Aprendendo R

Magno TF Severino

2023-12-06

Contents

Chapter 1

Intro

Este livro compila tutoriais da linguagem `R` que usei em diversas aulas que dei ao longo dos últimos anos.

Chapter 2

Básicos da linguagem R

Para instalação,

- faça o download do R em http://www.r-project.org;
- sugestão: utilizar a IDE R Studio.

É muito importante saber como obter ajuda no R. Sempre que estiver em dúvidas quanto as características de alguma função, consule a aba *help* do R.

```
?mean #abre a página de ajuda da função 'mean'
??plot #procura por tópicos contendo a palavra 'plot'
```

2.1 Usando o R como uma calculadora

O operador + realiza a adição entre dois elementos.

```
1 + 2
```

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

Um vetor é um conjunto ordenado de valores. O operador : cria uma sequência a partir de um número até outro. A função c concatena valores, criando um vetor.

```
1:5
```

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

```
c(1, 2, 3, 4, 5)
```

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

Além de adicionar dois números, o operador + pode ser usado para adicionar dois vetores.

```
1:5 + 6:10
```

```
## [1]  7  9 11 13 15
```

```
c(1, 2, 3, 4, 5) + c(6, 7, 8, 9, 10)
```

```
## [1]  7  9 11 13 15
```

```
1:5 + c(6, 7, 8, 9, 10)
```

```
## [1]  7  9 11 13 15
```

Os próximos exemplos mostram subtração, multiplicação, exponenciação e divisão.

```
c(2, 3, 5, 7, 11, 13) - 2        #subtração
```

```
## [1]  0  1  3  5  9 11
```

```
-2:2 * -2:2                      #multiplicação
```

```
## [1] 4 1 0 1 4
```

```
(1:10) ^ 2                         #exponenciação
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```
1:10 / 3                         #divisão
```

```
##  [1] 0.3333333 0.6666667 1.0000000 1.3333333 1.6666667 2.0000000 2.3333333
##  [8] 2.6666667 3.0000000 3.3333333
```

2.2 Atribuindo variáveis

Fazer cálculos com o R é bem simples e útil. A maior parte das vezes queremos armazenar os resultados para uso posterior. Assim, podemos atribuir valor à uma variável, através do operador <-.

```
a <- 1

b <- 5 * 3

x <- 1:5

y <- 6:10
```

Agora, podemos reutilizar esses valores para fazer outros cálculos.

```
a + 2 * b
```

```
## [1] 31
```

```
x + 2 * y - 3
```

```
## [1] 10 13 16 19 22
```

Observe que não temos que dizer ao R qual o tipo da variável, se era um número (as variáveis a e a) ou vetor (x e y).

2.3 Números especias

Para facilitar operações aritméticas, R suporta quatro valores especiais de números: Inf, -Inf, NaN e NA. Os dois primeiros representam infinito positivo e negativo. NaN é um acrônimo inglês para "not a number", ou seja, não é um número. Ele aparece quando um cálculo não faz sentido, ou não está definido. NA significa "not available", ou seja, não disponível, e representa um valor faltante.

```
c(Inf + 1, Inf - 1, Inf - Inf, NA + 1)
```

```
## [1] Inf Inf NaN  NA
```

```
c(0 / 0, Inf / Inf, 1 / Inf)
```

```
## [1] NaN NaN   0
```

2.4 Vetores, Matrizes e Dataframes

Previamente, vimos alguns tipos de vetores para valores lógicos, caracteres e números. Nessa seção, utilizaremos técnicas de manipulação de vetores e introduziremos o caso multidimensional: matrizes e dataframes.

Abaixo relembramos as operações que já foram feitas com vetores

```
10:5                    #sequência de números de 10 até 5
```

```
## [1] 10  9  8  7  6  5
```

```
c(1, 2:5, c(6, 7), 8)   #valores concatenados em um único vetor
```

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

2.4.1 Vetores

Existem funções para criar vetores de um tipo e com tamanho específicos. Todos os elementos deste vetor terá valor zero, FALSE, um caracter vazio, ou o equivalente à *nada/vazio* para aquele tipo. Veja abaixo duas maneiras de definir um vetor.

```
vector("numeric", 5) #cria um vetor numérico de 5 elementos
```

```
## [1] 0 0 0 0 0
```

```
numeric(5) #equivalente ao comando acima
```

```
## [1] 0 0 0 0 0
```

```
vector("logical", 5) #cria um vetor lógico de 5 elementos
```

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

```r
logical(5) #equivalente ao comando acima
```

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

```r
vector("character", 5) #cria um vetor de caracteres de 5 elementos
```

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

```r
character(5) #equivalente ao comando acima
```

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

2.4.1.1 Sequências

Podemos criar sequências mais gerais que aquelas criadas com o operador :. A função seq te permite criar sequências em diferentes maneiras Veja abaixo.

```r
seq(3, 12) #equivalente à 3:12
```

```
##  [1]  3  4  5  6  7  8  9 10 11 12
```

```r
seq(3, 12, 2) #o terceiro argumento indica a distância entre os elementos na lista.
```

```
## [1]  3  5  7  9 11
```

```r
seq(0.1, 0.01, -0.01)
```

```
##  [1] 0.10 0.09 0.08 0.07 0.06 0.05 0.04 0.03 0.02 0.01
```

2.4.1.2 Tamanhos

Todo vetor tem um tamanho, um número não negativo que representa a quantidade de elementos que o vetor contém. A função length retorna o tamanho de um dado vetor.

```r
length(1:5)
```

