# Introdução ao programa R

*Ronald Targino, DEMA-UFC*

*Notas de aula*

## 4. Construir funções

```
# Antes de construirmos funções, vejamos algumas funções predefinidas e
# informações importantes sobre elas. Tomaremos como exemplos as funções
# combn, sample e barplot.

# Vamos gerar todas as combinações dos elementos de 3, 4 e 5 tomados 2 a 2
combn(c(3, 4, 5), 2)
```

```
##      [,1] [,2] [,3]
## [1,]    3    3    4
## [2,]    4    5    5
```

```
m1 = combn(c(3, 4, 5), 2)  # a função combn retorna um array (matriz) ou uma lista
m1  # m1 é uma matriz
```

```
##      [,1] [,2] [,3]
## [1,]    3    3    4
## [2,]    4    5    5
```

```
m2 = combn(c(3, 4, 5), 2, simplify = FALSE)
m2  # m2 é uma lista
```

```
## [[1]]
## [1] 3 4
##
## [[2]]
## [1] 3 5
##
## [[3]]
## [1] 4 5
```

```
# argumentos da função combn: x: vetor de elementos para a combinação; m:
# números de elementos na combinação; FUN: função a ser aplicada a cada
# combinação; simplify: instrução lógica para retornar matriz (TRUE é o
# padrão) ou lista (FALSE)
args(combn)
```

```
## function (x, m, FUN = NULL, simplify = TRUE, ...)
## NULL
```

```
# Somando os valores de cada combinação
combn(c(3, 4, 5), 2, FUN = sum)
```

```
## [1] 7 8 9
```

```
# Calculando a média de cada combinação
combn(c(3, 4, 5), 2, FUN = mean)
```

```
## [1] 3.5 4.0 4.5
```

```r
# Vamos gerar uma amostra aleatória de tamanho 20 de uma população formada
# por 30% de fumantes (F) e 70% de não fumantes (NF). Como a amostra é
# 'aleatória', não necessariamente você obterá a mesma amostra abaixo.
sample(c("F", "NF"), 20, replace = TRUE, prob = c(0.3, 0.7))
```

```
##  [1] "NF" "NF" "NF" "F"  "NF" "NF" "F"  "NF" "NF" "NF" "F"  "F"  "F"  "F"
## [15] "NF" "NF" "NF" "NF" "NF" "NF"
```

```r
# Argumentos da função sample
args(sample)
```

```
## function (x, size, replace = FALSE, prob = NULL)
## NULL
```

```r
# Não é necessário que os argumentos sejam colocados na sequência indicada,
# basta que sejam identificados. Veja os exemplos a seguir.
```

```r
sample(c("F", "NF"), 20, prob = c(0.3, 0.7), replace = TRUE)
```

```
##  [1] "F"  "NF" "NF" "NF" "NF" "NF" "NF" "NF" "F"  "NF" "NF" "F"  "NF" "NF"
## [15] "NF" "F"  "NF" "NF" "NF" "F"
```

```r
sample(replace = TRUE, c(0.3, 0.7), size = 20, x = c("F", "NF"))
```

```
##  [1] "F"  "NF" "F"  "NF" "NF" "NF" "F"  "NF" "NF" "F"  "NF" "F"  "NF" "NF"
## [15] "NF" "NF" "F"  "NF" "NF" "NF"
```

```r
sample(prob = c(0.3, 0.7), replace = TRUE, size = 20, x = c("F", "NF"))
```

```
##  [1] "NF" "NF" "F"  "F"  "F"  "NF" "NF" "NF" "NF" "F"  "NF" "F"  "F"  "F"
## [15] "NF" "NF" "NF" "F"  "NF" "NF"
```

```r
# Tabela de frequência para o objeto dados. Gere várias vezes as duas
# instruções abaixo e verifique que as frequências mudam entre as amostras.
dados = sample(c("F", "NF"), 1000, replace = TRUE, prob = c(0.3, 0.7))
table(dados)
```

```
## dados
##   F  NF
## 282 718
```

```r
# Agora, vamos construir nossas próprias funções

# função f1 sem argumentos. Essa função apenas retorna a string 'Minha
# primeira função!'
f1 = function() {
    cat("Minha primeira função!")
