Linguagem R

Aula 6

- Objetivo da aula:
- Introdução ao RStudio;
- Tipos de dados e operadores;
- Funções;
- Importação de dados;

Introdução ao RStudio

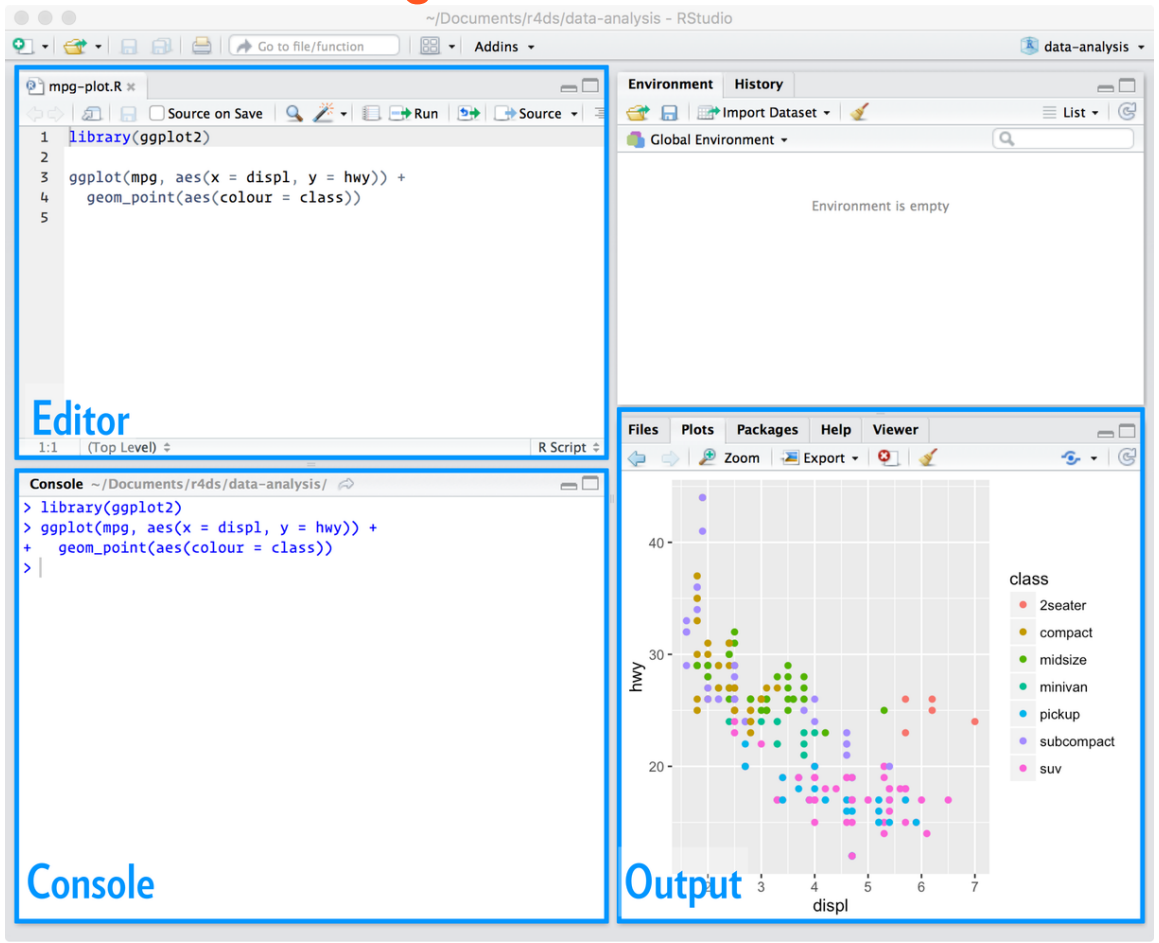
O RStudio é o ambiente de desenvolvimento (ide) mais completo para se trabalhar com a linguagem R.

