Ficha de trabalho 1

Madalena Andrade, Hugo Anjos & Tiago A. Marques

Contents

Exercício 1

Realizar o seguinte cálculo no R: $2 * 85/854 + (5 - 3)$

Criamos uma variavel, "x" cujo valor é o resultado da expressão acima. E, de seguida vemos o seu valor.

```r
x <- 2 * 85 / 854 + (5-3)
x
```

```
## [1] 2.199063
```

Exercício 2

Calcular:

a. O logaritmo de 6 (base 10) (log)
b. O logaritmo de 6 (base neperiana) (log)
c. A raíz quadrada de 7 (sqrt ou ^)
d. A raíz cúbica de 7 (^)
e. O seno de 48º (sin) (converter em radianos multiplicando por $\pi/180$)

O logaritmo de base 10 é dada pela função `log10`

```r
log10(6)
```

```
## [1] 0.7781513
```

O logaritmo de base "e" é dada pela função `log(x, base=exp(1))` onde x neste caso é = 6.

```r
log(6, base = exp(1))
```

```
## [1] 1.791759
```

esta é a base default no r, pelo que o código abaixo é equivalente

```r
log(6)
```

```
## [1] 1.791759
```

A raíz quadradada pode ser dada pela função `sqrt()` ou aplicando o exponente `^(1/2)`

```r
sqrt(7)
```

```
## [1] 2.645751
```

```r
7 ^ (1 / 2)
```

```
## [1] 2.645751
```

Da mesma forma, a raíz cúbica pode ser calculada aplicando o exponente ^(1/3)

```r
7 ^ (1 / 3)
```

```
## [1] 1.912931
```

Para trabalhar com funções trigonométricas, que trabalham em radianos por default, é necessário converter os graus para radianos primeiro antes de calcular o seu seno através da função `sin()`

```r
sin(48 * pi / 180)
```

```
## [1] 0.7431448
```

Exercicio 3

Obter:

a. 10 números pseudoaleatórios entre 0 e 1 (runif)
b. 10 números pseudoaleatórios entre 10 e 20 (runif)

3 a)

A função necessita de 3 argumentos : `runif`, com 3 argumentos (nº de números gerados, limite minimo, limite máximo). Cada vez que esta função é corrida geram-se números diferentes.

```
runif(n = 10, min = 0, max = 1)
```

```
##  [1] 0.4635826 0.8264960 0.1608876 0.6572885 0.4761137 0.5992695 0.9879186
##  [8] 0.2033767 0.8801787 0.5573921
```

3 b)

```
runif(10, 10, 20)
```

```
##  [1] 13.51906 12.77560 15.80467 12.67892 14.68973 15.94839 17.56941
##  [8] 16.18785 19.26596 13.74623
```

Exercicio 4

a. Criar o vector "peso", com os seguintes valores: 60, 72, 57, 90, 95, 72
b. Acrescentar os valores 23 e 43 ao vector peso
c. Retirar o valor 57 ao vector peso
d. Somar todos os valores
e. Dividir todos os valores do vector por 10
f. Renomear o vector peso
g. Retirar o vector peso da memória
h. Crie um outro vector com os valores 23, 10, 12, 14, 23, 8, 6
i. Some o vector peso a este novo vector.

4 a)

Um vetor é apenas um conjunto de números concatenados e por isso basta junta-los com a função c()

```
peso <- c(60, 72, 57, 90, 95, 72)
```

4 b)

Podemos acrescentar valores a um vector com a função c() e tratando esse vetor como um número.

```
peso <- c(peso, 23, 43)
```

4 c)

Como o 57 está na posição 3, é só fazer

```
peso <- peso[-3]
```

4 d)

```
sum(peso)
```

```
## [1] 455
```

4 e)

um cálculo sobre o vetor devolve o mesmo cálculo sobre todos os valores que pertencem ao vetor

```
peso <- peso / 10
```

4 f)

```
NovoNome <- peso
```

4 g)

Podemos retirar qualquer variavel do Environment usando a função `rm` com o argumento "nome da variavel"

```
rm(peso)
```

