

# Introdução ao tidyverse

## 1 Revisão rápida de R e RStudio

xaringan [presentation ninja]

---

Maurício Vancine

25/04/2019



# 1 Revisão rápida de R e RStudio

## Conteúdo

1.1 Linguagem R

1.2 RStudio

1.3 Aplicações da linguagem R

1.4 Editor/Roteiro (*code/script*)

1.5 Comentários (#)

1.6 Atribuição (<-)

1.7 Objetos

1.8 Operadores

1.9 Funções

1.10 Pacotes

1.11 Ajuda (*help*)

1.12 Ambiente (*environment/workspace*)

1.13 Citações

1.14 Principais erros



# 1.1 Linguagem R

E por que usar o R?

1. É **grátis!**
2. Implementação de **rotinas** (repetir várias operações)!
3. Faz **gráficos** de forma eficiente
4. Atualmente é uma das **principais linguagens** utilizadas para análises de dados

E de onde surgiu o R?

# 1.1 Linguagem R

## Histórico - Linguagem S

John M. Chambers (Universidade de Stanford, CA, EUA)

- Old S (1976-1987)
  - New S (1988-1997)
  - S4 (1998)
- 
- Interface: S-PLUS (1988-2008)



# 1.1 Linguagem R

## Histórico - Linguagem R

Robert Gentleman e Ross Ihaka (Universidade de Auckland, NZ)

### Versões

- Desenvolvimento (1997-2000)
- Versão 1 (2000-2004)
- Versão 2 (2004-2013)
- Versão 3 (2013-2020)
- Versão 4 (2020-????)

### API

- Interface: RStudio (2011-atual)
- Atualmente: **R Core Team**

[\*] <http://vita.had.co.nz/papers/r-s.pdf>



# 1.1 Base R

## Base R

### Base R Cheat Sheet

#### Getting Help

Accessing the help files

`?mean`  
Get help of a particular function.  
`help.search("weighted mean")`  
Search the help files for a word or phrase.  
`help(package = "dplyr")`  
Find help for a package.

More about an object

`str(iris)`  
Get a summary of an object's structure.  
`class(iris)`  
Find the class an object belongs to.

#### Using Libraries

`install.packages('dplyr')`  
Download and install a package from CRAN.

`library(dplyr)`  
Load the package into the session, making all its functions available to use.

`dplyr::select`  
Use a particular function from a package.

`data(iris)`  
Load a build-in dataset into the environment.

#### Working Directory

`getwd()`  
Find the current working directory (where inputs are found and outputs are sent).

`setwd("C://file/path")`  
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

### Vectors

#### Creating Vectors

<code>c(2, 4, 6)</code>	<code>2 4 6</code>	Join elements into a vector
<code>2:6</code>	<code>2 3 4 5 6</code>	An integer sequence
<code>seq(2, 3, by=0.5)</code>	<code>2.0 2.5 3.0</code>	A complex sequence
<code>rep(1:2, times=3)</code>	<code>1 1 1 2 1 2</code>	Repeat a vector
<code>rep(1:2, each=3)</code>	<code>1 1 1 2 2 2</code>	Repeat elements of a vector

#### Vector Functions

<code>sort(x)</code>	<code>rev(x)</code>
Return x sorted.	Return x reversed.
<code>table(x)</code>	<code>unique(x)</code>
See counts of values.	See unique values.

#### Selecting Vector Elements

By Position

<code>x[4]</code>	The fourth element.
<code>x[-4]</code>	All but the fourth.
<code>x[2:4]</code>	Elements two to four.
<code>x[-(2:4)]</code>	All elements except two to four.
<code>x[c(1, 5)]</code>	Elements one and five.

By Value

<code>x[x == 10]</code>	Elements which are equal to 10.
<code>x[x &lt; 0]</code>	All elements less than zero.
<code>x[x %in% c(1, 2, 5)]</code>	Elements in the set 1, 2, 5.

Named Vectors

<code>x['apple']</code>	Element with name 'apple'.
-------------------------	----------------------------

### Programming

#### For Loop

<code>for (variable in sequence){   Do something }</code>	Example
<code>for (i in 1:4){   j &lt;- i + 10   print() }</code>	

#### While Loop

<code>while (condition){   Do something }</code>	Example
<code>while (i &lt; 5){   print()   i &lt;- i + 1 }</code>	

#### If Statements

<code>if (condition){   Do something } else {   Do something different }</code>	Example
<code>if (i &lt; 3){   print('Yes') } else {   print('No') }</code>	

#### Functions

<code>function_name &lt;- function(var){   Do something   return(new_variable) }</code>	Example
<code>square &lt;- function(x){   squared &lt;- x*x   return(squared) }</code>	

### Reading and Writing Data

Function	Output	Description
<code>read.table(file.txt)</code>	<code>write.table(df, "file.txt")</code>	Read and write a delimited text file.
<code>df &lt;- read.csv("file.csv")</code>	<code>write.csv(df, "file.csv")</code>	Read and write a comma separated value file. This is a special case of <code>read.table("file.csv")</code> .
<code>load("file.RData")</code>	<code>save(df, file = "file.RData")</code>	Read and write an R data file. A file type special for R.

#### Conditions

<code>a == b</code>	Are equal	<code>a &gt; b</code>	Greater than	<code>a &gt;= b</code>	Greater than or equal to	<code>a &lt; b</code>	Less than	<code>a &lt;= b</code>	Less than or equal to	<code>is.na(a)</code>	is null
<code>a != b</code>	Not equal	<code>a &lt; b</code>	Less than	<code>a &lt;= b</code>	Less than or equal to	<code>is.null(a)</code>	is null				

[\*] <http://github.com/rstudio/cheatsheets/raw/master/base-r.pdf>

# 1.2 RStudio

Todos conhecem o R e o RStudio?



# Não confundir a **Linguagem R** com o API **RStudio**

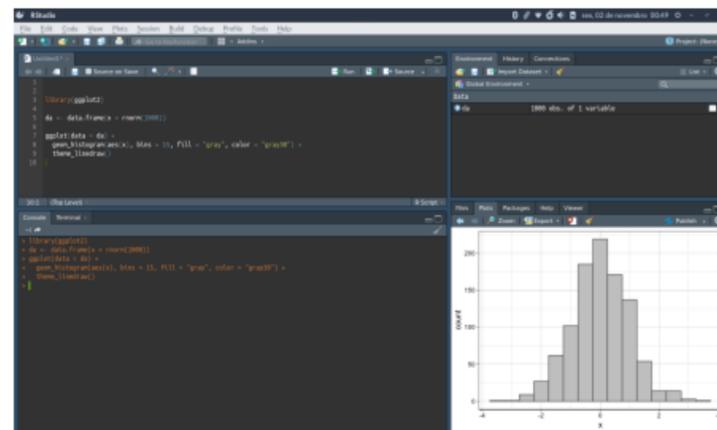
# 1.2 RStudio

**Linguagem de  
programação**



Robert Gentleman e Ross Ihaka

**Software  
RStudio**

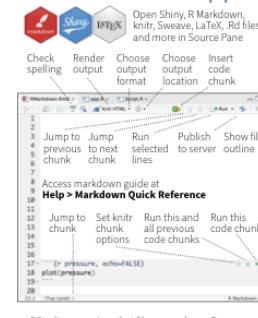


# 1.2 RStudio

## RStudio IDE Cheatsheet

### RStudio IDE :: CHEAT SHEET

#### Documents and Apps



Access markdown guide at [Help > Markdown Quick Reference](#)

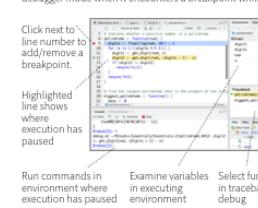


RStudio recognizes that files named `app.R`, `server.R`, `ui.R`, and `global.R` belong to a shiny app

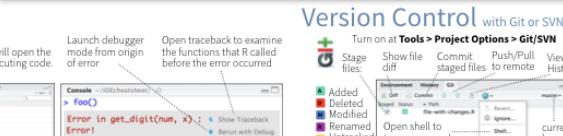
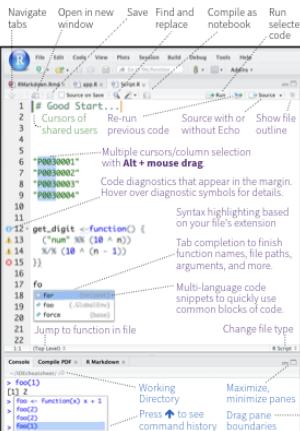


#### Debug Mode

Open with `debug(l, browser())`, or a breakpoint. RStudio will open the debugger mode when it encounters a breakpoint while executing code.



#### Write Code



#### Version Control



RStudio is a trademark of RStudio, Inc. • CC BY SA. RStudio - info@rstudio.com • 844-448-1212 • [rstudio.com](#) • Learn more at [www.rstudio.com](#) • RStudio IDE 0.99.832 • Updated: 2016-01

[\*] <https://github.com/rstudio/cheatsheets/raw/master/rstudio-ide.pdf>

# E o que o R pode fazer?

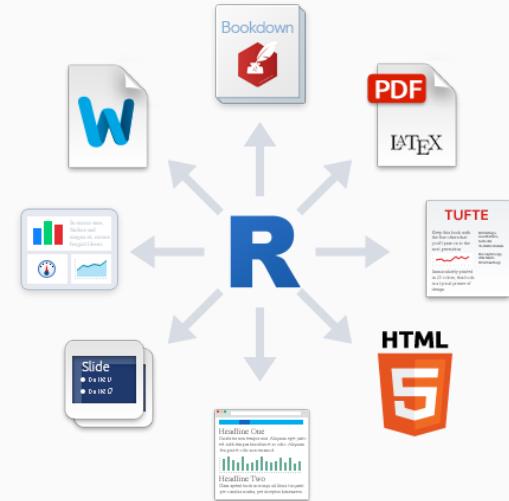
# 1.3 Aplicações da linguagem R

## Análises e Visualização

- Estatísticas univariadas e multivariadas
- Análises de dados ecológicos (população, comunidades e ecossistemas)
- Análise de dados espaciais, temporais e sonoros
- Análise de dados funcionais, genéticos e filogenéticos
- Análise de dados geoespaciais e sensoriamento remoto
- Visualização de todos os dados anteriores
- "Data Science"

## R Markdown

- Textos em HTML, PDF, Word, ODT, Markdown
- Apresentação de slides
- Websites e Blogs
- Livros
- Artigos para publicação
- Shiny



Há uns 10 anos, um nome tem se destacado no avanço da linguagem R, na parte de *manejo, visualização e análises de dados* (tidyverse)

# Hadley Wickham

Cientista Chefe no RStudio e Professor Adjunto de Estatística na Universidade de Auckland, Stanford e Rice



[\*] <http://hadley.nz/>

Há uns 5 anos, outro nome tem se destacado no avanço da linguagem R, na parte de *textos, sites e apresentações (R Markdown)*

# Yihui Xie

## Engenheiro de software no RStudio



*Principal material para estudo*

# Recomendação

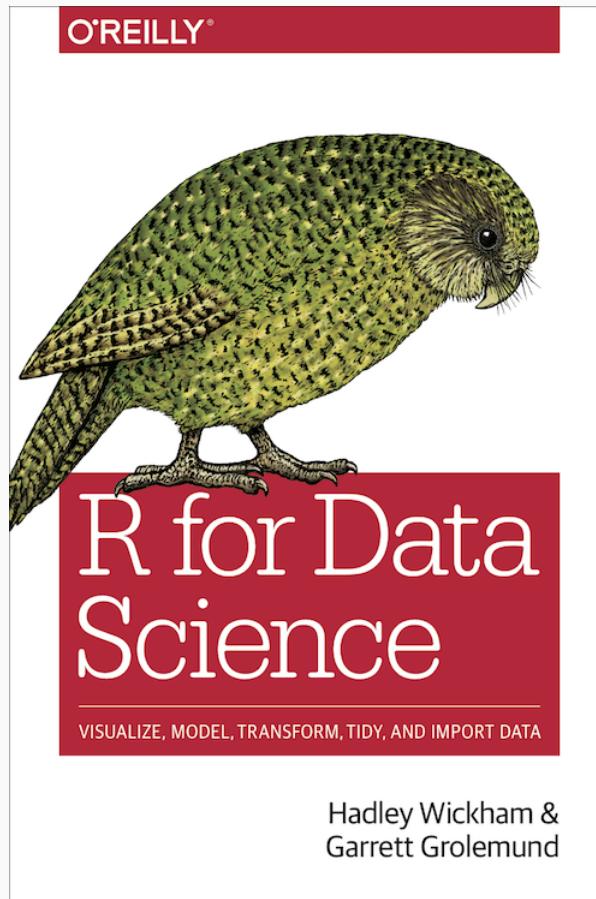
Ciência de Dados com R: introdução (2018)