Ele possui várias ferramentas e recursos para ciência de dados com a linguagem R

Faça a intalação através do link:

https://download1.rstudio.org/electron/windows/RStudio-2023.03.1-446.exe

Introdução ao RStudio



https://livro.curso-r.com/2-1-telas.html

Introdução ao RStudio

Veja alguns atalhos que podem ser úteis:

- CTRL+ENTER: roda a(s) linha(s) selecionada(s) no script. O atalho mais utilizado.
- ALT+-: cria no script um sinal de atribuição (<-). Você o usará o tempo todo.
- CTRL+SHIFT+M: (%>%) operador pipe. Guarde esse atalho, você o usará bastante.
- CTRL+1: altera cursor para o script.
- CTRL+2: altera cursor para o console.
- CTRL+ALT+I: cria um chunk no R Markdown.
- CTRL+SHIFT+K: compila um arquivo no R Markdown.
- ALT+SHIFT+K: janela com todos os atalhos disponíveis.

Tipos de dados e operadores

Vetores:

Vetores são conjuntos de diferentes tipos de dados

Podemos criar um vetor com a função c(), podemos acessar seus elementos utilizando os colchetes [ ].

Exemplo:

```
vetor <- c("Engenharia","de","Meio","Ambiente")
vetor[3] # Retorna o item 3 da variável vetor, "Meio".
```

Tipos de dados e operadores

Listas:

As listas são um tipo especial de variável dentro do R, pois elas podem armazenar diferentes tipos dentro dela. Sua criação acontece usando a função list()

Para acessar os componentes e seus dados, utilizamos colchetes duplos após o nome da variável.

Exemplo:

```
lista <- list(nome="Fernando", idade=28L, phd=FALSE, cursos_id=c(10, 12, 15))

lista[[2]] # Apresentará como resultado o número inteiro 28L
lista[2] # Apresentará o nome do componente (idade) e seus valores associados (28L)
lista[[4]][3] # Apresentará como resultado o número 15.
lista$phd # Apresentará como resultado lógico FALSE
lista[["phd"]] # Idem resultado anterior
```

Tipos de dados e operadores

Matrizes:

Matrizes são conjuntos de dados bidimensionais sendo criados pelo comando matrix(). Dentro dele, podemos definir o número de linhas (nrow), colunas (ncol) e se os dados serão inseridos por colunas (byrow = FALSE) ou linhas (byrow = TRUE)

Exemplo:

```
mat_2E <- matrix(data = 1:18, nrow = 6, ncol = 3, byrow=FALSE)

mat_2E[1,1] # Retorna o item da primeira linha e da primeira coluna
mat_2E[3,4] # Retorna o item da terceira linha e da quarta coluna
mat_2E[,4] # Retorna todos os itens da quarta coluna
mat_2E[1,] # Retorna todos os itens da primeira linha
```

Tipos de dados e operadores

Arrays:

Os arrays são semelhantes às matrizes, mas podem ter mais que duas dimensões, onde o número de dimensões é configurado pelo termo dim. Os arrays são criados pela função array()

Exemplo:

```
ar_2E <- array(data = c("Eng", "MSc", "PhD"), dim=c(4,4,2))

ar_2E[,,1] # Apresentará os dados da primeira matriz
ar_2E[,1,1] # Apresentará os dados da primeira coluna da primeira matriz
ar_2E[1,1,1] # Apresentará o dados na primeira linha, da primeira coluna, da primeira matriz
```

Tipos de dados e operadores

Fatores:

Fatores são utilizados para criar rótulos, sendo bastante úteis em análises estatísticas, podendo ser ordenados ou não ordenados

Exemplo:

```
residuos <- c("Classe I", "Classe I", "Classe IIA", "Classe IIB", "Classe I", "Classe IIB")
tipos_residuos <- factor(x = residuos)
```

```
# Note que quando você acessar os dados da variável tipos_resíduos, irá ver uma linha
#chamada levels, indicando as classes que adotamos (Classe I, Classe IIA e Classe IIB), sem
#repeti-las.
```

