30ª Semana de Estudos da Biologia

Introdução à linguagem R: manipulação e visualização de dados

2 Funcionamento da linguagem R

Maurício Vancine
Helena Oliveira
Lucas Almeida

xaringan [presentation ninja]
23/10/2019


R Studio®

2 Funcionamento da linguagem R

Tópicos

2.1 RStudio

2.2 Editor/Roteiro (*code*/*script*)

2.3 Comentários (#)

2.4 Atribuição (<-)

2.5 Objetos

2.6 Operadores

2.7 Funções

2.8 Pacotes

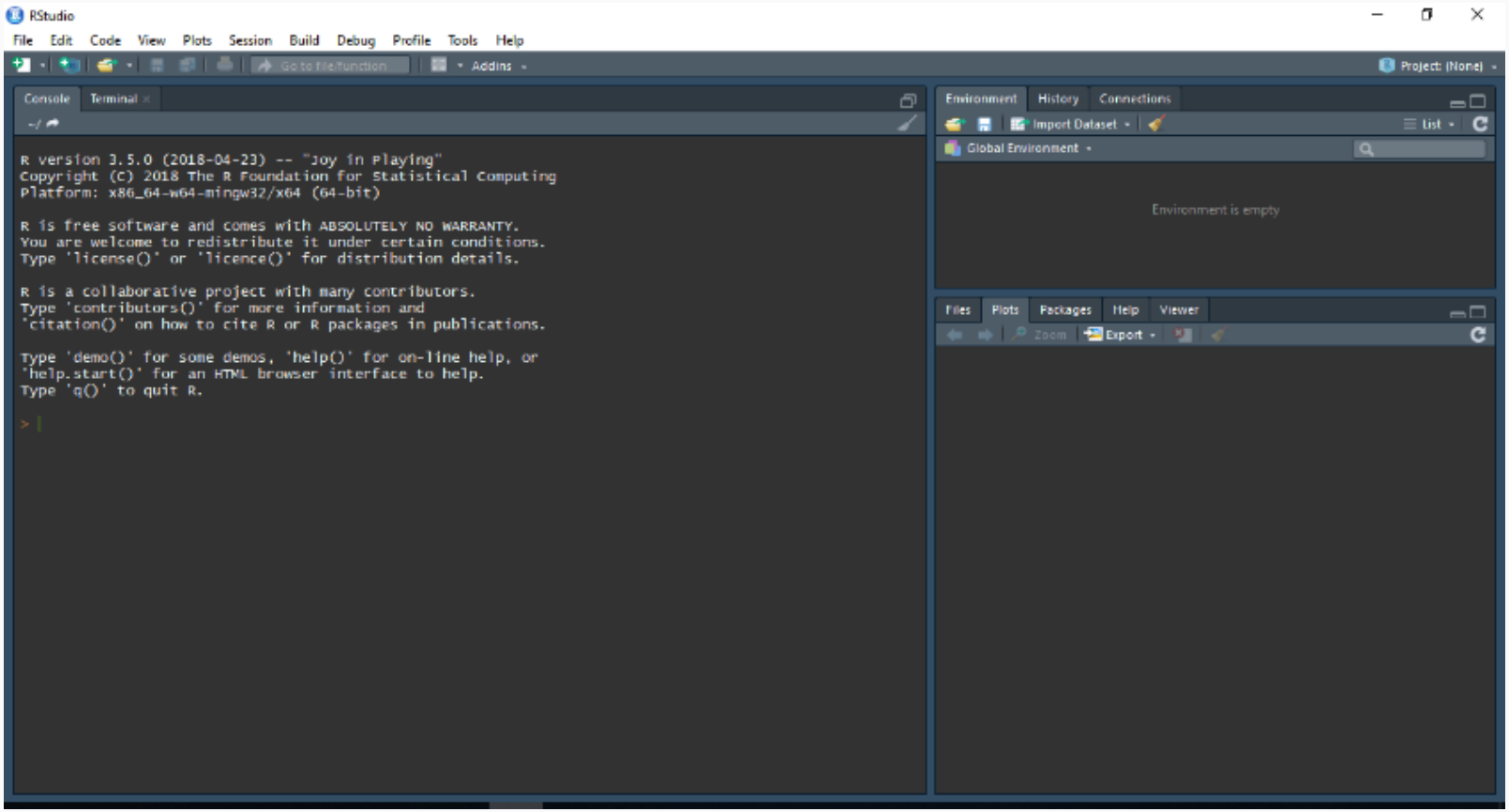
2.9 Ajuda (*help*)