[\*] <https://cdr.ibpad.com.br/index.html>

# Recomendação

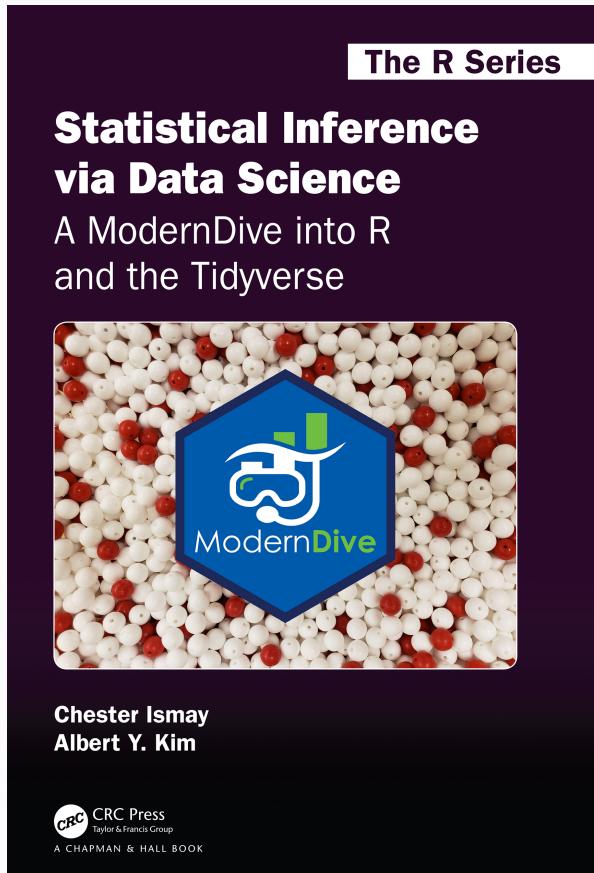
## R for Data Science (2017)



[\*] <https://r4ds.had.co.nz/>

# Recomendação

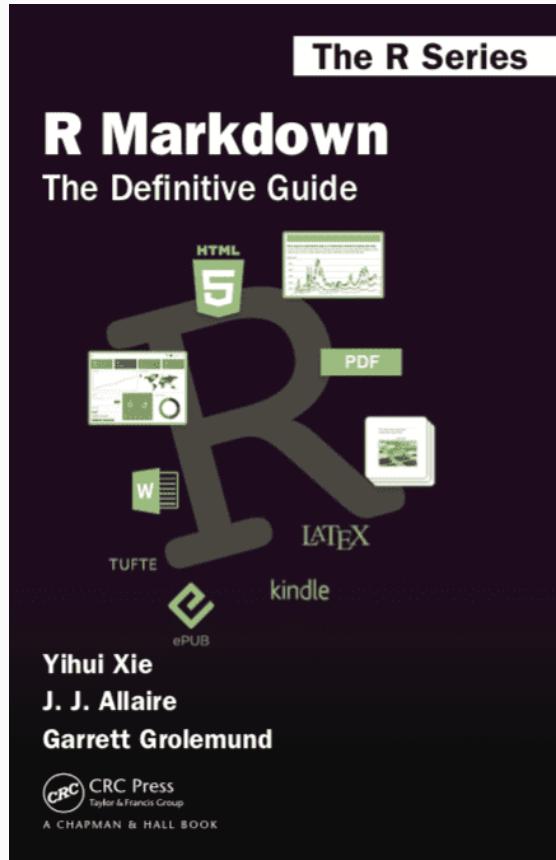
## Statistical Inference via Data Science (2019)



[\*] <https://moderndive.com>

# Recomendação

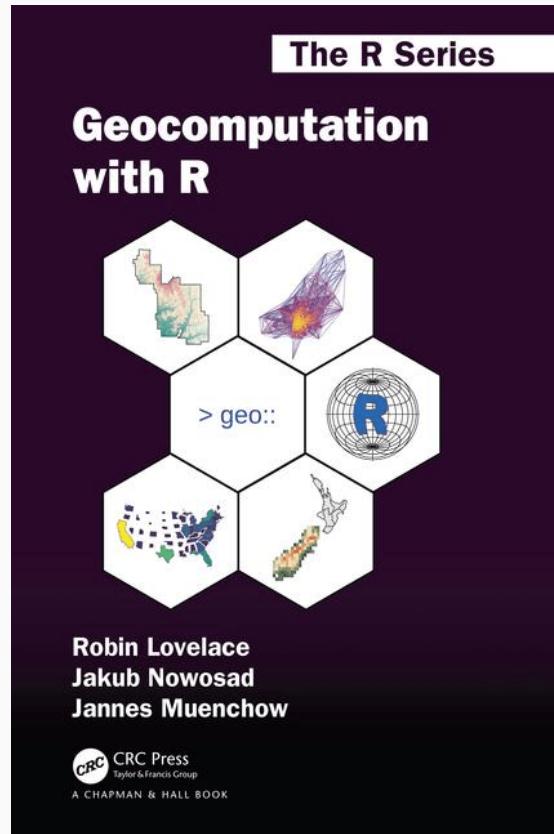
## R Markdown: The Definitive Guide (2018)



[\*] <https://bookdown.org/yihui/rmarkdown/>

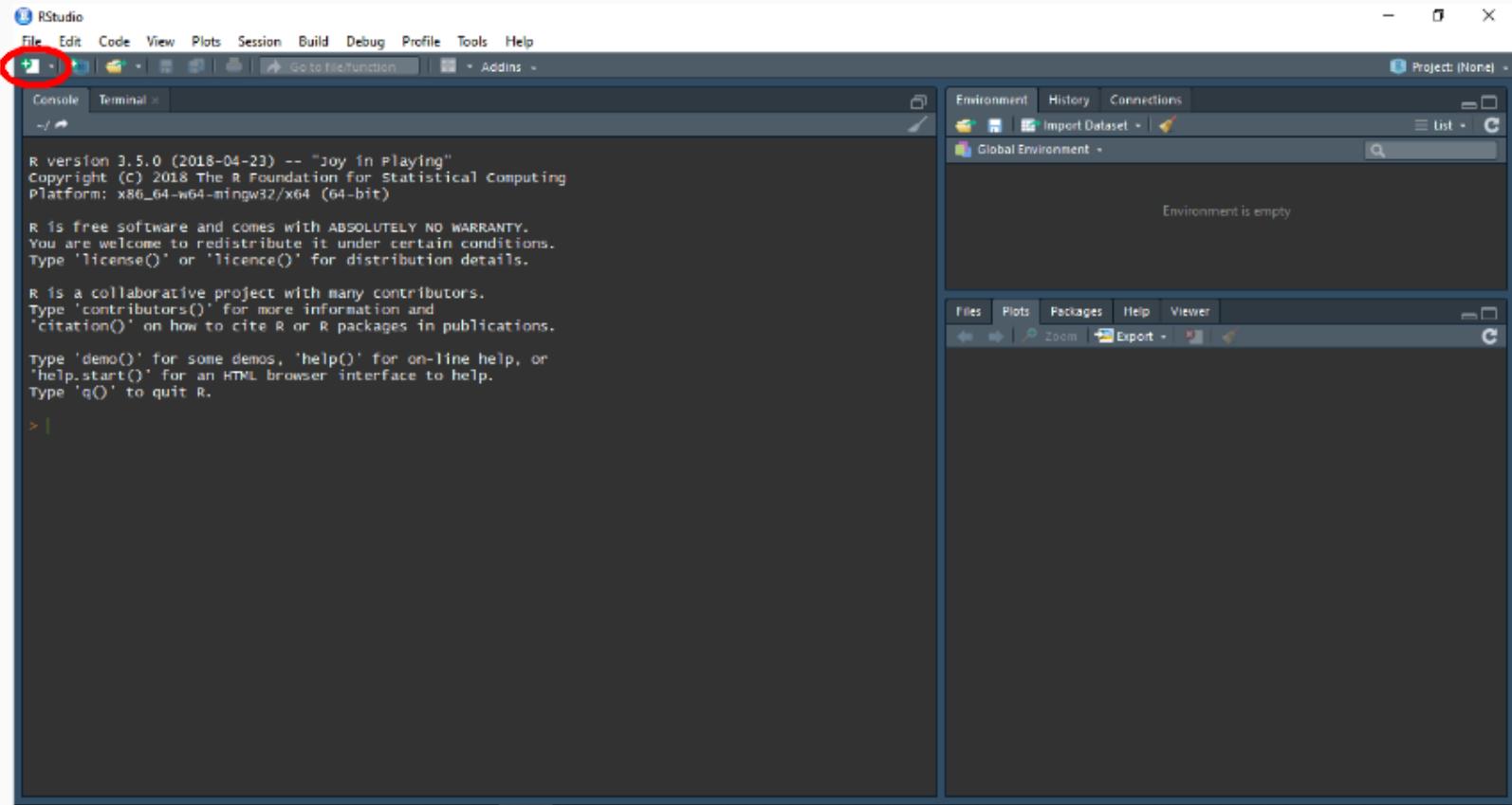
# Recomendação

## Geocomputation with R (2019)



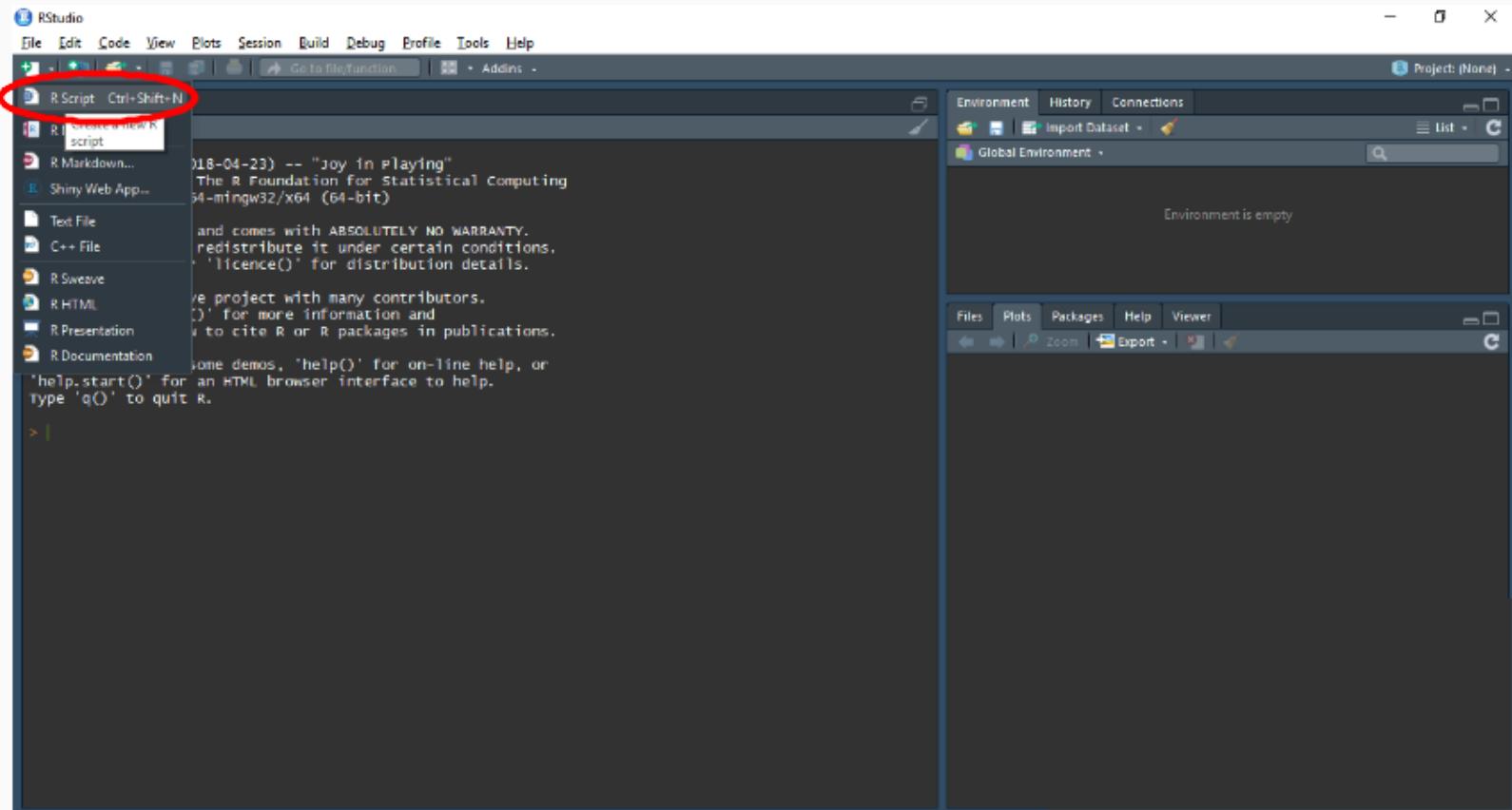
[\*] <https://geocompr.robinlovelace.net/>

