Ferramentas de Apoio à Economia

Aula 5 - Introdução ao R para Economistas

Tiago Afonso

2025-03-11

Table of contents

Instalação do R

Como instalar o R?

Para instalar e poder utiliza o `R`, siga os seguintes passos:

1. Download R

- Aceder ao site oficial do R: https://cran.r-project.org/
- Escolher o sistema operativo adequado (Windows, Mac, Linux).
- Descarregar o ficheiro de instalação e seguir as instruções.

2. Download RStudio

- Aceder ao site oficial do RStudio: https://posit.co/download/rstudio-desktop/
- Escolher a versão gratuita (RStudio Desktop).
- Descarregar o ficheiro de instalação e seguir as instruções.

3. (opcional) em alternativa ao RStudio, é possível utilizar uma nova inerface do R chamada Positron.

- Aceder ao site oficial do Positron: https://positron.posit.co/
- Escolher o sistema operativo adequado (Windows, Mac, Linux).
- Descarregar o ficheiro de instalação e seguir as instruções.

Porque é necessário instalar o R e o RStudio? O `R` é uma linguagem de programação e ambiente de software para análise de dados. O `RStudio` e `positron` são uma interface gráfica que facilita a utilização do `R` e a visualização dos resultados. O `R` é uma das linguagens mais populares para análise de dados económicos. Tem uma vasta comunidade de utilizadores e uma grande quantidade de pacotes disponíveis, quer seja para a estimação de modelos econométricos, quer seja para a visualização de dados.

Primeiros Passos no R

Ambiente de Trabalho: O Rstudio é composto por 4 paineis principais:

- Ambiente: onde é possível ver os objetos carregados (conjuntos de dadaos e valores)
- Consola R: onde é possível escrever e executar comandos.
- R Script: onde é possível escrever e guardar scripts de código.
- Output: onde é possível ver os resultados (gráficos e páginas).

1º passo - Limpar o ambiente de trabalho

Antes de iniciar qualquer projeto/análise, é importante limpar o ambiente do `R`. Desta forma, evitamos conflitos entre variáveis e funções de pprojetos antriores.

Para limpar todas as variáveis do ambiente de trabalho, utilizamos a função `rm` com o argumento `list=ls()`.

```
# limpar ambiente
rm(list = ls())
```

Executar código: Consola vs Script

O comando anterior pode ser inserido na consola do `R` ou no script de código. Para criar um script de código, basta clicar em `File > New File > R Script`. Para executar o código, basta selecionar o código (ou colocar o cursor na linhas) e clicar nas teclas `ctrl + enter`. Na consola basta apenas escrever o código e clicar em `enter`. A vantagem do script é que podemos guardar o código e reutilizá-lo mais tarde. Quase como um documento de texto. Na maioria das vezes é utilizado o script, normalmente a consola serve apenas para comando temporários (exemplo: apagar um objeto do ambiente criado por engano).

Operações Matemáticas

O `R` é uma linguagem de programação que permite realizar operações matemáticas simples como adição, subtração, multiplicação e divisão, como uma calculadora.

```
# operações matemáticas
2 + 2 # ctrl + enter para executar
```

```
[1] 4
```

Atribuir Valores a Objetos

Podemos atribuir valores a objetos e visualizar esses valores. Basta utilizar o operador `<-`. É possível também utilizar o operador `->` para atribuir valores a objetos, é necessário colocar o objeto à direita do operador.

```
# atribuir valores a objetos
a <- 2
a # ver valor objeto
```

```
[1] 2
```

```
6 -> b # atribuir valor a objeto
```

Esstes objetos (`a`, `b` e `c`) são armazenados no ambiente e podem ser utilizados em operações futuras.

```
c <- a + b
```

Remover Objetos específicos do Ambiente R

Para remover um ou mais objetos específicos do ambiente.

```
# remover objeto a
rm(a)

# remover lista de objetos b e c
rm(b, c)
```