4 h)

```
vetor <- c(23, 10, 12, 14, 23, 8, 6)
```

4 i)

```
SomaDeVetores <- vetor + NovoNome
```

Exercício 5

a. Crie uma matriz constituída por números consecutivos de 1 a 40, com 8 colunas e 5 linhas (matrix(data,ncol,nrow))
b. Crie a seguinte matriz (matrix)
c. Obtenha e atribue nomes aos seguintes elementos i- 1ª linha, 4ª coluna ii- 2ª linha iii- 3ª coluna, 3ª linha iv- 3ª coluna

5 a)

A função matrix cria uma matriz com os dados e por default cria uma matriz constituida por apenas 1 coluna. Podemos definir o número de colunas e linhas da matriz apartir dos argumentos `ncol` e `nrow` respetivamente.

```
matrix1 <- matrix(1:40, ncol = 8, nrow = 5)
matrix1
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    6   11   16   21   26   31   36
## [2,]    2    7   12   17   22   27   32   37
## [3,]    3    8   13   18   23   28   33   38
## [4,]    4    9   14   19   24   29   34   39
## [5,]    5   10   15   20   25   30   35   40
```

5 b)

10	45	98	5
25	42	56	9
36	41	32	12

```
dados <- c(10, 45, 98, 5, 25, 42, 56, 9, 36, 41, 32, 12)
matrix2 <- matrix(dados,
                  ncol = 4,
                  nrow = 3,
                  byrow = TRUE)
matrix2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   10   45   98    5
## [2,]   25   42   56    9
## [3,]   36   41   32   12
```

```
dim(matrix2)
```

```
## [1] 3 4
```

```
# a função dim() dá-nos a dimensao da matriz ou seja, o número de linhas e colunas
```

5 c)

5 c) i.

```
elemento1 <- matrix2[1, 4]
elemento1
```

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

5 c) ii.

```
# [2, ] dá-nos a 2º linha com todas as colunas .
elemento2 <- matrix2[2, ]
elemento2
```

```
## [1] 25 42 56  9
```

5 c) iii.

```
elemento3 <- matrix2[3, 3]
elemento3
```

```
## [1] 32
```

5 c) iv.

```
elemento4 <- matrix2[, 3]
elemento4
```

```
## [1] 98 56 32
```

Exercício 6

a. Obtenha os dados relativos ao eucaliptal
b. Obtenha os dados relativos ao chapim

A partir desta última matriz, crie um data frame com o nome aves, sabendo que os nomes das linhas são relativosa habitats (pinhal, montado, eucaliptal) e os das colunas a espécies (chapim, pardal, alvéola, pintassilgo)

Uma data.frame é como uma matriz mas as suas células podem ter valores não numéricos. Cada coluna da data.frame tem necessariamente o mesmo tipo de dados(e.g. números, ou caracters, ou os níveis de um factor).

```
aves <- data.frame(matrix2, row.names = c("pinhal", "montado", "eucaliptal"))
names(aves) <- c("chapim", "pardal", "alvéola", "pintassilgo")
```

6 a)

```
eucaliptal <- aves[3, ]
eucaliptal
```

```
##            chapim pardal alvéola pintassilgo
## eucaliptal     36     41      32          12
```

6 b)

```
chapim <- aves[, 1]
chapim
```

```
## [1] 10 25 36
```

Exercício 7

a. Os seguintes dados de uma outra espécie (felosa): 5, 6, 2
b. Os seguintes dados de um outro habitat (matos): 12, 0, 1, 3, 11

Adicione ao data frame "aves":

7 a)

Para adicionar um vetor a uma data frame é necessário primeiro transformar o vetor numa data.frame. Só depois é possivel adiciona-la à data frame original usando a função cbind ("bind columns")

```
felosa <- data.frame(felosa = c(5, 6, 2))
aves <- cbind(aves, felosa)
```

7 b)

```
matos <- c(12, 0, 1, 3, 11)
aves <- rbind(aves, c(12, 0, 1, 3, 11))
aves <- data.frame(aves, row.names = c("pinhal", "montado", "eucaliptal", "matos"))
```

