Introdução a Linguagem R

Beatriz Milz

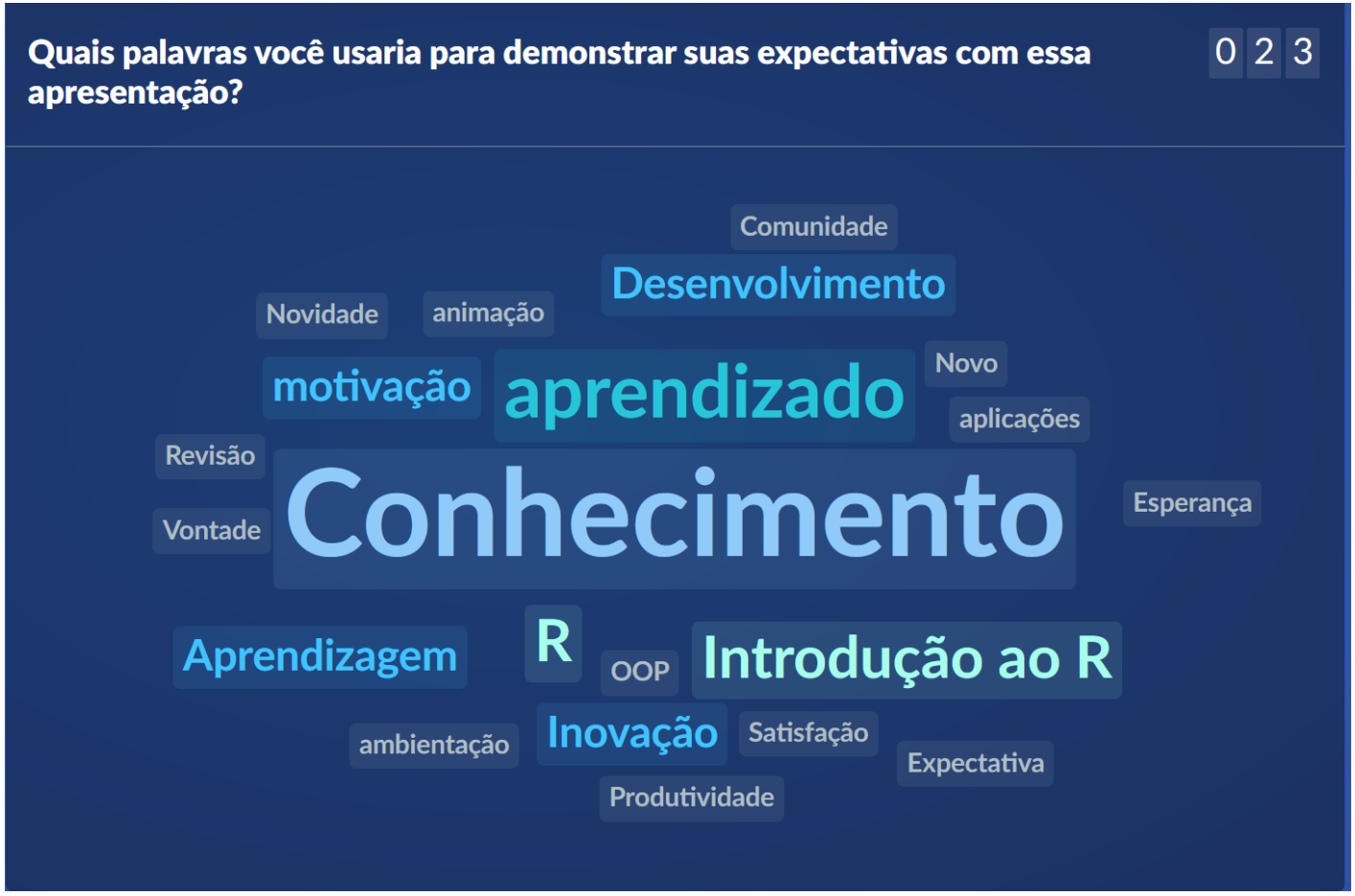
Meetup R-Ladies Curitiba

2º semestre/2020

Ilustração por Allison Horst

Atualizado em 27 de agosto de 2020.
Para ativar o modo tela cheia, pressione `ALT + F`, e depois `F`

R-Ladies Curitiba - 26/08/2020



Beatriz Milz

Contexto Acadêmico

- Doutoranda no Programa de Pós-Graduação em Ciência Ambiental (PROCAM)
  - Instituto de Energia e Ambiente - Universidade de São Paulo
- Equipe da Secretaria Executiva Editorial - Revista Ambiente & Sociedade
- Anteriormente:
  - Mestre em Ciências - UNIFESP;
  - Bacharel em Gestão Ambiental - EACH/USP

Comunidades de R

- Co-autora do pacote `{dados}`
- Co-organizadora: R-Ladies São Paulo ♡
- Comitê organizador:
  - satRday São Paulo
  - LatinR
  - useR! 2021
- Instrutora The Carpentries

Sobre este material

- Público-alvo: pessoas com interesse em começar a utilizar o R para análise de dados.
- Parte deste conteúdo é derivado deste material da R-Ladies São Paulo.

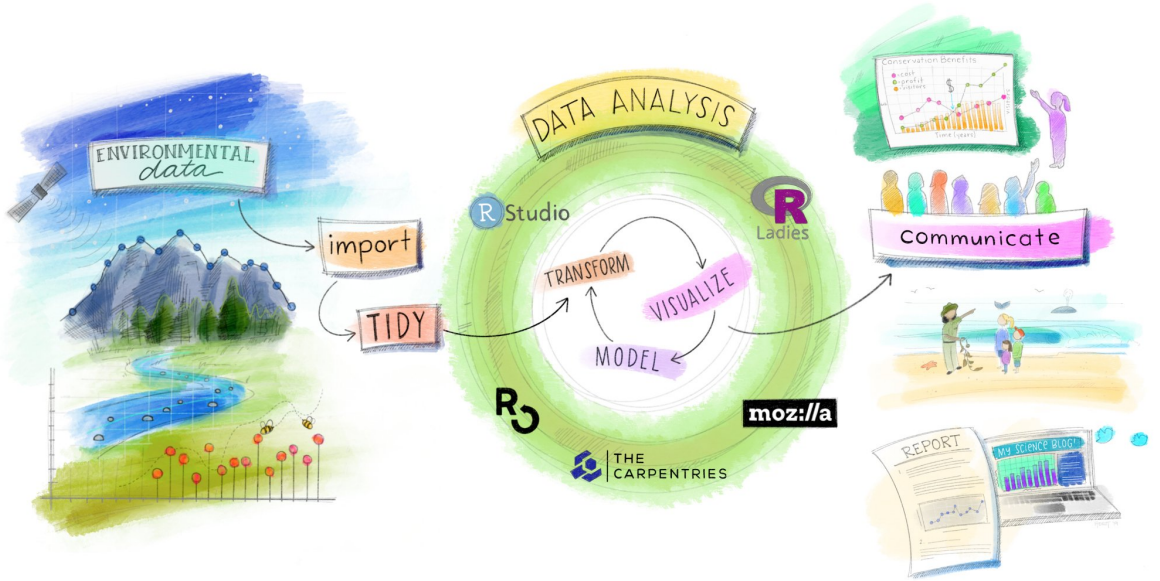


Ilustração por Allison Horst

- Conteúdo: principais conceitos necessários antes de começar a aprender o tidyverse
- Dados tabulares:

preco	quilate	corte	cor	transparencia	profundidade	tabela	x	y	z
326	0.23	Ideal	E	SI2	61.5	55	3.95	3.98	2.43
326	0.21	Premium	E	SI1	59.8	61	3.89	3.84	2.31
327	0.23	Bom	E	VS1	56.9	65	4.05	4.07	2.31
334	0.29	Premium	I	VS2	62.4	58	4.20	4.23	2.63
335	0.31	Bom	J	SI2	63.3	58	4.34	4.35	2.75
336	0.24	Muito Bom	J	VVS2	62.8	57	3.94	3.96	2.48

Base de dados `diamante` disponível no pacote `dados`.

