

**Prof. MSc. Marcos Alexandruk**

**E-mail: [alexandruk@uninove.br](mailto:alexandruk@uninove.br)**

## Conteúdo Programático

- Características do R
- Instalação do RStudio
- Comandos Básicos
- Tipos de Dados
- Adicionando Pacotes
- Estruturas Básicas do R
  - Vetores
  - Matrizes
  - Arrays
  - Fatores
  - Data Frames
  - Listas
- Entrada de Dados
- Tipos de Arquivos do R
- Funções: head, tail e summary
- Criando Funções
- Análise Descritiva
  - Medidas de Tendência Central
  - Medidas de Dispersão
  - Separatrizes
- Distribuições Simétricas e Assimétricas
- Curtose
- Gráficos
- ggplot2
- Modelos Probabilísticos Discretos
  - Distribuição binomial
  - Distribuição de Poisson
- Modelos Probabilísticos Contínuos
  - Distribuição Uniforme Contínua
  - Distribuição Normal ou Gaussiana
  - Distribuição Exponencial
- Regressão Linear Simples
- Correlação
- Testes de Hipóteses
  - t de Student
  - Shapiro-Wilk
  - ANOVA
  - Bartlett
  - Tukey HSD

## Características do R

## R: O que é?

R é uma linguagem e um ambiente de desenvolvimento disponível gratuitamente para computação estatística e gráficos que fornece uma ampla variedade de técnicas estatísticas e gráficas: modelagem linear e não linear, testes estatísticos, análise de séries temporais, classificação, agrupamento, etc.

## Principais características

- Open source
- Multi plataforma (Windows, Linux e MacOS)
- Ambiente de linha de comando
- Case sensitive (diferencia maiúsculas de minúsculas)
- Atualizado com frequência
- Extensível através de pacotes
- Milhares de funções de análise de dados
- Gera diferentes tipos de gráficos
- Processamento em memória (melhor performance)
- Integração com: Power BI, Tableau, Oracle, SQL Server, Java, Python, etc.

## Instalação do RStudio

## Instalação do R

O download do R é realizado através do **CRAN** (Comprehensive R Archive Network), uma rede de servidores ftp e web distribuída mundialmente, que armazena versões idênticas e atualizadas de código e documentação para R. Recomenda-se **utilizar o espelho CRAN mais próximo** para minimizar a carga da rede.

O **download** do R é realizado a partir do seguinte **link**:

[\*\*https://cran.r-project.org/bin/\*\*](https://cran.r-project.org/bin/)

## Instalação do R Studio

O RStudio é um **ambiente de desenvolvimento integrado (IDE)** para a linguagem R. Inclui um console, e um editor com destaque de sintaxe que oferece suporte à execução direta de código, bem como ferramentas para plotagem, histórico, depuração e gerenciamento de espaço de trabalho.

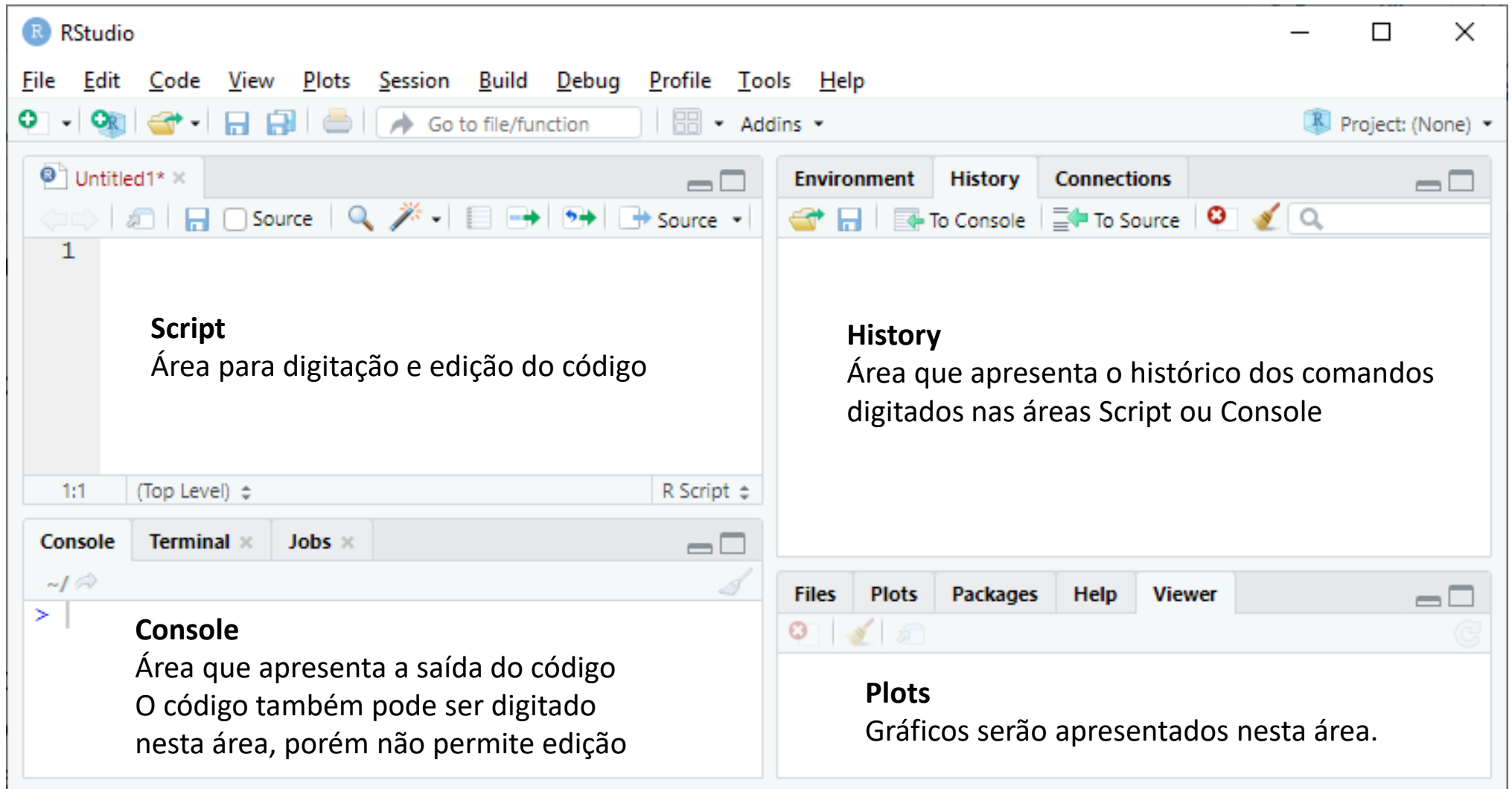
O RStudio está disponível em edições comerciais e de código aberto e pode ser executado nos sistemas operacionais Windows, Mac e Linux ou em um navegador conectado ao RStudio Server ou RStudio Server Pro (Debian, Ubuntu, Red Hat, CentOS e SUSE Linux).

O **download** do RStudio é realizado a partir do seguinte **link**:

<https://rstudio.com/products/rstudio/download/>

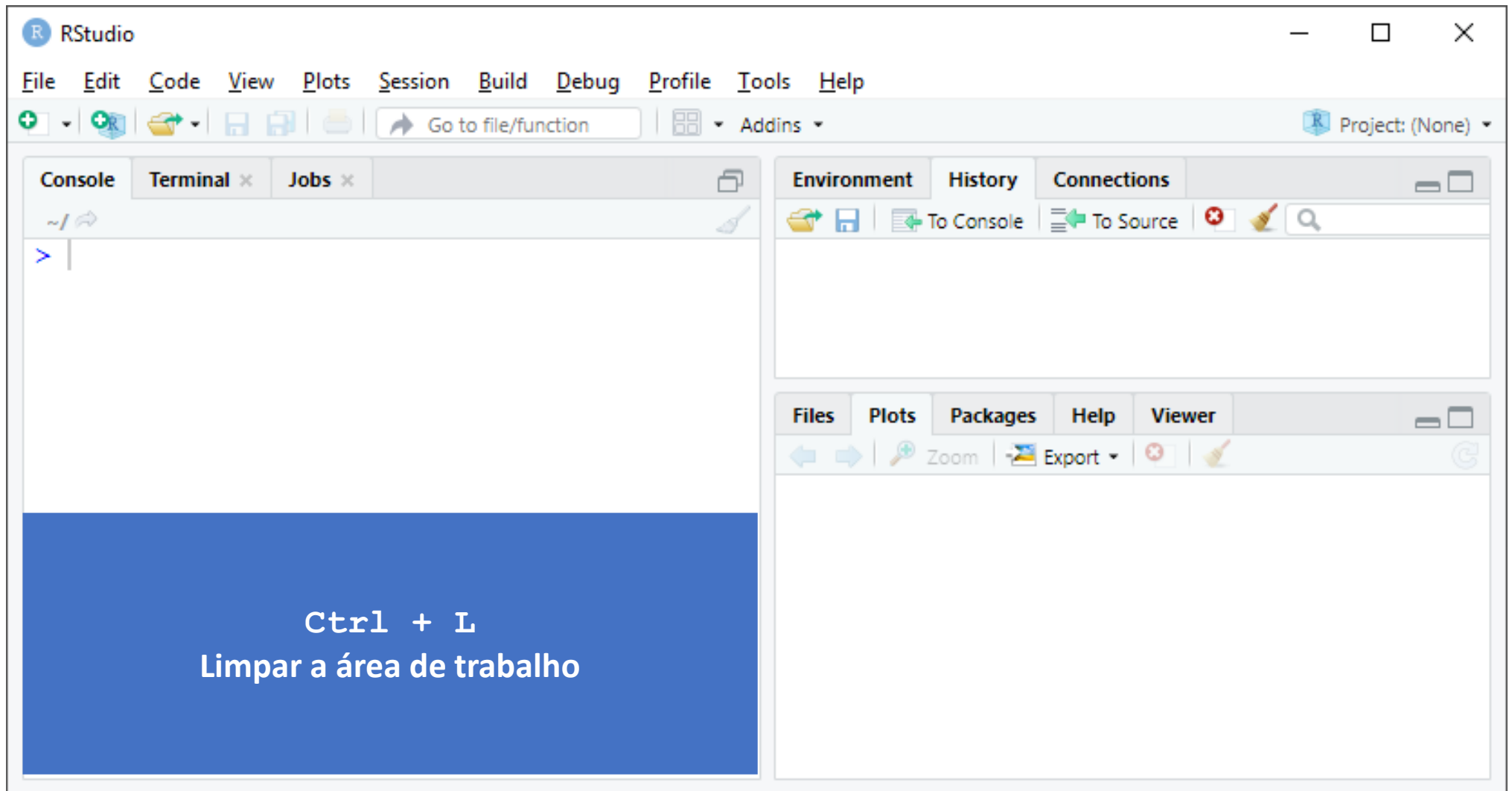


# Data Science

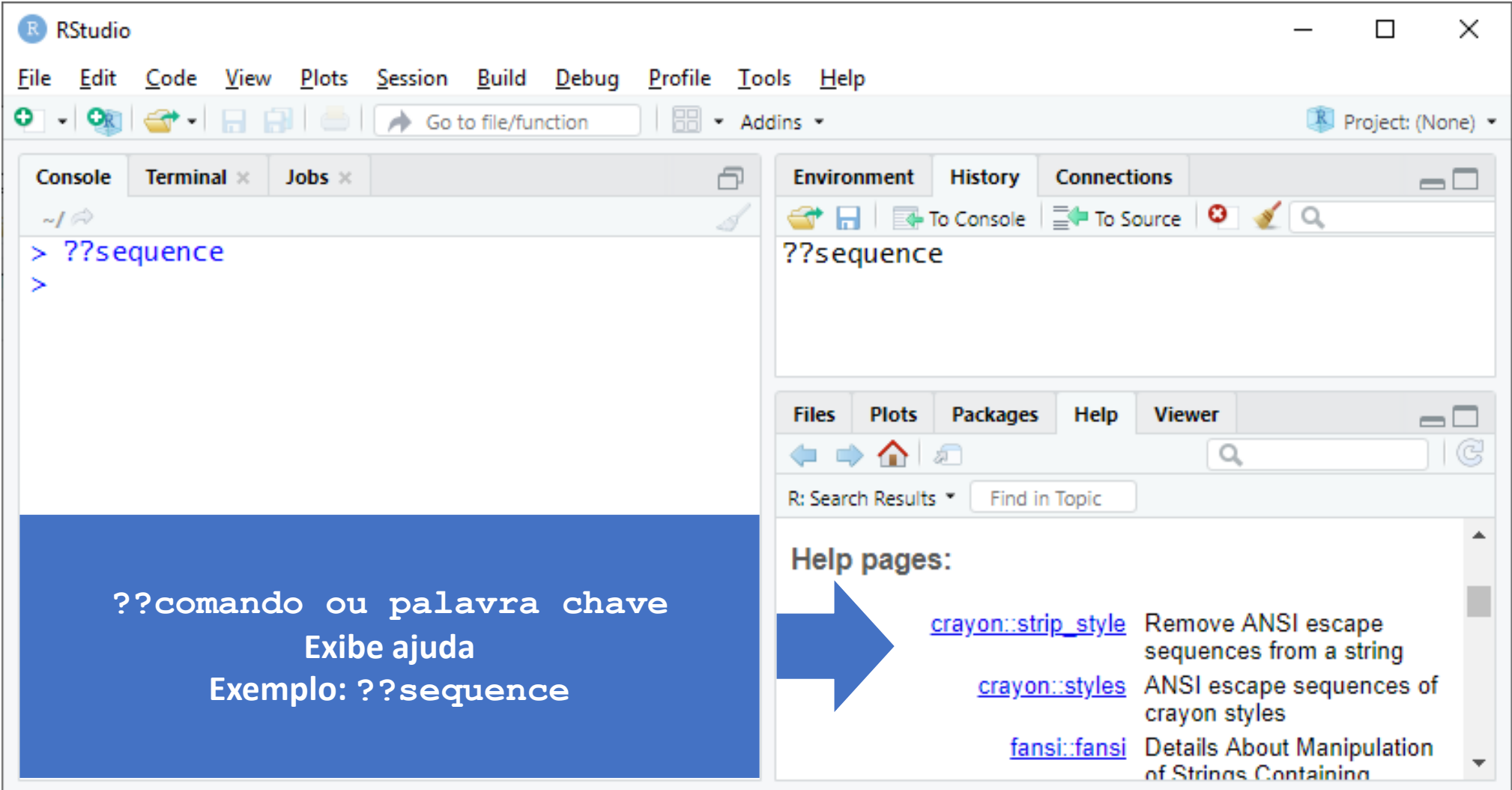


## Comandos Básicos

# Data Science



# Data Science



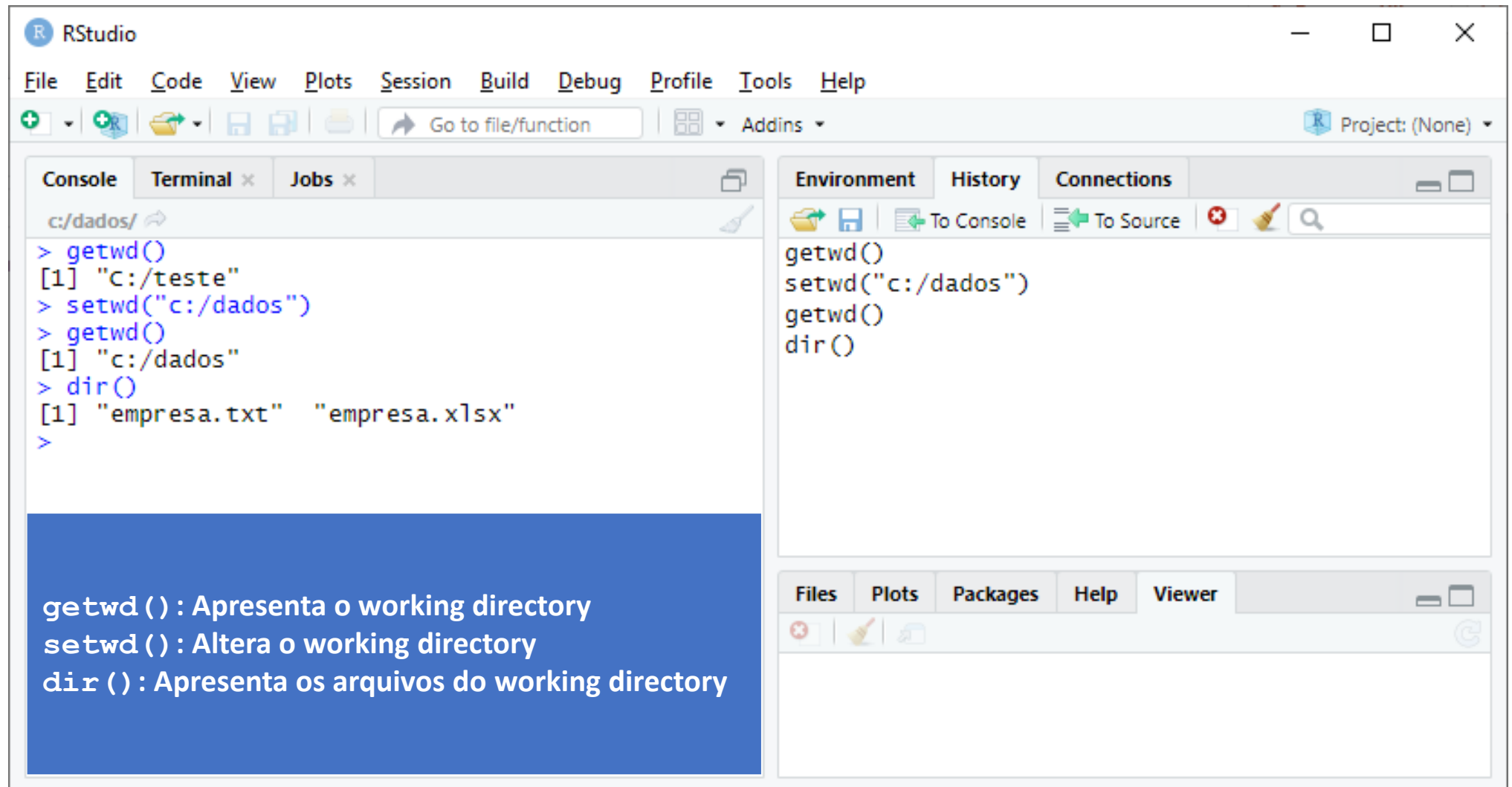
The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for creating a new file, opening a file, saving, printing, and navigating. The main window is divided into several panes. The left pane shows the Console with the command `> ??sequence` entered. The right pane is split into two sections. The top section, labeled 'Environment', shows the command `??sequence`. The bottom section, labeled 'Help pages:', displays a list of search results. A large blue arrow points from the text in the bottom-left box to the search results in the Help pages section.

??comando ou palavra chave  
Exibe ajuda  
Exemplo: `??sequence`

Help pages:

- [crayon::strip\\_style](#) Remove ANSI escape sequences from a string
- [crayon::styles](#) ANSI escape sequences of crayon styles
- [fansit::fansit](#) Details About Manipulation of Strings Containing

# Data Science



The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, and other standard operations. The main workspace is divided into several panels. The Console panel on the left shows the following R commands and their output:

```
> getwd()
[1] "c:/teste"
> setwd("c:/dados")
> getwd()
[1] "c:/dados"
> dir()
[1] "empresa.txt" "empresa.xlsx"
>
```

A blue box at the bottom of the Console panel contains the following text:

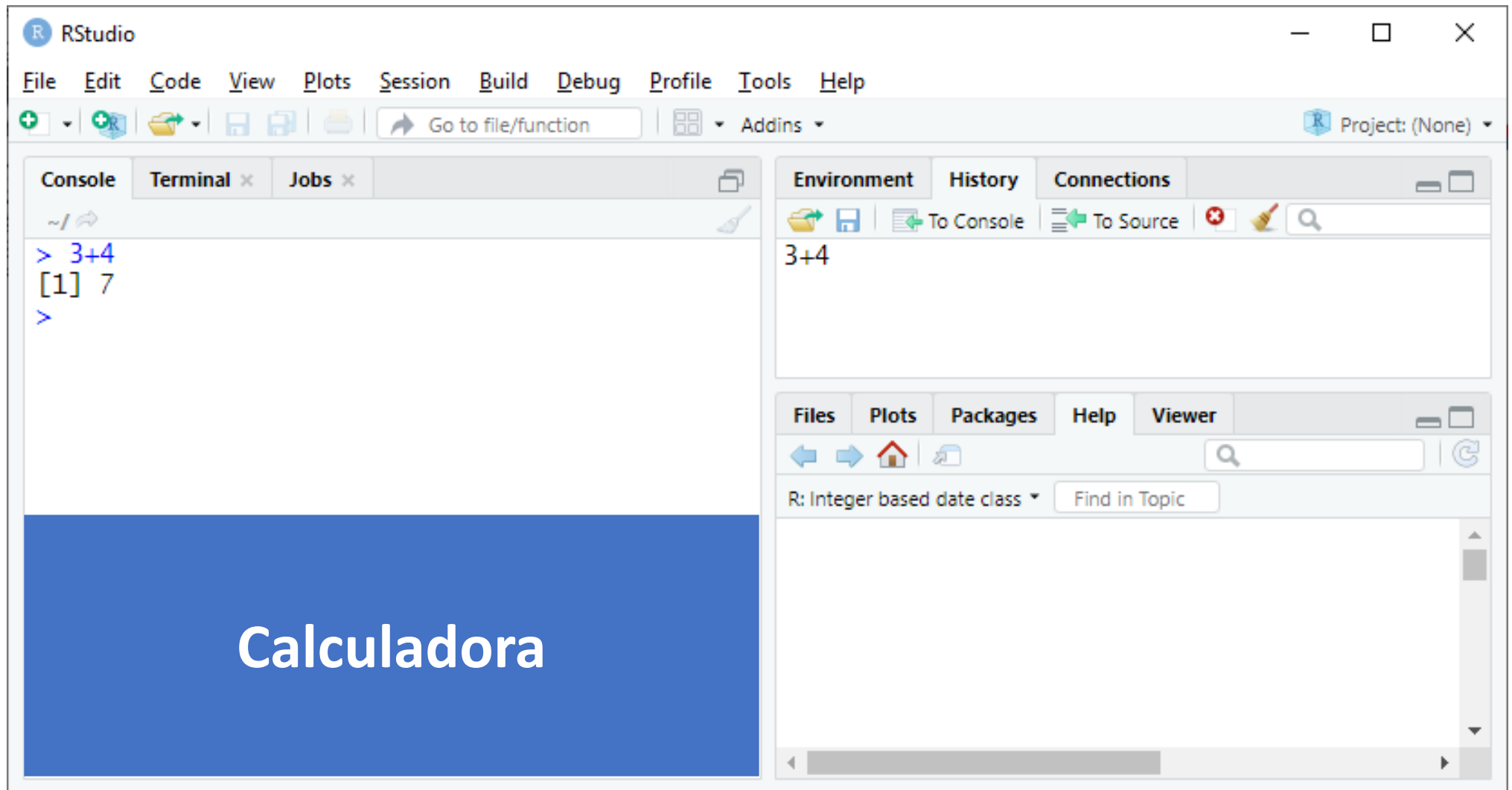
`getwd()` : Apresenta o working directory  
`setwd()` : Altera o working directory  
`dir()` : Apresenta os arquivos do working directory

The Environment panel on the right shows the current objects in the workspace:

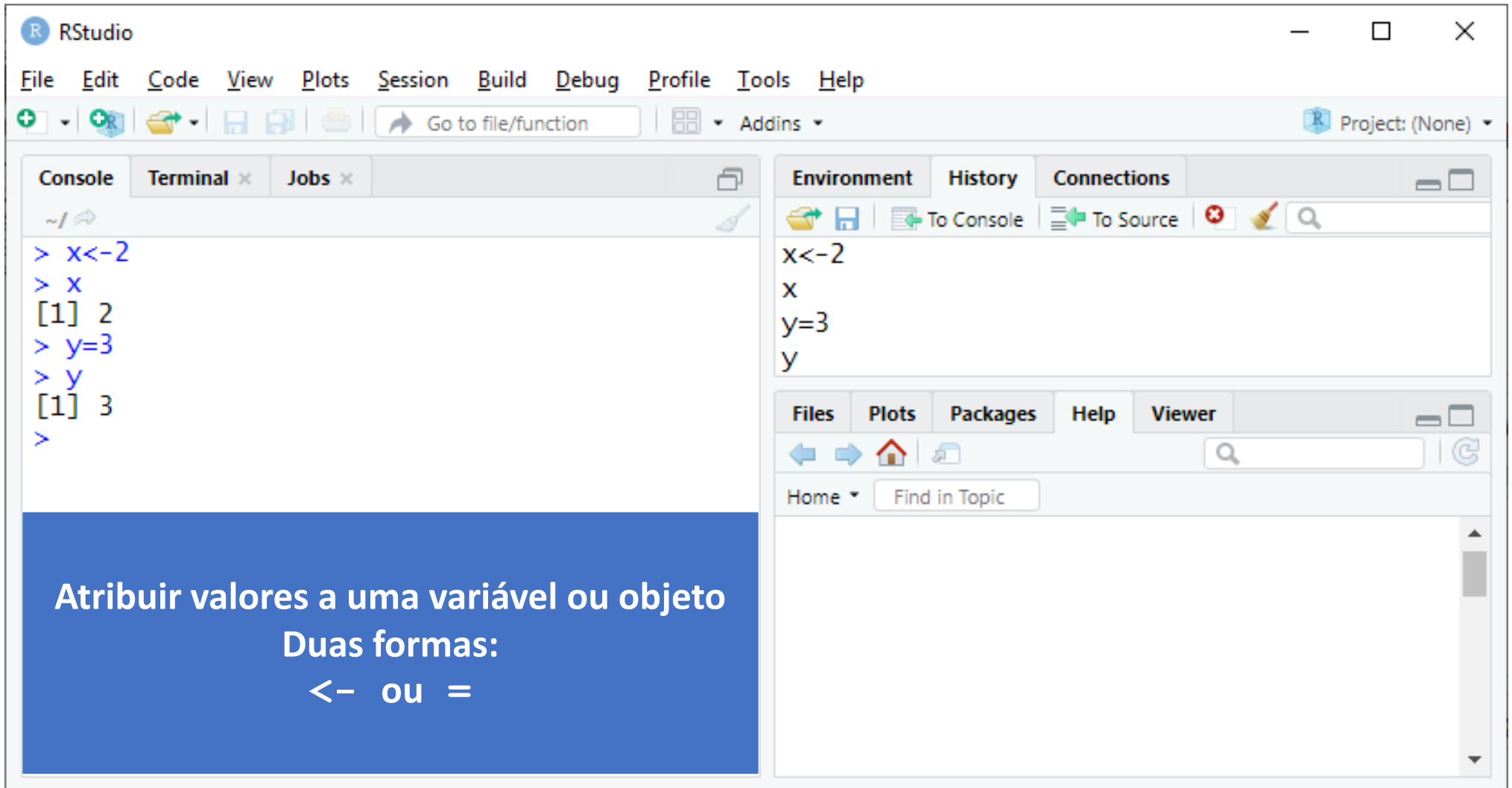
```
getwd()
setwd("c:/dados")
getwd()
dir()
```

The Files panel at the bottom shows the current directory structure.

# Data Science



# Data Science



The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, and other functions. The main workspace is divided into four panes: Console, Environment, History, and Connections. The Console pane on the left shows the following R code and output:

```
> x<-2
> x
[1] 2
> y=3
> y
[1] 3
>
```

The Environment pane on the right shows the current objects in the environment:

```
x<-2
x
y=3
y
```

At the bottom of the console pane, there is a blue box with white text that reads:

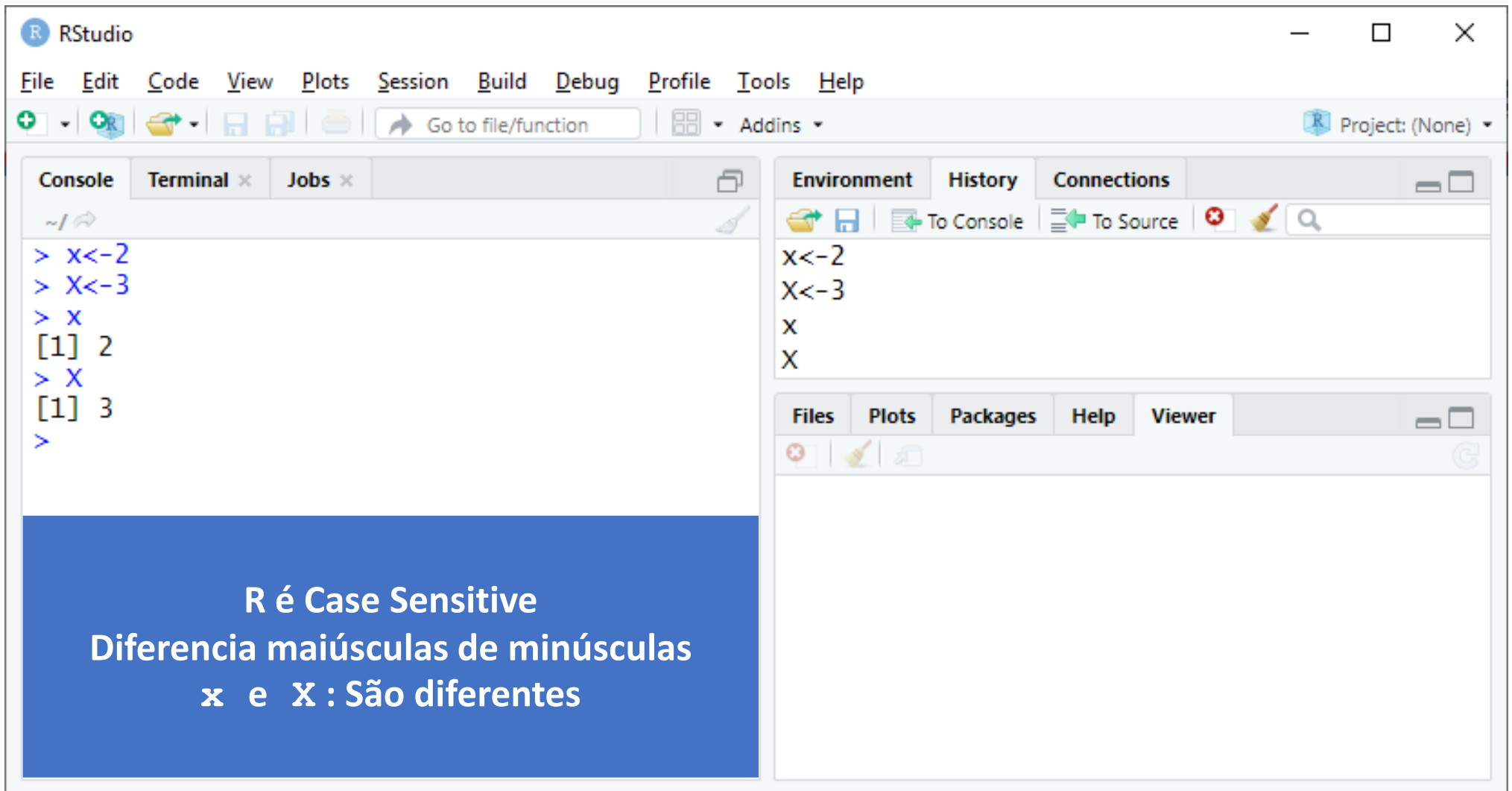
Atribuir valores a uma variável ou objeto  
Duas formas:  
<- ou =

## Comandos ou caracteres

Comandos ou caracteres	Descrição
#	Comentário
;	Separa dois comando na mesma linha
NA	Dado ausente (Not Available)
NaN	Não é um Número (Not a Number)
Inf e -Inf	Infinito positivo e Infinito negativo
q()	Sai do RStudio
ls()	Lista todos os objetos da sessão atual
rm(x)	Remove o objeto x
rm(x, y)	Remove os objetos x e y
c()	Concatena valores. Usado para criar vetores. Exemplo: c(1, 2, 3, 4)



# Data Science



The screenshot shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, and other functions. The main workspace is divided into several panes. On the left, the Console pane shows the following R code and output:

```
> x<-2  
> X<-3  
> x  
[1] 2  
> X  
[1] 3  
>
```

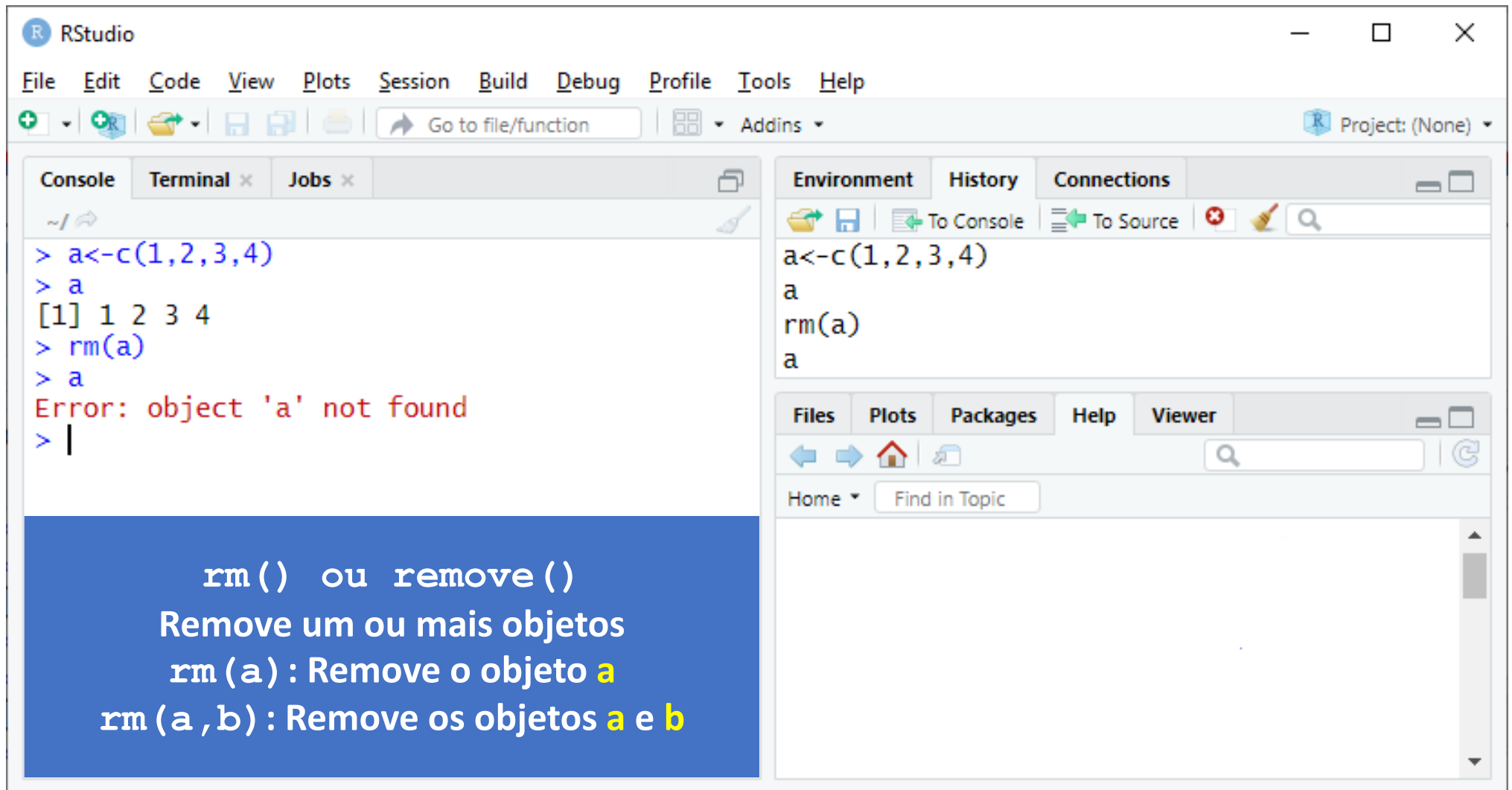
On the right, the Environment pane shows the objects created in the session:

```
x<-2  
X<-3  
x  
X
```

Below the Environment pane are tabs for Files, Plots, Packages, Help, and Viewer. A blue box is overlaid on the bottom left of the RStudio window, containing the text:

**R é Case Sensitive**  
**Diferencia maiúsculas de minúsculas**  
**x e X : São diferentes**

# Data Science



The screenshot shows the RStudio interface. The console on the left displays the following R code and output:

```
> a<-c(1,2,3,4)
> a
[1] 1 2 3 4
> rm(a)
> a
Error: object 'a' not found
> |
```

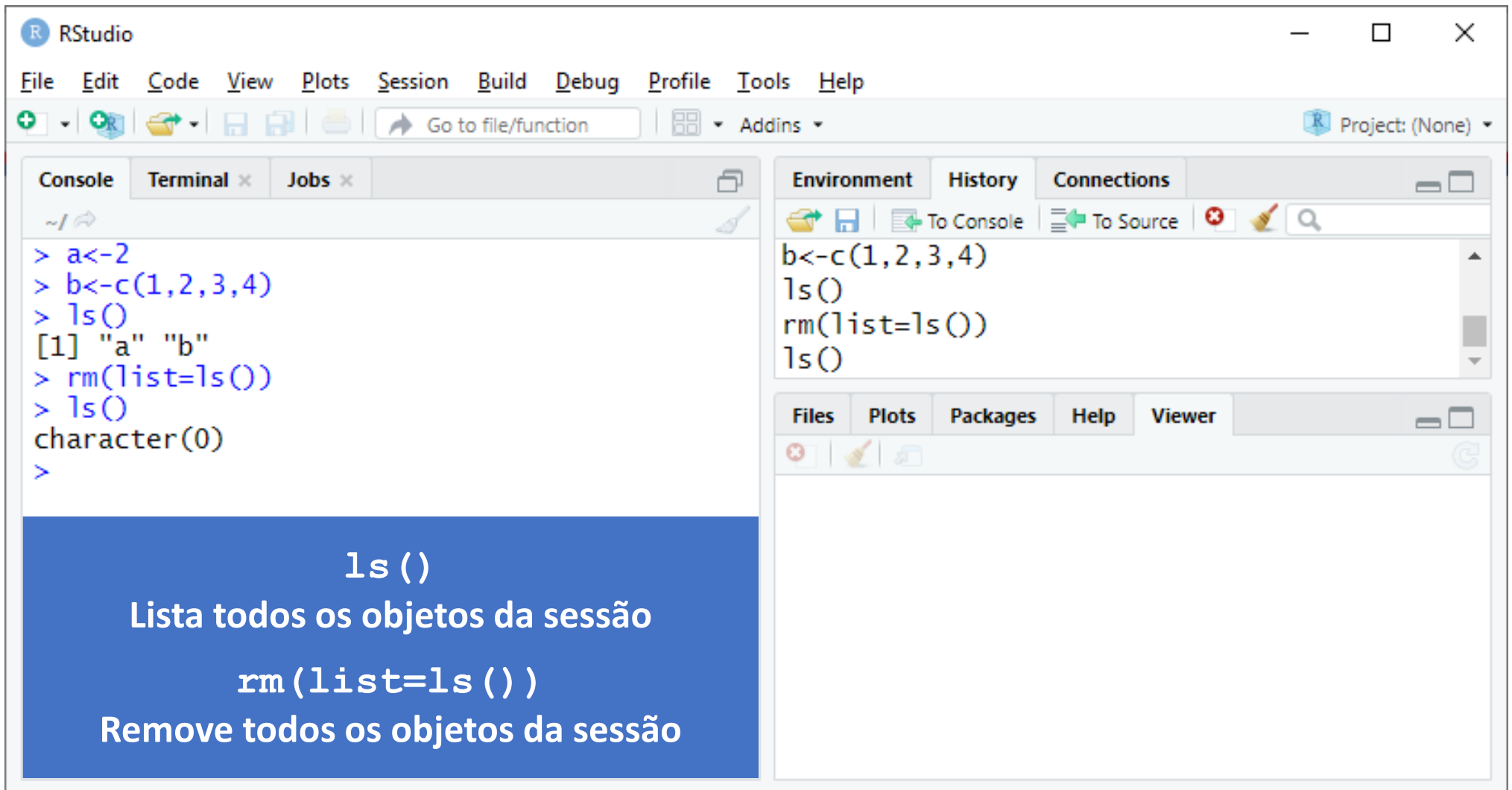
The Environment pane on the right shows the current objects in the workspace:

```
a<-c(1,2,3,4)
a
rm(a)
a
```

Below the console, a blue box contains the following text:

`rm()` ou `remove()`  
Remove um ou mais objetos  
`rm(a)` : Remove o objeto **a**  
`rm(a,b)` : Remove os objetos **a** e **b**

# Data Science



The screenshot shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, printing, and navigating. The main workspace is divided into four panes: Console, Environment, History, and Connections. The Console pane shows the following R code and output:

```
> a<-2
> b<-c(1,2,3,4)
> ls()
[1] "a" "b"
> rm(list=ls())
> ls()
character(0)
>
```

The Environment pane shows the objects currently in the session:

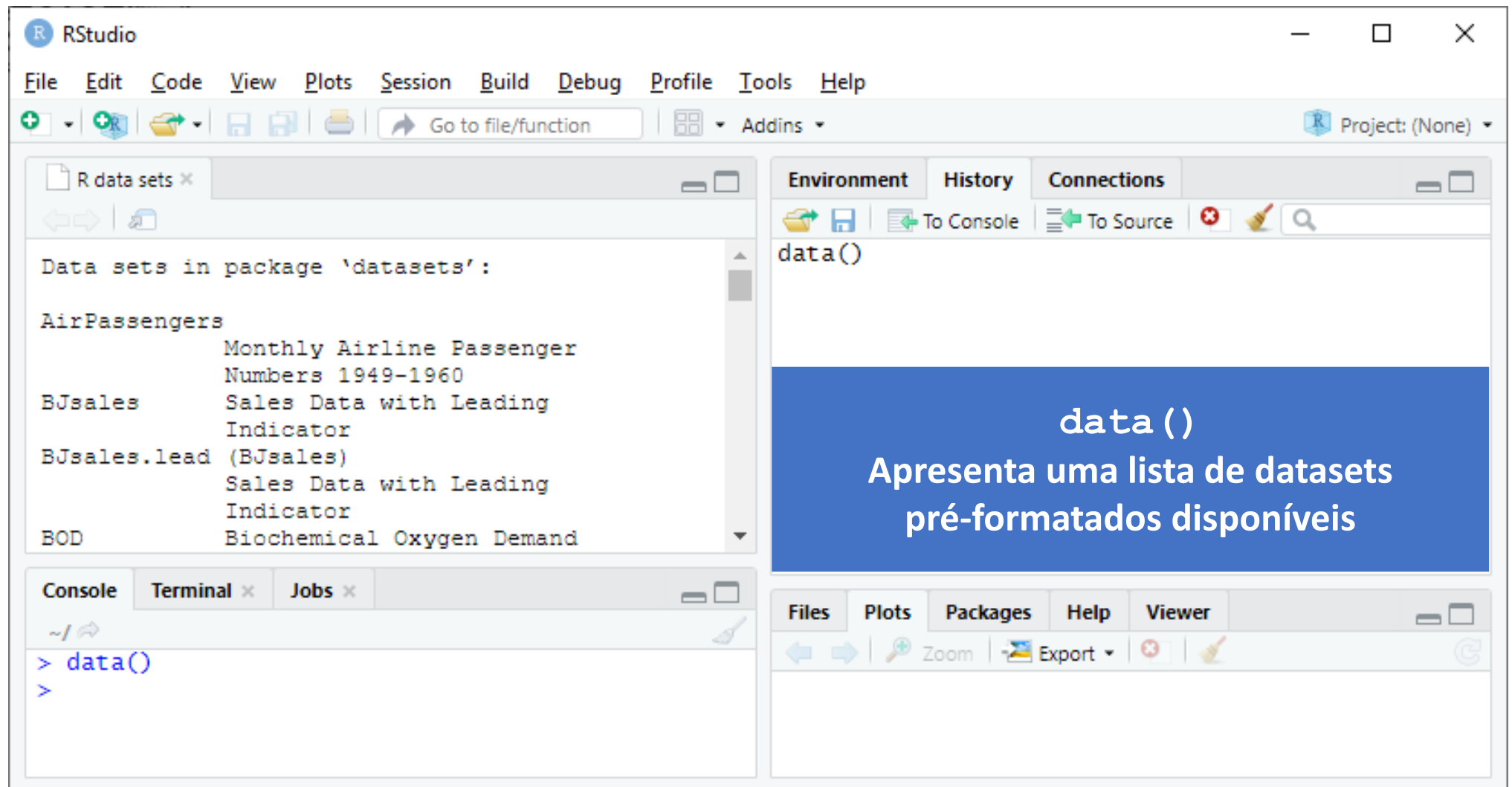
```
b<-c(1,2,3,4)
ls()
rm(list=ls())
ls()
```