# 1.4 Editor/Roteiro (*code/script*)

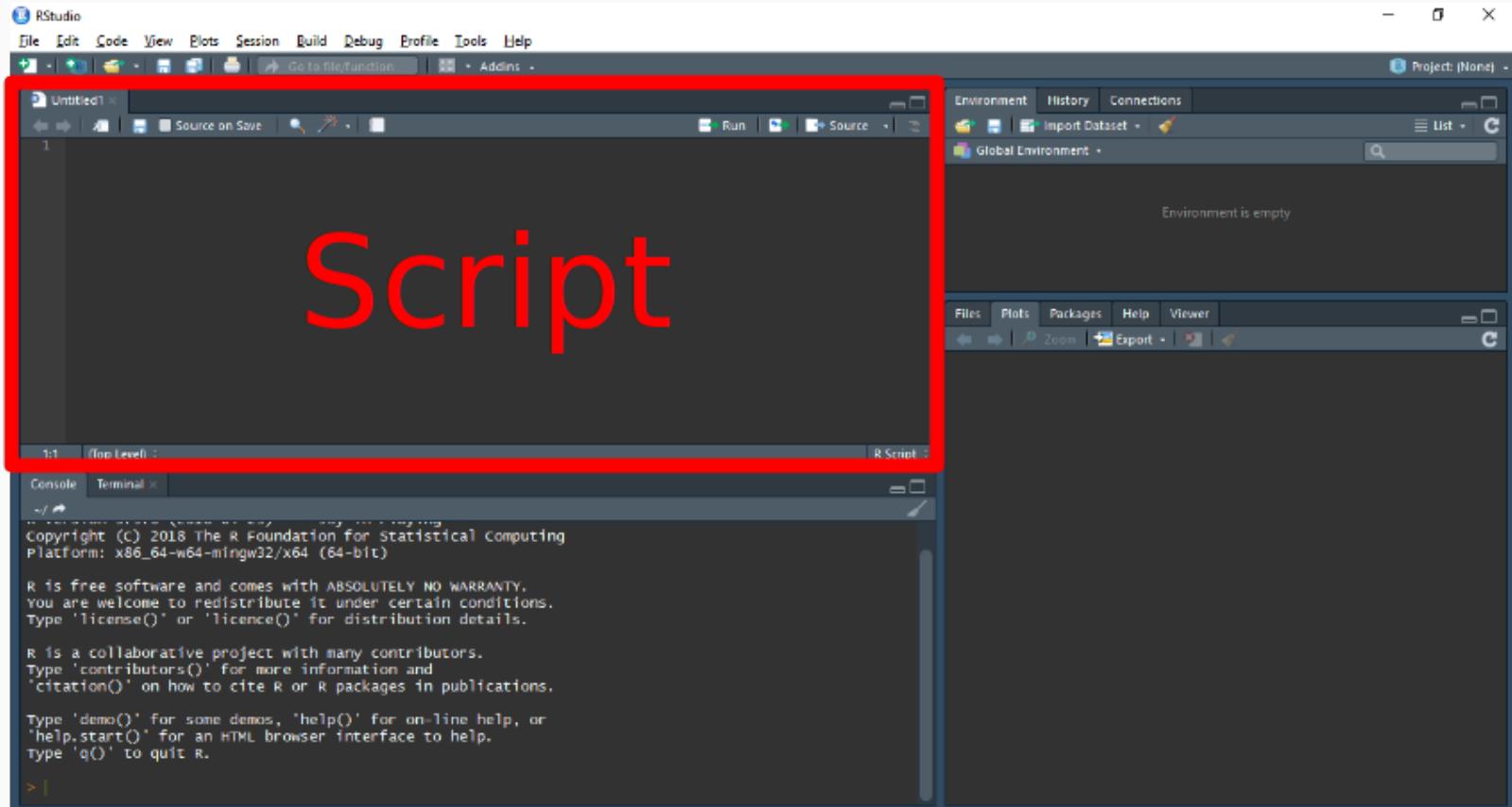


# 1.4 Editor/Roteiro (*code/script*)

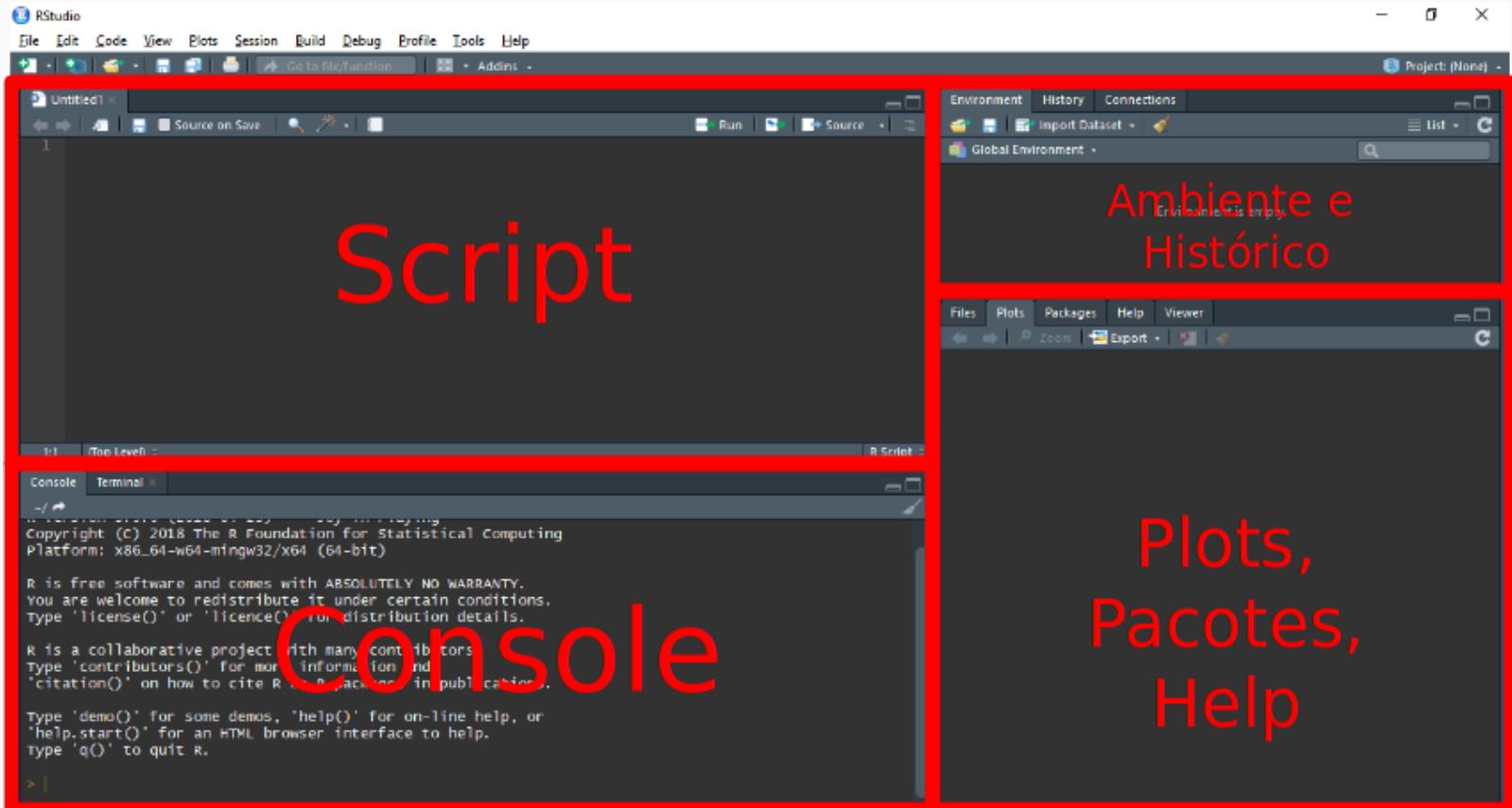
R Script (Ctrl + Shift + N)



# 1.4 Editor/Roteiro (*code/script*)



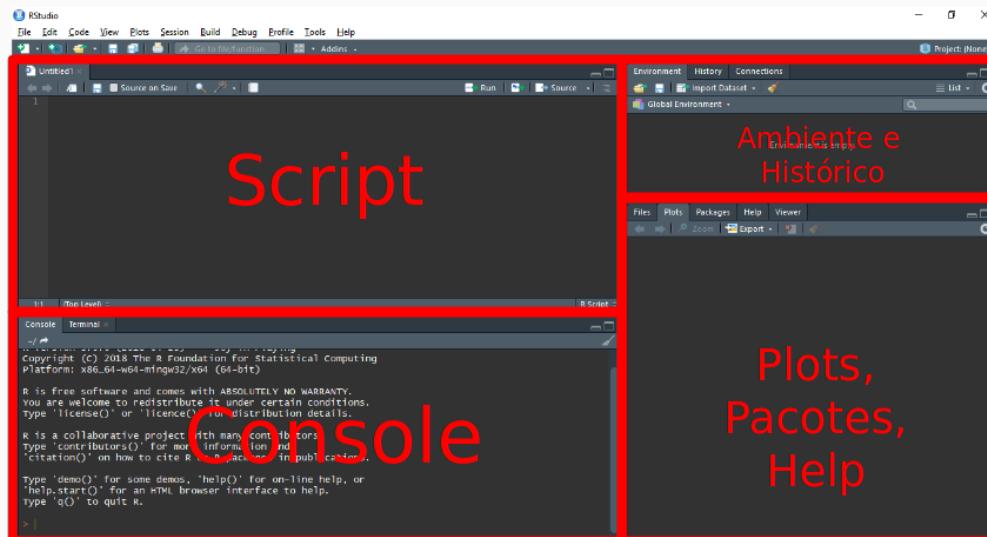
# 1.4 Editor/Roteiro (*code/script*)



# 1.4 Editor/Roteiro (*code/script*)

## Descrição das janelas

- **Editor/Script:** é onde escrevemos nossos códigos
- **Console:** é onde os códigos são rodados e vemos as saídas
- **Environment:** painel com todos os objetos criados na sessão
- **History:** painel com o histórico dos comandos rodados
- **Files:** painel que mostra os arquivos no diretório de trabalho
- **Plots:** painel onde os gráficos são apresentados
- **Packages:** painel que lista os pacotes
- **Help:** painel onde a documentação das funções é exibida



