Introdução ao R

Miguel Rocha

Ferramentas de trabalho: R

- O R é um software de computação estatística e gráfica. Utiliza uma linguagem de programação simples.
- É uma linguagem interpretada (os comandos são imediatamente executados)
- Os dados manipulados são armazenados na forma de objetos, que têm um nome e aos quais se podem aplicar ações.
- O R é uma aplicação de distribuição gratuita e de código público http://cran.r-project.org
- Dada a quantidade e qualidade de funções que possui e a sua natureza aberta é uma das principais ferramentas para a análise de dados e será usada neste curso

Ferramenta de trabalho: IDE

- Para facilitar o trabalho e aumentar a produtividade, podem utilizar-se *integrated development environments* (IDEs)
- Estes criam um ambiente integrado que inclui a linha de comandos R e um conjunto de ferramentas complementares
- IDE recomendado para o curso - **RStudio**: www.rstudio.com
- Inclui linha de comandos R + janelas para edição de scripts, análise do histórico e do workspace, visualização de gráficos, gestão de packages e de ficheiros/ pastas

Objetos e atribuição de variáveis

- **Objetos** (ou variáveis): “contentores” da informação, ou seja, permitem guardar dados de diversos tipos
- **Funções**:
  - os mecanismos de processamento e manipulação dos dados;
  - podem receber zero ou mais argumentos de entradas (objetos) e retornam como resultado um novo objeto (podendo não retornar resultado algum).
  - Podem ser chamadas em R através da sintaxe: **nome_função**(*argumentos*);
  - existem definidas em packages instalados ou podem ser programadas pelo utilizador

Packages

- Quando o R arranca são carregados alguns *packages* que possuem um conjunto de funções já disponíveis
- É possível carregar novos packages instalados na sua distribuição com a função **library**(*package*)
- É ainda possível instalar novos *packages* com a função **install.package**(*packages*)
- O Rstudio permite gestão dos packages carregados e instalados, bem como a instalação de novos packages de forma simples

Workspace e histórico

- ***Workspace*** (image) – conjunto de todos os objetos criados numa sessão
- Histórico (***history***) – conjunto de todos os comandos efetuados ao longo de uma sessão
- Ao terminar uma sessão pode guardar ambos em ficheiros chamados ".RData" e ".Rhistory", na sua pasta corrente de trabalho
- Pode guardar estes ficheiros em qualquer altura com o Rstudio ou com as funções **save.image** e **savehistory**
- Se o R for recomeçado da mesma diretoria estes ficheiros são automaticamente carregados no início da sessão seguinte
- Deve definir a sua pasta de trabalho no início da sessão (com função **setwd** ou usando o Rstudio)

Objetos e atribuição

- A cada objeto no R é atribuído um nome que o identifica e um determinado tipo, que indica qual a estrutura dos dados que pode guardar
- Os tipos são atribuídos de forma automática pelo conteúdo atribuído ao objeto
- **Atribuição** - permite associar um valor ou conteúdo (lado direito) a um nome (lado esquerdo) usando os símbolos **=** ou **<-**

```
> x <- 2
> y = sin (pi)
> nome <- “paulo”
> flag = TRUE
```

Listar e remover objetos

- A lista de objetos disponíveis numa dada sessão do R (*workspace*) pode ser consultada com a função **ls**()
- O Rstudio permite a visualização desta lista e dos tipos dos objetos numa janela independente
- O tipo do objeto pode ser verificado com a função **typeof**(*objecto*)
- A função **rm**(*objectos*) pode ser usada para remover objetos

Ajuda e documentação

- Função **help** serve para descobrir ajuda sobre uma função do sistema – **help**(*nome_função*)
- Função **example**(*nome_função*) permite consultar exemplos ilustrativos do uso de cada função
- No Rstudio, a ajuda pode ser consultada em janela independente; existe ainda um menu Help com acesso a documentação interessante

Valores numéricos/ operadores aritméticos

```
> 2+3
[1] 5
> 3 * 1/2
[1] 1.5
> 5.3 - 4.2 / 2.3
[1] 3.473913
```

```
> y = 3.1
> z = 6.2
> y / z
[1] 0.5
> w = y * z + 2
```

```
> y=5
> y^2
[1] 25
> y%%2
[1] 1
> y%/%2
[1] 2
```

Operadores aritméticos comuns: +, -, *, /
Exponenciação: ^
Divisão inteira: %/%
Resto divisão: %%

