Operadores lógicos e instruções condicionais

Anteriormente, você aprendeu que um **operador** é um símbolo que identifica o tipo de operação ou cálculo a ser realizado em uma fórmula. Durante a leitura, você aprenderá sobre os principais tipos de operadores lógicos em R e como usá-los para criar instruções condicionais no código de R.

# Operadores lógicos

Os **operadores lógicos** retornam um tipo de dado lógico, como TRUE ou FALSE.

Eles se dividem em três principais tipos:

* AND (também representado como “&” ou “&&” em R)
* OR (também representado como “|” ou “||” em R)
* NOT (!)

Esta tabela resume os operadores lógicos:

|  |  |  |
| --- | --- | --- |
| **Operador AND “&”** | **Operador OR “|”** | **Operador NOT “!”** |
| O operador AND processa dois valores lógicos, retornando TRUE somente se *ambos* os valores individuais são TRUE. Isso significa que TRUE & TRUE é avaliado como TRUE.No entanto, FALSE & TRUE, TRUE & FALSE e FALSE & FALSE são todos avaliados como FALSE. | O operador OR (|) funciona de forma parecida a do operador AND (&). A principal diferença é que, ao menos, um dos valores da operação OR deve ser TRUE para que toda a operação resulte em TRUE. | O operador NOT (!) basicamente nega o valor lógico ao qual se aplica. Em outras palavras, !TRUE resulta em FALSE e !FALSE resulta em TRUE. |
| Ou seja, TRUE | TRUE, TRUE | FALSE e FALSE | TRUE resultam em TRUE. Quando ambos os valores são FALSE, o resultado é FALSE. |
| Se você executar o código correspondente em R, obterá os seguintes resultados: | Se você escrever o código, obterá os seguintes resultados: | Ao executar o código, você obtém os seguintes resultados: |
| > TRUE & TRUE | > TRUE | TRUE | > !TRUE |
| [1] TRUE | [1] TRUE | [1] FALSE |
| > TRUE & FALSE | > TRUE | FALSE | > !FALSE |
| [1] FALSE | [1] TRUE | [1] TRUE |
| > FALSE & TRUE | > FALSE | TRUE |  |
| [1] FALSE | [1] TRUE | Assim como os operadores OR e AND, você pode usar o operador NOT em combinação com operadores lógicos. O zero é considerado FALSE, e os demais números são considerados |
| > FALSE & FALSE | > FALSE | FALSE |
| [1] FALSE | [1] FALSE |

|  |  |  |
| --- | --- | --- |
| Você pode ilustrar isso usando os resultados de nossas comparações. Imagine que você cria uma variável *x* que é igual a 10.  x <- 10  Para verificar se “x” é maior que 3, mas menor que 12, você pode usar x > 3 e x < 12 como os valores de uma expressão “AND”.  x > 3 & x < 12  Quando você executa a função, R retorna o resultado TRUE.  [1] TRUE  A primeira parte, x > 3 será avaliada como TRUE, pois 10 é maior que 3. A segunda parte, x < 12, também resultará em TRUE, já que 10 é menor do que 12. Portanto, como *ambos* os valores são TRUE, o resultado da expressão AND é TRUE. O número 10 está entre os números 3 e 12.  No entanto, se você fizer “x” igual a 20, a expressão x > 3 & x < 12 retornará um resultado diferente. | Por exemplo, suponha que você crie uma variável *y* igual a 7. Para verificar se *y* é menor que 8 ou maior que 16, você pode usar a seguinte expressão:  y <- 7  y < 8 | y > 16  O resultado da comparação é TRUE (7 é menor que 8) | FALSE (7 não é maior que 16). Como somente um valor da expressão OR precisa ser TRUE para que toda a expressão resulte em TRUE, R retorna um resultado TRUE.  [1] TRUE  Digamos que agora y é igual a 12. A expressão y < 8| y > 16 resulta em FALSE (12 < 8) | FALSE (12 > 16). As duas comparações são FALSE, portanto o resultado também é FALSE.  y <- 12  y < 8 | y > 16  [1] FALSE | como TRUE. O operador NOT resulta no valor lógico oposto.  Vamos imaginar que você tenha uma variável "x" igual a 2:  x <- 2  A operação NOT é avaliada como FALSE porque assume o valor lógico oposto de um número diferente de zero (TRUE).  > !x  [1] FALSE |

|  |  |  |
| --- | --- | --- |
| x <- 20  x > 3 & x < 12  [1] FALSE  Embora x > 3 seja TRUE (20 > 3), x < 12 é FALSE (20 < 12). Se uma parte da expressão AND resulta em FALSE, o mesmo valerá para toda a expressão (TRUE & FALSE = FALSE). Consequentemente, R retorna o resultado FALSE. |  |  |

Vamos ver um exemplo de como é possível usar operadores lógicos na análise de dados. Digamos que você está trabalhando com o conjunto de dados “airquality” pré-carregado em RStudio. O conjunto inclui dados sobre medidas de qualidade do ar de Nova York que vão de maio a setembro de 1973.

