Projetos, Objetos, Funções e Pacotes

Ricardo Theodoro

OBSCOOP/USP

Projetos

O que é um projeto?

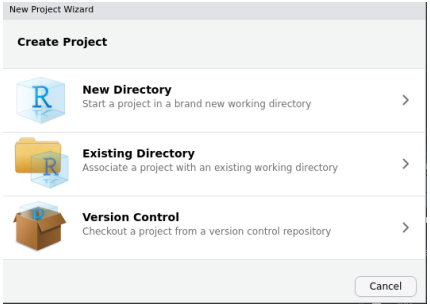
- Um projeto é uma pasta no seu computador em que estarão todos os arquivos que você usurá ou criará na sua análise.

Por que criar um projeto?

- Organização!

Como criar um Projeto?

File -> New Project...

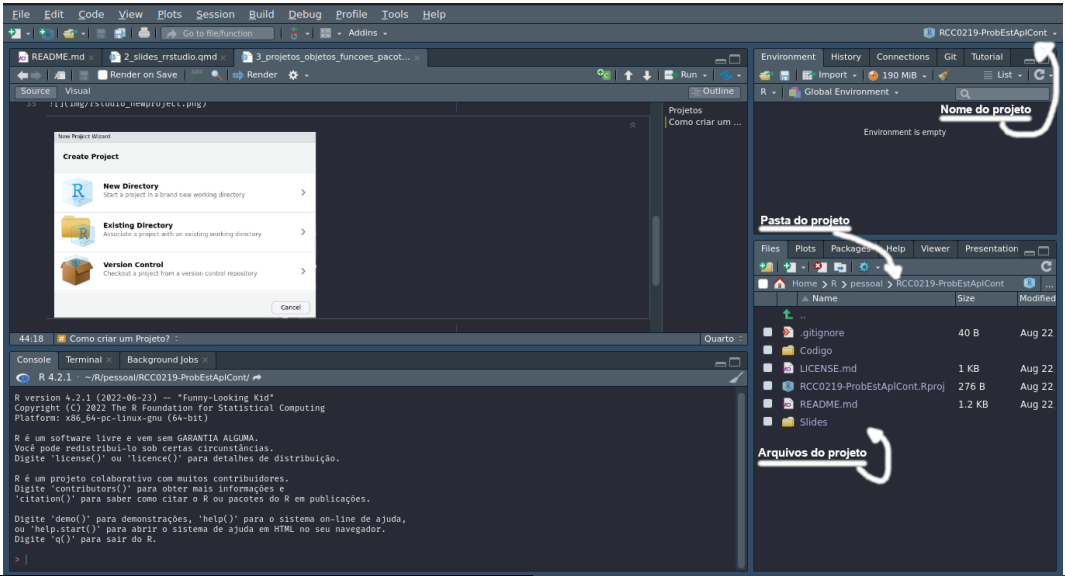


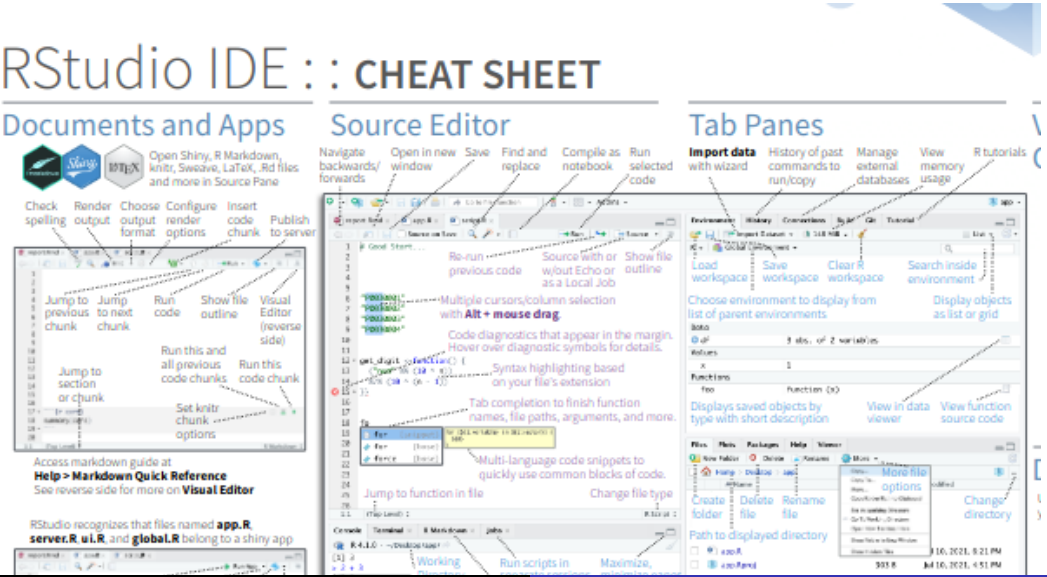
New Directory -> Cria o pojeto em uma pasta nova