Vetores numéricos

Vetores: constituem objetos que permitem guardar uma sequência de valores

Criando vetores com a função **c**()

```
> v1 = c(1,5,8,10)
> v1
[1]  1  5  8 10
> v2 = c(3,v1)
> v2
[1]  3  1  5  8 10
```

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> 10:2
[1] 10  9  8  7  6  5  4  3  2
```

```
> seq(1, 2, 0.1)
 [1]  1.0  1.1  1.2  1.3  1.4  1.5
1.6  1.7  1.8  1.9  2.0
> seq(1, 10)
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> x = c(2,4,6)
> rep(x, times=5)
 [1] 2 4 6 2 4 6 2 4 6 2 4 6 2 4 6
> rep(x, each=5)
 [1] 2 2 2 2 2 4 4 4 4 4 6 6 6 6 6
```

Geração de sequências

Aritmética vetorial

```
> x = 1:10
> 2*x+1
 [1]  3  5  7  9 11 13 15 17 19 21
> x/2
 [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> x-5
 [1] -4 -3 -2 -1  0  1  2  3  4  5
```

```
> vm = 1:5
> vn = 1:10
> vm+vn
 [1]  2  4  6  8 10  7  9 11 13 15+
```

```
> vm = 1:3
> vn = 1:10
> vm+vn
 [1]  2  4  6  5  7  9  8 10 12 11
Warning message:
longer object length is not a multiple of shorter object length in:
vm + vn
```

Indexação de vetores

```
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x[2]
[1] 2
> x[x>5]
 [1]  6  7  8  9 10
```

```
> v1
[1]  1  5  8 10
> ind=2:3
> v1[ind]
[1] 5 8
> v1[-2]
[1]  1  8 10
> v1[-(2:3)]
[1]  1 10
```

```
> v1 = c(1,5,8,10)
> names(v1) = c("azul","amarelo","verde","vermelho")
> v1
    azul  amarelo    verde vermelho
       1        5        8       10
> v1["verde"]
verde
    8
```

Funções sobre valores numéricos

```
> v =c(-1,2,3,-4)
> abs(v)
[1] 1 2 3 4
> sqrt(abs(v))
[1] 1.000000 1.414214 1.732051 2.000000
> sin(pi/2*v)
[1] -1.000000e+00  1.224606e-16 -1.000000e+00  2.449213e-16
> nr = c(1.23, 2.321, 4.07654, 3, 2.345)
> round(nr, 1)
[1] 1.2 2.3 4.1 3.0 2.3
> p10 = c(10,100,1000,10000)
> log10(p10)
[1] 1 2 3 4
```

```
> mapply(sqrt, abs(v))
[1] 1.000000 1.414214 1.732051 2.000000
> mapply("/",p10,10)
[1]    1   10  100 1000
```

Funções sobre vetores numéricos

```
> v = c(2,4,7,6,3,1)
> rev(v)
[1] 1 3 6 7 4 2
> sort(v)
[1] 1 2 3 4 6 7
> sort(v, decreasing=TRUE)
[1] 7 6 4 3 2 1
> order(v)
[1] 6 1 5 2 4 3
> v[order(v)]
[1] 1 2 3 4 6 7
> unique(c(1,2,1,2,3))
[1] 1 2 3
> cumsum(v)
[1]  2  6 13 19 22 23
> cumprod(v)
[1]    2    8   56  336 1008 1008
> diff(v)
[1]  2  3 -1 -3 -2
```

Resultado é um vetor

```
> vs = c(3,5,7,9)
> sum(vs)
[1] 24
> mean(vs)
[1] 6
> min(v)
[1] 3
> max(v)
[1] 9
> which.max(v)
[1] 4
> which.min(v)
[1] 1
> range(v)
[1] 3  9
```

Resultado é um valor numérico

Vetores e operadores lógicos

```
> vb1 = c(TRUE,TRUE,FALSE,TRUE,FALSE)
> vb1
[1]  TRUE  TRUE FALSE  TRUE FALSE
> vb1[3]
[1] FALSE
> v = 1:10
> vb2 = v > 5
> vb2
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Operadores lógicos

```
> vb2 | vb1
 [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> vb1 & !vb2
 [1]  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
> all(vb1)
[1] FALSE
> any(vb1)
[1] TRUE
> which (vb1)
[1] 1 2 4
```