Pré-requisitos

Nesta atividade:

- RStudio Cloud

Para usar no dia-a-dia:

- `R` e `RStudio` instalados no seu computador:


- Links para instalação:
  - R
  - RStudio

R e RStudio

(Acompanhe no `RStudio Cloud` )

01:00

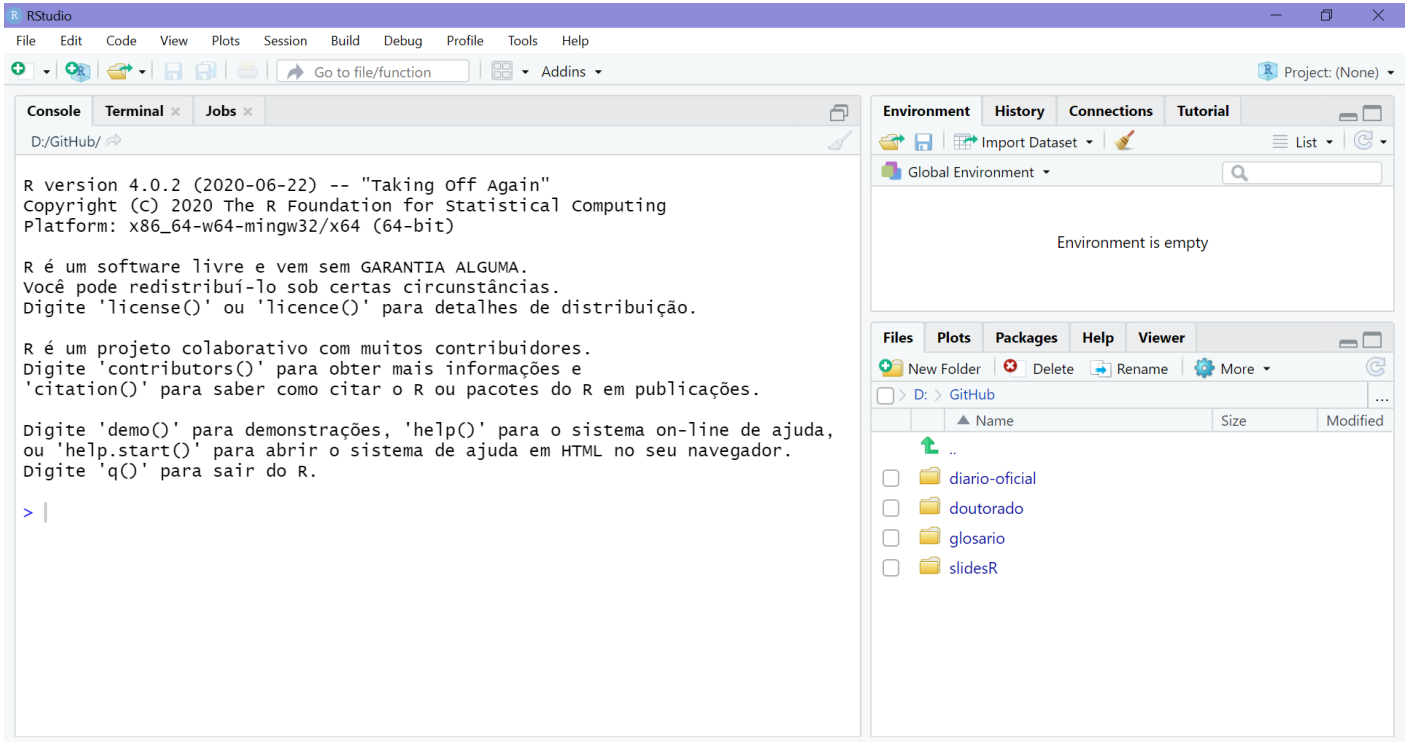
O que é o R?

> "R é um ambiente de software livre para computação estatística e gráficos". (https://www.r-project.org/)

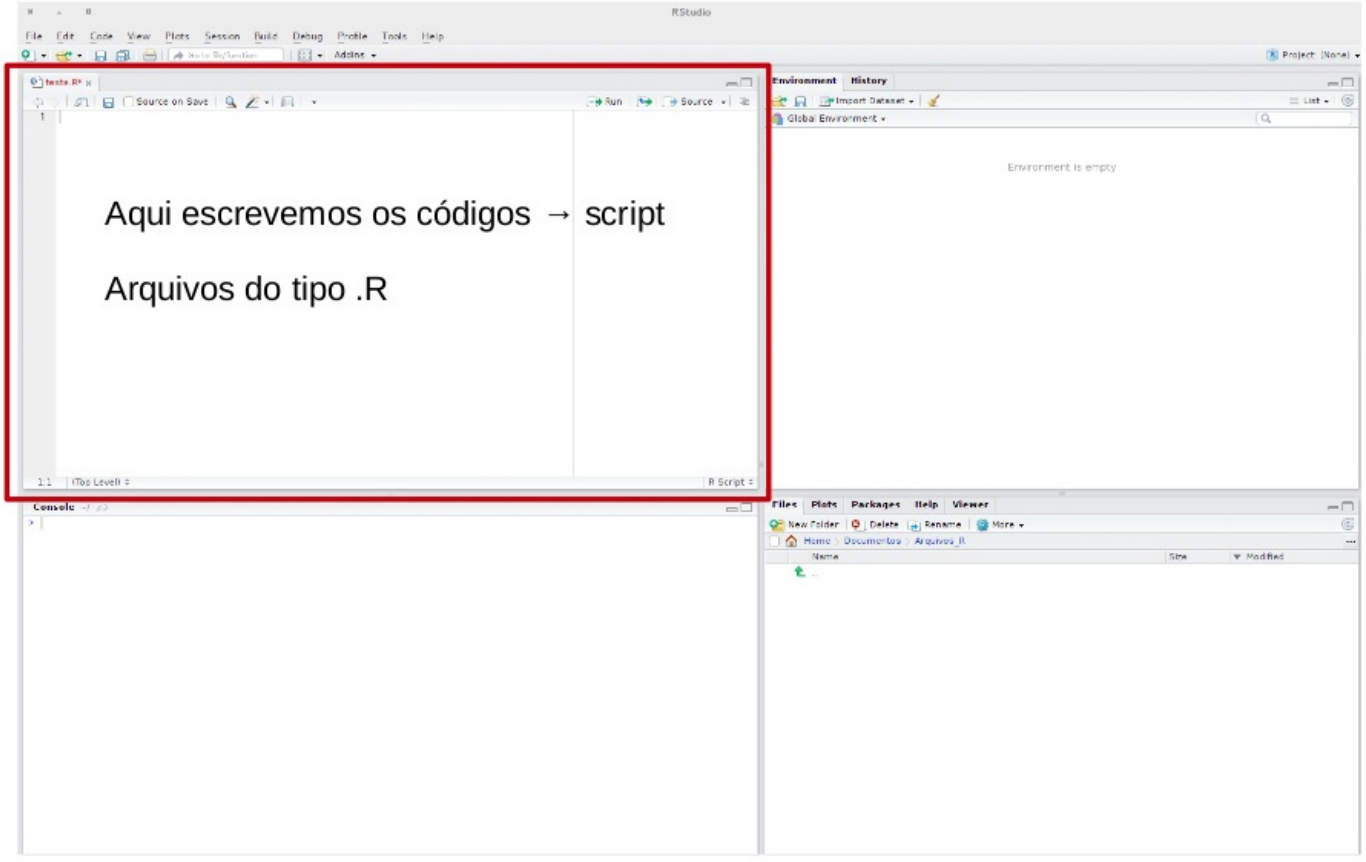
- **Por que usar o R**?
  - É uma linguagem de programação que possui muitas ferramentas para análise de dados
  - É *código aberto* (open source)
  - Possui uma comunidade ativa de pessoas desenvolvedoras
  - É flexível, permite desenvolver funções e pacotes para facilitar o trabalho
  - Está disponível, gratuitamente, em diferentes plataformas: Windows, Linux e Mac
  - Mantido pela R Development Core Team

RStudio

RStudio é uma IDE (*integrated development environment*) da Linguagem R, ou seja, um ambiente de desenvolvimento que utilizamos para editar e executar os códigos em R.

