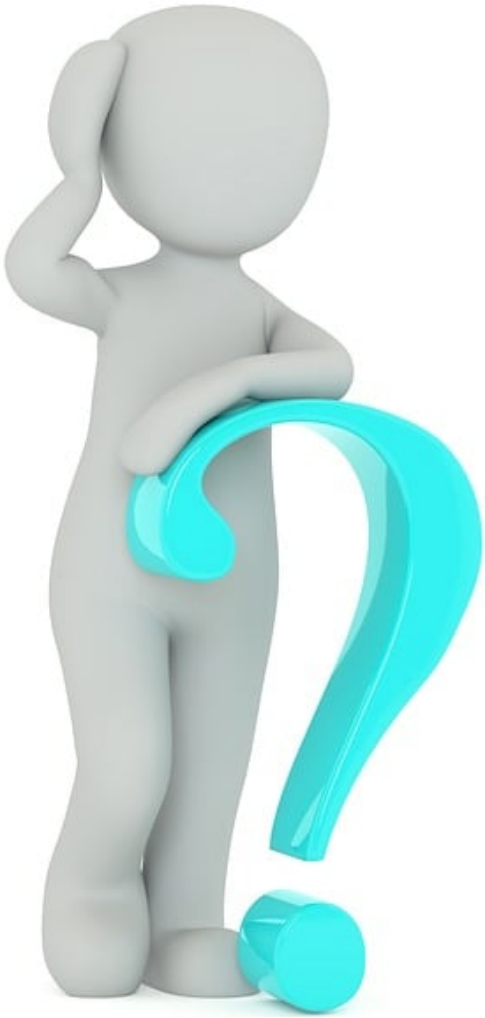
Introdução ao R

Daniel C. Mota e Kleber G. Abitante

Dúvidas?

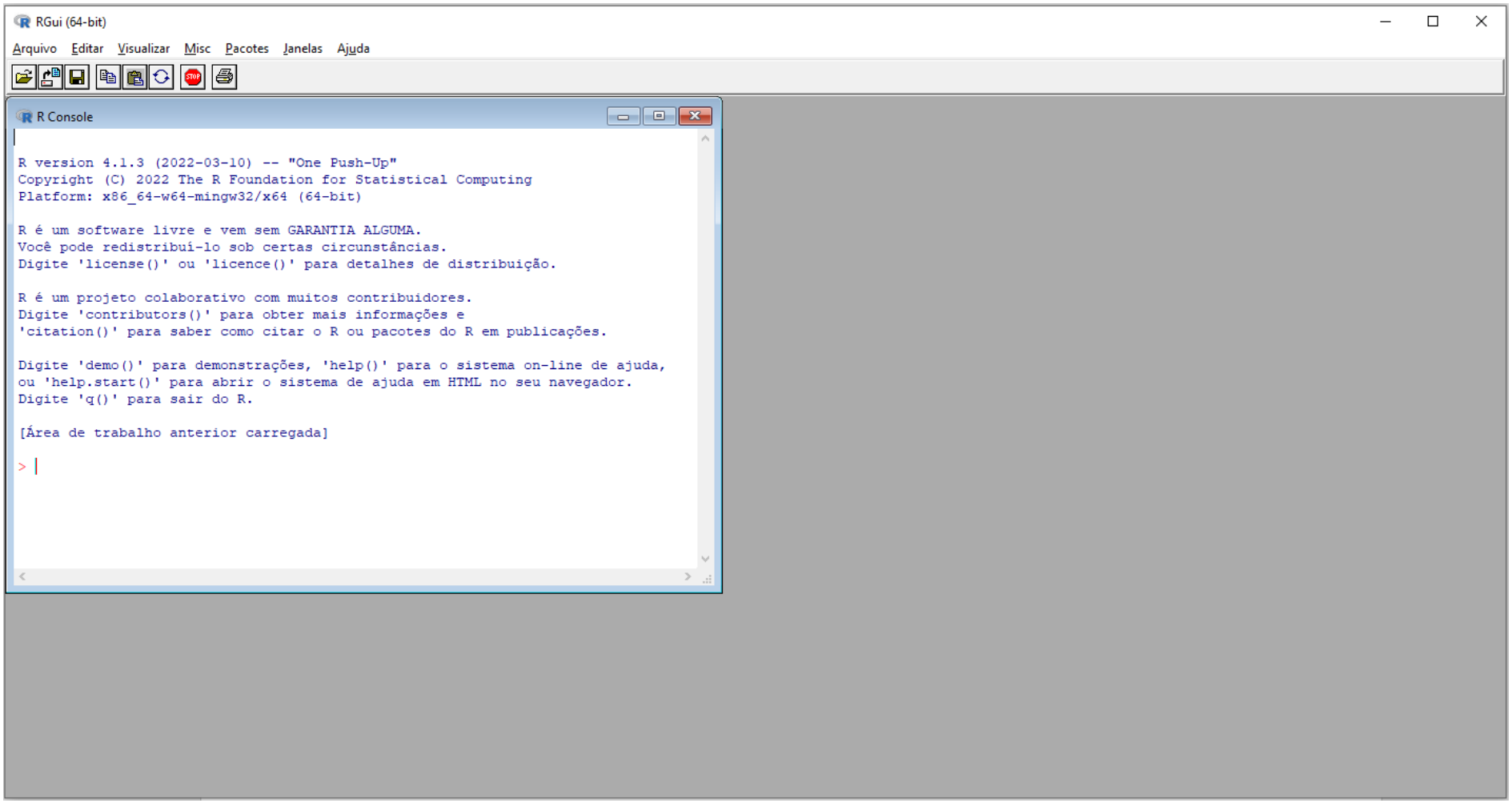


Breve histórico[1]

- Em 1976, foi criada a linguagem S por John Chambers e outros no antigo Bell Telephone Laboratories;
- Esta linguagem era voltada para análise de dados e buscava atender usuários e desenvolvedores;
- A linguagem S tinha uma limitação: estava disponível somente em um pacote comercial, chamado S-PLUS;
- Em 1991, a linguagem R foi criada por Ross Ihaka e Robert Gentleman no Departamento de Estatística da Universidade de Auckland, na Nova Zelândia, utilizando como base a linguagem S;
- Em 1995, o R aderiu à *GNU General Public License* se tornando um software livre; e
- Finalmente, em 2000, foi lançada a versão 1.0.0 para o público.

[1] PENG, ROGER D. R programming for data science. 2022. Disponível em: https://bookdown.org/rdpeng/rprogdatascience/. Acesso em: 23 fev. 2023

