Introdução à programação com R Filtros





Tópicos desta aula

- Operadores
- Mais sobre Data.frames



Operadores



Operações lógicas

Uma operação lógica é um teste que retorna **verdadeiro** ou **falso**. No R (e em outras linguagens de programação), esses valores dois valores recebem uma classe especial: logical.

O verdadeiro no R vai ser representado pelo valor TRUE e o falso pelo valor FALSE. Esses nomes no R são **reservados**, isto é, você não pode chamar nenhum objeto de TRUE ou FALSE.

```
TRUE <- 1
## Error in TRUE <- 1 : invalid (do_set) left-hand side to assignment
```



Valores lógicos

Checando a classe desses valores, vemos que são lógicos (também conhecidos como valores binários ou booleanos). Eles são os únicos possíveis valores dessa classe.

```
class(TRUE)

## [1] "logical"

class(FALSE)

## [1] "logical"
```

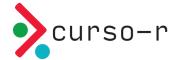
Agora que conhecemos o TRUE e FALSE, podemos explorar os teste lógicos.



Igualdades

Começando pela igualdade: vamos testar se um valor é igual ao outro. Para isso, usamos o operador ==.

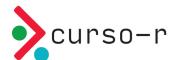
```
# Testes com resultado verdadeiro
1 == 1
## [1] TRUE
"a" == "a"
## [1] TRUE
# Testes com resultado falso
1 == 2
## [1] FALSE
"a" == "b"
## [1] FALSE
```



Diferenças

Também podemos testar se dois valores são diferentes. Para isso, usamos o operador !=.

```
# Testes com resultado falso
1 != 1
## [1] FALSE
"a" != "a"
## [1] FALSE
# Testes com resultado verdadeiro
1 != 2
## [1] TRUE
"a" != "b"
## [1] TRUE
```



Desigualdades

Para comparar se um valor é maior que outro, temos à disposição 4 operadores:

```
# Maior
3 > 3
## [1] FALSE
3 > 2
## [1] TRUE
# Maior ou igual
3 > 4
## [1] FALSE
3 >= 3
## [1] TRUE
```



```
# Menor
3 < 3
## [1] FALSE
3 < 4
## [1] TRUE
# Menor ou igual
3 < 2
## [1] FALSE
3 <= 3
## [1] TRUE
```



Pertence

Um outro operador muito útil é o %in%. Com ele, podemos verificar se um valor está dentro de um conjunto de valores (vetor).

```
3 %in% c(1, 2, 3)

## [1] TRUE

"a" %in% c("b", "c")

## [1] FALSE
```



Filtros

Os testes lógicos fazem parte de uma operação muito comum na manipulação de base de dados: os **filtros**. No Excel, por exemplo, quando você filtra uma planilha, o que está sendo feito por trás é um teste lógico.

Falamos anteriormente que cada coluna das nossas bases de dados será representada dentro do R como um vetor. O comportamento que explica a importância dos testes lógicos na hora de filtrar uma base está ilustrado abaixo:

```
minha_coluna <- c(1, 3, 0, 10, -1, 5, 20)
minha_coluna > 3

## [1] FALSE FALSE TRUE FALSE TRUE TRUE

minha_coluna[minha_coluna > 3]

## [1] 10 5 20
```



Muitas coisas aconteceram no código anterior, vamos por partes.

Primeiro, na operação minha_coluna > 3 o R fez um excelente uso do comportamento de reciclagem. No fundo, o que ele fez foi transformar (reciclar) o valor 3 no vetor c(3, 3, 3, 3, 3, 3, 3) e testar se c(1, 3, 0, 10, -1, 5, 20) > c(3, 3, 3, 3, 3, 3, 3).

Como os operadores lógicos também são vetorizados (fazem operações elemento a elemento), os testes realizados foram 1 > 3, 3 > 3, 0 > 3, 10 > 3, -1 > 3, 5 > 3 e, finalmente, 20 > 3. Cada um desses testes tem o seu próprio resultado. Por isso a saída de minha_coluna > 3 é um vetor de verdadeiros e falsos, respectivos a cada um desses 7 testes.

A segunda operação traz a grande novidade aqui: podemos usar os valores TRUE e FALSE para selecionar elementos de um vetor!

A regra é a seguinte: **retornar** as posições que receberem TRUE, **não retornar** as posições que receberem FALSE.



Portanto, a segunda operação é equivalente a:

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

```
## [1] 10 5 20
```

O vetor lógico filtra o vetor minha_coluna, retornando apenas os valores maiores que 3, já que foi esse o teste lógico que fizemos.

Essa é a *mágica* que acontece por trás de filtros no R. Na prática, não precisaremos usar colchetes, não lembraremos da reciclagem e nem veremos a cara dos TRUE e FALSE. Mas conhecer esse processo é muito importante, principalmente para encontrar problemas de código ou de base.



Mais sobre data frames

Chegou a hora de usarmos tudo o que aprendemos na seção anterior para exploramos ao máximo o nosso objeto favorito: o *data frame*.