2.10 Ambiente (*environment*/*workspace*)

2.11 Citações

2.12 Principais erros

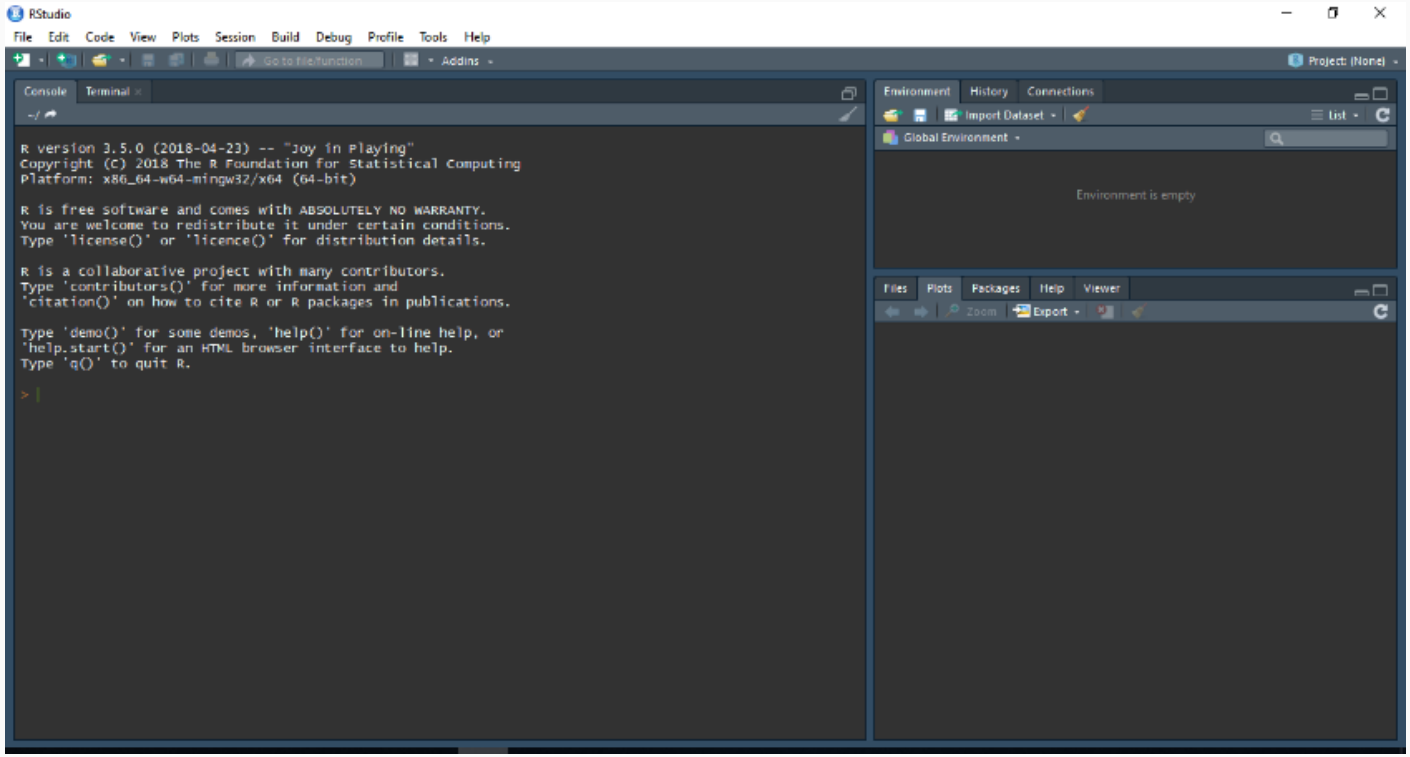
2.1 RStudio


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
~/
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export