Existing Directory -> Cria o projeto em uma pasta existente

Version Control -> Baixa um projeto do Github, por exemplo

Verificando o Projetos




RStudio IDE : : CHEAT SHEET
Documents and Apps
Open Shiny, R Markdown, knitr, Sweave, LaTeX, .Rd files and more in Source Pane
Check spelling
Render output
Choose output format
Configure render options
Insert code chunk
Publish to server
Jump to previous chunk
Jump to next chunk
Run code
Show file outline
Visual Editor (reverse side)
Jump to section or chunk
Run this and all previous code chunks
Run this code chunk
Set knitr chunk options
Access markdown guide at
Help > Markdown Quick Reference
See reverse side for more on Visual Editor
RStudio recognizes that files named app.R, server.R, ui.R, and global.R belong to a shiny app
Source Editor
Navigate backwards/ forwards
Open in new window
Save
Find and replace
Compile as notebook
Run selected code
Re-run previous code
Source with or w/out Echo or as a Local Job
Show file outline
Multiple cursors/column selection with Alt + mouse drag
Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.
Syntax highlighting based on your file's extension
Tab completion to finish function names, file paths, arguments, and more.
Multi-language code snippets to quickly use common blocks of code.
Jump to function in file
Change file type
Working
Run scripts in
Maximize,
Tab Panes
Import data with wizard
History of past commands to run/copy
Manage external databases
View memory usage
R tutorials
Load workspace
Save workspace
Clear R workspace
Search inside environment
Choose environment to display from list of parent environments
Display objects as list or grid
Displays saved objects by type with short description
View in data viewer
View function source code
Create folder
Delete file
Rename file
More file options
Change directory
Path to displayed directory

Objetos

São nomes que guardam valores.

- Variável

```
# Salvando `1` em `a`
a <- 1
# Avaliando o objeto `a`
a
```

```
[1] 1
```

```
# Salvando `2` em `A`
A <- 2
# Avaliando o objeto `A`
A
```

```
[1] 2
```

ATENÇÃO:

- O R diferencia letras maiúsculas de minísculas
- Não utilizem caracteres especiais, apenas _
- O nome deve representar seu valor

Classes dos Objetos

A classe de um objeto define o tipo de valor armezenado nele.

É a partir dela que as funções e operadores conseguem saber exatamente o que fazer com um objeto.

Classes dos Objetos

Para saber a classe de um objeto, basta rodarmos class(nome-do-objeto).

```
x <- 1
class(x)
```

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

```
y <- "a"
class(y)
```

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

```
class(mtcars)
```

```
[1] "data.frame"
```

```
class(TRUE)
```

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

Vetores

Vetores são apenas conjuntos indexados de valores.

Cada coluna de um data frame será representada como um vetor.

```
vetor_num <- c(1, 2, 3)
class(vetor_num)
```

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

```
vetor_letras <- c("a", "b", "c")
class(vetor_letras)
```

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

```
vetor_num_letras <- c("a", 3, "c")
class(vetor_num_letras)
```

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

Trabalhando com objetos e vetores

- Criando sequência

```
vetor_1a10 <- 1:10
```

```
vetor_10a1 <- 10:1
```

```
vetor_10a1[3] # Peguando um número na sequência
```

```
[1] 8
```

```
vetor_10a1[3:7] # Extraindo valor de um conjunto
```

```
[1] 8 7 6 5 4
```

Operações com vetores

```
vetor_10a1 + 6
```

```
 [1] 16 15 14 13 12 11 10  9  8  7
```

```
vetor_10a1 + vetor_num
```

```
 [1] 11 11 11  8  8  8  5  5  5  2
```

```
# Reparem que como um vetor é menor que o outro, o R recicla a diferença
```

Pertence

Como verificar se um valor está dentro de um vetor?

```
c(1, 2, 3) %in% vetor_1a10
```

```
[1] TRUE TRUE TRUE
```

```
vetor_num %in% vetor_1a10
```

```
[1] TRUE TRUE TRUE
```

Funções

São nomes que guardam um código de R.

Sempre que você precisar repetir um trecho de código para realizar uma tarefa, você deve criar uma função e utilizá-lá.

Exemplo: função soma

```
sum(a, A)
```

```
[1] 3
```

```
# sum() é a função
# a e A são os argumentos da função
```

Os valores dentro do parênteses da função são chamados de argumentos e serão utilizados como parâmetros para cálculo do resultado.

Criando Funções

Podemos criar nosas próprias funções

```
soma_3_divide_2 <- function(valor){

   ({{ valor }} + 3) / 2

}

soma_3_divide_2(7)
```

```
[1] 5
```

```
soma_3_divide_2(18)
```

```
[1] 10.5
```

```
soma_3_divide_2(9)
```

Pacotes

O que são pacotes?

- São conjuntos de funções
- Existem muitos pacotes já criados que facilitam a programação

Ex: Pacote `dplyr`. Pacote muito usado para tratamento de dados, como selecionar variáveis, filtrar valor, unir data.frames.

Como instalar pacotes

Todo pacote precisa ser instalado uma vez

```
install.packages("dplyr")
```

Como chamar pacotes

Sempre, antes de usar, é preciso chamar o pacote

Duas formas de chamar:

1. Carregar todas as funções presentes no pacote

```
library(dplyr)
mtcars |>
   select(cyl, mpg) |>
   filter(mpg > 33)
```

```
               cyl  mpg
Toyota Corolla   4 33.9
```

Como chamar pacotes

2. Carregar apenas a função chamada, durante seu uso

```
#pacote::funcao()

mtcars |>
   dplyr::select(cyl, mpg) |>
   dplyr::filter(mpg > 33)
```

```
               cyl  mpg
Toyota Corolla   4 33.9
```