Below the screenshot, a blue box contains the following text:

`ls()`  
Lista todos os objetos da sessão

`rm(list=ls())`  
Remove todos os objetos da sessão

# Data Science

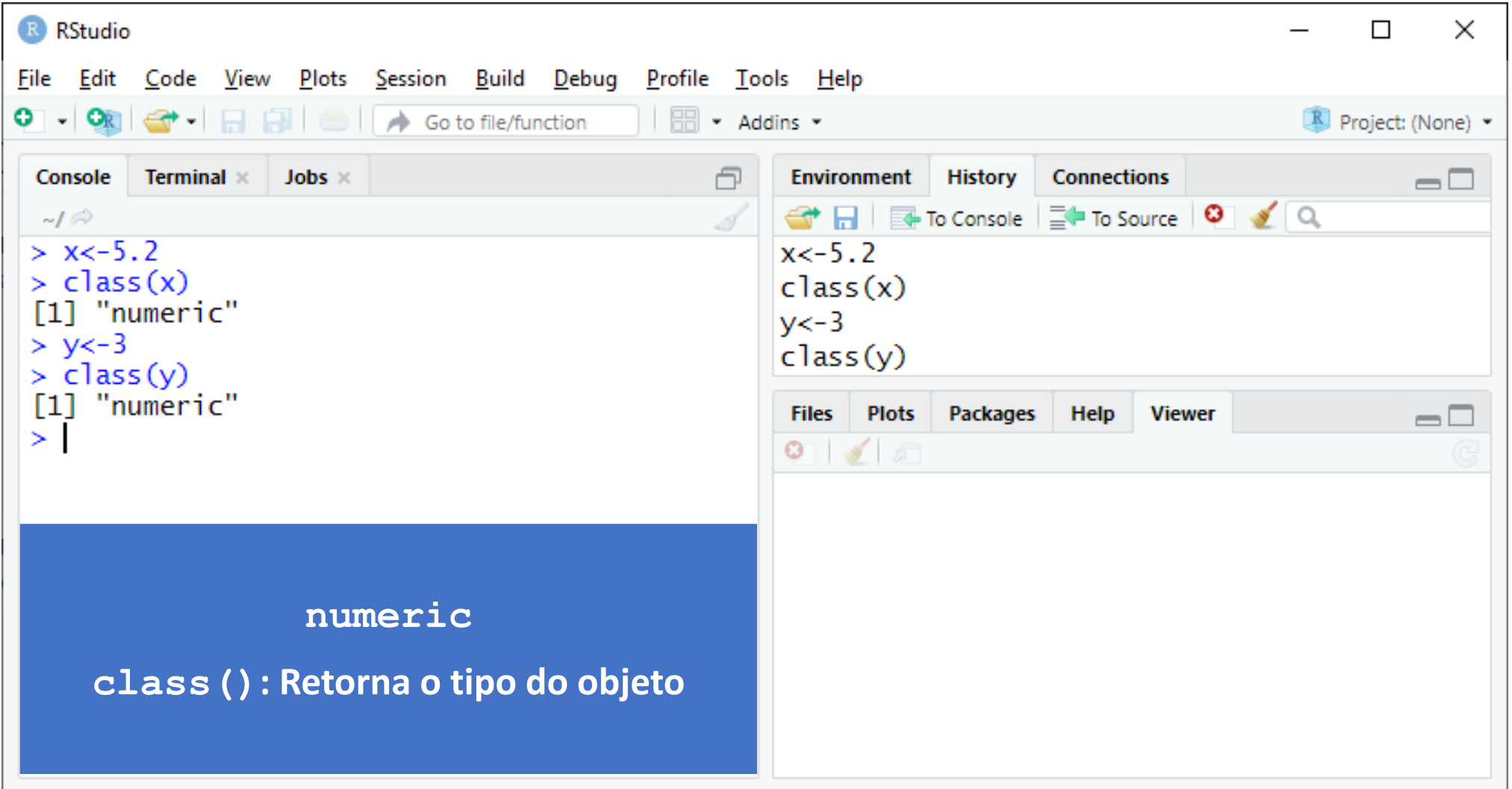


## Tipos de Dados

## Tipos de Dados

Tipo	Descrição
<code>numeric</code>	Números inteiros ou reais
<code>integer</code>	Números inteiros
<code>complex</code>	Números complexos
<code>logical</code>	Booleanos: verdadeiro ou falso (TRUE ou FALSE)
<code>character</code>	Caracteres (texto)

# Data Science



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

Console Terminal x Jobs x

```
> x<-5.2
> class(x)
[1] "numeric"
> y<-3
> class(y)
[1] "numeric"
> |
```

numeric

class(): Retorna o tipo do objeto

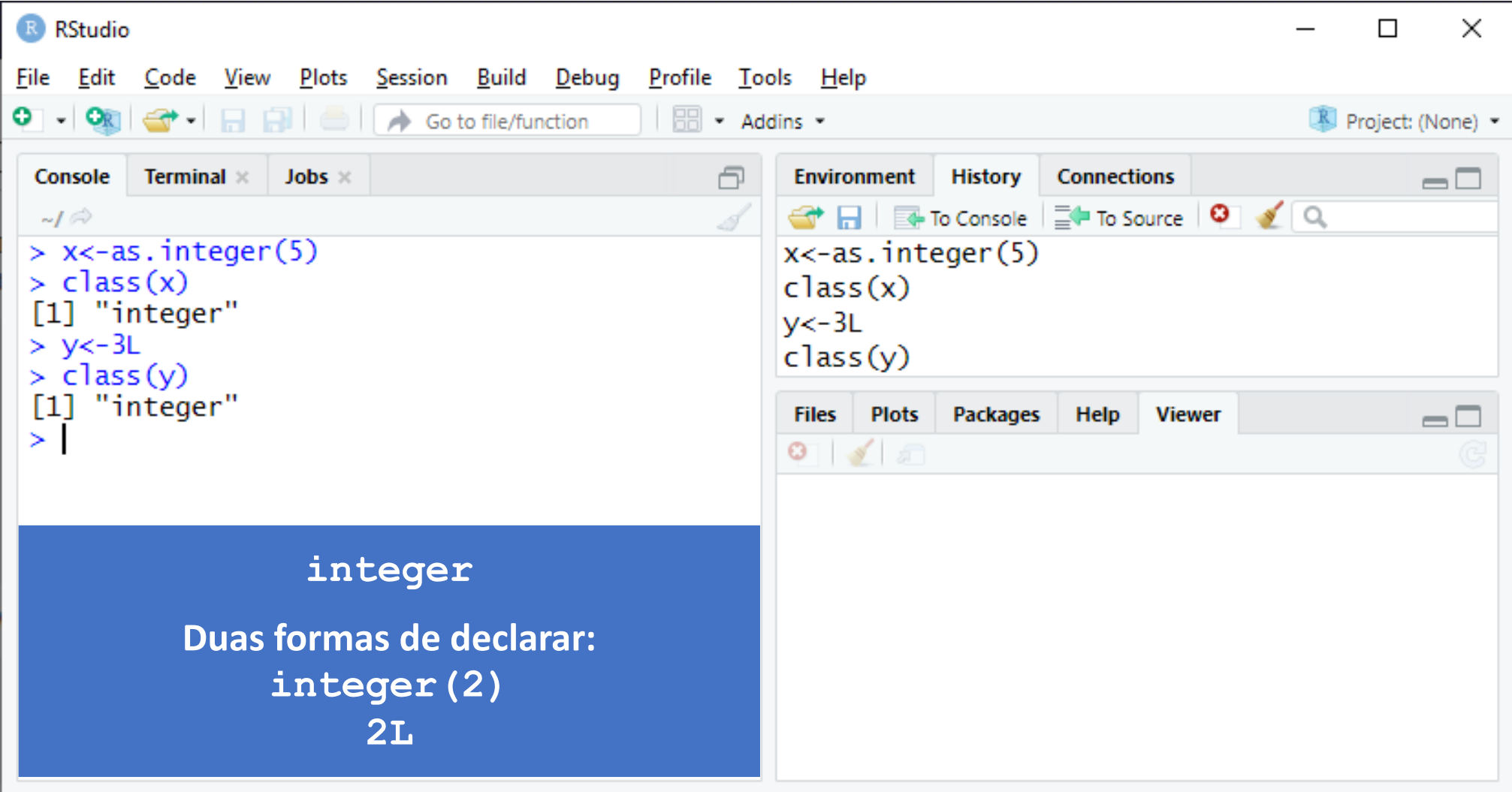
Environment History Connections

To Console To Source

x<-5.2  
class(x)  
y<-3  
class(y)

Files Plots Packages Help Viewer

# Data Science



The screenshot shows the RStudio interface with the following content:

**Console:**

```
> x<-as.integer(5)
> class(x)
[1] "integer"
> y<-3L
> class(y)
[1] "integer"
> |
```

**Environment:**

```
x<-as.integer(5)
class(x)
y<-3L
class(y)
```

**Overlay Text:**

**integer**

Duas formas de declarar:

```
integer(2)
2L
```



# Data Science

The screenshot shows the RStudio environment. The console on the left contains the following R code and output:

```
> x<-1+2i
> class(x)
[1] "complex"
> sqrt(-1)
[1] NaN
Warning message:
In sqrt(-1) : NaNs produced
> sqrt(as.complex(-1))
[1] 0+1i
```

Below the console, a blue box contains the following text:

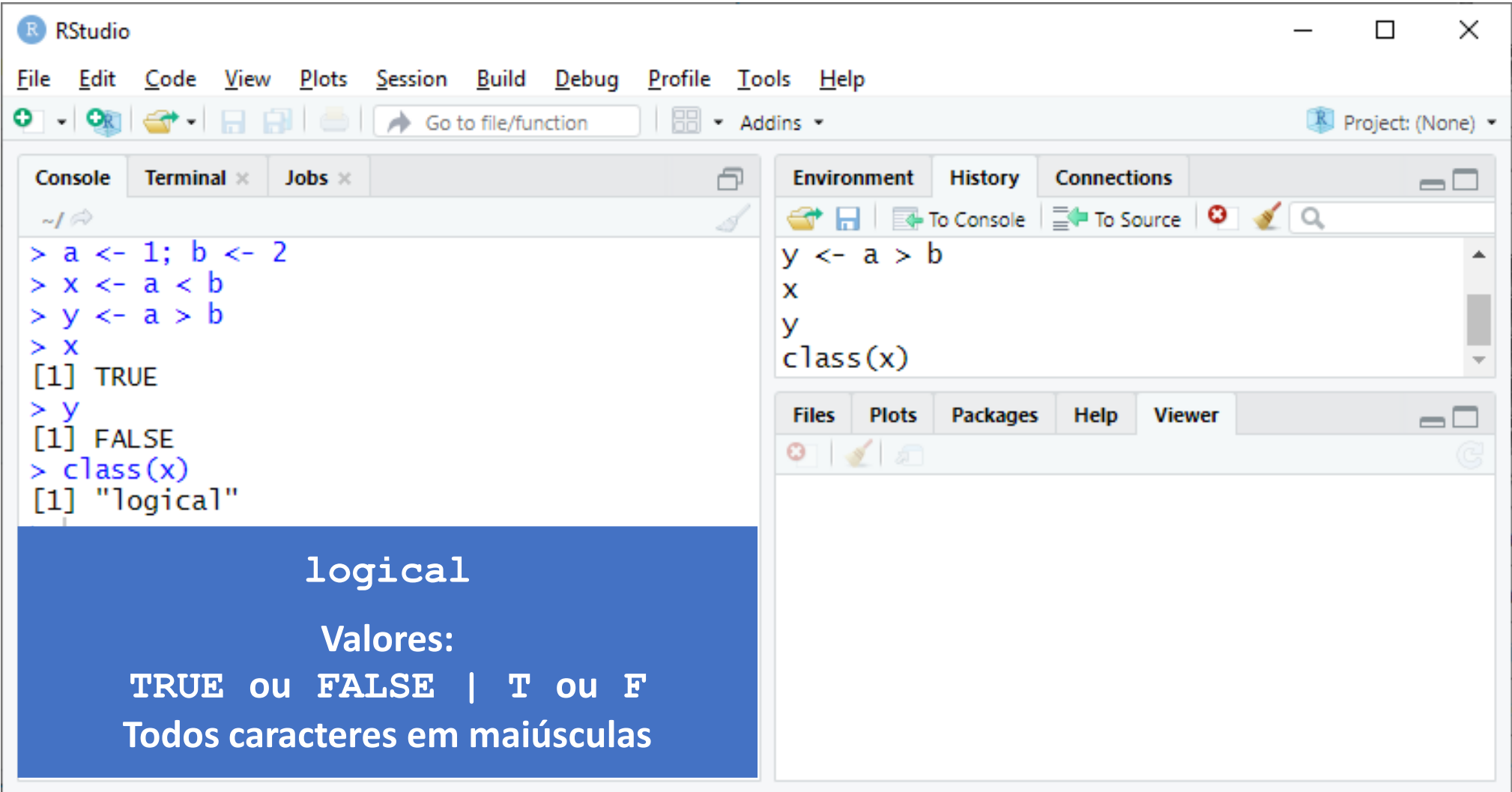
**complex**

**Números complexos:**  
Parte Real + parte imaginária i  
 $i = \sqrt{-1}$

The right pane of RStudio shows the Environment tab with the following objects:

```
x<-1+2i
class(x)
sqrt(-1)
sqrt(as.complex(-1))
```

# Data Science



The screenshot shows the RStudio environment. The console on the left contains the following R code and output:

```
> a <- 1; b <- 2
> x <- a < b
> y <- a > b
> x
[1] TRUE
> y
[1] FALSE
> class(x)
[1] "logical"
```

Below the console, a blue box contains the following text:

logical

Valores:

TRUE ou FALSE | T ou F

Todos caracteres em maiúsculas

The right pane of RStudio shows the Environment tab with the following objects:

```
y <- a > b
x
y
class(x)
```

# Data Science



The screenshot shows the RStudio interface with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for creating a new file, opening a file, saving, and a search bar with the text "Go to file/function".
- Console:** Contains the following R code and output:

```
> x <- 'John Smith'
> class(x)
[1] "character"
> y <- 3.14
> class(y)
[1] "numeric"
> z <- as.character(3.14)
> class(z)
[1] "character"
```
- Environment:** Shows the objects created in the environment:

```
y <- 3.14
class(y)
z <- as.character(3.14)
class(z)
```
- Files, Plots, Packages, Help, Viewer:** These panels are visible at the bottom of the interface.

**character**  
Devem ser inseridos entre aspas

## Adicionando Pacotes

## Adicionando Pacotes (Packages)

Packages são compostos por uma coleção de funções R, código compilado e dados de amostra, armazenados no diretório "library" no ambiente R.

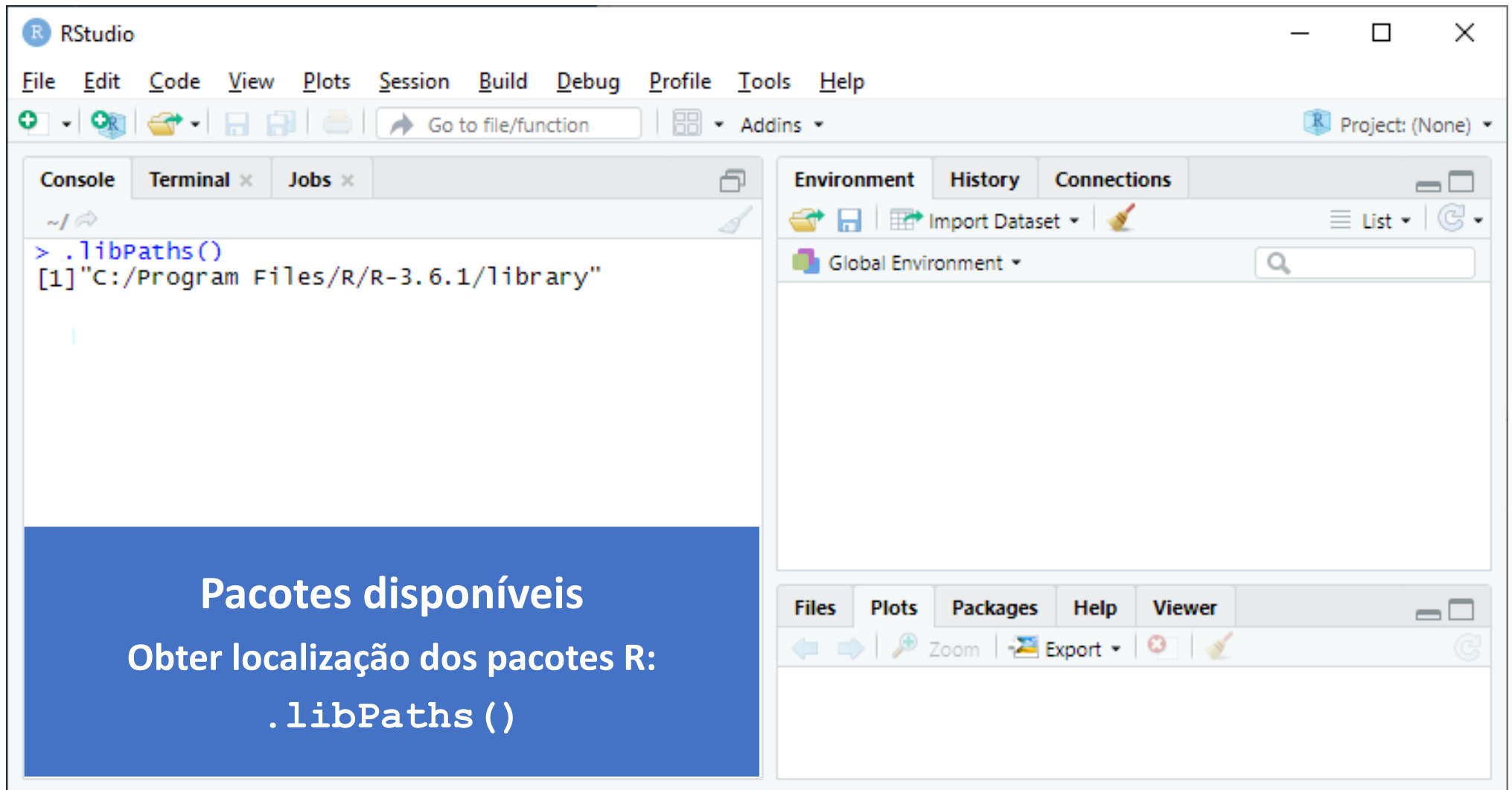
Um conjunto padrão de packages é instalado. Outros pacotes com funções específicas precisam ser instalados, conforme a necessidade.

Quando o console R é inicializado, apenas o conjunto padrão de pacotes fica disponível. Outros pacotes, mesmo que já estejam instalados, devem ser carregados explicitamente para serem usados.

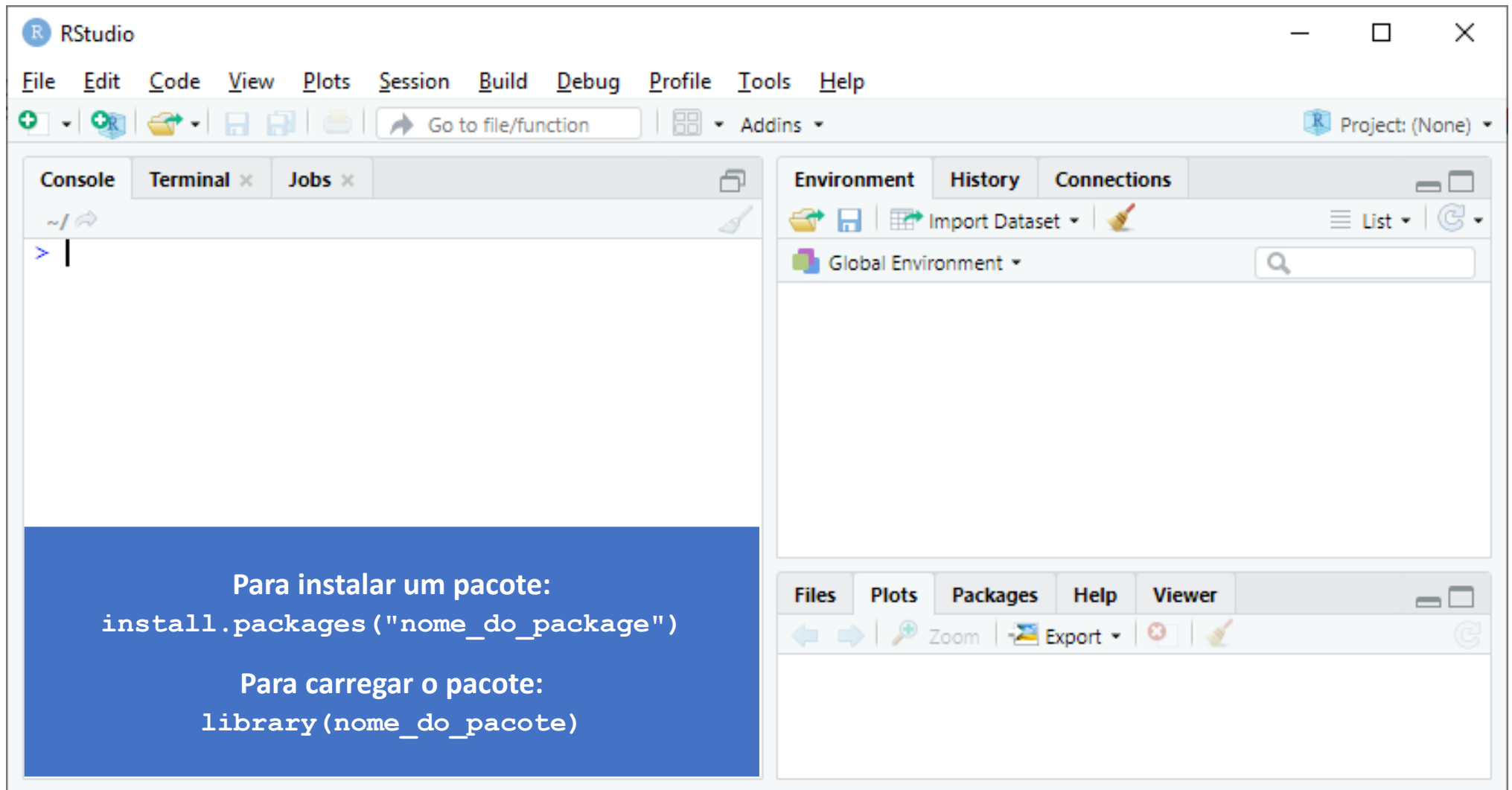
Todos os pacotes disponíveis na linguagem R estão no seguinte link:

[https://cran.r-project.org/web/packages/available\\_packages\\_by\\_name.html/](https://cran.r-project.org/web/packages/available_packages_by_name.html/)

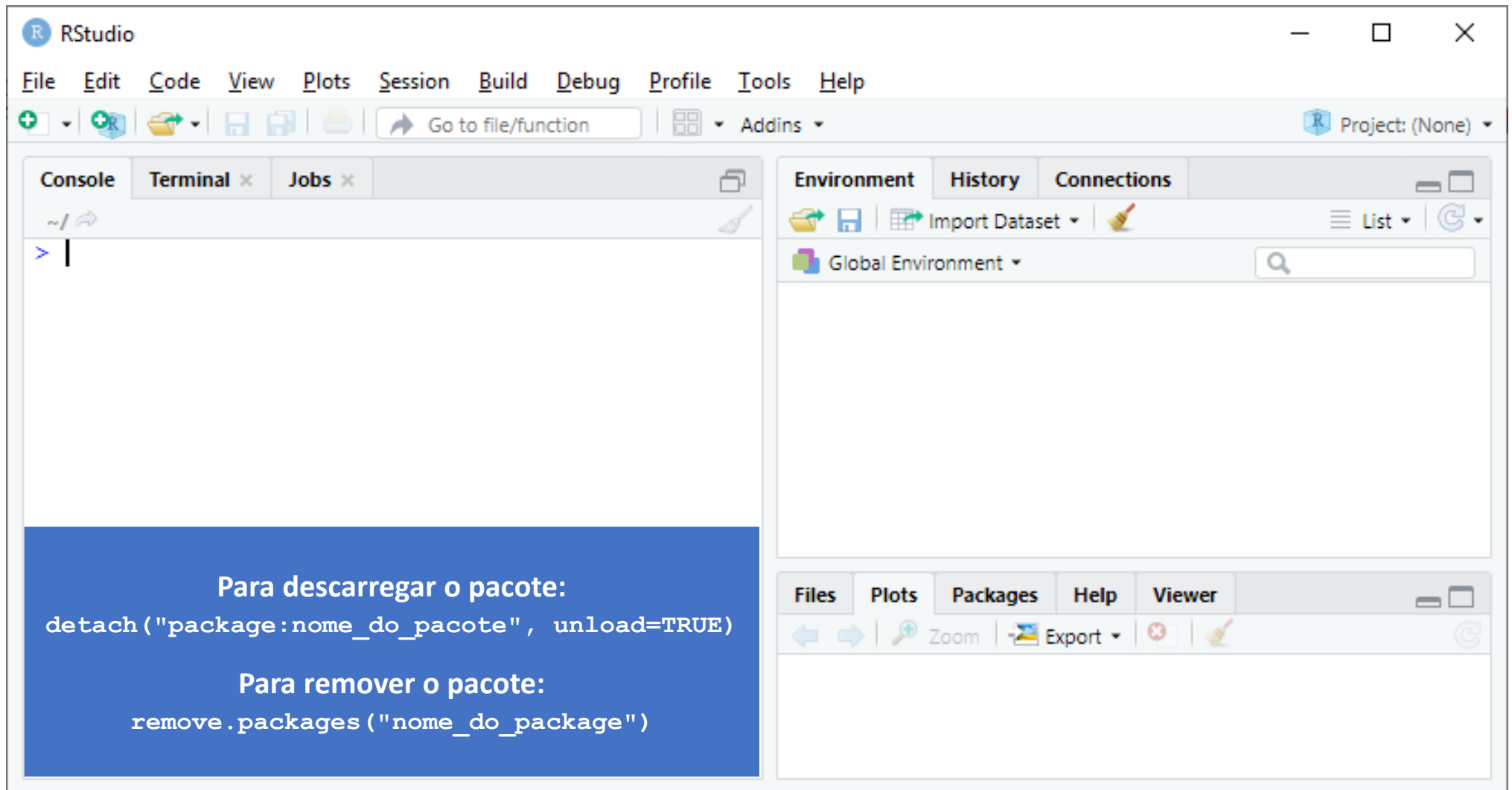
# Data Science



# Data Science

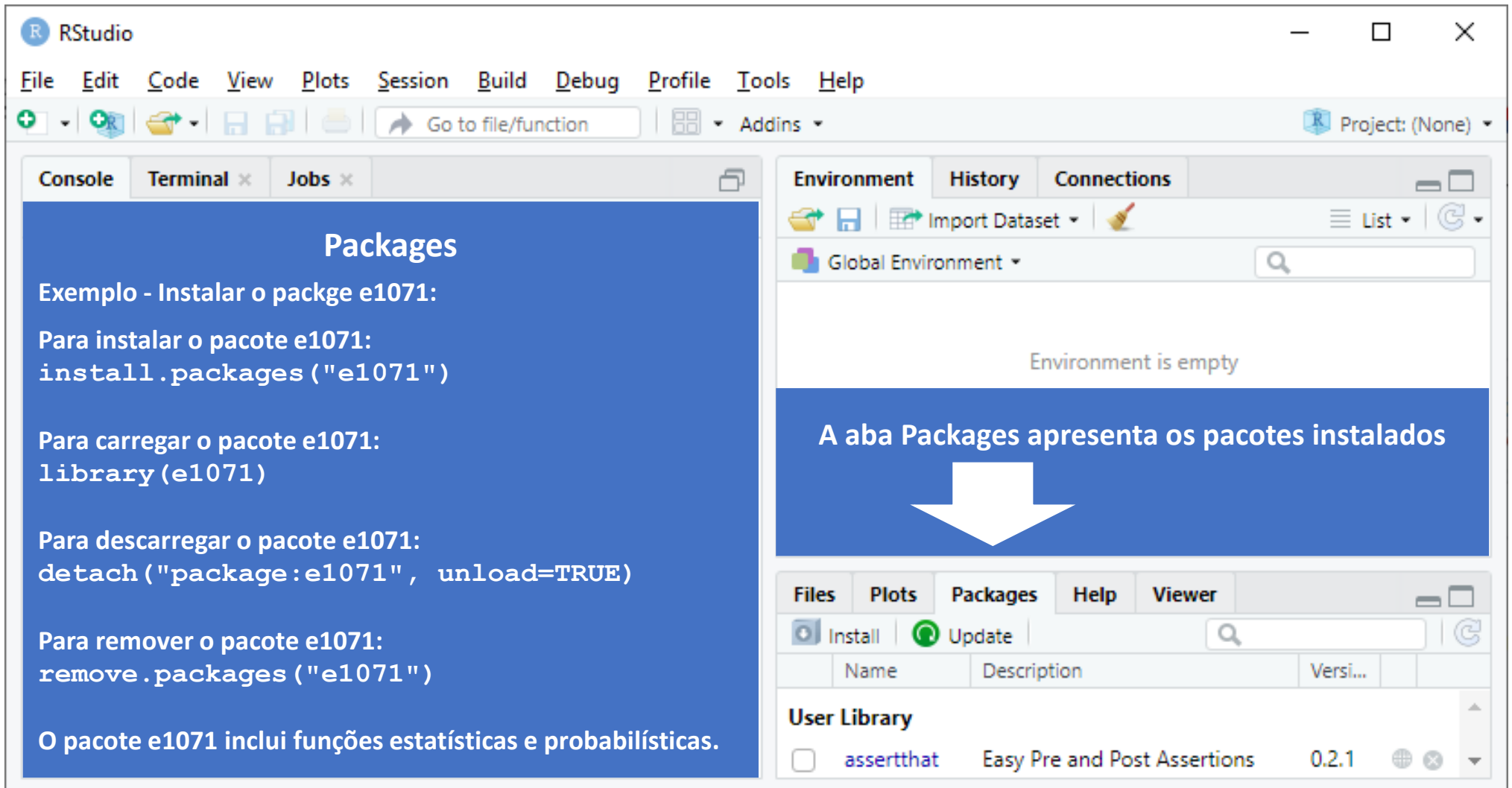


# Data Science





# Data Science



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for creating a new file, opening a file, saving, and other standard functions. The main window is divided into several panes. The left pane is the Console, which is currently displaying the output of the 'install.packages' function. The right pane is the Environment pane, which shows the 'Global Environment' and a search bar. Below the Environment pane is the Packages pane, which displays a table of installed packages. A large blue box with a white arrow points from the text 'A aba Packages apresenta os pacotes instalados' to the Packages pane.

**Packages**

Exemplo - Instalar o package e1071:

Para instalar o pacote e1071:

```
install.packages("e1071")
```

Para carregar o pacote e1071:

```
library(e1071)
```

Para descarregar o pacote e1071:

```
detach("package:e1071", unload=TRUE)
```

Para remover o pacote e1071:

```
remove.packages("e1071")
```

O pacote e1071 inclui funções estatísticas e probabilísticas.

**A aba Packages apresenta os pacotes instalados**

Name	Description	Versi...
assertthat	Easy Pre and Post Assertions	0.2.1

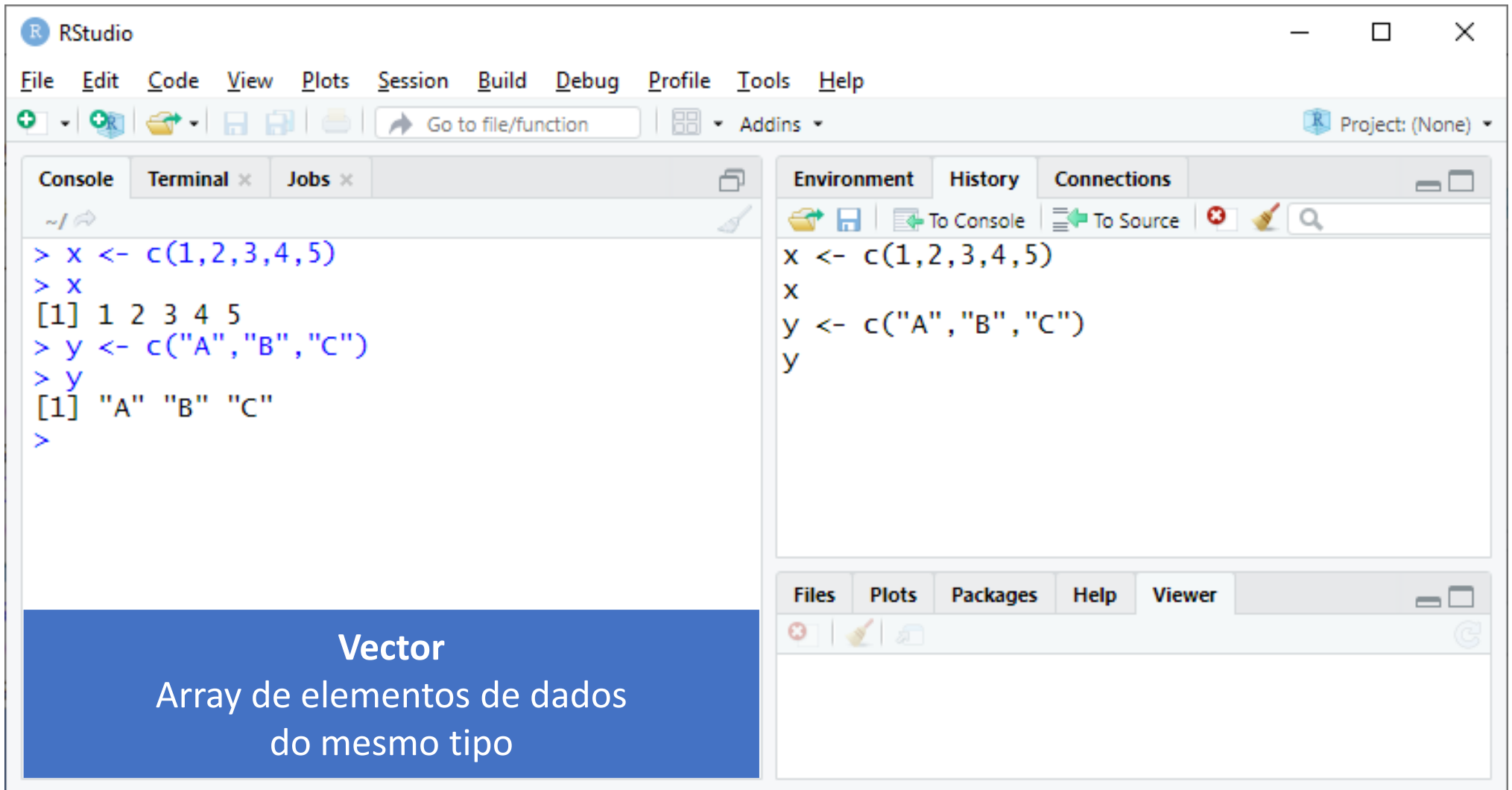
## Estruturas Básicas do R

## Estruturas Básicas do R

Estrutura	Descrição
<code>vector</code>	Array com uma dimensão (uma 'linha')
<code>matrix</code>	Array com duas dimensões ('linhas' e 'colunas')
<code>array</code>	Objeto que pode conter uma ou mais dimensões
<code>factor</code>	Vetor de dados categóricos
<code>data.frame</code>	Objeto com duas dimensões (armazena elementos de diferentes tipos)
<code>list</code>	Objeto que permite combinar diferentes estruturas

## Vetores

# Data Science



The screenshot shows the RStudio environment with the following components:

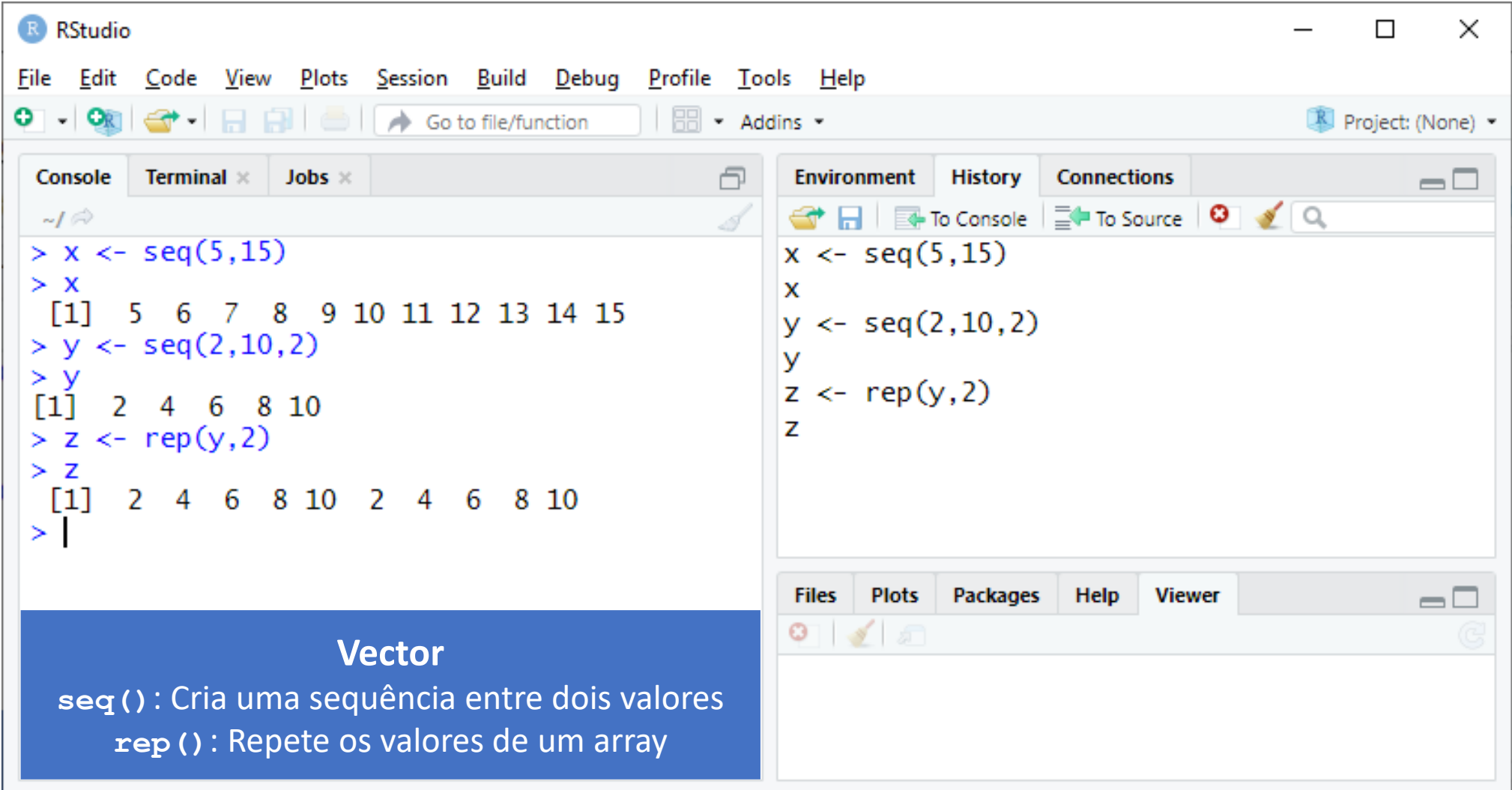
- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for creating a new file, opening a file, saving, printing, and a search bar. The project is set to "Project: (None)".
- Console:** Displays the following R code and output:

```
> x <- c(1,2,3,4,5)
> x
[1] 1 2 3 4 5
> y <- c("A","B","C")
> y
[1] "A" "B" "C"
>
```
- Environment:** Shows the objects created in the environment:

```
x <- c(1,2,3,4,5)
x
y <- c("A","B","C")
y
```
- Bottom Panel:** Contains tabs for Files, Plots, Packages, Help, and Viewer.