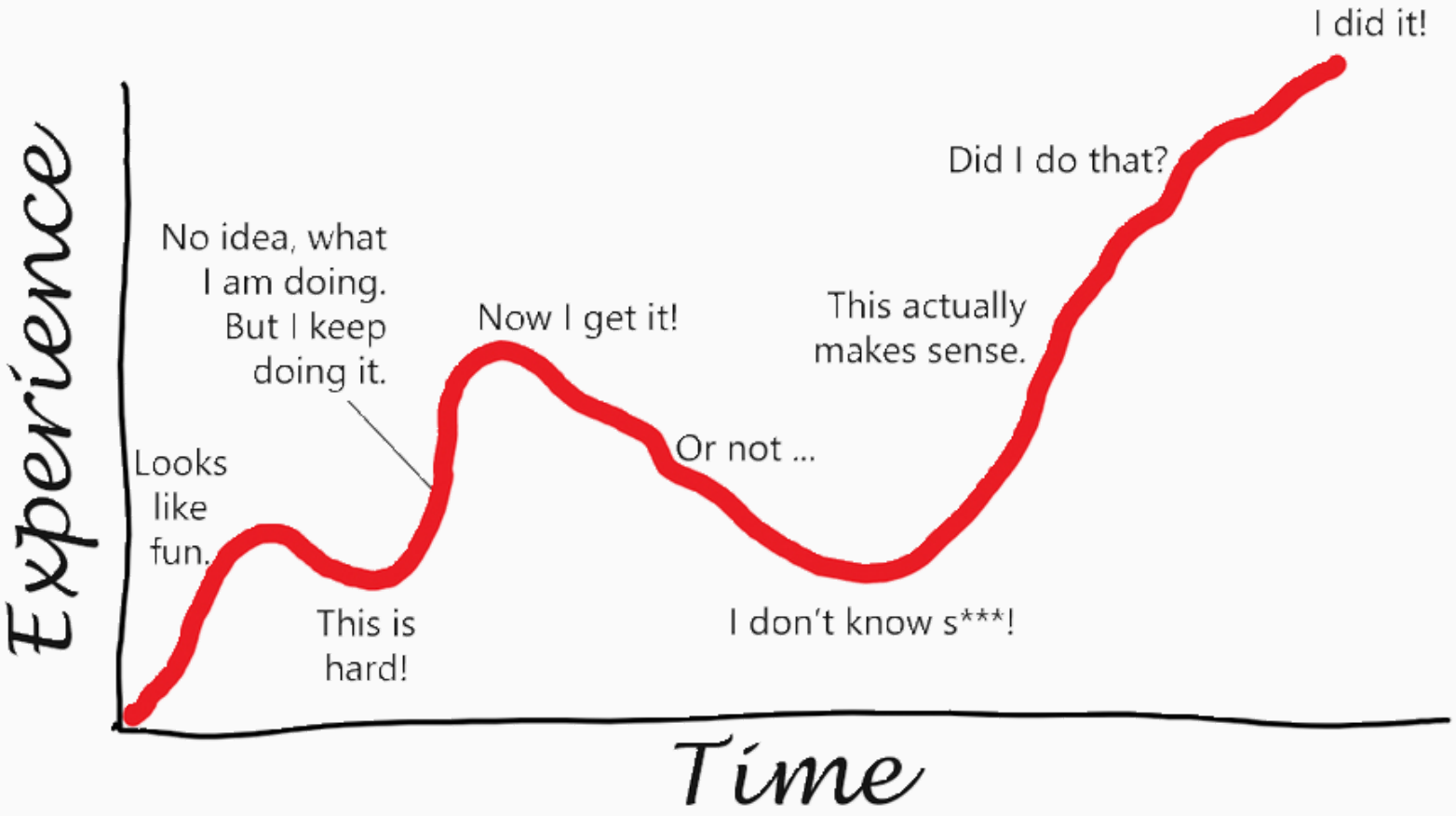
2.1 RStudio

Customizá-lo

`Ferramentas (Tools) -> Opções Globais (Global Options) -> Aparência (Appearance) -> Tema do editor (Editor Theme)`