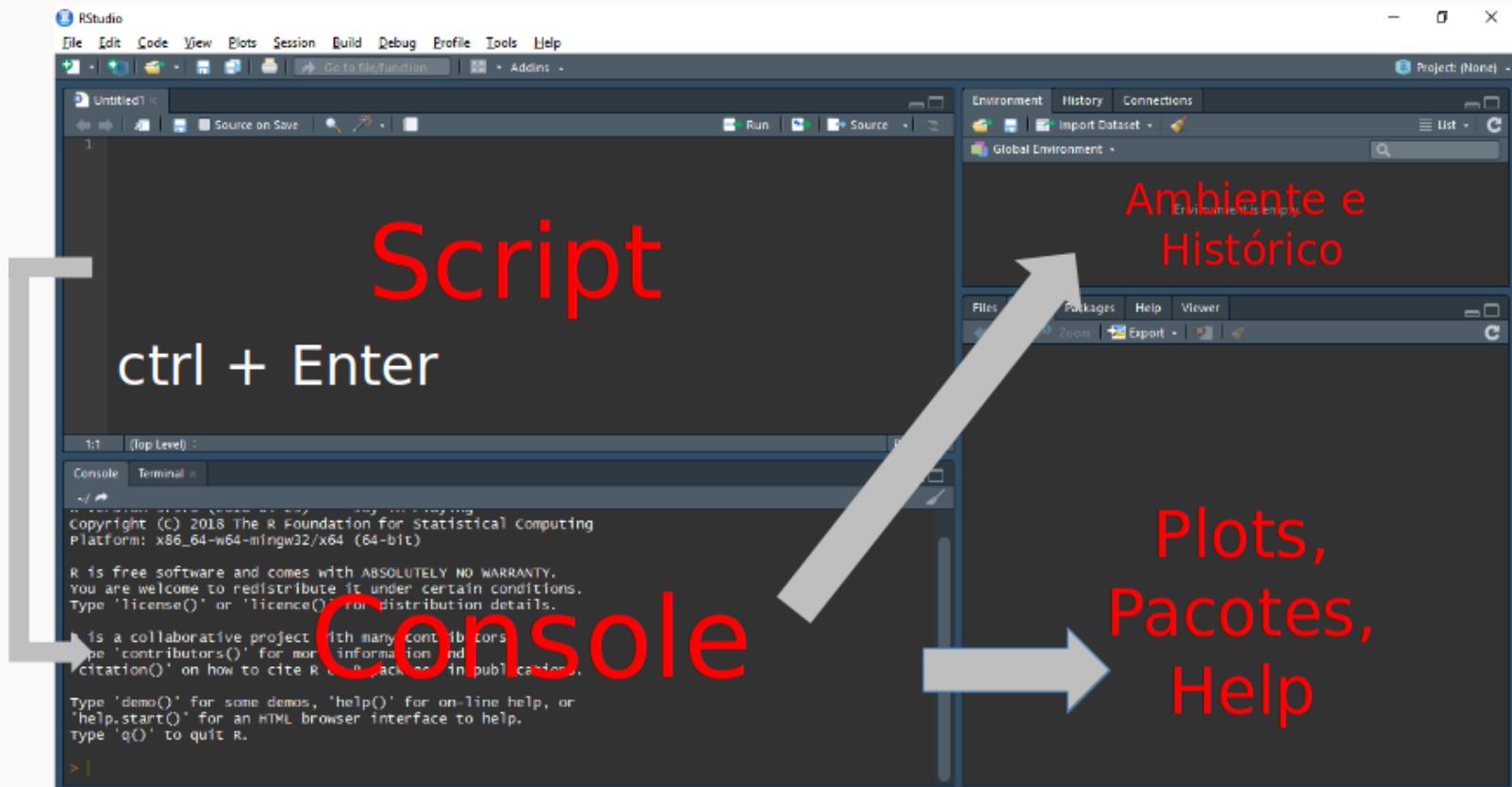
# 1.4 Editor/Roteiro (*code/script*)

## Atalhos úteis:

- **f1**: abre o painel de *Help*
- **ctrl + Enter**: roda a linha selecionada no script
- **ctrl + shift + N**: abre um novo script
- **ctrl + S**: salva um script
- **ctrl + Z**: desfaz uma operação
- **ctrl + shift + Z**: refaz uma operação
- **alt + -**: insere um sinal de atribuição (`<-`)
- **ctrl + shift + M**: insere um operador pipe (`%>%`)
- **ctrl + shift + C**: comenta uma linha no script - insere um `(#)`
- **ctrl + shift + R**: insere uma sessão `(# -----)`
- **ctrl + shift + H**: abre uma janela para selecionar o diretório de trabalho
- **ctrl + shift + f10**: reinicia o console
- **ctrl + L**: limpa os comandos do console
- **alt + shift + K**: abre uma janela com todos os atalhos disponíveis

# 1.4 Editor/Roteiro (*code/script*)

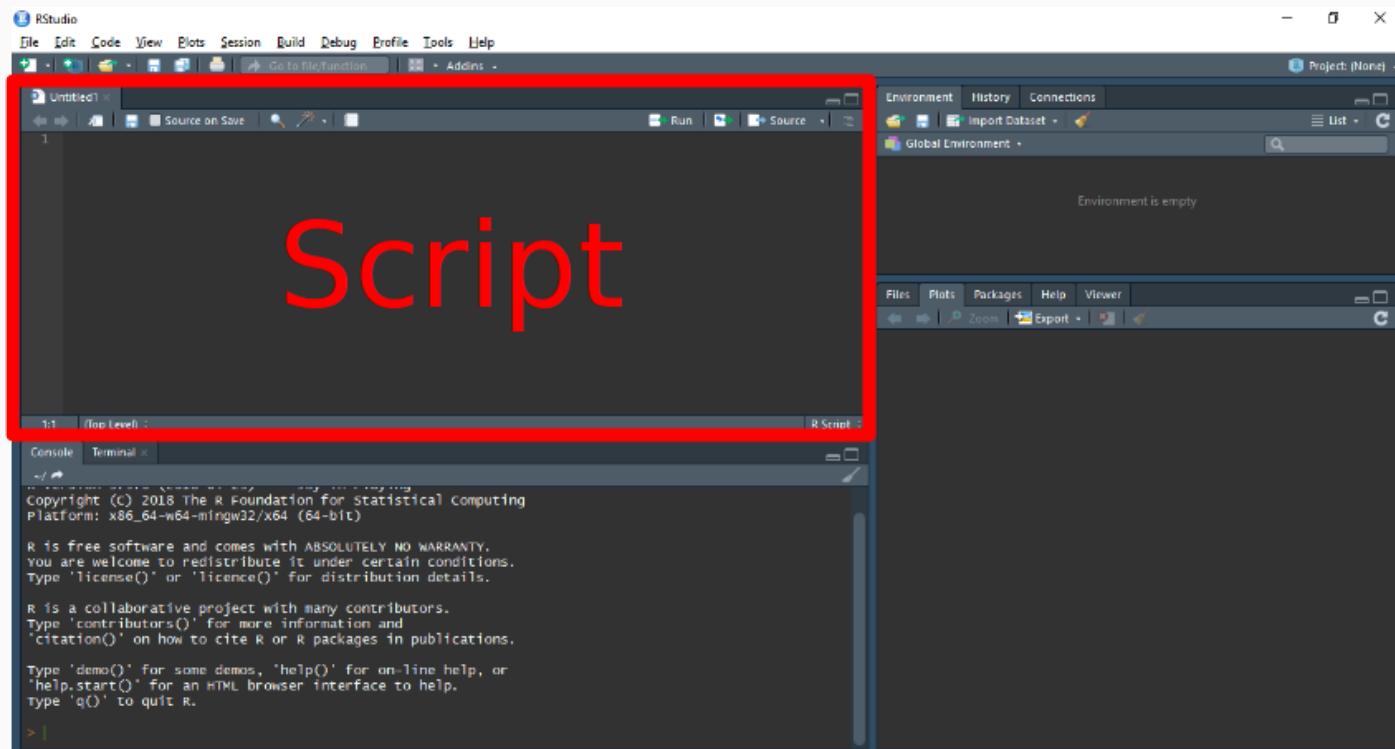
## Funcionamento



# 1.4 Editor/Roteiro (*code/script*)

## Script

- São **rascunhos** dos comandos
- Será neles que os **códigos serão escritos** e depois **enviados ao console do R**
- São **arquivos de texto simples**, que serão salvos no formato .R



# 1.4 Editor/Roteiro (*code/script*)

## Esclarecimentos

Isso é texto, não digite no R!

## Digitar no script

```
print("Isso é o resultado que deve aparecer no console")
```

## Resultado no console

```
## [1] "Isso é o resultado que deve aparecer no console"
```

# 1.4 Editor/Roteiro (*code/script*)

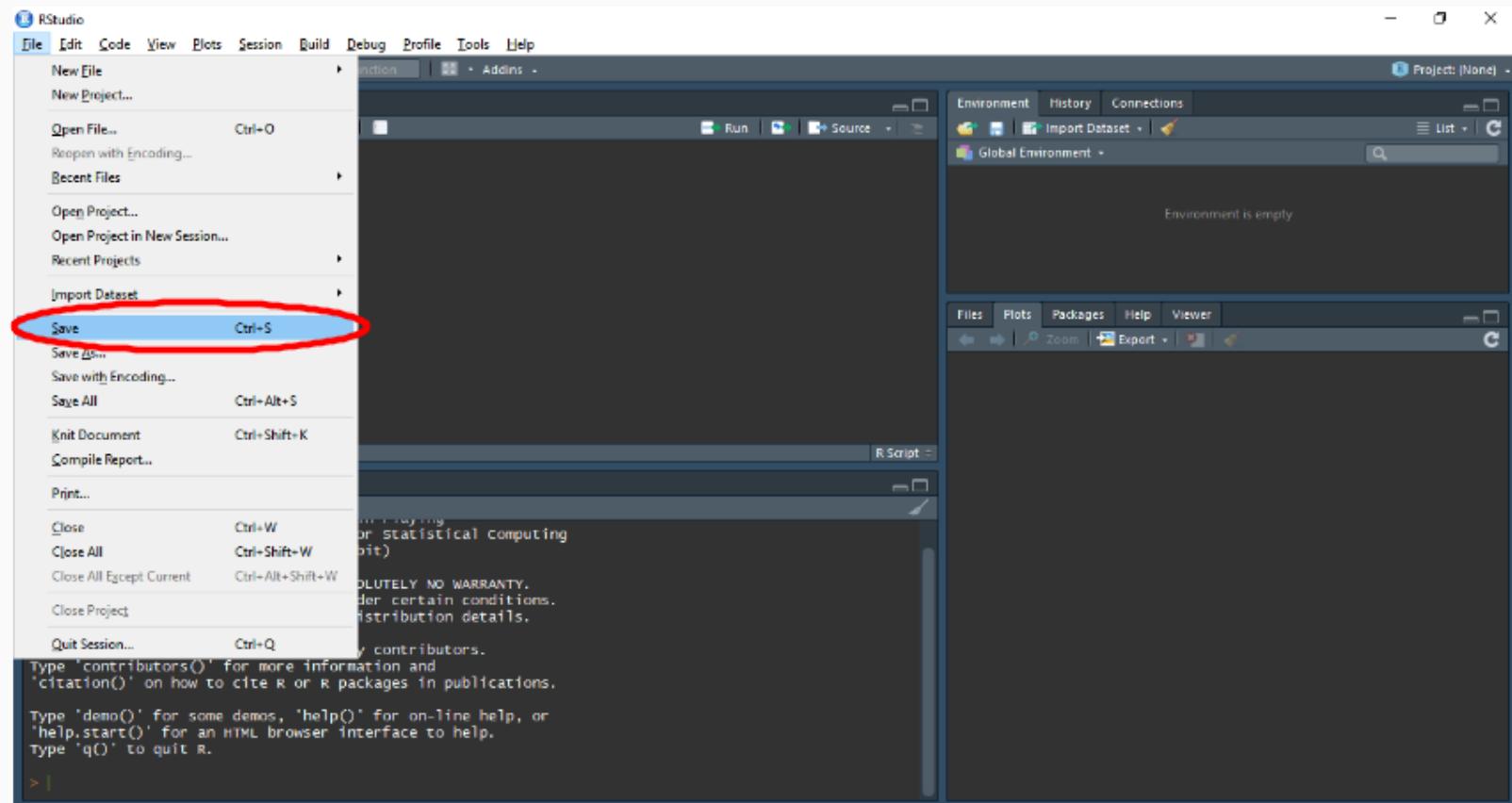
## Escolher UTF-8

Tools -> Global Options -> Code -> Saving -> Default text encoding  
(UTF-8)



