Introdução ao R

draft em construção

Manuel Ribeiro

CERENA - IST/UL

1. Introdução

1.1 O que é o R?

R é uma linguagem e um ambiente para realizar cálculo estatístico e gráfico. É de distribuição gratuita e corre em diferentes plataformas (Linux, MacOS, Windows).

Foi desenvolvido por Ross Ihaka e Robert Gentleman. Atualmente o desenvolvimento mantido por uma equipa R Core Team (https://www.r-project.org).

1.2 O que é o RStudio

É um ambiente integrado de desenvolvimento que permite interagir com o R de forma mais simples, através de uma interface amigável.

1.3 Download do R e do RStudio

Todos os computadores da sala tem instalados as versões R 3.6.1 e RStudio 1.2.1335 (são as mais recentes). Novas versões do R costumam aparecer de 6 em 6 meses, enquanto que o RStudio costuma sofrer actualizações mais frequentemente (cada 3-6 meses).

No caso de os alunos quererem instalar o R e RStudio nos portáteis, devem fazer download de uma cópia, usando um mirror próximo.

Nota: Instalar primeiro o R e só depois o RStudio.

1.4 Iniciar uma sessão com RStudio

A primeira vez que se abre o RStudio aparecem 3 janelas:

- Consola,
- Environment/History,
- Files/Plots/Packages/Help.

A janela de Scripts não deverá estar visível na primeira vez que se abre o RStudio. Pode ser activada clicando no menu File → New File → R Script.

Na consola, a prompt (>) indica onde se escrevem os comandos. Para executar os comandos carrega-se no Enter. Se o comando não estiver completo, o R produz a prompt (+) de continuação. Outro modo de escrever e execurtar comandos é via um script.

Os tabuladores Files/Plots/Packages/Help permitem fazer a gestão integrada do *workflow*: projetos, ficheiros, gráficos, *packages*, documentação de help.

1.5 Boas práticas

No *R style guide* (http://adv-r.had.co.nz/Style.html) encontram conselhos sobre boas práticas de notação, sintaxe e organização. Por baixo seguem mais alguns conselhos.

Criem um projecto em RStudio

Facilita a gestão do fluxo de trabalho. Para criar um projecto novo (.Rproj), usa-se o comando Create Project (menu File → New Project).

https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects

Neste *link* explica-se passo-a-passo como criar um projeto novo.

Escrevam o código em scripts

É recomendável escrever o código num script, pois pode depois gravar num ficheiro de extensão (.R) para uso futuro.

Comentem o código

O texto à direita do simbolo (#) é ignorado pelo R, serve para comentar o código.

Perguntas e respostas: usem os recursos online

https://stackoverflow.com/questions/tagged/r

É um site com comunidade muito ativa e útil, focado em perguntas e respostas sobre temas variados (ciência, tecnologia, …). Tem muito material sobre código R.

https://www.r-bloggers.com/

Site com contribuições de bloggers com novidades e tutoriais sobre R.

Usem o help

Na consola, escrevendo ?*nomedafunção* (ex: `?mean`, `?sin`); no separador *Help* do R-Studio.

Atalhos com teclado

No menu Tools → Keyboard Shortcuts Help tem a lista.

Também as podem modificar a vosso gosto no menu Tools → Modify Keyboard Shortcuts.

2. Expressões aritméticas e atribuições

2.1 Os operadores aritméticos, relacionais, lógicos

Os operadores aritméticos são `+, -, *` e `/` para adicionar, subtrair, multiplicar e dividir. O símbolo `^` serve para o expoente. Também se podem usar outras funções como `abs()`, `sqrt()`, `exp()` , `log()` (`log10` na base 10), `sin()`, `cos()`, ...

```
2-1
```

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

```
exp(1)
```

```
## [1] 2.718282
```

O R também usa operadores relacionais e lógicos para criar expressões relacionais. As expressões relacionais traduzem uma condição de decisão, o resultado é uma expressão boleana (verdadeiro / falso).

Os operadores relacionais seguintes podem ser usados no R:

- Menor ou igual (`<=`)
- Menor (`<`)
- Igual (`==`)
- Maior (`>`),
- Maior ou igual (`>=`)
- Diferente (`!=`)

```
6<5
```

```
## [1] FALSE
```

```
12!=10
```

```
## [1] TRUE
```

Os operadores relacionais podem ser combinados com os seguintes operadores lógicos:

- `|` (OU)
- `&` (E).

```
3>2 & 3>3     # e
```

```
## [1] FALSE
```

```
6<5 | 12!=10  # ou
```

```
## [1] TRUE
```

2.2 Atribuições

Os valores que calculámos até aqui não ficam guardados. Para guardar os valores, é necessário atribuí-los a variáveis. O operador de atribuição em R é `<-`, mas também pode ser o `=`.

```
i <- 6 # ou
i = 6
```

Digitando o nome da variável imprime o valor na consola:

```
i
```

```
## [1] 6
```

3. Objetos

Até agora apenas se falaram em escalares, mas o R está desenhado para trabalhar com vários tipos de objetos. Os tipos de objectos R

- vetores doubles, inteiros, logicos ou texto,
- matrizes,
- arrays,
- listas,
- data-frames,
- ...