**Vector**  
Array de elementos de dados do mesmo tipo

# Data Science



The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main workspace is divided into four panes: Console, Environment, History, and Connections. The Console pane shows the following R code and its output:

```
> x <- seq(5,15)
> x
[1] 5 6 7 8 9 10 11 12 13 14 15
> y <- seq(2,10,2)
> y
[1] 2 4 6 8 10
> z <- rep(y,2)
> z
[1] 2 4 6 8 10 2 4 6 8 10
> |
```

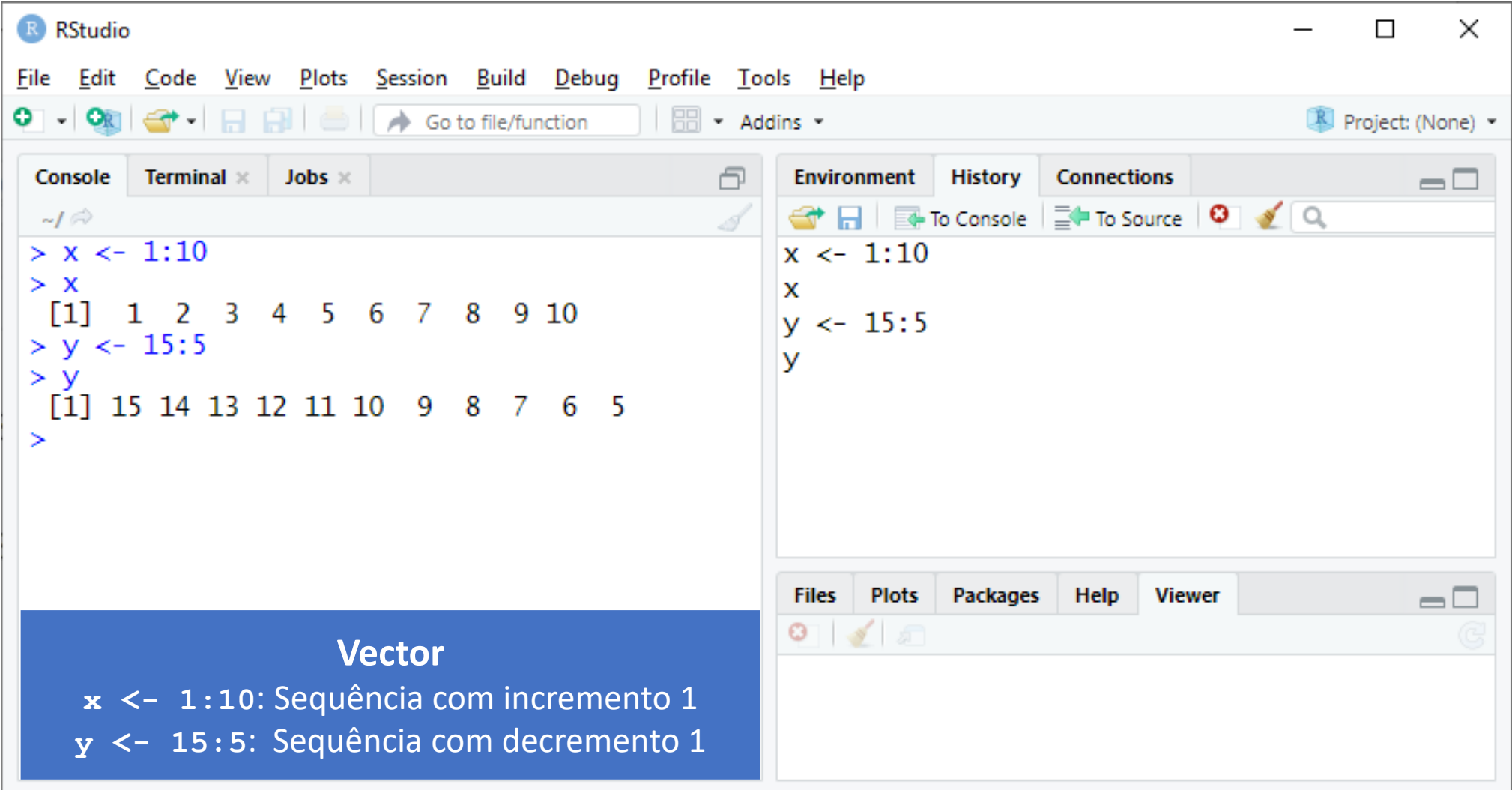
The Environment pane shows the variables created in the script:

```
x <- seq(5,15)
x
y <- seq(2,10,2)
y
z <- rep(y,2)
z
```

At the bottom of the RStudio window, there is a blue box with the following text:

**Vector**  
`seq()` : Cria uma sequência entre dois valores  
`rep()` : Repete os valores de um array

# Data Science



The screenshot shows the RStudio environment with the following components:

- Console:** Displays the execution of two R commands:

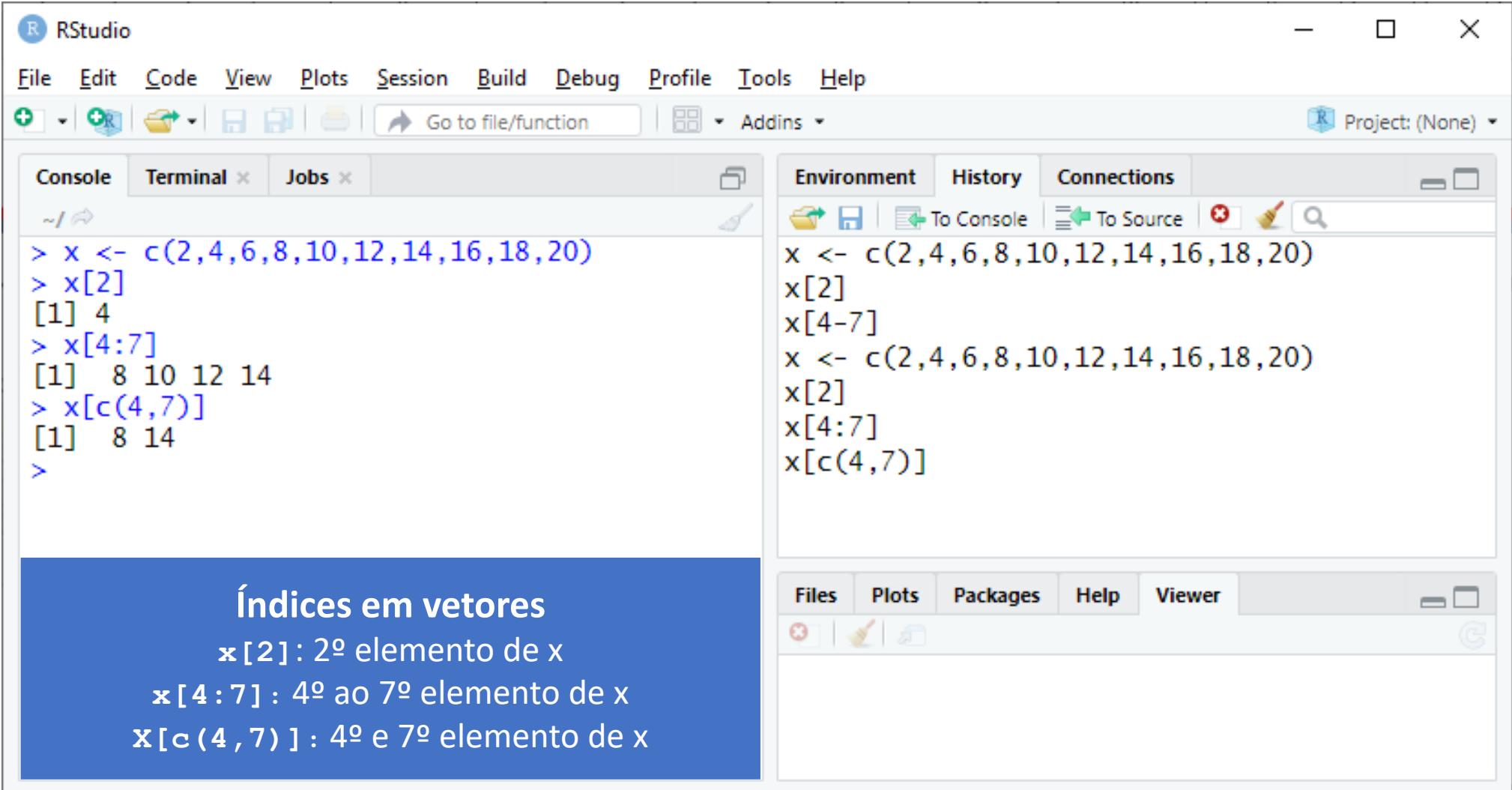
```
> x <- 1:10  
> x  
[1] 1 2 3 4 5 6 7 8 9 10  
> y <- 15:5  
> y  
[1] 15 14 13 12 11 10 9 8 7 6 5  
>
```
- Environment:** Shows the objects created in the environment:

```
x <- 1:10  
x  
y <- 15:5  
y
```
- Files, Plots, Packages, Help, Viewer:** These panels are visible at the bottom of the interface.

**Vector**

- `x <- 1:10`: Sequência com incremento 1
- `y <- 15:5`: Sequência com decremento 1

# Data Science



The screenshot shows the RStudio interface with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for creating a new file, opening a file, saving, printing, and navigating to a file/function. A search bar and an 'Addins' dropdown are also present.
- Project:** Project: (None)
- Console:** Displays the following R code and output:

```
> x <- c(2,4,6,8,10,12,14,16,18,20)
> x[2]
[1] 4
> x[4:7]
[1] 8 10 12 14
> x[c(4,7)]
[1] 8 14
>
```
- Environment:** Displays the following R code and output:

```
x <- c(2,4,6,8,10,12,14,16,18,20)
x[2]
x[4-7]
x <- c(2,4,6,8,10,12,14,16,18,20)
x[2]
x[4:7]
x[c(4,7)]
```
- Bottom Panel:** Includes tabs for Files, Plots, Packages, Help, and Viewer.

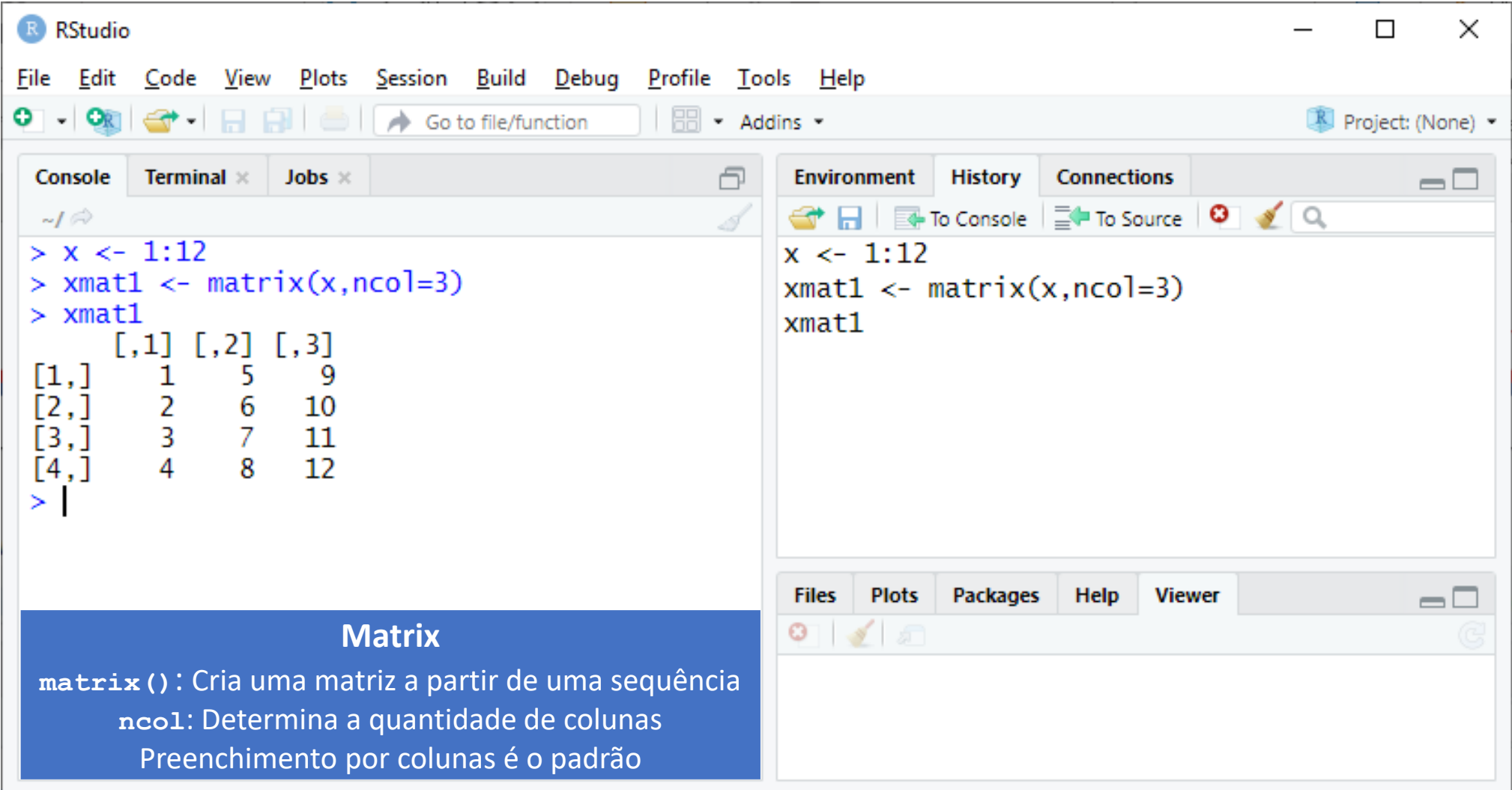
**Índices em vetores**

- $x[2]$ : 2º elemento de  $x$
- $x[4:7]$ : 4º ao 7º elemento de  $x$
- $x[c(4,7)]$ : 4º e 7º elemento de  $x$



## Matrizes

# Data Science



The screenshot shows the RStudio environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating new files, opening files, saving, and other standard operations. The main workspace is divided into several panes. On the left, the Console pane shows the following R code and its output:

```
> x <- 1:12
> xmat1 <- matrix(x,ncol=3)
> xmat1
```

	[,1]	[,2]	[,3]
[1,]	1	5	9
[2,]	2	6	10
[3,]	3	7	11
[4,]	4	8	12

On the right, the Environment pane shows the objects created in the workspace: x, xmat1, and xmat1. The bottom right pane contains tabs for Files, Plots, Packages, Help, and Viewer. A blue box at the bottom left of the RStudio window contains the following text:

**Matrix**

`matrix()`: Cria uma matriz a partir de uma sequência  
ncol: Determina a quantidade de colunas  
Preenchimento por colunas é o padrão

# Data Science



The screenshot shows the RStudio environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, printing, and navigating. The main workspace is divided into four panes: Console, Environment, History, and Connections. The Console pane shows the following R code and its output:

```
> x <- 1:12
> xmat2 <- matrix(x,nrow=3)
> xmat2
```

	[,1]	[,2]	[,3]	[,4]
[1,]	1	4	7	10
[2,]	2	5	8	11
[3,]	3	6	9	12

The Environment pane shows the objects `x` and `xmat2`. The History pane shows the commands entered in the console. The bottom pane is divided into Files, Plots, Packages, Help, and Viewer. A blue box at the bottom left contains the following text:

**Matrix**  
`matrix()`: Cria uma matriz a partir de uma sequência  
`nrow`: Determina a quantidade de linhas

# Data Science

The screenshot shows the RStudio environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, and other functions. The main workspace is divided into several panes. On the left, the Console pane shows the following R code and its output:

```
> x <- 1:12
> xmat3 <- matrix(x,ncol=3,byrow=T)
> xmat3
```

	[,1]	[,2]	[,3]
[1,]	1	2	3
[2,]	4	5	6
[3,]	7	8	9
[4,]	10	11	12

On the right, the Environment pane shows the objects created in the workspace: x, xmat3, and xmat3. The bottom right pane is empty. A blue box at the bottom left of the RStudio window contains the following text:

**Matrix**

`matrix()`: Cria uma matriz a partir de uma sequência  
ncol: Determina a quantidade de colunas  
Preenchimento por linhas

# Data Science

The screenshot shows the RStudio environment with the following components:

- Console:** Displays the execution of `xmat1` and `summary(xmat1)`. The matrix `xmat1` is a 4x3 matrix with values ranging from 1 to 12. The summary shows descriptive statistics for each column (V1, V2, V3).
- Environment:** Lists the objects in the environment: `xmat1` and `summary(xmat1)`.
- Matrix Summary:** A blue box highlights the text: **Matrix**  
`summary()` : Apresenta medidas descritivas por coluna: min, 1º quart, mediana, média, 3º quart, max

```
> xmat1
      [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> summary(xmat1)
      v1      v2      v3
Min.   :1.00  Min.   :5.00  Min.   : 9.00
1st Qu.:1.75  1st Qu.:5.75  1st Qu.: 9.75
Median :2.50  Median :6.50  Median :10.50
Mean   :2.50  Mean   :6.50  Mean   :10.50
3rd Qu.:3.25  3rd Qu.:7.25  3rd Qu.:11.25
Max.   :4.00  Max.   :8.00
```

# Data Science

The screenshot shows the RStudio environment with the following components:

- Console:** Displays the execution of `xmat1` and `summary(as.vector(xmat1))`. The matrix `xmat1` is a 4x3 matrix with values from 1 to 12. The summary statistics for the vectorized matrix are shown below.
- Environment:** Lists the objects `xmat1` and `summary(as.vector(xmat1))`.
- Matrix Summary:** A blue box highlights the output of `summary(as.vector())`, explaining that it presents descriptive measures of the entire integer set: min, 1st quart, median, mean, 3rd quart, and max.

```
> xmat1
      [,1] [,2] [,3]
[1,]     1     5     9
[2,]     2     6    10
[3,]     3     7    11
[4,]     4     8    12
> summary(as.vector(xmat1))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.00   3.75   6.50   6.50   9.25   12.00
>
```

**Matrix**

`summary(as.vector())`:

Apresenta medidas descritivas do conjunto inteiro  
min, 1º quart, mediana, média, 3º quart, max

# Data Science

The screenshot shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, and other standard operations. The main workspace is divided into several panes. On the left, the Console pane shows the following R code and its output:

```
> x <- 1:8
> xmat4 <- matrix(x,ncol=2)
> xmat4
      [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> xmat5 <- cbind(xmat4,9:12)
> xmat5
      [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> |
```

On the right, the Environment pane shows the objects created in the session: xmat1, summary(as.vector(xmat1)), x <- 1:8, xmat4 <- matrix(x,ncol=2), and xmat4. A blue overlay with white text is positioned over the Environment pane, stating:

**Matrix**  
**cbind()**  
Adiciona colunas às matrizes

At the bottom of the RStudio window, there are tabs for Files, Plots, Packages, Help, and Viewer. The Files tab is currently active, showing a list of files in the current directory.

# Data Science

The screenshot displays the RStudio environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main workspace is divided into three panes: Console, Environment, and Connections. The Console pane shows the following R code and output:

```
> xmat5
      [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> xmat6 <- rbind(xmat5,1:3)
> xmat6
      [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
[5,]    1    2    3
> |
```

The Environment pane shows the objects in the current environment:

```
xmat5
xmat6 <- rbind(xmat5,1:3)
xmat6
```

A blue overlay box in the Environment pane contains the text:

**Matrix**  
**rbind()**  
Adiciona linhas às matrizes

The bottom pane is divided into Files, Plots, Packages, Help, and Viewer tabs. The Files tab is currently active, showing a list of files in the current directory.



# Data Science

The screenshot displays the RStudio environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main workspace is divided into four panes: Console, Terminal, Jobs, and Environment. The Console pane shows the following R code and output:

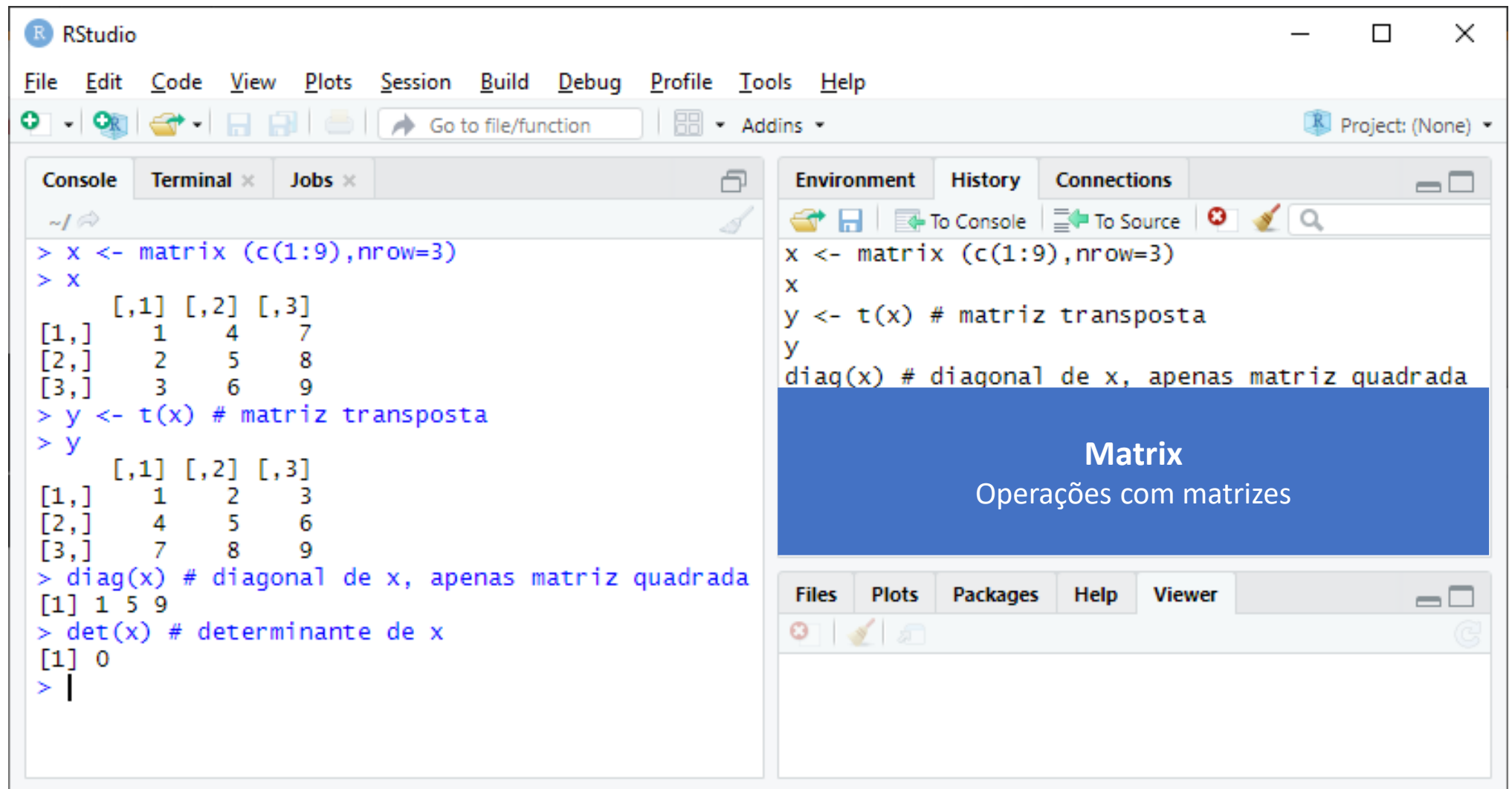
```
> xmat5
      [,1] [,2] [,3]
[1,]     1     5     9
[2,]     2     6    10
[3,]     3     7    11
[4,]     4     8    12
> xmat5[2,3] # linha 2 coluna 3
[1] 10
> xmat5[2,]  # linha 2
[1]  2  6 10
> xmat5[,2]  # coluna 2
[1]  5  6  7  8
> xmat5[c(1,3),] # linhas 1 e 3
      [,1] [,2] [,3]
[1,]     1     5     9
[2,]     3     7    11
>
```

The Environment pane on the right shows the objects in the current environment:

```
xmat5
xmat5[2,3] # linha 2 coluna 3
xmat5[2,]  # linha 2
xmat5[,2]  # coluna 2
xmat5[c(1,3),] # linhas 1 e 3
```

A blue overlay box with the text "Matrix" and "Índices de matrizes" is positioned over the Environment pane. The bottom of the RStudio window features a pane with tabs for Files, Plots, Packages, Help, and Viewer.

# Data Science



The screenshot displays the RStudio environment with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for creating a new file, opening a file, saving, and a search bar with the text "Go to file/function".
- Project:** Project: (None)
- Console:** Contains the following R code and output:

```
> x <- matrix (c(1:9),nrow=3)
> x
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> y <- t(x) # matriz transposta
> y
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> diag(x) # diagonal de x, apenas matriz quadrada
[1] 1 5 9
> det(x) # determinante de x
[1] 0
> |
```
- Environment Pane:** Shows the objects created in the environment:

```
x <- matrix (c(1:9),nrow=3)
x
y <- t(x) # matriz transposta
y
diag(x) # diagonal de x, apenas matriz quadrada
```
- Matrix Operations Panel:** A blue box with the text "Matrix" and "Operações com matrizes".
- Bottom Pane:** Includes tabs for Files, Plots, Packages, Help, and Viewer.

## Arrays

# Data Science

The screenshot shows the RStudio environment with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for creating a new file, opening a file, saving, and a search bar with the text "Go to file/function".
- Console:** Contains the following R code and its output:

```
> # cria um array com 2 dimensões
> # 3 linhas e 4 colunas
> x <- array(1:12, dim=c(3,4)) # duas dimensões
> x
```

	[,1]	[,2]	[,3]	[,4]
[1,]	1	4	7	10
[2,]	2	5	8	11
[3,]	3	6	9	12
- Environment:** Shows the same R code as the console. A blue box is overlaid on this panel with the following text:

**Array**  
Arrays podem ter qualquer número de dimensões  
**array()** : Cria um array  
**dim** : Define as dimensões do array
- Bottom Panel:** Includes tabs for Files, Plots, Packages, Help, and Viewer. The Viewer tab is currently active and empty.

# Data Science

The screenshot shows the RStudio environment with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations and a search bar with the text "Go to file/function".
- Project:** Project: (None)
- Console:** Contains the following R code and output:

```
> # cria um array com 3 dimensões
> # 2 linhas, 3 colunas e "profundidade" 2
> y <- array(1:12, dim=c(2,3,2))
> y
, , 1
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
, , 2
      [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
>
```
- Environment:** Shows the variable `y` as an array with dimensions `c(2,3,2)`.
- History:** Shows the execution history of the code.
- Connections:** Shows connections to the console and source.
- Files:** Shows a file named `.RData` with a size of 2.5 KB, modified on Nov 11.

**Array**  
Arrays podem ter qualquer número de dimensões  
**array()** : Cria um array  
**dim** : Define as dimensões do array

## Fatores

## Variáveis categóricas e contínuas

Em um conjunto de dados, podemos distinguir dois tipos de variáveis: categóricas e contínuas.

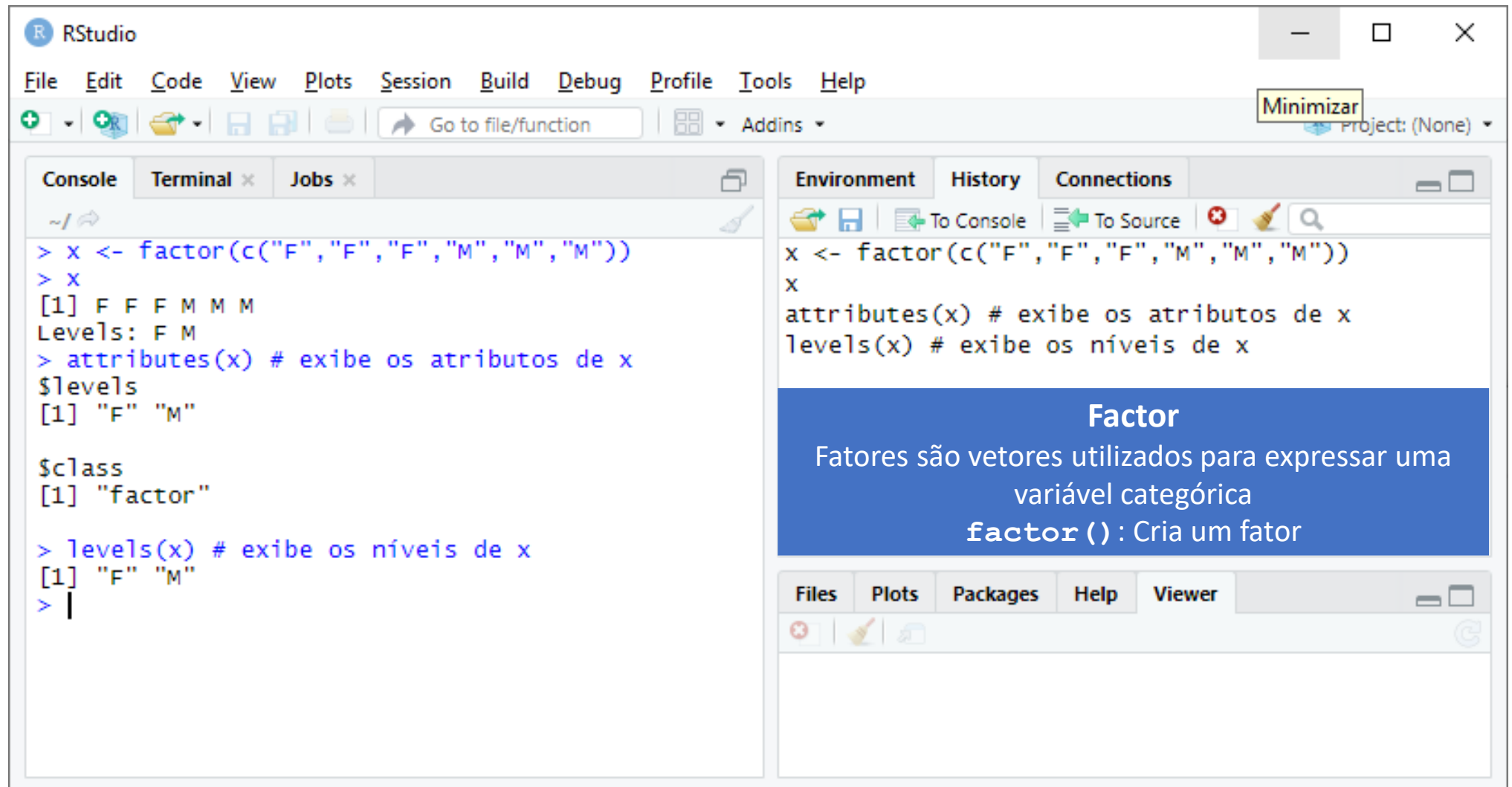
**Variáveis categóricas** apresentam um número limitado de valores diferentes geralmente baseados em um determinado grupo finito. Alguns exemplos de variáveis categóricas em R: gênero, ocupação, país, ano, etc.

**Variáveis contínuas**, por outro lado, podem assumir qualquer valor, de inteiro a decimal. Por exemplo, podemos ter a receita de uma empresa, o preço de uma ação, etc.

Variáveis categóricas no R são armazenadas em um fator.

Os fatores armazenam dados de string e inteiros como "levels" ou níveis.

# Data Science



The screenshot shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main workspace is divided into four panes: Console, Terminal, Jobs, and Environment. The Console pane shows the following R code and output:

```
> x <- factor(c("F","F","F","M","M","M"))
> x
[1] F F F M M M
Levels: F M
> attributes(x) # exibe os atributos de x
$levels
[1] "F" "M"

$class
[1] "factor"

> levels(x) # exibe os níveis de x
[1] "F" "M"
> |
```

The Environment pane on the right shows the variable `x` and its attributes. A blue box highlights the definition of a factor:

**Factor**  
Fatores são vetores utilizados para expressar uma variável categórica  
`factor()` : Cria um fator

The bottom of the RStudio window shows tabs for Files, Plots, Packages, Help, and Viewer.



## Data Frames

# Data Science

The screenshot shows the RStudio environment with the following components:

- Console:** Displays the R commands and their output for creating a data frame.
- Environment:** Lists the objects created in the environment.
- Environment Panel:** Contains a blue box with text explaining data frames.
- Files Panel:** Shows the file explorer.
- Plots Panel:** Shows the plot area.
- Packages Panel:** Shows the installed packages.
- Help Panel:** Shows the help documentation.
- Viewer Panel:** Shows the viewer area.

**Console Output:**

```
> nome<-c("John Smith","Mary Clark","Noah Baker")
> nome
[1] "John Smith" "Mary Clark" "Noah Baker"
> idade<-c(25,21,23)
> idade
[1] 25 21 23
> sexo<-factor(c("M","F","M"))
> sexo
[1] M F M
Levels: F M
> empresa<-data.frame(nome,idade,sexo)
> empresa
      nome idade sexo
1 John Smith   25    M
2 Mary Clark   21    F
3 Noah Baker   23    M
> |
```

**Environment Panel:**

```
nome<-c("John Smith","Mary Clark","Noah Baker")
nome
idade<-c(25,21,23)
idade
sexo<-factor(c("M","F","M"))
```

**Environment Panel Text:**

**Data frame**

"Matrizes" que podem armazenar elementos de tipos diferentes

**data.frame()** : Cria um data.frame

# Data Science

The screenshot shows the RStudio environment with the following components:

- Console:** Displays the execution of R code to create and modify a data frame named 'empresa'.
- Environment:** Shows the objects created in the workspace, including 'idade', 'sexo', and 'empresa'.
- Code Editor:** Contains the R script being executed.
- Files, Plots, Packages, Help, Viewer:** Other standard RStudio panels.

**Console Output:**

```
> empresa
      nome idade sexo
1 John Smith   25   M
2 Mary Clark   21   F
3 Noah Baker   23   M
> rownames(empresa)<-c("emp1","emp2","emp3")
> empresa
      nome idade sexo
emp1 John Smith   25   M
emp2 Mary Clark   21   F
emp3 Noah Baker   23   M
> colnames(empresa)<-c("nome_emp","idade_em
p","sexo_emp")
> empresa
      nome_emp idade_emp sexo_emp
emp1 John Smith   25         M
emp2 Mary Clark   21         F
emp3 Noah Baker   23         M
>
```

**Environment:**

```
idade<-c(25,21,23)
idade
sexo<-factor(c("M","F","M"))
sexo
empresa<-data.frame(nome,idade,sexo)
```

**Data frame**

- `rownames()` : Lista ou altera os nomes das linhas
- `colnames()` : Lista ou altera os nomes das colunas

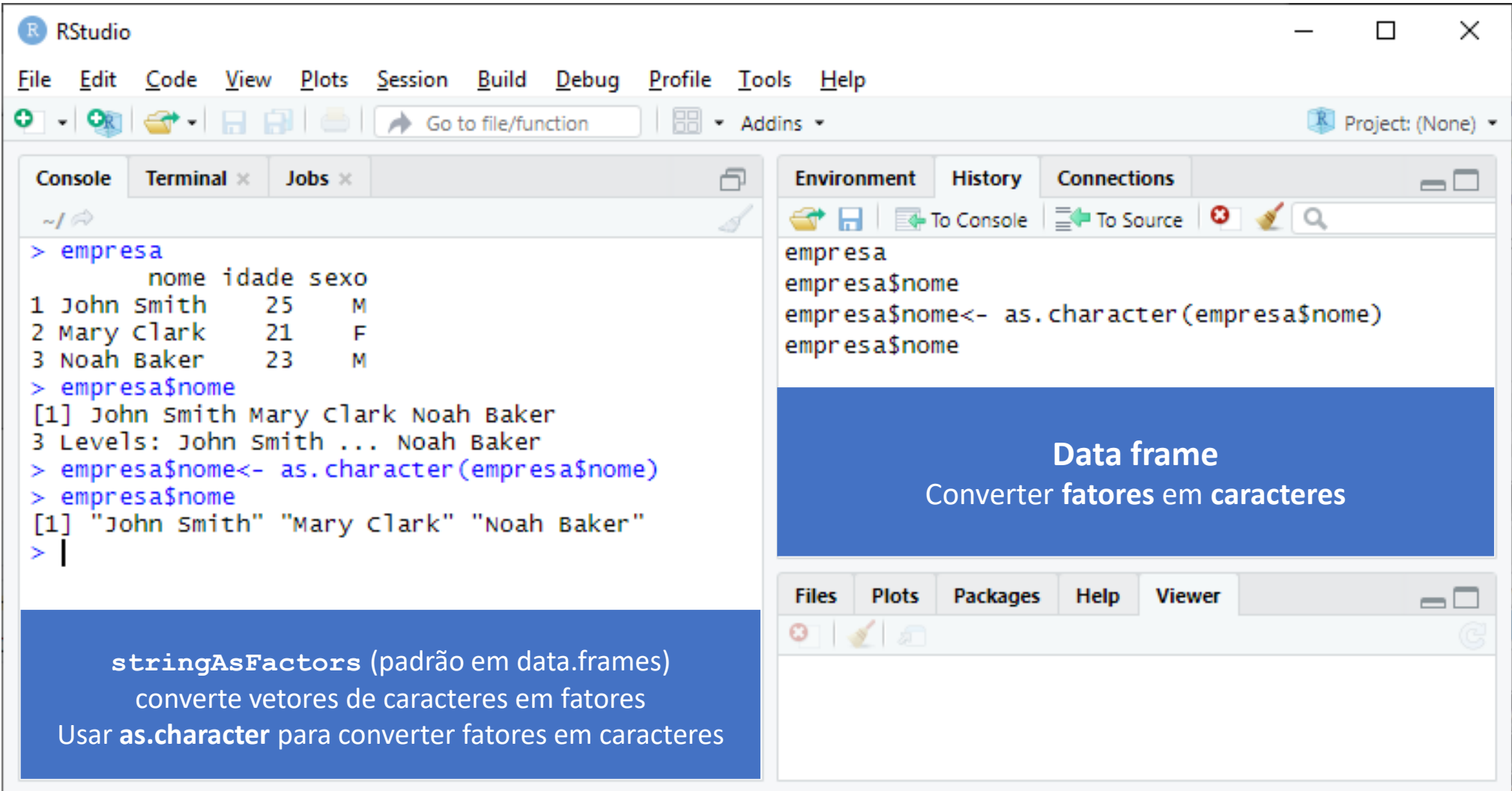
# Data Science

The screenshot shows the RStudio interface with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for creating a new file, opening a file, saving, and a search bar with the text "Go to file/function".
- Project:** Project: (None)
- Console:** Contains the following R code and output:

```
> empresa
      nome idade sexo
1 John Smith   25    M
2 Mary Clark   21    F
3 Noah Baker   23    M
> empresa[2,1]
[1] Mary Clark
3 Levels: John Smith ... Noah Baker
> empresa[2,]
      nome idade sexo
2 Mary Clark   21    F
> empresa$nome[2]
[1] Mary Clark
3 Levels: John Smith ... Noah Baker
> empresa$nome
[1] John Smith Mary Clark Noah Baker
3 Levels: John Smith ... Noah Baker
>
```
- Environment Pane:** Lists the objects in the environment:
  - empresa
  - empresa[2,1]
  - empresa[2,]
  - empresa\$nome[2]
  - empresa\$nome
- Blue Overlay:** A blue box with the text "Data frame" and "Índices dos data.frames".
- Bottom Pane:** Includes tabs for Files, Plots, Packages, Help, and Viewer.

# Data Science



The screenshot shows the RStudio environment with the following components:

- Console:** Displays the execution of R code to convert a factor variable 'nome' in a data frame 'empresa' to a character vector.
- Environment:** Shows the objects in the current environment, including 'empresa' and 'empresa\$nome'.
- Code Editor:** Contains the R script being executed.
- Annotation Boxes:** Two blue boxes provide additional context: one at the bottom left explains the difference between `stringAsFactors` and `as.character`, and another in the Environment pane highlights the conversion of factors to characters.

**Console Output:**

```
> empresa
      nome idade sexo
1 John Smith   25    M
2 Mary Clark   21    F
3 Noah Baker   23    M
> empresa$nome
[1] John Smith Mary Clark Noah Baker
3 Levels: John Smith ... Noah Baker
> empresa$nome<- as.character(empresa$nome)
> empresa$nome
[1] "John Smith" "Mary Clark" "Noah Baker"
> |
```

**Environment:**

```
empresa
empresa$nome
empresa$nome<- as.character(empresa$nome)
empresa$nome
```

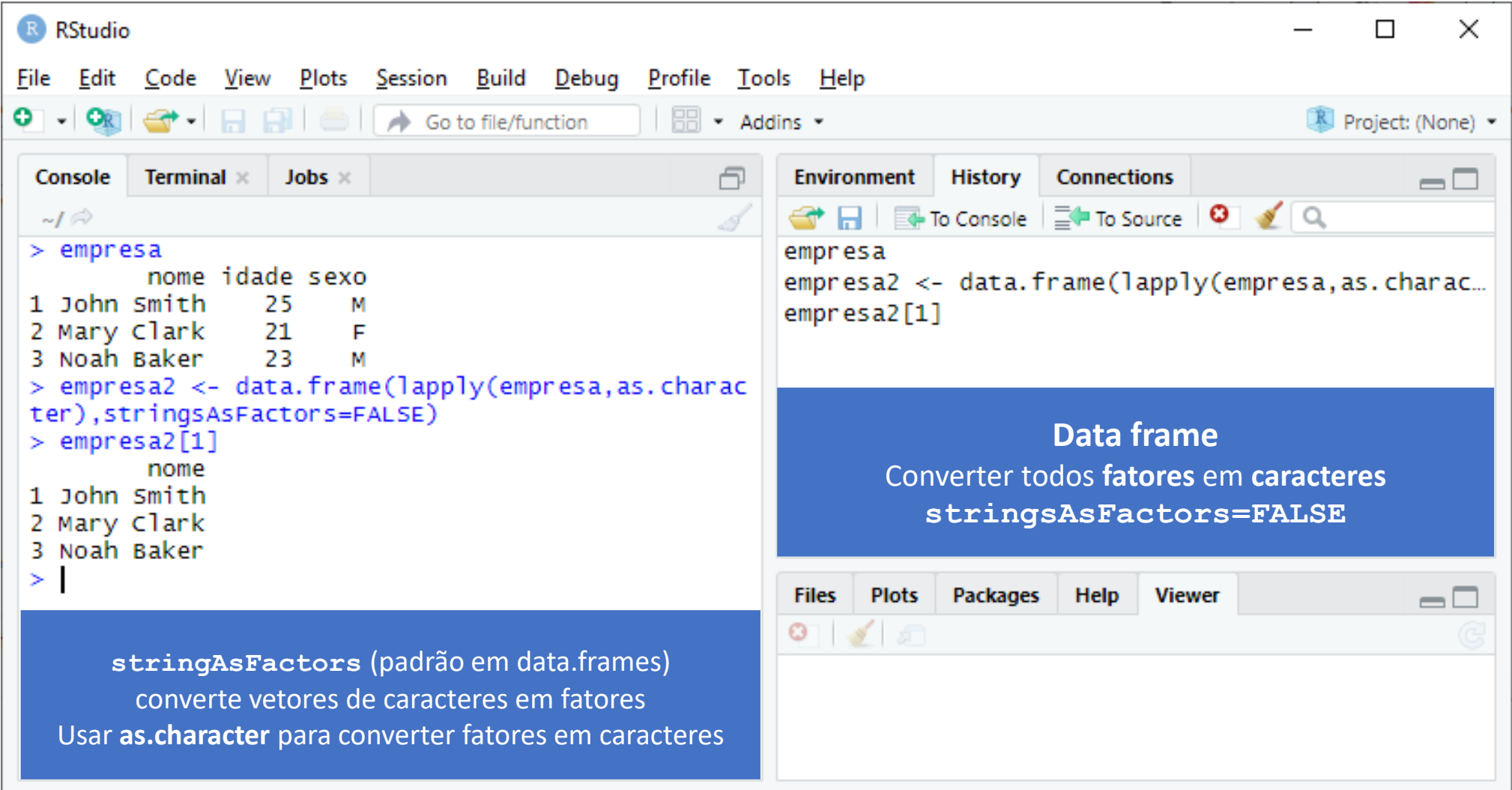
**Annotation Box (Bottom Left):**

`stringAsFactors` (padrão em data.frames)  
converte vetores de caracteres em fatores  
Usar `as.character` para converter fatores em caracteres

**Annotation Box (Environment Pane):**

**Data frame**  
Converter fatores em caracteres

# Data Science



The screenshot shows the RStudio environment with the following components:

- Console:** Displays the execution of R code to create a data frame from a list and convert string factors to characters.
- Environment:** Shows the objects in the workspace, including the newly created data frame.
- Annotations:** Two blue boxes provide additional context on the `stringsAsFactors` parameter.