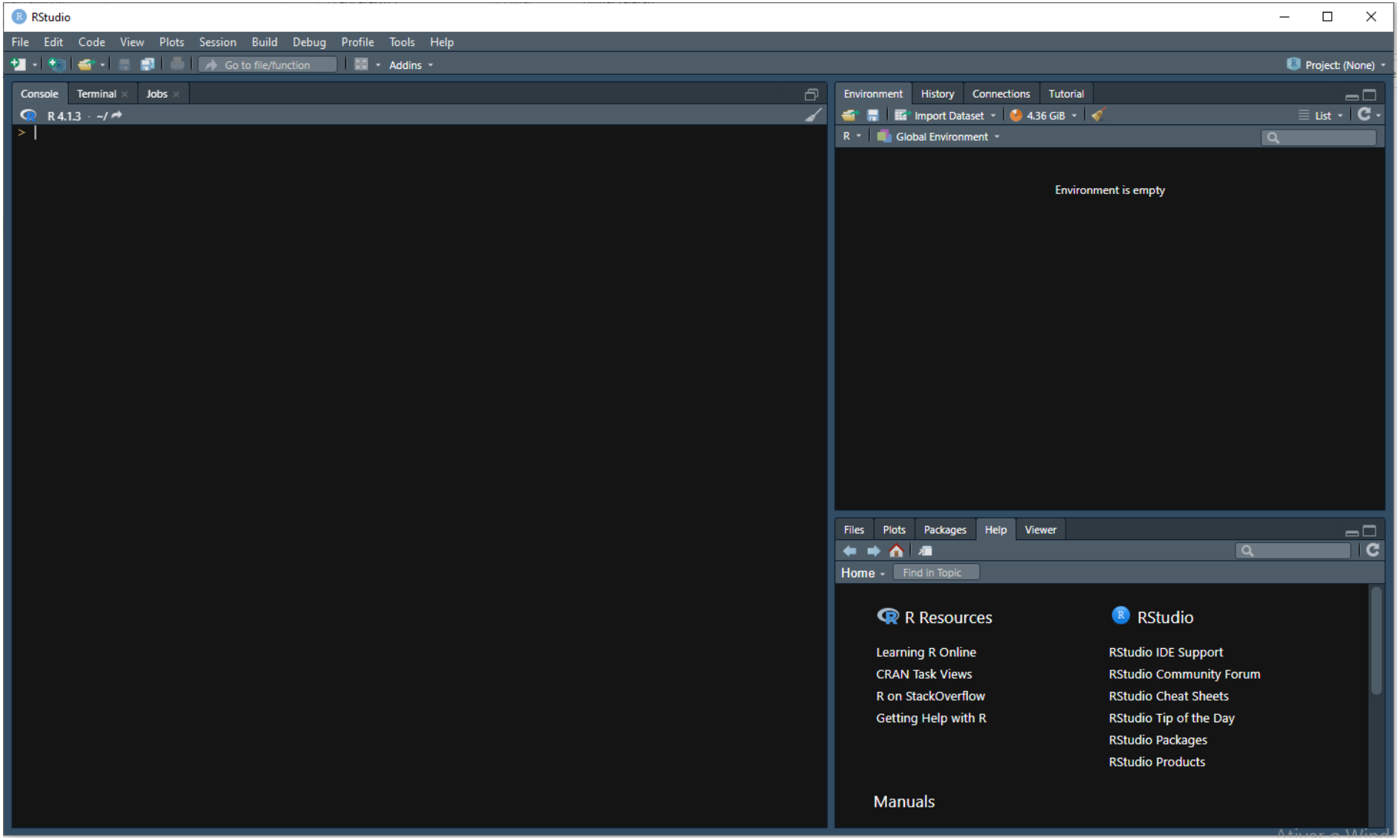
Acessando o R



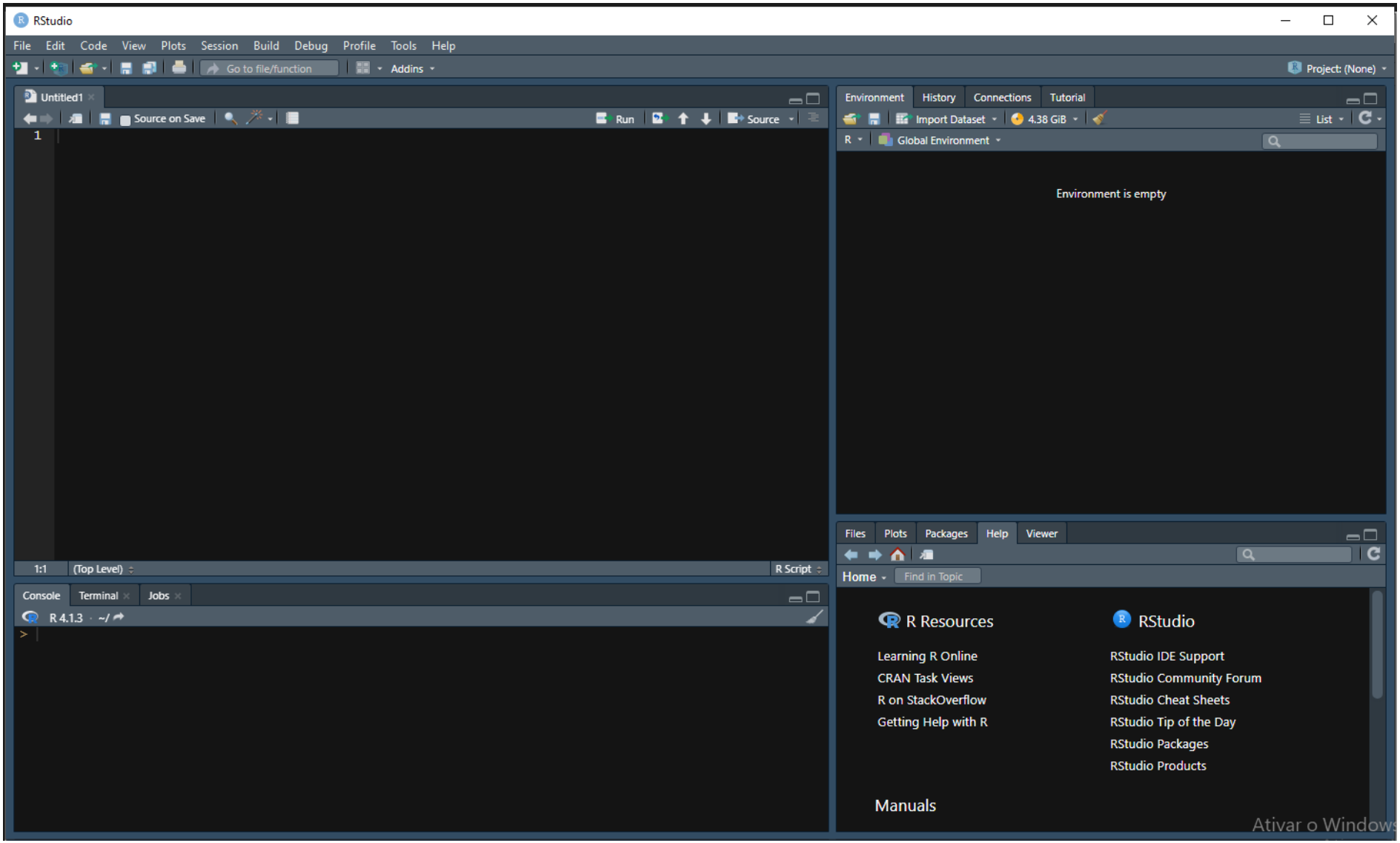
Acessando o RStudio

- Em 2011, foi lançado o RStudio, que é um ambiente integrado de desenvolvimento (IDE, na sigla em inglês).
- O objetivo deste software é facilitar a programação em R, fornecendo recursos práticos ao programador, como sugestão de funções ao começar a digitar e visualização gráfica dos objetos que foram criados em memória.