O data frame tem seis colunas: Ozone (a medida de ozônio), Solar.R (a medida solar), Wind (a medida do vento), Temp (a temperatura em Fahrenheit) e Month (mês) e Day (dia) das medidas (cada linha representa uma combinação específica de mês e dia).

Vamos conferir como os operadores AND, OR e NOT podem ser úteis nessa situação.

**Exemplo de AND**

Imagine que você quer especificar as linhas que são excessivamente ensolaradas e ventosas, que, segundo sua definição, têm medidas acima de 150 *e* 10 para as medições do sol e do vento, respectivamente.

Em R, você pode expressar a instrução lógica como Solar.R > 150 & Wind > 10. Somente as linhas com *ambas* as condições verdadeiras é que atendem aos critérios:

**Exemplo de OR**

Em seguida, imagine que você quer especificar as linhas em que está excessivamente ensolarado ou ventoso, que, segundo sua definição, têm medidas acima de 150 *ou 10* para Solar e Wind, respectivamente.

Em R, você pode expressar a instrução lógica como Solar.R > 150 | Wind > 10. Todas as linhas em que *alguma das duas* condições é verdadeira atendem aos critérios:

**Exemplo de NOT**

Agora imagine que você quer focar apenas em medidas climáticas para dias que *não* são o primeiro dia do mês. Em R, você pode expressar a instrução lógica como Day != 1.

As linhas em que a condição é verdadeira atendem aos critérios:

Por fim, imagine que você quer focar apenas em cenários que não sejam excessivamente ensolarados e não sejam excessivamente ventosos, com base em definições prévias de excessivamente ensolarado e excessivamente ventoso. Em outras palavras, a seguinte instrução *não* deve ser verdadeira: tanto uma medida Solar maior do que 150 *ou* uma medida Wind maior do que 10.

Observe que a instrução é o oposto da instrução OR acima. Para expressar a instrução em R, insira um ponto de exclamação (!) em frente à instrução OR anterior: !(Solar.R > 150 | Wind > 10). R aplicará o operador NOT a tudo o que estiver dentro dos parênteses.

Nesse caso, só uma linha atende aos critérios:

# Instruções condicionais

A **instrução condicional** é uma declaração de que caso haja uma determinada condição, um determinado evento deve ocorrer. Vejamos o exemplo: “*Se* a temperatura está superior ao congelamento, *então* eu sairei para caminhar”. Se a primeira condição é verdadeira (a temperatura está superior ao congelamento), então a segunda condição ocorrerá (eu sairei para caminhar). As instruções condicionais no código em R seguem uma lógica parecida.

Vamos ver como criar instruções condicionais em R com três instruções relacionadas:

* **if()**
* **else()**
* **else if()**

## Instrução if

A instrução **if** estipula uma condição; se a condição resultar em TRUE, o código em R associado à instrução if é executado.

Em R, insira o código da condição entre os parênteses da instrução if. O código que deve ser executado se a condição for TRUE se dá entre chaves (“expr”). Observe que, nesse caso, a segunda chave está na mesma linha do código e identifica o final do código a ser executado.

if (condition) { expr

}

Por exemplo, vamos criar uma variável “x” igual a 4.

x <- 4

Em seguida, vamos criar uma instrução condicional: se x é maior do que 0, então R gerará a string “x é um número positivo”.

if (x > 0) {

print("x é um número positivo")

}

Como x = 4, a condição é verdadeira (4 > 0). Portanto, ao executar o código, R gera a string “x é um número positivo”.

[1] "x é um número positivo"

Se, no entanto, você mudar x para um número negativo, como -4, a condição será falsa (FALSE) (-4 > 0). Ao executar o código, R não executará a instrução print; como resultado, no entanto, aparecerá uma linha em branco.