Exemplos de Operações

Alguns exemplos de operações matemáticas e funções em R.

```
# exemplos
a <- 3
b <- 5
c <- 9
d <- (a + b) / c
e <- b^2 #quadrado
f <- sqrt(c) #raiz quadrada
h <- log(b) # logaritmo natural
i <- log(b, 10) # logaritmo base 10
```

Funções Básicas

Uma função é um bloco de código que executa uma tarefa específica. O R possui várias funções incorporadas que podem ser utilizadas para realizar tarefas específicas. No exemplo acima foi utilizada a função `sqrft` para calcular a raiz quadrada de um número e a função `log` para calcular o logaritmo de um número.

Para ver a documentação de uma função, onde está a sintáxe, basta utilizar o operador `?` seguido do nome da função.

```
# help
?log
```

Vai aparecer, no canto superior direito do RStudio e do positron, a documentação da função `log`. Na nova janela que aparece, podemos ver os argumentos da função, a descrição e exemplos de utilização. Neste caso a função `log` tem dois argumentos, `x` e `base`. O argumento `x` é o número para o qual queremos calcular o logaritmo e o argumento `base` é a base do logaritmo. Se não especificarmos a base, o logaritmo é calculado na base `e` (logaritmo natural).

Os argumentos de qualquer função são separados por vírgulas. Se os argumentos são opcionais, são indicados entre parênteses retos `[]`. A ordem dos argumentos é importante. Se não soubermos a ordem dos argumentos, podemos ver a documentação da função. É possivel alterar a ordem dos argumentos, mas é necessário indicar o nome do argumento. por exemplo:

```
# logaritmo base 10 - ordem normal
log(100, 10)
```

```
[1] 2
```

ou

```
# logaritmo base 10 - ordem alterada
log(base = 10, x = 100)
```

```
[1] 2
```

O resultado é o mesmo. ## Objetos do `R`

O R tem vários tipos de objetos, como vetores, matrizes, data frames, listas, etc. Os vetores são uma sequência de elementos de um mesmo tipo. Podem ser criados com a função `c()`. OS objetos podem conter diferentes tipos de dados, como números, texto, lógicos, etc. Os objetos podem ser armazenados no ambiente de trabalho e utilizados em operações futuras. Seguem alguns exemplos de objetos do `R`:

Tipo de Objeto	Descrição
Vetor	Sequência de elementos do mesmo tipo. Criado com a função `c()`.

Tipo de Objeto	Descrição
Matriz	Coleção bidimensional de elementos do mesmo tipo. Criada com a função `matrix()`.
Data Frame	Tabela bidimensional onde cada coluna pode conter diferentes tipos de dados. Criado com a função `data.frame()`.
Lista	Coleção de elementos de diferentes tipos. Criada com a função `list()`.
Fator	Variável categórica com níveis. Criado com a função `factor()`.
Array	Coleção multidimensional de elementos do mesmo tipo. Criado com a função `array()`.
Valor	Um único valor numérico ou de texto.

Funções básicas do R Base mais Utilizadas

O `R` possui várias funções incorporadas que podem ser utilizadas para realizar tarefas específicas. Seguem algumas funções básicas mais utilizadas:

Função	Descrição
`mean`	Calcula a média de um vetor ou coluna de dados
`sum`	Calcula a soma dos elementos de um vetor
`sd`	Calcula o desvio padrão de um vetor
`var`	Calcula a variância de um vetor
`min`	Retorna o valor mínimo de um vetor
`max`	Retorna o valor máximo de um vetor
`length`	Retorna o número de elementos em um vetor
`sort`	Ordena os elementos de um vetor
`table`	Cria uma tabela de frequências para os elementos de um vetor
`summary`	Fornece um resumo estatístico básico (mínimo, 1º quartil, mediana, média, 3º quartil, máximo) de um vetor ou coluna de dados

