Metodologias Informacionais com R

Módulo III: Programação Essencial

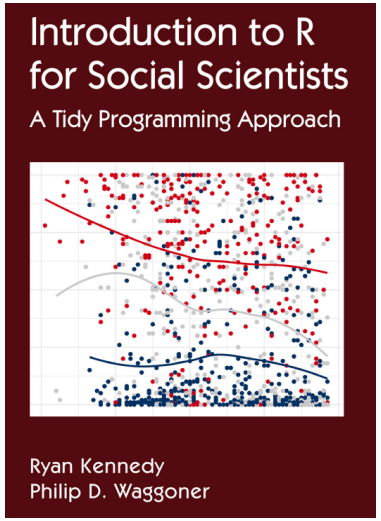
Telmo dos Santos Klipp (telmo.klipp@inpe.br)

Informações Gerais sobre o Curso

- Materiais disponibilizados via Classroom;
- O aprendizado requer a prática que será constante nas aulas;

Bibliografia Básica:

- Kennedy, R., & Waggoner, P. D. (2021). Introduction to r for social scientists: a tidy programming approach. CRC Press.



Bibliografia Complementar:

- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). R for data science (2e): import, tidy, transform, visualize, and model data. "O'Reilly Media, Inc.". Disponível em: https://r4ds.hadley.nz/. Acesso em: 14 de junho, 2023. (Online)
- Damiani, A. et. al., (2022). Ciência de Dados em R. Curso-R. Disponível em: https://livro.curso-r.com. Acesso em: 12 de maio, 2023. (Online)
- de Aquino, J. A. (2014). R para cientistas sociais. Editora da UESC (editus). Disponível em: http://www.uesc.br/editora/. Acesso em: 12 de maio, 2023.
- de Oliveira, P. F., Guerra, S., McDonnell, R. (2018). Ciência de Dados com R: Introdução. Editora IBPAD. Disponível em: https://cdr.ibpad.com.br/index.html. Acesso em: 12 de maio, 2023. (Online)

Nas últimas aulas vimos:

- Visualização de dados de forma estática com **ggplot2**
- Visualização de dados de forma interativa com **plotly**

> **Agora agregaremos algumas habilidades de programação.**

Conceitos de Programação

Habilidades de programação englobam aspectos como:

- Raciocínio lógico.
- Entendimento de problemas.
- Compreensão de métodos para a solução de problemas:
  - Solução algorítmica (sequência de passos);
  - Dividir para conquistar.
- Conhecimentos técnicos:
  - dos recursos computacionais;
  - de recursos da linguagem de programação em uso;
  - de estruturas de dados;
  - de formatos de dados e arquivos.

Conceitos de Programação

Linguagens de programação surgiram e evoluíram considerando a criação de recursos para:

- A solução de problemas com instruções de máquina (ex: acessar posições de memória, mover dados, executar operações matemáticas), lógica condicional, laços de repetição, entre outros.
- Representação, armazenamento e manipulação de dados: tipos básicos (ex: inteiro e carácter) e estruturas de dados (ex: vetores, matrizes e listas), leitura e escrita de arquivos, entre outros.
- Criação de recursos para reaproveitamento e organização de código como a modularização (ex: funções, classes e objetos).
- Elevar a abstração para flexibilizar e expandir as representações computacionais e facilitar o entendimento humano, exemplos:
  - Linguagens de programação de baixo nível (linguagem de máquina e de montagem) para de alto nível (C, C++, Java, Python, R);
  - Conceitos de programação como orientação a objetos, programação funcional e programação literal (ex: *grammar of graphics*).

> **O R é uma linguagem de alto nível que engloba vários dos pontos mencionados.**

Conceitos de Programação no R - objetos básicos

Já vimos que existem 6 tipos básicos (atômicos) de objetos:

- numeric;
- integer;
- character;
- logical;
- complex;
- raw.

> **Objetos tem classes, bem como, podem ter atributos (metadados), que determinam os tipos de manipulações e operações aplicáveis nestes individualmente e entre si.**

Teste as funções class() e attributes() no conjunto de dados **mtcars**.