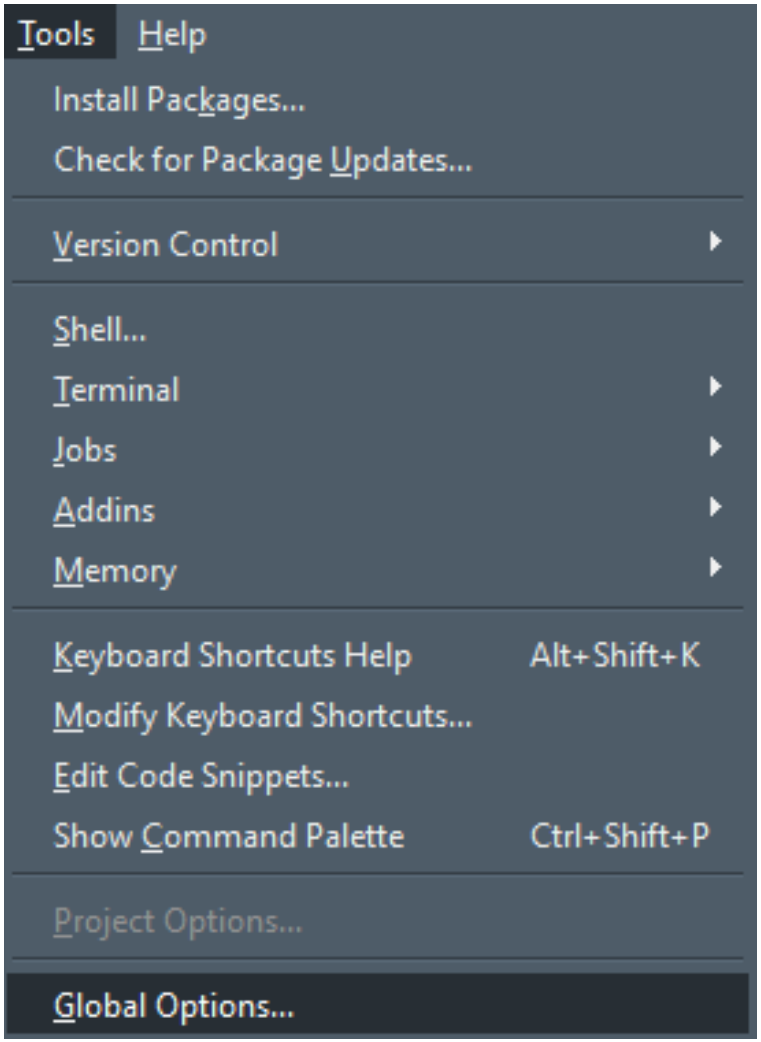
Acessando o RStudio



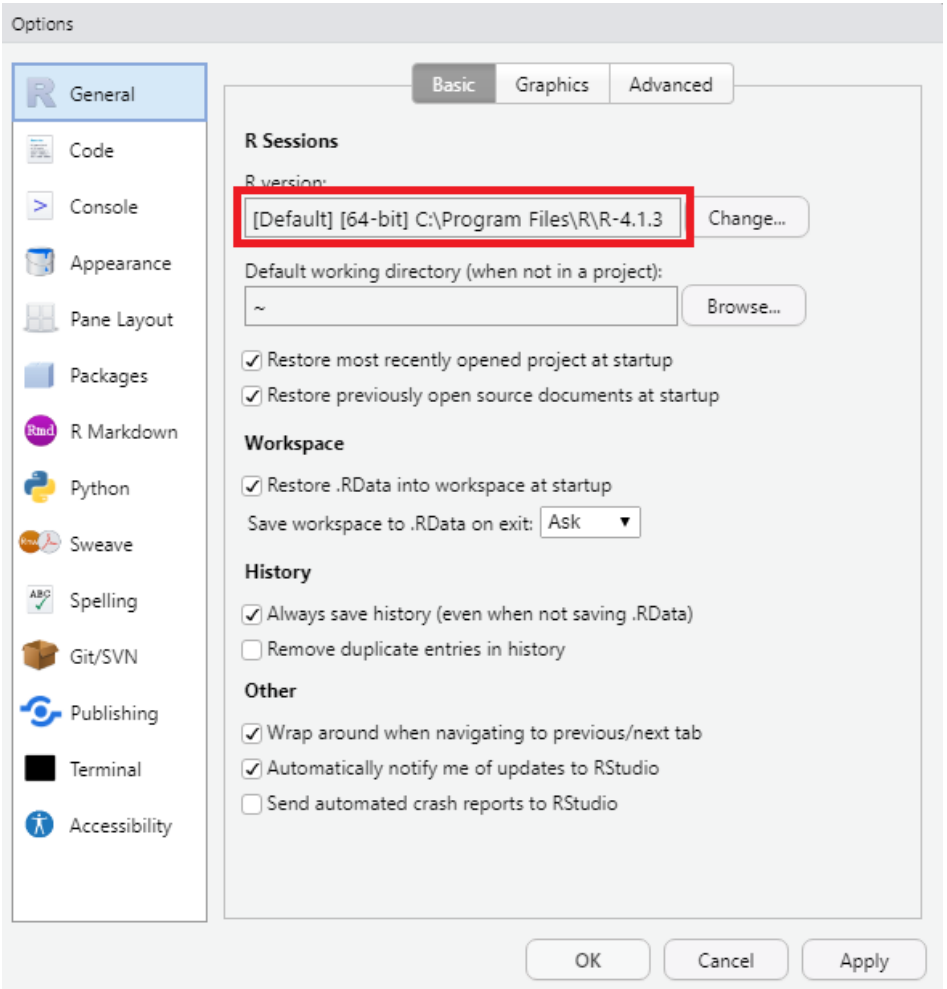
Acessando o RStudio



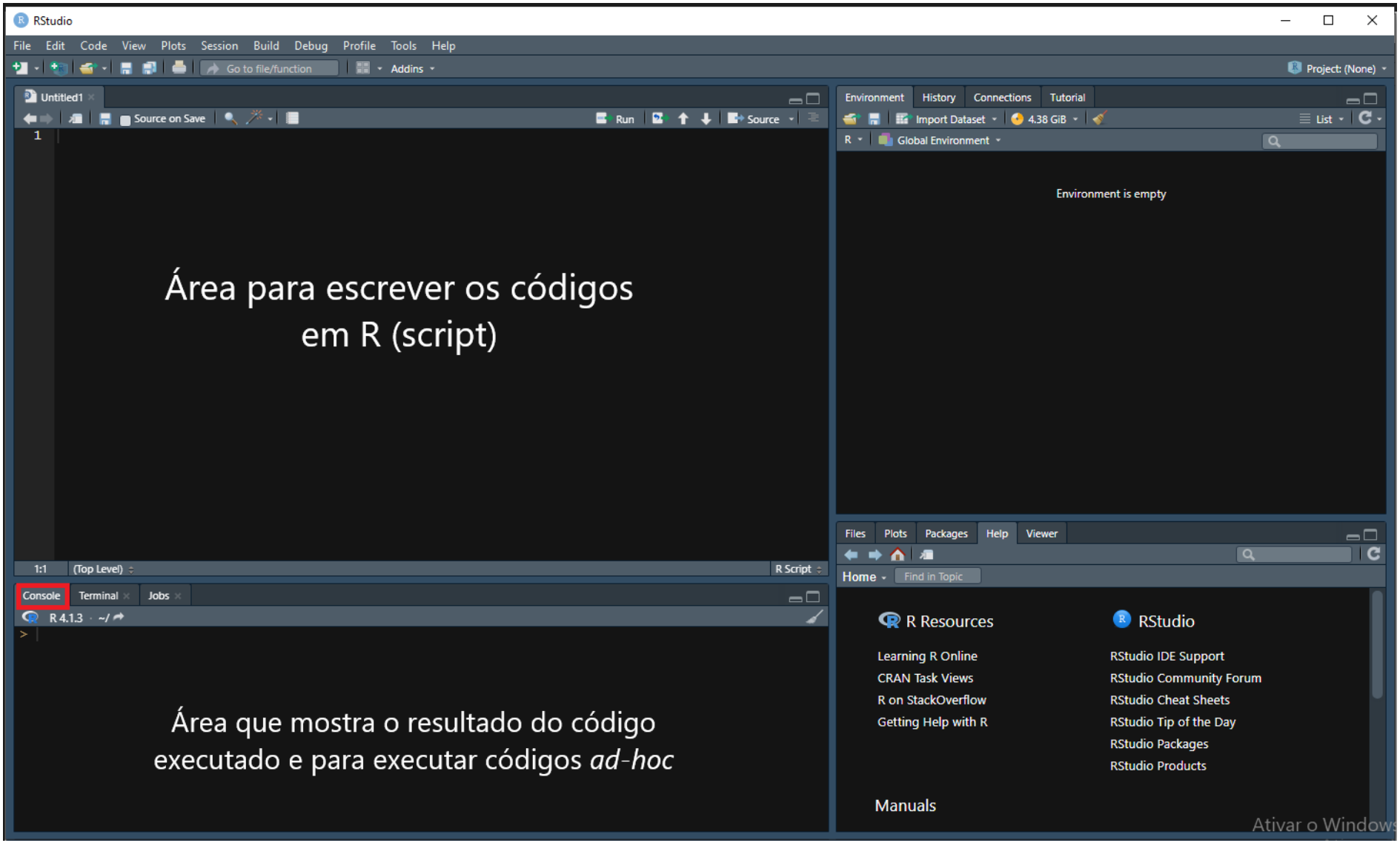
Consultar a versão do R em uso



Consultar a versão do R em uso



Acessando o RStudio



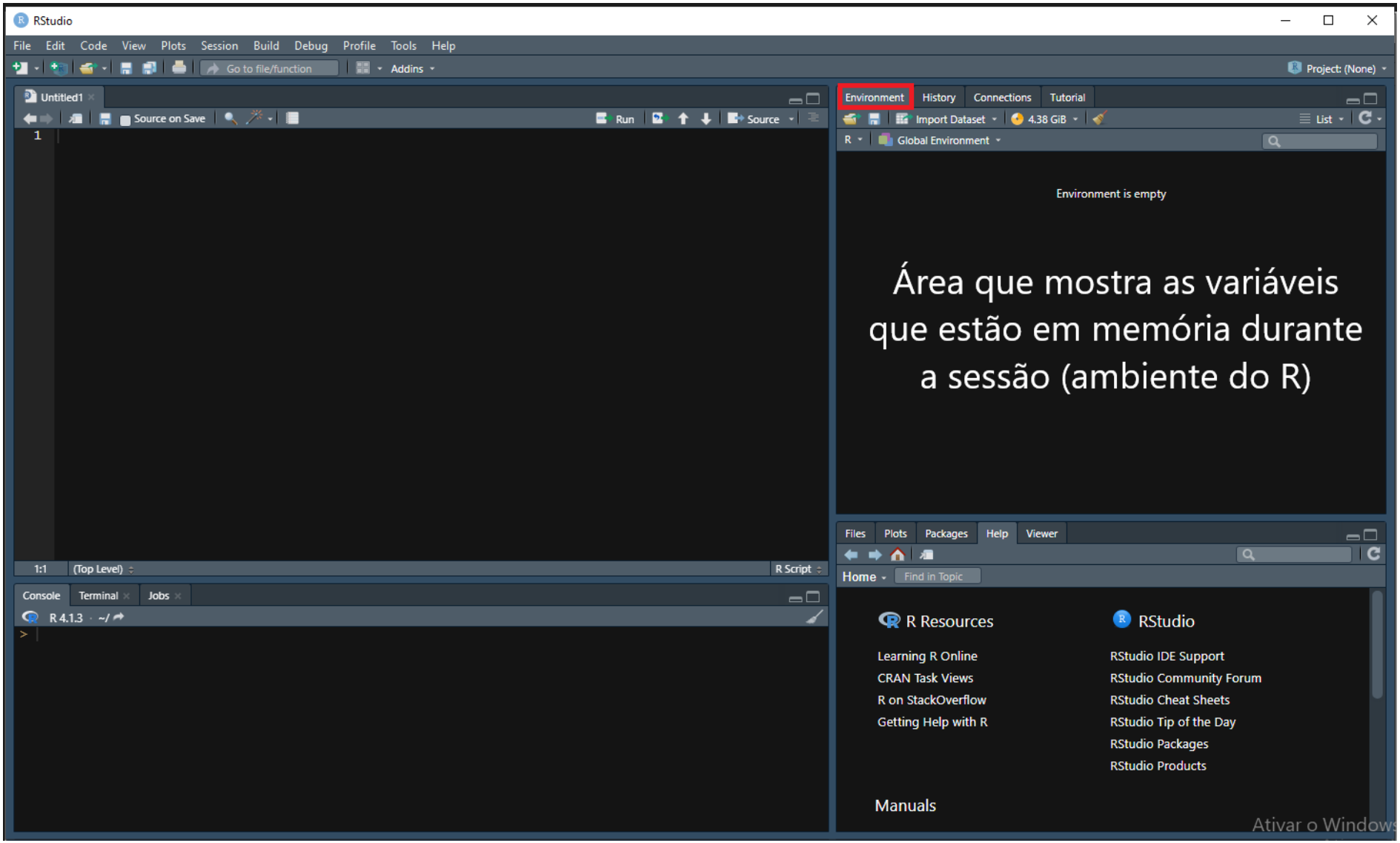
R Script

- O "R Script", ou apenas *script*, é um arquivo utilizado para escrever os códigos em R que serão, posteriormente, executados pelo usuário;
- Um aplicativo construído em R pode ser formado por um ou mais *scripts*;
- O motivo de se dividir um aplicativo em mais de um *script* é melhorar a organização e facilitar a localização dos assuntos dentro do código.

