixa, então este método resulta num estimador eficiente de I.

O estimador de $I = E_\phi(h(X))$, através de importance sampling será

$$I_{IS} = \frac{1}{m}\sum_{i=1}^{m} h(X_i),$$

onde $X_1, ..., X_m$ é uma amostra aleatória de ϕ.

A escolha da **função de importância** ϕ deve seguir estes critérios:

$\rightarrow$Tem de ser fácil de simular a partir de ϕ

$\rightarrow Var(I_{IS}) = \frac{X_\phi(h(X))}{m} < Var(I_{MC})$

Vamos considerar ϕ a *tilted density* de f, comummente usada

$$\phi(x) = \frac{e^{tx}f(x)}{M(t)}$$

em que $M(t) = E_f(e^{tX})$ e t será o valor que melhor aproxima ϕ da forma de $|g(x)|f(x)$.

Para $X \sim U(0,1)$, em que a p.d.f. é $f(x) = 1, \quad x \in [0,1]$, então

$$\phi(X) = \frac{e^{tx}}{M(t)},$$

com

$$M(t) = E_f(e^{tX}) = \int_0^1 e^{xt}dx = \frac{e^t - 1}{t}, \quad t \neq 0$$

i.e.,

$$\phi(x) = \frac{te^{tx}}{e^t - 1}, \quad t \neq 0. \tag{16}$$

Exercício 3

Let

$$I = \int_0^1 \frac{e^{-x}}{1+x^2}dx$$

a) Use the R function integrate() to compute the value of I.

Nesta alínea calculou-se o valor do integral dado no enunciado. Para isto, utilizou-se a função *integrate()* do software R.

```
funcaoToIntegrate = function(x) {
    exp(-x)/(1 + x^2)
}
integrate(funcaoToIntegrate, 0, 1)
```

```
## 0.5247971 with absolute error < 5.8e-15
```

b) Describe and implement in R the Monte Carlo method for estimating I (use size $m = 10000$). Report an estimate of the variance of the Monte Carlo estimator $\hat{I}_{MC}$ of I.

Para estimarmos o integral em cima exposto usando uma amostra aleatória reescreveremos o integral como a "expectativa" da amostra aleatória. Assim, considerando a equação (7) e o facto de $X \sim U(0,1)$ podemos reescrever o integral como

$$(1-0)E(g(X)).$$

Pela equação (8) temos que o estimador Monte Carlo de I fica:

$$I_{MC} = (1-0)\frac{1}{10000}\sum_{i=1}^{m} g(Xi)$$

Pela equação (9) a variância do estimador é dada por:

$$(1-0)^2\frac{V(g(X))}{10000}.$$

```
m = 10000

# Gerar uma amostra m observações da Uniforme (0,1)
obsNorm = runif(m, 0, 1)

# Calculo da estimativa do integral a partir de uma amostra
# aleatória
integralMC = (1 - 0) * mean(funcaoToIntegrate(obsNorm))
# 0.5245013

varianceIntegr = var(funcaoToIntegrate(obsNorm))/(m * (1 - 0)^2)
varianceIntegr
```

```
## [1] 6.073905e-06
```

Deste modo temos:

$$E[I_{MC}] \approx 0.5245$$
$$Var[I_{MC}] \approx 6.07e{-}6$$

c) Describe and implement in R the Monte Carlo methods of based on antithetic variables, control variables and importance sampling for estimating I (size $m = 10000$). Report an estimate of the variance of all the Monte Carlo estimators $\hat{I}_{ant}, \hat{I}_{cont}$ and $\hat{I}_{is}$ of I.

Variáveis Antitéticas

Iremos começar por estimar o valor do integral baseando-nos nas variáveis antitéticas. Para estas, seguindo a teoria explicada no início da secção, teríamos de utilizar o método IT porque necessitaríamos de $F^{-1}(U)$. No entanto, isto apenas se aplica a integrais que não tenham como suporte [0,1]. Assim, poderemos simplificar este passo do algoritmo e usar diretamente a função dada.

```
## Antithetic Variables


x = runif((m/2), 0, 1)
I.hat1 = mean(funcaoToIntegrate(x))
I.hat2 = mean(funcaoToIntegrate(1 - x))
I.hat.a = (I.hat1 + I.hat2)/2
# Icont
I.hat.a
```

```
## [1] 0.5249099
```

```
V.a <- 1/m * (1 + cor(funcaoToIntegrate(x), funcaoToIntegrate(1 -
    x))) * var(funcaoToIntegrate(x))
V.a
```

```
## [1] 2.214422e-07
```

```
# Percentage variance reduction Icont vs Imc
percentageVRIAnti = 100 * (1 - V.a/varianceIntegr)
percentageVRIAnti
```

```
## [1] 96.3542
```