---

Bibliotecas no R

As bibliotecas são conjuntos de funções e dados que estendem a funcionalidade do `R`. O `R` possui um vasto repositório de bibliotecas disponíveis para diferentes tarefas. Para instalar uma biblioteca, utilizamos a função `install.packages("nome_da_biblioteca")`. Para carregar uma biblioteca, utilizamos a função `library(nome_da_biblioteca)`. Para instalar é necessário utilizar aspas duplas `""`, para carregar a biblioteca não é necessário. Exemplo de como instalar a biblioteca `tidyverse`:

```
# instalar biblioteca tidyverse
install.packages("tidyverse")
```

Pode aparecer uma mensagem a perguntar se quer instalar as dependências. Responda `y` para sim. Para carregar a biblioteca `tidyverse` e ou uma nova janela para confirmar o mirror de instalação. Escolha o mirror mais próximo ou o genérico (primeiro).

Exemplo para carregar a biblioteca `tidyverse`:

```
# carregar biblioteca tidyverse
library(tidyverse)
```

A biblioteca `tidyverse` é uma coleção de pacotes para ciência de dados. Inclui pacotes como `ggplot2`, `dplyr`, `tidyr`, `readr`, `purrr`, `tibble`, `stringr`, `forcats`, etc. A biblioteca `tidyverse` é uma das bibliotecas mais populares para manipulação de dados e visualização de dados no `R`.

Não há qualquer problema carregar a biblioteca `tidyverse` várias vezes. Se a biblioteca já estiver carregada, o `R` não vai carregar novamente. Para remover a biblioteca do ambiente, utiliza-se a função `detach("package:tidyverse")`. Isto pode ser útil para libertar algum espaço na memória do computador quando se trabalha com grandes conjuntos de dados e muitas bibliotecas. Para a análise aqui aplicada não deverá ser um problema.

Se tentarmos carregar uma biblioteca que não está instalada, o `R` vai dar um erro: `there is no package called 'nome_da_biblioteca'`. Para instalar a biblioteca, basta seguir os passos anteriores.

Conflitos de funções

ao carregar uma biblioteca podem aparecer alguns conflitos de funções. Por exemplo ao carregar a bibliteca `dplyr` e a `base` do `R` pode aparecer um conflito com a função `filter`. Para resolver este problema, podemos utilizar o nome da biblioteca seguido de `::` e o nome da função. Por exemplo, para utilizar a função `filter` da biblioteca `dplyr`, utilizamos `dplyr::filter`.

Consultar a documentação de uma função

Para consultar a documentação de uma função de uma biblioteca utiliza-se, por exemplo, `?dplyr::filter`. Isto vai abrir uma nova janela com a documentação da função `filter` da biblioteca `dplyr`.

```
?dplyr::filter
```

Introduzir dados para o ambiente R

Vetores e Tabelas

Podemos introduzir dados "manualmente" ao criar vetores e convertê-los em tabelas.

```
# introduzir vetores (listas)
vetor1 <- c(1, 2, 3, 4)
vetor2 <- c(5, 6, 7, 8)
```

Converter em tabela (conjuntos de vetores) com a função `cbind` ou em data frame com a função `data.frame`.

```
# converter em tabela (conjuntos de vetores) - cbind
tabela1 <- cbind(vetor1, vetor2)
```

```
# data frame -> conjunto de dados
df <- data.frame(vetor1, vetor2)
```

```
# ver dados
head(df) # ver as primeiras linhas, ou
```

```
  vetor1 vetor2
1      1      5
2      2      6
3      3      7
4      4      8
```

```
View(df) # ver dados completos
```

Importar Dados de ficheiros externos

.csv

É possivel importar dados externos para o ambiente do R utilizando a função `read.csv()` para ficheiros `.csv` e `read_xlsx()` para ficheiros `.xlsx`. O R suporta muitos outros formatos de ficheiros, como `.txt`, `.dat`, `.dta`, `.sav`, etc. Pode ser necessário utilizar funções bibliotecas adicionais para importar alguns formatos de ficheiros.