Programar


I did it!
Did I do that?
No idea, what I am doing. But I keep doing it.
Now I get it!
This actually makes sense.
Looks like fun.
Or not ...
This is hard!
I don't know s***!
Experience
Time

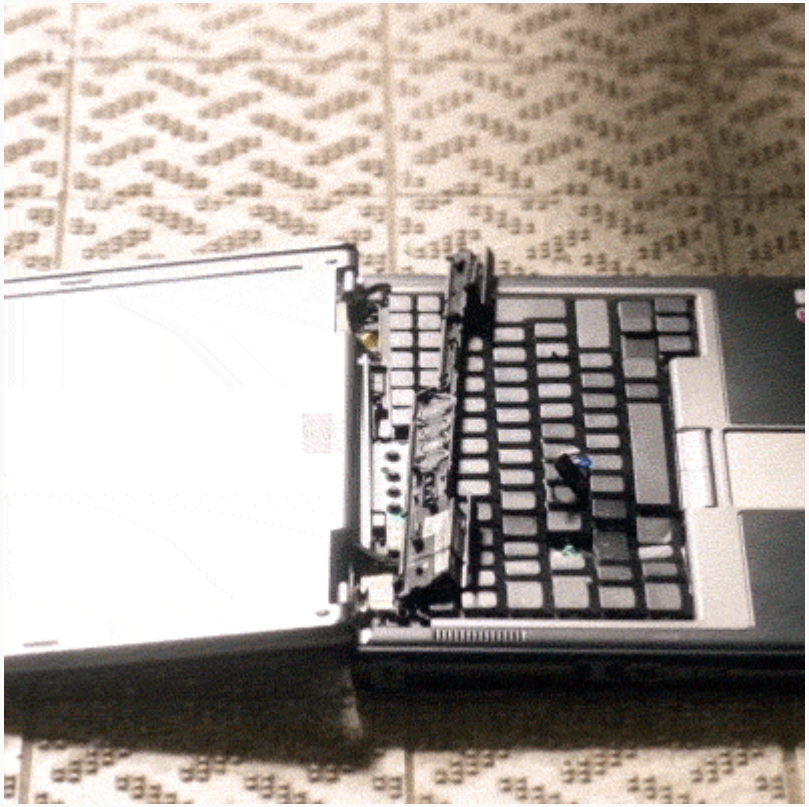
Programar

Ter calma e paciência...

2.1 RStudio

Programar

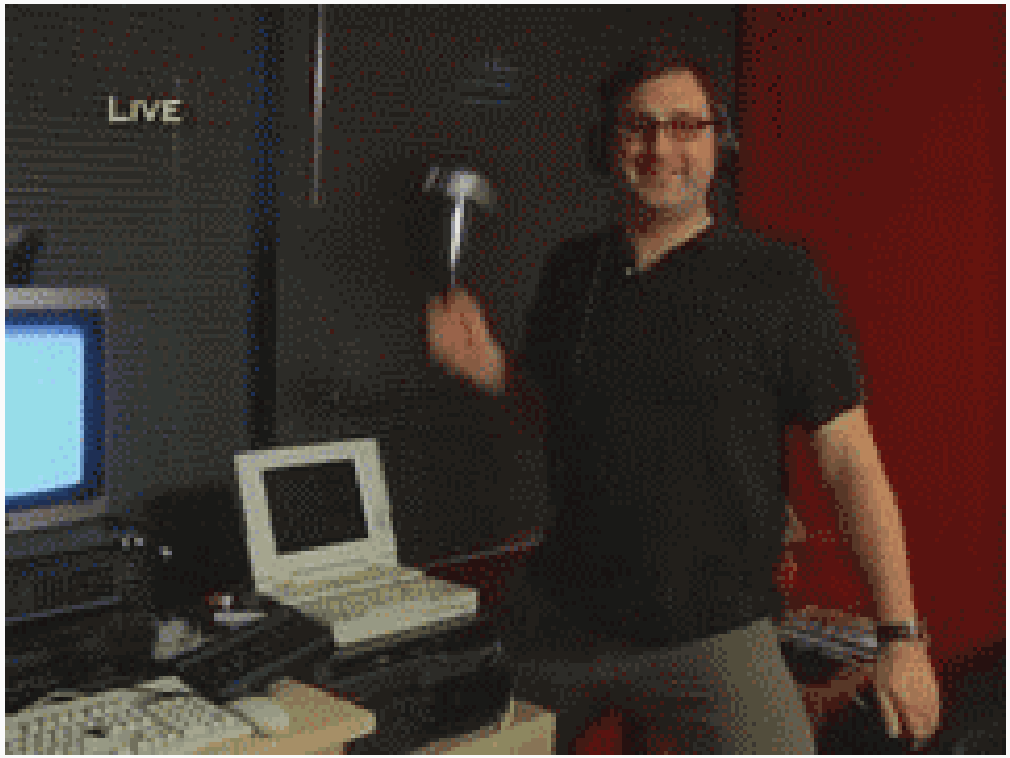
É muito complicado no começo...



