Introdução à Programação e Ciência de Dados para a Gestão Pública

Rebeca de Jesus Carvalho

FGV CEPESP

rebeca.jesus.carvalho@gmail.com

Tópicos da aula

Conteúdo da aula de hoje

- Vamos conhecer o básico da linguagem R.
- Abordaremos classes e tipos de dados e aprenderemos um pouco mais sobre vetores e operações matemáticas.
- Também veremos a importância dos operadores relacionais e lógicos, bem como as vantagens de se utilizar controles de fluxo na programação.

Vetores no R

- Um vetor é uma estrutura de dados básica do R, que permite armazenar sob um mesmo nome conjuntos indexados de valores (numéricos, lógicos ou caracteres).
- Para criá-los, basta colocar os valores separados por vírgulas dentro de um `c()` e atribuí-los a um objeto.
- Cada coluna de um *data frame* pode ser representada como um vetor.
- Podemos criar um novo conjunto de dados utilizando apenas vetores através da função `data.frame()`.

Data frames e vetores

- No caso do exemplo abaixo, teríamos 03 vetores distintos:
  - `municipios <- c("São Paulo", "Rio de Janeiro", "Brasília", "Fort "Salvador");`
  - `estados <- c("SP", "RJ", "DF", "CE", "BA");` e
  - `populacao <- c(11451245, 6211423, 2817068, 2428678, 2418005).`

Tabela 01: Cinco maiores cidades do Brasil em população

Município	Estado	População
São Paulo	SP	11.451.245
Rio de Janeiro	RJ	6.211.423
Brasília	DF	2.817.068
Fortaleza	CE	2.428.678
Salvador	BA	2.418.005

Fonte: IBGE.

Classes e tipos de dados

- Toda variável (ou vetor) dentro da linguagem R é interpretada como um objeto que pertence a uma determinada classe. Para saber qual a classe de um objeto podemos utilizar a função `class()`.
- As classes não se misturam, isto é, para um objeto ter a classe `character`, por exemplo, todos os seus valores precisam ser de texto.
- As classes mais comuns no R são: `numeric`, `logical`, `character` e `list`.

Operadores aritméticos

- Adição: `+`
- Subtração: `-`
- Multiplicação: `*`
- Divisão: `/`
- Potência: `^`
- Divisão inteira (sem resto): `%/%`
- Resto da divisão: `%%`

Operadores relacionais

- ”Igual a”: ==
- ”Diferente de”: !=
- ”Maior que”: >
- ”Menor que”: <
- ”Maior ou igual a”: >=
- ”Menor ou igual a”: <=

Operadores lógicos

- "E": `&`
- "OU": `|`
- "NÃO": `!`

- **Outros operadores:**
  - Sequência: `:`
  - "Está contido em": `%in%`

Cláusulas condicionais

- Um dos usos mais importantes dos operadores relacionais e lógicos é na construção de cláusulas condicionais, `if`, `else` e `else if`.
- Elas servem para executarmos um código apenas se uma condição (teste lógico) for satisfeita.

Comando `if`

- O comando `if` permite executar um bloco de código se uma condição for verdadeira.
- **Sintaxe:**
  - `if(condição){`
  - `    Código a ser executado se a condição for verdadeira}`

Comando `if` com `else`

- O comando `else` permite executar um bloco de código quando a condição do `if` não for verdadeira.

- **Sintaxe:**

```
if(condição){
    Código a ser executado se a condição for verdadeira
} else{
    Código a ser executado se a condição não for
verdadeira}
```

Comando `if` com `else if`

- O comando `else if` é usado para testar condições adicionais após um `if`.
- Pode haver múltiplos `else if` após o `if`, e apenas um bloco será executado.

- **Sintaxe:**

```
if(condição){
    Código a ser executado se a condição for verdadeira
} else if(outra condição){
    Código a ser executado se a outra condição for verdadeira
} else{
    Código a ser executado se nenhuma condição for verdadeira}
```

Estruturas de repetição

- As estruturas de repetição permitem executar um bloco de código várias vezes.
- Em R, temos duas estruturas de repetição principais: `while` e `for`.
- Elas são usadas quando precisamos executar um conjunto de instruções repetidamente.
- Cada repetição é chamada de **iteração** e o objeto que muda de valor em cada iteração é chamado de **iterador**.

Estrutura de Repetição `while`

- O `while` é usado para executar um bloco de código enquanto uma condição for verdadeira.

- **Sintaxe:**

```
while(condição){
    Código a ser executado enquanto a condição for verdadeira}
```

Estrutura de Repetição `for`

- O `for` é usado para iterar sobre uma sequência (lista, string, etc.).
- **Sintaxe:**
  - `for(variável in sequência){`
  - `    Código a ser executado em cada iteração}`

Criando funções no R

- Além de usar funções já prontas, você pode criar a sua própria função com o comando `function()`.
- **Sintaxe:**
  - `nome_funcao <- function(parametro1, parametro2){`
  - `    Código a ser executado pela função}`

Laboratório

Agora é o momento de partir para a ação! Temos dois tutoriais agendados para hoje, prontos para serem explorados. Eles estão disponibilizados no GitHub, e você pode acessá-los clicando neste **link**.

Se deixou algum tutorial inacabado do encontro anterior, comece por ele. Caso contrário, prossiga.

Dúvidas?