Exemplo para importar um ficheiro `.csv`:

```
# limpar ambiente
rm(list = ls())

# importar um ficheiro .csv
dados_csv <- read.csv("CSV.csv")

head(dados_csv) # ver as primeiras linhas
```

```
             Y          X1        X2
1 161589706000 34768754366 4.8983527
2 167902333000 36393681329 0.5810776
3 174309785000 38389407923 6.1486192
4 177697796000 39318850931 5.0276144
5 179067711000 41343708038 0.4361373
6 177401448000 43508667153 6.2669726
```

```
tail(dados_csv) # ver as últimas linhas
```

```
              Y          X1       X2
20 189771059000 83319584579 4.826503
21 195178255000 86312561472 3.237995
22 200414419000 90555041439 4.300252
23 183778988000 74706623651 1.742925
24 193891900000 83753092686 3.091140
25 206855030000 94621684304 3.611657
```

```
View(dados_csv) # ver todos os dados
```

Na função `read.csv()` é necessário indicar o caminho do ficheiro. Se o ficheiro estiver na mesma pasta que o script (ou paste diretório), basta indicar o nome do ficheiro. Se o ficheiro

estiver numa pasta diferente, é necessário indicar o caminho completo do ficheiro. Por exemplo, `C:/Users/username/Documents/CSV.csv`.

```
# ver pasta diretório
getwd()
```

```
[1] "C:/Users/tiago/OneDrive - Universidade da Beira Interior/Economia - Aulas/9_2024-25/3_2
```

Se o ficheiro a importar estiver na pasta diretório, basta indicar o nome do ficheiro. Se o ficheiro estiver numa pasta diferente, é necessário indicar o caminho completo do ficheiro. Por exemplo, `C:/Users/username/Documents/CSV.csv`.

.xlsx

É possivel ter vários ficheiro de dados em simultâneo no ambiente do `R`. Contudo, **é necessário ter atenção ao nomes dos objetos**. Se o nome for o mesmo, o R apenas **substitui o objeto** anterior pelo novo objeto sem qualquer avisa na consola.

Para ver a sintaxe da função, basta utilizar `?read.csv`. O `R`base não suporta a importação de ficheiros `.xlsx`. Para importar ficheiros `.xlsx`, é necessário instalar e carregar a biblioteca `readxl`. Como no exemplo:

```
# carregar biblioteca readxl
library(readxl)
```

```
Warning: package 'readxl' was built under R version 4.4.3
```

```
#carregagar dados
dados_xlsx <- read_xlsx("EXCEL.xlsx")
```

Se não estiver instalada, o `R` vai dar um erro: `there is no package called 'readxl'`. Para instalar a biblioteca, basta seguir os passos introduzir `install.packages("readxl")`.

Os dados são importados para o ambiente do `R` como objetos do tipo `data frame`. Para obter uma descrição dos dados, podemos utilizar a função `summary()`.

```
# estatística descritiva de um conjunto de dados
summary(dados_xlsx)
```

```
       Y                   X1                  X2          
 Min.   :1.616e+11   Min.   :3.477e+10   Min.   :0.4361  
 1st Qu.:1.774e+11   1st Qu.:4.451e+10   1st Qu.:2.2830  
 Median :1.833e+11   Median :5.388e+10   Median :3.6117  
 Mean   :1.833e+11   Mean   :5.959e+10   Mean   :3.8167  
 3rd Qu.:1.896e+11   3rd Qu.:7.471e+10   3rd Qu.:5.0276  
 Max.   :2.069e+11   Max.   :9.462e+10   Max.   :9.8955  
```