#Exercicio 8

Importe os dados do ficheiro DataTP1.csv

a. com recurso à linha de comando, e (setwd, read.table)
b. com recurso aos menus no RStudio (import dataset).

```
# importar os dados atraves da function read.table()
dados1 <- read.table("DataTP1.csv", header = TRUE, sep = ";", dec = ",")

# ou atraves do uso da function read.csv2()
dados1 <- read.csv2("DataTP1.csv")

# Qual a estrutura dos dados?
str(dados1)
```

```
## 'data.frame':    19 obs. of  3 variables:
##  $ Habitat.A: num  73.1 13.7 9 48.8 86.1 ...
##  $ Habitat.B: num  23.7 6.9 6.2 13.8 17.7 14.2 0.9 9.3 12.4 9.4 ...
##  $ Habitat.C: num  58.3 55 54.6 51.2 51.4 57 52.3 61.6 52 67.2 ...
```

```r
# Alterar o nome das 3 colunas dos dados
names(dados1) <- c("x1", "x2", "x3")
```

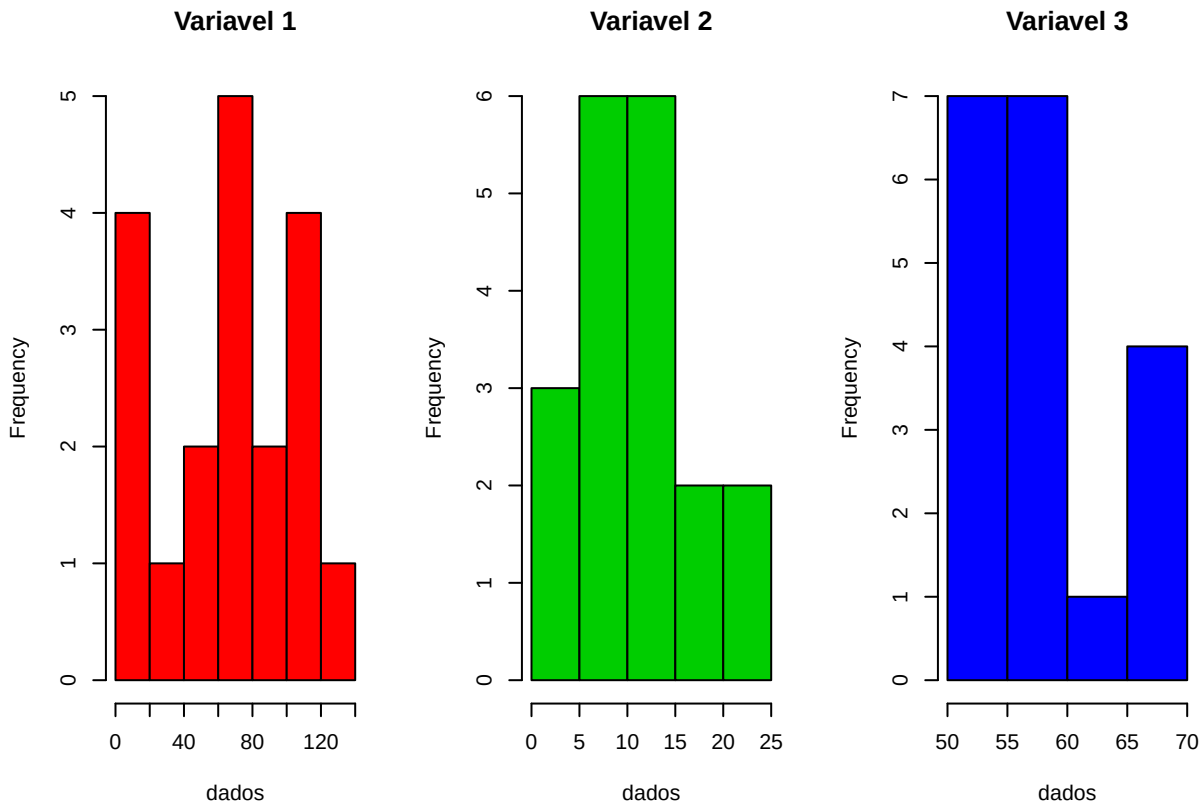
Exercicio 9

a. Histograma (hist)
b. Box-plot (boxplot)
c. Gráfico de dispersão (plot)
d. Altere as opções dos gráficos e obtenha gráficos com uma edição melhorada para publicação (col, pch, main, xlab,ylab, xlim, ylim, etc)

