ME115 - Linguagem R

Atividade Prática 02

1º semestre de 2023

Introdução

Nessa atividade, exploraremos as seguintes operações:

1. indexação de matrizes
2. operações lógicas do tipo `>`, `<`, `&` e `||`, `all`, `any`
3. indexação de `data.frame`
4. Operadores lógicos: no contexto de indexação, comparação entre objetos como mecanismo de parada em blocos de repetição
5. Controle de fluxo: `if/else`; `ifelse`
6. Blocos de repetição: `for`; `while`

Atividade

1. Considere o código a seguir para criar a matrix

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

```
X <- matrix(data=seq(1, 9), nrow=3, ncol=3)
X
```

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

Essa forma de escrever a matriz não coloca os elementos na posição certa. Podemos usar o argumento `byrow` da função para colocar os elementos na posição certa.

```
X <- matrix(data=seq(1, 9), nrow=3, ncol=3, byrow=T)
X
```

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

Para acessar os elementos de uma matriz usamos o indexador "`[r, l]`", onde o elemento `r` (à esquerda da vírgula) representa a linha e o `l` (à direita) a coluna. A seguir experimente os códigos que ilustram como indexar matrizes:

```
X[1, 2]      # retorna o elemento da linha 1 e coluna 2
X[2, 1]      # retorno o elemento da linha 2 e coluna 1
X[1,]        # retorna os elementos da linha toda
X[,1]        # retorna os elementos da coluna toda
```

```
X[, 1:2]      # retorna colunas 1 e 2
X[c(1,3), ]   # retorna linhas 1 e 3
diag(X)       # retorna os elementos da diagonal principal da matrix
```

Considere a matrix X. Indexando elemento a elemento, calcule:

(a) a soma dos elementos da linha 2.
(b) a soma dos elementos da coluna 3.
(c) a soma dos elementos da diagonal de X.

2. Crie a matrix Y de tamanho 10×10 que contenha os valores de 1 a 100, dispostos por linha. Imprima na tela os seguintes elementos:

(a) Elementos das colunas pares de Y.
(b) Elementos das linhas pares de Y.
(c) Elementos das linhas e colunas pares de Y.

Dica: utilize como indexador números pares criados a partir de uma sequência, como feito para vetores.

3. O objeto `data.frame` pode ser pensado como uma matriz. Neste caso, sua indexação pode ser feita através do indexador `[,]`. Execute o código abaixo para carregar o *data frame* `murders` do pacote `dslabs`.

```
library(dslabs)
data(murders)
```

A seguir, responda:

(a) O que representa o comando `murders[1, 1]`?
(b) O que representa o comando `murders[1, ]`?
(c) O que representa o comando `murders[, 1]`?
(d) O que representa o comando `murders[1:2, ]`?
(e) O que representa o comando `murders[, 3:4]`?

4. O código abaixo retorna "Nem todos são positivos". Por que?

```
x <- c (1, 2, -3, 4)

if (all (x > 0)) {
  print("Todos são positivos")
  } else {
    print("Nem todos são positivos")
    }
```

(a) Reescreva o código tal que ele retorne "Todos são positivos". **Dica**: utilize o operador de negação `!`.

(b) Reescreva o código tal que ele retorne uma frase "Nem todos são números pares" para o caso de haver algum número ímpar em `x` e caso contrário "Todos são números pares". **Dica**: use `x %% 2`.

(c) Usando a função any (utilize `?any` para saber sobre ela) e o vetor `x`, reescreva o código para que a mensagem "Algum número negativo" seja exibida caso existam números negativos em `x` e "Nenhum número negativo" caso contrário.

(d) Defina um vetor `y` tal que usando o código em (b) ele retorne "Todos são números pares".