}
f1()  # chamada da função f1
```

```
## Minha primeira função!
```

```r
# Observe a diferença nos padrões das funções cat e print e do efeito de \n
# na string.
f2 = function() {
    cat("Minha primeira função! ")
    cat("Minha primeira função! ")
    cat("Minha primeira função! \n")
    print("Minha primeira função! ")
```

```
    print("Minha primeira função! ")
    print("Minha primeira função! \n")
}
f2()
```

```
## Minha primeira função! Minha primeira função! Minha primeira função!
## [1] "Minha primeira função! "
## [1] "Minha primeira função! "
## [1] "Minha primeira função! \n"
```

```
# função com um argumento (objeto 'aux'). Havendo mais de uma instrução
# entre as chaves {}, a função retornará a última instrução. No exemplo
# abaixo, o retorno será o resultado da função cat.
f3 = function(aux) {
    res = aux^2
    cat("O quadrado de ", aux, " é ", res, ".", sep = "")
}
f3(3)
```

```
## O quadrado de 3 é 9.
```

```
# função com um argumento com valor padrão (objeto 'num')
f4 = function(num = 3) {
    for (i in 1:num) {
        cat("Minha primeira função!\n")
    }
}
f4()
```

```
## Minha primeira função!
## Minha primeira função!
## Minha primeira função!
```

```
f4(1)
```

```
## Minha primeira função!
```

```
f4(4)
```

```
## Minha primeira função!
## Minha primeira função!
## Minha primeira função!
## Minha primeira função!
```

```
f4(6)
```

```
## Minha primeira função!
## Minha primeira função!
## Minha primeira função!
## Minha primeira função!
## Minha primeira função!
## Minha primeira função!
```

```
# função com um argumento (objeto 'aux')
f5 = function(aux) {
    for (i in 1:(aux + 2)) {
        cat(i + 10)
        cat(". Minha primeira função!\n")
```

```
    }
}
f5(3)
```

```
## 11. Minha primeira função!
## 12. Minha primeira função!
## 13. Minha primeira função!
## 14. Minha primeira função!
## 15. Minha primeira função!
```

```
# função para calcular a potência de um número 'n'
f6 = function(n) {
    n^2
}
f6(9)
```

```