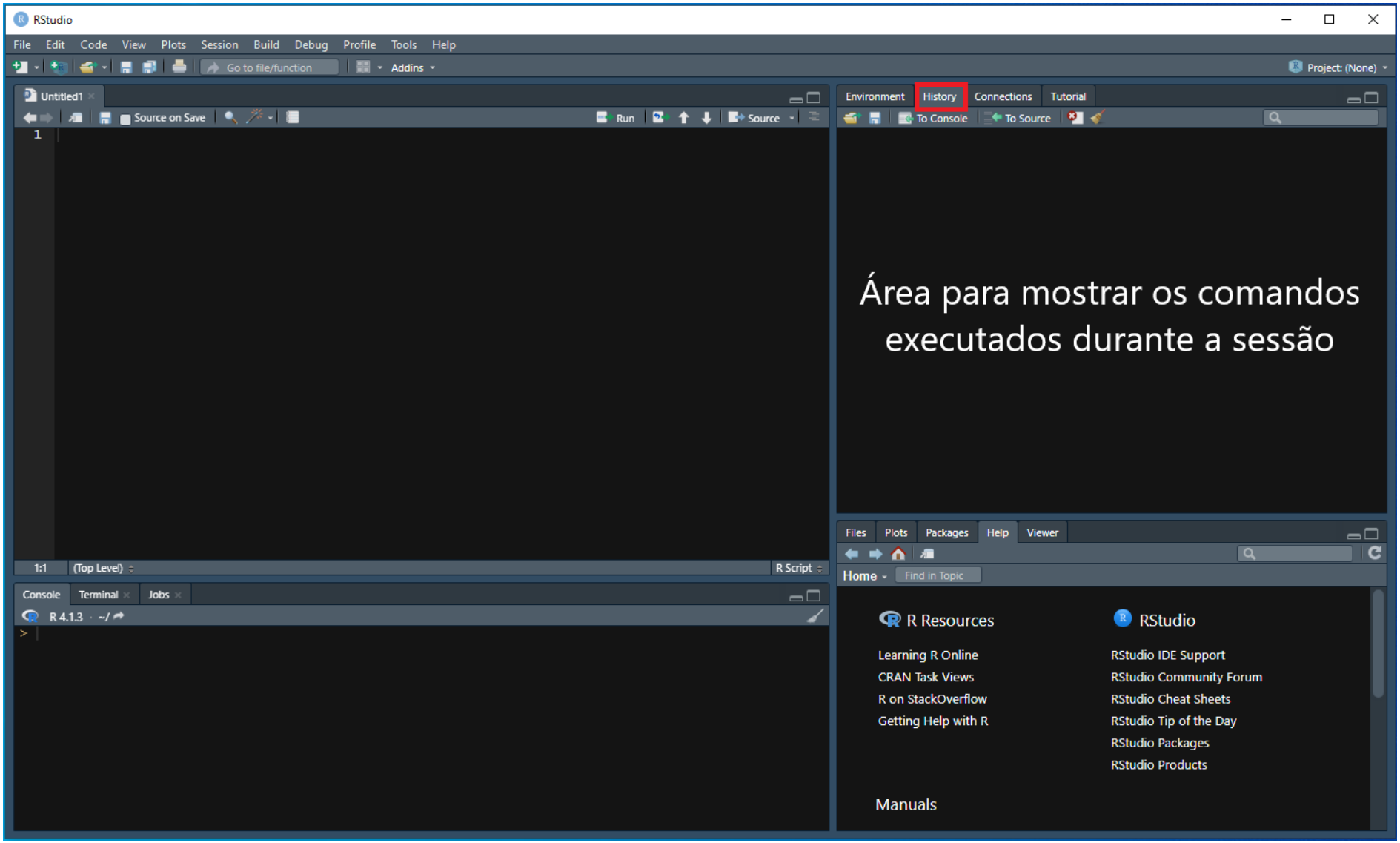
Execução de códigos no *script*

- Para executar um código, selecioná-lo e apertar o botão "Run" (parte superior direita do *script*) ou apertar a tecla de atalho CTRL+ENTER;
- Para executar uma linha inteira de código, há duas opções:
  - selecionar a linha toda e apertar CTRL+ENTER; ou
  - colocar o cursor em qualquer parte da linha e apertar CTRL+ENTER.
- Para executar todo o código contido em um *script*, apertar o botão "Source" (parte superior direita do *script*).