Os objetos que interessam particularmente neste curso são os vetores, matrizes, listas e data-frames.

3.1 Vetores

São colecções ordenadas de elementos do mesmo tipo (que podem ser doubles, inteiros, lógicos, texto,...).

A função `c()` pode ser usada para criar vetores. Por ex. vamos criar o vetor seguinte de elementos do tipo texto:

```
# elementos do tipo texto
c("azul","amarelo","roxo")
```

```
## [1] "azul"    "amarelo" "roxo"
```

Outros exemplos:

```
# elementos do tipo double
c(18,18.2,18.5,19.3,20.2)
```

```
## [1] 18.0 18.2 18.5 19.3 20.2
```

```
# elementos do tipo logico
c(T,F,T,F,F,F)
```

```
## [1]  TRUE FALSE  TRUE FALSE FALSE FALSE
```

Vamos atribuir ao vetor texto o nome cores:

```
# elementos do tipo texto
cores = c("azul","amarelo","roxo")
```

Podemos concatenar vetores:

```
c(cores,"rosa")
```

```
## [1] "azul"    "amarelo" "roxo"    "rosa"
```

Para sequências de números pode usar-se o operador `:`

```
1:10
```

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

Caso a sequência tenha um início, um fim e um incremento específico, pode usar-se a função `seq()`:

```
seq(0,1,0.1)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

O R reconhece o símbolo especial NA para representar valores omissos.

```
c("Rita","Sara","Nuno", NA)
```

```
## [1] "Rita" "Sara" "Nuno" NA
```

Operações com vectores numéricos

As operações com vectores numéricos são realizadas elemento a elemento. Por exemplo, no caso de `x` ser um vector, `log(x)` gera um vector com o logaritmo dos elementos de x:

```
x <- 1:2
log(x)
```

```
## [1] 0.0000000 0.6931472
```

No caso de dois vetores `x` e `y` da mesma dimensão ou ordem, `x + y` gera um vector com elementos iguais à soma dos elementos correspondentes de `x` e `y`.

Se `y` for um número é adicionado a cada elemento do vector x:

```
y <- 3:4
x + y
```

```
## [1] 4 6
```

Se os vectores x e y tiverem diferentes dimensões, o vector de menor dimensão é repetido (total ou parcialmente) até ficar com o mesmo número de elementos do maior (neste caso é emitido um aviso).

```
a <- c(1,2)
b <- c(1,3,2)
c = a + b
```

```
## Warning in a + b: longer object length is not a multiple of shorter ob
ject
## length
```

```
d = a * b
```

```
## Warning in a * b: longer object length is not a multiple of shorter ob
ject
## length
```

Atenção: Os objectos `vector` (e `matrix`) do R não são exactamente o mesmo que vetor e matriz definidas na Álgebra Linear. Por exemplo, em Álgebra a soma de vetores só está definida para vetores de igual dimensão, enquanto no R é possível somar vetores com diferentes dimensões.

Algumas funções com vectores

O número de elementos de um vector é devolvido pela função `length()`.

```
length(b)
```

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

Pode aceder-se aos elementos individuais do vector usando índices (*subscripts*) dentro de `[ ]`. Por ex. `b[1]` é o primeiro elemento de b, `b[2]` o segundo, ...

```
b[1]          # elemento 1 de b
```

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

```
b[2]          # elemento 2 de b
```

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

```
b[length(b)]  # último elemento de b
```

```
## [1] 2
```

O índice pode também ser um vector. Por ex. `b[1:2]` devolve os elementos de `b` nas posições 1 e 2.

```
b[1:2]
```

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

Um índice negativo exclui o elemento correspondente, devolvendo todos os elementos excepto o indicado.

```
b[-2]
```

```
## [1] 1 2
```

As funções `sum()` e `prod()` são usadas para calcular a soma e o produto dos elementos de um vector numerico.

Se houver elementos omissos (NA) no vector, a função não dá o resultado desejado. Nesses casos, é necessário adicionar à função o argumento `na.rm=TRUE`, para os elementos omissos não serem considerados.