**Console Output:**

```
> empresa
      nome idade sexo
1 John Smith    25   M
2 Mary Clark    21   F
3 Noah Baker    23   M
> empresa2 <- data.frame(lapply(empresa, as.character), stringsAsFactors=FALSE)
> empresa2[1]
      nome
1 John Smith
2 Mary Clark
3 Noah Baker
> |
```

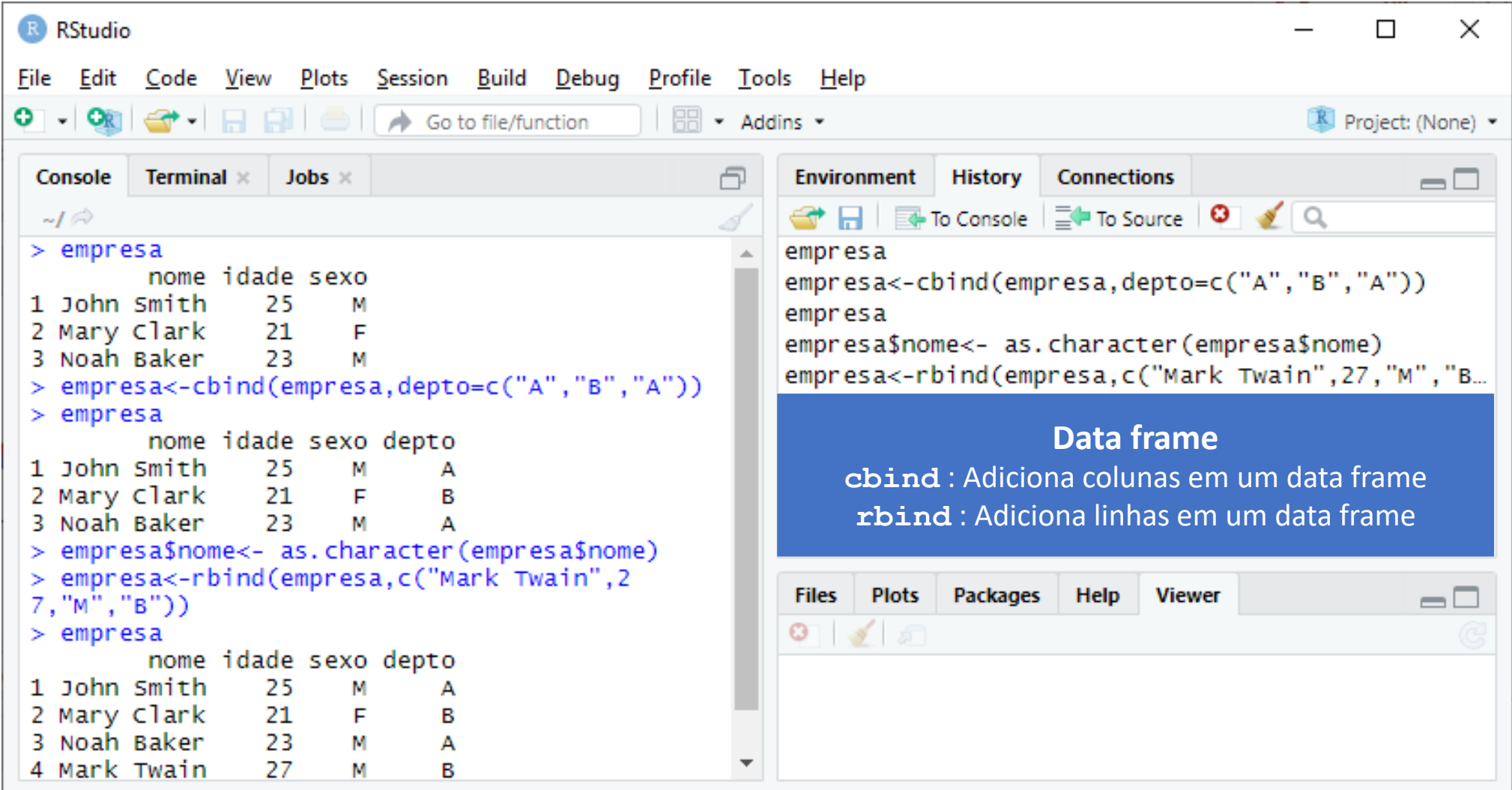
**Environment:**

```
empresa
empresa2 <- data.frame(lapply(empresa, as.character), stringsAsFactors=FALSE)
empresa2[1]
```

**Annotations:**

- Data frame**  
Converter todos fatores em caracteres  
`stringsAsFactors=FALSE`
- `stringsAsFactors` (padrão em data.frames)  
converte vetores de caracteres em fatores  
Usar `as.character` para converter fatores em caracteres

# Data Science



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

Console Terminal Jobs

```
> empresa
      nome idade sexo
1 John Smith   25    M
2 Mary Clark   21    F
3 Noah Baker   23    M
> empresa<-cbind(empresa,depto=c("A","B","A"))
> empresa
      nome idade sexo depto
1 John Smith   25    M     A
2 Mary Clark   21    F     B
3 Noah Baker   23    M     A
> empresa$nome<- as.character(empresa$nome)
> empresa<-rbind(empresa,c("Mark Twain",27,"M","B...
7,"M","B"))
> empresa
      nome idade sexo depto
1 John Smith   25    M     A
2 Mary Clark   21    F     B
3 Noah Baker   23    M     A
4 Mark Twain   27    M     B
```

Environment History Connections

empresa

```
empresa<-cbind(empresa,depto=c("A","B","A"))
empresa
empresa$nome<- as.character(empresa$nome)
empresa<-rbind(empresa,c("Mark Twain",27,"M","B...)
```

**Data frame**

**cbind** : Adiciona colunas em um data frame

**rbind** : Adiciona linhas em um data frame

Files Plots Packages Help Viewer

# Data Science

The screenshot shows the RStudio environment with the following components:

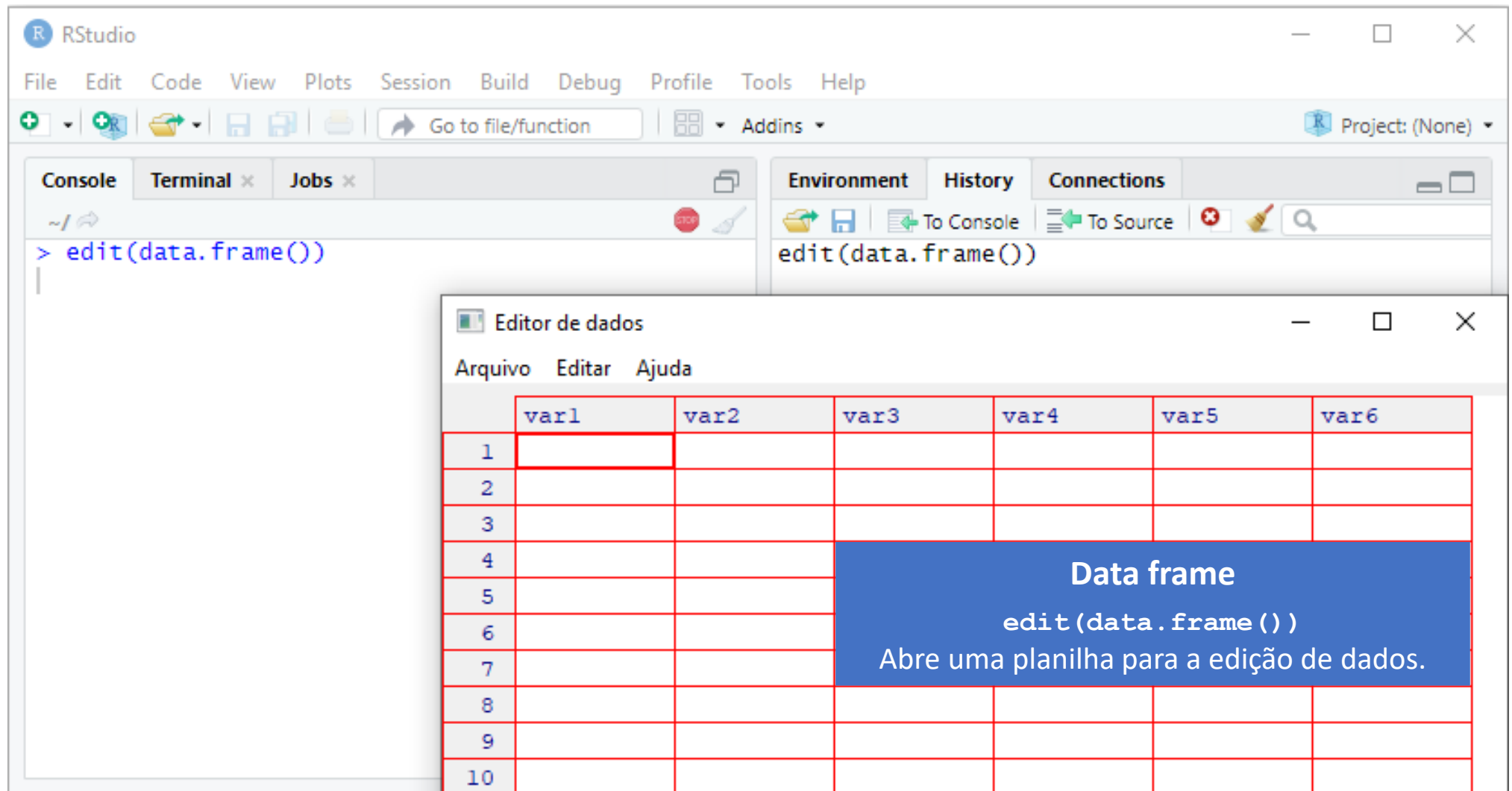
- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for creating a new file, opening a file, saving, and a search bar labeled "Go to file/function".
- Console:** Contains the following R commands and their output:

```
> empresa
      nome idade sexo
1 John Smith   25    M
2 Mary Clark   21    F
3 Noah Baker   23    M
> empresa[empresa$sexo=="F",]
      nome idade sexo
2 Mary Clark   21    F
> empresa[order(empresa$idade),]
      nome idade sexo
2 Mary Clark   21    F
3 Noah Baker   23    M
1 John Smith   25    M
> empresa[rev(order(empresa$idade)),]
      nome idade sexo
1 John Smith   25    M
3 Noah Baker   23    M
2 Mary Clark   21    F
>
```
- Environment Pane:** Shows the variable `empresa` and the results of the last three commands:

```
empresa
empresa[empresa$sexo=="F",]
empresa[order(empresa$idade),]
empresa[rev(order(empresa$idade)),]
```
- Environment Pane Overlay:** A blue box with the text "Data frame" and "Realizando consultas".
- Bottom Pane:** Includes tabs for Files, Plots, Packages, Help, and Viewer.



# Data Science



The image shows the RStudio interface. In the console, the command `> edit(data.frame())` has been entered. This command opens the 'Editor de dados' (Data Editor) window, which is a spreadsheet-like interface for editing data frames. The window has a menu bar with 'Arquivo', 'Editar', and 'Ajuda'. Below the menu is a grid with 10 rows and 6 columns. The columns are labeled 'var1', 'var2', 'var3', 'var4', 'var5', and 'var6'. The rows are numbered 1 through 10. A blue box is overlaid on the right side of the grid, containing the text 'Data frame' and 'edit(data.frame()) Abre uma planilha para a edição de dados.'

	var1	var2	var3	var4	var5	var6
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

**Data frame**  
`edit(data.frame())`  
Abre uma planilha para a edição de dados.

## Listas

# Data Science

The screenshot displays the RStudio environment with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations and a search bar labeled "Go to file/function".
- Console:** Shows the execution of the following R code:

```
> aluno<-list(nome="John Smith",notas=c(8,7,9))
> aluno
$nome
[1] "John Smith"

$notas
[1] 8 7 9

> aluno$nome
[1] "John Smith"
> aluno[1]
$nome
[1] "John Smith"

> |
```
- Environment:** Displays the objects in the current environment:

```
aluno<-list(nome="John Smith",notas=c(8,7,9))
aluno
aluno$nome
aluno[1]
```
- Information Panel:** A blue box titled "List" containing the text:

**list()** : Cria uma lista  
Listas combinam diferentes estruturas de dados em um mesmo objeto
- Bottom Panel:** Includes tabs for Files, Plots, Packages, Help, and Viewer.

## Entrada de Dados

## Entrada de Dados

Método	Descrição
<code>scan()</code>	Entrada através do console ou de arquivo texto (.txt)
<code>read.table()</code>	Entrada através de arquivo texto com separadores (.csv)
<code>read.xlsx()</code>	Entrada através de arquivo MS Excel (.xlsx)
<code>data()</code>	Entrada através de datasets

# Data Science

The screenshot displays the RStudio environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main workspace is divided into several panes:

- Console:** Shows the execution of the `scan()` function. The first call reads 5 items (10, 15, 20, 25, 30) into variable `x`. The second call reads 4 items (2, 4, 6, 8) into variable `y`.
- Environment:** Lists the objects in the environment: `x` and `y`.
- Text Editor:** A small window titled "Arquivo" showing a list of numbers: 2, 4, 6, 8.

A blue overlay box in the Environment pane provides information about the `scan` function:

**scan**

**scan ()** : Entrada de dados

Os dados podem ser inseridos através do console ou copiados de outro arquivo e colados no console

# Data Science

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ + + + + Go to file/function Addins Project: (None)

Console Terminal x Jobs x

```
> empresa<-read.table("C:/dados/empresa.txt",
+ header=T,
+ sep=",")
> empresa
```

	nome	idade	sexo
1	John Smith	25	M
2	Mary Clark	21	F
3	Noah Baker	23	M

> |

emp... — □ ×

Arquivo Editar Formatar

Exibir Ajuda

nome,idade,sexo  
John Smith,25,M  
Mary Clark,21,F  
Noah Baker,23,M

Wir Ln 4 10

**read.table**

`read.table()` : Lê os dados de um arquivo .txt  
`header=T` : O arquivo-fonte contém cabeçalho  
`sep=" , "` : Campos são separados por , (vírgula)

Files Plots Packages Help Viewer

R usa / (barra normal) e não \ (barra invertida)  
inclusive na versão Windows

# Data Science

The screenshot shows the RStudio environment with the following components:

- Console:** Displays the R commands and the resulting data frame.
- Environment:** Shows the loaded data frame 'empresa'.
- Files:** Shows the active spreadsheet 'Planilha1'.

**Console Output:**

```
> install.packages("xlsx") #instala pacote xlsx
> library(xlsx)           #carrega pacote xlsx
> empresa<-read.xlsx(     #lê arquivo .xlsx
+ file="C:/dados/empresa.xlsx", #local do arquivo
+ sheetName="Planilha1",      #nome da planilha
+ header=TRUE)               #cabeçalho
> empresa
```

	nome	idade	sexo
1	John Smith	25	M
2	Mary Clark	21	F
3	Noah Baker	23	M

**Environment:**

```
empresa<-read.xlsx( #lê arquivo .xlsx
file="C:/dados/empresa.xlsx", #local do arquivo
sheetName="Planilha1", #nome da planilha
header=TRUE) #cabeçalho
empresa
```

**Files:**

	A	B	C
1	nome	idade	sexo
2	John Smith	25	M
3	Mary Clark	21	F
4	Noah Baker	23	M
5			

Planilha1

**Entrada de dados do Excel (xlsx)**  
`read.xlsx()` : Lê conteúdo de um arquivo .xlsx



# Data Science

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main workspace is divided into four panes: Console, Terminal, Jobs, and Environment. The Console pane shows the following R code and output:

```
> data("women") #carrega dataset women
> women        #exibe dataset women
  height weight
1     58    115
2     59    117
3     60    120
4     61    123
5     62    126
6     63    129
7     64    132
8     65    135
9     66    139
10    67    142
11    68    146
12    69    150
13    70    154
14    71    159
15    72    164
>
```

The Environment pane on the right shows the loaded objects: `data("women")` and `women`. Below the Environment pane, a blue box contains the text:

**datasets**  
`data ()` : Carrega o conteúdo de um dataset

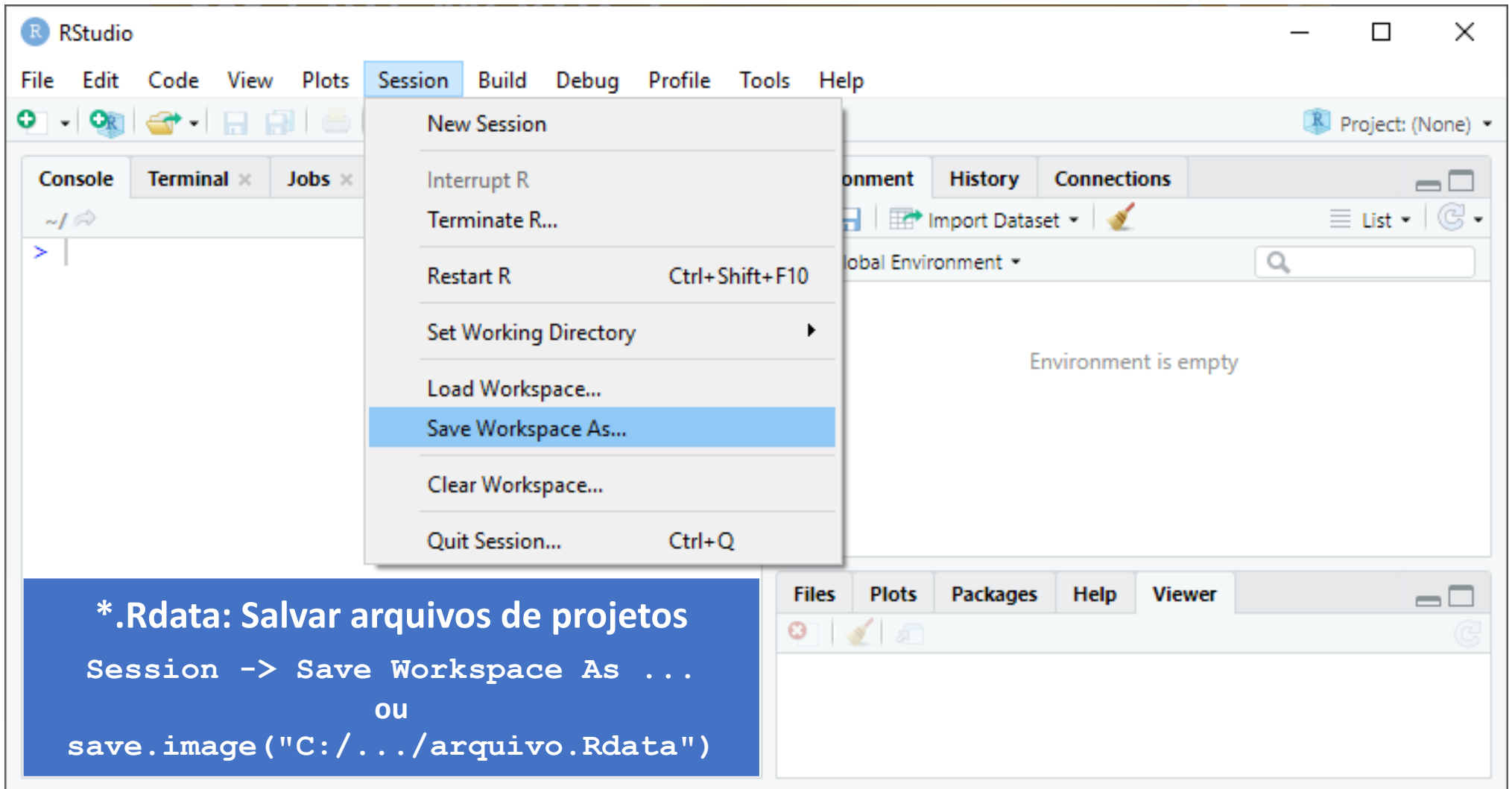
At the bottom of the RStudio window, there is a toolbar with icons for Files, Plots, Packages, Help, and Viewer.

## Tipos de arquivos do R

## Tipos de arquivos do R

Tipo de Arquivo	Descrição
* .RData	Arquivos de projetos. Salvam os objetos criados em uma sessão.
* .Rhistory	Contêm o histórico de todos os comandos digitados em uma sessão.
* .R	Códigos criados pelos usuários (funções, rotinas de análise, etc.)
* .Rmd	Arquivos Markdown usados para criação de conteúdo

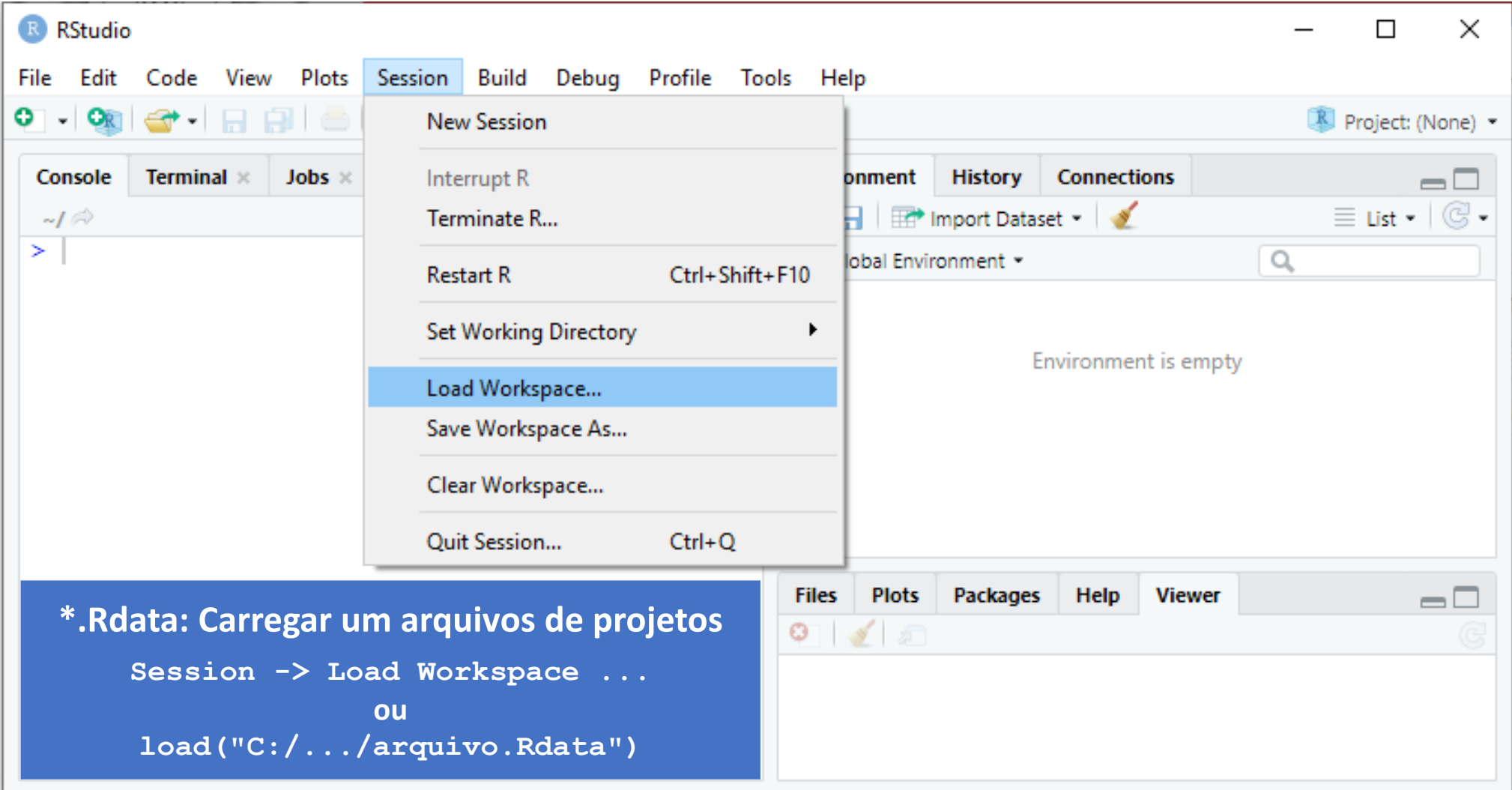
# Data Science



The screenshot shows the RStudio application window. The 'Session' menu is open, displaying options: New Session, Interrupt R, Terminate R..., Restart R (with shortcut Ctrl+Shift+F10), Set Working Directory, Load Workspace..., Save Workspace As... (highlighted in blue), Clear Workspace..., and Quit Session... (with shortcut Ctrl+Q). The background shows the RStudio interface with a console, terminal, and jobs pane on the left, and environment, history, and connections panes on the right. The environment pane shows 'Global Environment' and 'Environment is empty'.

**\*.Rdata: Salvar arquivos de projetos**  
Session -> Save Workspace As ...  
ou  
`save.image("C:/.../arquivo.Rdata")`

# Data Science

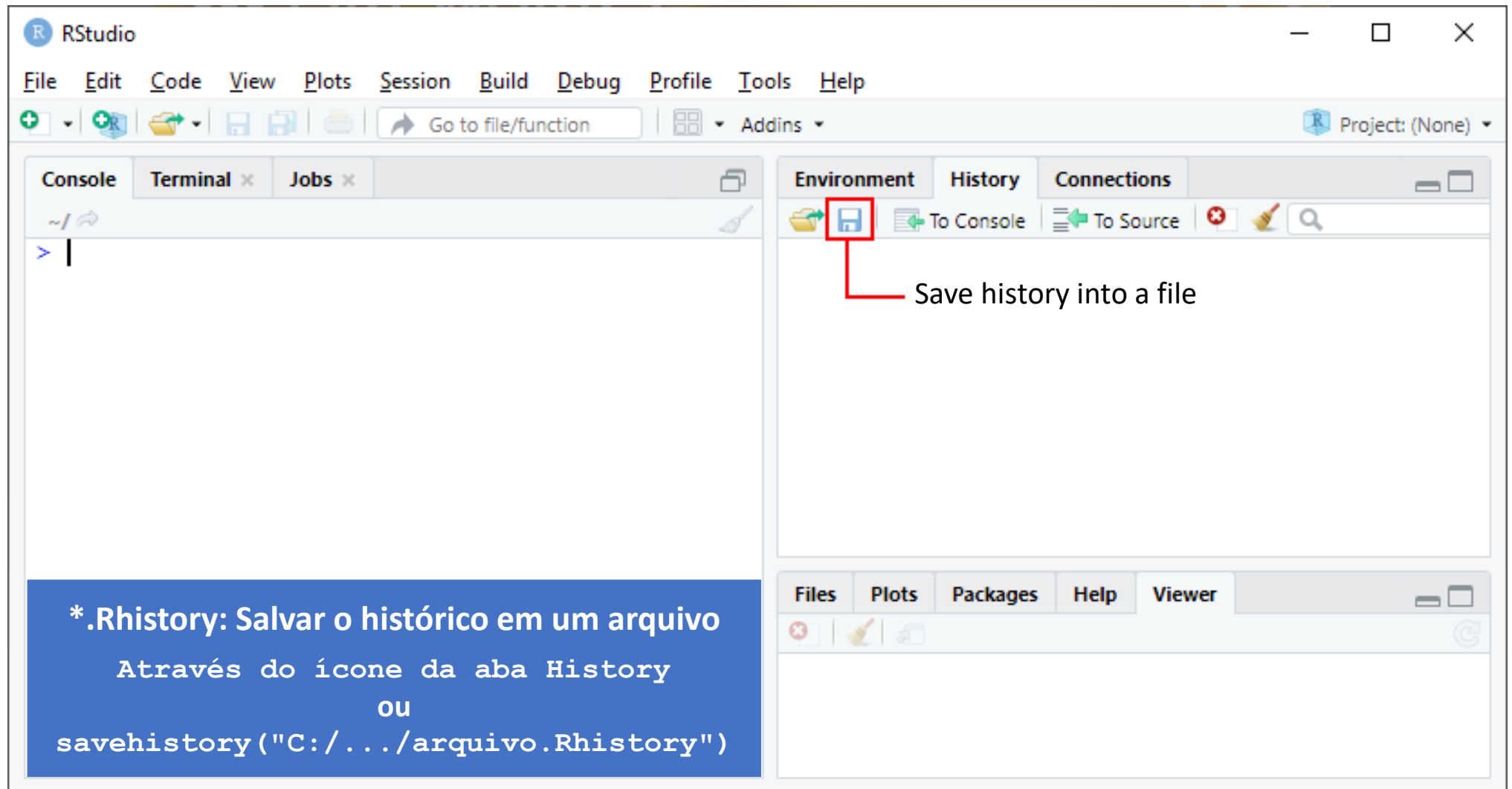


The screenshot shows the RStudio application window. The 'Session' menu is open, displaying options: New Session, Interrupt R, Terminate R..., Restart R (with shortcut Ctrl+Shift+F10), Set Working Directory, Load Workspace... (highlighted), Save Workspace As..., Clear Workspace..., and Quit Session... (with shortcut Ctrl+Q). The background shows the RStudio interface with the Console, Terminal, and Jobs panes on the left, and the Environment, History, and Connections panes on the right. The Environment pane shows 'Global Environment' and 'Environment is empty'.

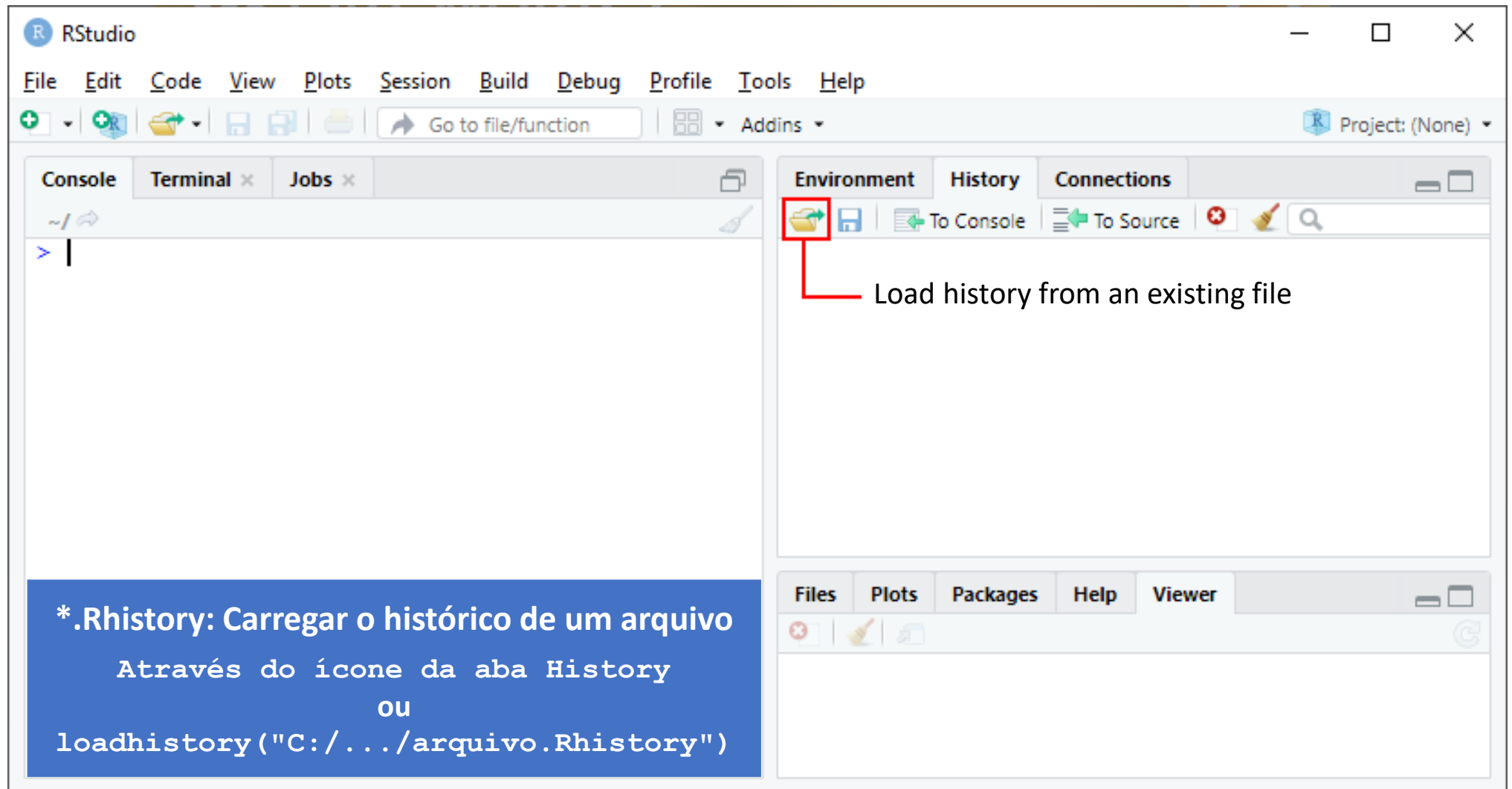
**\*.Rdata: Carregar um arquivos de projetos**

```
Session -> Load Workspace ...  
ou  
load("C:/.../arquivo.Rdata")
```

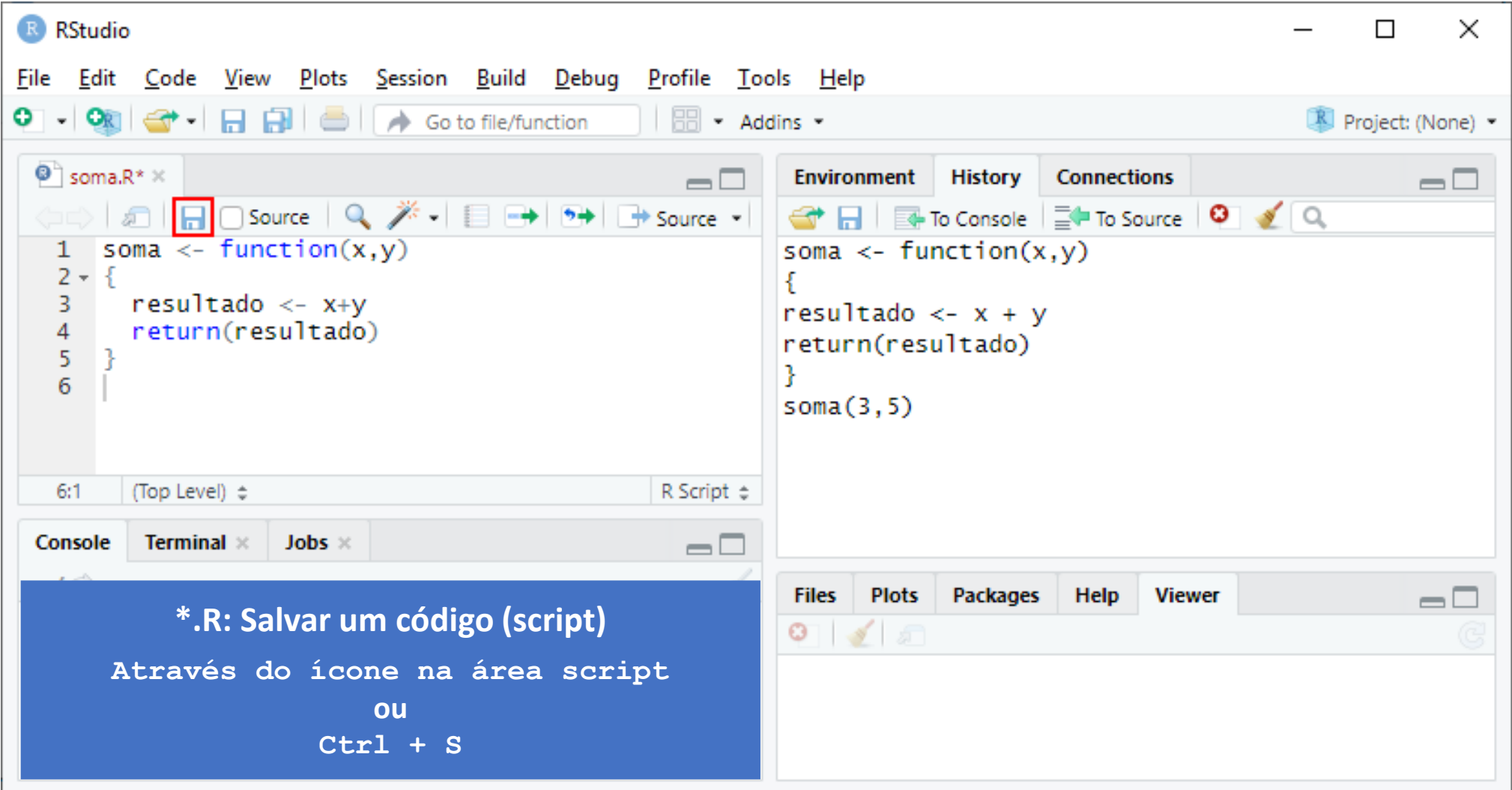
# Data Science



# Data Science



# Data Science



The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, and other functions. The main editor window shows a script named 'soma.R' with the following code:

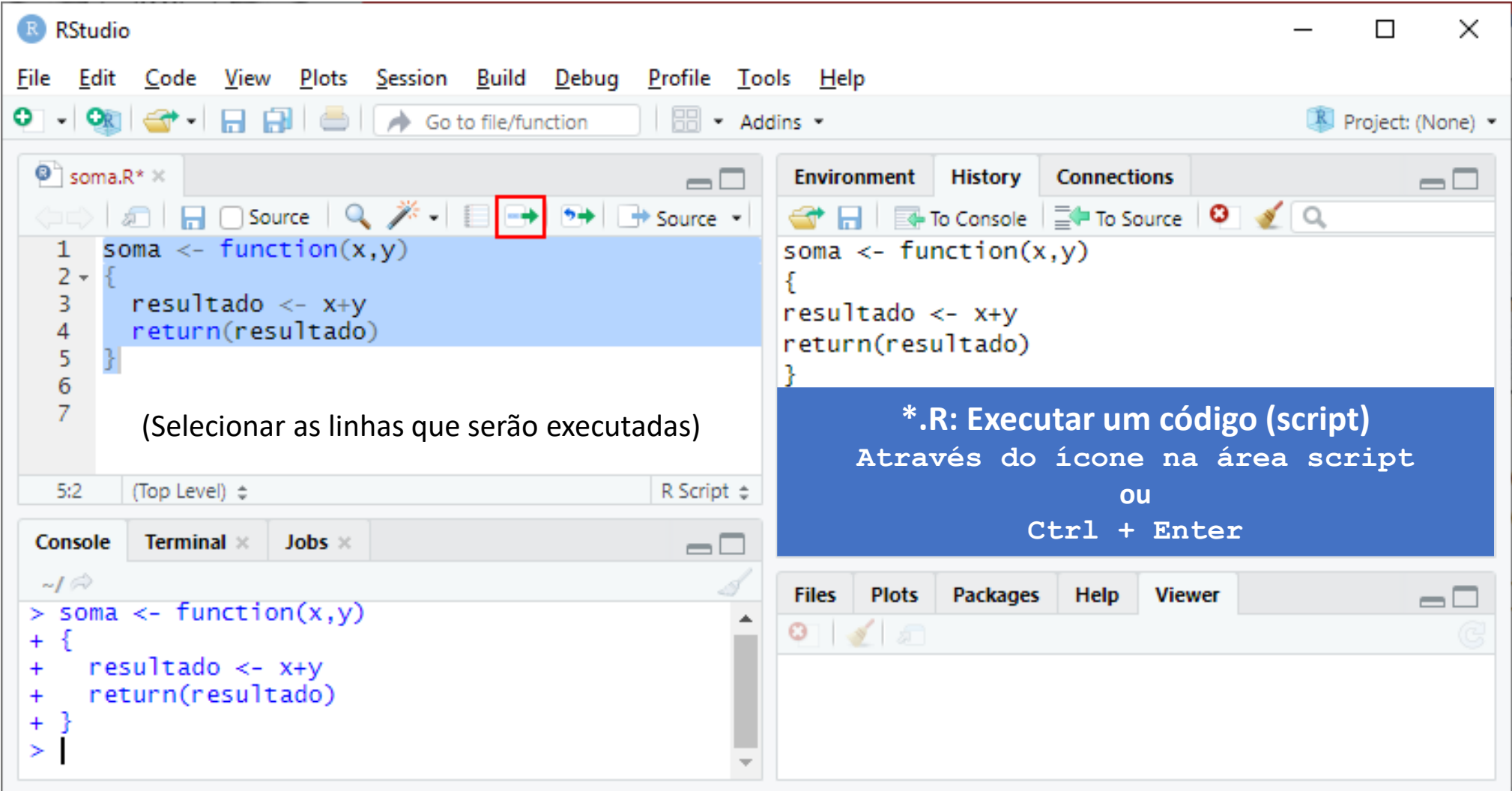
```
1 soma <- function(x,y)
2 {
3   resultado <- x+y
4   return(resultado)
5 }
6
```

The 'Save' icon (a floppy disk) in the toolbar is highlighted with a red square. To the right of the editor is the Environment pane, which shows the function 'soma' and its execution. At the bottom of the window is a console area. A blue overlay box is positioned over the console, containing the following text:

**\*.R: Salvar um código (script)**  
Através do ícone na área script  
ou  
Ctrl + S



# Data Science



The screenshot displays the RStudio environment with a script named `soma.R` open. The script defines a function `soma` that takes two arguments, `x` and `y`, and returns their sum. The function code is highlighted in blue. A red box highlights the 'Run' button (a green arrow) in the script editor's toolbar. Below the script, the console shows the function being executed, with the prompt `>` and the function code repeated. On the right side, the 'Environment' pane shows the function `soma` defined. A blue overlay box contains the text:   
\*.R: Executar um código (script)  
Através do ícone na área script  
ou  
Ctrl + Enter

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ + + + + Go to file/function Addins Project: (None)

soma.R x

Source

```
1 soma <- function(x,y)
2 {
3   resultado <- x+y
4   return(resultado)
5 }
6
7
```

(Selecionar as linhas que serão executadas)

5:2 (Top Level) R Script

Console Terminal x Jobs x

```
> soma <- function(x,y)
+ {
+   resultado <- x+y
+   return(resultado)
+ }
+ 
```

