Tópico 1: Introdução a R

MRS

1. Introdução a R

O que é R?

- R é uma linguagem de programação estatística voltado à análise de dados, estatística e visualização gráfica.
- Criado originalmente por estatísticos, tornou-se uma das ferramentas mais utilizadas em ciência de dados, pesquisa académica, economia, bioinformática, entre outras áreas.

Vantagens do R

- Gratuito e de código aberto.
- Capacidade para visualizações gráficas e relatórios dinâmicos.
- Extensível com pacotes disponíveis.
- Integração com diversas ferramentas e formatos de dados (Excel, SQL, JSON, APIs, etc.).
- Suporte ativo da comunidade científica e técnica

Trabalhar com R

- R é o motor da linguagem.
- IDEs (ambientes de desenvolvimento): Visual Studio Code, RStudio, etc.
- Suporte a markdown: Quarto, Posit, etc.

Instalação

1. VSCode: https://code.visualstudio.com/.
2. R: https://cran.r-project.org.
3. Quarto : https://quarto.org/docs/get-started/
4. No VSCode, instalar as extensões:

- R, REditorSupport
- R Extension Pack, Yuki Ueda
- Quarto, Quarto

5. Se necessário, criar o terminal R:

- Aceder à palete de comandos com ctrl+shift+p
- Escrever R: Create Terminal

2. Testar se o R está a funcionar

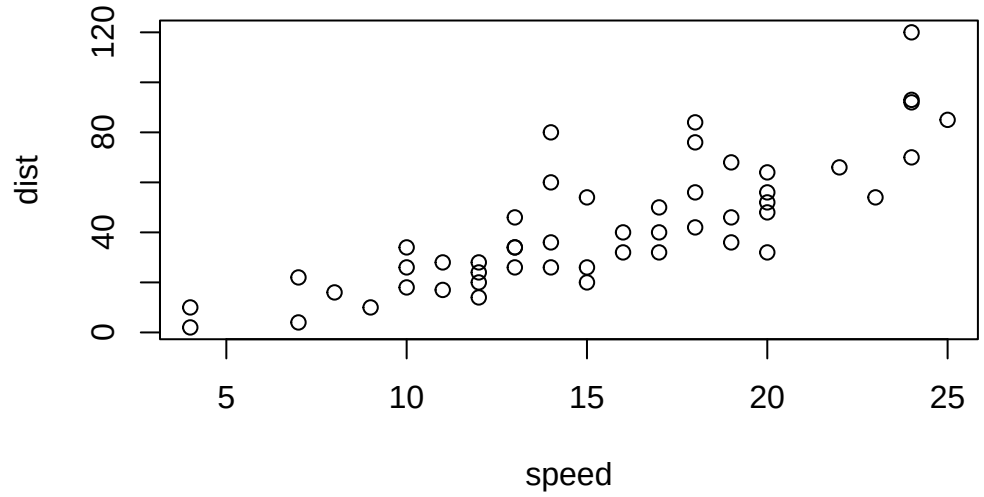
```
head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

```
summary(cars)
```

```
     speed           dist       
 Min.   : 4.0   Min.   :  2.00  
 1st Qu.:12.0   1st Qu.: 26.00  
 Median :15.0   Median : 36.00  
 Mean   :15.4   Mean   : 42.98  
 3rd Qu.:19.0   3rd Qu.: 56.00  
 Max.   :25.0   Max.   :120.00  
```