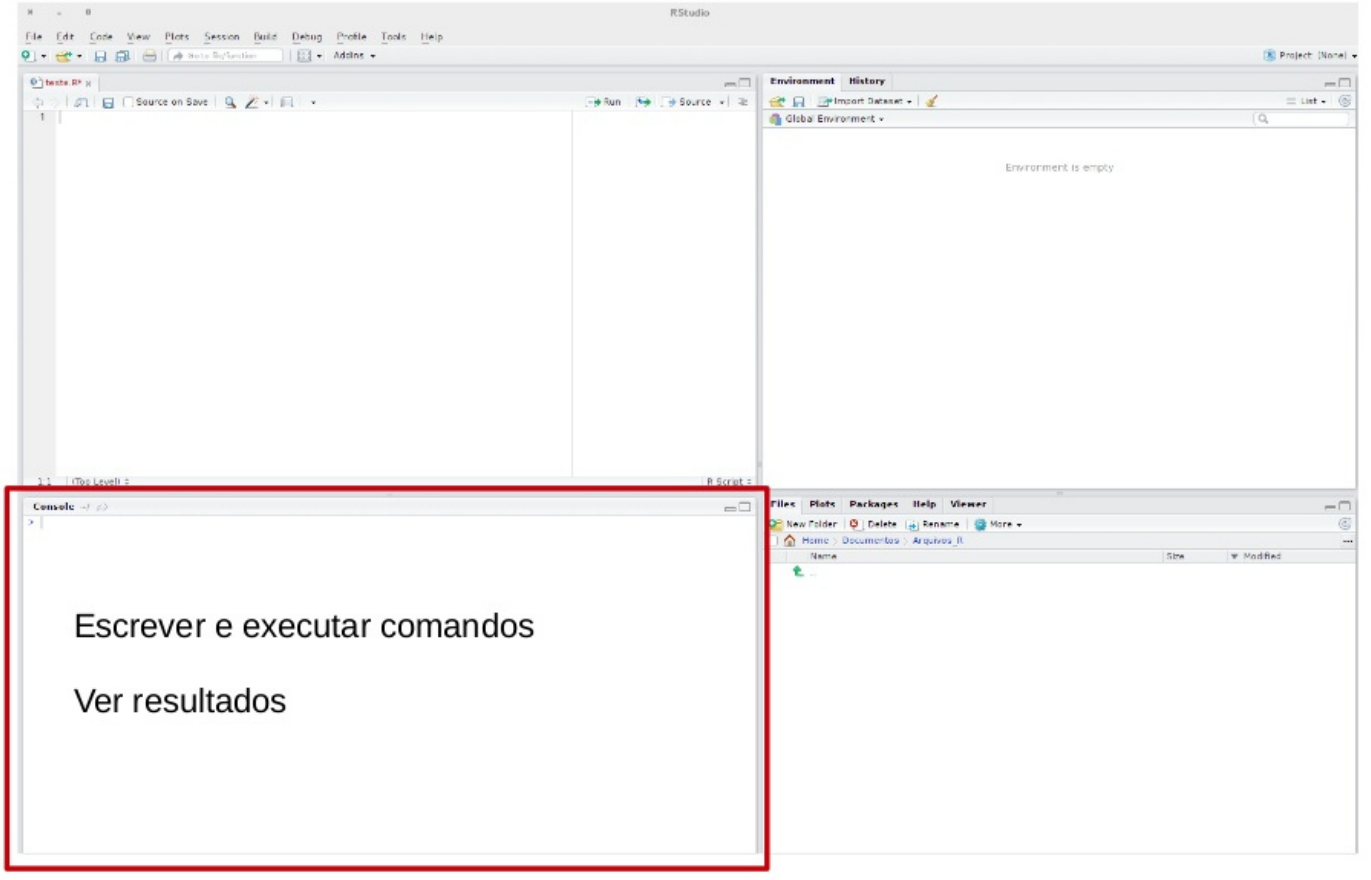


RStudio



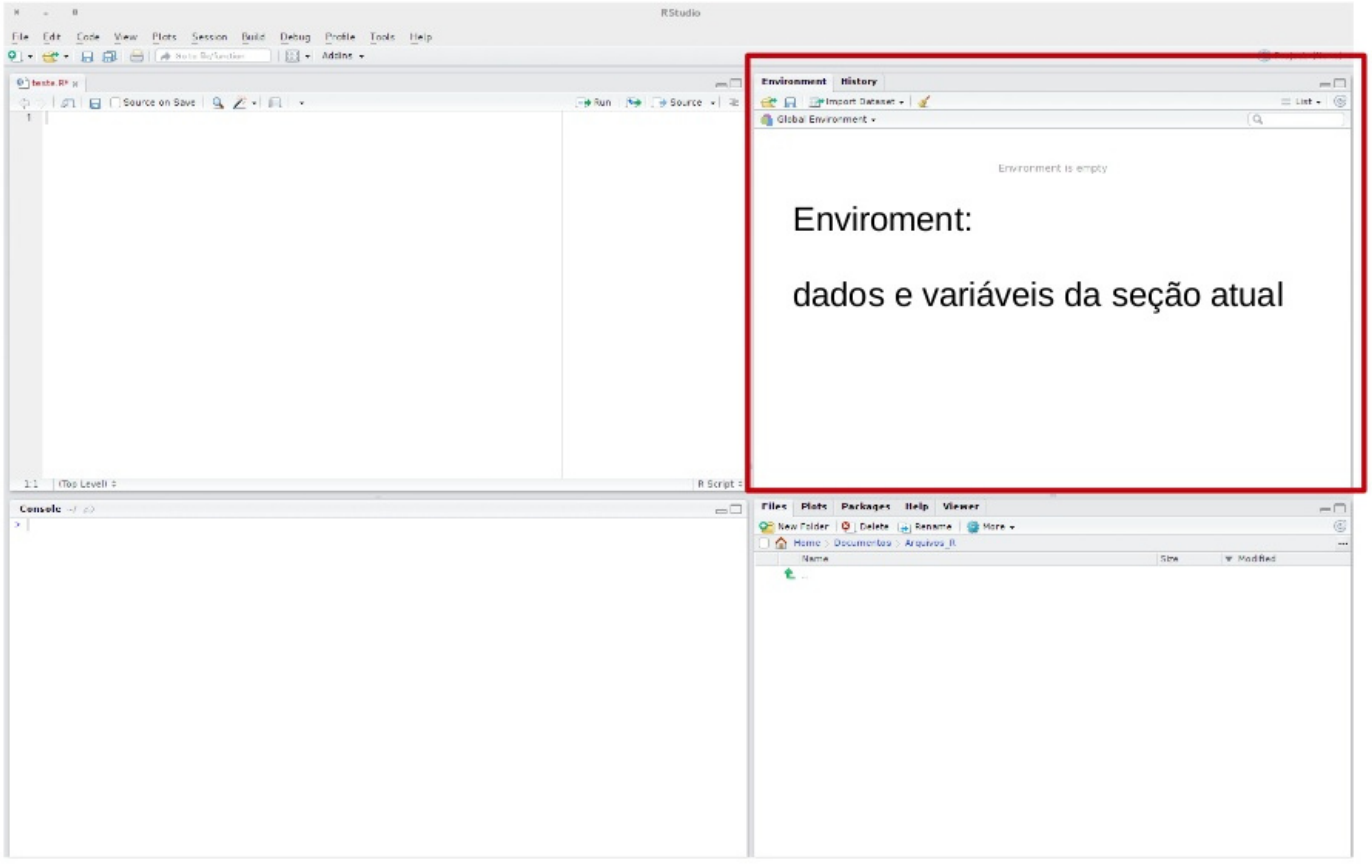
Fonte: Haydee Svab

RStudio



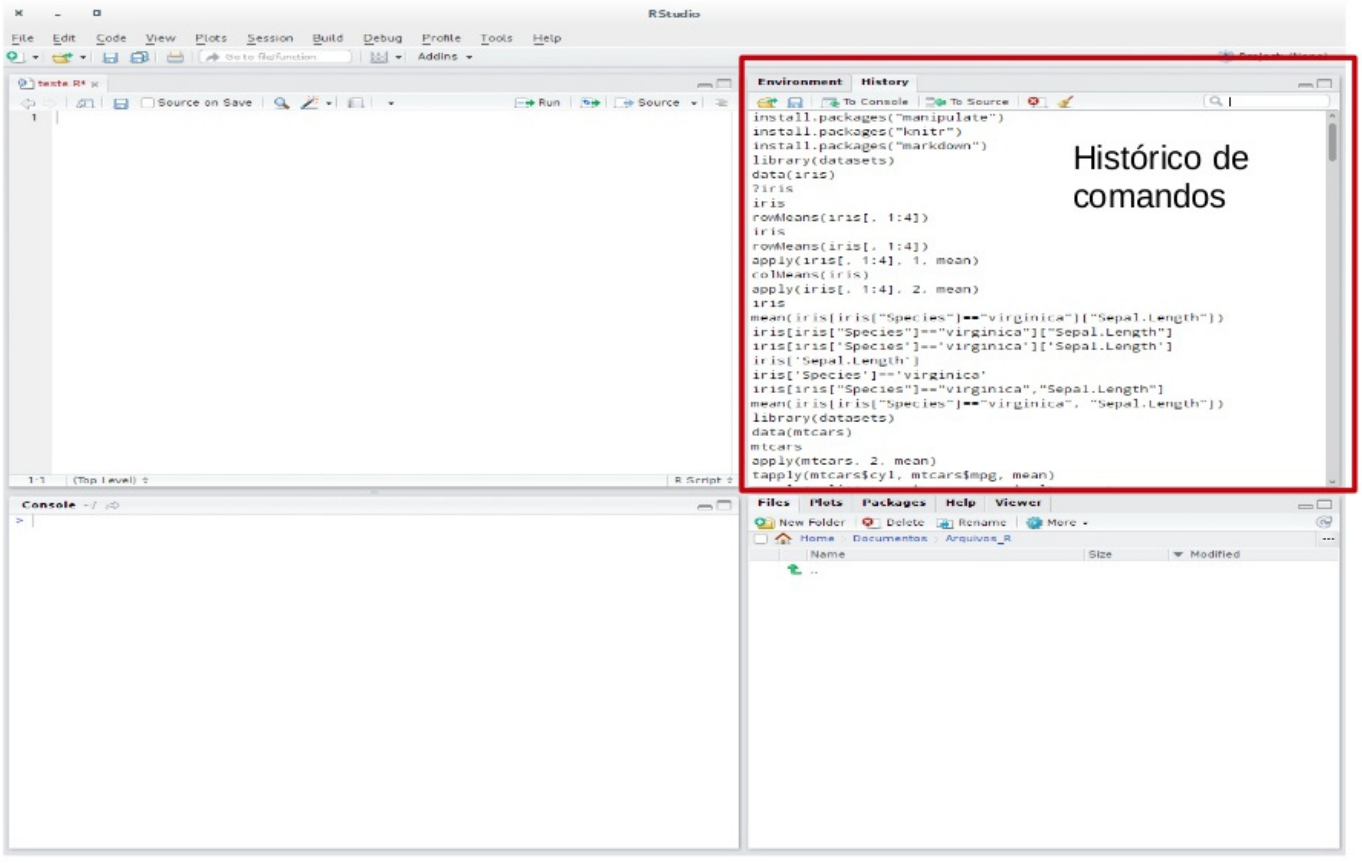
Fonte: Haydee Svab

RStudio



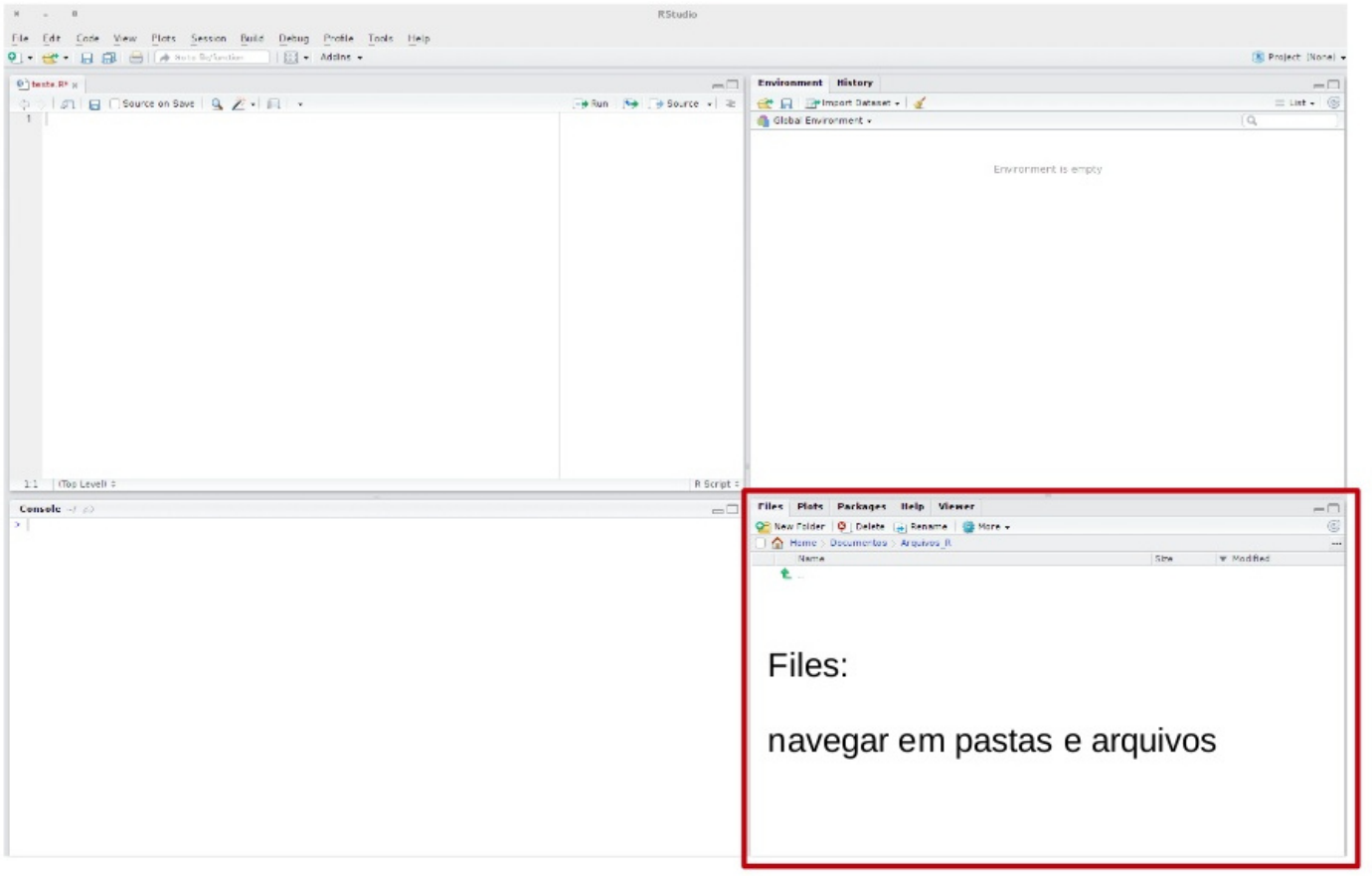
Fonte: Haydee Svab

RStudio



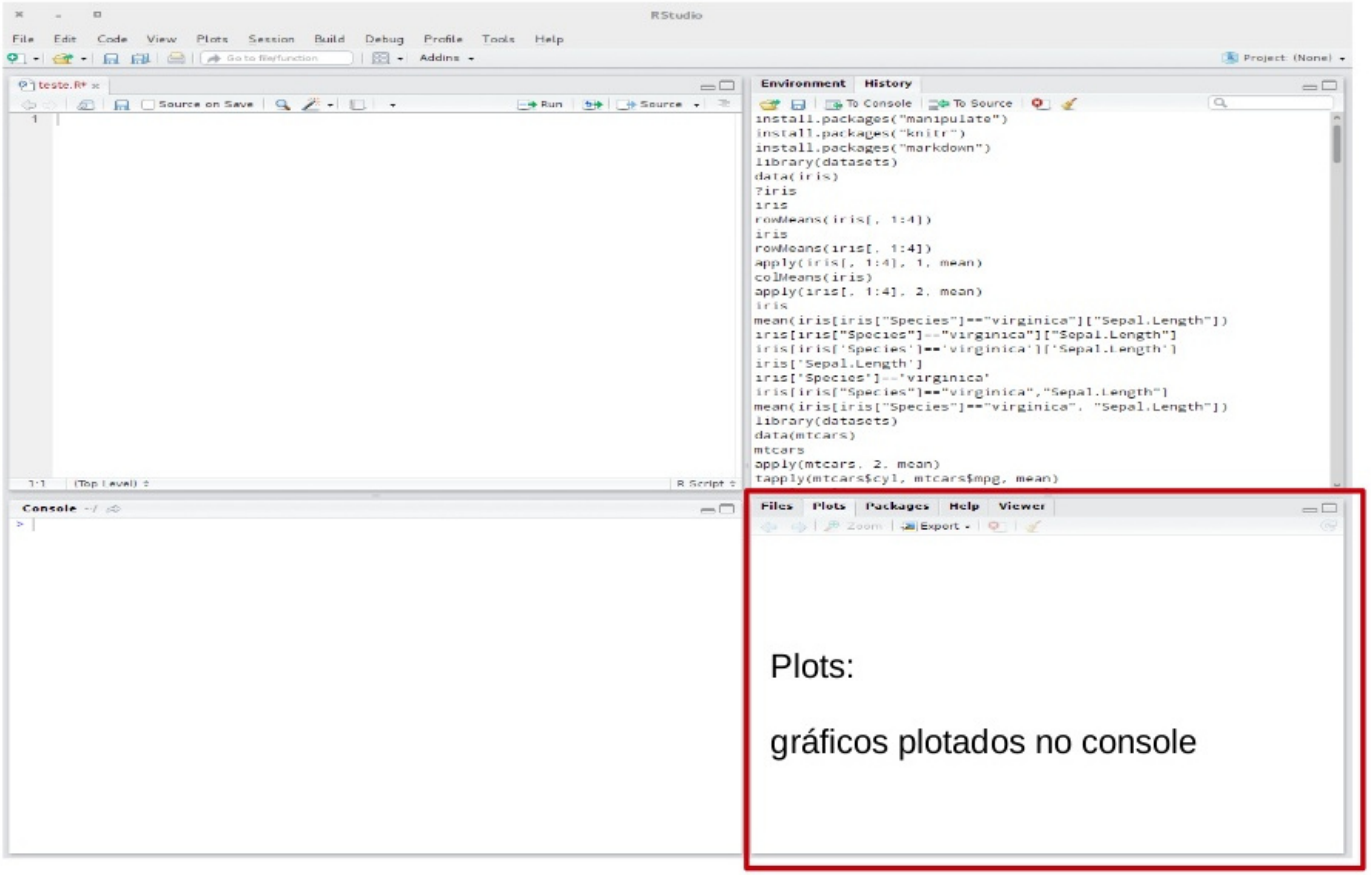
Fonte: Haydee Svab

RStudio



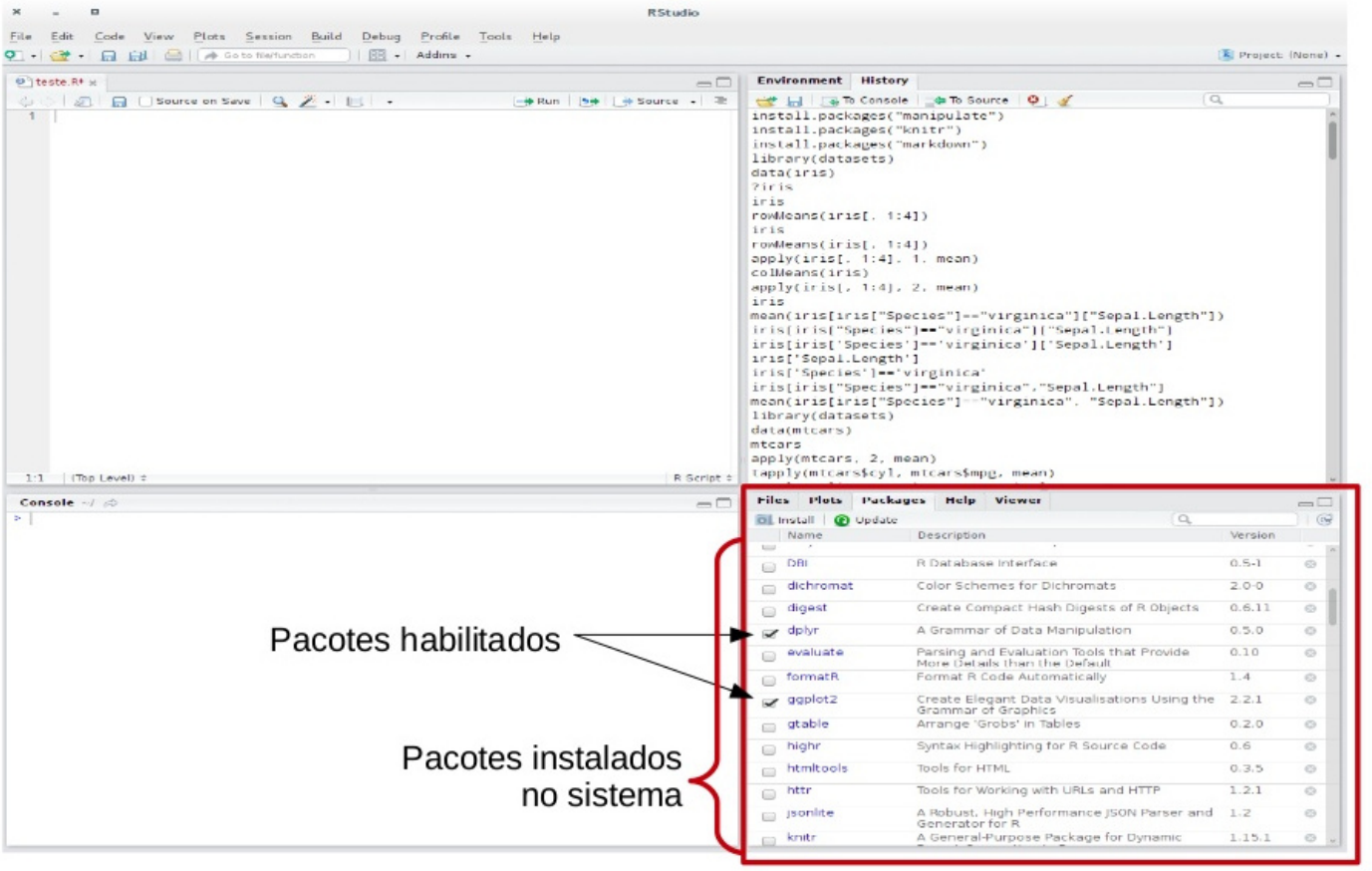
Fonte: Haydee Svab

RStudio



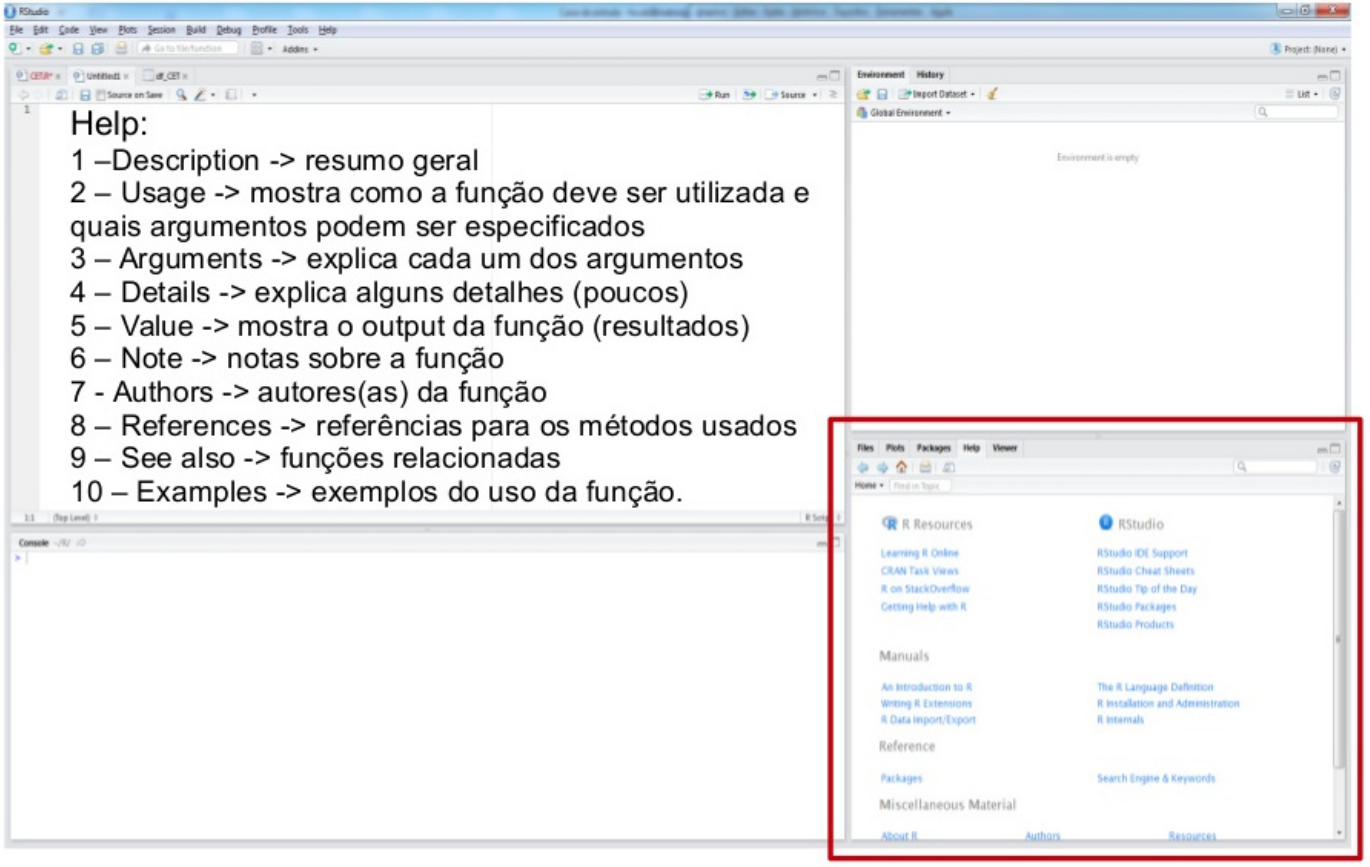
Fonte: Haydee Svab

RStudio



Fonte: Haydee Svab

RStudio



Fonte: Haydee Svab

Help!

- Pedir ajuda: **help**(nome_da_funcao) ou **?**nome_da_funcao.

```
help(sum)
?sum
```

- Se a dúvida permanecer, procure no Stack OverFlow ou Google.