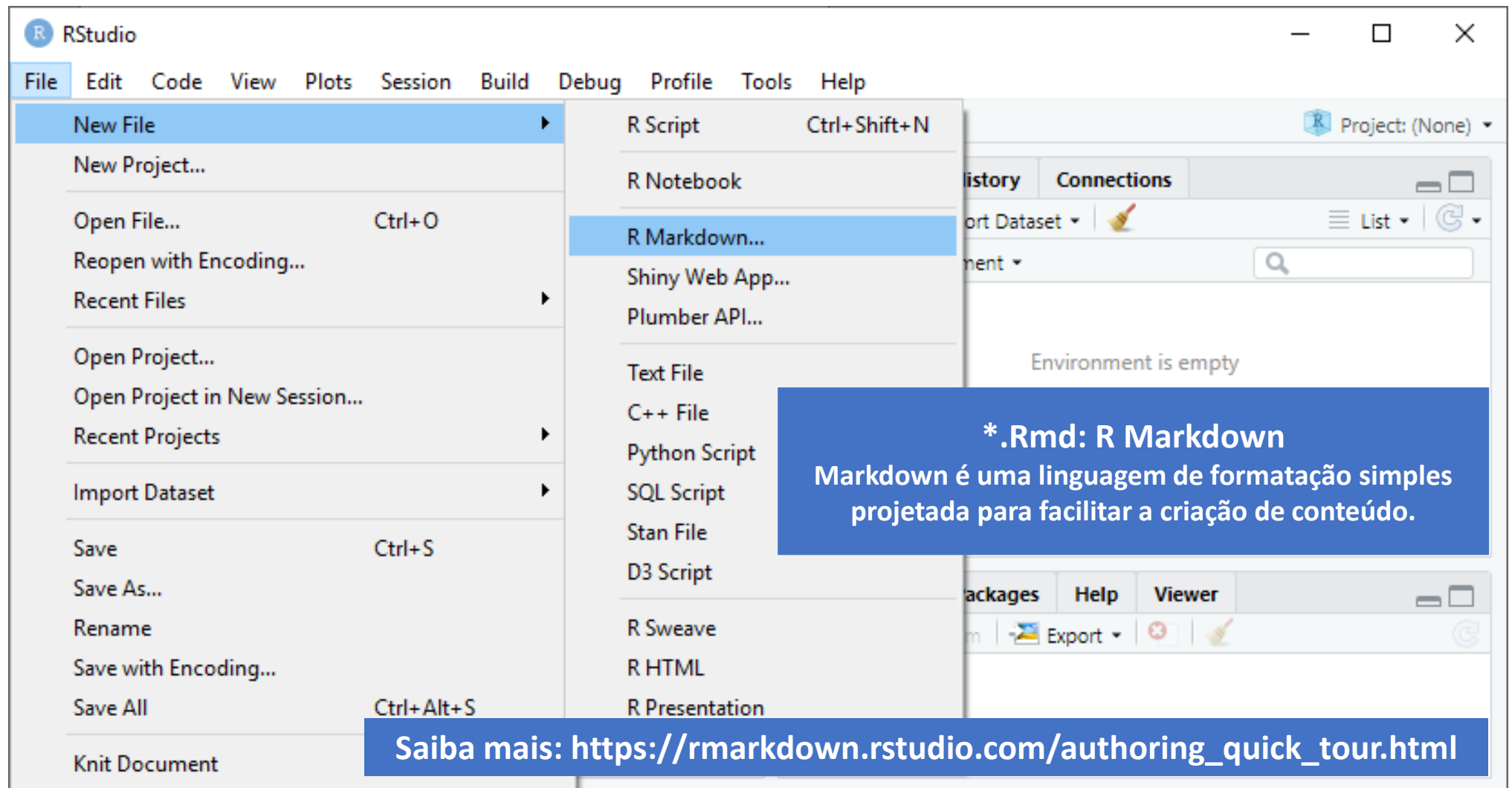
Environment History Connections

To Console To Source

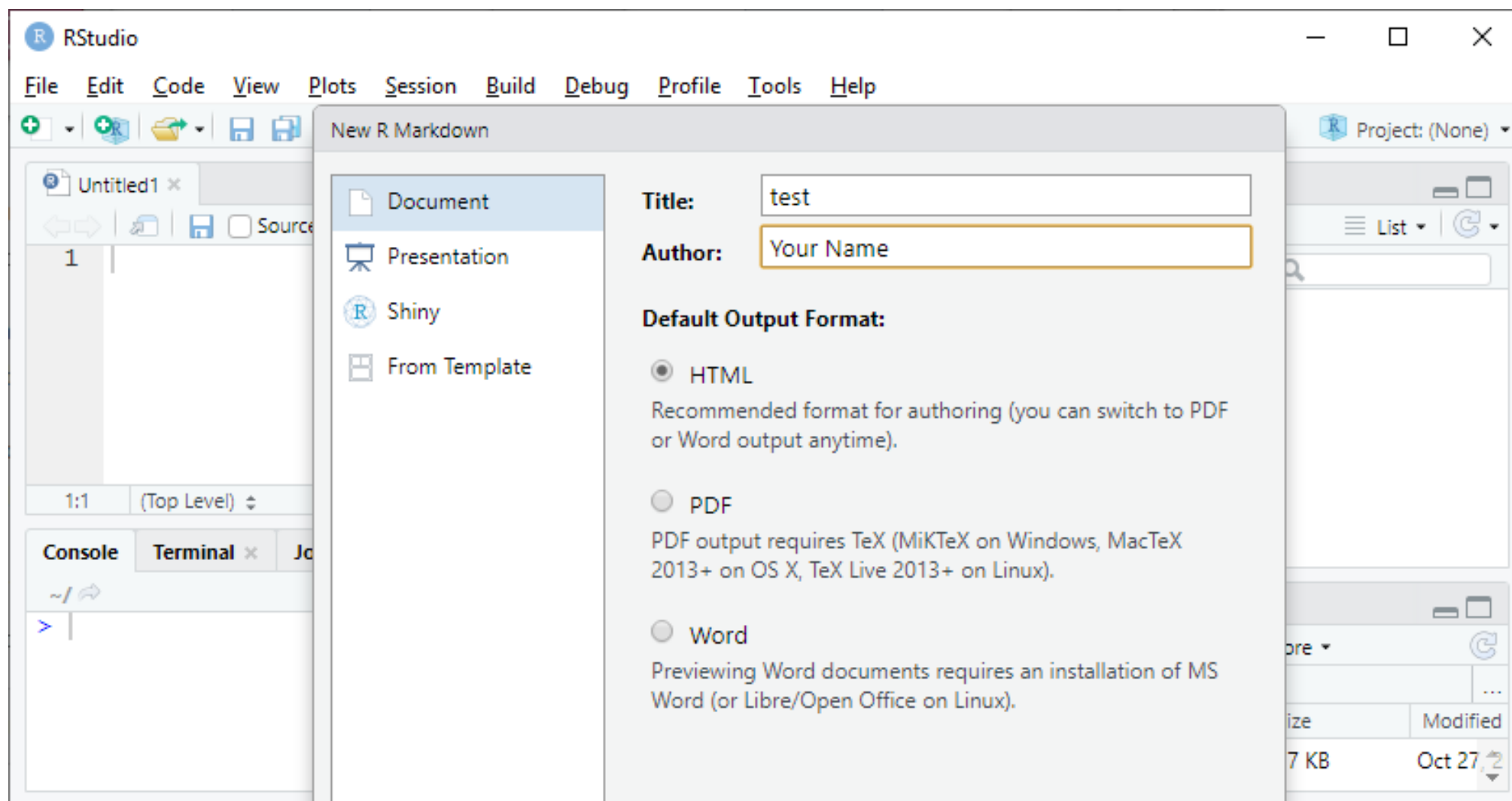
```
soma <- function(x,y)
{
  resultado <- x+y
  return(resultado)
}
```

Files Plots Packages Help Viewer

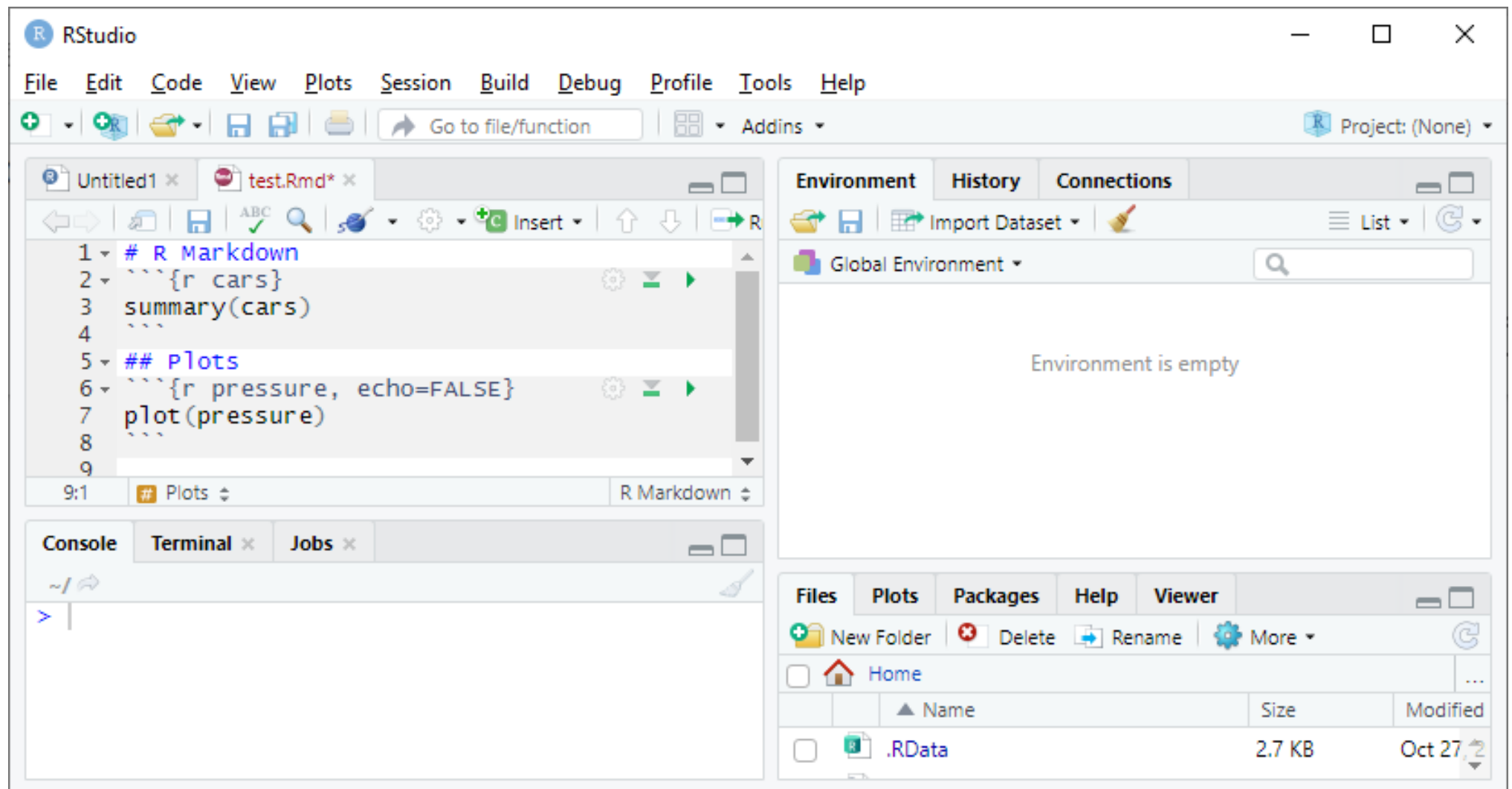
# Data Science



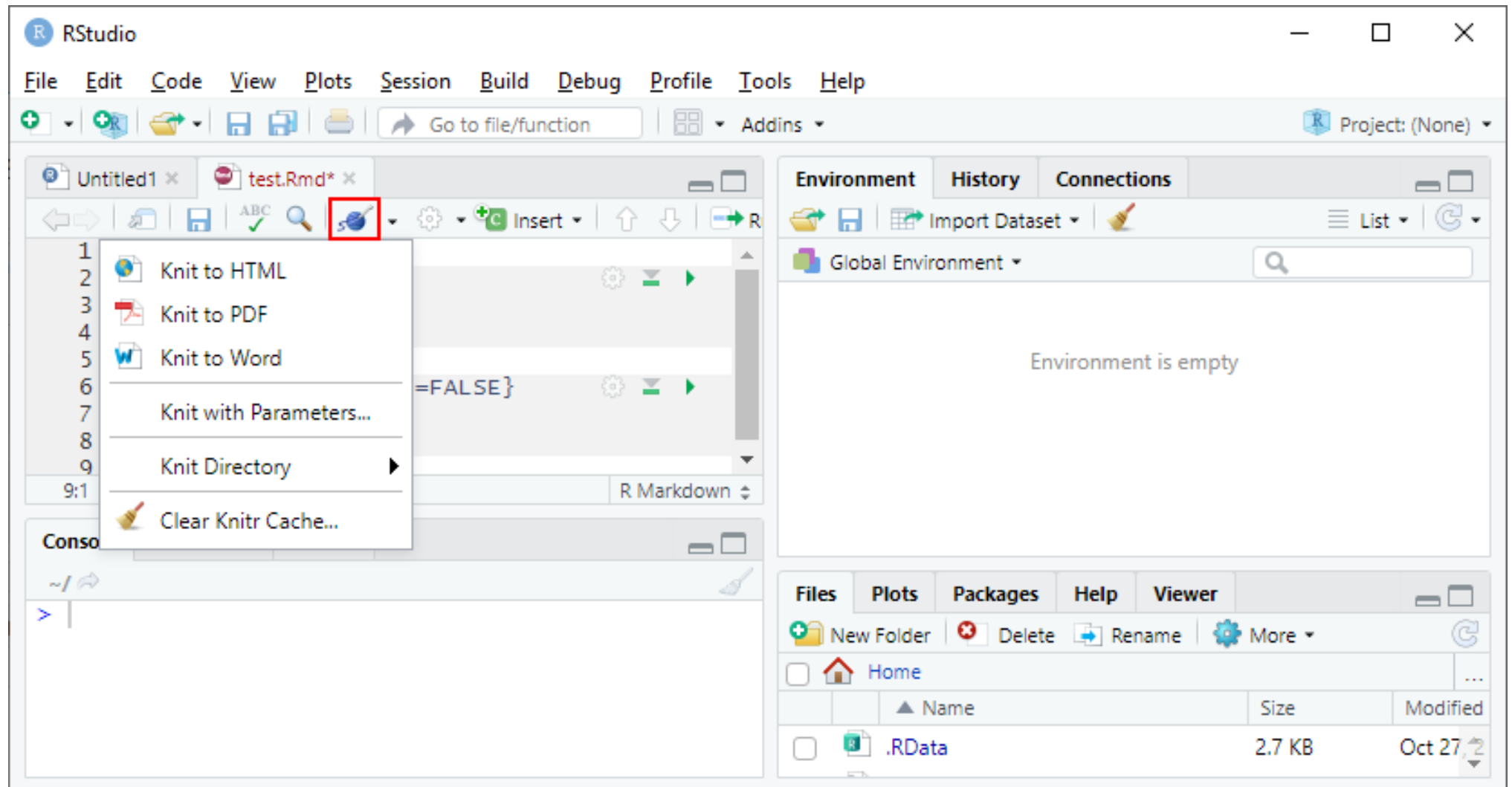
# Data Science



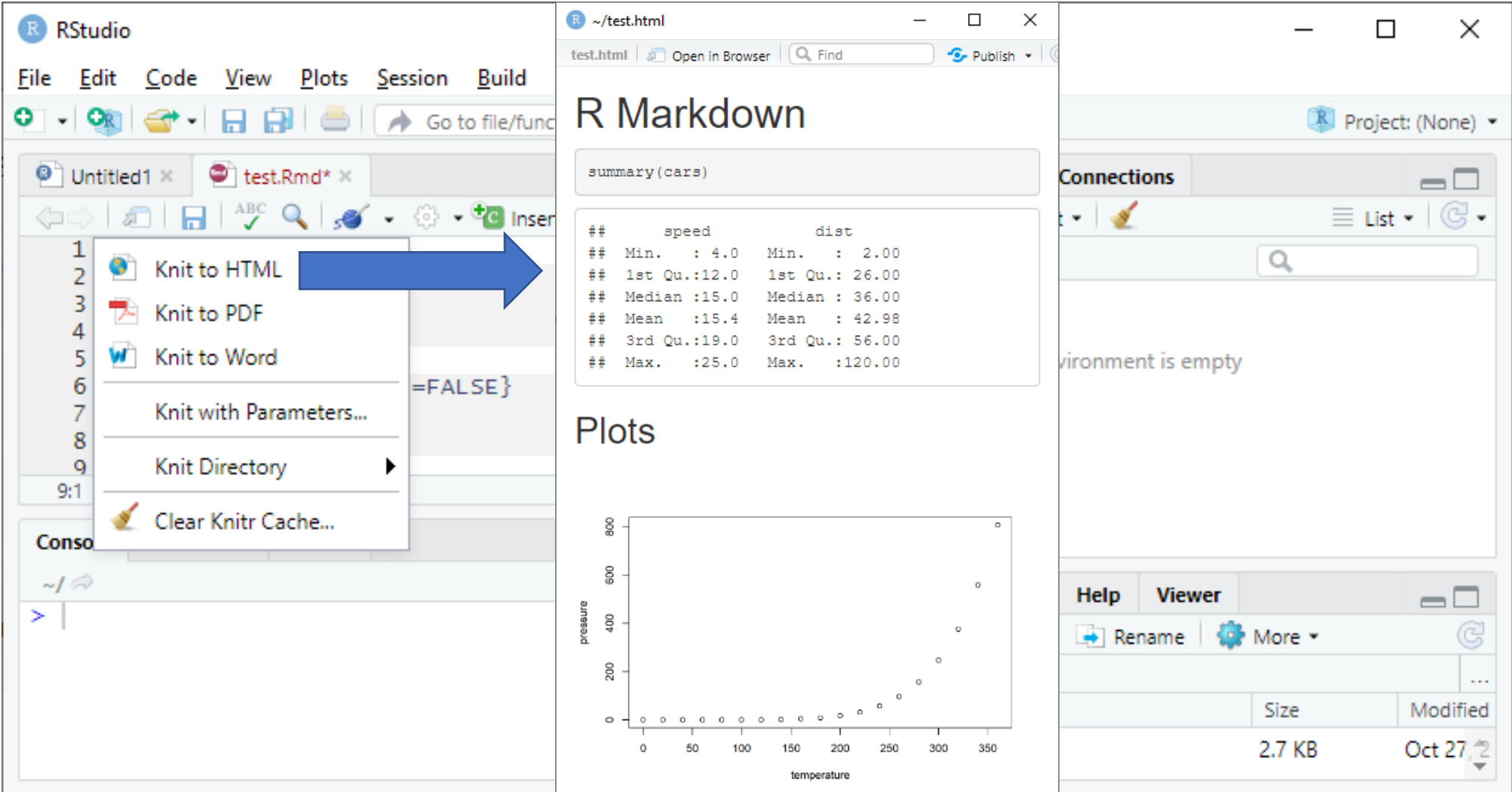
# Data Science



# Data Science



# Data Science



The screenshot displays the RStudio environment with the 'Build' menu open, highlighting the 'Knit to HTML' option. A blue arrow points from this option to the 'R Markdown' window. The 'R Markdown' window shows the output of the `summary(cars)` command, which is a table of statistics for the 'cars' dataset. Below the table, the 'Plots' pane displays a scatter plot of 'pressure' versus 'temperature'. The 'Connections' pane on the right shows the environment is empty. The 'Help' and 'Viewer' panes at the bottom right show the file size (2.7 KB) and modification date (Oct 27, 2021).

**R Markdown**

```
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

**Plots**

pressure

temperature

environment is empty

Help Viewer

Rename More

Size Modified

2.7 KB Oct 27, 2021

## Funções: head, tail e summary

# Data Science

The screenshot shows the RStudio environment with the following components:

- Console:** Displays the execution of `head(cars)` and `tail(cars)`.  
`head(cars)` output:  

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

  
`tail(cars)` output:  

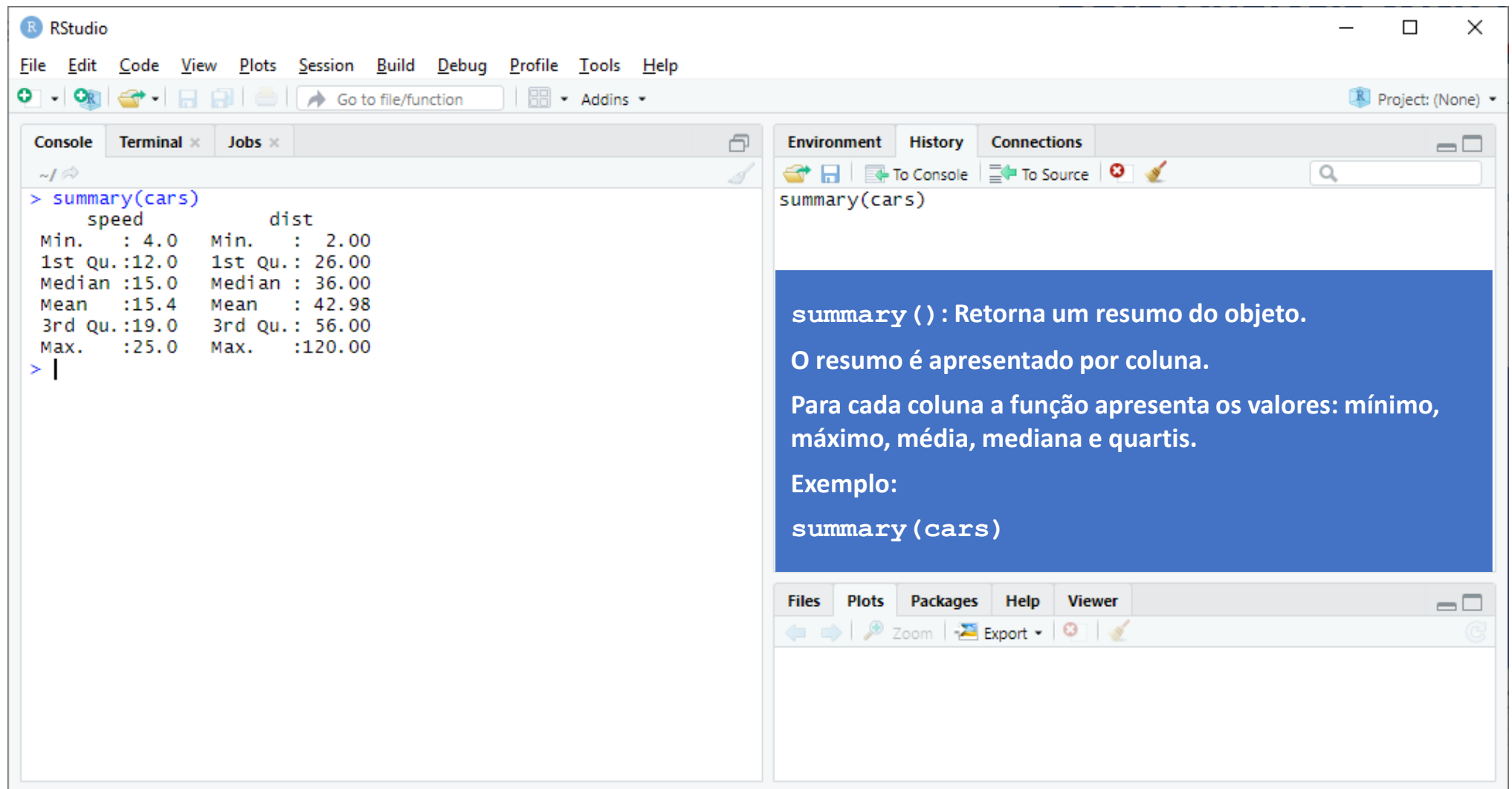
	speed	dist
45	23	54
46	24	70
47	24	92
48	24	93
49	24	120
50	25	85
- Environment:** Shows the objects `head(cars)` and `tail(cars)` in the workspace.
- Blue Overlay Box:** Contains the following text:  

**head () :** Permite visualizar os primeiros registros de um objeto. Exemplo:  
`head (cars)`

**tail () :** Permite visualizar os últimos registros de um objeto. Exemplo:  
`tail (cars)`



# Data Science



The screenshot shows the RStudio interface. The console on the left displays the output of the `summary(cars)` command, which provides a summary of the 'speed' and 'dist' variables. The Environment pane on the right shows the `summary(cars)` object. A blue text box in the Environment pane explains the function and provides an example.

**Console Output:**

```
> 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
> |
```

**Environment Pane:**

`summary(cars)`

**summary () : Retorna um resumo do objeto.**

**O resumo é apresentado por coluna.**

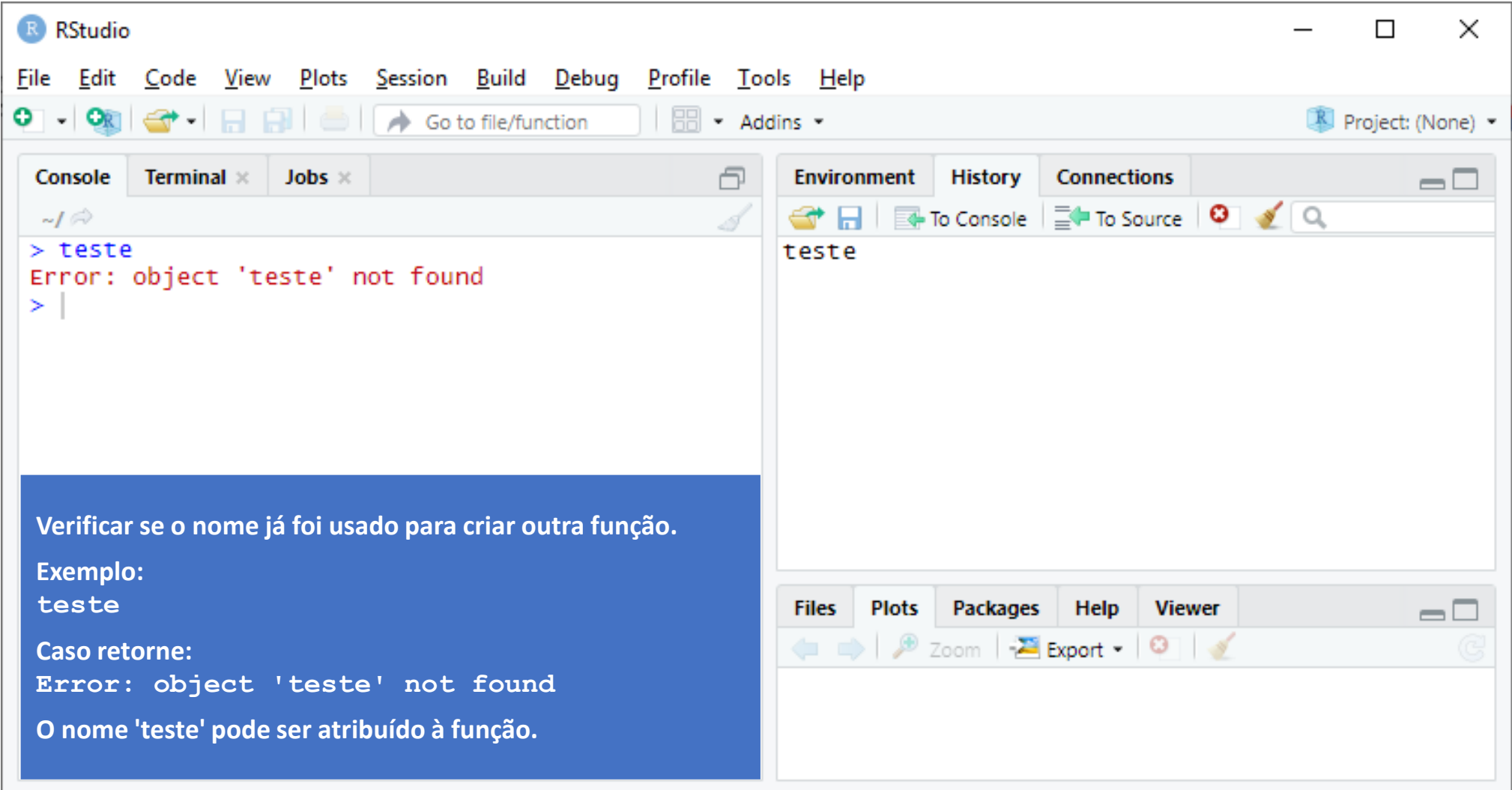
**Para cada coluna a função apresenta os valores: mínimo, máximo, média, mediana e quartis.**

**Exemplo:**

```
summary(cars)
```

## Criando Funções

# Data Science



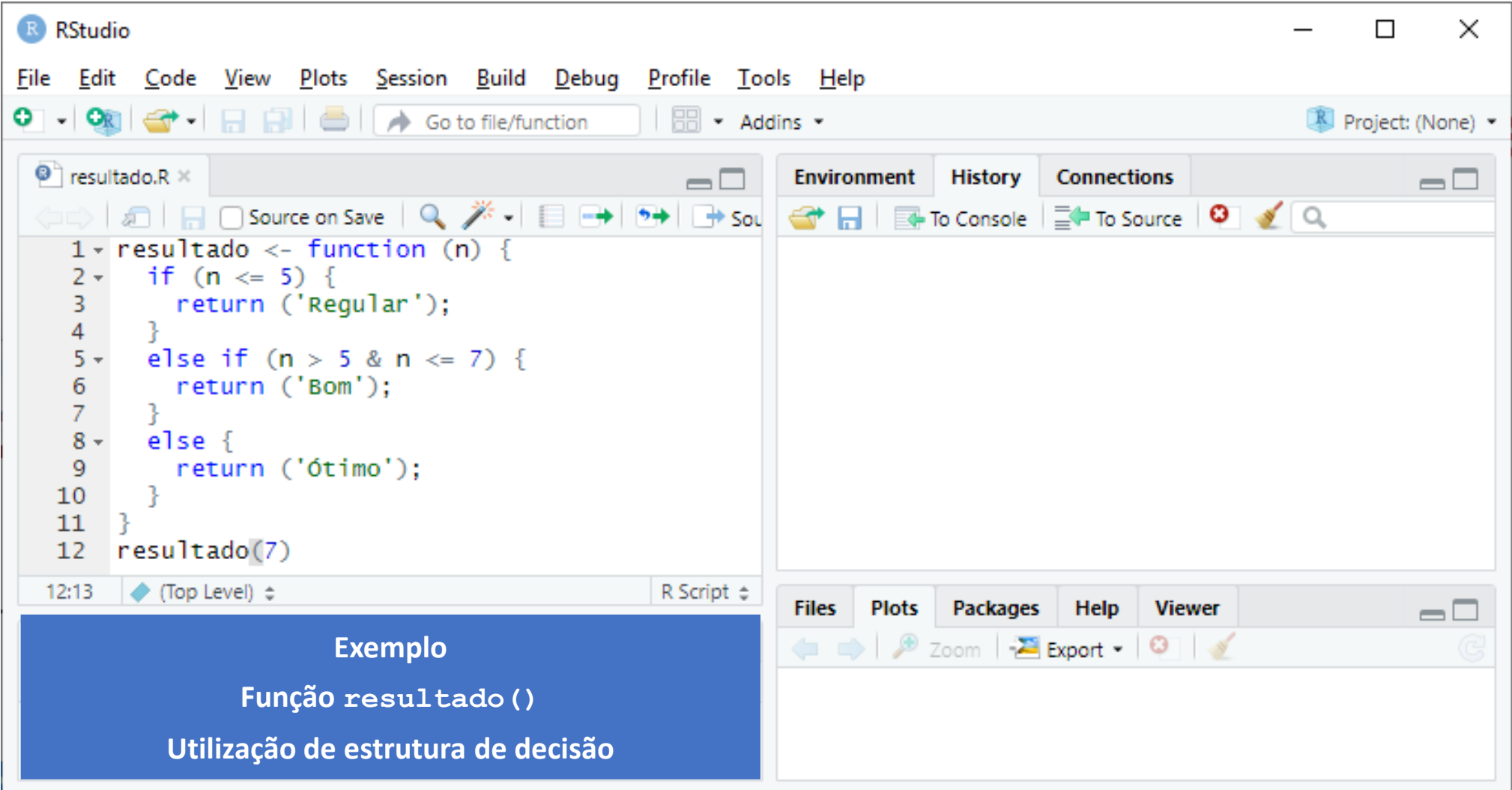
The screenshot shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for creating a new file, opening a file, saving, and other functions. The main workspace is divided into four panes: Console, Environment, History, and Connections. The Console pane shows the following text:

```
> teste  
Error: object 'teste' not found  
> |
```

The Environment pane shows a single object named 'teste'. The bottom toolbar includes icons for Files, Plots, Packages, Help, and Viewer.

Verificar se o nome já foi usado para criar outra função.  
Exemplo:  
teste  
Caso retorne:  
Error: object 'teste' not found  
O nome 'teste' pode ser atribuído à função.

# Data Science



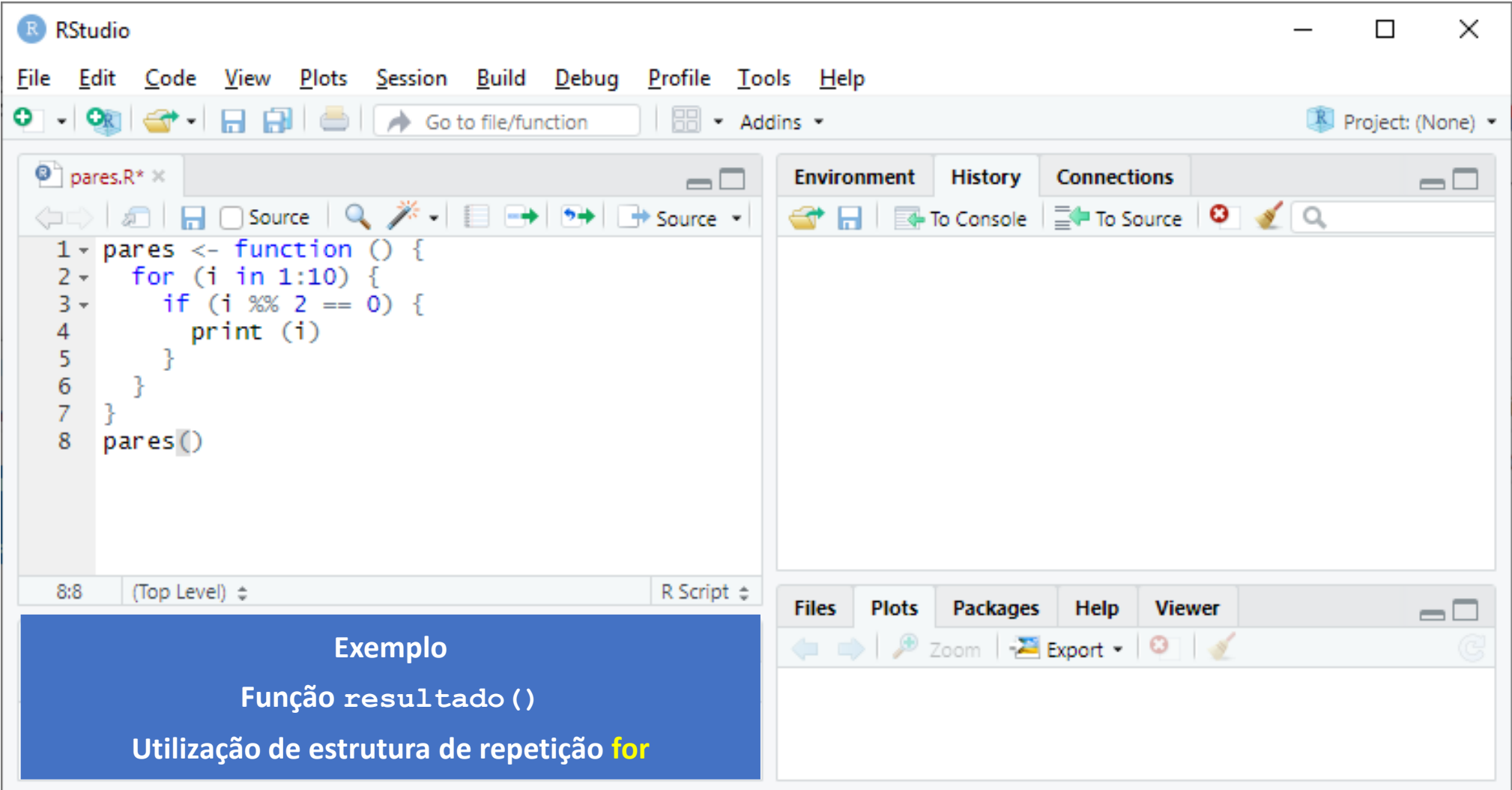
The screenshot displays the RStudio interface. The main editor window shows a script file named 'resultado.R' containing the following R code:

```
1 resultado <- function (n) {  
2   if (n <= 5) {  
3     return ('Regular');  
4   }  
5   else if (n > 5 & n <= 7) {  
6     return ('Bom');  
7   }  
8   else {  
9     return ('Ótimo');  
10  }  
11 }  
12 resultado(7)
```

The right-hand pane is divided into three tabs: 'Environment', 'History', and 'Connections'. The 'Environment' tab is active, showing an empty workspace. The bottom status bar indicates the current position is at line 12, column 13, at the 'Top Level' of the script.

**Exemplo**  
**Função resultado()**  
**Utilização de estrutura de decisão**

# Data Science



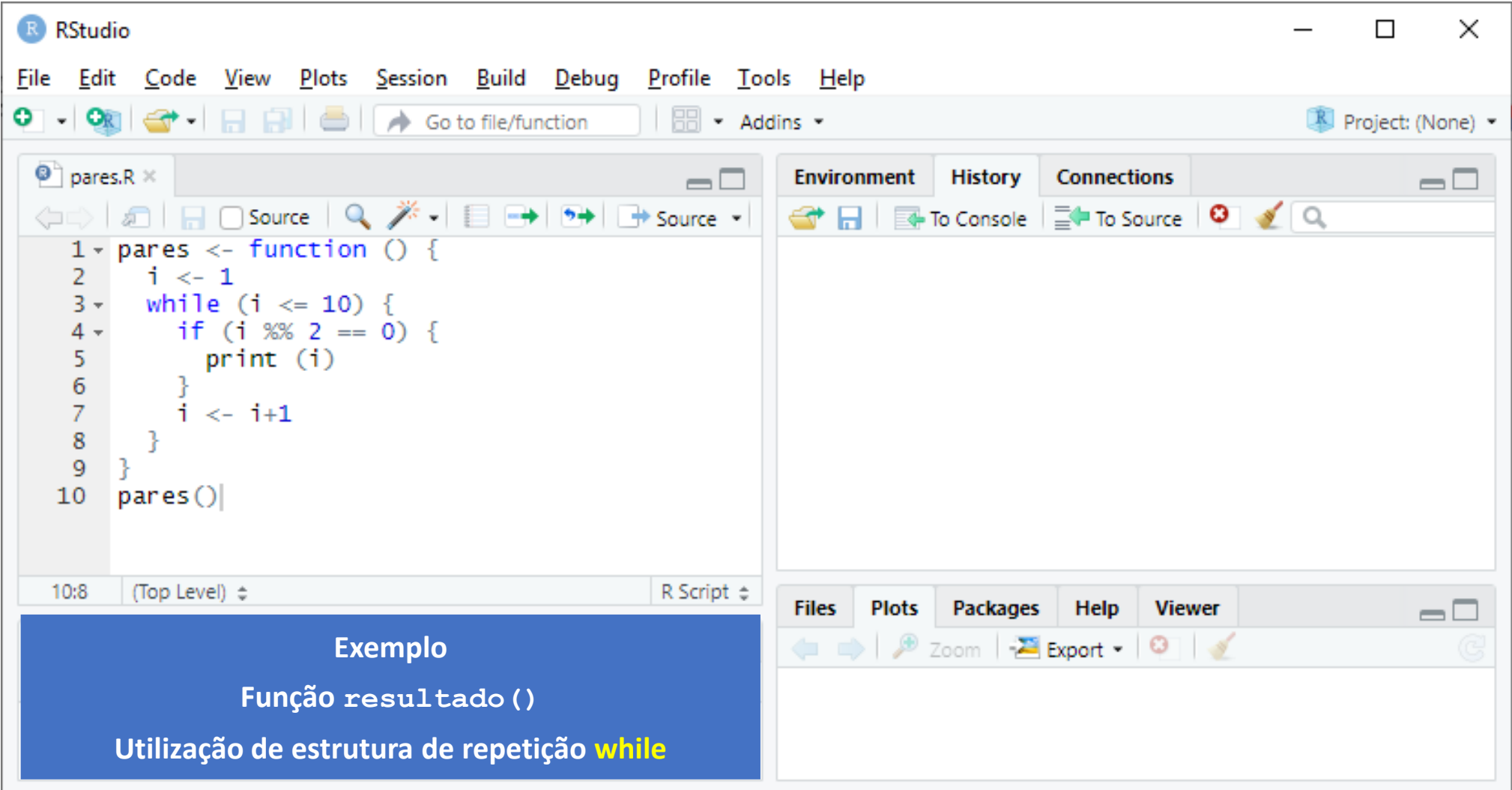
The screenshot displays the RStudio interface. The main editor window shows a script named 'pares.R' with the following R code:

```
1 pares <- function () {  
2   for (i in 1:10) {  
3     if (i %% 2 == 0) {  
4       print (i)  
5     }  
6   }  
7 }  
8 pares()
```

The status bar at the bottom of the editor indicates '8:8 (Top Level) R Script'. To the right of the editor are panels for 'Environment', 'History', and 'Connections'. At the bottom of the RStudio window are panels for 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. A blue overlay box is positioned at the bottom left of the RStudio window, containing the following text:

Exemplo  
Função resultado ()  
Utilização de estrutura de repetição for

# Data Science



The screenshot shows the RStudio interface. The main editor window displays the following R code:

```
1 pares <- function () {  
2   i <- 1  
3   while (i <= 10) {  
4     if (i %% 2 == 0) {  
5       print (i)  
6     }  
7     i <- i+1  
8   }  
9 }  
10 pares()
```

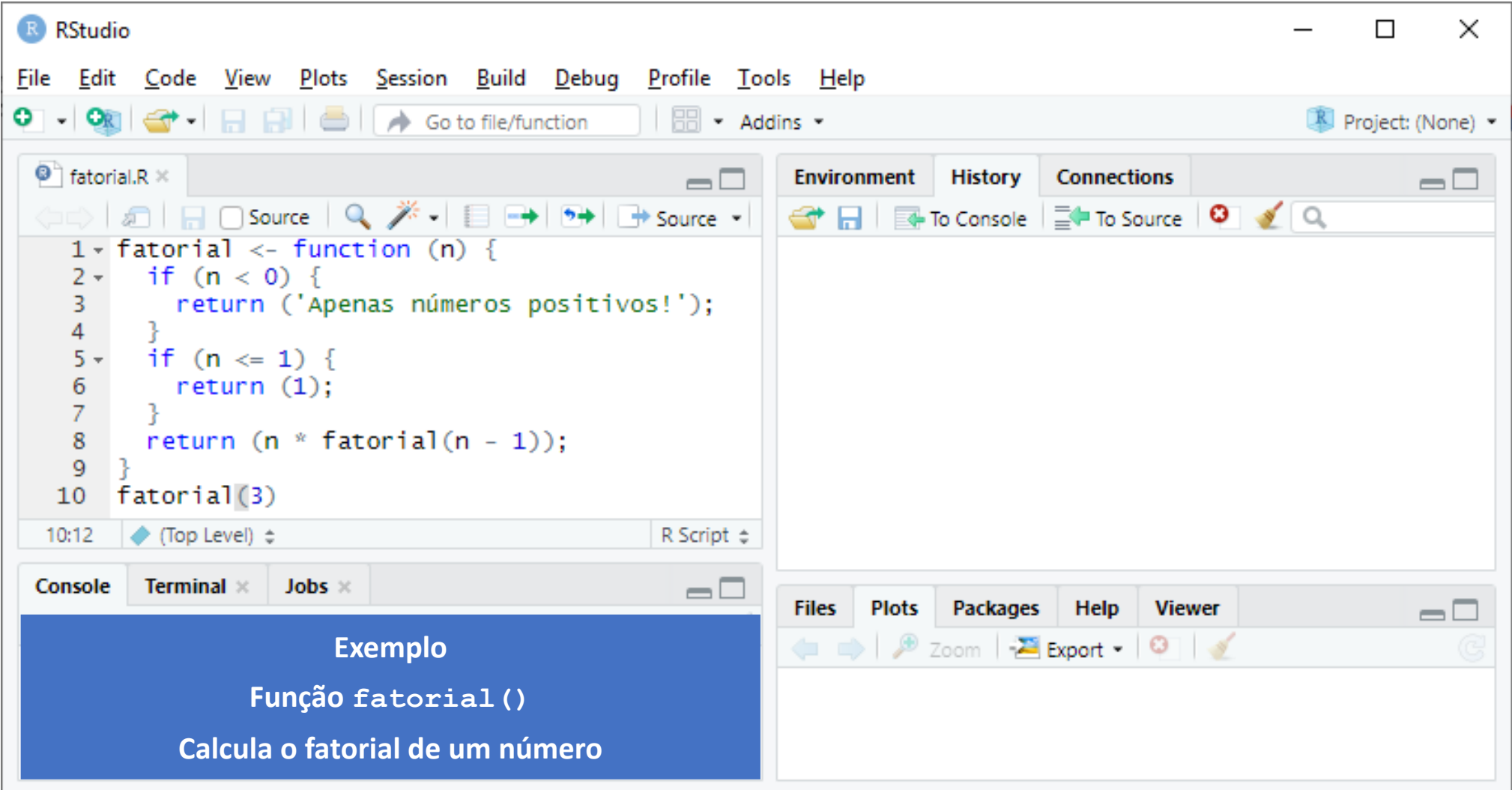
The status bar at the bottom of the editor shows "10:8 (Top Level) R Script".

Below the editor, a blue box contains the following text:

**Exemplo**  
**Função resultado ()**  
**Utilização de estrutura de repetição while**

The right-hand side of the RStudio window contains several panels: "Environment", "History", "Connections", "Files", "Plots", "Packages", "Help", and "Viewer". The "Environment" panel is currently active and empty. The "Files" panel shows a list of files in the current project.

# Data Science



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ + Source Go to file/function Addins Project: (None)

fatorial.R x