```r
plot(cars)
```



3. Primeiros comandos

Comandos

```r
# Ver a pasta de trabalho atual

getwd()
```

```
[1] "C:/_FORMACAO/CA_2025-05_AiDAPT01/02_FEADR"
```

```r
# Definir a pasta de trabalho

setwd("C:/_FORMACAO/CA_2025-05_AiDAPT01/02_FEADR")
```

```r
# Pedir ajuda sobre uma função

help(mean)
```

```
starting httpd help server ... done
```

```
?sum
```

Atalhos de teclado

- Inserir uma célula de código: **ctrl+shift+i**
- Executar a célula de código atual: **shift+enter**
- Executar os comandos, um a um: **ctrl+enter**
- Executar todo o script: **ctrl+shift+enter**
- Compilar o documento em HTML, PDF ou Word (knit): **ctrl+shift+k**
- Chamar a palete de comandos: **ctrl+shift+p**
- Limpar o terminal: **ctrl+l**
- Comutar entre editor e o terminal: **ctrl+j**
- Auto completar / sugestão: **ctrl+espaço**
- Comentar/Descomentar linha(s): **ctrl+/**
- Alternar entre ficheiros abertos: **ctrl+tab**

4. Dados em R

Variáveis

- São usadas para armazenar dados e objetos em R.
- O nome de uma variável deve começar com uma letra e pode conter letras, números, pontos e sublinhado.
- Usar camelCase para designar variáveis.
- A atribuição pode ser feita com <- (mais comum), = ou ->.
- As variáveis não têm de ser declaradas com nenhum tipo específico e podem mudar de tipo depois de serem definidas.

```
# Declarar variáveis

x <- 10       # atribuição com <-
y = 5         # atribuição com =
z <- x + y    # operação com variáveis

# Ver variáveis
x
```

```
[1] 10
```

```r
y
```

```
[1] 5
```

```r
print(z)
```

```
[1] 15
```

Tipos de dados

- R possui vários tipos de dados padrão.
- Os principais são:
  - numeric (inclui inteiros e decimais): 10.5, 55, 787
  - integer: 1L, 55L, 100L
  - character (string): “R is ok”, ‘testing...’
  - logical (boolean): TRUE or FALSE
  - factors: usados para representar variáveis categóricas

```r
# Numéricos
valorInteiro <- 10
valor2 <- 3.14   # o carater decimal é o ponto (.)

# Inteiros
contagem <- 14L

# Caracteres
nome <- "MRS"
curso <- 'R'

curso
```

```
[1] "R"
```

```r
# Lógicos
aprovado <- TRUE
inscrito <- FALSE

inscrito
```

```
[1] FALSE
```

```r
# Fatores
classificacao <- factor(c("Baixo", "Médio", "Alto"))
situacao <- factor(c("Resolvido", "Adiado", "Por concluir"))

print(situacao)
```

```
[1] Resolvido    Adiado       Por concluir
Levels: Adiado Por concluir Resolvido
```

```r
# Ver o tipo de dados das variáveis

class(valorInteiro)
```

```
[1] "numeric"
```

```r
class(contagem)
```

```
[1] "integer"
```

```r
class(curso)
```

```
[1] "character"
```

```r
class(aprovado)
```

```
[1] "logical"
```

```r
class(situacao)   # gera Levels
```

```
[1] "factor"
```

Classificação das variáveis

- Ao trabalhar com dados, é essencial entender os **tipos de variáveis** e as suas **escalas de mensuração**, pois isso influencia as análises estatísticas.
- Tipo > escala > Explicação > exemplos
- **Categórica nominal**
  - Nominal
  - Sem ordem entre categorias
  - Género, cor dos olhos
- **Categórica ordinal**
  - Ordinal
  - Com ordem, mas sem diferença numérica definida entre níveis
  - Grau de satisfação, escolaridade
- **Numérica discreta**
  - Discreta
  - Valores inteiros e contáveis
  - Número de filhos, número de testes
- **Numérica contínua**
  - Contínua
  - Pode assumir qualquer valor dentro de um intervalo (decimais)
  - Altura, peso, temperatura

Nominal (fator simples)

```r
genero <- factor(c("Masculino", "Feminino", "Feminino", "Masculino"))

# os níveis representam os valores únicos possíveis da categoria
levels(genero)
```

```
[1] "Feminino"  "Masculino"
```

Ordinal (fator ordenado)

```r
nivel <- factor(c("Baixo", "Médio", "Alto", "Médio"), ordered = TRUE)

levels(nivel)
```

```
[1] "Alto"  "Baixo" "Médio"
```

Discreta (inteiro)

```r
filhos <- c(1L, 2L, 0L, 3L)
```

Contínua (numérico decimal)

```r
peso <- c(60.5, 72.3, 68.0, 80.2)
```

Objetos

- Em R, tudo é um objeto.
- As principais estruturas de dados:
  - Vetores
  - Matrizes
  - Data frames
  - Listas

Vetores

- Coleção de elementos do mesmo tipo.

```r
idades <- c(23, 34, 28, 40)
```

Matrizes

- Tabelas bidimensionais com elementos do mesmo tipo.
- Por defeito, a matriz é preenchida por colunas.
- Para preencher por linhas, usar o argumento byrow = TRUE.

```r
# 1:6 → Gera uma sequência de números de 1 até 6
# matrix(...): cria ma matriz
# nrow = 2: define o número de linhas
# ncol = 3: define o número de colunas
matriz <- matrix(1:6, nrow = 2, ncol = 3)

# por coluna
matrizPorColuna <- matrix(c(2, 45, 13, 85, 100, 4), nrow = 2, ncol = 3)

# por linha
matrizPorLinha <- matrix(c(2, 45, 13, 85, 100, 4), nrow = 2, ncol = 3, byrow = TRUE)

# mostrar
matriz
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```r
matrizPorColuna
```

```
     [,1] [,2] [,3]
[1,]    2   13  100
[2,]   45   85    4
```

```r
matrizPorLinha
```

```
     [,1] [,2] [,3]
[1,]    2   45   13
[2,]   85  100    4
```

Data frames

- Tabelas onde cada coluna pode ter um tipo de dados diferente.

```
df <- data.frame(nome = c("Maria", "Carlos"), idade = c(20, 60))

df
```

```
    nome idade
1  Maria    20
2 Carlos    60
```

Listas

- Coleções heterogéneas, podem conter objetos de diferentes tipos e estruturas de dados.

```
idades <- c(20, 40, 30, 50)

pessoas <- data.frame(primeiroNome = c("Ana", "João"), localidade = c("Espinho", "Porto"))

lista <- list(numeros = idades, tabela = pessoas)

print(lista)
```

```
$numeros
[1] 20 40 30 50

$tabela
  primeiroNome localidade
1          Ana    Espinho
2         João      Porto
```

5. Pacotes essenciais: tidyverse, dplyr, ggplot2

- Pacotes (packages) em R são extensões que adicionam novas funcionalidades.
- Os mais importantes para análise de dados são:
  - `tidyverse`: conjunto de pacotes para ciência de dados.
  - `dplyr`: manipulação eficiente de dados.
  - `ggplot2`: criação de gráficos.

Instalação e carregamento

```
# Instalar o completo
# install.packages("tidyverse")   # inclui dplyr e ggplot2, entre outros

# Instalar individualmente
# install.packages(c("dplyr", "ggplot2"))

# Ver a informação completa dos pacotes instalados
# installed.packages()

# Ver o nome dos pacotes instalados
# rownames(installed.packages())

# No VSCode, ver o nome dos pacotes instalados
# View(installed.packages())

# Ver pacotes carregados na sessão atual
# search()

# Ver se um pacote específico está instalado
# "ggplot2" %in% rownames(installed.packages())
```

```
# Carregar pacotes
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

6. Operações com dataframes

```r
compras <- data.frame(
  cliente = c("Ana", "João", "Ana", "Carlos", "João", "Beatriz"),
  produto = c("Livro", "Caneta", "Caderno", "Livro", "Caderno", "Caneta"),
  valor = c(12.5, 1.2, 3.0, 12.5, 3.0, 1.2),
  data = as.Date(c("2025-05-01", "2025-05-01", "2025-05-02", "2025-05-03", "2025-05-04", "202
)
```

Operações comuns

1. Filtrar linhas

```r
compras[compras$valor > 5, ]        # compras acima de 5,00 €
```

```
  cliente produto valor       data
1     Ana   Livro  12.5 2025-05-01
4  Carlos   Livro  12.5 2025-05-03
```

2. Selecionar colunas

```r
compras[c("produto", "valor")]      # só produto e valor
```

```
  produto valor
1   Livro  12.5
2  Caneta   1.2
3 Caderno   3.0
4   Livro  12.5
5 Caderno   3.0
6  Caneta   1.2
```

3. Adicionar nova coluna

```r
compras$valor_com_iva <- compras$valor * 1.23       # valor com IVA
```

4. Ordenar linhas

```r
compras[order(compras$valor), ]     # crescente
```

```
  cliente produto valor       data valor_com_iva
2    João  Caneta   1.2 2025-05-01         1.476
6 Beatriz  Caneta   1.2 2025-05-04         1.476
3     Ana Caderno   3.0 2025-05-02         3.690
5    João Caderno   3.0 2025-05-04         3.690
1     Ana   Livro  12.5 2025-05-01        15.375
4  Carlos   Livro  12.5 2025-05-03        15.375
```

```r
compras[order(-compras$valor), ]    # decrescente
```

```
  cliente produto valor       data valor_com_iva
1     Ana   Livro  12.5 2025-05-01        15.375
4  Carlos   Livro  12.5 2025-05-03        15.375
3     Ana Caderno   3.0 2025-05-02         3.690
5    João Caderno   3.0 2025-05-04         3.690
2    João  Caneta   1.2 2025-05-01         1.476
6 Beatriz  Caneta   1.2 2025-05-04         1.476
```

5. Agrupar e resumir

```r
aggregate(valor ~ cliente, data = compras, sum)     # total gasto por cliente
```

```
  cliente valor
1     Ana  15.5
2 Beatriz   1.2
3  Carlos  12.5
4    João   4.2
```

5. Agrupar e resumir com dplyr

```r
library(dplyr)

compras %>%
  group_by(cliente) %>%
  summarise(total_gasto = sum(valor))
```

```
# A tibble: 4 x 2
  cliente total_gasto
  <chr>         <dbl>
1 Ana            15.5
2 Beatriz         1.2
3 Carlos         12.5
4 João            4.2
```

6. Contar

```r
table(compras$produto)      # compras por produto
```

```
Caderno  Caneta   Livro
      2       2       2
```

7. Filtrar

```r
subset(compras, data == as.Date("2025-05-04"))      # data = 2025-05-04
```

```
  cliente produto valor       data valor_com_iva
5    João Caderno   3.0 2025-05-04         3.690
6 Beatriz  Caneta   1.2 2025-05-04         1.476
```

Resumo

Operação	Função base R	Equivalente dplyr
Filtrar	`subset()`	`filter()`
Selecionar col.	`df[c(...)]`	`select()`

Operação	Função base R	Equivalente dplyr
Adicionar col.	`df$new_col <- ...`	`mutate()`
Ordenar	`order()`	`arrange()`
Agrupar/resumir	`aggregate()`	`group_by()` + `summarise()`
Contar	`table()`	`count()`

7. Exercício Prático

Crie um pequeno conjunto de dados representando pessoas com nome, idade e cidade. Em seguida:

1. Crie um `data frame` com essas informações.
2. Adicione uma nova coluna chamada `idade_2030` que contenha a idade estimada da pessoa em 2030.
3. Filtre apenas as pessoas com mais de 30 anos.
4. Liste os nomes e cidades dessas pessoas.

Solução

```r
# Criar o data frame
pessoas <- data.frame(
  nome = c("Ana", "João", "Carlos", "Beatriz", "Mariana"),
  idade = c(25, 34, 29, 41, 22),
  cidade = c("Lisboa", "Porto", "Coimbra", "Faro", "Braga")
)

# Adicionar coluna com idade estimada em 2030
pessoas$idade_2030 <- pessoas$idade + (2030 - 2025)

# Filtrar pessoas com mais de 30 anos
maiores_30 <- pessoas[pessoas$idade > 30, ]

# Mostrar nome e cidade dessas pessoas
maiores_30[c("nome", "cidade")]
```

```
     nome cidade
2    João  Porto
4 Beatriz   Faro
```

```r
nome = c("Ana", "Joana", "Maria", "Eduardo", "Jose", "Carlota", "Joaquim")
idade = c(37,25,42,55,27,32,37)
cidade = c("Espinho", "Tondela", "Aveiro", "Viseu", "Porto", "Braga","Matosinhos")
df <- data.frame(nome, idade, cidade)
df$idade_2030 = df$idade + 5
filtro = subset(df, df$idade > 30)
filtro[c("nome", "cidade")]
```

```
     nome     cidade
1     Ana    Espinho
3   Maria     Aveiro
4 Eduardo      Viseu
6 Carlota      Braga
7 Joaquim Matosinhos
```