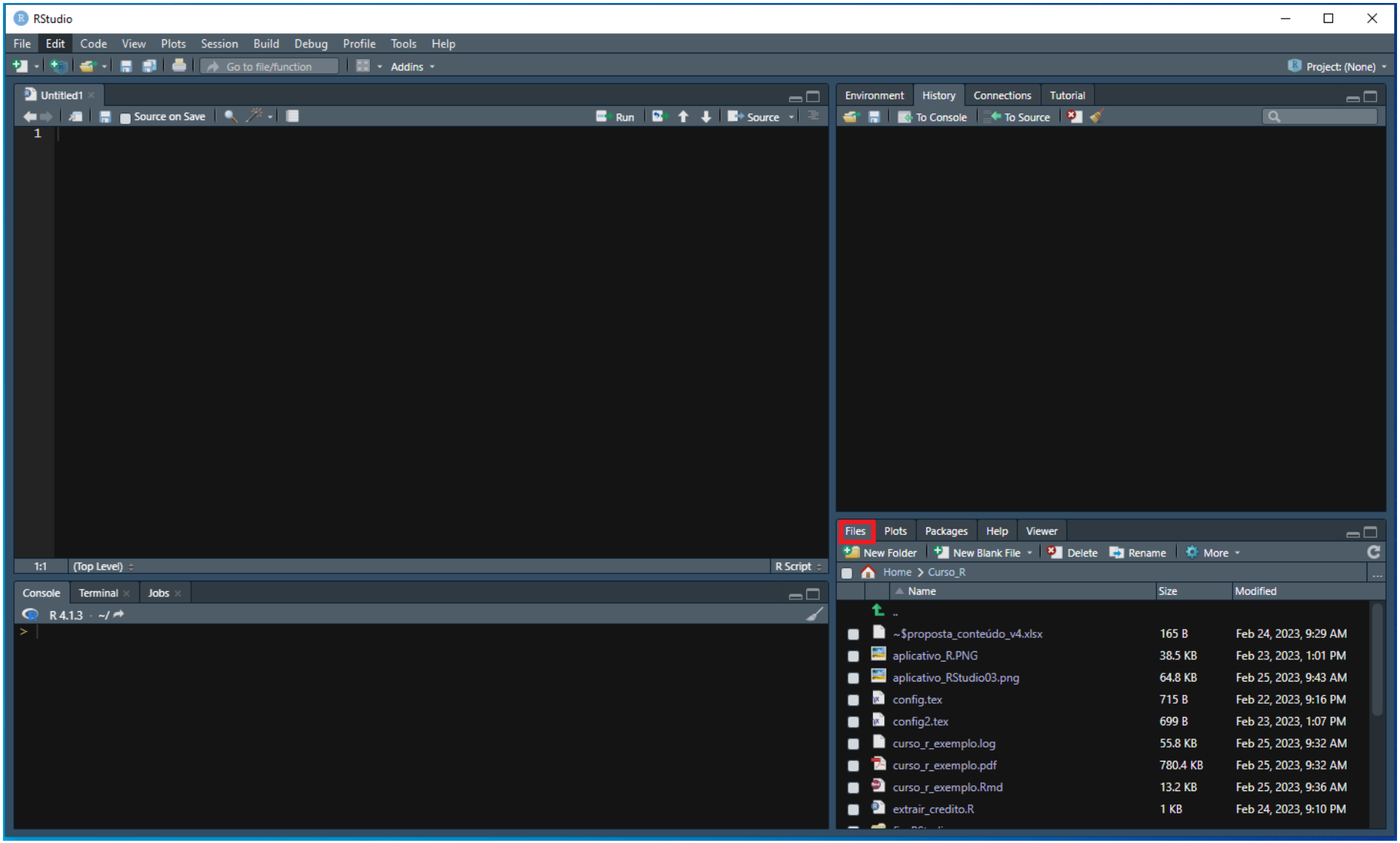
Acessando o RStudio



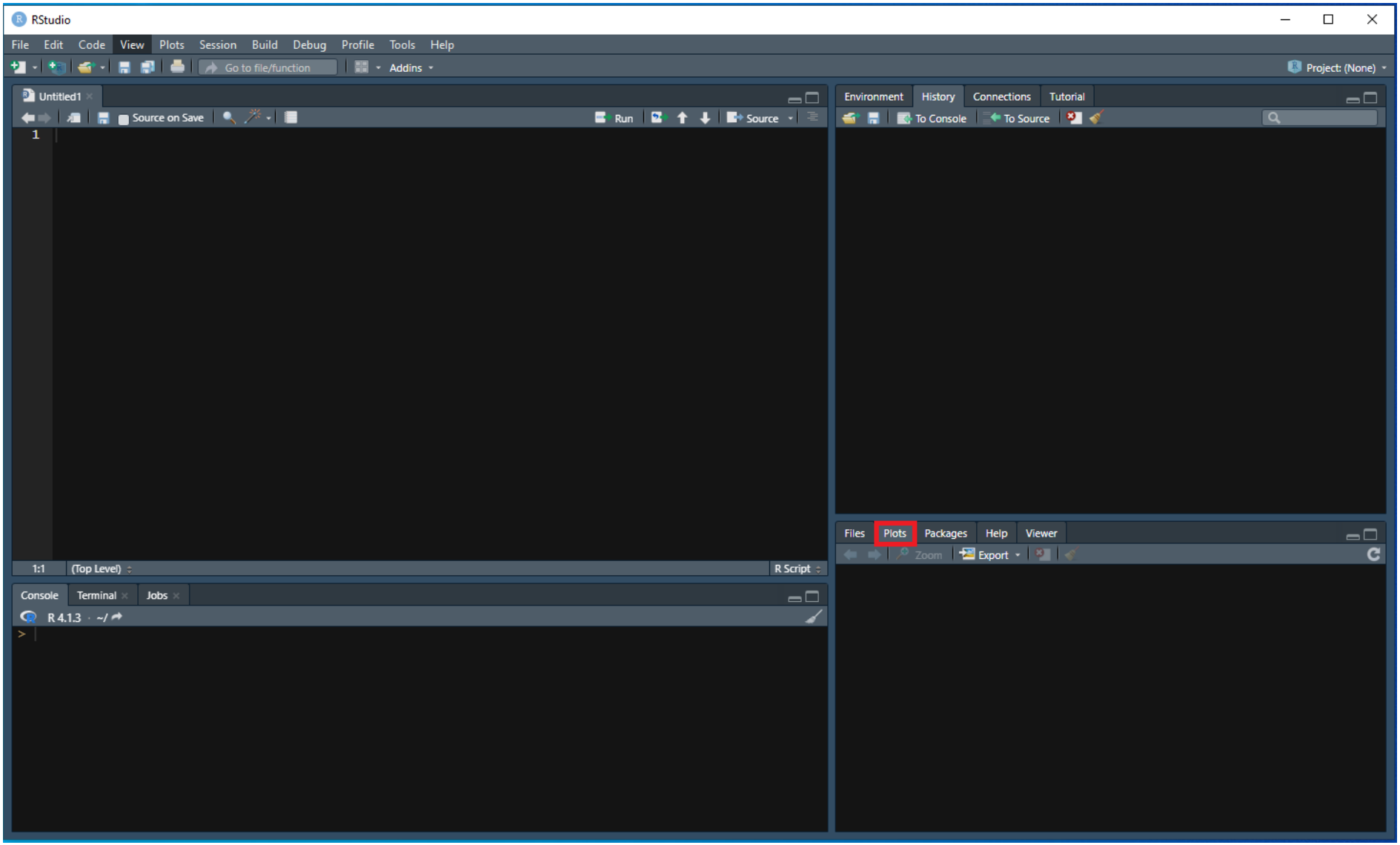
Acessando o RStudio



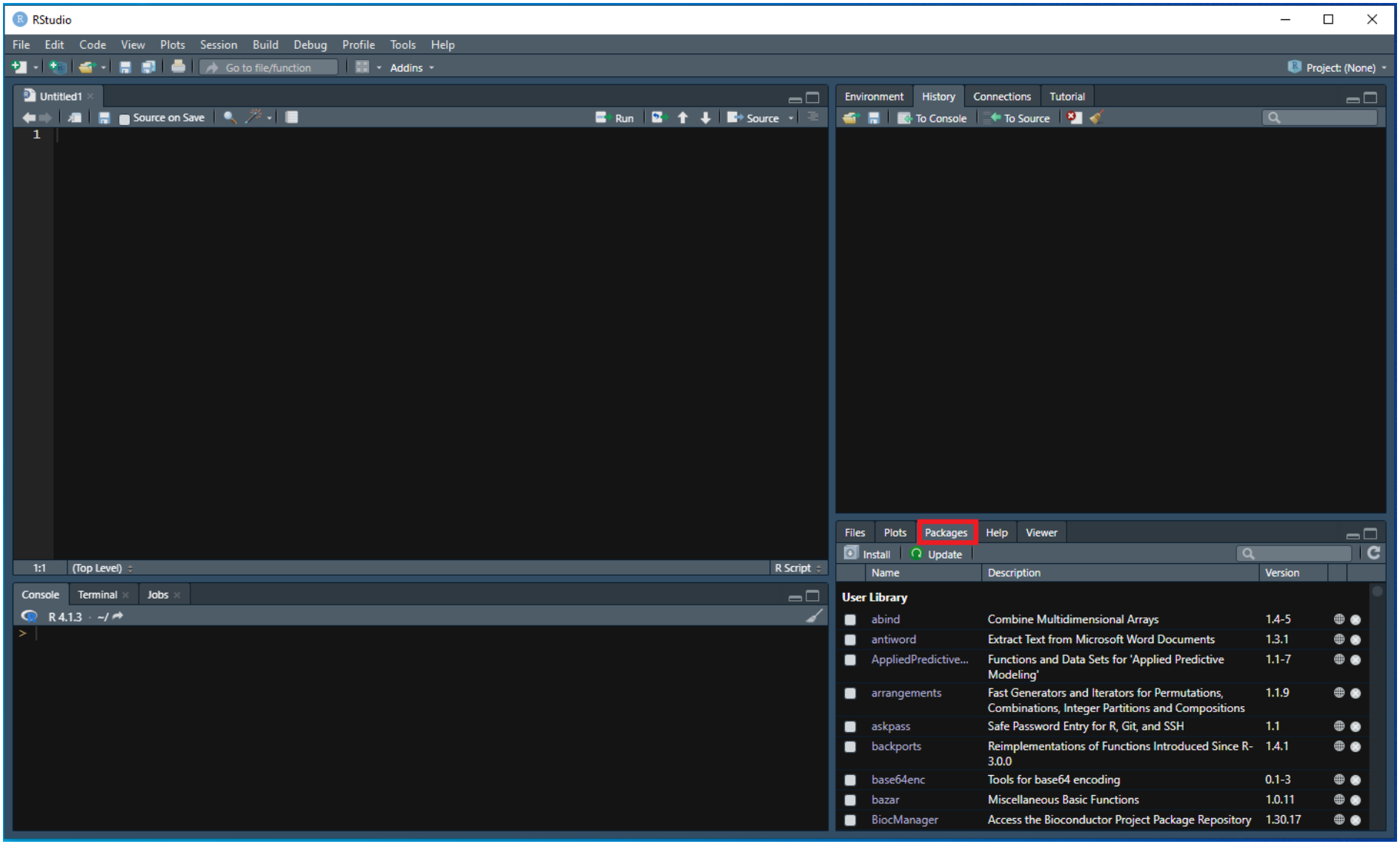
Acessando o RStudio



Acessando o RStudio



Acessando o RStudio



Funções

- Funções são objetos que contém múltiplos códigos interrelacionados que são executados em uma ordem predefinida sempre que a função for executada.[2]
  Exemplo: `class(x)`
- A funções geralmente possuem um ou mais argumentos que devem ser informados pelo usuário para que a função seja executada.
  Exemplo: `x` é um argumento da função `class`.