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

```
frase <- c("Observe", "o", "resultado", "dos", "comandos", "abaixo")

length(frase)
```

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

```
nchar(frase)
```

```
## [1] 7 1 9 3 8 6
```

2.4.1.3 Indexando vetores

A indexação é útil quando queremos acessar elementos específicos de um vetor. Considere o vetor

```
x <- (1:5) ^ 2
```

Abaixo, três métodos de indexar os mesmos valores do vetor x.

```
x[c(1, 3, 5)]

x[c(-2, -4)]
```

```
x[c(TRUE, FALSE, TRUE, FALSE, TRUE)]
```

```
## [1]  1  9 25
```

Se nomearmos os elementos do vetor, o método abaixo obtém os mesmos valores de x.

```
names(x) <- c("one", "four", "nine", "sixteen", "twenty five")
x[c("one", "nine", "twenty five")]
```

```
##         one        nine twenty five
##           1           9          25
```

Cuidado, acessar um elemento fora do tamanho do vetor não gera um erro no R, apenas NA.

```
x[6]
```

```
## <NA>
##   NA
```

2.4.2 Matrizes

Uma matriz é o equivalente à um vetor, porém em duas dimensões. Abaixo, um exemplo de definição de uma matriz com 4 linhas e 3 colunas (total de 12 elementos).

```
?matrix
```

```
uma_matriz <- matrix(
      1:12,
      nrow = 4, #ncol = 3 gera o mesmo resultado. Verifique!
      dimnames = list(
        c("L1", "L2", "L3", "L4"),
        c("C1", "C2", "C3")
      )
)

class(uma_matriz)
```

```
## [1] "matrix" "array"
```

```
uma_matriz
```

```
##    C1 C2 C3
## L1  1  5  9
## L2  2  6 10
## L3  3  7 11
## L4  4  8 12
```

Por padrão, ao criar uma matrix, o vetor passado como primeiro argumento preenche a matrix por colunas. Para preencher a matrix por linhas, basta especificar o argumento byrow=TRUE

A função dim retorna um vetor de inteiros com as dimensões da variável.

```
dim(uma_matriz)
```

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

```
nrow(uma_matriz) #retorna o número de linhas da matrix
```

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

```
ncol(uma_matriz) #retorna o número de colunas da matrix
```

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

```
length(uma_matriz) #retorna o número de elementos da matrix
```

```
## [1] 12
```

2.4.2.1 Nomeando linhas, columas e dimensões

Da mesma forma para vetores, podemos nomear (e obter os nomes de) linhas e colunas de matrizes.

```
rownames(uma_matriz)
```

```
## [1] "L1" "L2" "L3" "L4"
```

```
colnames(uma_matriz)
```

```
## [1] "C1" "C2" "C3"
```

```
dimnames(uma_matriz)
```

```
## [[1]]
## [1] "L1" "L2" "L3" "L4"
##
## [[2]]
## [1] "C1" "C2" "C3"
```

2.4.2.2 Indexação

A indexação de matrizes funciona de maneira similar à de vetores, com a diferença que agora precisam ser especificadas mais de uma dimensão.

```
uma_matriz[1, c("C2", "C3")] #elementos na primeira linha, segunda e terceira colunas
```

```
## C2 C3
##  5  9
```

```
uma_matriz[1, ] #todos elementos da primeira linha
```

```
## C1 C2 C3
##  1  5  9
```

```
uma_matriz[, c("C2", "C3")] #todos elementos da segunda e terceira colunas
```

```
##    C2 C3
## L1  5  9
## L2  6 10
## L3  7 11
## L4  8 12
```

```
uma_matriz[, c(2, 3)] #todos elementos da segunda e terceira colunas
```

```
##    C2 C3
## L1  5  9
## L2  6 10
## L3  7 11
## L4  8 12
```

2.4.2.3 Combinando matrizes

Considere a seguinte matriz.

```
outra_matriz <- matrix(
      seq(2, 24, 2),
      nrow = 4,
      dimnames = list(
        c("L5", "L6", "L7", "L8"),
        c("C5", "C6", "C7")
      ))
```

A combinação de matrizes pode ser feita através das funções cbind e rbind, que combina matrizes por colunas e por linhas, respectivamentes.

```
cbind(uma_matriz, outra_matriz)
```

```
##    C1 C2 C3 C5 C6 C7
## L1  1  5  9  2 10 18
## L2  2  6 10  4 12 20
## L3  3  7 11  6 14 22
## L4  4  8 12  8 16 24
```

```
rbind(uma_matriz, outra_matriz)
```

```
##    C1 C2 C3
## L1  1  5  9
## L2  2  6 10
## L3  3  7 11
## L4  4  8 12
## L5  2 10 18
## L6  4 12 20
## L7  6 14 22
## L8  8 16 24
```

2.4.2.4 Operações com matrizes

As operações básicas (+, -, *, /) funcionam de elemento a elemento em matrizes, da mesmo forma como em vetores:

```
uma_matriz + outra_matriz
```

```
##    C1 C2 C3
## L1  3 15 27
## L2  6 18 30
## L3  9 21 33
## L4 12 24 36
```

```
uma_matriz * outra_matriz
```

```
##    C1  C2  C3
## L1  2  50 162
## L2  8  72 200
## L3 18  98 242
## L4 32 128 288
```

Cuidado: as matrizes e vetores devem ter tamanhos compativeis!