Efectue análises gráficas com recurso aos seguintes tipos de gráficos:

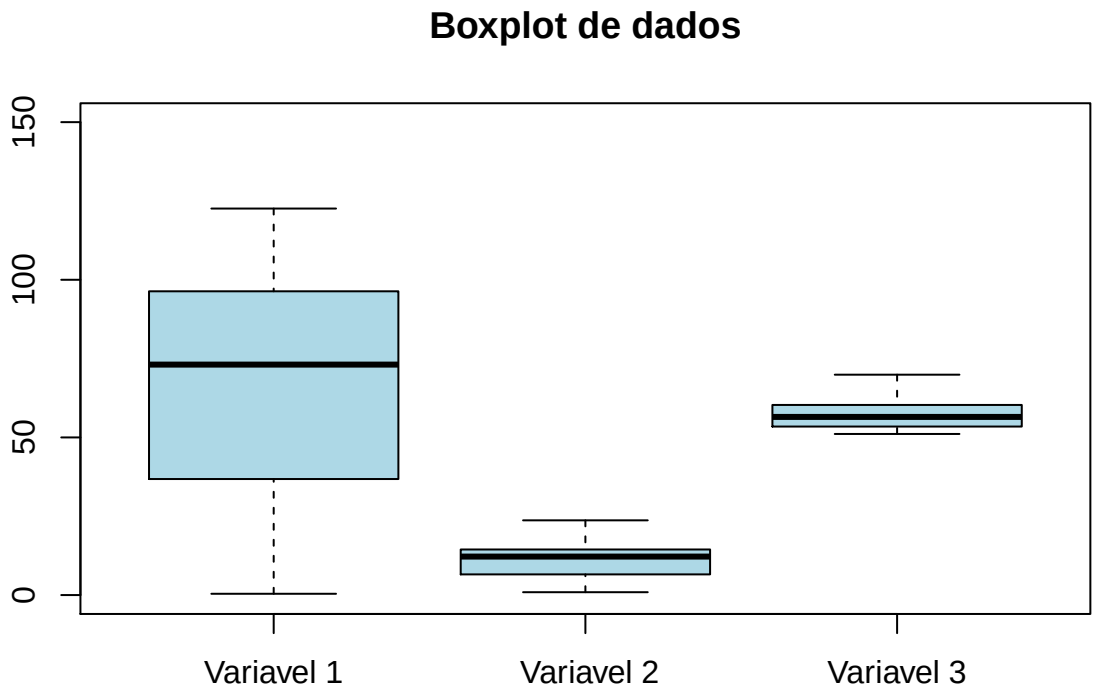
9 a)

```r
par(mfrow = c(1, 3))
hist(dados1$x1, main = "Variavel 1", xlab = "dados", col = 2)
hist(dados1$x2, main = "Variavel 2", xlab = "dados", col = 3)
hist(dados1$x3, main = "Variavel 3", xlab = "dados", col = 4)
```