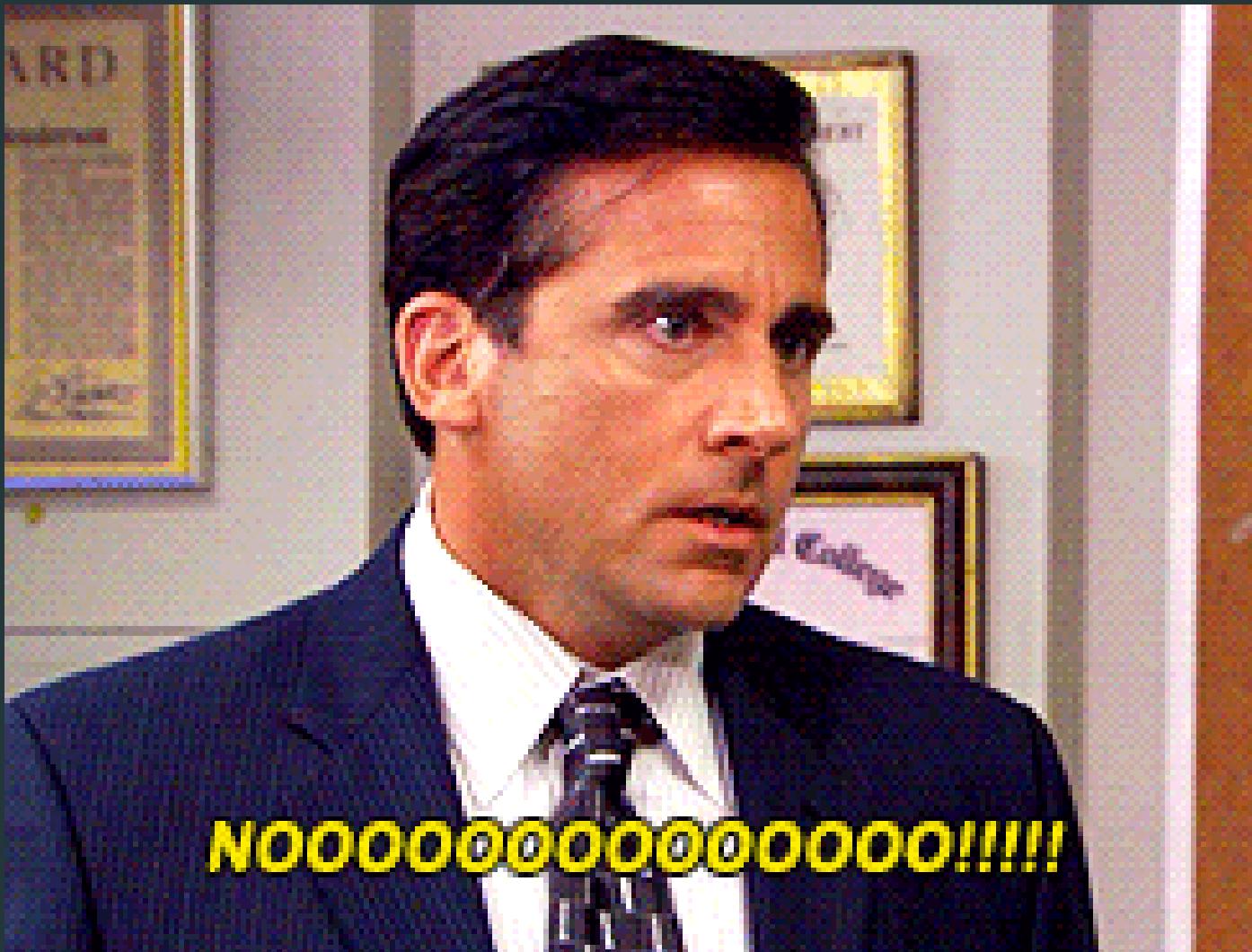
# 1.4 Editor/Roteiro (*code/script*)

ctrl + S



Calma!!!

Onde vocês iam salvar?!



# Vamos padronizar:

Pasta do diretório `/minicurso-tidyverse/`:

`00_plano_ensino`

`01_aulas`

**`02_scripts`**

`03_dados`

Calma!!!

Que nome vocês iam salvar?!



# Vamos padronizar

```
script_aula_01.R
```

# Primeiros comandos

Todos os **comandos** serão digitados no **script**!

Deixem o **cursor** em **qualquer local da linha** e executem essa linha utilizando essa **combinação**:

ctrl + Enter

Vamos testar:

```
1
```

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

```
1 + 2
```

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

É isso que faremos pelo resto de nossas  
vidas...

Muito bem, apaguem essas linhas do script

# 1.5 Comentários (#)

Comentários **não são lidos** pelo R e **descrevem informações**

## Cabeçalho

```
#' ---
#' title: aula 01 - revisao r
#' author: mauricio vancine
#' date: 2020-04-25
#' ---
```

## Informações sobre os comandos

```
## comentarios
# o r nao le o codigo depois do # (hash)

42 # essas palavras nao sao executadas, apenas o 42
```

```
## [1] 42
```

# Organização dos scripts

## Organização

Organização em um script é fundamental!

Separe as linhas das análises e comente cada comando

Há livros apenas sobre assunto: "[Clean Code](#)"

## Estilos de códigos

[Google's R Style Guide](#)

[Style guide](#) - Hadley Wickham

[The tidyverse style guide](#) - Hadley Wickham

# Calculadora

## Operações aritméticas

```
## operacoes aritmeticas (+, -, *, /, ^)  
10 + 2 # adicao
```

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

```
10 - 2 # subtracao
```

```
## [1] 8
```

```
10 * 2 # multiplicacao
```

```
## [1] 20
```

```
10 / 2 # divisao
```

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

# Calculadora

## Ordem das operações aritméticas

`^ >> * ou / >> + ou -`

```
# sem especificar - segue a ordem  
1 * 2 + 2 / 2 ^ 2
```

```
## [1] 2.5
```

```
# especificando - segue os parênteses  
( (1 * 2) + (2 / 2) ) ^ 2
```

```
## [1] 9
```

# Exercícios

# Exercício 01

Resolva essa equação...

Escolha a alternativa correta:

$$7 + 7 \div 7 + 7 \times 7 - 7 = ?$$

- a) 00      b) 08
- c) 50      d) 56

02 : 00

# Exercício 01

## Resposta

```
# exercício 01  
7 + 7 / 7 + 7 * 7 - 7
```

```
## [1] 50
```

Alguém notou alguns colchetes a mais?

# Colchetes

Famigerados colchetes na resposta do console....

```
## famigerados colchetes [] na resposta  
10 + 2 # adicao
```

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

```
# indicam a posicao do numero em uma sequencia  
10:60 # sequencia unitaria de 10 a 60
```

```
## [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33  
## [41] 50 51 52 53 54 55 56 57 58 59 60
```

# 1.6 Atribuição (<-)

Atribuição possibilita a **manipulação de dados**

Dados são "atribuídos" a **objetos**, que são **palavras** que "guardam" esses dados

Iremos utilizar os símbolos "<" (**menor**), seguido de "-" (**menos**), **sem espaço!!!**

**palavra <- dados**

Atalho: alt + -

# 1.6 Atribuição (<-)

Vamos atribuir o **valor 10** à palavra **obj\_10**

```
## atribuicao - simbolo (<-)  
obj_10 <- 10
```

Agora a palavra **obj\_10** vale **10**

Mas não aconteceu nada....



# 1.6 Atribuição (<-)

Sempre **confira** a atribuição!!!

Chame o objeto **novamente**!!!

```
## atribuicao - simbolo (<-)
obj_10 <- 10
obj_10
```

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

Outro exemplo:

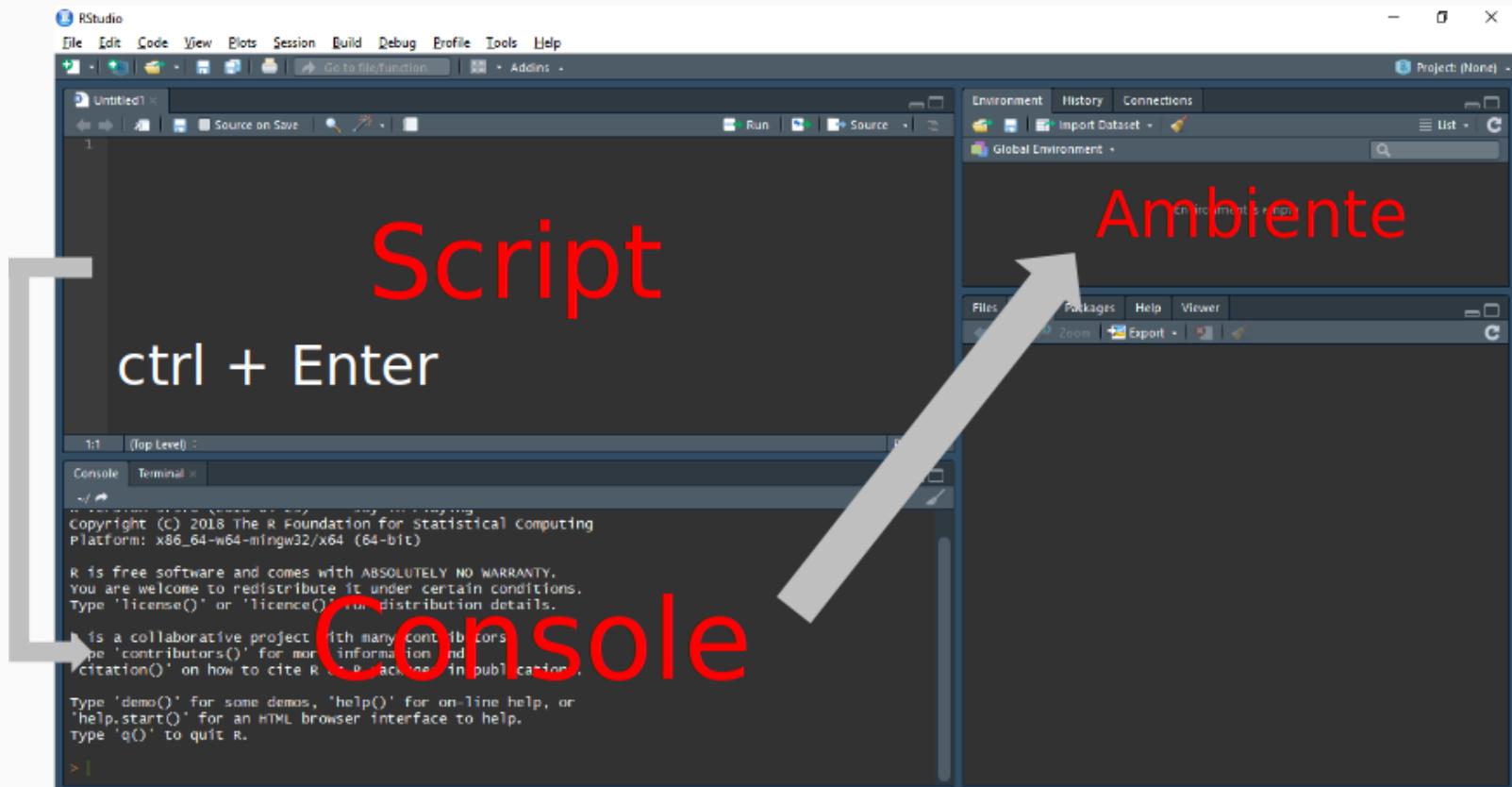
```
## atribuicao - simbolo (<-)
obj_2 <- 2
obj_2
```

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

Os objetos podem ser visualizados no painel  
*Environment*

# 1.6 Atribuição (<-)

## Ambiente (*Environment*)



# 1.6 Atribuição (<-)

CUIDADO!

O R **sobrescreve** os valores dos objetos com o **mesmo nome**!

```
# sobrescreve o valor dos objetos
obj <- 100
obj
```

```
## [1] 100
```

```
# obj agora vale 2
obj <- 2
obj
```

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

# 1.6 Atribuição (<-)

## CUIDADO!

O R tem **limitações** ao nomear objetos!