- E se ainda tiver dúvidas, pergunte para a comunidade (há grupos no Telegram e outras redes sociais).

Começando!

(Acompanhe no arquivo `1-introducao.R` 🤖)

00:30

Começando

- Abrir o `RStudio Cloud`.
- Você pode criar um novo `R` script em: `file -> new file -> R script`
- Utilizaremos o arquivo que está na cloud: `1-introducao.R`. Abra-o na aba `Files`.

R como calculadora

- O R permite realizar muitas operações matemáticas!

```
2 + 5    # adição
```

```
## [1] 7
```

```
9 - 4    # subtração
```

```
## [1] 5
```

```
5 * 2    # multiplicação
```

```
## [1] 10
```

```
7 / 5    # divisão
```

```
## [1] 1.4
```

- CTRL + ENTER: executa a linha selecionada no script.

- **R como calculadora**


```
9 %% 4   # resto da divisão de 9 por 4
```

```
## [1] 1
```

```
7 %/% 4  # parte inteira da divisão de 7 por 4
```

```
## [1] 1
```

```
8 ^ 2    # potenciação
```

```
## [1] 64
```

```
sqrt(1024) # radiciação
```

```
## [1] 32
```

A ordem matemática das operações também vale no R.

Funções matemáticas


```
sin(1)  # funções trigonométricas
```

```
## [1] 0.841471
```

```
log(1)  # logaritmo natural (base e)
```

```
## [1] 0
```

```
log10(10) # logaritmo na base 10
```

```
## [1] 1
```

```
exp(0.5) # e^(1/2)
```

```
## [1] 1.648721
```

Fonte: SW Carpentry

O que é um objeto?