[2] DATAQUEST. How to write functions in R (with 18 code examples). 2022. Disponível em: https://www.dataquest.io/blog/write-functions-in-r/. Acesso em: 25 fev. 2023

O que são *packages* (pacotes)

- Quando você não encontra a função que precisa no *Help* do RStudio, é necessário utilizar outras funções ou códigos para atingir o seu objetivo (por exemplo, estimar um modelo econométrico).
- Contudo, talvez outras pessoas já tenham programado o que você necessita. E, além disso, criaram funções, colocaram elas em *packages* e disponibilizaram no site do R.
- Portanto, *packages* são grupos de funções e documentação destas que podem ser instalados no R e utilizados pelos usuários. Alguns pacotes contém também ou apenas bases de dados.
- Muitos *packages* também executam ações que o R já executa, mas os *packages* o fazem de forma mais fácil e/ou rápida, motivo para utilizá-los.
- Qualquer usuário pode criar *packages* e não precisa necessariamente disponibilizá-lo no site do R. Ele pode ser distribuído localmente.

Packages carregados pelo R

- Digite `sessionInfo()` no console para verificar os *packages* que estão carregados na sessão atual do R.
- O package `base` é o pacote que contém as funções básicas do R.
- Quando o `base` é carregado, outros pacotes são carregados juntos com ele (`stats`, `graphics`, `grDevices`, `utils`, `datasets` e `methods`).

Encontrando *packages*

- Para consultar os *packages* disponíveis no site do R, acesse o site do R (https://cran.r-project.org/) e, em seguida, clique em "Packages" (lado esquerdo).

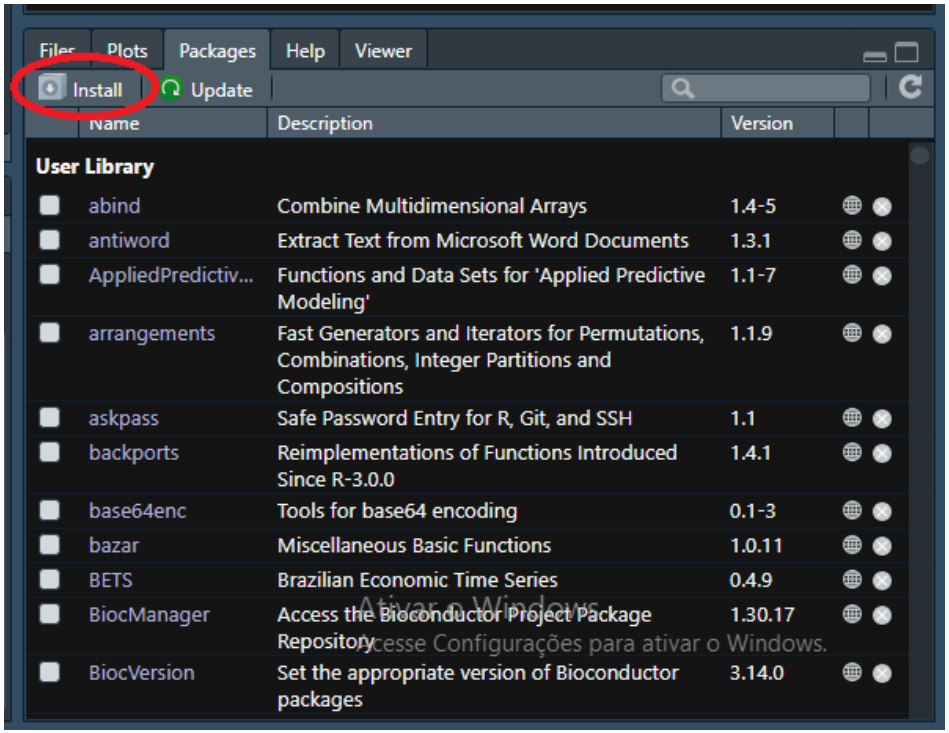
Como instalar *packages*

- *Packages* que estão no site do R:

```
install.packages("nome_do_pacote", repos =
"https://cran.r-project.org/")
```

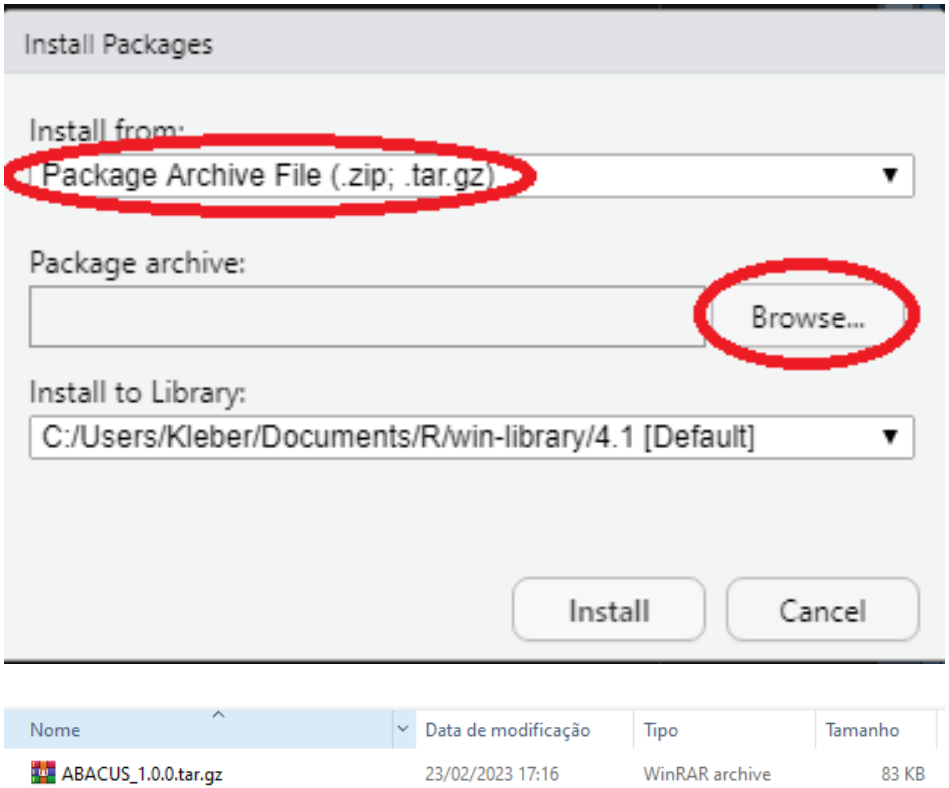
Como instalar *packages*

- *Packages* que estão salvos localmente:



Como instalar *packages*

- *Packages* que estão salvos localmente:



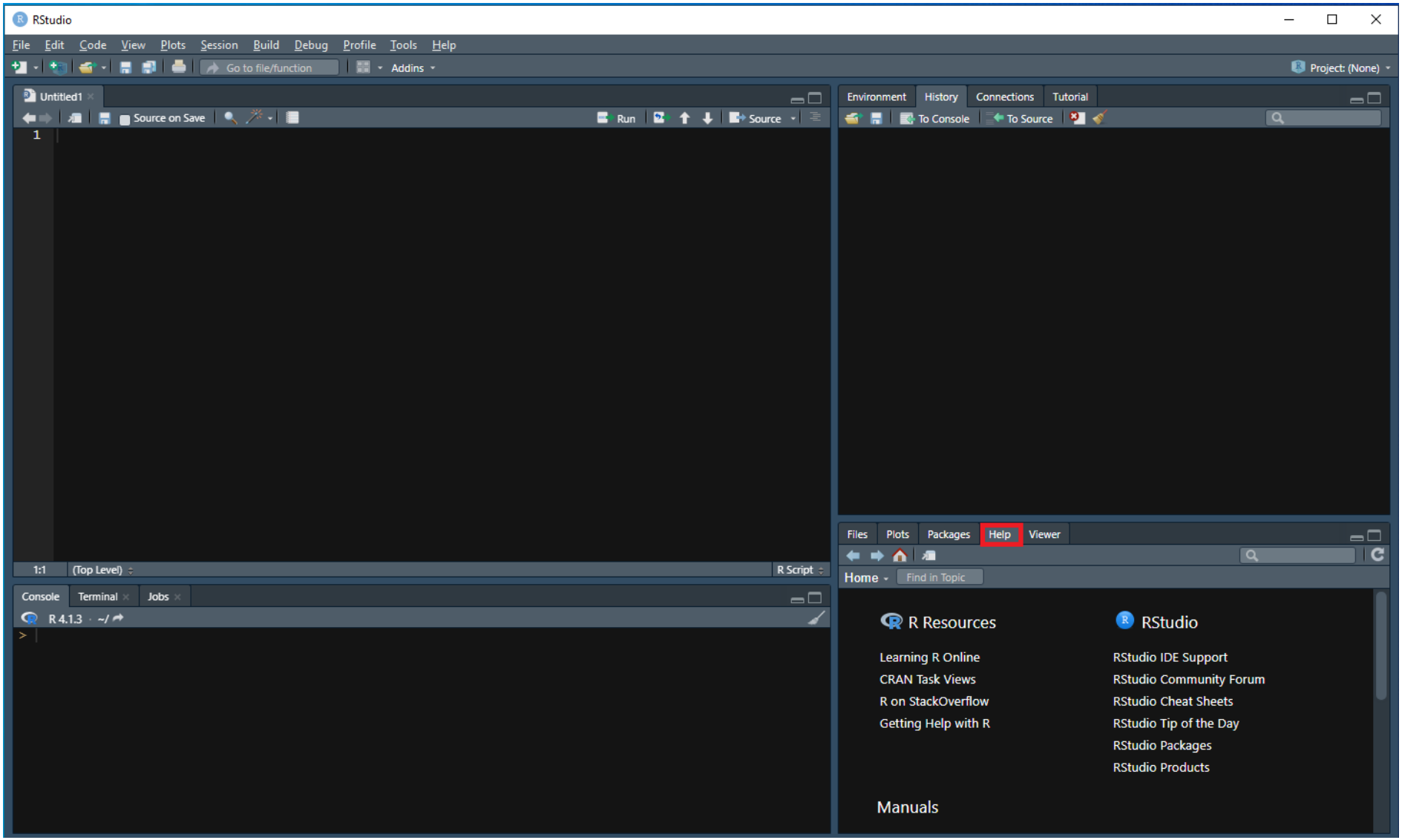
Instalando *packages*

- Vamos instalar o *package* `ggplot2`.
- Esse procedimento deve ser feito uma única vez. Se o R for atualizado (apenas o R, não o RStudio), é necessário instalar os *packages* novamente na nova versão do R.

