Funções úteis para manipulação de índices no R

31 de março de 2017

Operadores lógicos

Igual a: ==

Diferente de: !=

Maior que: >

Menor que: <

Maior ou igual a: >=

Menor ou igual a: <=

a ou b: (a)|(b)

a e b: (a)&(b)

Operadores lógicos

Exemplos:
z <- c(15, 13, 11, 9, 7)

Operadores lógicos

Exemplos:
z <- c(15, 13, 11, 9, 7)

z == 11

Operadores lógicos

Exemplos:

```
z <- c(15, 13, 11, 9, 7)

z == 11
[1] FALSE FALSE TRUE FALSE FALSE
```

Operadores lógicos

Exemplos:

```
z <- c(15, 13, 11, 9, 7)

z == 11
[1] FALSE FALSE TRUE FALSE FALSE

z != 11
```

Operadores lógicos

Exemplos:

```
z <- c(15, 13, 11, 9, 7)

z == 11
[1] FALSE FALSE TRUE FALSE FALSE

z != 11
[1] TRUE TRUE FALSE TRUE TRUE
```

Operadores lógicos

Exemplos:

z <- c(15, 13, 11, 9, 7)

z == 11

[1] FALSE FALSE TRUE FALSE FALSE

z != 11

[1] TRUE TRUE FALSE TRUE TRUE

z > 11

Operadores lógicos

Exemplos:

z <- c(15, 13, 11, 9, 7)

z == 11

[1] FALSE FALSE TRUE FALSE FALSE

z != 11

[1] TRUE TRUE FALSE TRUE TRUE

z > 11

TRUE TRUE FALSE FALSE FALSE

Operadores lógicos

Exemplos:

```
z <- c(15, 13, 11, 9, 7)

z == 11
[1] FALSE FALSE TRUE FALSE FALSE

z != 11
[1] TRUE TRUE FALSE TRUE TRUE

z > 11
TRUE TRUE FALSE FALSE FALSE

(z>=11)&(z<=11)
```

Operadores lógicos

Exemplos:

```
z <- c(15, 13, 11, 9, 7)

z == 11
[1] FALSE FALSE TRUE FALSE FALSE

z != 11
[1] TRUE TRUE FALSE TRUE TRUE

z > 11
TRUE TRUE FALSE FALSE FALSE

(z>=11)&(z<=11)
[1] FALSE FALSE TRUE FALSE FALSE
```

Manipulando índices em vetores

Funções que podem ser úteis para manipular índices:

c()

a:b; $a, b \in \mathbb{N}$

which()

rep()

seq()

Manipulando índices em vetores

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

z[3]
[1] 11

z[-3]
[1] 15 13 9 7
```

Manipulando índices

Exemplo c():

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

z[c(1,5)]
[1] 15 7
```

Manipulando índices em vetores

Exemplo a:b :

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

1:3
[1] 1 2 3

z[1:3]
[1] 15 13 11
```

Manipulando índices em vetores

Exemplo which():

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

which(z != 11)
[1] 1 2 4 5

z[which(z != 11)]
[1] 15 13 9 7
```

Outra forma de fazer a mesma coisa, mas simples:

```
z[z != 11]
[1] 15 13 9 7
```

Manipulando índices em vetores

Exemplo:

É possível utilizar os índices de outros vetores!

```
x <- c(5, 4, 3, 2, 1)
z[x == 5]
[1] 15
"Valor(es) de z tal que x seja igual a 5."
```

Manipulando índices em vetores

Exemplo rep():

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

rep(1:2, times = 3, each = 2)
[1] 1 1 2 2 1 1 2 2 1 1 2 2

z[rep(1:2, times = 3, each = 2)]
 [1] 15 15 13 13 15 15 13 13 15 15 13 13
```

Manipulando índices em vetores

Exemplo seq():

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

seq(from = 1, to = 5, by = 2)
[1] 1 3 5

z[seq(from = 1, to = 5, by = 2)]
[1] 15 11 7
```

Manipulando índices em vetores

Vantagens de se usar a saída de funções como índices:

- Extremamente útil quando o número de elementos de um vetor (ou matriz) é grande;
- Menos linhas de código;
- Processamento otimizado (menor tempo final de processamento).

Manipulando índices em vetores

Exemplo

Gerar um vetor com os valores de z cujos índices são ímpares:

Uma forma complicada:

```
impares <- rep(NA, ceiling(length(z)/2))
for (i in 0:2){
impares[i+1] <- z[2*i+1]
}
```

Manipulando índices em vetores

Exemplo

Gerar um vetor com os valores de z cujos índices são ímpares:

Forma simples:

```
impares <- z[seq(1, length(z), 2)]
```

Atenção: se possível, evite utilizar funções de loop (iterativas ou condicionais) como if(), for() e while(). Elas ocupam muito espaço no código e, na maior parte das vezes, podem ser substituídas por expressões mais simples e otimizadas para o processamento.

Manipulando índices em matrizes

A manipulação de índices em matrizes é a extensão bi-dimensional do caso vetorial.

Manipulando índices em matrizes

```
M1 <- matrix(data = c(1,2,3,4,5,6,7,8,9), nrow = 3, byrow = T)
M1
```

Manipulando índices em matrizes

```
M1 <- matrix(data = c(1,2,3,4,5,6,7,8,9), nrow = 3, byrow = T)
M1
```

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

Manipulando índices em matrizes

```
M1 <- matrix(data = c(1,2,3,4,5,6,7,8,9), nrow = 3, byrow = T)
M1
```

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

```
M2 <- matrix(data = c(1,2,3,4,5,6,7,8,9), nrow = 3, byrow = F)
M2
```

Manipulando índices em matrizes

```
M1 <- matrix(data = c(1,2,3,4,5,6,7,8,9), nrow = 3, byrow = T)
M1
```

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

```
M2 <- matrix(data = c(1,2,3,4,5,6,7,8,9), nrow = 3, byrow = F)
M2
```

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

Manipulando índices em matrizes

Manipulando elementos de matrizes

M1

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

M1[2,]
[1] 4 5 6

M1[,2]
[1] 2 5 8

M1[2,2]
[1] 5

Manipulando índices em matrizes

Manipulando elementos de matrizes

M1

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

M1[-1,-2]

	[, 1]	[, 2]
[1,]	4	6
[2,]	7	9

Manipulando índices em matrizes

Manipulando elementos de matrizes

Exemplo:

M1

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

M1[2:3, c(1,3)]

	[, 1]	[, 2]
[1,]	4	6
[2,]	7	9

Manipulando índices em matrizes

No R, bases de dados se comportam exatamente como matrizes. A manipulação de índices em folhas de dados (data frames) é análogo ao caso matricial!

Funções para rodar em casa:

- colSums(), rowSums(), colMeans(), rowMeans() e toda a família de funções apply()!
- Dúvidas? help(nomedafunção) ou ??nomedafunção
- Mais dúvidas? Procure por aplicações no Google