

Ficha de trabalho 1

Madalena Andrade, Hugo Anjos & Tiago A. Marques

Contents

Exercício 1	2
Exercício 2	2
Exercício 3	3
a	3
b	3
Exercício 4	3
a	3
b	3
c	4
d	4
e	4
f	4
g	4
h	4
i	4
Exercício 5	4
a	5
c	5
Exercício 6	6
a	6
b	7
Exercício 7	7
a	7
b	7
Exercício 8	7
Exercício 9	8
a	8
b	8
c	9
d	10
Exercício 11	11
a	11
b	12
c	12
Exercício 12	12

Exercício 1

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

Criamos uma variável, “x” cujo valor é o resultado da expressão acima. E, de seguida vemos o seu valor.

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

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

Exercício 2

Calcular:

- O logaritmo de 6 (base 10) (log)
- O logaritmo de 6 (base neperiana) (log)
- A raiz quadrada de 7 (sqrt ou ^)
- A raiz cúbica de 7 (^)
- O seno de 48° (sin) (converter em radianos multiplicando por $\pi/180$)

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

```
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.

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

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

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

```
log(6)
```

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

A raiz quadrada pode ser dada pela função `sqrt()` ou aplicando o expoente $^{(1/2)}$

```
sqrt(7)
```

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

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

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

Da mesma forma, a raiz cúbica pode ser calculada aplicando o expoente $^{(1/3)}$

```
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()`

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

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

Exercicio 3

Obter:

- 10 números pseudoaleatórios entre 0 e 1 (runif)
- 10 números pseudoaleatórios entre 10 e 20 (runif)

a

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

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

```
## [1] 0.5795415 0.3656011 0.1652251 0.2904271 0.1497087 0.2013967 0.5795817
## [8] 0.3671233 0.6477101 0.9796399
```

b

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

```
## [1] 12.00678 14.91604 16.67790 15.32586 19.26243 13.68680 19.19531
## [8] 13.12800 12.58410 13.34456
```

Exercicio 4

- Criar o vector “peso”, com os seguintes valores: 60, 72, 57, 90, 95, 72
- Acrescentar os valores 23 e 43 ao vector peso
- Retirar o valor 57 ao vector peso
- Somar todos os valores
- Dividir todos os valores do vector por 10
- Renomear o vector peso
- Retirar o vector peso da memória
- Crie um outro vector com os valores 23, 10, 12, 14, 23, 8, 6
- Some o vector peso a este novo vector.

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)
```

b

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

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

c

Como o 57 está na posição 3, é só fazer

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

d

```
sum(peso)
```

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

e

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

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

f

```
NovoNome<-peso
```

g

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

```
rm(peso)
```

h

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

i

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

Exercício 5

- Crie uma matriz constituída por números consecutivos de 1 a 40, com 8 colunas e 5 linhas (`matrix(data,ncol,nrow)`)
- Crie a seguinte matriz (`matrix`)

c. Obtenha e atribua nomes aos seguintes elementos i- 1ª linha, 4ª coluna ii- 2ª linha iii- 3ª coluna, 3ª linha iv- 3ª coluna

a

A função `matrix` cria uma matriz com os dados e por default cria uma matriz constituída por apenas 1 coluna. Podemos definir o número de colunas e linhas da matriz a partir 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
```

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 dimensão da matriz ou seja, o número de linhas e colunas

c

i

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

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

ii

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

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

iii

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

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

iv

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

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

Exercício 6

- Obtenha os dados relativos ao eucaliptal
- 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 relativos 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 caracteres, ou os níveis de um factor).

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

a

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

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

b

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

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

Exercício 7

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

Adicione ao data frame “aves”:

a

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

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

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"))
```

Exercício 8

Importe os dados do ficheiro DataTP1.csv

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

```
dados1 = read.table("DataTP1.csv",header=TRUE,sep=";",dec=",")
dados1 <- read.csv2("DataTP1.csv")
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 ...
```