## [1] 81
```

```
# f7 é a mesma função f6. As chaves {} não são necessárias quando apenas uma
# instrução é passada.
f7 = function(n) n^2
f7(4)
```

```
## [1] 16
```

```
# função para calcular a potência e a raiz quadrada de um número 'n'
f8 = function(n) {
    c(n^2, sqrt(n))  # a função retorna um vetor contendo a potência e a raiz quadrada do valor n
}
f8(9)
```

```
## [1] 81  3
```

```
# função usando objeto não definido nos argumentos da função
f9 = function(x, y) {
    x^2 + (y/w)
}
w = 3
f9(2, 3)
```

```
## [1] 5
```

```
# função que retorna uma matriz quadrada com elementos dados pelo produto do
# número da linha pelo número da coluna
g1 = function(dimen) {
    M = matrix(0, dimen, dimen)
    for (i in 1:dimen) {
        for (j in 1:dimen) {
            M[i, j] = i * j
        }
    }
    return(M)
}
g1(2)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    2    4
```

```r
# função dentro de função
g2 = function(y) {
    y^2
}
g3 = function(x, y) {
    x^2 + g2(y)
}
g3(2, 3)
```

```
## [1] 13
```

```r
g3(3, 5)
```

```
## [1] 34
```

```r
g4 = function(x, type) {
    switch(type, media = mean(x), mediana = median(x), `desvio padrão` = sd(x),
        gráfico = hist(x, freq = FALSE, main = "Histograma", xlab = "Peso",
            ylab = "densidade"))
}

dados = c(4.2, 5.9, 4.7, 4.8, 6.3, 5.9, 5.6, 4.6, 5.3, 5.4, 4.8, 7, 5.6, 5.7,
    5.4, 4.9, 6.1, 5.2, 5.6, 5.6, 5.5, 6, 5.1, 5.5, 6.4, 4.4, 4.3, 5.6, 5.9)

g4(dados, "media")
```

```
## [1] 5.424138
```

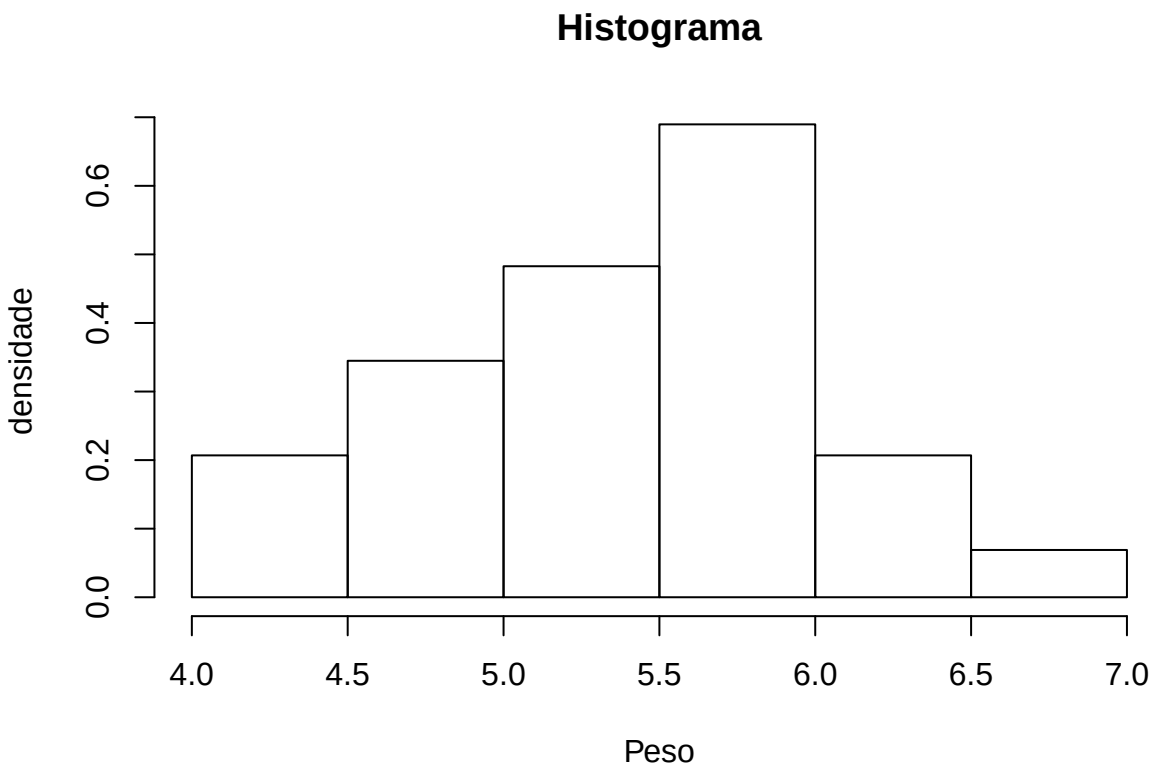
```r
g4(dados, "mediana")
```

```
## [1] 5.5
```

```r
g4(dados, "desvio padrão")
```

```
## [1] 0.6588167
```

```r
g4(dados, "gráfico")
```




```
# funções para contagem de tempo: proc.time e system.time
t0 = proc.time()
for (i in 1:1000) mean(runif(50000))
proc.time() - t0
```

```
##    user  system elapsed
##    2.36    0.22    2.79
```

```
system.time(for (i in 1:1000) mean(runif(50000)))
```

```
##    user  system elapsed
##    2.55    0.20    2.86
```