```
1 fatorial <- function (n) {  
2   if (n < 0) {  
3     return ('Apenas números positivos!');  
4   }  
5   if (n <= 1) {  
6     return (1);  
7   }  
8   return (n * fatorial(n - 1));  
9 }  
10 fatorial(3)
```

10:12 (Top Level) R Script

Environment History Connections

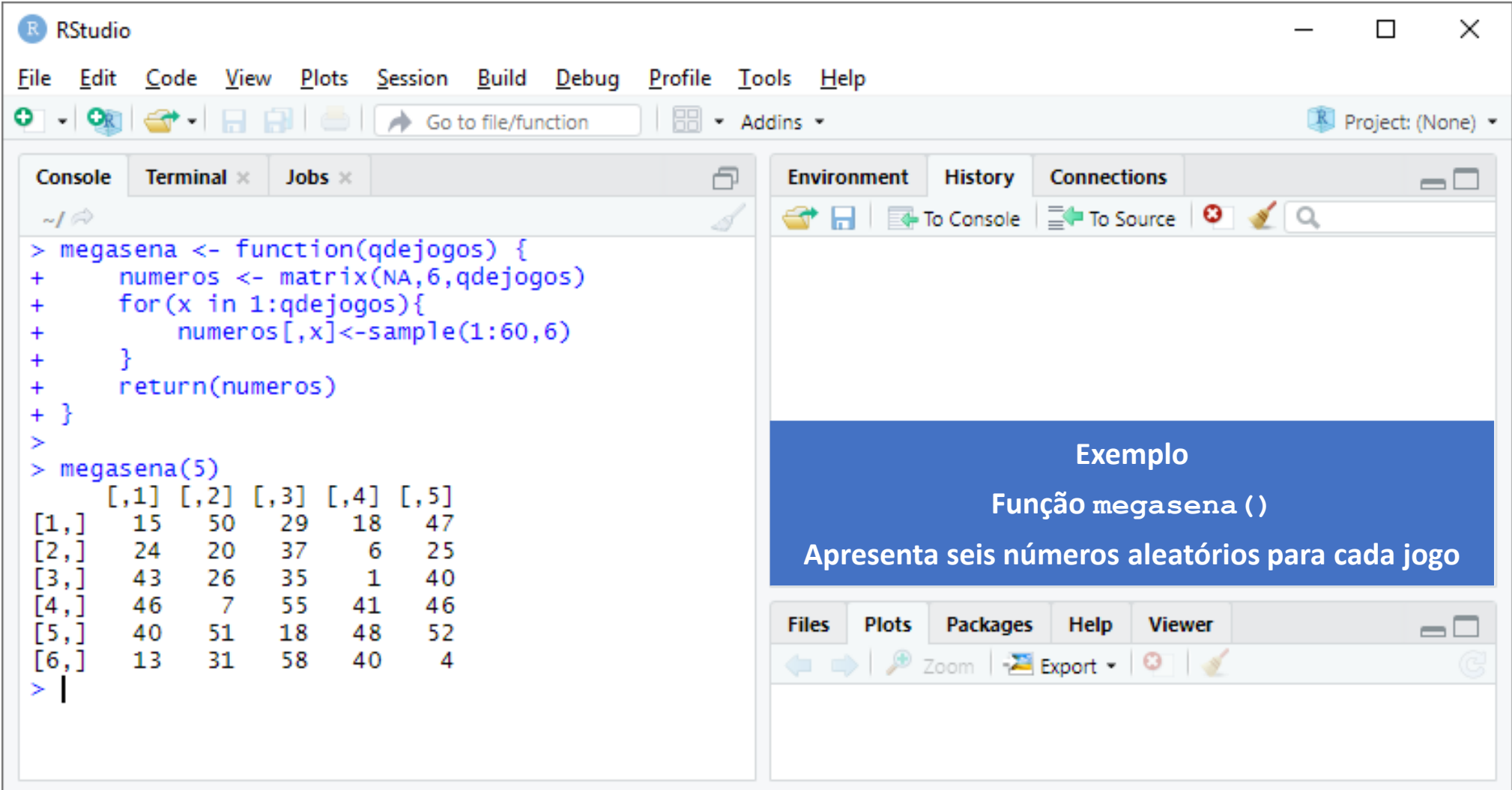
To Console To Source

Files Plots Packages Help Viewer

Zoom Export

**Exemplo**  
**Função fatorial()**  
**Calcula o fatorial de um número**

# Data Science



The screenshot shows the RStudio environment with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for creating a new file, opening a file, saving, and a search bar labeled "Go to file/function".
- Console:** Displays the execution of the `megasena` function. The function definition is as follows:

```
> megasena <- function(qdejogos) {  
+   numeros <- matrix(NA,6,qdejogos)  
+   for(x in 1:qdejogos){  
+     numeros[,x]<-sample(1:60,6)  
+   }  
+   return(numeros)  
+ }  
>  
> megasena(5)
```

The output is a 6x5 matrix of random numbers:

```
      [,1] [,2] [,3] [,4] [,5]  
[1,]   15   50   29   18   47  
[2,]   24   20   37    6   25  
[3,]   43   26   35    1   40  
[4,]   46    7   55   41   46  
[5,]   40   51   18   48   52  
[6,]   13   31   58   40    4
```
- Environment/History/Connections Panel:** Contains a blue box with the text:

**Exemplo**  
**Função megasena ( )**  
**Apresenta seis números aleatórios para cada jogo**
- Files/Plots/Packages/Help/Viewer Panel:** Includes icons for navigating between panels, zooming, and exporting.



## **Análise Descritiva**

### **Medidas de Resumo Numérico**

## Medidas de resumo numérico

Uma forma de resumir os dados é através das medidas-resumo. A natureza da variável vai orientar qual tipo de medida deve ser utilizada.

Há três tipos de medidas-resumo:

- Tendência central
- Dispersão(variabilidade)
- Separatrizes

## Medidas de resumo numérico

### Medidas de Tendência Central

Informam o valor do ponto em torno do qual os dados se distribuem.

Representam um conjunto de valores ou dados através de um único número.

As medidas de tendência central mais utilizadas são:

- **Média aritmética:** soma de todos os valores e dividido pelo número de dados do conjunto;
- **Moda:** valor com maior frequência no conjunto de dados;
- **Mediana:** valor central (quando o número de elementos do conjunto é ímpar) ou média aritmética dos dois valores mais centrais (quando o número de elementos do conjunto é par).

## Medidas de resumo numérico

### Medidas de Dispersão ou Variabilidade

Usadas para verificar a regularidade de um conjunto de dados, quanto menor o grau de dispersão, maior é a regularidade.

As principais medidas de dispersão são:

- **Amplitude:** Apresenta a diferença entre o maior e o menor valor de um conjunto de dados.
- **Desvio absoluto médio:** Apresenta o afastamento médio dos elementos em relação à média aritmética ou em relação à mediana.
- **Variância:** Mede a variabilidade do conjunto de dados em termos de desvios quadrados em relação à média aritmética.
- **Desvio padrão:** É a raiz quadrada da variância. Indica o quanto um conjunto de dados é uniforme. Quanto mais próximo de 0 (zero) for o desvio padrão, mais homogêneos são os dados.
- **Coeficiente de variação:** É a razão do desvio padrão pela média. É frequentemente expresso como uma porcentagem.

## Medidas de resumo numérico

### Medidas de Separatrizes

Dividem a distribuição em um certo número de partes iguais. O objetivo das separatrizes é proporcionar uma melhor ideia da dispersão do conjunto de dados, principalmente da simetria ou assimetria da distribuição.

As medidas de separatrizes são:

- **Quartis:** Dividem a distribuição em quatro partes iguais.
  - O primeiro quartil ou quartil inferior delimita os 25% menores valores.
  - O segundo quartil ou quartil central é a mediana, que separa os 50% menores dos 50% maiores valores.
  - O terceiro quartil ou quartil superior delimita os 25% maiores valores.
- **Decis:** Dividem a distribuição em 10 partes iguais.
- **Percentis:** Dividem a distribuição em 100 partes iguais.

## Tendência Central

## Tendência Central

Indicam, em geral, um valor central em torno do qual os dados estão distribuídos.

As principais medidas de tendência central são:

- média aritmética
- mediana
- moda

## Média aritmética

A média aritmética indica o valor em torno do qual há um equilíbrio na distribuição dos dados e pode ser expressa da seguinte forma:

$$\bar{x} = \sum_{i=1}^n \frac{x_i}{n}$$

Isto significa que deve-se somar todos os valores observados e dividir o resultado pelo tamanho da amostra.

A função **mean ( )** do R calcula a **média**:

```
> a <- c(1,2,2,2,2,3,4,4,5,6,6,7,8,8,8,9)
> mean(a)
[1] 4.8125
```



## Mediana

A mediana é o valor central em um conjunto de dados após ordenado. Esta medida divide os dados em duas partes iguais, sendo metade dos valores abaixo da mediana e, a outra metade, acima.

Se um conjunto de dados contém um total de  $n$  observações, no qual  $n$  é ímpar, a mediana é o valor do meio. Se  $n$  for par, a mediana é usualmente tomada como média dos dois valores mais centrais do intervalo.

A função `median()` do R calcula a mediana:

```
> a <- c(1,2,2,2,2,3,4,4,5,6,6,7,8,8,8,9)
```

```
> median(a)
```

```
[1] 4.5
```

## Moda

A moda é o valor mais frequente no conjunto de dados.

Esta medida-resumo também pode ser aplicada a variáveis qualitativas.

R não tem uma função in-built padrão para calcular a moda. Portanto, para calcular a moda, é preciso instalar o package **modeest**. Após a instalação, a função `mfv()` "most frequent value" deve ser usada para obter-se a **moda**.

```
> install.packages("modeest")
```

```
> library(modeest)
```

```
> a <- c(1,2,2,2,2,3,4,4,5,6,6,7,8,8,8,9)
```

```
> mfv(a)
```

```
[1] 2
```

## Dispersão

## Dispersão

Medidas de dispersão têm o objetivo de quantificar a **variabilidade** dos valores em um conjunto de dados. Há várias medidas de dispersão:

- Amplitude total
- Desvio médio absoluto
- Variância
- Desvio padrão
- Coeficiente de variação

## Amplitude total

A amplitude total é obtida a partir da diferença entre o máximo( $x_{(n)}$ ) e mínimo( $x_{(1)}$ ) em um conjunto de dados ordenados.

Esta medida possui o valor 0 (zero) como limite inferior e é altamente sensível à valores extremos.

$$\Delta = x_{(n)} - x_{(1)}$$

A diferença entre dos valores retornados pelas funções **max()** e **min()** do R apresenta a **amplitude** do conjunto de dados:

```
> a <- c(1,2,3,4,5,6,7,8,9)
> amplitude <- (max(a) - min(a))
> amplitude
[1] 8
```

### *Saiba mais:*

A função **range()** retorna os valores max e min:

```
> range(a)
[1] 1 9
```

## Desvio mediano absoluto

O desvio mediano absoluto (MAD) é uma medida robusta da tendência central. Apresenta, portanto, bom desempenho para dados extraídos de uma ampla gama de distribuições de probabilidade não normalmente distribuídas.

Diferente da combinação de média e desvio padrão, o MAD não é sensível à presença de outliers.

```
> a <- c(1,2,3,4,5,6,7,8,9)
```

```
> mad(a)
```

```
[1] 2.9652
```

## Desvio médio absoluto

É o cálculo da média dos desvios absolutos. Para o seu cálculo, primeiramente deve ser calculada a média, posteriormente os desvios das observações em relação a média em módulo, e por último a média aritmética destes desvios.

$$dma = \frac{\sum |x_i - \bar{x}|}{n}$$

*Lembre-se:*

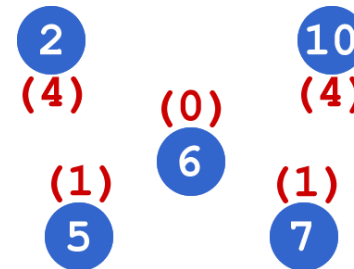
$\bar{x} \rightarrow$  média

Calcular o **desvio médio absoluto** no R:

```
> x <- c(2,5,6,7,10)
```

```
> dma <- sum(abs(x-mean(x))) / length(x)
```

```
[1] 2
```



### Desvio médio absoluto:

$x = (2, 5, 6, 7, 10)$

$n = 5$

média =  $(2+5+6+7+10) / 5 = 6$

$|2-6| = 4$

$|5-6| = 1$

$|6-6| = 0$

$|7-6| = 1$

$|10-6| = 4$

$(4+1+0+1+4) = 10$

$10 / n = 10 / 5 = 2$

**dma = 2**

## Variância

Variância é a medida de **dispersão** mais usada e conhecida na Estatística. Ela **quantifica a variabilidade dos dados em torno da média**, ou seja, mede o quão longe do valor esperado os dados se encontram. O seu cálculo é feito da seguinte maneira:

$$Var = \frac{\sum (x - \bar{x})^2}{n - 1}$$

*Lembre-se:*

$\bar{x}$  -> média

A função `var()` do R calcula a **variância amostral**:

```
> x <- c(10,12,16,17,20)
```

```
> var(x)
```

```
[1] 16
```

### Cálculo da variância:

$n = 5$

média =  $(10+12+16+17+20)/5 = 15$

$10-15 = -5 \Rightarrow -5^2 = 25$

$12-15 = -3 \Rightarrow -3^2 = 9$

$16-15 = 1 \Rightarrow 1^2 = 1$

$17-15 = 2 \Rightarrow 2^2 = 4$

$20-15 = 5 \Rightarrow 5^2 = 25$

$25+9+1+4+25 = 64$

variância =  $64 / (5-1) = 16$



## Desvio padrão

O cálculo da variância eleva ao quadrado a soma da diferença dos desvios. Isso faz com a variância não tenha a interpretação na mesma escala em os dados foram medidos. Utiliza-se, portanto, o desvio padrão, obtido através do cálculo da raiz quadrada da variância. Este novo valor possibilita analisar a variabilidade dos dados na mesma escala em que eles foram medidos.

$$sd = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$$

A função `sd()` do R calcula o **desvio padrão**:

```
> x <- c(10,12,16,17,20)
> sd(x)
[1] 4
```

Desvio padrão é frequentemente representado pela letra grega  $\sigma$  (sigma)

## Coeficiente de variação

O coeficiente de variação é a razão entre o desvio padrão e a média. É uma medida relativa que avalia o percentual de variabilidade em relação a média observada.

$$cv = \frac{sd}{\bar{x}} 100$$

O **coeficiente de variação** é calculado no R da seguinte forma:

```
x <- c(10,12,16,17,20)
> cv = (sd(x)/mean(x))*100
> cv
[1] 26.66667 # Aproximadamente 26%
```

## Separatrizes

## Separatrizes

**Separatrizes** são medidas que **dividem os dados em um número de partes iguais**. Para isso a amostra deve estar ordenada, portanto as separatrizes são aplicadas a variáveis quantitativas e qualitativas ordinais.

As principais separatrizes são:

- **percentis** (dividem os dados em 100 partes iguais)
- **decis** (dividem os dados em 10 partes iguais)
- **quartis** (dividem os dados em quatro partes iguais)

*Os quartis são utilizado com mais frequência.*

## Quartis

Os quartis dividem a amostra em quatro partes iguais. O primeiro quartil (Q1) estabelece o limite entre as 25% menores observações e as 75% maiores. O segundo quartil (Q2) é igual a mediana e o terceiro quartil (Q3) separa as 75% menores observações das 25% maiores.

Cálculo do IQR (Inter Quarter Range):  $Q3 - Q1$

```
> summary(women$height)
```

Min	1st Qu	Median	Mean	3rd Qu	Max
58.0	61.5	65.0	65.0	68.5	72.0

```
> quantile(women$height)
```

0%	25%	50%	75%	100%
58.0	61.5	65.0	68.5	72.0

```
> IQR(women$height)
```

```
[1] 7
```

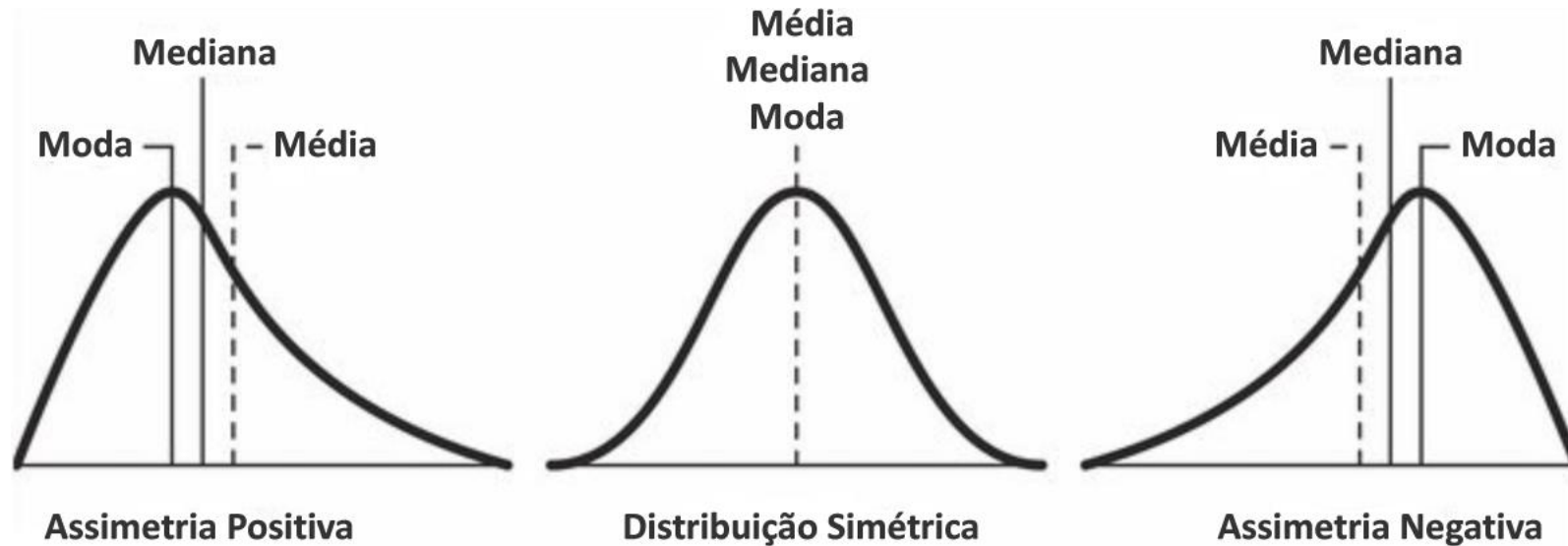
> women	height	weight
1	58	115
2	59	117
3	60	120
4	61	123
5	62	126
6	63	129
7	64	132
8	65	135
9	66	139
10	67	142
11	68	146
12	69	150
13	70	154
14	71	159
15	72	164

## Distribuições Simétricas e Assimétricas

## Distribuições Simétricas e Assimétricas

**Distribuições Simétricas:** Apresentam uma curva de frequências **unimodal** com duas "caudas" simétricas em relação a uma linha vertical que passa por seu ponto mais alto (eixo de simetria).

**Distribuições Assimétricas:** Apresentam uma curva de frequências **unimodal** com uma "cauda" mais longa para a direita (assimetria positiva) ou para a esquerda (assimetria negativa) em relação ao seu ponto mais alto.



## Coeficiente de Assimetria de Pearson

O Coeficiente de Assimetria de Pearson,  $A_p$ , baseia-se na posição relativa das medidas de tendência central de acordo com o tipo de assimetria dos dados.

$$A_p = \frac{3 \cdot (\bar{x} - md)}{sd}$$

$\bar{x}$  => média

$md$  => mediana

$sd$  => desvio padrão

$A_p = 0$  => Simetria

$0.15 < |A_p| < 1$  => Assimetria moderada

$|A_p| > 1$  => Assimetria forte

```
x <- c(10,20,20,30)
xa <- mean(x)      # 20
xb <- median(x)    # 20
xc <- sd(x)         # 8.16
xAp <- 3*(xa-xb)/xc
xAp
[1] 0 #simetria
```

```
y <- c(10,20,20,50)
ya <- mean(y)      # 25
yb <- median(y)    # 20
yc <- sd(y)         # 17.32
yAp <- 3*(ya-yb)/yc
yAp
[1] 0.8660 #positiva
```

```
z <- c(10,20,30,30)
za <- mean(z)      # 22.5
zb <- median(z)    # 25
zc <- sd(z)         # 9.57
zAp <- 3*(za-zb)/zc
zAp
[1] -0.7836 #negativa
```



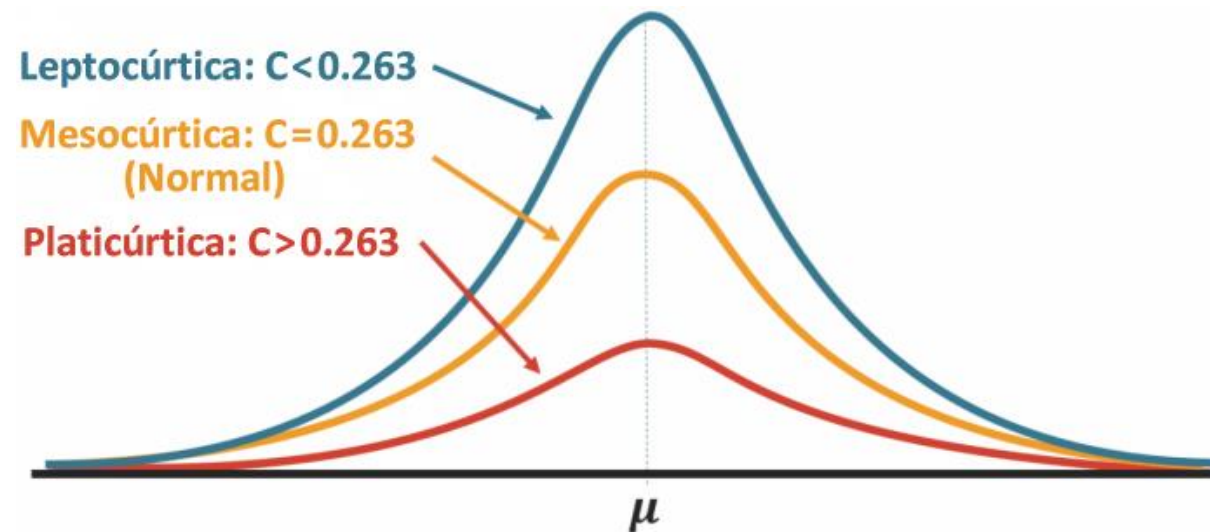
## Curtose

## Curtose

Curtose é uma medida de dispersão que caracteriza o "achatamento" da curva da função de distribuição.

Analisar um conjunto quanto à curtose significa verificar o "grau de achatamento da curva", ou seja, saber se a Curva de Frequência que representa o conjunto é mais "afilada" (leptocúrtica) ou mais "achatada" (platicúrtica) em relação a uma Curva Padrão, chamada de Curva Normal (mesocúrtica).

Assimetria e Curtose são medidas independentes e que não se influenciam mutuamente.



## Curtose

O Índice Percentílico de Curtose (**C**) é calculado conforme a seguinte fórmula:

$$C = \frac{(Q3 - Q1)}{2 \cdot (D9 - D1)}$$

**Q3:** 3º Quartil (75%)

**Q1:** 1º Quartil (25%)

**D9:** 9º Decil (90%)

**D1:** 1º Decil (10%)

```
> x <- c(10,15,20,25,30,35,40)