- Ao se desenvolver um projeto, você irá trabalhar com diversos tipos de arquivos, além de informações que serão repetidas ao longo do script.
- Para reutilizar essas informações ao longo do script utilizamos o que chamamos de **objeto**
- Um objeto retém e representa um valor, função ou base de dados

Atribuindo valor a um objeto no R

- Para atribuir um valor a um objeto no R, utilizamos o operador:

`<-`

```
nome_do_objeto <- valor
```

- Atalho: `ALT` + `-`: cria o `<-` sinal de atribuição.

Exemplo:

```
minha_idade <- 27
minha_idade
```

```
## [1] 27
```

Exemplo:

```
nome <- "Daenerys Targaryen"
nome
```

```
## [1] "Daenerys Targaryen"
```

```
horas_trabalhadas <- 160
horas_trabalhadas
```

```
## [1] 160
```

```
salario <- 3984.23
salario
```

```
## [1] 3984.23
```

```
ativo <- TRUE
ativo
```

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

Nomes de objetos

- Os nomes devem começar com uma letra. Podem conter letras, números, _ e .
- Não usar acentuação e espaços nos nomes de objetos.
- Recomendação do autor do livro `R` For Data Science: **usar_snake_case**, ou seja, palavras escritas em minúsculo separadas pelo underscore (_).
- O `R` é *case sensitive*, isto é, faz a diferenciação entre as letras minúsculas e maiúsculas. Portanto, um objeto chamado `teste` é diferente de uma outro objeto chamada `Teste`.

Funções

- Funções permitem **automatizar tarefas** comuns de forma mais poderosa do que copiar e colar.
- O R possui muitas funções já implementadas.
- Pacotes são coleções de funções, dados e documentação que ampliam as capacidades do R básico. Veremos como instalá-los no final desta atividade!
- Você pode desenvolver suas próprias funções! 

Exemplo de funções básicas do R


```r
# Combinar elementos - Função c()
ano_nascimento_irmaos <- c(1993, 1998, 2001, 2012, 2012)
```

```r
# Podemos fazer operações com o resultado
idade_irmaos <- 2020 - ano_nascimento_irmaos
idade_irmaos
```

```
## [1] 27 22 19  8  8
```

```r
# Calculando a média - Função mean()
media_idade_irmaos <- mean(idade_irmaos)
media_idade_irmaos
```

```
## [1] 16.8
```

Estrutura de uma função

- **Nome** - é como ela fica salva no ambiente, esse nome é importante para usarmos a função.
- **Argumentos** - são parâmetros usados internamente pela função. Muitas funções possuem argumentos com valores padrão.
- **Corpo** - o código que será executado. O resultado dependerá dos argumentos oferecidos.

```
# para criar uma funcao
nome_da_funcao <- function(argumentos){
  corpo da função
}
# para usar essa função
nome_da_funcao(argumentos = ...)
```

Exemplo sobre argumentos


```
# Arredondar valores - função round()
# help(round)
# round(x, digits = 0)

round(media_idade_irmaos)
```

```
## [1] 17
```

```
round(media_idade_irmaos, digits = 1)
```

```
## [1] 16.8
```

Funções importantes do R base

Função	O que retorna?
`sum()`	Soma
`mean()`	Média
`median()`	Mediana
`var()`	Variância (simples)
`sd()`	Desvio Padrão
`max()`	Valor máximo
`min()`	Valor mínimo
`round()`	Valor arredondado

Operadores Relacionais e Lógicos

(Acompanhe no arquivo `2-operadores.R` )

00:30

Operadores Relacionais

- Igual a: **==**
- Diferente de: **!=**
- Maior que: **>**
- Maior ou igual: **>=**
- Menor que: **<**
- Menor ou igual: **<=**

Operadores Relacionais

Igual a: **==**

```
TRUE == TRUE
```

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

```
TRUE == FALSE
```

```
## [1] FALSE
```

Operadores Relacionais

Diferente de **!=**

```
TRUE != TRUE
```

```
## [1] FALSE
```

```
TRUE != FALSE
```

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

Operadores Relacionais

Menor que: <

```
3 < 5
```

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

Maior ou igual a: >=

```
10 >= 10
```

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

Maior que: >=

```
10 > 10
```

```
## [1] FALSE
```

Operadores Lógicos

- AND - E: **&**

Será verdadeiro se os dois forem verdadeiros (TRUE)

```
x <- 5

 x >= 3 & x <=7
```

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

```
y <- 2

 y >= 3 & y <= 7
```

```
## [1] FALSE
```

Operadores Lógicos

- OR - OU: |

Será verdadeiro se um dos dois forem verdadeiros (TRUE)

```
y <- 2

 y >= 3 | y <=7
```

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

```
y <- 1

 y >= 3 | y == 0
```

```
## [1] FALSE
```

Operadores Lógicos

- NOT - Negação: `!`

```
!TRUE
```

```
## [1] FALSE
```

```
!FALSE
```

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

```
x <- 5
(!x < 4)
```

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

Tipos básicos de dados

(Acompanhe no arquivo `3-tipos_dados.R` 🤖)

00:30

Tipos básicos de dados

São os tipos básicos de dados que podem ser representados na linguagem R. É neles que guardamos as informações que necessitamos para um algoritmo.

- **Integer**: números inteiros
- **Double/Numeric**: números racionais
- **Logical**: tipos lógicos, `TRUE` ou `FALSE`
- **Character**: texto (sempre entre aspas)
- **Factor**: dados categóricos
- A função `class()` retorna o tipo do dado.

Números

- Um número inteiro seguido de `L` será considerado do tipo `integer`:

```
class(3L)
```

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

- Números racionais serão considerados como `double/numeric`:

```
class(3)
```

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

```
class(3.1)
```

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

Lógicos

- Verdadeiro (`TRUE`) ou Falso (`FALSE`)

```
class(TRUE)
```

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

- O R entende `TRUE` sendo igual a um, e `FALSE` sendo igual a zero.
- Isso significa que podemos fazer operações matemáticas com eles (como por exemplo somar):

```
TRUE + TRUE + TRUE + FALSE
```

```
## [1] 3
```

Textos

- **Qualquer** código entre aspas será interpretado como texto (`character`):

```
class("TEXTO")
```

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

```
escola <- c("Fundamental", "Médio", "Superior")

class(escola)
```

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

```
class("3")
```

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

Fatores


```
# Criando factor

escola_categorias <- factor(c("Fundamental", "Médio", "Superior"))

escola_categorias
```

```
## [1] Fundamental Médio       Superior
## Levels: Fundamental Médio Superior
```

```
class(escola_categorias)
```

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

A função `as.factor()` cria um objeto do tipo factor, ou converte um objeto existente.

Na linha `levels` aparecem os rótulos do fator.

Exercícios

1) Primeiro, tente adivinhar o tipo de dado dos objetos abaixo:

```
cor_favorita <- "rosa"
idade <- 27L
altura <- "1.75"
peso <- 61.1
gosta_brocolis <- TRUE
gosta_carne <- "FALSE"
```

2) Use a função `class()` e descubra qual é o tipo dos objetos acima.

Resposta do exercício:

```
class(cor_favorita)
```

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

```
class(idade)
```

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

```
class(altura)
```

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

```
class(peso)
```

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

```
class(gosta_brocolis)
```

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

```
class(gosta_carne)
```

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

NA

Uma característica importante do R que pode dificultar a comparação são os valores ausentes ou **NAs** (não disponíveis).

NA representa um valor desconhecido.

NA

- Operações envolvendo um valor desconhecido também será desconhecido:

```
NA > 10
```

```
## [1] NA
```

```
10 == NA
```

```
## [1] NA
```

```
NA + 10
```

```
## [1] NA
```

```
NA / 2
```

```
## [1] NA
```

NA


```
NA == NA
```

```
## [1] NA
```

NA

`is.na()` é a função que testa se um objeto é NA.

```
vetor_numerico <- c(NA, 1, 5, 2, 5, NA)

is.na(vetor_numerico)
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE  TRUE
```

```
!is.na(vetor_numerico)
```

```
## [1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE
```

Argumento importante:

`na.rm = TRUE`

```
sum(vetor_numerico)
```

```
## [1] NA
```

```
sum(vetor_numerico, na.rm = TRUE)
```

```
## [1] 13
```

```
mean(vetor_numerico)
```

```
## [1] NA
```

```
mean(vetor_numerico, na.rm = TRUE)
```

```
## [1] 3.25
```

Conversão de classes

Podemos alterar o tipo de dado de um objeto com as funções iniciadas com `as.`:

- `as.numeric()`
- `as.integer()`
- `as.logical()`
- `as.character()`
- `as.factor()`

Conversão de classes

Exemplos de conversão de classes

```
vetor_logical <- c(TRUE, FALSE, TRUE, FALSE)

as.integer(vetor_logical)
```

```
## [1] 1 0 1 0
```

```
as.numeric(vetor_logical)
```

```
## [1] 1 0 1 0
```

```
as.character(vetor_logical)
```

```
## [1] "TRUE"  "FALSE" "TRUE"  "FALSE"
```

```
as.factor(vetor_logical)
```

```
## [1] TRUE  FALSE TRUE  FALSE
## Levels: FALSE TRUE
```

Conversão de classes

Exemplos de conversão de classes

```
frutas <- c("banana", "maça", "melancia")

as.integer(frutas)
```

```
## [1] NA NA NA
```

```
as.numeric(frutas)
```

```
## [1] NA NA NA
```

```
as.character(frutas)
```

```
## [1] "banana"   "maça"     "melancia"
```

```
as.factor(frutas)
```

```
## [1] banana   maça     melancia
## Levels: banana maça melancia
```

Tipos de objetos

(Acompanhe no arquivo `4-tipos_objetos.R` 🤖)

00:30

Tipos de objetos

- No R, os 4 principais tipos de dados são: `vetor`, `matriz`, `lista`, `data.frame`
- Vamos focar em dois: `vetor` e `data.frame`, pois:
  - são os que mais usamos nas tarefas comuns de análise de dados,
  - são conceitos importantes para utilizar o tidyverse.