1. Nome de objetos só podem **começar por letras (a-z ou A-Z) ou pontos (.)**
2. Nome de objetos só podem **conter letras (a-z ou A-Z), números (0-9), underscores (\_) ou pontos (.)**
3. R é *case-sensitive*, i.e., ele difere **letras maiúsculas de minúsculas**. Assim, um objeto chamado "*resposta*" é diferente do objeto "*RESPOSTA*"
4. Evitem utilizar **letras maiúsculas, acentos ou cedilha (ç)**
5. Nome de objetos não podem ser iguais a **nomes especiais**:

```
break, else, FALSE, for, function, if, Inf, NA, NaN, next, repeat,  
return, TRUE, while
```

# 1.7 Objetos

Podemos utilizar **objetos** para fazer operações

```
# definir dois objetos
val1 <- 10
val1
```

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

```
va2 <- 2
va2
```

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

# 1.7 Objetos

Podemos utilizar **objetos** para fazer operações

```
# operacoes com objetos  
val1 + val2 # adicao
```

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

```
val1 - val2 # subtracao
```

```
## [1] 8
```

# 1.7 Objetos

Podemos ainda **atribuir os resultados** das operações a **objetos**

```
# operacoes com objetos e atribuicao  
adi <- val + va2 # adicao  
adi
```

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

```
sub <- val - va2 # subtracao  
sub
```

```
## [1] 8
```

# 1.8 Operadores

## Operadores aritméticos

### Resultados numéricos

Operador	Descrição	Uso
+	Adição	$a + b$
-	Subtração	$a - b$
*	Multiplicação	$a * b$
/	Divisão	$a / b$
%%	Resto da divisão	$a \% \% b$
%/%	Quociente da divisão	$a \% \% b$
^	Potenciação	$a^b$

# 1.8 Operadores

## Operadores relacionais

Resultados Booleanos (TRUE ou FALSE)

<b>Operador</b>	<b>Descrição</b>	<b>Uso</b>
<	Menor	$a < b$
>	Maior	$a > b$
==	Igual	$a == b$
<=	Menor ou igual	$a <= b$
>=	Maior ou igual	$a >= b$
!=	Não igual (diferente)	$a != b$

# Exercícios

## Exercício 02

Verifique se  $3 \times 2^3$  é maior que  $2 \times 3^2$

02 : 00

# Exercício 02

## Resposta

```
# exercicio 02  
3 * 2 ^ 3 >= 2 * 3 ^ 2
```

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

# 1.9 Funções

## Funções

**Comandos que realizam operações em argumentos**

Estrutura de uma função:

**nome\_da\_funcao(argumento1, argumento2)**

```
## funcoes
# comandos que realizam operacoes em argumentos
# estrutura de uma funcao
# 1. nome da funcao - remete ao que ela faz
# 2. parenteses - limitam a funcao
# 3. argumentos - onde a funcao ira atuar
# 4. virgulas - separam os argumentos
```

# 1.9 Funções

Os **argumentos** de uma função podem ser de **dois tipos**:

1. **Valores ou Objetos**: a função irá **alterar os valores** em si ou os valores **atribuídos** aos objetos
2. **Parâmetros**: valores fixos que informam um **método** ou a realização de uma **operação**. Informa-se o **nome desse argumento**, seguido de "**=**" e um *número, texto* ou *TRUE* ou *FALSE*

Exemplo:

```
sum(1, NA)
```

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

```
sum(1, NA, na.rm = TRUE)
```

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

# 1.9 Funções

## Argumentos como **valores**

```
# funcoes - argumentos como valores
# soma
sum(10, 2)
```

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

```
# produto
prod(10, 2)
```

```
## [1] 20
```

# 1.9 Funções

## Argumentos como objetos

```
# funcoes - argumentos como objetos  
# soma  
sum(val, va2)
```

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

```
# produto  
prod(val, va2)
```

```
## [1] 20
```

## 1.9 Funções

# Argumentos como parâmetros

```
## [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2
```

```
# repeticao - cada  
rep(x = 1:5, each = 10)
```

```
## [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4
```

## 1.9 Funções

# Atribuir resultados das funções à objetos

```
## [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2
```

```
## [1] 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4
```

# Atribuição, função e linha temporal

**Criar** dois objetos

```
# criar dois objetos
foo <- 2
bar <- 3
```

**Somar** esses objetos e **atribuição** ao objeto *su*

```
# somar e atribuir
su <- sum(foo, bar)
su
```

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

**Raiz quadrada** do *su* e **atribuição** ao *sq*

```
# raiz e atribuir
sq <- sqrt(su)
sq
```

# Atribuição, função e linha temporal

Esse é o processo de programação no R:

1. **Atribuição** de dados a objetos
2. **Funções que operam e mudam** esses dados
3. Nova **atribuição** desses resultados a novos objetos

# Exercícios

# Exercício 03

Criem dois objetos (qualquer nome) com os valores 100 e 300

Multipliquem esses objetos (função **prod**) e atribuam ao objeto *mult*

Façam o logaritmo natural (função **log**) do objeto *mult* e atribuam ao objeto *loge*

02 : 00

# Exercício 03

## Resposta

```
# criar dois objetos
foo <- 100
bar <- 300
```

```
# multiplicar e atribuir
mult <- prod(foo, bar)
mult
```

```
## [1] 30000
```

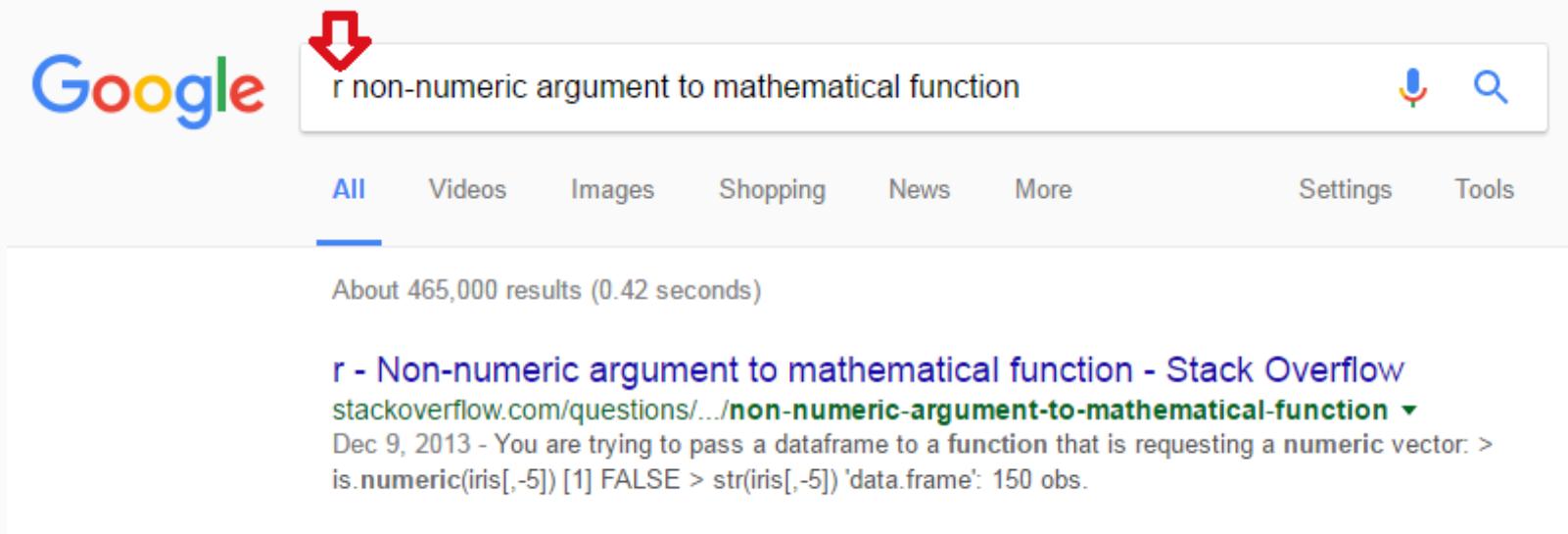
```
# raiz e atribuir
loge <- log(mult)
loge
```

```
## [1] 10.30895
```

E vocês devem estar se perguntando: e como saber o nome das funções?!



# Uma maracutaia para ajudar!



A screenshot of a Google search results page. The search query "r non-numeric argument to mathematical function" is entered into the search bar, with a red arrow pointing to the search term. Below the search bar, the "All" tab is selected, along with other categories like Videos, Images, Shopping, News, More, Settings, and Tools. The search results indicate "About 465,000 results (0.42 seconds)". The top result is a link to a Stack Overflow question titled "r - Non-numeric argument to mathematical function - Stack Overflow" from stackoverflow.com. The snippet of the answer explains that trying to pass a data frame to a function that expects a numeric vector will fail, as shown in the R code: `is.numeric(iris[, -5])` returns FALSE and `str(iris[, -5])` shows 'data.frame': 150 obs.

E de onde vêm as funções?!

# 1.9 Funções

Funções vêm de **duas fontes**:

1. Pacotes já **instalados por padrão** e que são **carregados** quando abrimos o R
2. Pacotes que **instalamos** e **carregamos** com comandos

E o que são pacotes afinal?!

# 1.10 Pacotes

**Coleção de funções para executar tarefas específicas**

Duas fontes: **CRAN** (*finalizados*) e **GitHub** (em *desenvolvimento*)

**Verificar pacotes carregados**

```
# verificar pacotes carregados  
search()
```

```
## [1] ".GlobalEnv"           "package:vegan"        "package:lattice"      "package:perm"  
## [7] "package:stringr"       "package:dplyr"         "package:purrr"        "package:read  
## [13] "package:ggplot2"        "package:tidyverse"     "package:xaringan"     "package:page  
## [19] "package:graphics"       "package:grDevices"    "package:utils"        "package:dat  
## [25] "package:base"
```

# 1.10 Pacotes

**Coleção de funções para executar tarefas específicas**

Duas fontes: **CRAN** (*finalizados*) e **GitHub** (em *desenvolvimento*)

**Verificar pacotes instalados**

```
# verificar pacotes instalados  
library()
```

# 1.10 Pacotes

Ex.: pacote `vegan`

Fontes:

Pacotes do CRAN

<https://cran.r-project.org/web/packages/vegan/index.html>

Pacotes do GitHub

<https://github.com/vegandevs/vegan>

# 1.10 Pacotes

## Instalar pacotes

1. Instala-se apenas **uma vez**
2. **Precisa** estar conectado à **internet**
3. O **nome do pacote precisa** estar entre **aspas**
4. Função (CRAN):

```
install.packages()
```

```
# instalar pacotes
install.packages("vegan")
```

# 1.10 Pacotes

## Carregar pacotes

1. Carrega-se **toda vez** que se abre **uma nova sessão do R**
2. **Não precisa** estar conectado à **internet**
3. O **nome do pacote não precisa** estar entre **aspas**
4. Funções:

`library()` ou `require()`

```
# carregar pacotes
library(vegan)
```

# 1.10 Pacotes

## Instalar pacotes do GitHub

### 1. Instalar pacote **devtools**

```
# instalar pacote devtools
install.packages("devtools")

# carregar pacote devtools
library(devtools)
```

### 2. Instalar usando a função `install_github`

Atentar para usar essa forma **usuário/repositório**

```
# instalar pacote do github
install_github("vegandevs/vegan")

# carregar pacote do github
library("vegan")
```

# 1.10 Pacotes

## Atualização de pacotes

Pacotes são **atualizados com frequência** (mensal | semestral | anual)

Pacotes **não atualizam sozinhos**

A instalação de um pacote pode depender da **versão** dos pacotes dependentes

É uma função que **demora** para rodar

```
# atualizacao dos pacotes instalados  
update.packages(ask = FALSE)
```

E onde ficam esses pacotes no meu  
notebook?