```
names(dados1)=c("x1","x2","x3")
```

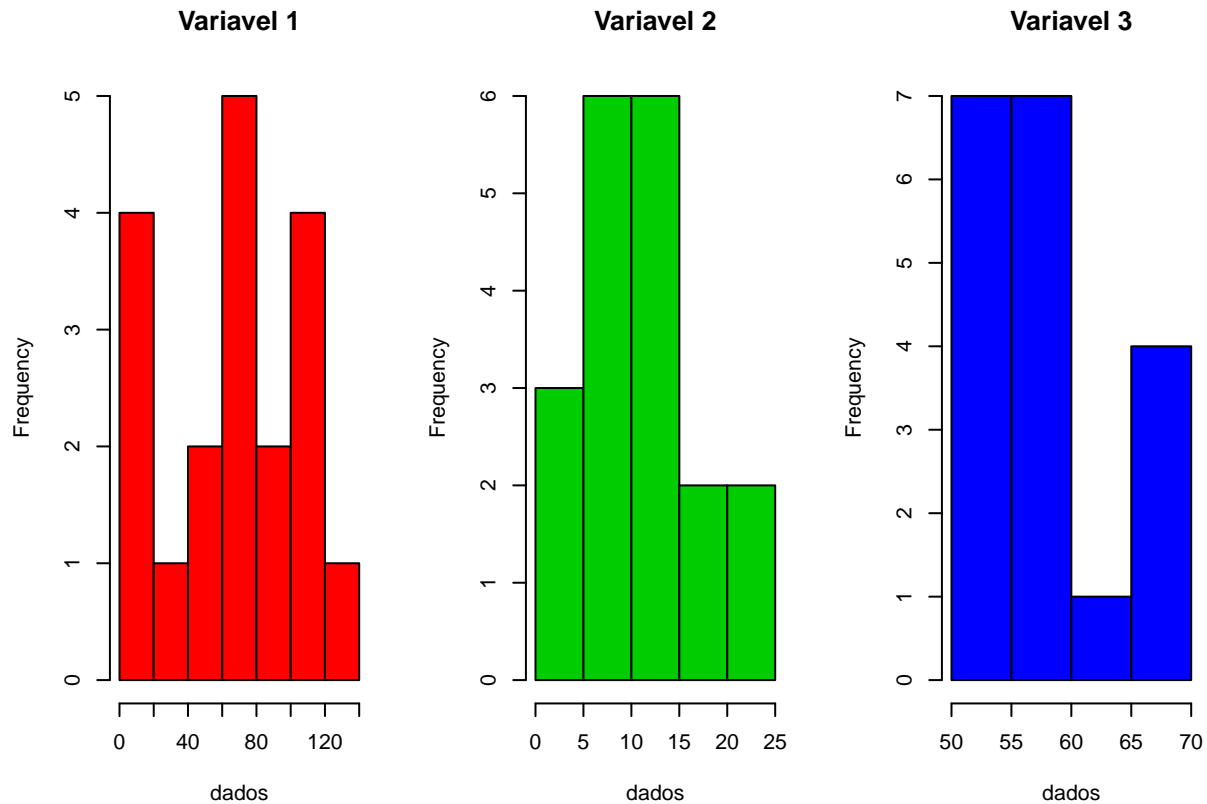
Exercício 9

- Histograma (hist)
- Box-plot (boxplot)
- Gráfico de dispersão (plot)
- 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:

a

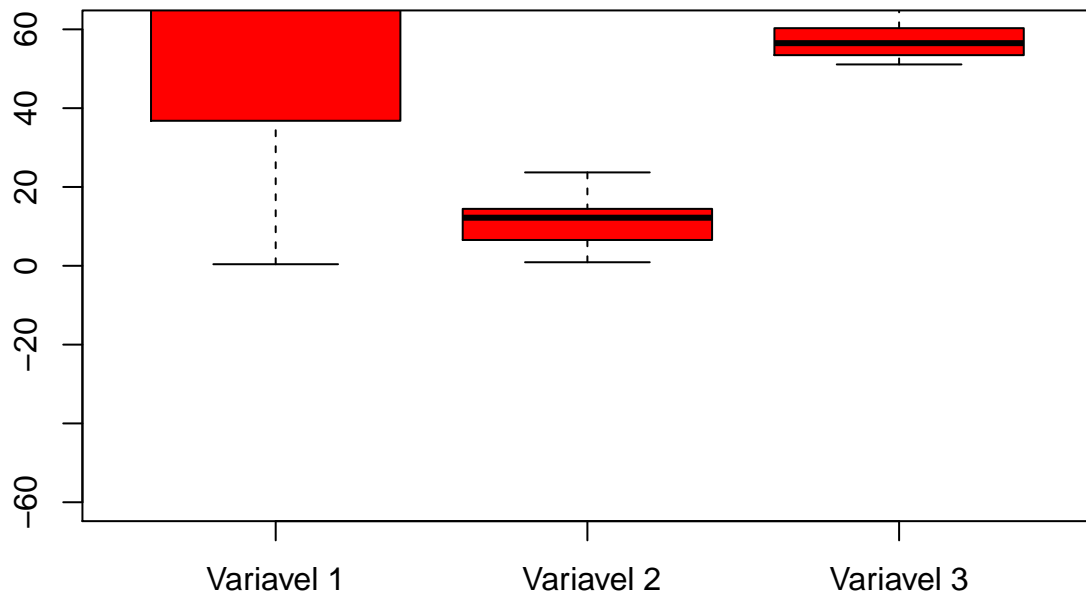
```
par(mfrow=c(1,3))
hist(dados1$x1, main="Variavel 1", xlab="dados", col=2)
hist(dados1$x2, main="Variavel 2", xlab="dados", col=3)
hist(dados1$x3, main="Variavel 3", xlab="dados", col=4)
```



b

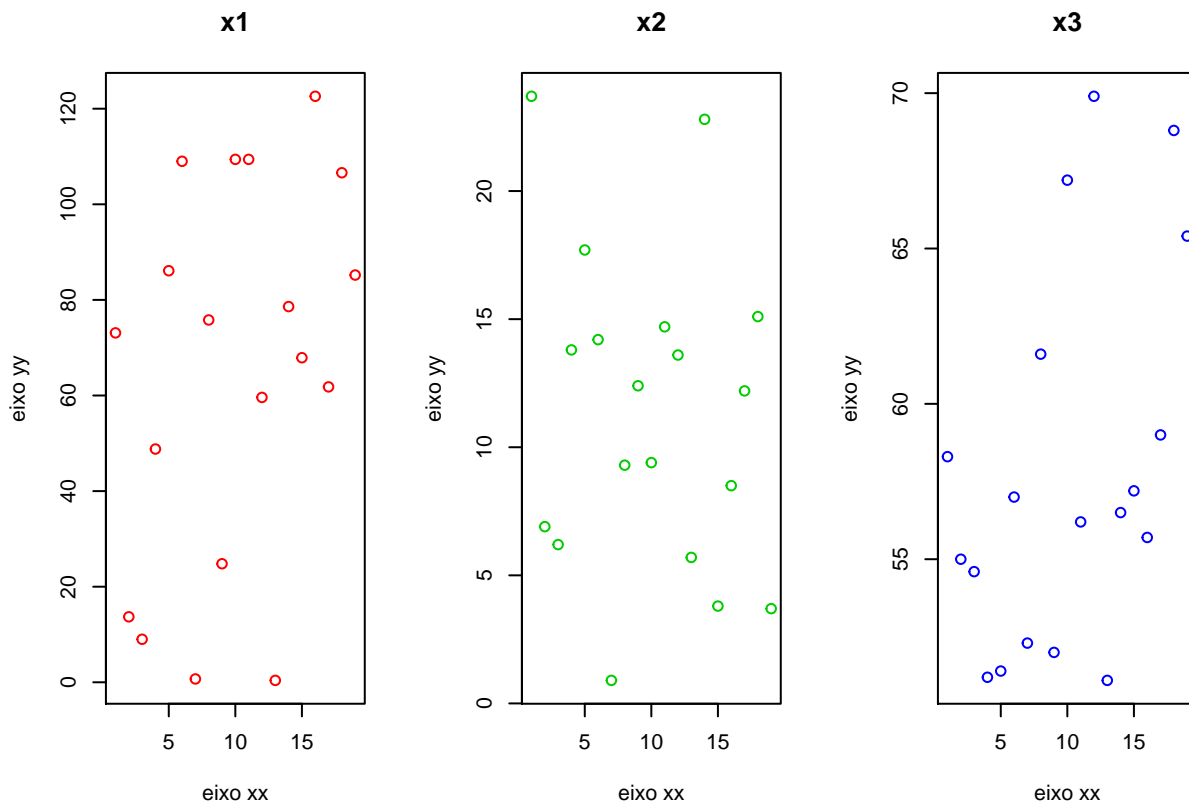
```