Vetores

- Armazena elementos de mesma classe, apenas uma dimensão.
- São criados usando a função `c()`
- Exemplo:

```
primeiro_semestre <- c("Janeiro", "Fevereiro", "Março",
                       "Abril", "Maio", "Junho")
```

```
# Retorna o comprimento do vetor - quantos elementos ele tem?
length(primeiro_semestre)
```

```
## [1] 6
```

Vetores

- Selecionando elementos por índice: utilizar `[]`

```
# primeiro elemento do vetor
primeiro_semestre[1]
```

```
## [1] "Janeiro"
```

```
# primeiro elemento até o segundo
primeiro_semestre[1:2]
```

```
## [1] "Janeiro"   "Fevereiro"
```

```
# remove o elemento 1
primeiro_semestre[-1]
```

```
## [1] "Fevereiro" "Março"     "Abril"     "Maio"      "Junho"
```

```
# seleciona o elemento seis até o quatro
# (e muda a ordem dos elementos)
primeiro_semestre[6:4]
```

```
## [1] "Junho" "Maio"  "Abril"
```

Dataframes

- Possuem duas dimensões: linhas e colunas.
- Cada colunas pode ser de classes diferentes.
- Pense em uma tabela, como está acostumada a ver no Excel por exemplo.

Exemplo: Base de dados disponível no R Community Explorer.

city	country	region	members
Warsaw	Poland	Europe	2556
Santiago	Chile	Latin America	2060
Washington	USA	US/Canada	1996
Istanbul	Turkey	Europe	1891
New York	USA	US/Canada	1868
Taipei	Taiwan	Asia	1662

Funções úteis


```
# Retorna o número de colunas
ncol(rladies)
```

```
## [1] 12
```

```
# Retorna o número de linhas
nrow(rladies)
```

```
## [1] 192
```

```
# Retorna o número de colunas e linhas
dim(rladies)
```

```
## [1] 192  12
```

Funções úteis


```r
# Retorna algumas informações sobre a base
str(rladies)
```

```
## tibble [192 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ X1             : num [1:192] 1 2 3 4 5 6 7 8 9 10 ...
##  $ name           : chr [1:192] "R-Ladies Barcelona" "R-Ladies Bilbao" "R-L
##  $ city           : chr [1:192] "Barcelona" "Bilbao" "Tucson" "San Carlos d
##  $ country        : chr [1:192] "Spain" "Spain" "USA" "Argentina" ...
##  $ region         : chr [1:192] "Europe" "Europe" "US/Canada" "Latin Americ
##  $ members        : num [1:192] 558 176 280 231 1582 ...
##  $ fullurl        : chr [1:192] "https://www.meetup.com/rladies-barcelona/"
##  $ created        : Date[1:192], format: "2016-10-22" "2019-02-27" ...
##  $ status         : chr [1:192] "Active" "Inactive" "Active" "Active" ...
##  $ last_event     : Date[1:192], format: "2020-07-01" "2020-01-14" ...
##  $ past_events    : num [1:192] 18 4 18 12 57 7 7 10 4 12 ...
##  $ upcoming_events: num [1:192] 0 0 2 0 1 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   X1 = col_double(),
##   ..   name = col_character(),
##   ..   city = col_character(),
##   ..   country = col_character(),
##   ..   region = col_character(),
##   ..   members = col_double(),
##   ..   fullurl = col_character(),
##   ..   created = col_date(format = ""),
```

Funções úteis


```
summary(rladies)
```

```
##        X1             name               city             country
##  Min.   :  1.00   Length:192         Length:192         Length:192
##  1st Qu.: 48.75   Class :character   Class :character   Class :character
##  Median : 96.50   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 96.50
##  3rd Qu.:144.25
##  Max.   :192.00
##
##     region             members          fullurl             created
##  Length:192         Min.   :   2.0   Length:192         Min.   :2012-10-01
##  Class :character   1st Qu.:  76.0   Class :character   1st Qu.:2017-10-03
##  Mode  :character   Median : 237.5   Mode  :character   Median :2018-09-14
##                     Mean   : 371.7                      Mean   :2018-08-19
##                     3rd Qu.: 481.0                      3rd Qu.:2019-09-04
##                     Max.   :2556.0                      Max.   :2020-08-15
##
##     status           last_event          past_events     upcoming_events
##  Length:192         Min.   :2019-03-05   Min.   : 0.00   Min.   :0.000
##  Class :character   1st Qu.:2020-02-14   1st Qu.: 3.00   1st Qu.:0.000
##  Mode  :character   Median :2020-07-01   Median : 8.50   Median :0.000
##                     Mean   :2020-04-29   Mean   :13.35   Mean   :0.224
##                     3rd Qu.:2020-07-31   3rd Qu.:18.00   3rd Qu.:0.000
##                     Max.   :2020-08-20   Max.   :77.00   Max.   :5.000
##                     NA's   :21
```

Funções úteis

- Eu gosto da função glimpse, do pacote `tibble`:

```
tibble::glimpse(rladies)
```

```
## Rows: 192
## Columns: 12
## $ X1              <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
## $ name            <chr> "R-Ladies Barcelona", "R-Ladies Bilbao", "R-Ladies
## $ city            <chr> "Barcelona", "Bilbao", "Tucson", "San Carlos de Bar
## $ country         <chr> "Spain", "Spain", "USA", "Argentina", "Australia",
## $ region          <chr> "Europe", "Europe", "US/Canada", "Latin America", "
## $ members         <dbl> 558, 176, 280, 231, 1582, 346, 511, 800, 59, 197, 6
## $ fullurl         <chr> "https://www.meetup.com/rladies-barcelona/", "https
## $ created         <date> 2016-10-22, 2019-02-27, 2017-11-14, 2018-05-10, 20
## $ status          <chr> "Active", "Inactive", "Active", "Active", "Active",
## $ last_event      <date> 2020-07-01, 2020-01-14, 2020-08-19, 2020-06-29, 20
## $ past_events     <dbl> 18, 4, 18, 12, 57, 7, 7, 10, 4, 12, 23, 37, 9, 0, 3
## $ upcoming_events <dbl> 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
```

Funções úteis


```
# Primeiras 6 linhas de uma tabela
head(rladies)
```

```
## # A tibble: 6 x 12
##      X1 name  city  country region members fullurl created    status last_e
##   <dbl> <chr> <chr> <chr>   <chr>    <dbl> <chr>   <date>     <chr>  <date>
## 1     1 R-La~ Barc~ Spain   Europe     558 https:~ 2016-10-22 Active 2020-0
## 2     2 R-La~ Bilb~ Spain   Europe     176 https:~ 2019-02-27 Inact~ 2020-0
## 3     3 R-La~ Tucs~ USA     US/Ca~     280 https:~ 2017-11-14 Active 2020-0
## 4     4 R-La~ San ~ Argent~ Latin~     231 https:~ 2018-05-10 Active 2020-0
## 5     5 R-La~ Melb~ Austra~ Austr~    1582 https:~ 2016-09-02 Active 2020-0
## 6     6 R-La~ Port~ Brazil  Latin~     346 https:~ 2017-10-30 Active 2020-0
## # ... with 2 more variables: past_events <dbl>, upcoming_events <dbl>
```

Funções úteis


```
# Últimas 6 linhas de uma tabela
tail(rladies)
```

```
## # A tibble: 6 x 12
##      X1 name  city  country region members fullurl created    status last_e
##   <dbl> <chr> <chr> <chr>   <chr>    <dbl> <chr>   <date>     <chr>  <date>
## 1   187 R-La~ Walt~ USA     US/Ca~      15 https:~ 2020-07-03 Unbeg~ NA
## 2   188 R-La~ Conc~ Chile   Latin~     357 https:~ 2019-04-16 Active 2020-0
## 3   189 R-La~ Talca Chile   Latin~      52 https:~ 2020-05-14 Active 2020-0
## 4   190 R-La~ Adel~ Austra~ Austr~     307 https:~ 2017-02-02 Inact~ 2019-1
## 5   191 R-La~ Bogo~ Colomb~ Latin~     276 https:~ 2017-09-22 Inact~ 2019-0
## 6   192 Dead~ Bend  USA     US/Ca~      95 https:~ 2019-08-11 Active 2020-0
## # ... with 2 more variables: past_events <dbl>, upcoming_events <dbl>
```