# 1.10 Pacotes

## Windows

C:/Users/**nome\_do\_computador**/Documentos/R/win-library/**numero\_da\_versao\_r**

## Unix (Linux e MacOS):

/home/**nome\_do\_computador**/R/**tipo\_do\_computador**/**numero\_da\_versao\_r**

# 1.10 Pacotes

Exemplos:

**vegan** – análises de comunidades

**raster** – manejo de rasters

**ggplot2** – gráficos

**bblme** – seleção de modelos (AIC)

**dismo** – modelos de distribuição de espécies

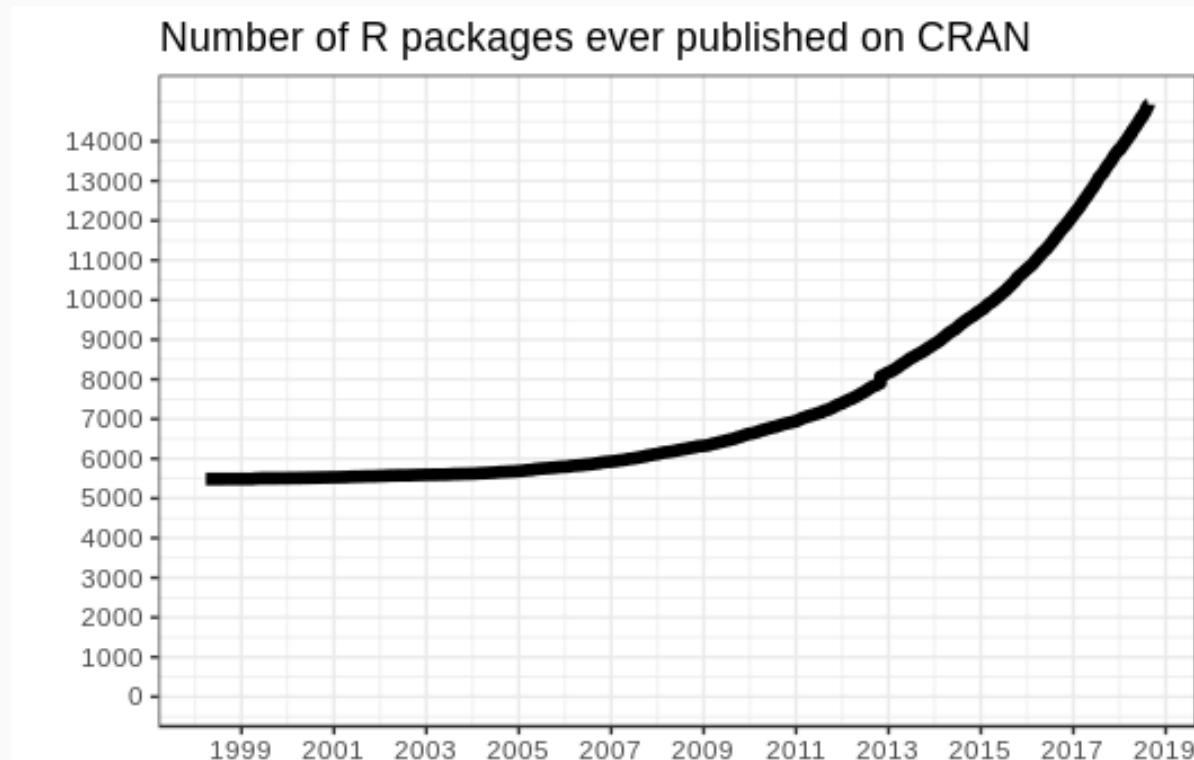
**tidyverse** – data science

E quantos pacotes existem?

# 1.10 Pacotes

```
nrow(available.packages(repos = "http://cran.r-project.org"))
```

```
## [1] 15535
```



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

# Exercícios

# Exercício 04

Instalem o pacote **tidyverse** do CRAN

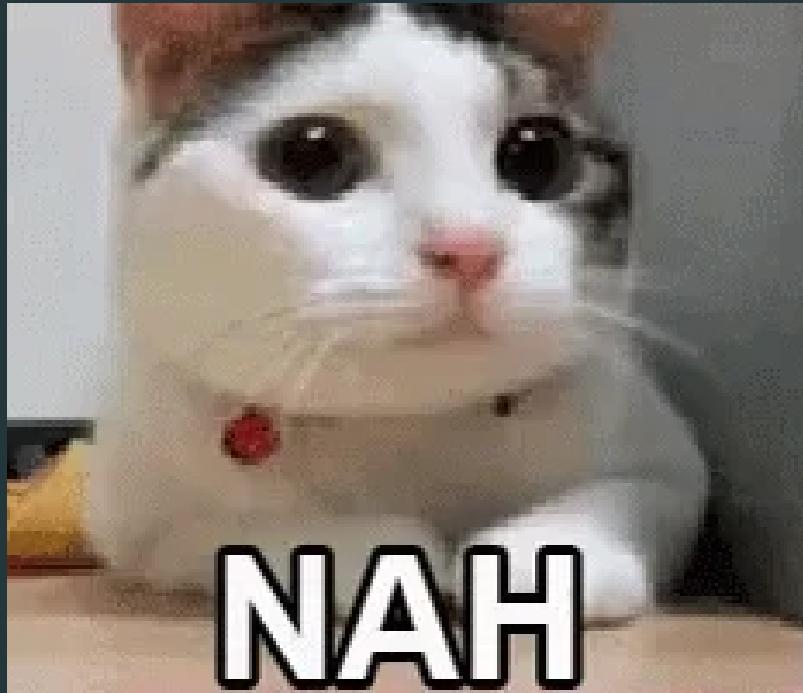
00 : 30

# Exercício 04

## Resposta

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

Alguém aqui lê o manual de alguma coisa?



# 1.11 Ajuda (*help*)

Descreve as informações de uma função

```
## ajuda  
# descreve as informacoes de uma funcao  
  
help("mean") # arquivo .html  
  
?mean
```

# 1.11 Ajuda (*help*)

mean {base}

R Documentation

## Arithmetic Mean

### Description

Generic function for the (trimmed) arithmetic mean.

### Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

### Arguments

- x An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for trim = 0, only.
- trim the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
- na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
- ... further arguments passed to or from other methods.

### Value

If trim is zero (the default), the arithmetic mean of the values in x is computed, as a numeric or complex vector of length one. If x is not logical (coerced to numeric), numeric (including integer) or complex, [NA\\_real\\_](#) is returned, with a warning.

If trim is non-zero, a symmetrically trimmed mean is computed with a fraction of trim observations deleted from each end before the mean is computed.

### References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

### See Also

[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

### Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

# 1.11 Ajuda (*help*)

## Detalhes de um pacote

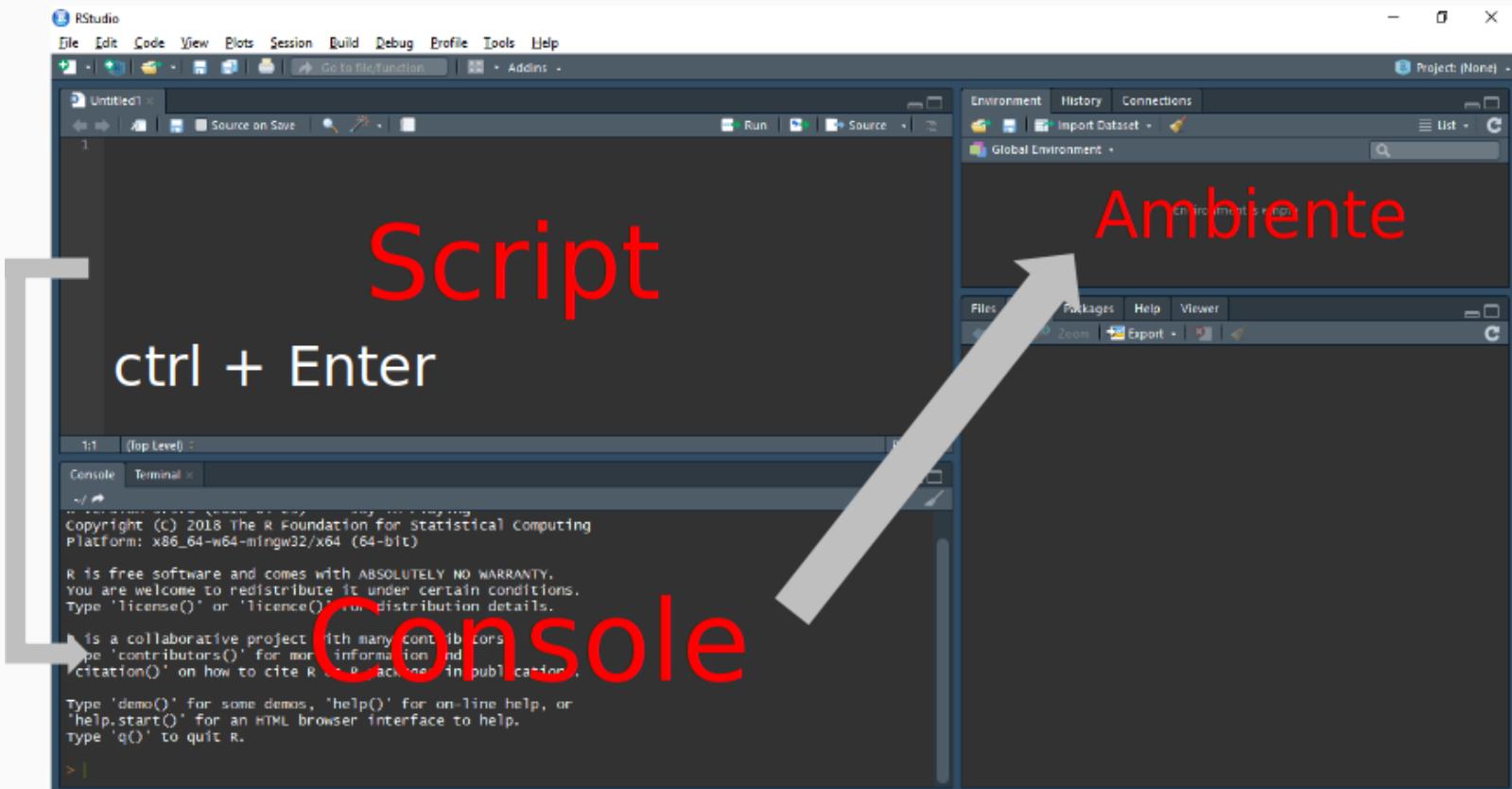
```
library(help = "vegan")
```

- Descrição
- Versão
- Autores
- Dependências
- Sites
- Repositório
- Índice de funções
- Diretório

Todos se lembram da atribuição e criação de  
objetos?

palavra <- dados

# 1.12 Ambiente (*environment*)



# 1.12 Ambiente (*environment*)

Listar todos os objetos criados

```
# listar objetos  
ls()
```

```
## [1] "adi"        "bar"        "foo"        "i"          "loge"       "mult"  
## [11] "rep_times" "sq"         "su"         "sub"        "val"        "va2"
```

```
# listar objetos  
objects()
```

```
## [1] "adi"        "bar"        "foo"        "i"          "loge"       "mult"  
## [11] "rep_times" "sq"         "su"         "sub"        "val"        "va2"
```

# 1.12 Ambiente (*environment*)

CUIDADO!

Toda a vez que **fechamos o R**, os objetos criados são **apagados**!



# 1.12 Ambiente (*environment*)

Salvar todos os objetos criados

```
Session -> Save Workspace As... -> meus_objetos.RData
```

Carregar os objetos criados e salvos

```
Session -> Load Workspace... -> meus_objetos.RData
```

# 1.12 Ambiente (*environment*)

## Remover um objeto

```
# listar objetos  
ls()  
  
## [1] "adi"        "bar"        "foo"        "i"          "loge"       "mult"  
## [11] "rep_times" "sq"         "su"         "sub"        "val"        "va2"
```

```
# remover o objeto "bar"  
rm(bar)
```

```
# listar objetos  
ls()  
  
## [1] "adi"        "foo"        "i"          "loge"       "mult"       "obj"  
## [11] "sq"         "su"         "sub"        "val"        "va2"
```

# 1.12 Ambiente (*environment*)

## Remover todos os objetos

```
# listar objetos  
ls()
```

```
## [1] "adi"          "foo"          "i"            "loge"         "mult"        "obj"  
## [11] "sq"           "su"           "sub"          "val"          "va2"
```

```
# remover todos os objetos  
rm(list = ls())
```

```
# listar objetos  
ls()
```

```
## character(0)
```

# 1.12 Ambiente (*environment*)

## Carregar os objetos criados e salvos

```
Session -> Load Workspace... -> meus_objetos.RData
```

```
# rodem para verificar  
ls()
```

```
## [1] "adi"        "bar"        "foo"        "lo"         "mu"         "obj"  
## [11] "sq"         "su"         "sub"        "val"        "va2"
```

# 1.13 Citações

## Como citar o R e os pacotes em trabalhos?

```
## citacao do r e dos pacotes  
  
# citacao do R  
citation()
```

```
##  
## To cite R in publications use:  
##  
##   R Core Team (2020). R: A language and environment for statistical computing  
##   Computing, Vienna, Austria. URL https://www.R-project.org/.  
##  
## A BibTeX entry for LaTeX users is  
##  
## @Manual{,  
##   title = {R: A Language and Environment for Statistical Computing},  
##   author = {{R Core Team}},  
##   organization = {R Foundation for Statistical Computing},  
##   address = {Vienna, Austria},  
##   year = {2020},
```