Para isso, continuaremos a usar o mtcars.

```
mtcars
```

```
##
                    mpg cyl disp hp drat
                                            wt qsec vs am gear carb
## Mazda RX4
                   21.0
                         6 160.0 110 3.90 2.620 16.46
                   21.0
## Mazda RX4 Wag
                         6 160.0 110 3.90 2.875 17.02 0 1
## Datsun 710
                   22.8
                         4 108.0 93 3.85 2.320 18.61 1 1
                   21.4
## Hornet 4 Drive
                         6 258.0 110 3.08 3.215 19.44 1 0
## Hornet Sportabout 18.7
                         8 360.0 175 3.15 3.440 17.02 0 0
## Valiant
                   18.1
                         6 225.0 105 2.76 3.460 20.22 1 0
                   14.3
                         8 360.0 245 3.21 3.570 15.84 0 0
## Duster 360
                         4 146.7 62 3.69 3.190 20.00 1 0 4
                   24.4
## Merc 240D
                   22.8
                         4 140.8 95 3.92 3.150 22.90 1 0
## Merc 230
                   19.2
## Merc 280
                         6 167.6 123 3.92 3.440 18.30 1 0
```



Acessando as colunas

Lembrando que cada coluna de um *data frame* é um vetor, podemos usar o operador \$ para acessar cada uma de suas colunas.



Dimensões

A classe *data frame* possui uma característica especial: seus objetos possuem duas **dimensões**.

```
class(mtcars)

## [1] "data.frame"

dim(mtcars)
```

[1] 32 11

O resultado do código dim(mtcars) nos diz que a primeira dimensão tem comprimento 32 e a segunda dimensão tem comprimento 11. Em outras palavras: a base mtcars tem 32 linhas e 11 colunas.



Subsetting

Ter duas dimensões significa que devemos usar dois índices para acessar os valores de um *data frame* (fazer *subsetting*). Para isso, ainda usamos o colchete, mas agora com dois argumentos: [linha, coluna].

```
mtcars[2, 3]
```

[1] 160

O código acima está nos devolvendo o valor presente na segunda linha da terceira coluna da base mtcars.



Também podemos pegar todos as linhas de uma coluna ou todas as colunas de uma linha deixando um dos argumentos vazio:

```
# Todas as linhas da coluna 1
mtcars[,1]

## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4 10.4
## [17] 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4

# Todas as colunas da linha 1
mtcars[1,]

## mpg cyl disp hp drat wt qsec vs am gear carb
## Mazda RX4 21 6 160 110 3.9 2.62 16.46 0 1 4 4
```



Selecionando colunas

Podemos usar o *subsetting* para selecionar colunas:

```
mtcars[, c(1, 2)]
```

```
##
                  mpg cyl
                  21.0 6
## Mazda RX4
## Mazda RX4 Wag 21.0 6
## Datsun 710 22.8 4
## Hornet 4 Drive
                 21.4 6
## Hornet Sportabout 18.7 8
## Valiant
                  18.1 6
## Duster 360
               14.3 8
                24.4 4
## Merc 240D
## Merc 230 22.8 4
          19.2
## Merc 280
```



```
mtcars[, c("mpg", "am")]
```

```
##
                    mpg am
## Mazda RX4
                   21.0
## Mazda RX4 Wag
                   21.0 1
## Datsun 710
               22.8 1
## Hornet 4 Drive 21.4
## Hornet Sportabout 18.7 0
## Valiant
                   18.1
## Duster 360
                   14.3
## Merc 240D
                   24.4
## Merc 230
                   22.8
## Merc 280
                   19.2 0
```

Nos dois exemplos, exibimos apenas as 5 primeiras linhas do data frame.



Filtrando linhas

Também podemos usar o subsetting para filtrar linhas:

```
mtcars$cyl == 4
   [1] FALSE FALSE
                  TRUE FALSE FALSE FALSE TRUE
                                                    TRUE FALSE FALSE FALSE
  [14] FALSE FALSE FALSE TRUE
                                        TRUE TRUE FALSE FALSE FALSE TRUE
                                  TRUE
## [27]
       TRUE
            TRUE FALSE FALSE FALSE
                                   TRUE
mtcars[mtcars$cyl == 4, ]
##
                 mpg cyl disp
                               hp drat
                                         wt qsec vs am gear carb
## Datsun 710
                22.8
                      4 108.0
                               93 3.85 2.320 18.61
                24.4
## Merc 240D
                      4 146.7
                               62 3.69 3.190 20.00
                22.8
## Merc 230
                      4 140.8
                               95 3.92 3.150 22.90 1 0
## Fiat 128
                32.4 4 78.7
                              66 4.08 2.200 19.47 1 1
## Honda Civic
                30.4 4 75.7
                               52 4.93 1.615 18.52
## Toyota Corolla 33.9
                      4 71.1
                              65 4.22 1.835 19.90
## Toyota Corona 21.5
                      4 120.1
                               97 3.70 2.465 20.01
## Fiat X1-9
                27.3
                      4 79.0
                               66 4.08 1.935 18.90 1 1
## Porsche 914-2 26.0
                      4 120.3
                              91 4.43 2.140 16.70 0 1
## Lotus Europa
                30.4
                      4 95.1 113 3.77 1.513 16.90 1 1
## Volvo 142E
                21.4
                      4 121.0 109 4.11 2.780 18.60 1 1
```



O código mtcars\$cyl == 4 nos diz em quais linhas estão os carros com 4 cilindros. Quando usamos o vetor de TRUE e FALSE resultante dentro do *subsetting* das linhas em mtcars[mtcars\$cyl == 4,], o R nos devolve todos as colunas dos carros com 4 cilindros. A regra é a seguinte: linha com TRUE é retornada, linha com FALSE não.

Outro exemplo:

