Logical Expressions

Marcella Pedro

29/09/2021

Expressões Lógicas

No script anterior *Data Types*, vimos o tipo de dados lógico `TRUE` ou `FALSE`. Aqui vamos ver como combinar esses tipos com **Operadores Lógicos e Relacionais**

Operadores Relacionais

Muitas vezes ao programar é necessário comparar a relação entre duas variáveis, entre 2 valores, se são iguais, diferentes, maior ou menor, e por aí vai. O retorno da expressão pode ser `TRUE` ou `FALSE`.

Operador	Significado
x == y	x igual a y?
x != y	x diferente de y?
x > y	x maior que y?
x >= y	x maior ou igual a y?
x < y	x menor que y?
x <= y	x menor ou igual a y?

Segue abaixo na prática alguns exemplos:

```
1 == 2 # 1 não é igual a 2
```

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

```
1 != 2 # 1 é diferente de 2
```

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

```
1 > 2 # 1 não é maior que 2
```

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

```
1 >= 2 # 1 não é maior nem igual a 2
```

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

```r
1 < 2 # 1 é menor que 2
```

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

```r
1 <= 2 # 1 é menor que 2 (não é igual, mas o resultado é true pois a pergunta é se é menor OU igual)
```

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

Operadores Lógicos

Existem 3 operadores lógicos:

- NOT (NÃO) -> representado pelo símbolo !
- AND (E) -> representado pelo símbolo &&
- OR (OU) -> representado pelo símbolo ||

Sendo assim podemos ver expressões como `!x`, `x && y`, e `x||y`. X e Y vão carregar valores como `TRUE` ou `FALSE`, então no lugar do x e y podemos ler !FALSE, TRUE && TRUE, e assim por diante, depende do valor que a variável carrega.

Vamos ver a seguir uma tabela que ilustra o resultado das combinações:

Expressão NOT	Resultado
`!TRUE`	`FALSE`
`!FALSE`	`TRUE`

Expressão AND	Resultado
`TRUE && TRUE`	`TRUE`
`TRUE && FALSE`	`FALSE`
`FALSE && TRUE`	`FALSE`
`FALSE && FALSE`	`FALSE`

Expressão OR	Resultado
`TRUE || TRUE`	`TRUE`
`TRUE || FALSE`	`TRUE`
`FALSE || TRUE`	`TRUE`
`FALSE || FALSE`	`FALSE`

Vamor encaixar a teoria acima num exemplo prático.

Vamos pensar que temos uma campanha de marketing e temos que verificar se a pessoa está dentro do nosso público alvo.

Público alvo Campanha de Marketing para venda de Cerveja:

- Sexo Masculino, E
- Maior de Idade, E

- Gosta de Futebol

Pessoa a ser analisada:

- Sexo Feminino
- Maior de Idade
- Não gosta de futebol

Vamos aplicar o texto acima na programação

```
#pessoa analisada
tem_sexo_masc <- FALSE
maior_idade <- TRUE
gosta_futebol <- FALSE
#requisitos para a campanha
tem_sexo_masc && maior_idade && gosta_futebol
```

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

O resultado acima é que a pessoa selecionada não faz parte do público alvo.

Vamos pensar numa variação da campanha, onde agora o foco é no sexo feminino:

```
tem_sexo_fem <- TRUE
maior_idade <- TRUE
gosta_futebol <- FALSE

tem_sexo_fem && maior_idade && gosta_futebol
```

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

Ainda assim a pessoa avaliada não entra para o público alvo por não gostar de futebol.

Uma variação da lógica acima que ainda assim apresenta o mesmo resultado é ao invés de mudar a variável tem_sexo_masc para tem_sexo_fem, basta usar o NOT, !tem_sexo_masc.

```
tem_sexo_masc <- FALSE
maior_idade <- TRUE
gosta_futebol <- FALSE

!tem_sexo_masc && maior_idade && gosta_futebol
```

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

Com os exemplos acima fica claro que todas as váriáveis precisam ser verdadeiras para o resultado ser verdadeiro.

Agora usando o OR como exemplo, vamos pensar que a campanha agora não exige um público que gosta de futebol, tanto faz:

```
tem_sexo_fem <- TRUE
maior_idade <- TRUE
gosta_futebol <- FALSE

tem_sexo_fem && maior_idade || gosta_futebol
```

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

Agora sim a pessoa analisada faz parte do público alvo.

No entanto é preciso tomar cuidado com a lógica construída, pois da forma acima também pode entrar como público alvo pessoas que gostam de futebol e não são mulheres nem são maiores de idade:

```
tem_sexo_fem <- FALSE
maior_idade <- FALSE
gosta_futebol <- TRUE

tem_sexo_fem && maior_idade || gosta_futebol
```

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

Esses foram exemplos mais lúdicos para representar o funcionamento desses operadores e ficar mais concreto como eles funcionam.