# 1.13 Citações

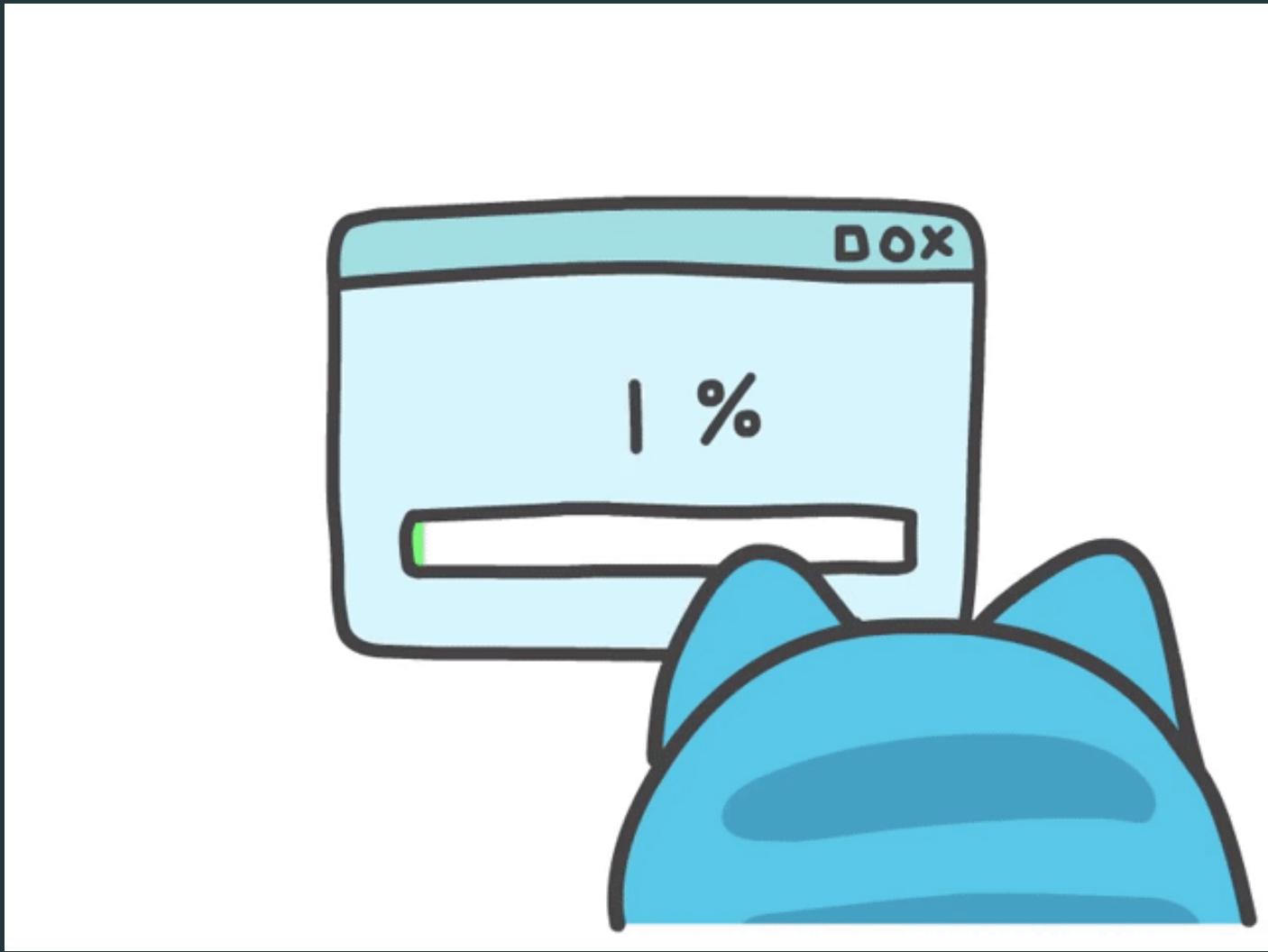
## Como citar o R e os pacotes em trabalhos?

```
# citacao dos pacotes
citation("vegan")

##
## To cite package 'vegan' in publications use:
##
##   Jari Oksanen, F. Guillaume Blanchet, Michael Friendly, Roeland Kindt, Pierre
##   Minchin, R. B. O'Hara, Gavin L. Simpson, Peter Solymos, M. Henry H. Stevens,
##   (2019). vegan: Community Ecology Package. R package version 2.5-6. https://CRAN.R-project.org/package=vegan
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {vegan: Community Ecology Package},
##   author = {Jari Oksanen and F. Guillaume Blanchet and Michael Friendly and
##             ...},
##   year = {2019},
##   note = {R package version 2.5-6},
##   url = {https://CRAN.R-project.org/package=vegan},
```

# Erros!!!

Se seu script rodou sem erros, tem algo errado... (Maurício Vancine)



I'm fine



BugCat-CAPOO

# 1.14 Principais erros

## 1. Esquecer de completar um comando (+)

Parênteses

```
sum(1, 2  
+
```

```
## Error: <text>:3:0: unexpected end of input  
## 1: sum(1, 2  
## 2: +  
##     ^
```

Aspas

```
"string  
+
```

```
## Error: <text>:1:1: unexpected INCOMPLETE_STRING  
## 1: "string  
## 2: +
```

# 1.14 Principais erros

## 2. Esquecer da vírgula

```
sum(1 2)
```

```
## Error: <text>:1:7: unexpected numeric constant
## 1: sum(1 2
##                  ^
```

# 1.14 Principais erros

## 3. Chamar um objeto errado

```
obj <- 10  
OBJ
```

```
## Error in eval(expr, envir, enclos): object 'OBJ' not found
```

# 1.14 Principais erros

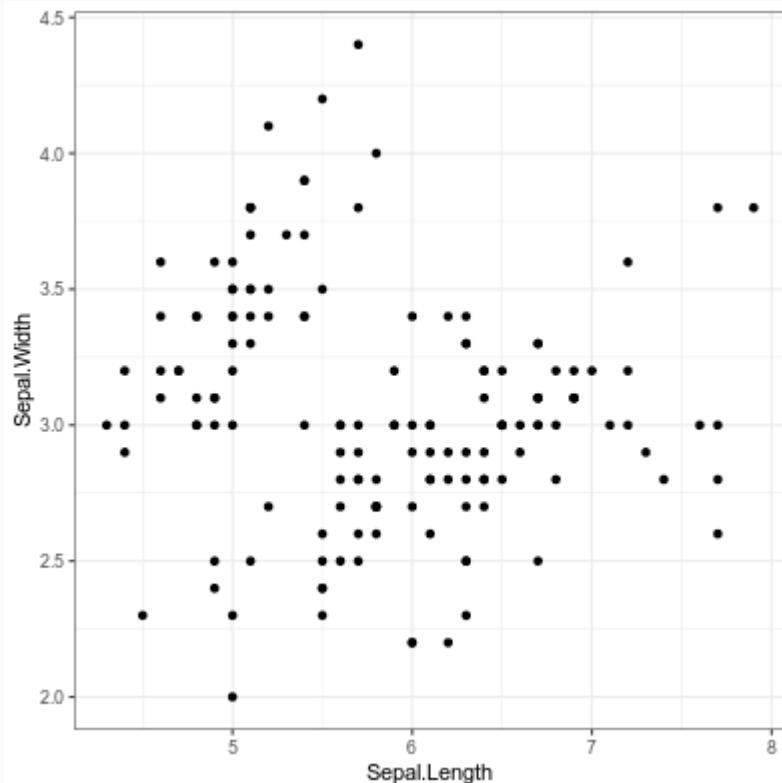
## 4. Esquecer de carregar um pacote

```
ggplot(iris) + aes(Sepal.Length, Sepal.Width) + geom_point()
```

# 1.14 Principais erros

## 4. Esquecer de carregar um pacote

```
library(ggplot2)  
ggplot(iris) + aes(Sepal.Length, Sepal.Width) + geom_point() + theme_bw()
```



# 1.14 Principais erros

## 5. Usar o nome da função de forma errônea

```
rowSums(iris[1:10, -5])
```

```
##   1    2    3    4    5    6    7    8    9    10
## 10.2 9.5 9.4 9.4 10.2 11.4 9.7 10.1 8.9 9.6
```



# 1.14 Principais erros

## 5. Usar o nome da função de forma errônea

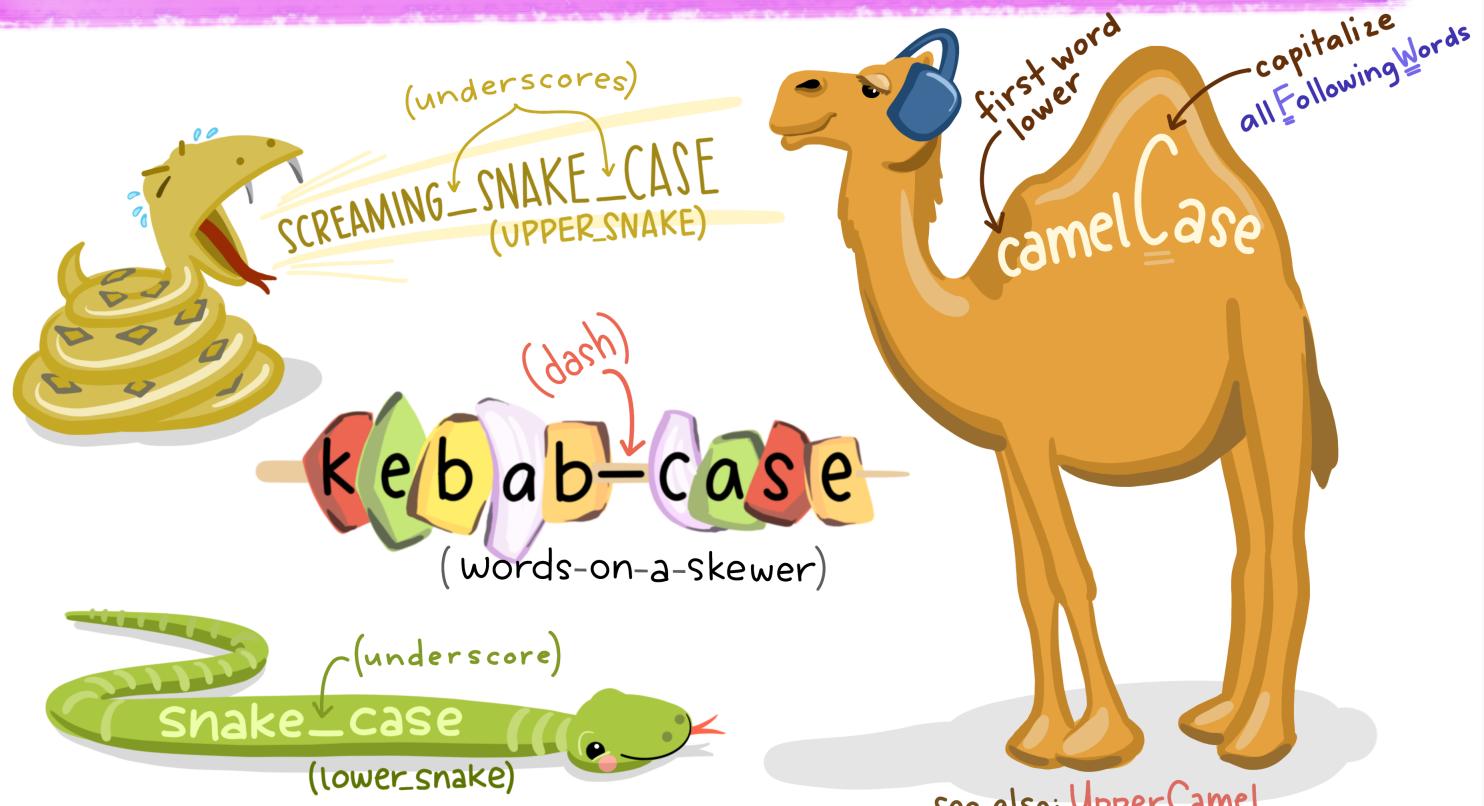
```
rowsums(iris[1:10, -5])
```

```
## Error in rowsums(iris[1:10, -5]): could not find function "rowsums"
```



# Cases

in that case...



Allison Horst

[\*] <https://github.com/allisonhorst/stats-illustrations>

@allison\_horst

131 / 133

Dúvidas?

# Maurício Vancine

Contatos:

 [mauricio.vancine@gmail.com](mailto:mauricio.vancine@gmail.com)

 [mauriciovancine](https://twitter.com/mauriciovancine)

 [mauriciovancine.netlify.com](https://mauriciovancine.netlify.com)

Slides criados via pacote [xaringan](#) e tema [Metropolis](#)