Conceitos de Programação no R - objetos básicos

Outra característica do R é ser uma linguagem vetorial. Os objetos básicos vistos anteriormente são vetores, podendo armazenar um ou mais elementos, desde que do mesmo tipo. Observe:

```
vetor_de_numeros <- 7
vetor_de_numeros[1]
```

```
## [1] 7
```

```
vetor_de_numeros[2:6] <- c(1, 2, 3, 4, 99)
vetor_de_numeros[6]
```

```
## [1] 99
```

```
length(vetor_de_numeros)
```

```
## [1] 6
```

```
typeof(vetor_de_numeros)
```

```
## [1] "double"
```

Conceitos de Programação no R - tipos de operadores

Operadores são divididos basicamente em quatro categorias:

- **Aritméticos:** +, -, /, *, ^, %% e %/%.
- **Relacionais:** <, >, <=, >=, == e !=.
- **Lógicos:** !, &, &&, | e ||.
- **De atribuição:** <-, <<-, -> e ->>.

Vejamos detalhes neste material online.

Conceitos de Programação no R - controle de fluxo

Um importante fator em soluções algorítmicas é o controle de fluxo da sequência de operações. O R conta com as estruturas condicionais e de repetição comulmente conhecidas em linguagens de programação:

- if(condicional) {instruções} ou if(condicional) {instruções} ... else {instruções} ou
- if(condicional 1) {instruções} else if(condicional 2) {instruções} ... else {instruções};
- ifelse(teste, verdadeiro, falso)
- switch(teste, caso 1, caso 2, caso 3....);
- while(condicional) {instruções};
- for(variavel in sequencia) {instruções};
- break e next;
- repeat{instrução 1, instrução 2, instrução 3 ... break};

Vejamos mais sobre estas estruturas nesse material online.

Conceitos de Programação no R - objetos não básicos

Vimos também alguns exemplos de objetos mais complexos:

- factor;
- data.frame;
- function;
- lists;
- matrices, arrays.

Vejamos detalhes no mesmo material online.

Conceitos de Programação no R - funções

Já vimos que funções realizam tarefas manipulando dados. Estas podem possuir mais de um parâmetro ou nenhum e retornam ao menos um novo objeto. Um dos principais motivos para o uso de funções é a modularização de código. Quebrar códigos em partes mais simples ajuda na manutenção e entendimento dos mesmos.

Vejamos igualmente detalhes no material online.

Conceitos de Programação no R - funções

Elabore uma função que calcule a hipotenusa de um triângulo retângulo conforme o Teorema de Pitágoras: $c^2 = a^2 + b^2$.

Abaixo segue um protótipo com os argumentos necessários para a função:

```
calcula_hipotenusa <- function(a, b) {}
```

Uma possível solução:

```
calcula_hipotenusa <- function(a, b) {
  return(sqrt(a^2 + b^2))
}
calcula_hipotenusa(3, 4)
```

```
## [1] 5
```

Conceitos de Programação no R – funções

Observe o seguinte exemplo de função:

```
calcula_hipotenusa <- function(a, b) {
  if(!is.numeric(a)) {
    stop('"a" must be numeric\n',
         'You have provided an object of class: ', class(a))
  }
  sqrt(a^2 + b^2)
}
```

```
calcula_hipotenusa <- function(a, b) {
  #if(!is.numeric(a)) {
  #  stop('"a" must be numeric\n',
  #       'You have provided an object of class: ', class(a))
  #}
  sqrt(a^2 + b^2)
}
calcula_hipotenusa(3, 4)
```

```
## [1] 5
```

Conceitos de Programação no R – funções

Considerando o exemplo anterior de função, que melhorias seriam possíveis para incrementar a proteção contra entradas erradas inseridas por usuários? Tente codificar essas melhorias ou observe:

```
 calcula_hipotenusa <- function(a, b) {
   if(!is.numeric(a) | !is.numeric(b)) {
       stop("\"a\" and \"b\" must be numeric!\n",
            "You have provided object of classes \"a\": ",
            class(a), " \"b\": ", class(b))
   }
   return(sqrt(a^2 + b^2))
}
```

Conceitos de Programação no R

Pontos importantes para guardar:

- Abuse dos recursos da linguagem para modularização de código.
  - Crie suas próprias funções;
  - Crie seus próprios temas de gráficos do **ggplot2**, suas escalas, entre outros. Mantenha salvo em código para reúso.
- Tente se familiarizar com o uso de funções. Muitos problemas já foram resolvidos, bastando apenas conhecer os pacotes e funções disponíveis.
- Tente sempre automatizar tarefas.
- Documente seu trabalho por meio de *scripts*, análises de dados através de arquivos R markdown ou em plataformas web específicas como GitHub, Kaggle e Google Colab.

Metotologias Informacionais com 

Muito Obrigado pela Atenção!