Operações com colunas em data frames

Para realizar operações com colunas em data frames, basta utilizar o nome do data frame seguido do símbolo `$` e o nome da coluna. Por exemplo, para somar duas colunas `X1` e `X2` e guardar o resultado na coluna `Z`, basta utilizar a seguinte sintaxe:

```
# estatística descritiva de uma variável num conjunto de dados
summary(dados_xlsx$Y)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
1.616e+11 1.774e+11 1.833e+11 1.833e+11 1.896e+11 2.069e+11 
```

Suponto agora que queremos somar duas colunas `X1` e `X2` e guardar o resultado na coluna `Z`, ou seja, $Z = X1 + X2$, basta utilizar a seguinte sintaxe:

```
# Z = X1 + X2
dados_xlsx$Z <- dados_xlsx$X1 + dados_xlsx$X2
head(dados_xlsx)
```

```
# A tibble: 6 x 4
             Y           X1    X2            Z
         <dbl>        <dbl> <dbl>        <dbl>
1 161589706000 34768754366. 4.90  34768754371.
2 167902333000 36393681329. 0.581 36393681330.
3 174309785000 38389407923. 6.15  38389407929.
4 177697796000 39318850931. 5.03  39318850936.
5 179067711000 41343708038. 0.436 41343708038.
6 177401448000 43508667153. 6.27  43508667160.
```

Elimnar colunas em data frames

Para eliminar uma coluna dentro de um data frame, basta utilizar o operador `NULL`.

```
# eliminar uma coluna dentro de um data frame
dados_xlsx$Z <- NULL
head(dados_xlsx)
```

```
# A tibble: 6 x 3
             Y           X1    X2
         <dbl>        <dbl> <dbl>
1 161589706000 34768754366. 4.90
2 167902333000 36393681329. 0.581
3 174309785000 38389407923. 6.15
4 177697796000 39318850931. 5.03
5 179067711000 41343708038. 0.436
6 177401448000 43508667153. 6.27
```

Ao contrário dos restantes objetos em que utilisamos `rm()` para eliminar. Netse caso `rm` seria utilizado para eliminar o data frame completo.

Operações coim várias colunas em data frames

Para realizar várias operações com colunas em dita francs, podemos utilizar a sintaxe anterior. O que neste caso ficaria da seguinte forma:

```
# calcular ln's de todas as vars

dados_xlsx$ln_y <- log(dados_xlsx$Y)
dados_xlsx$ln_x1 <- log(dados_xlsx$X1)
dados_xlsx$ln_x2 <- log(dados_xlsx$X2)
```

Aqui existe um excesso de repetição do nome do data frame. Para evitar este problema, podemos utilizar a função `mutate` da biblioteca `dplyr`. A função `mutate` permite adicionar novas colunas a um data frame. Para utilizar a função `mutate`, é necessário carregar a biblioteca `dplyr`.

```r
# carregar biblioteca dplyr
library(dplyr)

dados_xlsx <- dados_xlsx |>
  mutate(ln_y = log(Y),
         ln_x1 = log(X1),
         ln_x2 = log(X2))
```

Aqui foi utilizado o pipe `|>` para encadear as funções. O pipe `|>` é um operador que permite encadear funções. O pipe operator significa "then" ou "então". O resultado da função anterior é passado como argumento para a próxima função. O pipe `|>` é muito útil para encadear várias funções e fazer com que o código seja mais legível. O anterior pipe operator `%>%` foi substituído pelo pipe `|>`.

No bloco de código anterior é calculado o logaritmo natural de todas as variáveis e acrescetado ao data frame `dados_xlsx`. É essa a razão de estar duas vezes o nome do data frame. Se o código fosse o seguinte:

```r
dados_xlsx_2 <- dados_xlsx |>
  mutate(ln_y = log(Y),
         ln_x1 = log(X1),
         ln_x2 = log(X2))
```

era criado um novo data frame `dados_xlsx_2` com as novas variáveis e as do `data frame` original. O `data frame` mantém-se inalterado

---

Aplicação

Calcular o logartimo natural de todas as variáveis numéricas do data frame `ficheiro_dados1.xlsx`.