Com isto, temos que primeiro realizar $m_0 = m/2$ observações aleatórias de uma distribuição $U(0,1)$. Para isto, m tem de ser par. De seguida, iremos calcular I_1 e I_2, em que $I_1 = \frac{1}{m_0}\sum_{i=1}^{m_0} g(U_i)$ e $I_2 = \frac{1}{m_0}\sum_{i=1}^{m_0} g(1-U_i)$.

Depois de calculados estes dois valores, o valor estimado será calculado como

$$I_{ant} = \frac{I1 + I2}{2}.$$

Neste caso temos que

$$I_{ant} = 0.5247112.$$

Relativamente à variância estimada, esta será calculada como

$$V(I_{ant}) = \frac{1}{m}\left(1 + \rho\left(g(u), g(1-u)\right)\right) V(g(X)).$$

Portanto, temos o valor da variância de

$$V(I_{ant}) = 2.21e-07.$$

Variáveis de Controlo

Usando o método das variáveis de controlo e tendo em conta o explicado, começaremos por gerar m observações de uma distribuição $U(0,1)$ para poder calcular o nosso c^*. Este valor corresponde ao coeficiente óptimo para usarmos na estimação propriamente dita, usando o método das variáveis de controlo. Este coeficiente é óptimo na medida em que reduz a variância sobre o que queremos calcular.

Pelo que foi visto anteriormente temos:

```
## Control Variables
g = function(x) {
    exp(x)
}
u1 = runif(m)
cast = -(cov(u1, funcaoToIntegrate(u1)))/var(u1)
cast
```

```
## [1] 0.8396287
```

Desta forma, temos:

$$c* = 0.84$$

Agora, tendo o melhor coeficiente calculado, de modo a reduzir a variância iremos fazer a implementação da abordagem explicada acima. Uma vez que a nossa função é uma exponencial, faz sentido a variável de controlo ser $h(X) = X$. Como podemos verificar pelo output deste excerto de código, o valor estimado é praticamente igual ao valor verdadeiro tendo apenas um valor para a variância muito baixo.

Pelas equações (13), (14) e (15) temos:

```r
# control-based MC nr observacoes
m = 10000
# variavel de controlo
h <- function(x) {
    x
}
x = runif(m, 0, 1)

# Icont
I.hat.c.mc = mean(funcaoToIntegrate(x) + cast * (h(x) - 0.5))
I.hat.c.mc
```

```
## [1] 0.5240668
```

```r
V.c.mc <- var(funcaoToIntegrate(x)) * (1 - cor(funcaoToIntegrate(x),
    h(x))^2)/m
V.c.mc
```

```
## [1] 1.09912e-07
```

```r
# Percentage variance reduction Icont vs Imc
percentageVRICont = 100 * (1 - V.c.mc/varianceIntegr)
percentageVRICont
```

```
## [1] 98.19042
```

Assim,

$$I_{cont} = 0.5241$$
$$V(I_{cont}) = 1.0991e{-}7$$
$$Reduction = 98.1904\%$$

Importance Sampling

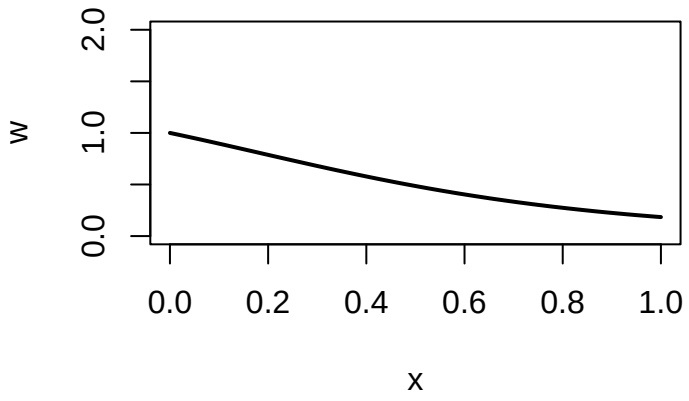
Pelo enunciado temos que

$$I = \int_0^1 \frac{e^{-x}}{1+x^2} dx = \int_0^1 g(x)f(x)dx = E(g(X))$$

em que $X \sim f$.