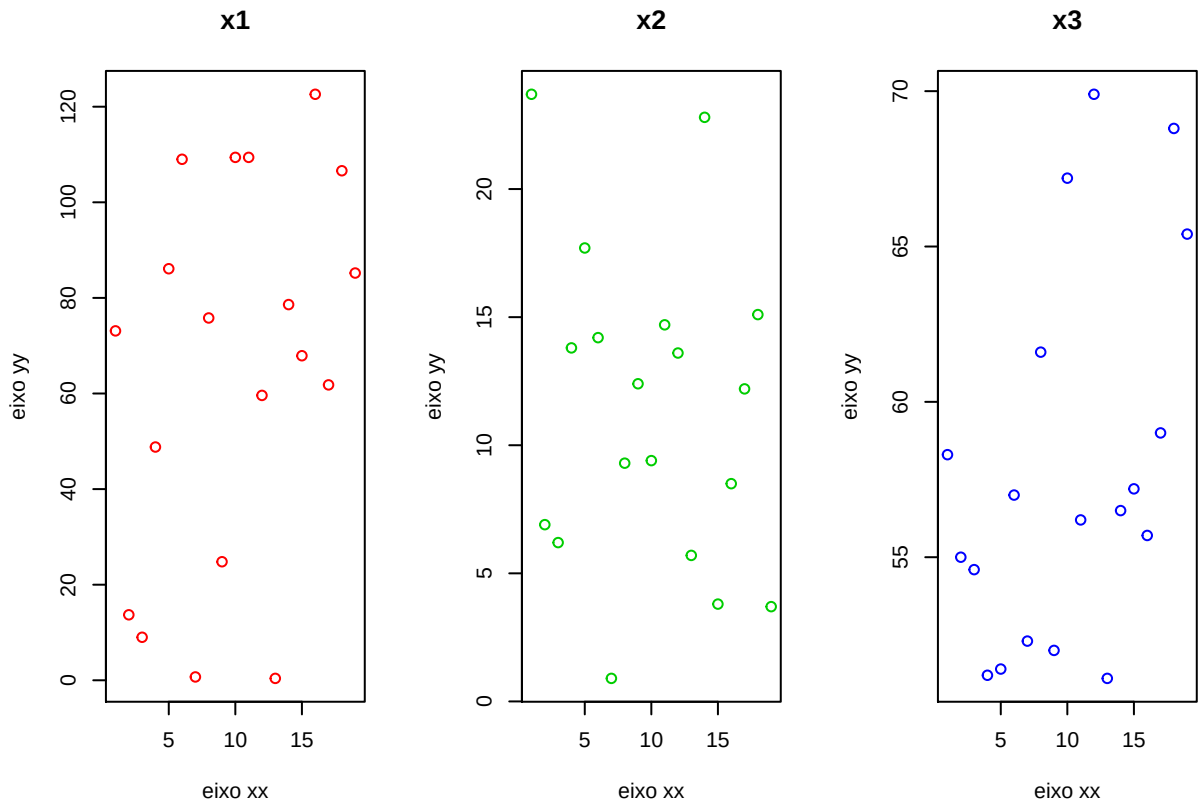
9 b)

```
boxplot(dados1,
        col = "lightblue",
        ylim = c(0, 150),
        main = "Boxplot de dados",
        names = c("Variavel 1", " Variavel 2", "Variavel 3"))
```