Como utilizar *packages*

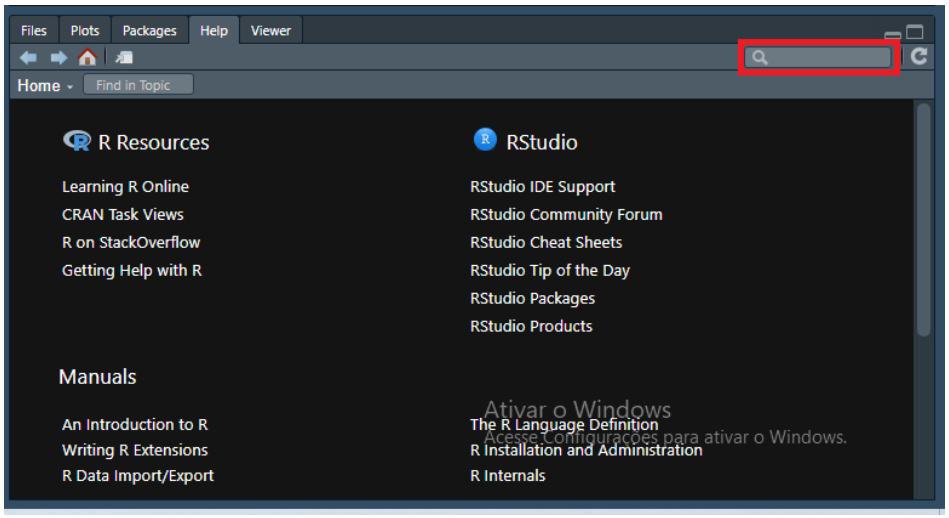
- Para utilizá-lo, nós usamos a função `library(x)`, onde `x` é o nome do *package*.
- Sempre que o RStudio é fechado, os pacotes carregados são removidos da memória. Portanto, ao abrir o RStudio de novo, será necessário executar novamente a função anterior.

Acessando o RStudio



Como obter ajudar no R

1. Clicar nas opções dentro da área de *Help* ou digitar na caixa de *search* da área de *Help*:



Como obter ajudar no R

2. Procurar na internet, principalmente no site Stackoverflow (https://stackoverflow.com/);
3. Colocar o cursor sobre o nome da função e apertar F1; e
4. Digitar no *console* "?" seguido do nome da função. Exemplo: `?class`.

Exercício 1 (1 min.)

a. Obtenha ajuda para as seguintes funções: `sum()`, `apply()`, `median()` e `head()`.

Variável ou objeto

- Variável ou objeto é um espaço alocado na memória do computador que armazena dados, sendo que o nome da variável é usado pelo aplicativo para se referir a estes dados[3];
- Pode ser um único valor ou uma tabela com milhares de valores, o que muda, neste caso, são as estruturas de dados armazenadas nas variáveis (`vector`, `list`, `data.frame`, ...)

[3] GEEKS FOR GEEKS. R - Variables. 2022. Disponível em: https://www.geeksforgeeks.org/r-variables/. Acesso em: 23 fev. 2023

Variável ou objeto

a. Regras de nomes para variáveis[4]:

- deve começar com letra e pode conter letras, números, *underscore* (_) e ponto (.);
- pode começar com (.), mas não deve ser seguido por número. Neste caso, a variável não aparece no painel Ambiente do RStudio (não recomendado);
- caracteres especiais, como "#", "&", entre outros, e espaço em branco não são permitidos;
- não é recomendado usar nomes de variáveis iguais à nomes de funções do R.

[4] DATACAMP. Definitive guide: variables in R tutorial. 2020. Disponível em: https://www.datacamp.com/tutorial/variables-in-r. Acesso em: 23 fev. 2023

Convenção para nomes de variáveis

- Formato recomendado: todas as letras em minúsculo e separar as palavras por *underscore* .
  Exemplos: `rec_adm` e `ipca_serv`.

Case sensitive

- O R é uma linguagem *case sensitive*, ou seja, ela diferencia maiúsculas e minúsculas;
- Por exemplo, a variável `PIB` é diferente da variável `pib`.

Exercício 2 (1 min.)

a. Utilize a convenção de nomes das variáveis dos *slides* anteriores para criar as seguintes variáveis: PIB, com valor de 2.9, IPCA, com valor de 3.25 e resultado primário com valor de -260.00

Visualizar o valor da variável

a. Como mostrar o valor da variável no console do RStudio:

- usar a função `print()` no console ou no *script*;
- digitar o nome da variável no console e teclar ENTER; ou
- digitar o valor da variável no *script*, selecioná-lo ou colocar o cursor em algum local dele e apertar CTRL+ENTER.

b. Para limpar o console, colocar o cursor nele e aperta CTRL+L.

Visualizar o valor da variável

a. Como mostrar o valor de uma variável como uma nova aba do painel de *script*:
   - para variáveis do tipo `vector`, usar a função `View()`;
   - para `data.frame`, `matrix` ou `list`, usar a função `View()` ou clicar no objeto no painel Ambiente.

Visualizar o valor da variável

- Atenção: para variáveis com muitos dados (por exemplo, `data.frame` com milhares ou milhões de linhas), as funções `print()` e `View()` podem demorar a ser executadas e consumir muita memória do computador.
- Uma alternativa é visualizar apenas os primeiros valores da variável com a função `head()`. Por *default*, ela mostra os 6 primeiros valores. Mas a quantidade pode ser alterada;
- Para ver os últimos valores da variável, usar a função `tail()`.

Exercício 3 (2 min.)

a. Crie um `vector` com os seguintes valores: 10, 1.1 e 7;

b. Crie um `data.frame` com duas colunas: a coluna `ano` com valores de 2020, 2021 e 2022; e a `valor` com os valores 5, 4.1 e 2;

c. Utilize a função `View()` para visualizar o valor das duas variáveis como uma nova aba do painel de *script*;

d. Visualize o valor das variáveis no console do RStudio.

Inserir comentários no script

- Os comentários são informações que o programador coloca no *script* para explicar trechos do código;
- Eles são úteis para lembrar a intenção do programador com aquele trecho e também explicá-lo a outros programadores;
- É realizado incluindo o símbolo *hashtag* (#) seguido do comentário.
- A tecla de atalho CTRL+SHIFT+C pode ser usada para transformar código em comentário.

Exercícios 4 (4 min.)

a. Criar a seguinte variável (use as teclas CTRL+ENTER para executar) (colocar algum comentário explicando o objetivo do código a ser feito):

- criar o `data.frame` com as seguintes colunas:
  - "ano": com os valores de 2014 até 2022, no formato `numeric`;
  - "res_fed": resultado primário do Nível Federal, com os valores -22.5, -118.4, -160.3, -119.4, -112.7, -78.6, -745.9, -32.8 e 59.7;
  - "res_reg": resultado primário do Nível Regional, com os valores de -10.1, 7.1, 4.5, 8.8, 4.4, 16.7, 42.9, 97.6 e 66.3;
  - "res_set_pub": resultado primário do Setor Público Consolidado, como sendo a soma das duas colunas anteriores.

Exercícios 4 (4 min.)

- consultar a classe da variável `class()`;
- consulte o `help` da função `str()`
- consultar a estrutura da variável com o comando `str()`; e
- inserir código no *script* para que ele imprima apenas as seis primeiras linhas da variável no console.