Strings e vetores de strings

```
> nome = "paulo"
> apelido = "silva"
> toupper(nome)
[1] "PAULO"
> paste(nome, apelido)
[1] "paulo silva"
> substr(apelido,2,4)
[1] "ilv"
> nchar(nome)
[1] 5
> sub("lo","la",nome)
[1] "paula"
> str = "abracadabra"
> gsub("a","x",str)
[1] "xbrxcxdxbrx"
> chartr("abc","ABC",str)
[1] "ABrACAdABrA”
```

```
> nomes = c("joao", "joaquim",
"jose")
> apelidos = c("silva", "sousa")
> paste(nomes, apelidos, sep=" ")
[1] "joao silva"    "joaquim sousa"
"jose silva"
> toupper(nomes)
[1] "JOAO"    "JOAQUIM" "JOSE"
> mapply(nchar,nomes)
   joao joaquim    jose
      4       7       4
> sub("j","J",nomes)
[1] "Joao"    "Joaquim" "Jose"
```

Fatores

Permitem representar variáveis categóricas

```
> racas = c("bulldog", "rafeiro", "doberman", "rafeiro", …
> fr = factor(racas)
> fr
[1] bulldog  rafeiro  doberman rafeiro  bulldog  rafeiro …
Levels: bulldog doberman rafeiro
> levels(fr)
[1] "bulldog"  "doberman" "rafeiro”
> table(fr)
fr
 bulldog doberman  rafeiro
       2        2        4
> pesos = c(12, 15, 35, 10, 20, 8, 13, 25)
> tapply(pesos,fr,mean)
 bulldog doberman  rafeiro
    16.0     30.0     11.5
```

Cada possível valor (*level*) representado por um número inteiro internamente

Matrizes

Matriz: coleção de dados, todos do mesmo tipo, referenciados por dois índices (linhas e colunas)

```
> mat = matrix(1:20, 4, 5)
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

```
> mat[2,3]
[1] 10
> mat[1:2,4:5]
     [,1] [,2]
[1,]   13   17
[2,]   14   18
```

```
> dim(mat)
[1] 4 5
> nrow(mat)
[1] 4
> ncol(mat)
[1] 5
```

```
> mat[2,]
[1]  2  6 10 14 18
> mat[,1]
[1] 1 2 3 4
> mat[2:3,]
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    6   10   14   18
[2,]    3    7   11   15   19
```

Representadas internamente por um vector com os valores e outro com a organização (*dim*)

Operações sobre matrizes

```
> m1 <- cbind(x, x^2, x^3)
> m1
       x
 [1,]  1   1    1
 [2,]  2   4    8
 [3,]  3   9   27
> rbind(m1, c(20, 40, 60))
       x
 [1,]  1   1    1
 [2,]  2   4    8
[10,] 10 100 1000
[11,] 20  40   60
```

Retornam matrizes

```
> mean(mat)
[1] 10.5
> sqrt(mat[2:3,3:4])
         [,1]     [,2]
[1,] 3.162278 3.741657
[2,] 3.316625 3.872983
> apply(mat, 1, sum)
[1] 45 50 55 60
```

```
> A = matrix(1:4,2,2)
> B = matrix(4:1,2,2)
> C= A%*%B
> C
     [,1] [,2]
[1,]   13    5
[2,]   20    8
> t(C)
     [,1] [,2]
[1,]   13   20
[2,]    5    8
> det(C)
[1] 4
> A = matrix(1:4,2,2)
> b = c(1,3)
> solve (A,b)
[1]  2.5 -0.5
```

Retornam vetores ou valores numéricos

Lists

- Coleção ordenada de objetos, que podem ser de tipos diferentes (vetores numéricos, vetores lógicos, matrizes, listas, funções, ...)
- Os elementos da lista são numerados e têm normalmente um nome associado.

```
> auto = list(marca="ford", modelo="fiesta", nportas=5,
velocMax=155, consumos=c(6,7.1,9.3))
> auto
$marca
[1] "ford"
$modelo
[1] "fiesta"
...
> is.list(auto)
[1] TRUE
> auto[[1]]
[1] "ford"
> auto$marca
[1] "ford”
```

Data frames

- *Data frames* são tipos especiais de *lists* onde os componentes são vetores numéricos ou fatores e todos têm o mesmo comprimento
- Podem ser encarados como matrizes especiais onde pode haver um tipo diferente para cada coluna
- Existem diversos data.frames disponíveis em packages do R que podem ser carregados com a função **data**(*nome_dataset*); exemplo: data(iris); iris
- Para obter mais conjuntos de dados:

http://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html

Data frames