5. Usando o código da aula prática 01 dado a seguir:

```
library(dslabs)
data(murders)
murder_rate <- murders$total/murders$population*100000
```

Execute o código abaixo, o qual nos diz qual(is) estado(s) tem a taxa de assassinato por 100.000 habitantes menor do que 0.5 pessoas. Veja que o `if` é usado aqui para cobrir também os casos em que nenhum estado apresenta taxa menor que o limite estabelecido.

```
ind <- which(murder_rate < 0.5)

if(length(ind) > 0){
  print(murders$state[ind])
  } else{
    print("No state has murder rate that low")
    }
```

```
## [1] "New Hampshire" "Vermont"
```

A seguir, usando o código fornecido, encontre:

(a) Os estados com taxa por 100.000 habitantes maior ou igual a 2 pessoas.

(b) Os estados com taxa por 100.000 habitantes entre 0.5 e 2 pessoas.

(c) Os estados com taxa por 100.000 habitantes menor do que 0.25 pessoas.

6. O controle de fluxo `ifelse` trabalha em vetores, como exemplificado no código a seguir.

```
a <- c(0, 1, 2, -4, 5)
ifelse(a > 0, 1/a, NA)
```

```
## [1]  NA 1.0 0.5  NA 0.2
```

A seguir,

(a) Crie uma sequência de inteiros -10 a 120 e armazene. Utilizando `ifelse`, retorne a raiz quadrada dos elementos que são maiores ou iguais a zero e NA caso contrário.

(b) Utilizando a sequência criada em (a) e o comando `ifelse`, retorne 1 se o número for par e 0 caso contrário.

(c) Utilizando a sequencia criada em (a) e o comando `ifelse`, retorne o próprio número se este for divisível por 3 e NA caso contrário.

7. O comando `for` é um comando de repetição. Considere os códigos a seguir como exemplos.

```
for(i in 1:5){
  print(i)
}
```

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

```
for(i in seq(1, 10, by=2)){
     print(i)
  }
```

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

```r
for(i in c("a", "b", "c")){
     print(i)
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
```

```r
## bloco de repetição aninhado

for(i in c("a", "b", "c")){
  for(j in seq(1, 10, by=2)){
    print(i)
    print(j)
    }
  }
## o resultado é extenso e por isso não foi apresentado
```

```r
soma <- 0
for (i in 1:10){
  soma <- soma + i
}

print(soma)
```

```
## [1] 55
```

A seguir,

(a) Utilizando o *data frame* `murders`, considere a taxa de mortes por 100000 habitantes. Calcule a média da taxa de mortes e armazene na variável `media`.

(b) Calcule a média da taxa de mortes para a região Sul (*South*) utilizando um bloco de `for`, como nos exemplos, e guarde na variável `media.south`.

(c) Calcule a média da taxa de mortes para regiões que não sejam a região "South" ou "North Central" utilizando um único bloco de repetição `for` e guarde na variável `media.outras`. **Dica**: use o código do item (b) com operador de negação `!`.

(d) Imprima na tela o valor das variáveis em (a), (b) e (c).

8. Considere o exemplo de bloco de repetição aninhado do exercício anterior. A seguir, calcule a seguinte expressão

$$m = \frac{1}{100} \sum_{i=1}^{10} \sum_{j=1}^{10} i \times j$$

e imprima na tela o valor final de m.

9. Considere bloco de repetição usando o comando `while`.

```r
## note que no bloco de repetição while temos que incrementar manualmente
ind <- 1
while (ind < 100) {
  print(ind);
  ind <- ind + 1
}
```

(a) Considere a sequência `-108:88` e armazene-a em `s`. Utilizando o bloco de repetição `while`, faça a soma dos valores de `s` até que um valor positivo em `s` seja encontrado.

(b) Usando `s` e o bloco de repetição `while`, faça a soma dos valores de `s` a partir do valor 88 até que um valor negativo de `s` seja encontrado. **Dica**: você pode usar decremento de um índice.

Agrecimentos

O material foi produzido pela Profa. Tatiana Benaglia para o curso de ME115.