```
sum(b)     # soma os elementos de b
```

```
## [1] 6
```

```
prod(b)    # multiplica os elementos de b
```

```
## [1] 6
```

```
b2 <- c(b,NA) # novo vetor com 1 elemento NA

# resultado indesejado
sum (b2)
```

```
## [1] NA
```

```
prod (b2)
```

```
## [1] NA
```

```
# com na.rm=TRUE
sum (b2, na.rm = TRUE)
```

```
## [1] 6
```

```
prod (b2, na.rm = TRUE)
```

```
## [1] 6
```

3.2 Matrizes

A função `matrix()` permite criar uma matriz $n \times p$.

Por ex. criar uma matriz 3 x 3 cujos termos são valores 1 até 9 (em coluna, por defeito).

```
M = matrix(1:9, nrow=3, ncol=3)
M
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Se se pretender que os valores sejam preenchidos em linha, é necessário adicionar o argumento byrow=TRUE.

```
M = matrix(1:9, nrow=3, ncol=3, byrow=TRUE)
M
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

Pode aceder-se a um termo especifico da matriz indicando M[*n linha*, *n coluna*]. Por ex. M[2,1] devolve o valor da linha 2 e coluna 1 da matriz M:

```
M[2,1]
```

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

Caso se queira a linha completa, deixamos o índice de coluna em branco. Por ex. M[2,] devolve os valores da linha 2:

```
M[2,]
```

```
## [1] 4 5 6
```

Raciocinio analogo aplica-se no caso das colunas. Por ex. M[,1] devolve todos os valores associados à primeira coluna.

```
M[,1]
```

```
## [1] 1 4 7
```

Qualquer dos índices pode ser um vector de elementos. Por ex. M[1:2,1:2] devolve uma matriz com os elementos das linhas 1 e 2 e colunas 1 e 2.

```
M[1:2,1:2]
```

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

O número de linhas e colunas de uma matriz são devolvidas pelas funções `ncol()` e `nrow()`.

```
nrow(M)
```

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

```
ncol(M)
```

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

Para realizar operações de algebra linear com vetores e matrizes, o R tem um conjunto de funções básicas implementadas. Um resumo dessas funções pode ser consultado no site https://www.statmethods.net/advstats/matrix.html.

3.3 Data-frames

Os dataframes permitem reunir coleções de objectos de diferentes tipos de dados (ex. vetores numericos, logicos, texto).

No R gera-se um data-frame usando a função `data.frame()`.

```
df1 = data.frame(num=c(3,1,2), text=c("lis","bru","lon"), logi=c(T,T,F))
```

É necessário que os vetores tenham todos o mesmo comprimento. Cada vetor é uma componente do data-frame e pode ser referido de diferentes formas:

- usando o simbolo `$` (*nome_data_frame*$*nome_componente*),
- usando índices dentro de duplo parentesis rectos `[[]]`,
- usando índices dentro de `[]` (como nos vectores).

Por ex. o vetor text (de indice 2):

```
df1[[2]]
```

```
## [1] lis bru lon
## Levels: bru lis lon
```

```
df1$text
```

```
## [1] lis bru lon
## Levels: bru lis lon
```

```
df1[,2]
```

```
## [1] lis bru lon
## Levels: bru lis lon
```

Cada elemento do data-frame e pode ser referido de diferentes formas. Por ex. o elemento 2 do vetor `num`:

```
df1$num[2]
```

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

```
df1[2,1]
```

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

```
df1[[1]][2]

## [1] 1
```

Deve evitar-se criar data-frames de grandes dimensões à mão (são um convite ao erro). Uma boa alternativa é obter os dados via ficheiro de dados estruturado, que possa ser importado para o R. Por exemplo, a linha de comando seguinte importa o ficheiro de texto data.txt (descarregar o ficheiro data.txt do fenix):

```
df2 <- read.table("data.txt", header = TRUE, sep = "\t", dec = ".")
```

No RStudio os ficheiros podem ainda ser importados usando o icone *Import Dataset*.

3.4 Listas

As listas são parecidas com os vetores, mas em vez de agruparem valores, agrupam objetos R (vetores, data-frames, outras listas …).

Para gerar uma lista usa-se a função `list()`.

Por exemplo, podemos juntar numa lista um vetor numérico (doubles, inteiro) de comprimento 5 com um vetor texto de comprimento 2 e um vetor logico com comprimento 3:

```
lista <- list(c(3,2,1,1,8), c("T","R"), c(F, T, F))
lista

## [[1]]
## [1] 3 2 1 1 8
##
## [[2]]
## [1] "T" "R"
##
## [[3]]
## [1] FALSE  TRUE FALSE
```

Os índices dentro de duplo parentesis rectos `[[]]` indicam o elemento da lista, enquanto que o parentesis recto `[]` indica o indice do sub-elemento. Por exemplo, 3 é o primeiro sub-elemento do primeiro elemento da lista, “T” é o primeiro sub-elemento do segundo elemento da lista.