> quantile(x,c(0.75,0.25,0.9,0.1))
 75%  25%  90%  10%
32.5 17.5 37.0 13.0

> C <- (32.5-17.5)/(2*(37.0-13.0))
> C
[1] 0.3125 # C > 0.263 -> Platicúrtica
```

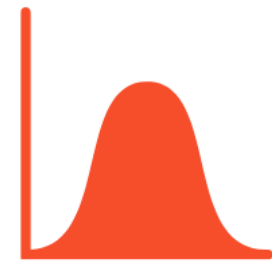
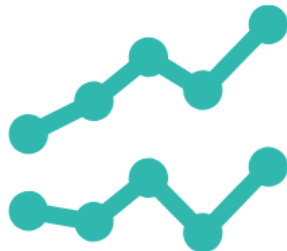
## Gráficos

## Gráficos

A visualização dos dados é uma etapa muito importante da análise estatística.

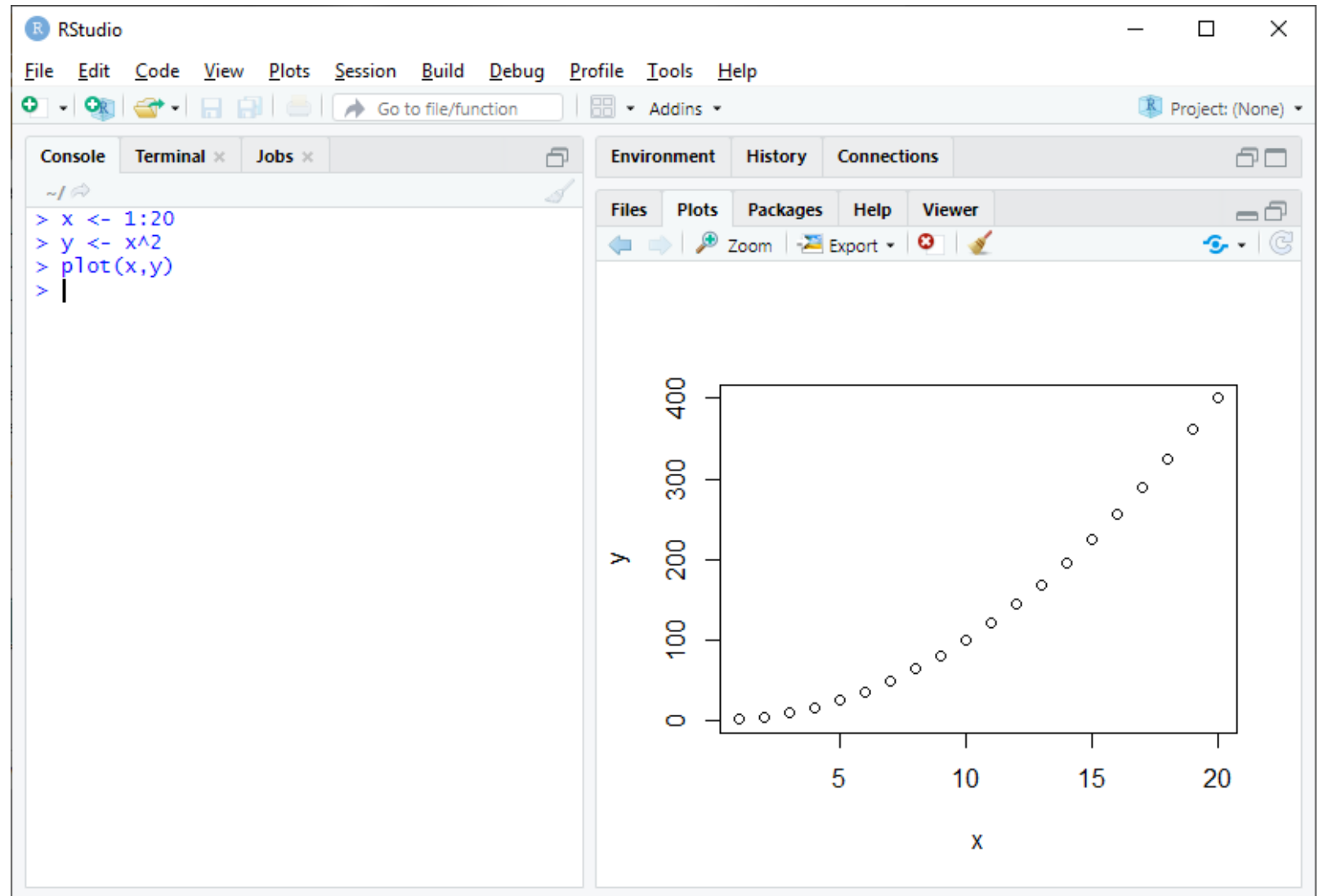
Visualizações podem ser uma simples medida resumo (frequência, média, variância, mínimo, máximo, etc.), um conjunto dessas medidas organizadas em uma tabela ou a representação (de uma parte) dos dados em um gráfico.

As ferramentas disponíveis no R facilitam muito a criação de gráficos dos mais diversos tipos.



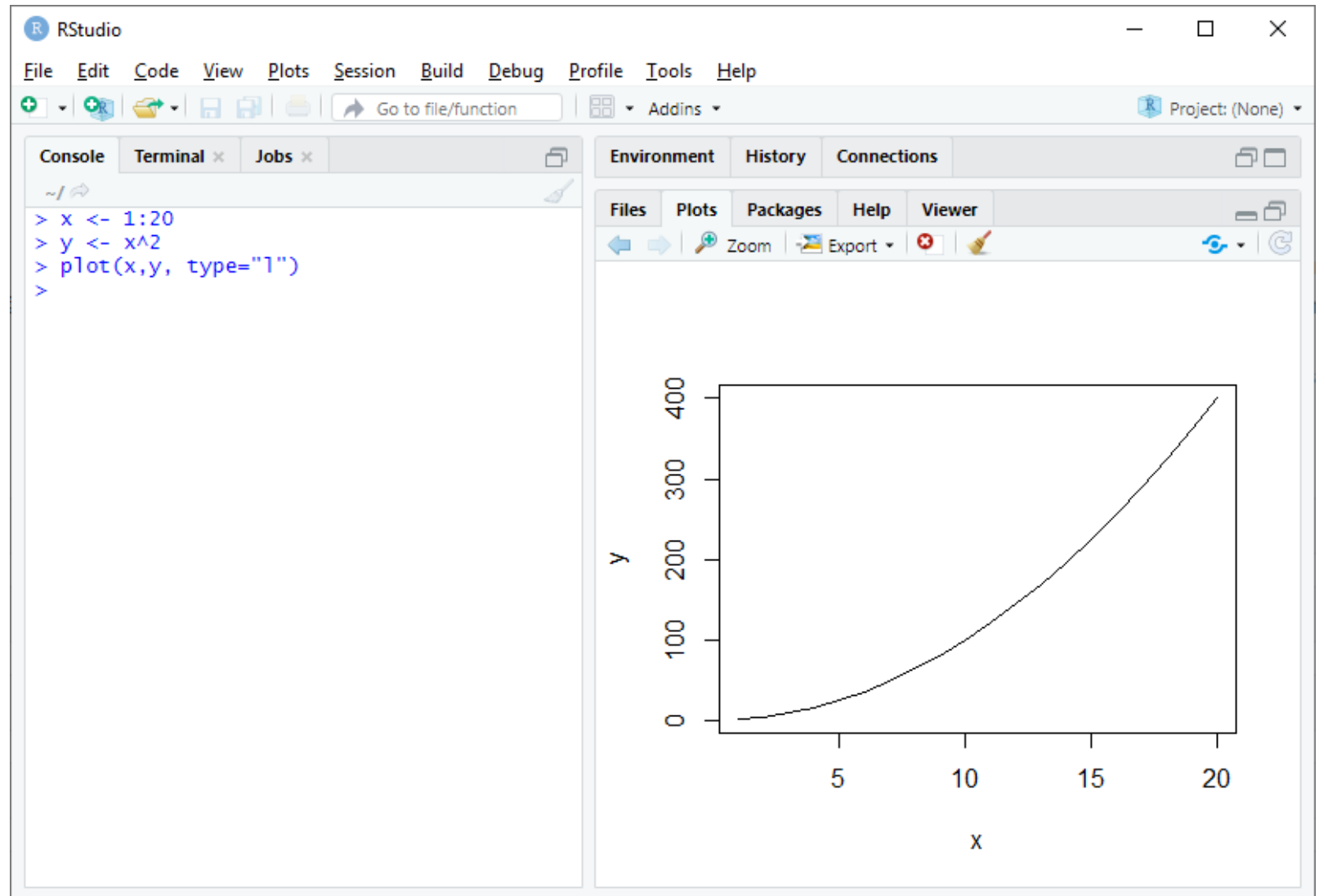
`plot()`

Gera um gráfico simples, atribuindo pontos em coordenadas cartesianas.



`plot()`

Acrescentando o argumento `type='l'`  
substitui os pontos por uma linha.



## `plot()`

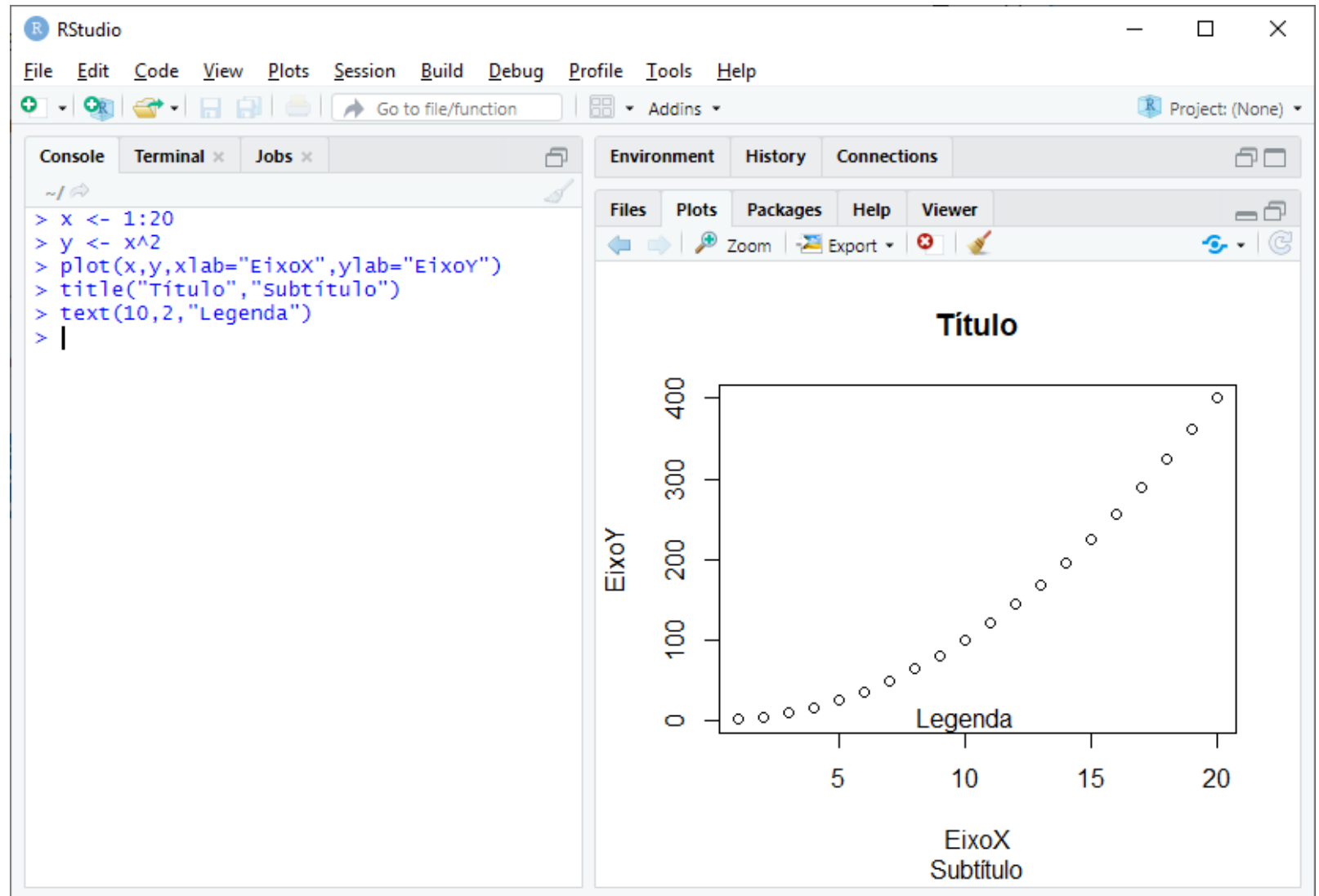
*Outros parâmetros:*

`xlab`: rótulo do eixo x (horizontal)

`ylab`: rótulo do eixo y (vertical)

`title("Título","Subtítulo")`

`text(posX,poxY,"Legenda")`





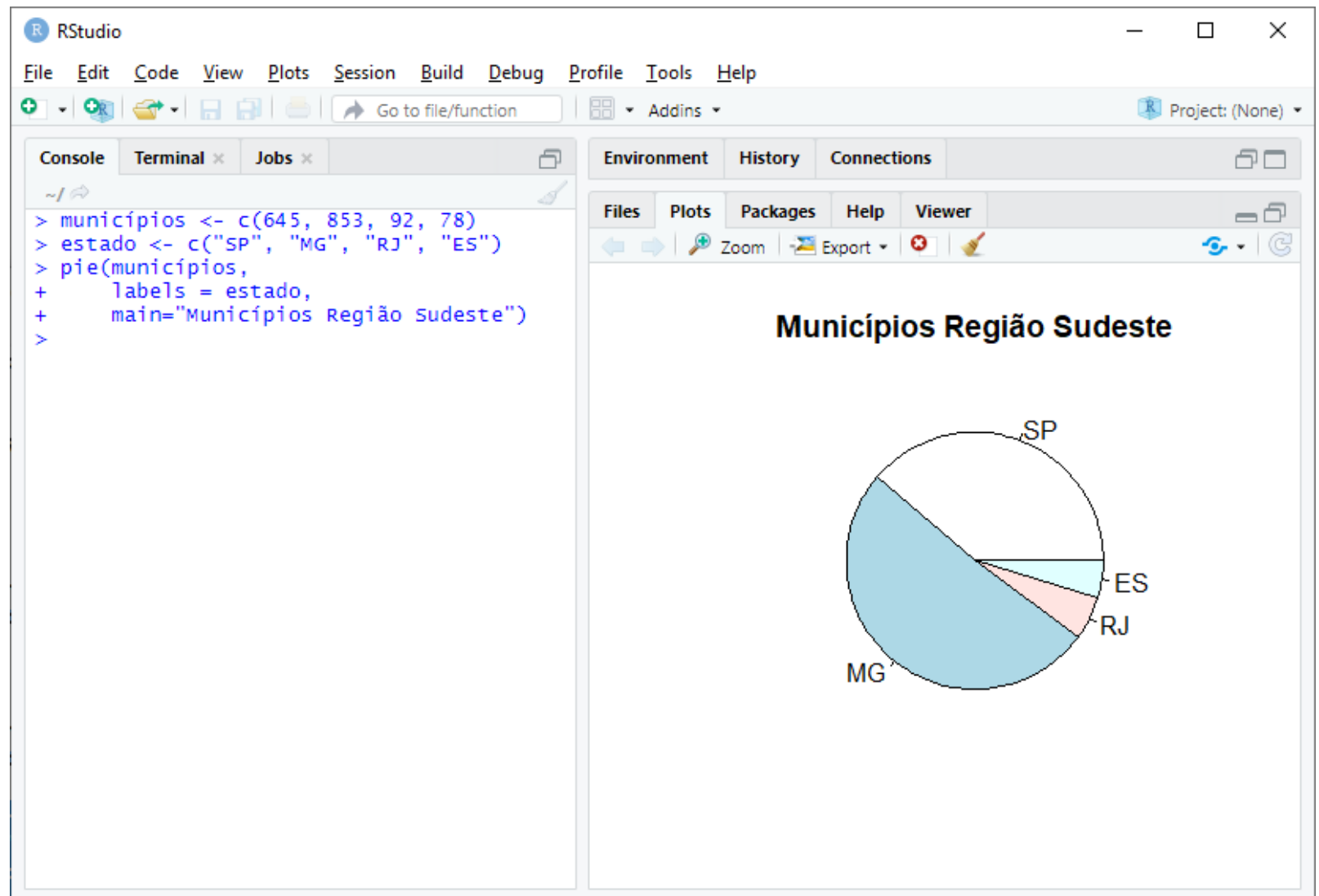
`pie()`

## Gráficos de pizza ou de setor

*Os gráficos de pizza exibem os dados como proporção de um todo.*

*São especialmente úteis para fazer comparações entre grupos.*

*Exemplo: Proporção de municípios de cada estado na Região Sudeste do Brasil.*



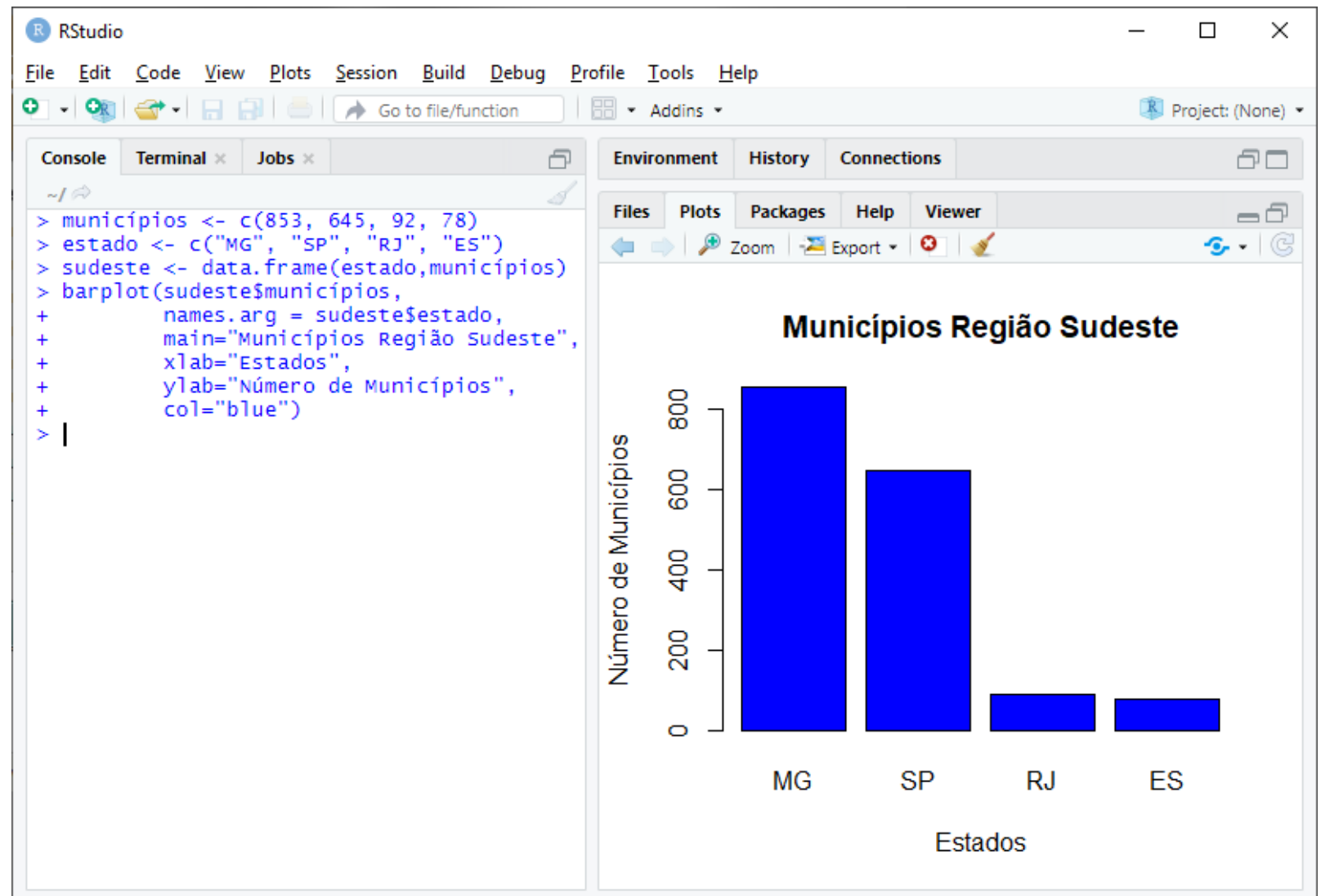
## barplot()

### Gráficos de Barras

*No gráfico de barras cada barra representa a medida de cada elemento do vetor.*

*Cada barra é proporcional à "dimensão" do elemento.*

*Exemplo: Quantidade de municípios de cada estado da Região Sudeste do Brasil.*

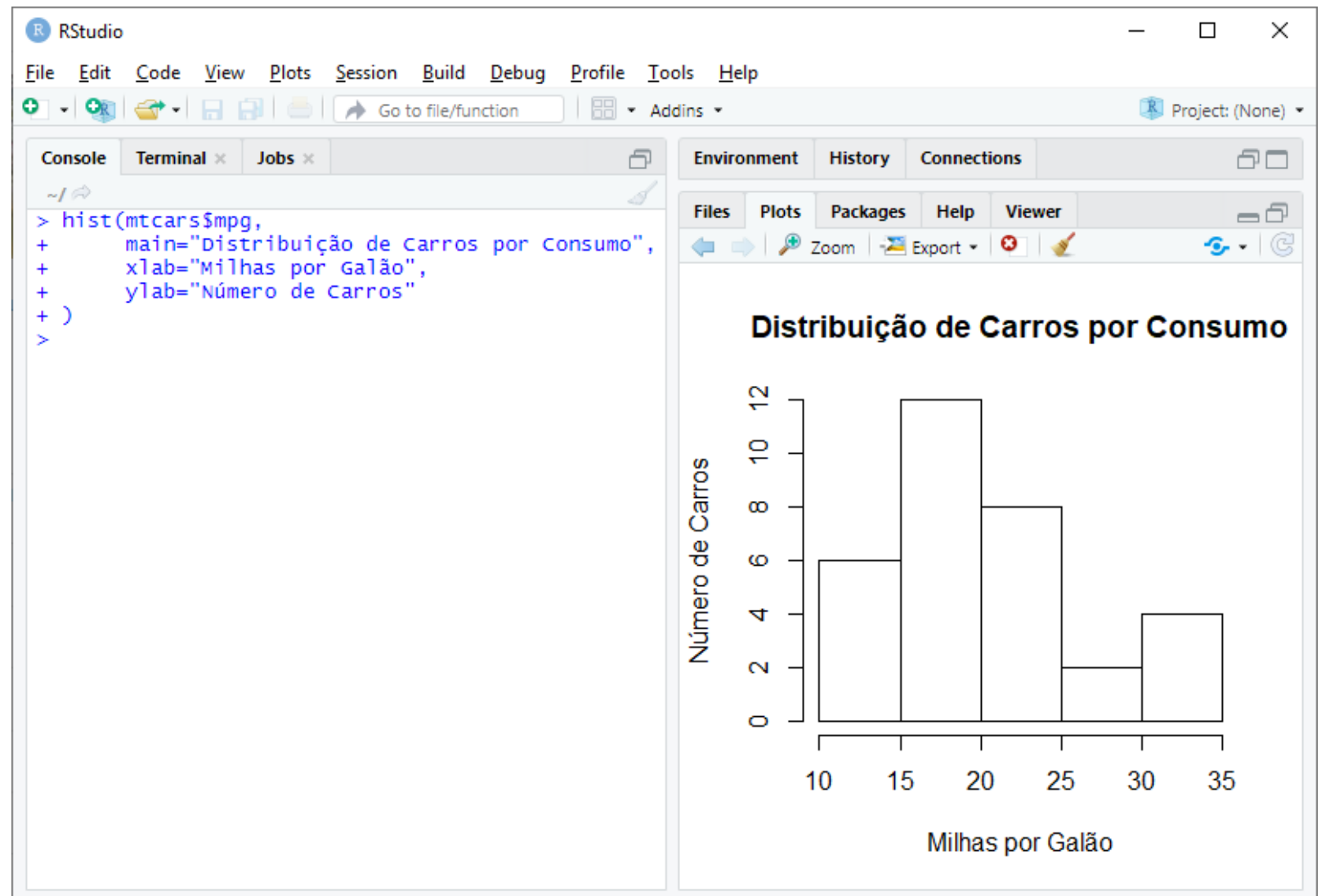


`hist()`

## Histogramas

*Um histograma divide uma série de dados em diferentes classes com igual espaçamento e mostra a frequência de valores em cada classe.*

*O eixo vertical (ordenadas) mostra a **frequência** relativa de cada classe e o eixo horizontal (abscissas) mostra os valores e intervalos das classes.*



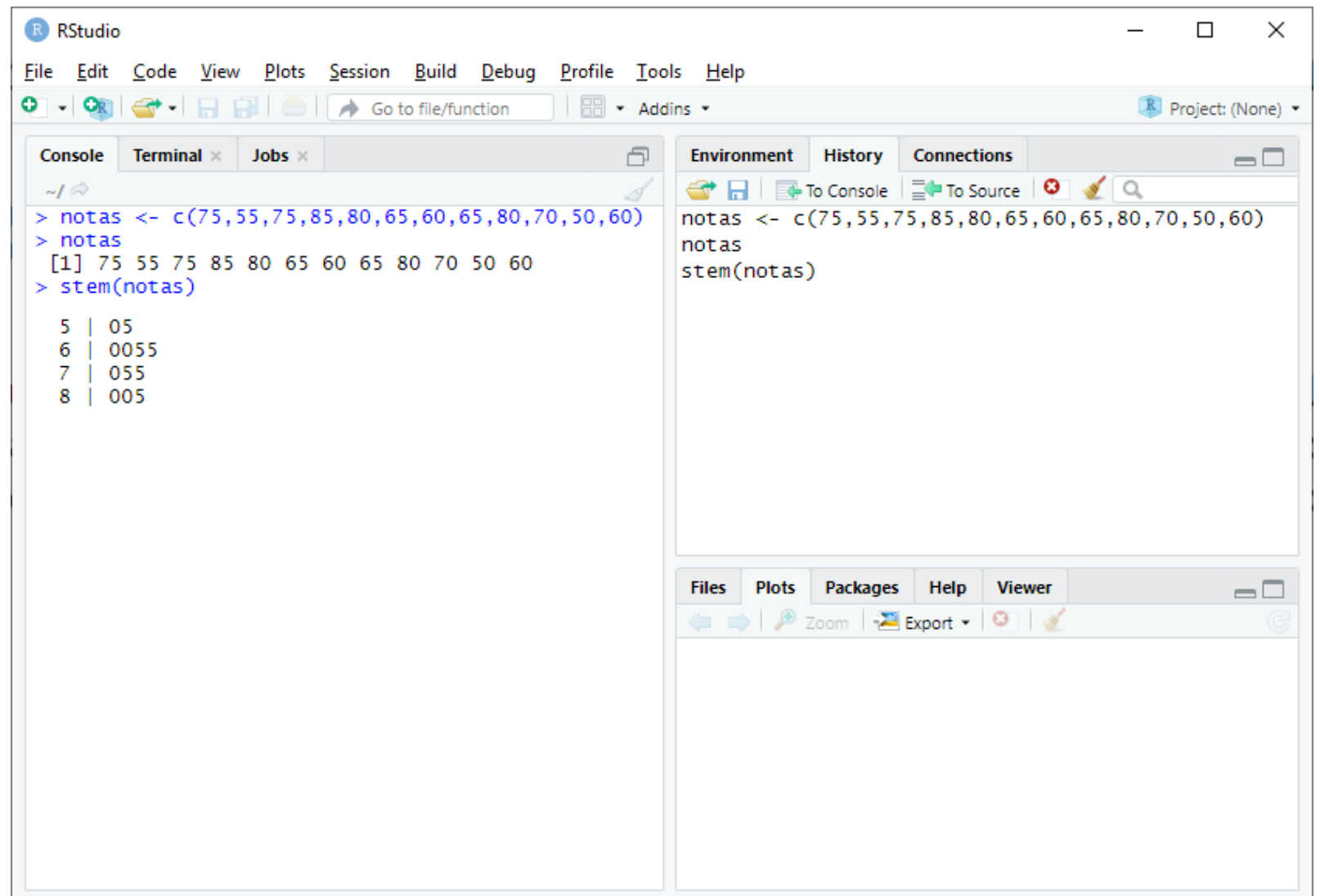
## stem()

## Ramos e Folhas

## Stem and Leaf

*É um tipo de representação entre a tabela e o gráfico, uma vez que, de um modo geral, são apresentados os verdadeiros valores dos dados, mas numa representação, que faz lembrar um histograma.*

*Os ramos são os dígitos dominantes colocados do lado esquerdo de um eixo vertical. Para cada dado toma-se o dígito que se segue imediatamente aos dígitos dominantes e coloca-se do lado direito do eixo, em frente ao respectivo ramo. Estes dígitos são as folhas.*



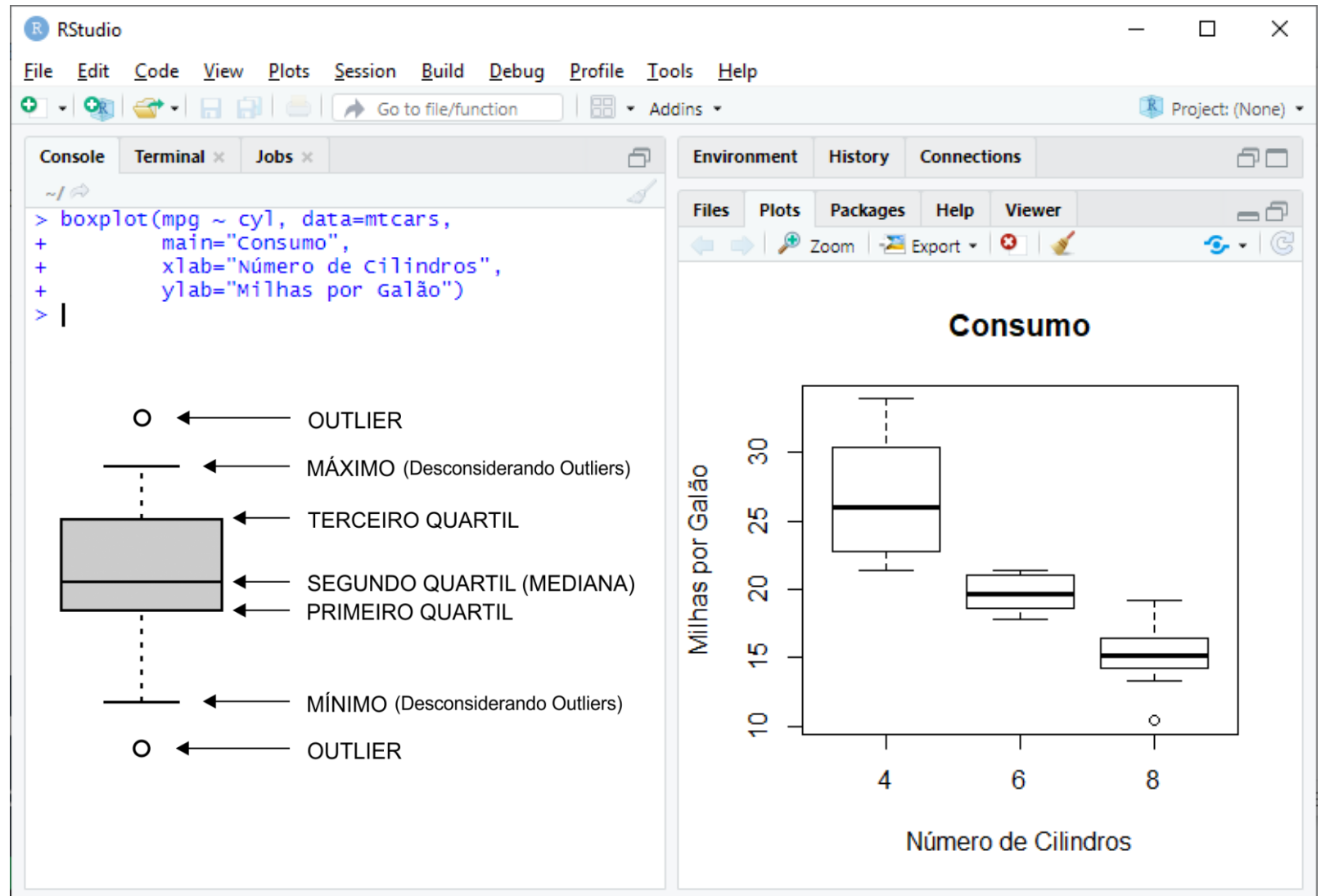
## boxplot()

### Gráficos de Caixa

*Apresenta a distribuição de um conjunto de dados com base em alguns de seus parâmetros descritivos: mediana e quartis.*

*Permite avaliar a simetria e a dispersão do dados.*

*É recomendado para comparação de dois ou mais conjuntos de dados correspondentes às categorias de uma variável qualitativa.*

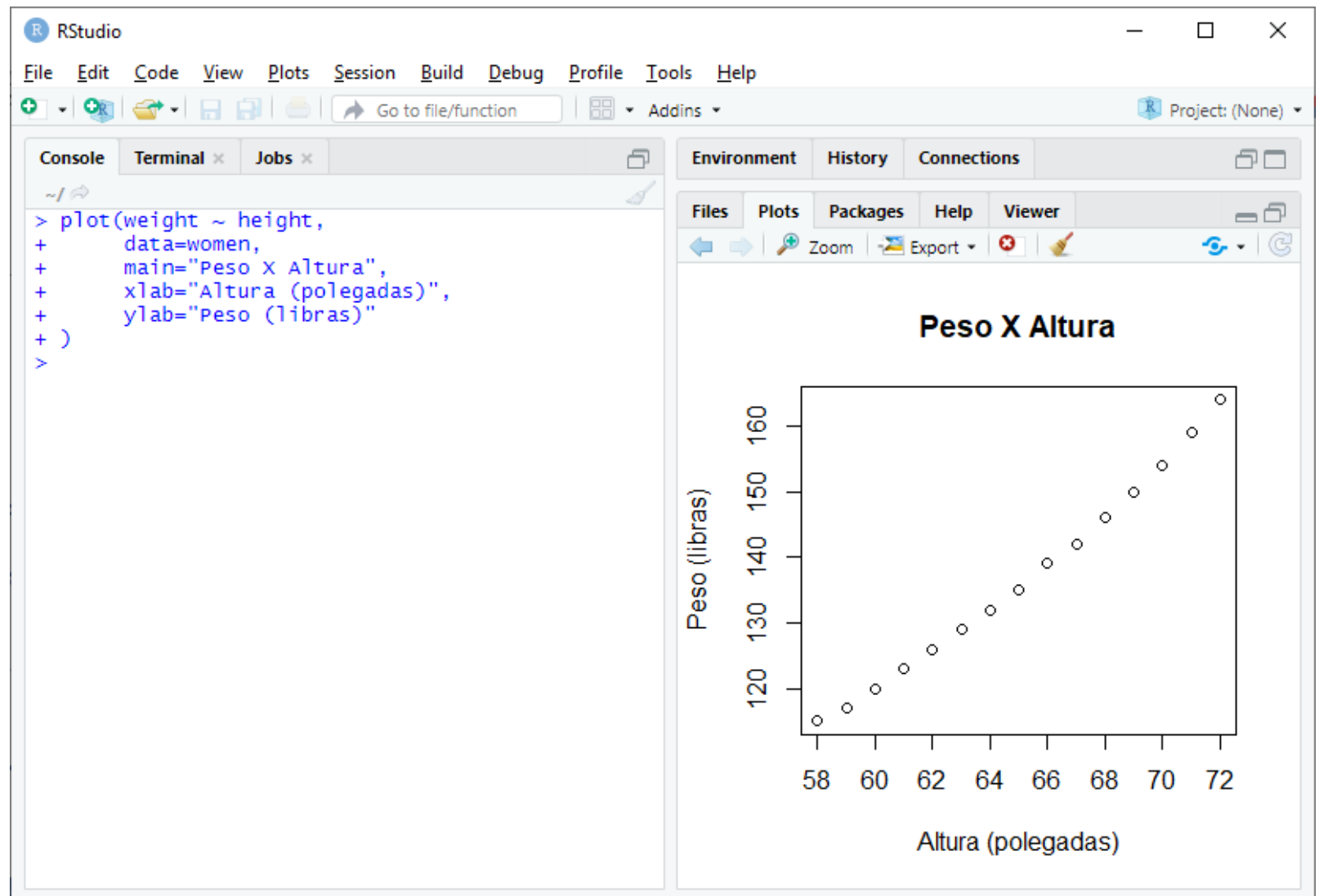


`plot()`

## Gráficos de dispersão

*O gráfico dispersão é usado para correlacionar duas variáveis quantitativas.*

*Importante: Correlação não implica em Causalidade. (Duas coisas correlacionadas não implicam, necessariamente, no fato de uma ser causa da outra.)*

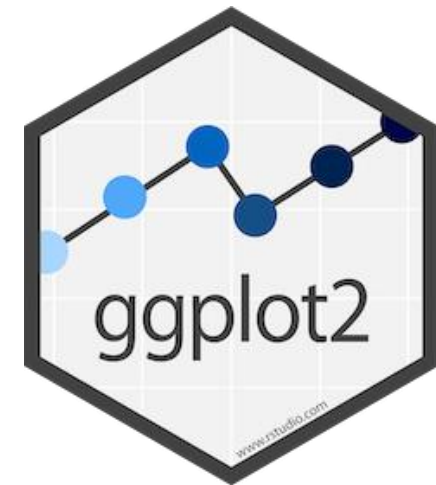


## ggplot2

## ggplot2

Em 2005, Leland Wilkinson publicou o livro *The Grammar of Graphics*, uma fonte de princípios fundamentais para a construção de gráficos estatísticos. O autor defende que um gráfico é o mapeamento dos dados a partir de atributos estéticos – posição, cor, forma e tamanho – e de objetos geométricos – pontos, linhas, barras e caixas.

A partir destes princípios, Hadley Wickham escreveu *A Layered Grammar of Graphics*, sugerindo que os principais aspectos de um gráfico – dados, sistema de coordenadas, rótulos e anotações – podiam ser divididos em camadas, construídas uma a uma na elaboração do gráfico. Essa é a essência do **ggplot2**.

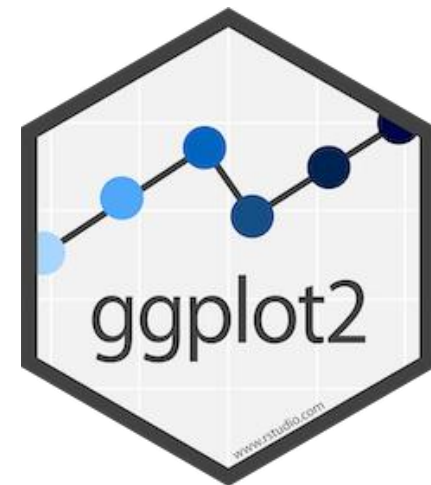




## ggplot2

Vantagens do ggplot2 em relação aos gráficos básicos do R:

- produz gráficos de melhor aparência;
- facilidade para ajustar o gráfico da forma desejada;
- estrutura padronizada das funções (aprendizado mais intuitivo);
- permite criar uma imensa gama de gráficos com poucas linhas de código.

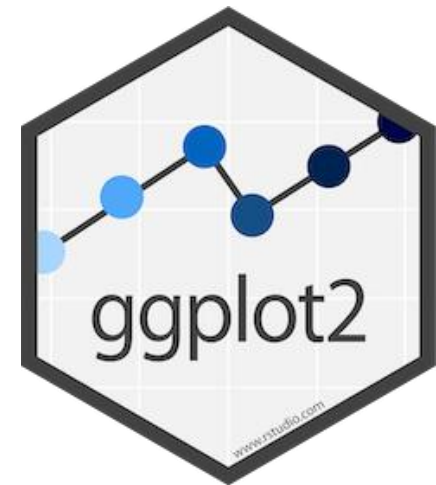


## ggplot2

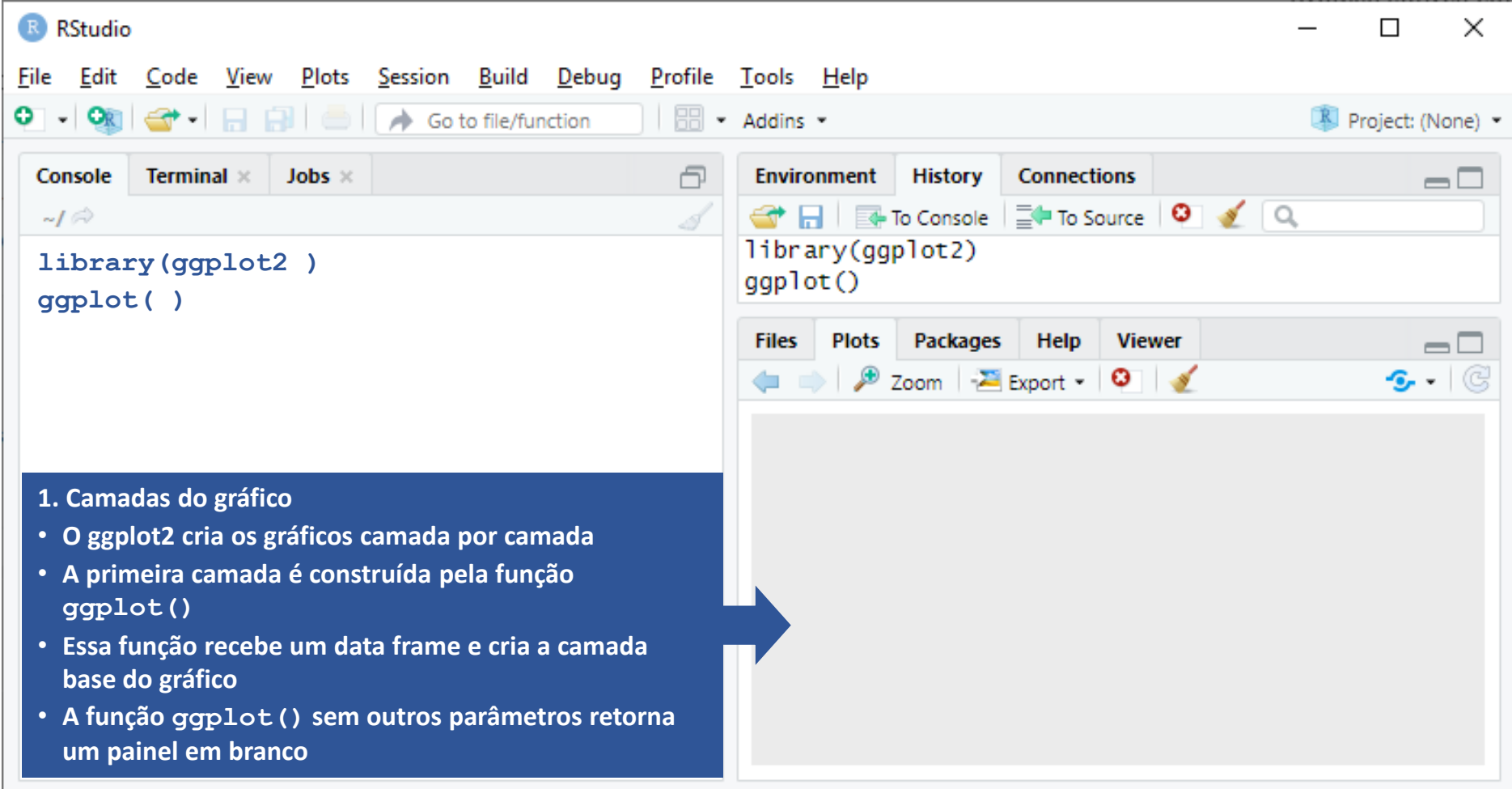
Instalar e carregar o pacote ggplot2:

```
install.packages("ggplot2")
```

```
library(ggplot2)
```



## ggplot2



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

Console Terminal x Jobs x

```
library(ggplot2 )
ggplot( )
```

Environment History Connections

library(ggplot2)  
ggplot()

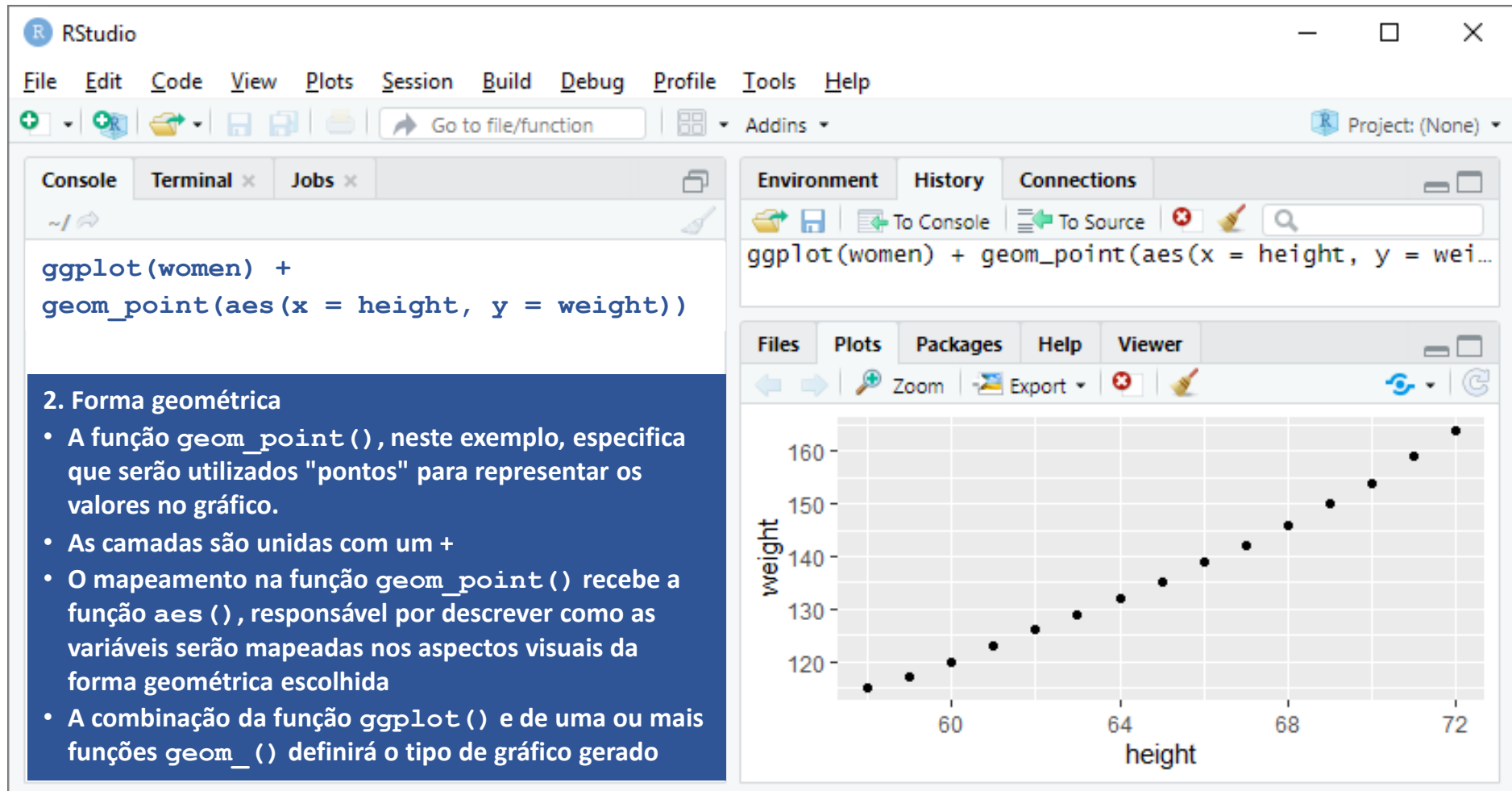
Files Plots Packages Help Viewer

Zoom Export

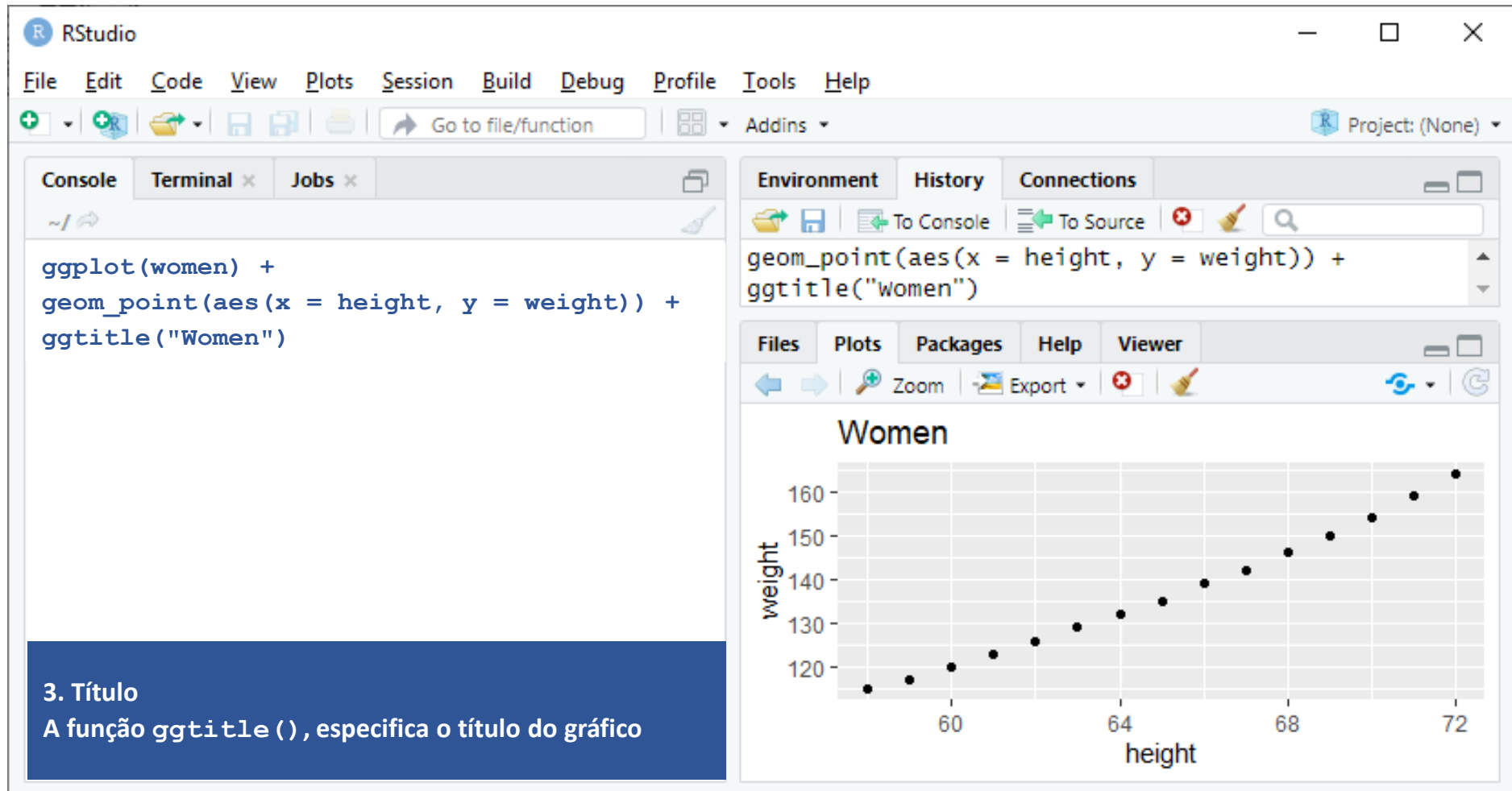
1. Camadas do gráfico

- O ggplot2 cria os gráficos camada por camada
- A primeira camada é construída pela função `ggplot()`
- Essa função recebe um data frame e cria a camada base do gráfico
- A função `ggplot()` sem outros parâmetros retorna um painel em branco

## ggplot2



## ggplot2



## ggplot2

### Atributos visuais

A função `aes()` - **aesthetics** - indica a relação entre os dados e cada aspecto visual do gráfico, como qual variável será representada no eixo x, qual será representada no eixo y, a cor e o tamanho dos componentes geométricos, etc.

Os aspectos visuais mais utilizados são:

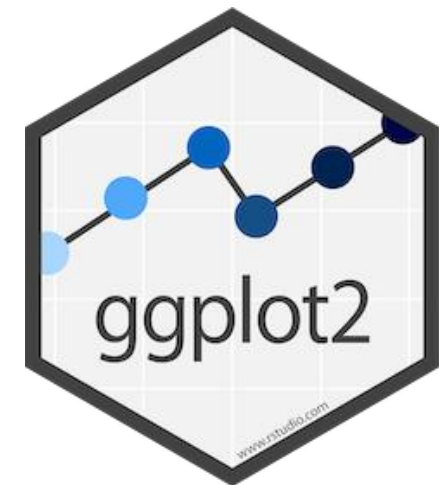
**color:** altera a cor de formas que não têm área (pontos e retas);

**fill:** altera a cor de formas com área (barras, caixas, densidades, áreas);

**size:** altera o tamanho de formas;

**type:** altera o tipo da forma, geralmente usada para pontos;

**linetype:** altera o tipo da linha.



## ggplot2

### Formas geométricas

Os atributos que iniciam com **geom\_** definem qual forma geométrica será utilizada para a visualização das observações.

As formas geométrica mais utilizadas são:

**geom\_point:** para gráficos de dispersão transformando pares (x,y) em pontos;

**geom\_line:** para linhas definidas por pares (x,y);

**geom\_abline:** para retas definidas por um intercepto e uma inclinação;

**geom\_hline:** para retas horizontais;

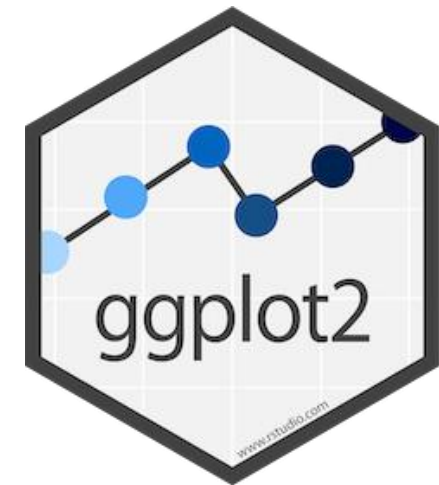
**geom\_bar:** para barras;

**geom\_histogram:** para histogramas;

**geom\_boxplot:** para boxplots;

**geom\_density:** para densidades;

**geom\_area:** para áreas.



## Modelos Probabilísticos Discretos



## Modelos Probabilísticos

Os **modelos de probabilidade** são construídos para **entender melhor fenômenos aleatórios**.

Conceitos importantes:

- **Espaço amostral** é o conjunto de todos os possíveis resultados do experimento (denotado pela letra grega  $\Omega$  (ômega)).
- Um espaço amostral é denominado **discreto** quando for **finito ou infinito enumerável** e é denominado **contínuo** quando for **infinito, formado por intervalos de números reais**.
- É denominado **evento** qualquer **subconjunto do espaço amostral**.

## Modelos Probabilísticos Discretos

Os principais modelos probabilísticos discretos são:

- Uniforme discreto
- Binomial
- Poisson
- Geométrico
- Hipergeométrico

Nas próximas aulas abordaremos dois modelos utilizados com muita frequência:

- Distribuição Binomial
- Distribuição de Poisson

## **Modelos Probabilísticos Discretos: Distribuição Binomial**

## Distribuição binomial

A distribuição binomial é a distribuição de probabilidade discreta do número de sucessos numa sequência de  $n$  tentativas que atende aos seguintes requisitos:

- Um número fixo ( $n$ ) de tentativas;
- Cada tentativa deve ser independente das outras;
- Cada tentativa tem apenas dois resultados possíveis, chamados de "sucesso" (o resultado de interesse) ou "fracasso" (distribuição de Bernoulli).

Há uma probabilidade constante ( $p$ ) de sucesso para cada tentativa, cujo complemento é a probabilidade  $(1 - p)$  de fracasso, muitas vezes representada por  $q = (1 - p)$ .

$$P(X = x) = \frac{n!}{x! (n - x)!} \cdot p^x \cdot q^{(n-x)}$$

**$n$** : número de tentativas

**$x$** : número de sucessos

**$p$** : probabilidade de sucesso a cada tentativa

## Distribuição binomial

Exemplo: Uma moeda é jogada 5 vezes. Qual é a probabilidade de resultar em cara 3 vezes?

X é uma variável binomial pois cada realização do evento resulta em duas possibilidades: cara ou coroa (sucesso ou fracasso);

Cada lançamento da moeda é independente do lançamento anterior;

Sair cara tem a mesma probabilidade 0.5 (50%) para todos lançamentos.

Identificando os parâmetros na fórmula:

**n** = 5 - número de tentativas

**p** = 0.5 - probabilidade de resultar cara

**x** = 3 - contagem dos sucessos.

$$P(X = x) = \frac{n!}{x! (n - x)!} \cdot p^x \cdot q^{(n-x)}$$

Substituindo na fórmula:

$$P(X = x) = \frac{5!}{3! (5 - 3)!} \cdot 0.5^3 \cdot 0.5^{(5-3)} = 0.3125 = 31.25\%$$

Resolvendo no R:

```
> dbinom(3,5,0.5)
```

```
[1] 0.3125
```

## **Modelos Probabilísticos Discretos: Distribuição de Poisson**

## Distribuição de Poisson

A distribuição de Poisson (publicada por Siméon Denis Poisson, em 1838) é uma distribuição discreta de probabilidade aplicável a ocorrências de um número de eventos em um intervalo específico.

Para aplicar a distribuição de Poisson, três aspectos devem ser observados:

- O experimento calcula quantas vezes um evento ocorre em um determinado intervalo (tempo, área, volume, etc.);
- A probabilidade do evento ocorrer é a mesma para cada intervalo;
- O número de ocorrências de um intervalo é independente do outro.

A distribuição Poisson tem apenas um parâmetro,  $\lambda$  (lambda) que é interpretado como uma taxa média de ocorrência do evento, e a probabilidade de ocorrerem exatamente  $x$  eventos é dada por:

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

onde:

$e$  (número de Euler)  $\approx 2,7183$

$\lambda > 0$

## Distribuição de Poisson

Exemplo: Um usuário recebe em média 5 mensagens em seu WhatsApp a cada 15 minutos. Qual a probabilidade desse usuário receber 8 mensagens no mesmo intervalo (15 minutos)?

$$\lambda = 5$$

$$n = 8$$

$$e = 2.7183$$

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{n!} \Rightarrow P(X = x) = \frac{5^8 e^{-5}}{8!} \Rightarrow 0.065278$$

Resolvendo no R:

```
> dpois(8,5)
```

```
[1] 0.06527804 #6.5278%
```



## Modelos Probabilísticos Contínuos

## Modelos Probabilísticos Contínuos

A distribuição contínua descreve as probabilidades dos possíveis valores de uma variável aleatória contínua.

Uma variável aleatória contínua apresenta um conjunto de valores possíveis (conhecidos como intervalos) que é infinito e incontável.

As probabilidades de variáveis aleatórias contínuas são definidas como a área sob a curva da sua distribuição.

Assim, apenas faixas de valores podem ter uma probabilidade diferente de zero. A probabilidade de que uma variável aleatória contínua seja exatamente igual a algum valor é sempre zero.

Portanto, é possível calcular a probabilidade de que um homem pesa entre 70 e 80 quilos. No entanto, a probabilidade de um homem pesar exatamente 70 quilos para a precisão infinita é zero. Porque a área sob a curva em um único ponto (que não tem nenhuma largura) é zero.

## Modelos Probabilísticos Contínuos

Alguns dos principais modelos probabilísticos contínuos são:

- Uniforme Contínuo
- Normal ou Gaussiana
- Qui-quadrado
- t de Student
- F de Snedecor
- Gama
- Beta
- Exponencial
- Weibull
- Logística

Nas próximas aulas abordaremos dois modelos utilizados com muita frequência:

- Distribuição Normal ou Gaussiana
- Distribuição Exponencial

## **Modelos Probabilísticos Contínuos: Distribuição Normal ou Gaussiana**

## Distribuição Normal (Gaussiana)

A distribuição normal é um dos principais modelos de probabilidade para **variáveis aleatórias contínuas**.

Exemplos de variáveis aleatórias contínuas: comprimento, peso, temperatura, pressão, tempo, etc.

Em uma coleta aleatória de dados de fontes independentes, geralmente observa-se que a distribuição dos dados é normal. Ou seja, ao traçar um gráfico com o valor da variável no eixo horizontal **x** e a contagem dos valores no eixo vertical **y**, obtemos uma curva em forma de sino. O **centro da curva** representa a **média** do conjunto de dados. No gráfico, cinquenta por cento dos valores ficam à esquerda da média e os outros cinquenta por cento ficam à direita. Isso é conhecido como distribuição normal.

Variáveis aleatórias com diferentes médias  $\mu$  e variâncias  $\sigma^2$  podem ser modeladas pelas funções de densidade de probabilidade normal. A média  $\mu$  determina o centro do gráfico em forma de sino e a variância  $\sigma^2$  determina a largura da distribuição.

## Distribuição Normal (Gaussiana)

Gerando valores aleatórios para uma distribuição normal:

A função `rnorm()` abaixo gera **dois conjuntos diferentes** com 10 valores aleatórios de uma distribuição normal:

```
> rnorm(10)
[1]  0.3599659 -1.5875803 -0.3890161  0.1307562  0.5890599
[6] -2.1539719  1.4859364 -0.4329219  0.6385853  1.9210244
> rnorm(10)
[1] -1.15513692 -2.11647932 -1.13449016  1.61930642 -1.02581667
[6] -0.45670775 -0.05491379 -0.58710610  0.34518977  1.49852742
```

A função `set.seed()` é utilizada para reproduzir os resultados dos geradores de números pseudo-aleatórios. Isto é importante, por exemplo, em simulações ou para ajustar um modelo de classificação com dois subconjuntos dos dados, um para treinar o modelo e outro para o testar. O valor do seed (semente) não é importante desde que a sua utilização seja consistente.

A função `rnorm()` abaixo gera **dois conjuntos iguais** com 10 valores aleatórios de uma distribuição normal:

```
> set.seed(128); rnorm(10)
[1]  0.596771824  0.482611999  1.664405236 -0.025995522  1.209026965
[6]  0.580352168 -0.458345286  0.004552681  0.209954522  1.511140204
> set.seed(128); rnorm(10)
[1]  0.596771824  0.482611999  1.664405236 -0.025995522  1.209026965
[6]  0.580352168 -0.458345286  0.004552681  0.209954522  1.511140204
```

## Distribuição Normal (Gaussiana)

R tem quatro funções integradas para gerar distribuição normal:

`dnorm(x, mean, sd)` – função de densidade ou probabilidade

`pnorm(x, mean, sd)` – função de densidade cumulativa

`qnorm(p, mean, sd)` – função quantil

`rnorm(n, mean, sd)` – função de desvios aleatórios

Onde:

**x** é um vetor de números.

**p** é um vetor de probabilidades.

**n** é o número de observações (tamanho da amostra).

**mean** é o valor médio dos dados da amostra. O valor padrão é zero.

**sd** é o desvio padrão. O valor padrão é 1.

## Distribuição Normal (Gaussiana)

**dnorm()** Fornece a altura da distribuição de probabilidade em cada ponto para uma determinada média e desvio padrão.