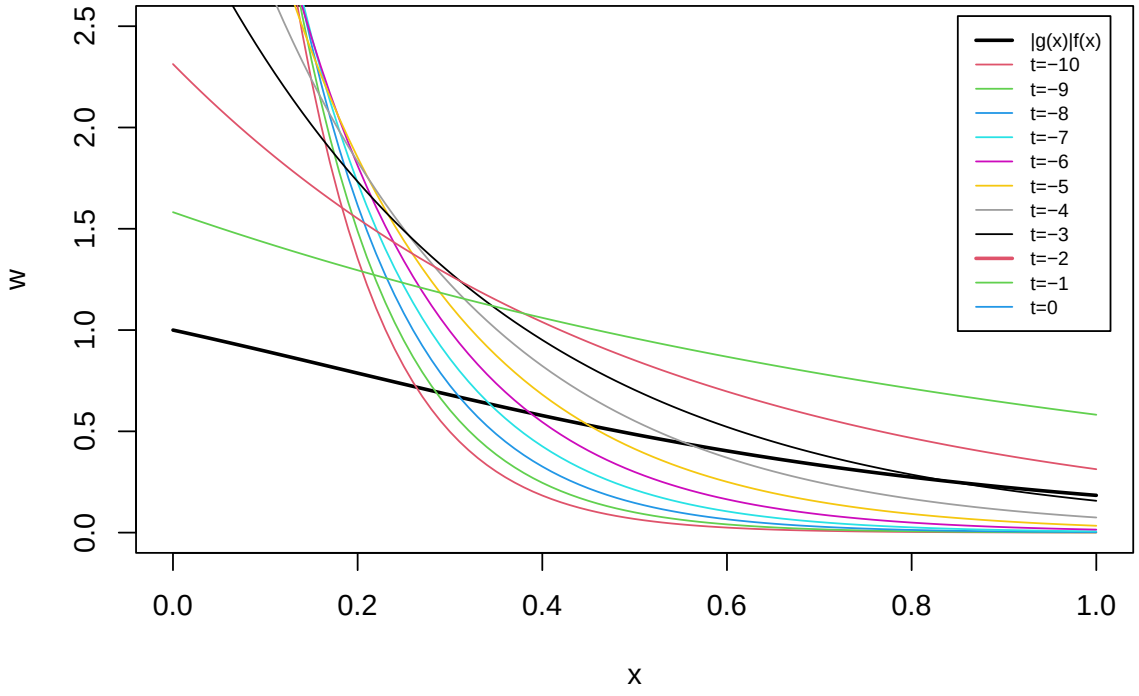
Para a escolha do valor t, na equação (16), iremos primeiro visualizar a função resultante de $|g(x)|f(x)$ a que chamaremos de $w(x)$.

```r
f = function(x) {
    1
}
w = function(x) {
    abs(funcaoToIntegrate(x)) * f(x)
}
plot(w, lwd = 2, ylim = c(0, 2))
```



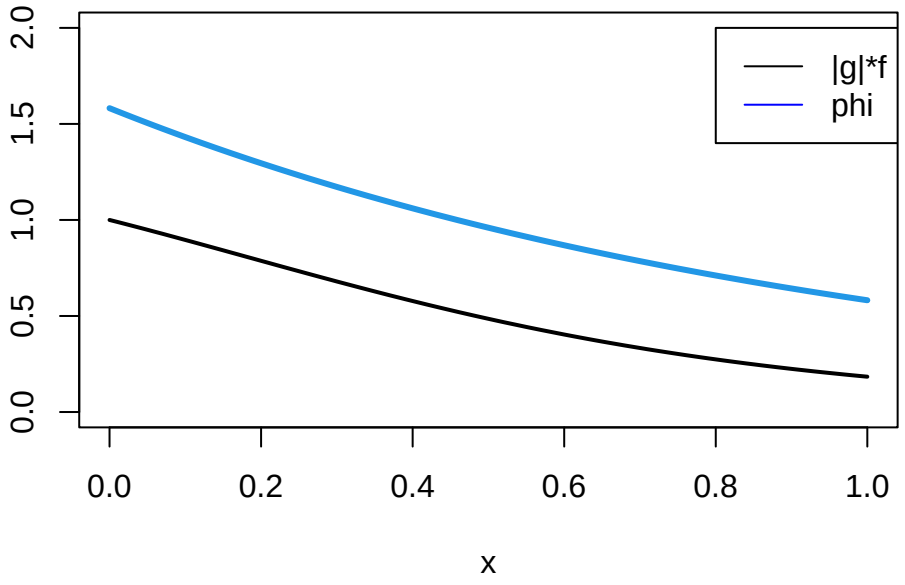
Visto que a função é estritamente decrescente iremos representar ϕ com valores de t a variar entre -10 e 0 para verificarmos qual a melhor curva se aproxima da forma de w.

```
plot(w, lwd = 2, ylim = c(0, 2.5))
currentColor = 2
for (t in seq(-10, 0, 1)) {
    phi = function(x) {
        t * exp(t * x)/(exp(t) - 1)
    }
    curve(phi, add = T, col = currentColor)
    currentColor = currentColor + 1
}

legend(0.85, 2.55, col = 1:8, lwd = c(2, rep(1, 8)), cex = 0.7,
    legend = c("|g(x)|f(x)", paste0("t=", seq(-10, 0, 1))))
```