Selecionando elementos - Colunas

- Para selecionar colunas, utilize o `$`:

Dica: após escrever o `nome_do_dataframe$`, aperte `tab` para que o RStudio faça sugestões de colunas da base.

```
rladies$city
```

```
##   [1] "Barcelona"                "Bilbao"
##   [3] "Tucson"                   "San Carlos de Bariloche"
##   [5] "Melbourne"                "Porto Alegre"
##   [7] "Guayaquil"                "Quito"
##   [9] "Kansas City"              "Canberra"
##  [11] "Johannesburg"             "Madrid"
##  [13] "Frankfurt"                "Houston"
##  [15] "Puebla"                   "Xalapa"
##  [17] "East Lansing"             "San Diego"
##  [19] "Freiburg"                 "Riverside"
##  [21] "Nijmegen"                 "Lyon"
##  [23] "Lausanne"                 "Belgrade"
##  [25] "Novi Sad"                 "Bucharest"
##  [27] "Vienna"                   "London"
##  [29] "Salvador"                 "Kyiv"
##  [31] "Highland Park"            "Athens"
##  [33] "Perth"                    "Brisbane"
##  [35] "Santa Barbara"            "Auckland"
```

Selecionando elementos

- Outra forma de selecionar elementos é utilizando os colchetes: `data_frame[numero da linha , numero da coluna]`!

```
rladies[1, 2] # Seleciona a linha 1 e a coluna 2
```

```
## # A tibble: 1 x 1
##   name
##   <chr>
## 1 R-Ladies Barcelona
```

Selecionando elementos

- Caso deixe algum dos espaços dentro do colchete vazio, o R retornará todas (as colunas ou linhas):

```
rladies[1, ] # Seleciona a linha 1 e TODAS as colunas
```

```
## # A tibble: 1 x 12
##       X1 name  city  country region members fullurl created    status last_e
##    <dbl> <chr> <chr> <chr>   <chr>    <dbl> <chr>   <date>     <chr>  <date>
## 1      1 R-La~ Barc~ Spain   Europe     558 https:~ 2016-10-22 Active 2020-0
## # ... with 2 more variables: past_events <dbl>, upcoming_events <dbl>
```

```
rladies[ , 2] # Seleciona TODAS as linhas e apenas a coluna 2
```

```
## # A tibble: 192 x 1
##    name
##    <chr>
##  1 R-Ladies Barcelona
##  2 R-Ladies Bilbao
##  3 R-Ladies Tucson AZ
##  4 R-Ladies Bariloche
##  5 R-Ladies Melbourne
##  6 R-Ladies Porto Alegre
##  7 R-Ladies Guayaquil
##  8 R-Ladies Quito
##  9 R-Ladies Kansas City
```

Dataframes e funções

- Podemos utilizar as funções que já vimos (e muitas outras) em colunas do dataframe:

```
# Qual é a soma de pessoas membras na plataforma do Meetup inscritas
# nos capítulos da R-Ladies?
sum(rladies$members)
```

```
## [1] 71361
```

```
# Menor valor encontrado: o menor número de pessoas membras
# encontrado na base
min(rladies$members)
```

```
## [1] 2
```

```
# Maior valor encontrado: o maior número de pessoas membras
# encontrado na base
max(rladies$members)
```

```
## [1] 2556
```

Dataframes e funções

- Podemos utilizar as funções que já vimos (e muitas outras) em colunas do dataframe:

```
# Média do número pessoas membras por capítulo
mean(rladies$members)
```

```
## [1] 371.6719
```

```
# Mediana do número pessoas membras por capítulo
median(rladies$members)
```

```
## [1] 237.5
```

```
# Variância do número pessoas membras por capítulo
var(rladies$members)
```

```
## [1] 197985.6
```

```
# Desvio padrão do número pessoas membras por capítulo
sd(rladies$members)
```

```
## [1] 444.9557
```

Pacotes no R

(Acompanhe no arquivo `5-pacotes.R` )

00:30

Pacotes no R

Pacotes são coleções de funções, dados e documentação que estendem as capacidades do R básico.

Eles precisam ser instalados e carregados.

Instalação de Pacotes:

- Via CRAN:

```
install.packages("nome-do-pacote")
```

```
install.packages("tidyverse")
```

- Via GitHub:

```
devtools::install_github("nome-da-org/nome-do-repo)
```

```
devtools::install_github("tidyverse/dplyr")
```

Carregar pacotes:

- Função: `library(nome-do-pacote)`

```
library(tidyverse)
```

Dicas sobre Pacotes

1. Você só precisa instalar o pacote uma vez, mas precisa carregá-lo sempre que começar uma nova sessão;

2. Para instalar o pacote use as aspas;

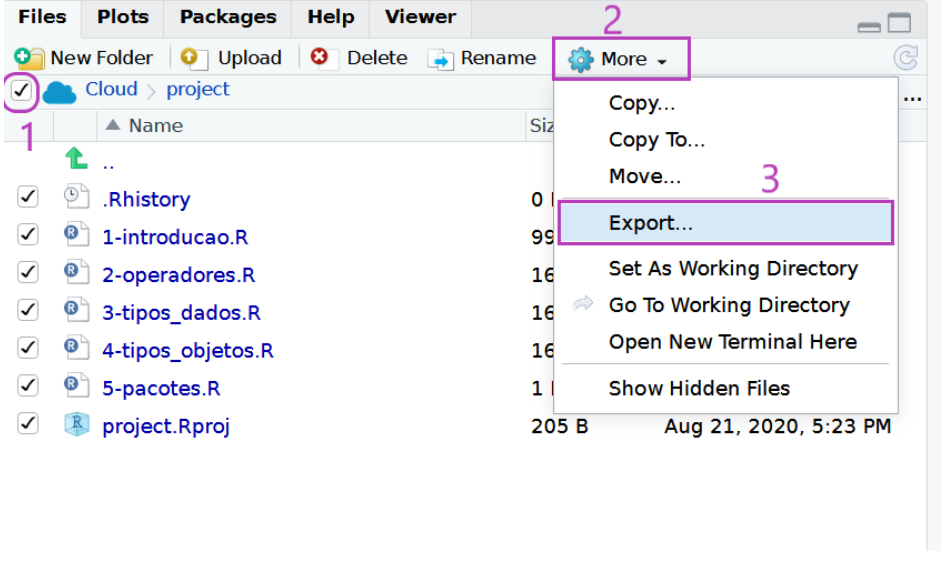
3. Para carregar o pacote, não é necessário utilizar aspas.


Ilustração por Allison Horst
stringr
forcats
Ladies
tidyverse
www.rstudio.com
dplyr
tidyr
UNOPS
lubridate
TIBBLE
www.rstudio.com
purrr

Download do material

Ao final, se quiser fazer download dos arquivos que usamos, é só realizar o procedimento abaixo:

1. Na aba `Files`, selecionar todos os arquivos do projeto, clicando na caixinha ao lado de ☁ `Cloud`
2. Clique em ⚙ `More`
3. Selecione a opção `Export...`
4. Um arquivo `.zip` será baixado com os arquivos 😃



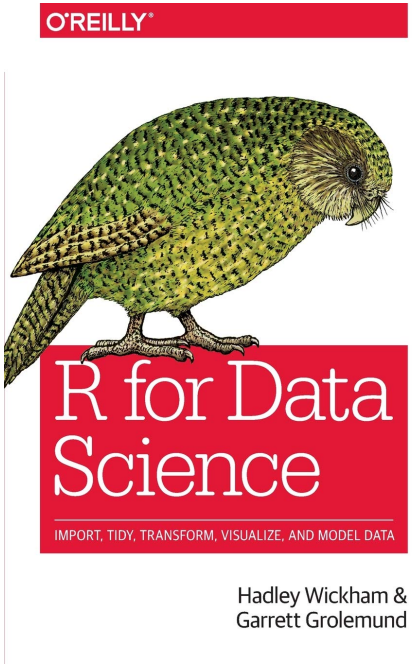
Aprendendo mais!

Em Português

- Material da Curso-R

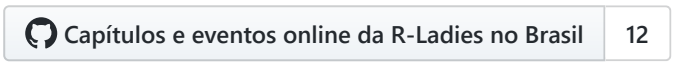
Em Inglês

- Livro R for Data Science



R-Ladies no Brasil

- Quer saber mais sobre próximos eventos das R-Ladies por aqui? Favorite este repositório no GitHub 🩶:



O que achou da apresentação?

- Por favor, responda este formulário, pois me ajuda **muito** a melhorar os materiais e as apresentações 🩶.

Obrigada!

R-Ladies São Paulo por este material.

Slides criados com o pacote **xaringan**.

Tema criado com o pacote **xaringanthemer** e funções extras com **xaringanExtra** e **countdown**.

Várias ilustrações usadas na apresentação foram feitas por Allison Horst. Clique aqui para ver várias outras artes feitas por ela!

O maravilhoso logo da R-Ladies usado nesta apresentação é uma obra de Bea @Chucheria! Obrigada!