## Instrução else

A instrução **else** é usada juntamente com uma instrução **if**. É assim que o código é estruturado em R:

if (condition) {

expr1

} else {

expr2

}

O código associado à instrução else é executado sempre que a condição da instrução if *não for* TRUE. Traduzindo, se a condição resultar em TRUE, R executará o código na instrução if (“expr1”); se a condição *não* resultar em TRUE, R executará o código na instrução else (“expr2”).

Vamos testar um exemplo. Primeiramente, crie uma variável “x” igual a 7.

x <- 7

Em seguida, defina as seguintes condições:

* Se x é maior do que 0, R gerará “x é um número positivo”.
* Se x é menor ou igual a 0, R gerará “x é um número negativo ou zero”.

A primeira condição do nosso código (x > 0) será parte da instrução if. A segunda condição (x é menor ou igual a 0), por sua vez, está inferida na instrução else. Se x > 0, então R gerará “x é um número positivo”. Caso contrário, R gerará “x é um número negativo ou zero”.

x <- 7

if (x > 0) {

print("x é um número positivo")

} else {

print ("x é um número negativo ou zero")

}

Como 7 é maior do que 0, a condição da instrução if é verdadeira (TRUE). Portanto, ao executar o código, R gera “x é um número positivo”.

[1] "x é um número positivo"

No entanto, se você definir que x é igual a -7, a condição da instrução if *não* será verdadeira (TRUE) (-7 não é maior do que 0). Assim sendo, R executará o código na instrução else. Ao executar o código, R gera “x é um número positivo ou zero”.

x <- -7

if (x > 0) {

print("x é um número positivo")

} else {

print ("x é um número negativo ou zero")

}

[1] "x é um número negativo ou zero"

## Instrução else if

Em alguns casos, você pode personalizar sua instrução condicional ainda mais com uma instrução **else if**. Essa instrução se dá entre a instrução if e a instrução else.

Vejamos a estrutura do código:

if (condition1) {

expr1

} else if (condition2) {

expr2

} else {

expr3

}

Se a condição if (“condition1”) for atendida, R executará o código na primeira expressão (“expr1”). Se esse não for o caso, mas a condição else if (“condition2”) for atendida, R executará o código na segunda expressão (“expr2”). Se nenhuma das duas condições forem atendidas, R executará o código na terceira expressão (“expr3”).

Em nosso outro exemplo, que usa somente as instruções if e else, R somente gera “x é um número negativo ou zero” se x é igual ou menor do que 0. Digamos que você quer que R gere a string “x é zero” se x for igual a 0. É preciso adicionar outra condição usando a instrução else if.

Vamos testar um exemplo. Primeiramente, crie uma variável “x” igual a 1 negativo (“-1”).

x <- -1

Agora você quer definir as seguintes condições:

* Se x for menor do que 0, gerar “x é um número negativo”.
* Se x for igual a 0, gerar “x é zero”.
* Caso contrário, gera “x é um número positivo”.

A primeira condição do nosso código fará parte da instrução if, enquanto a segunda e a terceira condição farão parte das instruções else if e else, respectivamente. Se x < 0, então R gerará “x é um número positivo”. Se x = 0, então R gerará “x é zero”. Caso contrário, R gerará “x é um número positivo”.

x <- -1

if (x < 0) {

print("x é um número negativo")

} else if (x == 0) {

print("x é zero")

} else {

print("x é um número positivo")

}

Como -1 é menor do que 0, a condição da instrução if resulta em TRUE e R gera “x é um número negativo”.

[1] "x é um número negativo"

Se você definir que x é igual a 0, R irá primeiro verificar a condição if (x <0) e, então, determinar que é FALSE e avaliar a condição else if. Essa condição, x ==0, é TRUE. Por isso, nesse caso, R gera “x é zero”.

Se você definir que x é igual a 1, ambas as condições if e else resultam em FALSE. Assim, R irá executar a instrução else e gerar “x é um número positivo”.

Assim que R identificar uma condição que resulte como TRUE, ele irá executar o código correspondente e ignorar o restante.

# Recursos

Para saber mais sobre operadores lógicos e instruções condicionais, confira o tutorial sobre [“Conditionals and Control Flow in R” no](https://www.datacamp.com/community/tutorials/conditionals-and-control-flow-in-r) [site do DataCamp](https://www.datacamp.com/community/tutorials/conditionals-and-control-flow-in-r). DataCamp é um recurso conhecido usado por quem busca aprender sobre programação de computadores. O tutorial é repleto de exemplos bacanas de aplicações de programação para operadores lógicos e instruções condicionais (e operadores relacionais) e ainda traz uma visão geral importante de cada tópico e as correlações entre eles.