```
> library(MASS)
> data(Cars93)
> head(Cars93)
```

	Manufacturer	ModelType	Min.Price	Price	Max.Price	MPG.city	MPG.highway
1	Acura Integra	Small	12.9	15.9	18.8	25	31
2	Acura Legend	Midsize	29.2	33.9	38.7	18	25
3	Audi 90	Compact	25.9	29.1	32.3	20	26
4	Audi 100	Midsize	30.8	37.7	44.6	19	26
5	BMW 535i	Midsize	23.7	30.0	36.2	22	30
6	Buick Century	Midsize	14.2	15.7	17.3	22	31

Data frame: exemplo - criação

```
> r = c("bulldog", "rafeiro", "doberman", "rafeiro", "bulldog",
"rafeiro", "rafeiro", "doberman")
> p = c(12, 15, 35, 10, 20, 8, 13, 25)
> t = c("medio", "medio", "grande", "pequeno", "grande",
"pequeno", "medio", "grande”)
> df = data.frame(racas=r, tamanhos=t, pesos=p)
> df
     racas tamanhos pesos
1  bulldog    medio    12
2  rafeiro    medio    15
3 doberman   grande    35
4  rafeiro  pequeno    10
5  bulldog   grande    20
6  rafeiro  pequeno     8
7  rafeiro    medio    13
8 doberman   grande    25
```

Raça	Pesos	Tamanhos
bulldog	12	medio
rafeiro	15	medio
doberman	35	grande
rafeiro	10	pequeno
bulldog	20	grande
rafeiro	8	pequeno
rafeiro	13	medio
doberman	25	grande

Data frame: exemplo – acesso aos dados

```
> df$tamanhos
[1] medio   medio   grande  pequeno grande  pequeno
medio   grande
Levels: grande medio pequeno
> df$pesos[1:4]
[1] 12 15 35 10
> df[2,2]
[1] medio
Levels: grande medio pequeno
> df[1:3,]
     racas tamanhos pesos
1  bulldog    medio    12
2  rafeiro    medio    15
3 doberman   grande    35
> df[,2]
[1] medio   medio   grande  pequeno grande  pequeno
medio   grande
Levels: grande medio pequeno
```

Data frame: exemplo – acesso aos dados

```
> df[df$racas=="bulldog",]
    racas tamanhos pesos
1 bulldog    medio    12
5 bulldog   grande    20
> df[df$racas=="bulldog" & df$tamanhos=="medio",]
    racas tamanhos pesos
1 bulldog    medio    12
> vl = df$pesos > 12
> vl
[1] FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
> df[vl,]
     racas tamanhos pesos
2  rafeiro    medio    15
3 doberman   grande    35
5  bulldog   grande    20
7  rafeiro    medio    13
8 doberman   grande    25
```

Definição de novas funções

- O R permite que o utilizador possa definir novas funções que poderão ser utilizadas da mesma forma que as pré-definidas no R
- Para a definição de uma nova função usa-se a palavra chave **function**

Definição

Chamada

```
> "quadrado" = function(x) x^2

> quadrado(3)
[1] 9
```

Definição de novas funções

- Novas funções poderão ser definidas na linha de comandos
- Funções poderão também ser definidas num ficheiro de texto e carregadas no R usando a função **source**(*nome_ficheiro*)
- O Rstudio permite gerir as scripts e o seu carregamento de forma mais adequada numa janela própria

Definição de novas funções: exemplos

```
"volume.esfera" = function (r)
# esta função calcula o volume de uma esfera
{
  res = 4/3 * pi * cubo(r)
  res
}
```

```
> volume.esfera(2)
[1] 33.51032
```

```
"distancia" = function(x1, y1, x2, y2)
{
  r = (x1-x2)^2 + (y1-y2)^2
  sqrt(r)
}
```

```
> distancia(0,0,1,1)
[1] 1.414214
> distancia(0,2,2,0)
[1] 2.828427
```

Programação em R: exemplos

O R tem uma linguagem de programação que pode ser usada para criar expressões condicionais, ciclos, etc

```
> x <- -5
> if(x > 0){
+    print("Non-negative number")
+ } else {
+    print("Negative number")
+ }
```

```
> i <- 1
> while (i < 6) {
+     print(i)
+     i = i+1
+ }
```

```
> for (k in 1:5){
+   print(k)
+ }
```

Em muitos casos, os ciclos devem ser substituidos pelo uso de funções internas do R que têm **vetorização** e por isso são mais eficientes