boxplot(dados1, col="red", ylim=c(-60,60),main="boxplot de dados", names=c("Variavel 1"," Variavel 2",
```


boxplot de dados



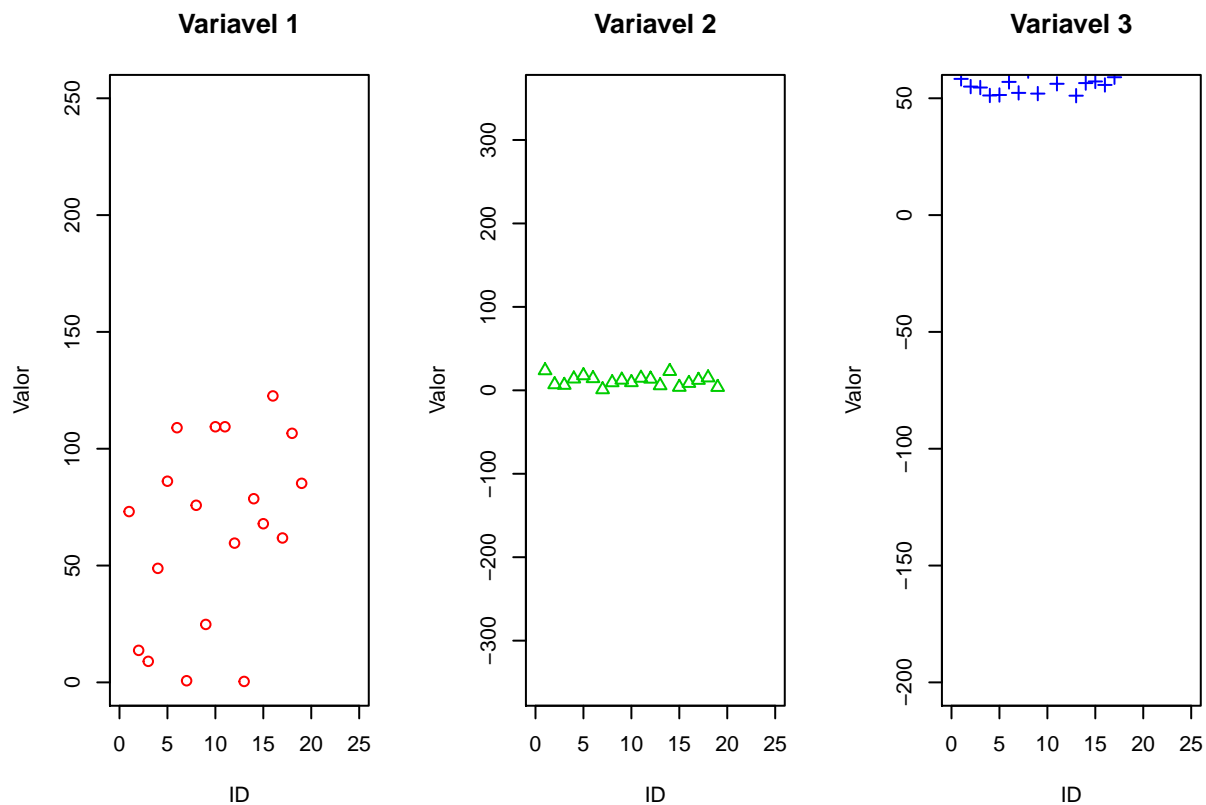
c

```
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)
```



d

```
par(mfrow=c(1,3))
plot(dados1$x1, main="Variavel 1", ylab = "Valor", xlab = "ID", col=2, pch=1, xlim =c(0,25), ylim = c(0,25))
plot(dados1$x2, main="Variavel 2", ylab = "Valor", xlab = "ID", col=3, pch=2, xlim =c(0,25), ylim = c(-35,0))
plot(dados1$x3, main="Variavel 3", ylab = "Valor", xlab = "ID", col=4, pch=3, xlim =c(0,25),ylim=c(-200,50))
```



Exercicio 11

- multiplique por dois todos os valores
- divida por 10 todos os valores
- 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:

a

```
func1<-function(x)
  {x*2
}
func1(x=1:20) #estes são os valores sobre os quais serão feitas as operações da função
```

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

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
```

c

```
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)
```