Pelo plot feito, percebemos que das curvas geradas com os diferentes t, a que mais se assemelha à curva de w é $t = -1$. Agora, iremos apenas fazer o plot destas duas curvas para observar melhor a sua semelhança.

```
# com t = -1
phi = function(x) {
    -1 * exp(-1 * x)/(exp(-1) - 1)
}

plot(w, xlab = "x", ylab = "", ylim = c(0, 2), lwd = 2)
curve(phi, add = T, col = 4, lwd = 3)
legend(0.8, 2, legend = c("|g|*f", "phi"), col = c("black", "blue"),
    lty = 1)
```



Iremos agora calcular o x, para usar no algoritmo como explicado na parte teórica. Primeiramente, iremos substituir $t = -1$, o valor de t escolhido, na função ϕ

$$\phi(x) = \frac{-e^{-x}}{e^{-1} - 1}$$

Como sabemos que $x = F_{\phi}^{-1}(u)$ com $u \sim U(0,1)$ pelo método de IT explicado na primeira parte do trabalho, obteremos m amostras de u e, por conseguinte, m amostras de x.

Cálculo da integral:

$$\int_0^x \phi(z)dz = \int_0^x \frac{-e^{-x}}{e^{-1} - 1}dz = \frac{e^{-x} - 1}{e^{-1} - 1}$$

Cálculo da inversa:

$$\frac{e^{-x} - 1}{e^{-1} - 1} = u \Leftrightarrow e^{-x} = u(e^{-1} - 1) + 1 \Leftrightarrow x = -ln(ue^{-1} - u + 1)$$

```
m = 10000
## Importance Sampling

u = runif(m, 0, 1)
x = -log(u * exp(-1) - u + 1)
h = function(x) {
    funcaoToIntegrate(x) * f(x)/phi(x)
}


I.IS = mean(h(x))
I.IS
```

```
## [1] 0.5247023
```

```
var.I.IS = var(h(x))/m
var.I.IS
```

```
## [1] 9.386998e-07
```

	I_MC	I_Ant	I_Cont	I_IS
Estimativa SE	6.07e-6	2.21e-7	1.09e-7	9.39e-7
% Redução de Variância	-	96.3%	98.2%	84.5%

```
percentageVRIImp = 100 * (1 - var.I.IS/varianceIntegr)
percentageVRIImp
```

```
## [1] 84.54537
```

d) What's the percentage of variance reduction that is achieved when using those MC estimators instead of $\hat{I}_{MC}$?

Agora, tendo usado estes 3 métodos de estimação de Monte Carlo, iremos calcular as percentagens de redução de variância ao usarmos estes estimadores comparativamente ao primeiro.

Analisando a tabela no início da página, contendo esta os valores para as variâncias dos vários estimadores bem como a percentagem de redução de todos eles comparados ao Naive Monte Carlo, podemos verificar que, para todos os métodos, variáveis antitéticas, variáveis de controlo e *Importance Sampling*, estes estimadores tiveram uma redução da variância de 96.3% e 98.2% e 84.5% respetivamente. Em todos os métodos, houve uma redução da variância comparativamente ao original. Deste modo, o método mais favorável para fazer a estimação é o das variáveis de controlo, que resultou na maior diminuição da variância.