```
# Cria uma sequência de números entre -10 e +10 com incremento 0.1
```

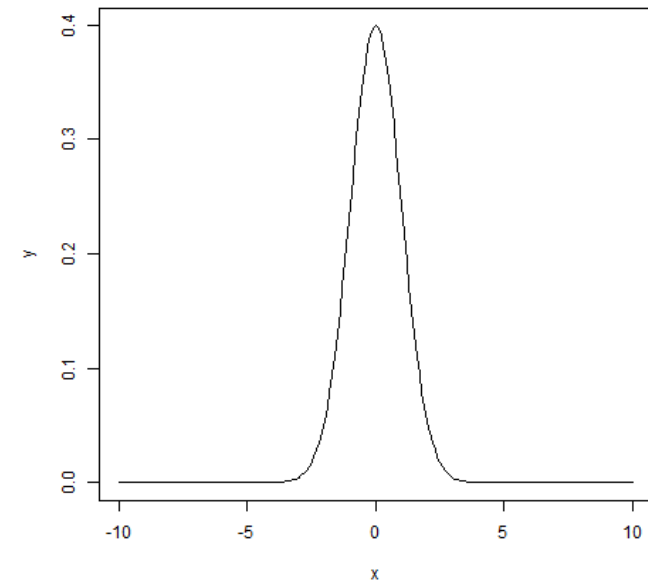
```
x <- seq(-10, 10, by = .1)
```

```
# Média = 0 e desvio padrão = 1
```

```
y <- dnorm(x, mean = 0, sd = 1)
```

```
# Gera o gráfico
```

```
plot(x,y, type="l")
```





## Distribuição Normal (Gaussiana)

**pnorm()** Fornece a probabilidade de um número aleatório normalmente distribuído ser menor que o valor de um determinado número. (Chamada também de "Função de distribuição cumulativa".)

```
# Cria uma sequência de números entre -10 e +10 com incremento 0.2
```

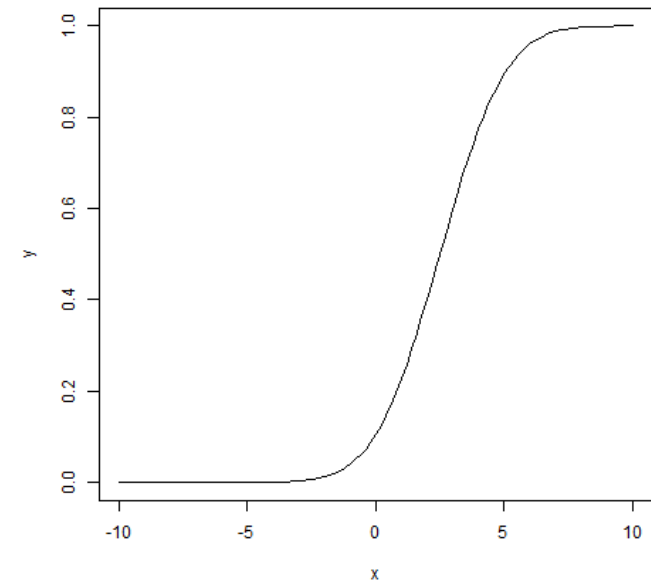
```
x <- seq(-10,10,by = .2)
```

```
# Média = 2.5 e desvio padrão = 2
```

```
y <- pnorm(x, mean = 2.5, sd = 2)
```

```
# Gera o gráfico
```

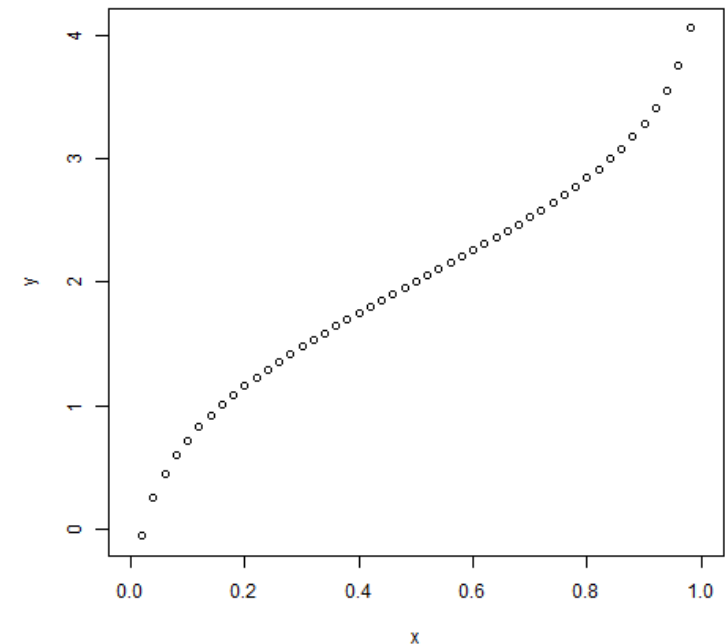
```
plot(x,y, type="l")
```



## Distribuição Normal (Gaussiana)

**qnorm()** A função quantil é simplesmente o inverso da função densidade cumulativa (pnorm), isto é, mapeia de probabilidades para valores.

```
# Cria uma sequência de valores de probabilidade com incremento 0.02  
x <- seq(0, 1, by = 0.02)  
  
# Média = 2 e desvio padrão = 1  
y <- qnorm(x, mean = 2, sd = 1)  
  
# Gera o gráfico  
plot(x,y)
```



## Distribuição Normal (Gaussiana)

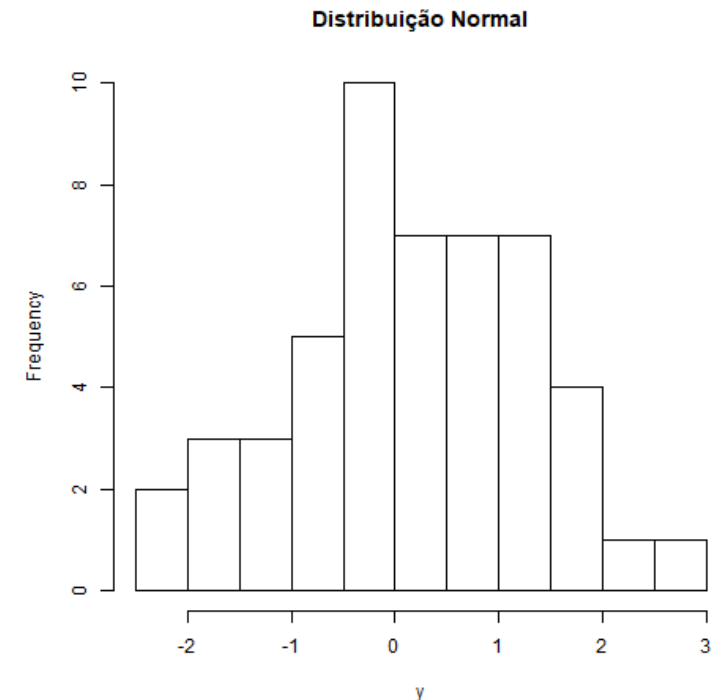
`rnorm()` Gera números aleatórios cuja distribuição é normal. Usa o tamanho da amostra como entrada e gera os números aleatórios. O histograma apresenta a distribuição dos números gerados.

```
# Cria uma amostra de com 50 números normalmente distribuídos
```

```
y <- rnorm(50)
```

```
# Gera o histograma para amostra criada
```

```
hist(y, main = "Distribuição Normal")
```

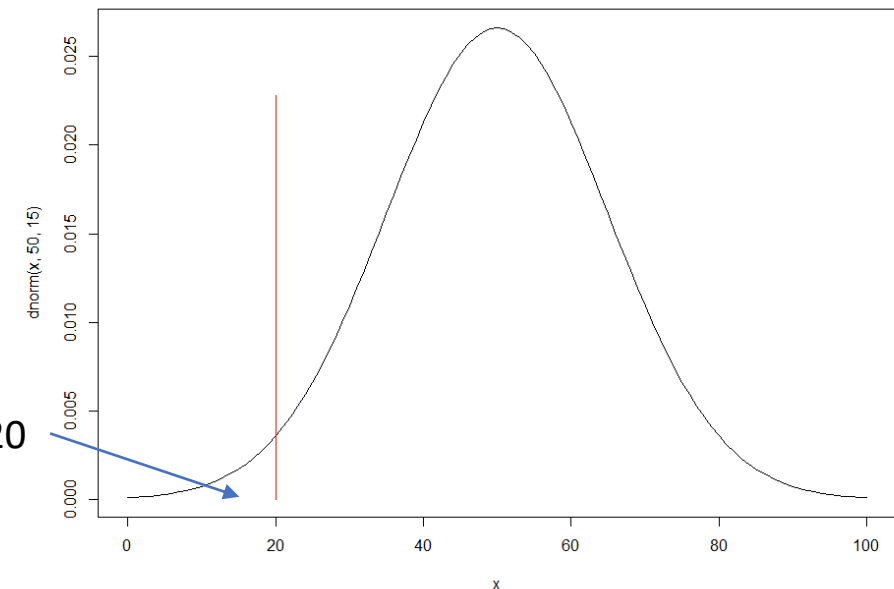


## Distribuição Normal (Gaussiana)

**Exemplo 1:** Qual a probabilidade de ocorrência de um valor **menor** que 20 em uma distribuição normal de média igual a 50 e desvio padrão igual a 15?

```
# x=20; mean=50; sd=15  
  
pnorm(20, 50, 15)  
  
[1] 0.02275013 # ≈ 2.28%  
  
curve(dnorm(x, 50, 15), 0, 100)  
  
lines(c(20, 20), c(0, 0.0228), col="red")
```

Probabilidade de valor < 20



## Distribuição Normal (Gaussiana)

**Exemplo 2:** Qual a probabilidade de ocorrência de um valor **maior** que 20 em uma distribuição normal de média igual a 50 e desvio padrão igual a 15?

```
1-pnorm(20, 50, 15)
```

```
[1] 0.9772499 # ≈ 97.72%
```

**Exemplo 3:** Qual o valor em que a probabilidade de encontrar valores menores que ele seja de 80% em uma distribuição normal de média igual a 50 e desvio padrão igual a 15?

```
qnorm(0.8, 50, 15)
```

```
[1] 62.62432
```

## **Modelos Probabilísticos Contínuos: Distribuição Exponencial**

## Distribuição Exponencial

A distribuição exponencial, muito utilizada em engenharia, descreve o tempo de chegada de uma sequência de eventos independentes recorrentes aleatoriamente.

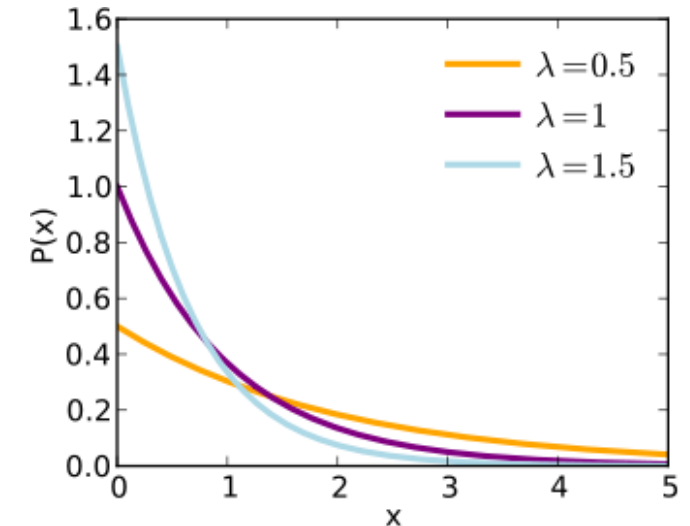
O modelo ajusta-se bem a dados de tempo para ocorrência de um evento. Exemplo: tempo para atendimento de uma chamada.

É uma distribuição também muito utilizada na prática para modelar tempo de falha de objetos. Por exemplo, pode ser usada para modelar o tempo que demora até uma lâmpada falhar.

Se  $\mu$  é o tempo médio de espera para a próxima recorrência do evento, sua função de densidade de probabilidade é:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

em que  $\lambda$  é o parâmetro de taxa da distribuição e deve satisfazer  $\lambda > 0$ . Neste caso,  $\lambda$  é o tempo médio de vida e  $x$  é um tempo de falha. O parâmetro deve ter a mesma unidade do tempo da falha  $x$ . Isto é, se  $x$  é medido em horas,  $\lambda$  também será medido em horas.



## Distribuição Exponencial

O tempo médio de checkout de um caixa de supermercado é de três minutos. Qual a probabilidade do checkout de um cliente ser concluído pelo caixa em menos de dois minutos?

Tomando como base o enunciado, verifica-se que a taxa de processamento do checkout é igual a um dividido pelo tempo médio de conclusão do checkout. Portanto, a taxa de processamento é de  $1/3$  checkouts por minuto. Em seguida, aplica-se a função `pexp` da distribuição exponencial com taxa =  $1/3$ .

```
> pexp(2, rate=1/3)
[1] 0.4865829
```

A probabilidade de concluir um checkout em menos de dois minutos no caixa é de 48.7%.

A distribuição exponencial é a única distribuição contínua que possui **perda de memória**.

Portanto, suponha que o cliente já esperou um minuto para conclusão do checkout. Isto significa que a probabilidade de ser atendido no próximo minuto seja maior do que no primeiro minuto? A resposta é não. Não importa o quanto tempo o cliente tenha esperado. Esta propriedade da distribuição exponencial é chamada de perda de memória.



## Regressão Linear Simples e Correlação

## Regressão Linear Simples

## Regressão linear simples

A regressão linear simples constitui uma tentativa de estabelecer uma equação matemática linear (reta) com apenas uma variável dependente que descreva o relacionamento entre duas variáveis.

A reta de regressão (equação linear) apresenta como principais características:

- **Coeficiente angular** ( $\alpha$ ) da reta, dado pela tangente da reta;
- **Coeficiente linear** ( $\beta$ ), a cota da reta em determinado ponto (o valor de  $y$  quando  $x$  for igual a zero).

$$y = \alpha + \beta x + \varepsilon$$

Onde:

**x**: variável independente que busca explicar  $y$

**y**: variável dependente a ser prevista

$\varepsilon$ : erro que corresponde ao desvio entre o valor real e o aproximado (pela reta) de  $y$

## Equação linear

A equação linear pode ser obtida no R através da função `lm()` que calcula a regressão linear simples.

A regressão linear simples é obtida por meio do seguinte comando:

`lm(y~x, data)`

**lm:** linear model

**y~x:** y depende de x

**data:** conjunto de dados (data.frame) valores das "colunas" que correspondem a x e y

## Equação linear

Exemplo:

```
> x <- c(205,225,305,380,560,600,685,735)
> y <- c(15,20,21,22,22,25,28,28)
> dados <- data.frame(x,y) #cria um data.frame
> regressão <- lm(y~x,data=dados) #ou regressão <- lm(y~x)
> regressão
```

Call:

```
lm(formula = y ~ x, data = dados)
```

Coefficients:

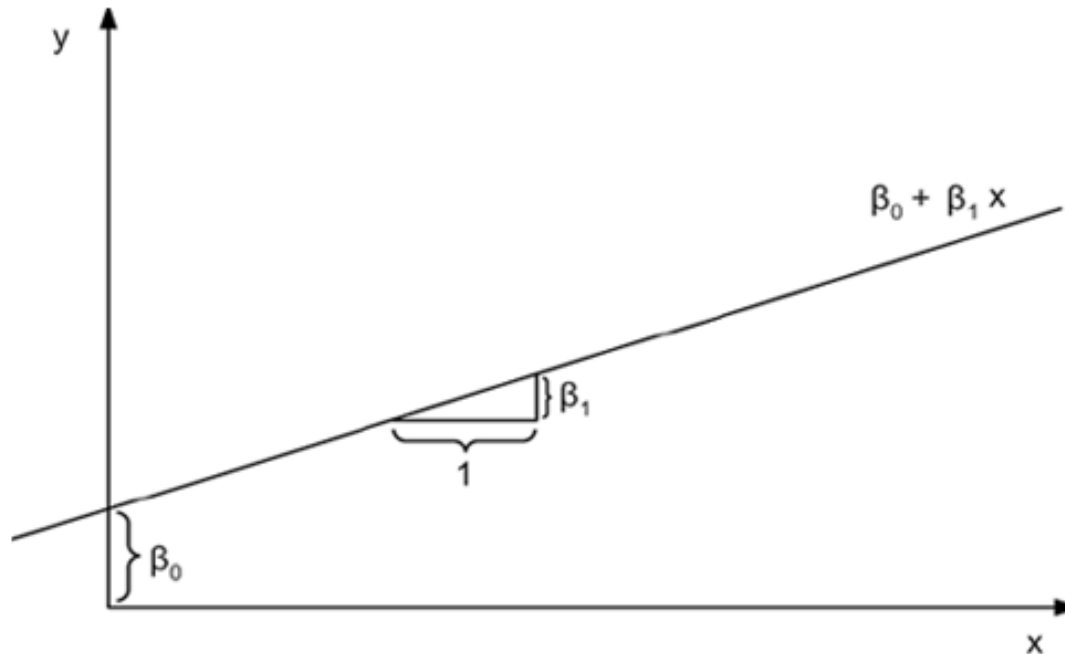
(Intercept)	x
13.85362	0.01899

Equação da reta:  $y = 13.85362 + 0.01899x$

## Equação linear

O parâmetro  $\beta_0$  é chamado intercepto ou coeficiente linear e representa o ponto em que a reta corta o eixo dos y's, quando  $x = 0$ .

O parâmetro  $\beta_1$  representa a inclinação da reta e é chamado de coeficiente de regressão ou coeficiente angular.



## Equação linear

**regressão** é uma lista com os seguintes elementos:

```
> names(regressão)
[1] "coefficients"  "residuals"    "effects"      "rank"
[5] "fitted.values" "assign"        "qr"           "df.residual"
[9] "xlevels"       "call"         "terms"        "model"
```

Valores preditos da variável resposta para cada elemento da amostra (previsão):

```
> regressão$fitted.values
      1      2      3      4      5      6      7      8
17.74673 18.12655 19.64582 21.07013 24.48847 25.24811 26.86233 27.81187
```

Erro ou resíduos (valor observado - valor predito) para cada ponto da amostra:

```
> regressão$residuals
      1      2      3      4      5      6      7      8
-2.7467348  1.8734490  1.3541839  0.9298729 -2.4884736 -0.2481061  1.1376747  0.1881341
```

Estimativa dos coeficientes da regressão:

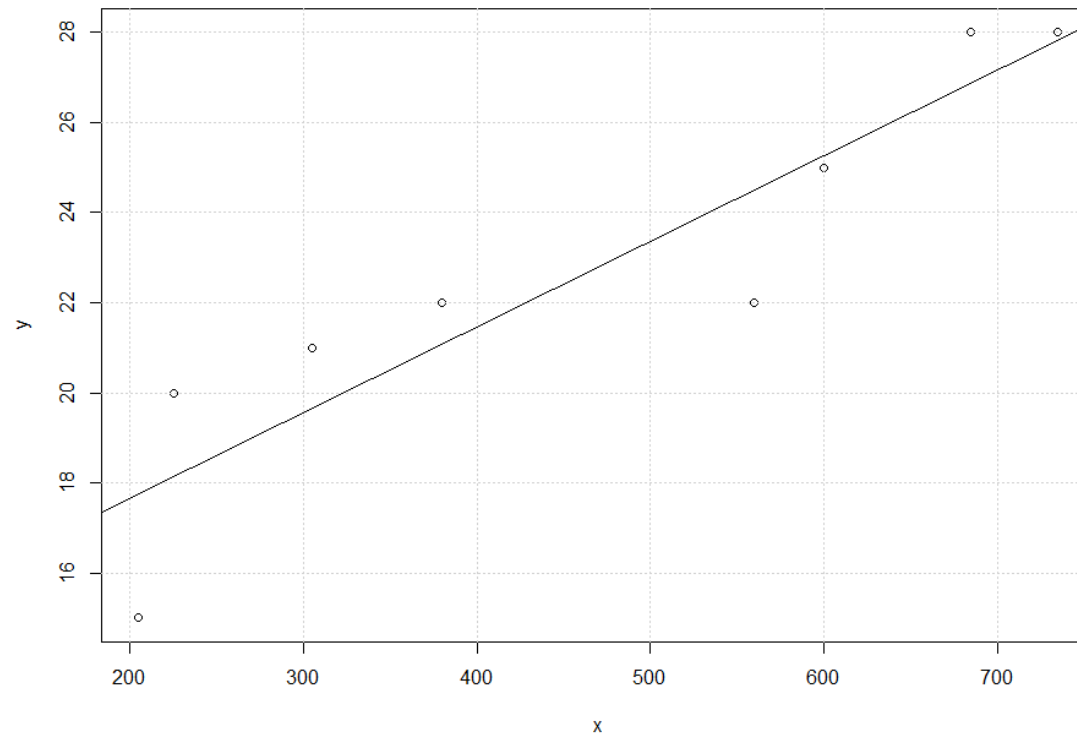
```
> regressão$coefficients
(Intercept)      x
13.85361800  0.01899081
```

## Equação linear

```
z = plot(x,y)
```

```
grid(z) #aplicando grid ao gráfico
```

```
abline(regressão)
```





## Correlação

## Coeficiente de correlação (r)

O gráfico de dispersão pode indicar se a correlação linear é positiva, negativa ou a inexistência de correlação.

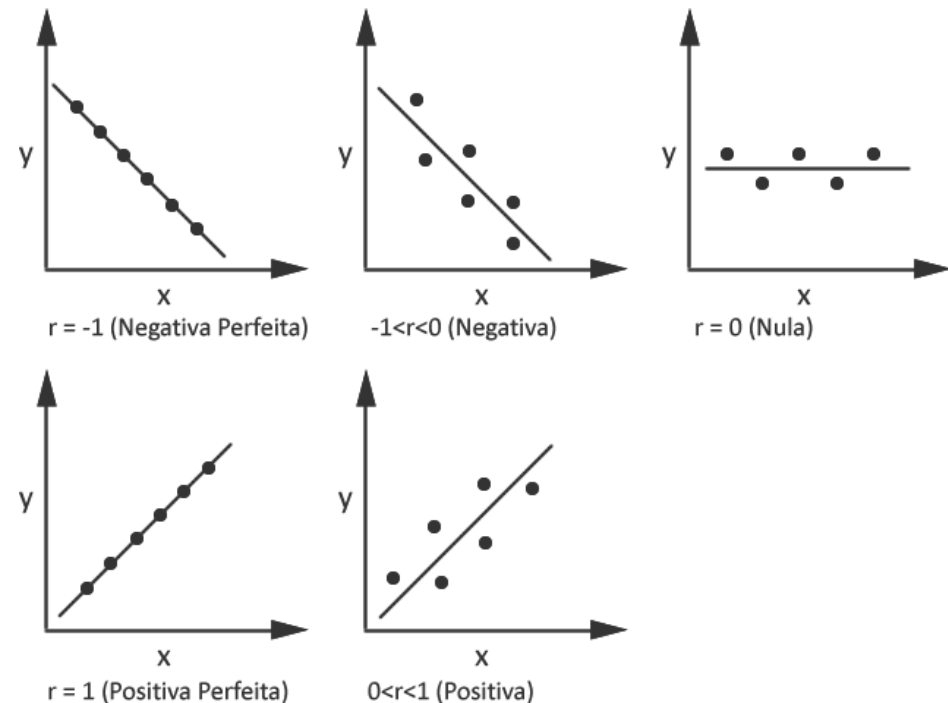
Através do coeficiente de correlação pode-se identificar o tipo de correlação, conforme a seguinte tabela:

r	Tipo	Descrição
$r = -1$	Perfeita Negativa	Os pontos estão perfeitamente alinhados, porém os valores crescentes de x associados a valores decrescentes de y
$-1 < r < 0$	Negativa	Os valores crescentes de x estão associados a valores decrescentes de y
$r = 0$	Não há correlação	Os valores de x e y ocorrem independentemente
$0 < r < 1$	Positiva	Os valores crescentes de x estão associados a valores crescentes de y
$r = 1$	Positiva Perfeita	Os pontos estão perfeitamente alinhados e os valores crescentes de x estão associados a valores crescentes de y

## Coeficiente de correlação linear (r)

O coeficiente de correlação demonstra que a correlação será tanto mais forte quanto mais próximo o coeficiente estiver de  $-1$  ou  $+1$ , e será tanto mais fraca quanto mais próximo o coeficiente estiver de zero.

r	Tipo
-1	Negativa perfeita
$-1 < r < -0.7$	Negativa forte
$-0.7 < r < -0.5$	Negativa moderada
$-0.5 < r < 0$	Negativa fraca
0	Nula
$0 < r < 0.5$	Positiva fraca
$0.5 < r < 0.7$	Positiva moderada
$0.7 < r < 1$	Positiva forte
$r = 1$	Positiva perfeita



## Coeficiente de correlação linear (r)

Exemplo:

```
> x <- c(205,225,305,380,560,600,685,735)
> y <- c(15,20,21,22,22,25,28,28)
> cor(x,y)
[1] 0.9155359
```

$r = +0.916$  indica uma correlação **positiva forte**, ou seja, a variável dependente **y** cresce quase na mesma proporção que a variável independente **x**.

## Coeficiente de determinação ( $r^2$ )

O coeficiente de determinação explica o grau de ajuste do modelo, isto é, o percentual de variação de  $y$  que é explicado pela variabilidade de  $x$ . Seu valor varia de 0 a 1.

O valor do coeficiente de determinação corresponde ao valor do coeficiente de variação elevado ao quadrado.

Exemplo:

```
> x <- c(205,225,305,380,560,600,685,735)
> y <- c(15,20,21,22,22,25,28,28)
> cor(x,y)^2
[1] 0.8382059 #aproximadamente 84%
```

**84%** da variação da variável dependente  $y$  é explicada pela variável independente  $x$ . Os outros 16% possuem causas aleatórias desconhecidas (independentes de  $x$ ).

## Coeficiente de correlação linear (r)

```
> x <- c(-1,2,3,4)
> y <- c(-1,5,7,9)
> cor(x,y)
[1] 1
```

$r = +1$  indica uma **correlação perfeita positiva** entre as duas variáveis  $x$  e  $y$ .

$$y = \alpha + \beta x + \varepsilon$$

$$\beta = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

$$\alpha = \frac{\sum y - \beta \sum x}{n}$$

$$n = 4$$

x	y	xy	x <sup>2</sup>	y <sup>2</sup>
-1	-1	1	1	1
2	5	10	4	25
3	7	21	9	49
4	9	36	16	81
$\Sigma$ 8	20	68	30	156

$$\beta = \frac{4 \cdot 68 - 8 \cdot 20}{4 \cdot 30 - (8)^2} = \frac{112}{56} = 2$$

$$\alpha = \frac{20 - 2 \cdot 8}{4} = \frac{4}{4} = 1 \quad \leftarrow$$

$$y = 1 + 2x + 0$$

# **Testes de Hipóteses**

## Testes de Hipóteses

Usar dados de uma amostra para tentar dizer algo sobre a população que lhes deu origem é um importante ramo da estatística, chamado **inferência estatística**, e o teste de hipótese é uma das suas principais ferramentas.

Um **teste de hipótese** é uma técnica estatística cujo **objetivo é verificar se uma dada amostra de dados é, ou não, compatível com uma hipótese feita sobre a população que lhe deu origem**.

O teste de hipótese coloca lado-a-lado duas hipóteses sobre a população que deu origem à amostra de dados: uma hipótese inicial, ou hipótese nula, e uma hipótese alternativa, designadas por  $H_0$  e  $H_1$  respectivamente.

É retirada uma amostra da população, cuja informação será tratada para encontrar evidência para se rejeitar, ou não a hipótese nula.

Caso ocorra a rejeição da hipótese nula, deve-se considerar, à partir desta constatação, a hipótese alternativa.



## Teste t de Student

## Teste t de Student

O test t de Student foi introduzido em 1908 por William Sealy Gosset, matemático e estatístico que trabalhava na cervejaria Guinness, em Dublin, na Irlanda.

Gosset desenvolveu o teste t, a princípio, para monitorar a qualidade da cerveja.

O teste t pode ser aplicado quando o tamanho da amostra é pequeno. Isso permite que se façam inferências usando um menor número de elementos, reduzindo os custos da pesquisa.

O uso de métodos estatísticos na fabricação da cerveja era considerado um segredo industrial. Portanto, quando Gosset publicou o artigo sobre o teste t na revista acadêmica Biometrika em 1908, teve de usar um pseudônimo "Student" e, por isso, o teste t passou a ser conhecido como **teste t de Student**.

## Teste t de Student

A pluviosidade média em uma região é: 70.6

O vetor `pluv` apresenta mês-a-mês a pluviosidade em um determinado ano com relação a uma determinada região:

`pluv = c(110,100,60,80,70,18,17,17,42,89,108,143)`

Observando-se que a média mensal de pluviosidade é de 70.6 mm, terá o ano "pluv" sido excepcionalmente chuvoso?

A hipótese nula contém sempre uma igualdade e a hipótese alternativa contém sempre uma desigualdade.

Testes com a alternativa diferente são denominados bilaterais. Testes com alternativa menor são designados unilaterais esquerdos, e com alternativa maior são designados unilaterais direitos.

As hipóteses podem ser formalmente descritas assim:

$H_0 : \mu = \mu_0$  vs  $H_1 : \mu \neq \mu_0 \Rightarrow H_0 : \mu = 70.6$  vs  $H_1 : \mu \neq 70.6$  (Bilateral)

$H_0 : \mu = \mu_0$  vs  $H_1 : \mu < \mu_0 \Rightarrow H_0 : \mu = 70.6$  vs  $H_1 : \mu < 70.6$  (Unilateral esquerdo)

$H_0 : \mu = \mu_0$  vs  $H_1 : \mu > \mu_0 \Rightarrow H_0 : \mu = 70.6$  vs  $H_1 : \mu > 70.6$  (Unilateral direito)

## Teste t de Student

Utilizando o R para resolver  $H_0 : \mu = 70.6$  vs  $H_1 : \mu \neq 70.6$  (teste bilateral)

```
t.test(pluv, mu = 70.6)
```

```
data:  pluv
t = 0.047327, df = 11, p-value = 0.9631
alternative hypothesis: true mean is not equal to 70.6
95 percent confidence interval:
 44.81350 97.51983
sample estimates:
mean of x
 71.16667
```

O **p-value** (valor de prova) de **96.3%** indica "não rejeição" de  $H_0$ , ou seja, não há razão para rejeitar a hipótese de que o ano "pluv" tenha sido um ano de pluviosidade normal.

Os limites entre não rejeição e rejeição situa-se entre 1% e 10% de **p-value**. Ou seja, para p-values menores que 1% rejeita-se a hipótese nula. Para valores maiores que 10% não se costuma rejeitar. O intervalo entre 1 e 10% é uma "zona cinzenta", pois o julgamento fica à critério do pesquisador.

## Teste t de Student

Utilizando o R para resolver  $H_0 : \mu = 70.6$  vs  $H_1 : \mu < 70.6$  (teste unilateral esquerdo)

```
t.test(pluv, mu = 70.6, alternative = "less")
```

```
data:  pluv
t = 0.047327, df = 11, p-value = 0.5184
alternative hypothesis: true mean is less than 70.6
95 percent confidence interval:
 -Inf 92.66942
sample estimates:
mean of x
 71.16667
```

O **p-value** (valor de prova) de **51.8%** indica "não rejeição" de  $H_0$ .

## Teste t de Student

Utilizando o R para resolver  $H_0 : \mu = 70.6$  vs  $H_1 : \mu > 70.6$  (teste unilateral direito)

```
t.test(pluv, mu = 70.6, alternative = "greater")
```

```
data:  pluv
t = 0.047327, df = 11, p-value = 0.4816
alternative hypothesis: true mean is greater than 70.6
95 percent confidence interval:
 49.66391      Inf
sample estimates:
mean of x
 71.16667
```

O **p-value** (valor de prova) de **48.2%** indica "não rejeição" de  $H_0$ .

## Teste de Shapiro-Wilk

Existem alguns pressupostos teóricos que devem ser observados para que o resultado (p-value) seja confiável.

Para amostras pequenas,  $N \leq 30$ , a população em estudo deve apresentar uma distribuição normal (gaussiana). Distribuição normal é uma distribuição contínua, frequentemente associada a características físicas (peso, altura, temperatura, etc.).

Para amostras grandes,  $N > 30$ , não há restrições em relação à população.

A amostra "pluv" é pequena ( $N = 12$ ) e para verificar se é uma distribuição normal deve-se realizar um teste de hipóteses preliminar, o teste de Shapiro-Wilk, disponível no R.

## Teste de Shapiro-Wilk

O comando R para executar o teste Shapiro-Wilk é **shapiro.test**:

```
> shapiro.test(pluv)
```

```
Shapiro-Wilk normality test
```

```
data:  pluv  
W = 0.93775, p-value = 0.4694
```

No teste Shapiro-Wilk a hipótese nula é: "a distribuição 'pluv' é normal" e a alternativa é "a distribuição 'pluv' não é normal".

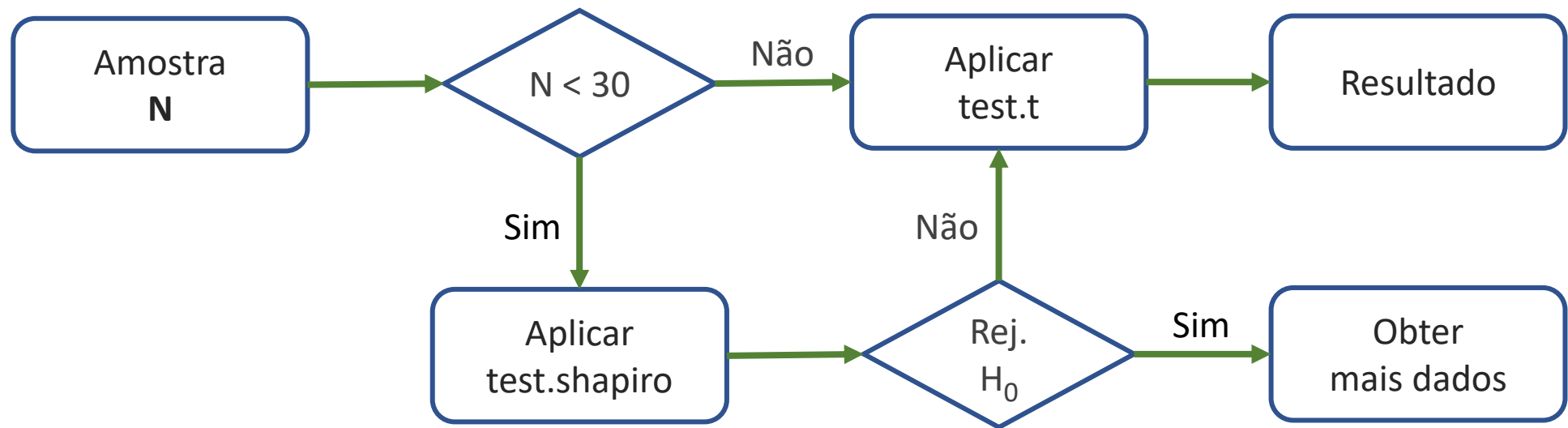
O p-value 46.9% indica que não há evidência para rejeitar a hipótese nula, concluindo-se, portanto, que "pluv" provém de uma distribuição normal.

Desta forma os resultados obtidos anteriormente são válidos.

Caso o p-value do teste Shapiro-Wilk apresentasse valores baixos, (por exemplo, < 1%) seria necessário obter mais dados de "pluv" para chegar a uma amostra com pelo menos 30 valores.



## Teste de Shapiro-Wilk



## ANOVA - Análise de Variância

A análise de variância (ANOVA - analysis of variance) é um **teste de hipóteses aplicado para comparar médias de mais de duas amostras**. A técnica foi desenvolvida inicialmente para fins agrícolas.

Há diversos tipos de ANOVA. A seguir será aplicado a ANOVA de um fator de efeitos fixos (one-way fixed effects analysis of variance).

Pressupostos a serem observados:

1. Amostras devem ser aleatórias e mutuamente independentes;
2. Populações normalmente distribuída (aplicar o teste de Shapiro-Wilk para  $N < 30$ );
3. Populações apresentam variância homogênea (aplicar o teste de Bartlett).

## ANOVA - Análise de Variância

Três áreas agrícolas foram submetidas a três tratamentos diferentes com adubos identificados por A, B e C. Foram recolhidas 5 colheitas de cada área, obtendo-se as seguintes produtividades por hectare para o mesmo produto:

```
A <- c(14,13,20,15,13)
B <- c(13,14,13,18,15)
C <- c(19,16,17,20,19)
```

Média dos três grupos:

```
> mean(A)
[1] 15
> mean(B)
[1] 14.6
> mean(C)
[1] 18.2
```

A diferença entre as médias amostrais é estatisticamente relevante? ou (em linguagem matemática): Os três grupos têm a mesma média de população, ou pelo menos um grupo tem média de população diferente dos outros?

## ANOVA - Análise de Variância

### 1. Amostras devem ser aleatórias e mutuamente independentes.

Deve-se confirmar que não há interferência de um grupo no outro. Por exemplo: se as áreas fossem contíguas e a fertilização ocorresse por polinização, poderia haver mistura genética entre os grupos, invalidando a análise.

O pressuposto de independência não pode ser verificado por testes estatísticos preliminares.

No exemplo, assume-se que não há problemas de contaminação de uma área para outra e, portanto há independência.

## ANOVA - Análise de Variância

### 2. Populações normalmente distribuída (aplicar o teste de Shapiro-Wilk para $N < 30$ )

```
> shapiro.test(A)
Shapiro-Wilk normality test
data:  A
W = 0.77559, p-value = 0.0505
```

```
> shapiro.test(B)
Shapiro-Wilk normality test
data:  B
W = 0.84215, p-value = 0.171
```

```
> shapiro.test(C)
Shapiro-Wilk normality test
data:  C
W = 0.91367, p-value = 0.4899
```

O grupo A (p-value 5.05%) está no borderline da normalidade, mas ainda assim pode ser considerado como seguindo uma distribuição normal.

## ANOVA - Análise de Variância

### 3. Populações têm mesma variância (aplicar o teste de Bartlett).

No teste de Bartlett aceitar a hipótese  $H_0$  indica que os grupos têm variância homogênea. Aceitar  $H_1$  indica que não há variância homogênea (pelo menos um grupo não mantém variância homogênea em relação aos outros).

```
> bartlett.test(list(A,B,C))
```

```
    Bartlett test of homogeneity of variances
```

```
data:  list(A, B, C)
```

```
Bartlett's K-squared = 1.2051, df = 2, p-value = 0.5474
```

O p-value 54.7% indica não rejeição de  $H_0$ , validando o pressuposto da homogeneidade da variância.

Validados os três pressupostos, pode-se preparar e aplicar, a seguir, o **teste ANOVA de um fator**.

## ANOVA - Análise de Variância

### Formatação dos dados

Para aplicar o teste ANOVA no R é preciso agregar os dados de produtividade das três áreas, A, B e C, em um único vetor.

```
> prod <- c(A,B,C)
> prod
[1] 14 13 20 15 13 13 14 13 18 15 19 16 17 20 19
```

A seguir, cria-se o vetor com os elementos que correspondem aos grupos (áreas agrícolas).

```
> grupos <- c(rep("A",5),rep("B",5),rep("C",5))
> grupos
[1] "A" "A" "A" "A" "A" "B" "B" "B" "B" "B" "C" "C" "C" "C" "C"
```

O próximo passo será juntar os dois vetores em um único data frame.

## ANOVA - Análise de Variância

### Formatação dos dados (continuação)

O data frame apresentado a seguir junta os dois vetores: **prod** e **grupos**.

```
> df <- data.frame(prod, grupos)
> df
```

	prod	grupos
1	14	A
2	13	A
3	20	A
4	15	A
5	13	A
6	13	B
7	14	B
8	13	B
9	18	B
10	15	B
11	19	C
12	16	C
13	17	C
14	20	C
15	19	C

Esta visualização é importante para verificar se os dados foram inseridos corretamente.



## ANOVA - Análise de Variância

### Aplicando o teste ANOVA

O comando a seguir executa o ANOVA sobre um modelo linear (**lm**) em que cada valor de produtividade (**prod**) está associado ao respectivo grupo (**grupos**).

```
> anova(lm(prod ~ grupos))
```

```
Analysis of Variance Table
```

```
Response: prod
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
grupos	2	38.933	19.4667	3.7677	0.05372
Residuals	12	62.000	5.1667		

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

O p-value do ANOVA é o valor da coluna  $\text{Pr(>F)} = 0.05372$ , aproximadamente 5.3%.

É um valor no limite da rejeição de  $H_0$ , isto é, existe a suspeita de que pelo menos um tratamento tenha gerado diferença de produtividade.

Qual ou quais grupos são responsáveis pelas diferenças?

Um teste de comparação múltipla **post-hoc** tentará identificar o(s) grupo(s) responsáveis pela "quase" rejeição de  $H_0$  no ANOVA.

## Tukey HSD - Teste de comparações múltiplas

### Aplicando o teste Tukey HSD

O R oferece recursos para realizar a maioria dos testes de comparações múltiplas. Um dos mais utilizados é o Tukey HSD (Honest Significant Difference).

O TukeyHSD atua sobre objetos da classe aov (Analysis Of Variance), portanto é preciso usá-la no comando a seguir:

```
> TukeyHSD(aov(lm(prod ~ grupos)))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = lm(prod ~ grupos))

$grupos
      diff      lwr      upr    p adj
B-A -0.4 -4.2352956  3.435296 0.9583671
C-A  3.2 -0.6352956  7.035296 0.1068512
C-B  3.6 -0.2352956  7.435296 0.0665354
```

O p-value na comparação entre os grupos B-A (95.8%) indica clara não rejeição de  $H_0$ .

O p-value na comparação entre os grupos C-A (10.7%) não é suficientemente forte para rejeição de  $H_0$ .

O p-value na comparação entre os grupos C-B (6.7%) está no limite para não rejeição de  $H_0$ .

Se houve um tratamento que impactou na produtividade este foi aplicado no grupo C. O p-value próximo do limite de rejeição indica que é recomendável recolher mais amostras para testes.

## **Contato:**

**Prof. MSc. Marcos Alexandruk**  
**E-mail: [alexandruk@uninove.br](mailto:alexandruk@uninove.br)**