Tipos de dados e operadores

DataFrames:

Eles são tabelas e podem agrupar dados de diferentes classes em cada coluna, sendo criados pelo comando data.frame()

Exemplo:

```
solos <- data.frame(
 amostra = c(1,2,3,4),
 horizonte = c("A","B","A1","A2"),
 mat_org = c(3.2, 0.4, 5.3, 4.4),
 pastagem = c(FALSE, FALSE, TRUE, TRUE)
)
solos$horizonte # Apresentará os horizontes coletados
solos$horizonte[2] # Apresentará o segundo valor da coluna Horizonte
solos[3] # Apresentará os valores de matéria orgânica
solos[3,2] # Apresentará a terceira linha da segunda coluna
```

Tipos de dados e operadores

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

Tipos de dados e operadores

Operador	Descrição	Exemplo	Explicação
&	AND lógico	> x <- 0:2 > y <- 2:0 > (x < 1) & (y > 1) [1] TRUE FALSE FALSE	Versão vetorizada. Compara dois elementos do tipo vetor e retorna um vetor de TRUEs e FALSEs
&&	AND lógico	> x <- 0:2 > y <- 2:0 > (x < 1) && (y > 1) [1] TRUE	Versão não-vetorizada. Compara apenas o primeiro valor de cada vetor, retornando um valor lógico.
|	OR lógico	> x <- 0:2 > y <- 2:0 > (x < 1) | (y > 1) [1] TRUE FALSE FALSE	Versão vetorizada. Compara dois elementos do tipo vetor e retorna um vetor de TRUEs e FALSEs
||	OR lógico	> x <- 0:2 > y <- 2:0 > (x < 1) || (y > 1) [1] TRUE	Versão não-vetorizada. Compara apenas o primeiro valor de cada vetor, retornando um valor lógico.
!	NOT lógico	> x <- 0:2 > y <- 2:0 > !y == x [1] TRUE FALSE TRUE	Negação lógica. Retorna um valor lógico único ou um vetor de TRUE / FALSE.
xor	XOR	> x <- TRUE > y <- FALSE > xor(x,y) [1] TRUE	Ou Exclusivo. Retorna valor lógico TRUE se ambos os valores de entrada forem diferentes entre si, e retorna FALSE se os valores forem iguais.

Funções

As funções em R possuem o mesmo objetivo de funções de outras linguagens, como o Python.

Sintaxe:

```
nome = function (argumento1 , ... , argumento n) {
  Comandos da função
}
```

Principais funções

- **head()** - Mostra as primeiras 6 linhas.
- **tail()** - Mostra as últimas 6 linhas.
- **dim()** - Número de linhas e de colunas.
- **names()** - Os nomes das colunas (variáveis).
- **str()** - Estrutura do data frame. Mostra, entre outras coisas, as classes de cada coluna.
- **cbind()** - Acopla duas tabelas lado a lado.
- **rbind()** - Empilha duas tabelas.

Essas são funções úteis para lidar com DataFrames

Importando dados

Importando arquivos *comma separated values* (.csv)

```
dados1 <- read.csv(file.choose(), header=TRUE)

dados2 <- read.table(file.choose(), header=T, sep=“,”)
#Nesta opção, é utilizado um comando mais genérico, sendo necessário explicitar o separador.
```

OBS: Podemos obter ajuda sobre determinada função com o `help(read.table)` ou `?(read.table)`

Importando dados

Importando arquivos *tab separated* (.txt)

```
dados1 <-  read.delim(file.choose(), header=T)

dados2 <- read.table(file.choose(), header=T, sep=“ \t”)
#Nesta opção, é necessário informar que os dados estão separados por um tab
```

Importando dados

Importando arquivos *Excel*(.xls ou .xlsx)

Para isto usamos um pacote que já está instalado no R. Basta ir no Files/Import Dataset/From Excel ou ir na aba Environment e depois na opção Import Dataset escolhendo a opção From Excel.