Intervalos de Confiança

Como é do nosso conhecimento, se $\mathbf{X} = (X_1, ..., X_N)$ é uma amostra aleatória de uma população X, com valor médio μ desconhecido e variância $\theta^2 < \infty$, então

$$Pr(\mu_L(X) < \mu < \mu_U(X)) = 1 - \alpha,$$

é um intervalo de confiança para μ, a $(1-\alpha)100\%$.

Se considerarmos uma observação $\mathbf{x}$ de $\mathbf{X}$, então dizemos que o intervalo de confiança $CI(X) = (\hat{\mu}_L(X), \hat{\mu}_U(X))$ contém μ $(1-\alpha)100\%$ das vezes. A esta proporção de vezes que o intervalo de confiança contém o verdadeiro valor de μ chamamos Probabilidade de Cobertura (coverage probability).

Assim, para determinar se de facto a probabilidade de cobertura é igual a $(1-\alpha)$ devemos calcular o Nível de Confiança Empírico, isto é, devemos gerar bastantes amostras da população, calcular os intervalos de confiança correspondentes e, por fim, a proporção de intervalos que contêm μ.

Para tal, apresentamos o algoritmo de simulação de Monte Carlo:

1. Gerar m conjuntos de observações $x_1, ..., x_n$ da população X, ou seja,

$$x_1^1, ..., x_n^1 \quad , ..., \quad x_1^m, ..., x_n^m$$

2. Calcular $CI_1(x), ..., CI_m(x)$, relativos a cada conjunto de observações, para um determinado nível de confiança α

3. Determinar o nível de confiança empírico, dado por

$$t_{MC} = \frac{1}{m} \sum_{j=1}^{m} t_j,$$

onde t_j representa a pertença (1), ou não (0), de μ no intervalo j.

4. Estimar o erro padrão de T_{MC}:

$$\mathbf{SE}(\hat{T}_{MC}) = \sqrt{\frac{t_{MC}(1 - t_{MC})}{m}}.$$

Exercício 4

4. Assume $X \sim N(,\sigma^2)$ with μ unknown and σ^2 known. Let $X_1, ..., X_n$ be a random sample of population X. One has that the random interval given by

$$\left[\bar{X} - z_{1-\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}, \bar{X} + z_{1-\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}\right]$$

is a $(1-\alpha)\times 100$ confidence interval (CI) for μ regardless of sample size n. In this exercise, one wishes to assess how a certain type of data contamination affects the coverage probability of a 95% CI for , given a random sample of a population $X \sim N(,1)$ of size n = 20, via a Monte Carlo simulation study (with $m = 10000$ simulations). For that purpose, consider that 90% of those n observations are drawn from a $X \sim N(0,1)$ distribution and that the remaining 10% are drawn from the contaminated normal distribution $X \sim N(k,1)$ with $k = 1, 5, 9$. The bad data points generated in this way are called shift outliers because the bad data points are shifted from μ = 0 in k standard deviations. Present a thorough discussion of your results. How harmful can this type of contamination be in terms of the level of confidence of a CI? Would your results still hold for other levels of confidence other than 95%?

Neste exercício vamos aplicar o método de Monte Carlo para intervalos de confiança, porém vamos comparar o comportamento de amostras sem e com contaminação.

Primeiramente, começámos por determinar o nível de confiança empírico para uma amostra, sem contaminação, de uma população $X \sim N(0,1)$:

```
# intervalo de confiança sem contaminação ou seja, todas as
# n=20 são observ. provêm de X~N(0,1)

# uma vez que sigma é conhecido, o nosso i.c. será na forma
# x_bar +- z_{a/2}*(sigma/sqrt(n))

# fixar a semente e alguns parametros
set.seed(569)
n = 20
m = 10000
mu = 0
sigma = 1
alpha = 0.05
z = 1.96
ci_lower = ci_upper = numeric(m)

# gerar amostras aleatorias e determinar os limites
# inferior e superior do i.c.
for (i in 1:m) {
    x = rnorm(n, mu, sigma)
    x_bar = mean(x)
    ci_lower[i] = x_bar - z * (sigma/sqrt(n))
    ci_upper[i] = x_bar + z * (sigma/sqrt(n))
}

# nivel de confiança empirico
nce = mean(ci_lower <= mu & ci_upper >= mu)
nce
```

```
## [1] 0.9516
```

```
# analisar se o nce diverge significativamente dos 95% de
# 'cobertura'
binom.test(nce * m, m, p = 0.95)$p.value
```

```
## [1] 0.4769661
```

```r
# 0.3091904

# estimativa do SE do estimador de monte carlo
sqrt(nce * (1 - nce)/m)
```

```
## [1] 0.0021461
```