Função `is.xxx`

- Servem para verificar se um objeto é de uma determinada classe.
  Retorna `TRUE` ou `FALSE`:
  Exemplo: `is.data.frame()`, `is.vector()` e `is.matrix()`.

```
is.data.frame(iris)
```

```
[1] TRUE
```

Verificar o tamanho de um objeto

- Se for um objeto de duas dimensões: funções `ncol()` (número de colunas) e `nrow()` (número de linhas);
  Exemplos: `ncol(iris)` e `nrow(iris)`
- Se for um `array` ou objeto de duas dimensões: função `dim()`;
  Exemplo: `dim(ex_array)` ou `dim(iris)`
- Se for um vetor: função `length()`.
  Exemplo: `length(c(1,2,3,4,5))`

Verificar nomes de colunas e linhas

- Para colunas: função `colnames()`.
  Exemplo: `colnames(iris)`
- Para linhas: função `rownames()`.
  Exemplo: `rownames(iris)`

Alterar nomes de linhas e colunas

- Alterar todos os nomes:
  `colnames() <- c("novoNome1", "novoNome2", ...)`
  `rownames() <- c("novoNome1", "novoNome2", ...)`
  Exemplo: `colnames(iris) <- c("Sepal.Length","Sepal.Width","Petal.Length", "Petal.Width","Species")`
- Alterar apenas um nome (o primeiro, por exemplo) (forma não recomendada):
  `colnames(iris)[1] <- "novoNome1"`
- Alterar apenas um nome (o primeiro, por exemplo) (forma recomendada):
  `colnames(iris)[which(colnames(iris)=="Sepal.Length")] <- "novoNome1"`

Exercícios 5 (2 min.)

a. Crie o seguinte `data.frame` (coloque uma linha de comentário na linha anterior a cada código a ser escrito):

- criar um `data.frame` chamado `atividade`, o qual contém a variação do PIB real de alguns anos (em %), que tenha as seguintes colunas:
  - "ano": colocar os anos de 2016 a 2021, no formato `numeric`;
  - "pib": colocar os seguintes valores: -3.3, 1.3, 1.8, 1.2, -3.3 e 5.0, no formato `numeric`.
- execute a função necessária para verificar se as duas colunas de `atividade` estão no formato `numeric`;
- execute a função para verificar o número de linhas de `atividade`;
- execute a função para verificar o número de colunas de `atividade`;
- renomeie as colunas de `atividade` para "data" e "pib_real", respectivamente.

Coerção de classe

- Coerção de classe representa a ação para tentar transformar uma variável em outra classe. Por exemplo, de caractere para número;
- O R realiza dois tipos de coerção[5]:
  a. implícita (o R realiza automaticamente a coerção):
     - Se um `vector` ou uma coluna de `data.frame` possuir `character` e `numeric`, o `numeric` será convertido para `character`;
     - Se um `vector` ou uma coluna de `data.frame` possuir `logical` e `numeric`, o `logical` será convertido para `numeric` (`TRUE`= 1 e `FALSE` = 0);
     - Se um `vector` ou uma coluna de `data.frame` possuir `logical` e `character`, o `logical` será convertido para `character`.

     Exemplos:
     `TRUE+3`. O `TRUE` será convertido para o número 1;
     `c("casa", 10)`. O 10 será convertido para `character`.

[5] INSTROVATE TECHNOLOGIES. Coercion in R. 2019. Disponível em: https://technicaljockey.com/r-programming-training/coercion-in-r/. Acesso em: 26 fev. 2023

Coerção de classe

b. explícita (quando o usuário aplica uma função em uma variável para transformá-la em outra classe).
Exemplos:
`as.numeric(c("1","5"))`. Será convertido em `numeric`;
`as.character(c(1,2,3))`. Será convertido em `character`;
`iris[,1] <- as.character(iris[,1])` irá converter a primeira coluna de `iris` para caracter.

Valores faltantes ou impossíveis

- E se o usuário tentar transformar o vetor `c("1","2","casa")` para número?

Valores faltantes ou impossíveis

- O R usa o termo `NA` (*not available*) para representar valores que estão faltando (*missing*).
- Valores impossíveis, como divisão por zero, são representados por `Inf` (*infinity*).
- A divisão de zero por zero gera o `NaN` (*not a number*).
- Por fim, existe o termo `NULL` que é usado para criar um variável na memória, mas que não possui nenhum valor.

Remover os valores faltantes

- Para remover os valores faltantes (apenas `NA` e `NaN`) de uma `matrix` ou de um `data.frame`, pode ser usada a função `complete.cases()` para remover toda a linha que contém valor faltante. Exemplo: `x[complete.cases(x),]`.

Substituir os valores faltantes

- Para substituir os valores faltantes (apenas `NA` e `NaN`) de uma `matrix` ou de um `data.frame`, pode ser usada a função `is.na` para selecionar os valores de `NA` e, em seguida, usar o operador de atribuição para atribuir o valor desejado. Exemplo: `iris$Sepal.Length[is.na(iris$Sepal.Length)] <- 0`.

Valores faltantes ou impossíveis

- Resumo:

Summary	NA	NULL	NaN	Inf
class()	logical	NULL	numeric	numeric
length()	1	0	1	1
check	is.na()	is.null()	is.nan()	is.finite()

Exercícios 6 (3 min.)

a. Crie a seguinte variável:

1. criar um `data.frame` chamado `receita_adm` que tenha as seguintes colunas: - "ano": colocar os anos de 2020, 2021, 2022 e 2023 no formato `character`;
   - "receita_adm": colocar os valores "899.5", "1195.7", "1390.0" e `NA`, no formato `character`;
   - "receita_nao_adm": colocar os valores "163.9", "274.9", "387.7" e `NA` no formato `character`.
2. verificar a estrutura do `data.frame` usando a função `str()`;
3. remover a linha que contém `NA`;
4. transformar todas as colunas de `character` para `numeric`;
5. visualizar apenas a última linha do `data.frame` pela função `tail()` (veja o `Help` da função).

Operadores relacionais

a. Servem para fazer comparações entre um ou mais valores:
   - "==": igualdade;
   - ">": maior que;
   - "<": menor que;
   - ">=": maior ou igual que;
   - "<=": menor ou igual que;
   - "!=": diferente de.

b. Podem ser feitas múltiplas comparações com os seguintes operadores:
   - &: "e" (retorna `TRUE` se todos os valores forem `TRUE`);
   - |: "ou" (retorna `TRUE` se pelo menos um dos valores forem `TRUE`).

Operadores relacionais

a. Para comparar uma variável que possui mais de um valor (como `vector`, `matrix`, `data.frame`) e retornar um único resultado (`TRUE` ou `FALSE`) pode ser usada a função `identical()`;

- Exemplo: `identical(x,y)`.

Operadores relacionais

a. Para comparar vetores contendo apenas valores lógicos (`TRUE` ou `FALSE`), podem ser usadas as funções:

- `all()`: retorna `TRUE` se todos os valores forem `TRUE`;
- `any()`: retorno `TRUE` se algum valor for `TRUE`.

Operadores de conjuntos

a. Principais operadores para conjuntos (`vector`):

- `setdiff(x,y)`: retorna os elementos que estão no `vector` `x` e não estão no `vector` `y`;
- `intersect(x,y)`: retorna os elementos comuns aos dois vetores;
- `union(x,y)`: une os dois vetores, sem repetir os valores que são comuns;
- `x %in% y`: verifica os valores de `x` que estão contidos em `y`.

Exercícios 7 (4 min.)

a. criar as seguintes variáveis no formato `vector`:

- "valor1": nos valores de 10, 11 e 12, no formato `numeric`;
- "valor2": nos valores de 10, 13 e 15 no formato `numeric`.

Exercícios 7 (4 min.)

b. Realize as seguintes operações:

- verifique se os dois objetos são iguais por meio do operador == e atribua o resultado para uma nova variável chamada `valor3`;
- verifique se `valor1` e `valor2` são diferentes por meio do operador `!=`;
- verifique se `valor1` é maior do que `valor2`;
- verifique se `valor1` é maior ou igual do que `valor2`;
- verifique se `valor1` é menor do que `valor2`;
- verifique se `valor1` é menor ou igual do que `valor2`;
- verifique se todos valores de `valor3` são `TRUE`;
- verifique se pelo menos um valor de `valor3` é `TRUE`;
- verifique se `valor1` e `valor2` são iguais usando o `identical()`;
- verifique os valores que `valor1` e `valor2` possuem em comum;
- verifique os valores que estão em `valor2` mas que não estão em `valor1`;
- crie uma nova variável, com o nome `valor4`, formada pela união de `valor1` e `valor2`;
- verifique se o valor 11 está contido em `valor1`;
- verifique se o valor 17 está contido em `valor2`.

Listar os objetos do ambiente

- Use a função `ls()` (dessa forma, com o parênteses vazio) para listar as variáveis ou objetos da área de trabalho.

Apagar os objetos do ambiente

a. Apagar uma única variável: `rm()`;

b. Apagar todos as variáveis:

- Pode ser pelo ícone de vassoura que fica acima do painel do ambiente;
- Pode ser por meio da função `rm(list = ls())`.

Exercícios 8 (1 min.)

a. Listar as variáveis do Ambiente;
b. Apagar apenas as variáveis `valor1` e `valor2`;
c. Apagar todas as variáveis do Ambiente.