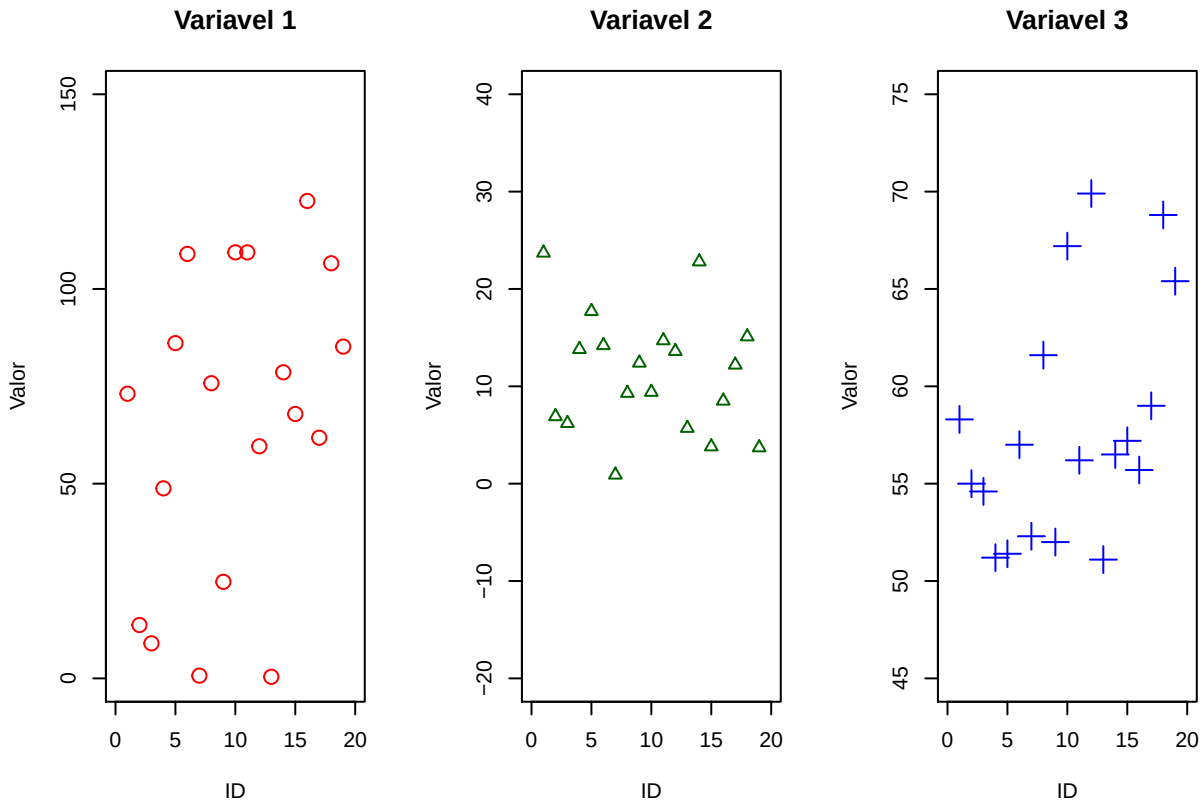
9 c)

```
par(mfrow = c(1, 3))
plot(dados1$x1, main = "x1", ylab = "eixo yy", xlab = "eixo xx", col = 2)
plot(dados1$x2, main = "x2", ylab = "eixo yy", xlab = "eixo xx", col = 3)
plot(dados1$x3, main = "x3", ylab = "eixo yy", xlab = "eixo xx", col = 4)
```



9 d)

```
par(mfrow = c(1, 3))
plot(dados1$x1, main = "Variavel 1",
     ylab = "Valor", xlab = "ID",
     xlim = c(0, 20), ylim = c(0, 150),
     col = 2, pch = 1,cex = 1.5)
plot(dados1$x2, main = "Variavel 2",
     ylab = "Valor", xlab = "ID",
     xlim = c(0, 20), ylim = c(-20, 40),
     col = "darkgreen", pch = 2)
plot(dados1$x3, main = "Variavel 3",
     ylab = "Valor", xlab = "ID",
     xlim = c(0, 20), ylim = c(45, 75),
     col = 4, pch = 3, cex = 2)
```



#Exercicio 11

a. multiplique por dois todos os valores
b. divida por 10 todos os valores
c. some 50, multiplique por 2.5, divida por 0.25 um determinado valor; aplique o logaritmo de base neperiana a um outro valor e calcule o produto dos 2 valores

Crie uma função para cada uma das situações, tal que:

11 a)

```
# primeiro criamos a function func1:
func1 <- function(x) {
  x * 2
}

# depois aplicamos a funcao
func1(x = 1:20) # estes são os valores sobre os quais serão feitas as operações da fun
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
```

11 b)

```
fun2 <- function(x) {
  x / 10
}

fun2(1:10)
```

```
##  [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

11 c)

```
# criamos a funcao que recebe 2 parametros:
func3 <- function(x, y) {
  n1  <- (x + 50) * 2.5 / 0.25
  n2  <- log(y)
  res <- n1 * n2
  return(res)
}

func3(x = 2, y = 3)
```

```
## [1] 571.2784
```

#Exercício 12

Verifique quais os packages que tem instalados na sua versão do R (installed.packages, library)

```
#installed.packages()
#library("nome do package que queremos usar")
```