Posteriormente, considerámos uma amostra com contaminação. Neste caso, 10% das observações são retiradas de $X \sim N(k, 1), k = 1, 5, 9$. Abaixo apresentamos o código R para $k = 1$.

```r
# k=1
n1 = 18
n2 = 2  #90% e 10% das observaçoes respet.
ci_lower = ci_upper = numeric(m)

# gerar amostras aleatorias e determinar os limites
# inferior e superior do i.c.
for (i in 1:m) {
    x1 = rnorm(n1, mu, sigma)
    x2 = rnorm(n2, 1, sigma)  #parte da amostra contaminada
    x <- c(x1, x2)  #amostra com contaminaçao
    x_bar = mean(x)
    ci_lower[i] = x_bar - z * (sigma/sqrt(n))
    ci_upper[i] = x_bar + z * (sigma/sqrt(n))
}

# nivel de confiança empirico
nce = mean(ci_lower <= mu & ci_upper >= mu)
nce
```

```
## [1] 0.9267
```

```r
# analisar se o nce diverge significativamente dos 95% de
# 'cobertura'

binom.test(nce * m, m, p = 0.95)$p.value
```

```
## [1] 1.056206e-23
```

```r
# estimativa do SE do estimador de monte carlo
sqrt(nce * (1 - nce)/m)
```

```
## [1] 0.002606283
```

Para $k = 5$ e $k = 9$ temos os seguintes resultados, respetivamente:

```r
# k=5
ci_lower = ci_upper = numeric(m)

# gerar amostras aleatorias e determinar os limites
# inferior e superior do i.c.
for (i in 1:m) {
    x1 = rnorm(n1, mu, sigma)
    x2 = rnorm(n2, 5, sigma)  #parte da amostra contaminada
    x <- c(x1, x2)  #amostra com contaminaçao
    x_bar = mean(x)
    ci_lower[i] = x_bar - z * (sigma/sqrt(n))
    ci_upper[i] = x_bar + z * (sigma/sqrt(n))
}

# nivel de confiança empirico
nce = mean(ci_lower <= mu & ci_upper >= mu)
```

```r
nce
```

```
## [1] 0.3916
```

```r
# analisar se o nce diverge significativamente dos 95% de
# 'cobertura'

binom.test(nce * m, m, p = 0.95)$p.value
```

```
## [1] 0
```

```r
# estimativa do SE do estimador de monte carlo
sqrt(nce * (1 - nce)/m)
```

```
## [1] 0.00488108
```

```r
# k=9
ci_lower = ci_upper = numeric(m)

# gerar amostras aleatorias e determinar os limites
# inferior e superior do i.c.

for (i in 1:m) {
    x1 = rnorm(n1, mu, sigma)
    x2 = rnorm(n2, 9, sigma)  #parte da amostra contaminada
    x <- c(x1, x2)  #amostra com contaminaçao
    x_bar = mean(x)
    ci_lower[i] = x_bar - z * (sigma/sqrt(n))
    ci_upper[i] = x_bar + z * (sigma/sqrt(n))
}

# nivel de confiança empirico
nce = mean(ci_lower <= mu & ci_upper >= mu)
nce
```

```
## [1] 0.0166
```

```r
# analisar se o nce diverge significativamente dos 95% de
# 'cobertura'

binom.test(nce * m, m, p = 0.95)$p.value
```

```
## [1] 0
```

```r
# estimativa do SE do estimador de monte carlo
sqrt(nce * (1 - nce)/m)
```

```
## [1] 0.001277671
```

Assim, podemos concluir que quanto maior for a contaminação na nossa amostra menor é o nível de confiança empírico. Uma vez que todos os p-valores do teste binomial nas amostras com contaminação são inferiores aos níveis usuais (≤ 0.05), leva-nos a rejeitar a hipótese nula, isto é, podemos assumir que o nível de confiança empírico diverge significativamente dos 95% de "cobertura". Note-se que no caso sem contaminação o p-valor é superior a 0.05, o que corresponde a não rejeitar a hipótese nula.

Referências

[1] Ross, S. (2010). *A first course in probability.* 8-th Edition. Prentice Hall.

[2] Lourenço, V. M. (2021). *Computational Numerical Statistics, slides week2.* First Semester - FCT-UNL