2.1 RStudio

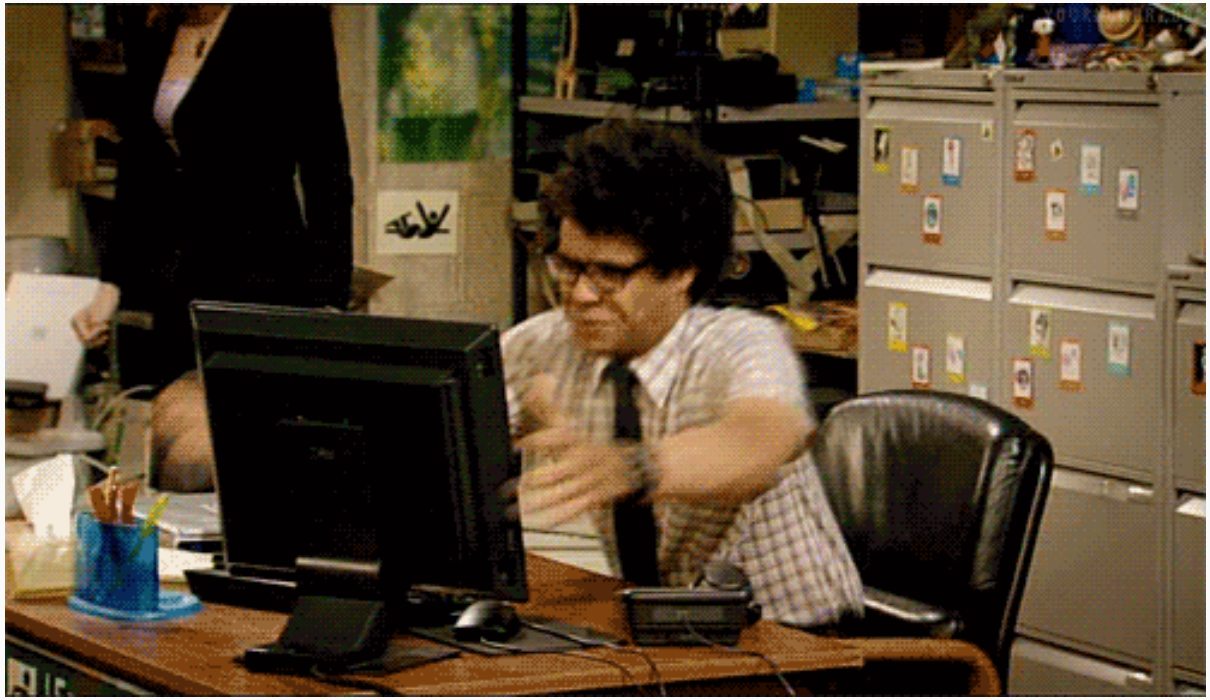
Programar

Tentem não ficar furioso(a)s...



Programar

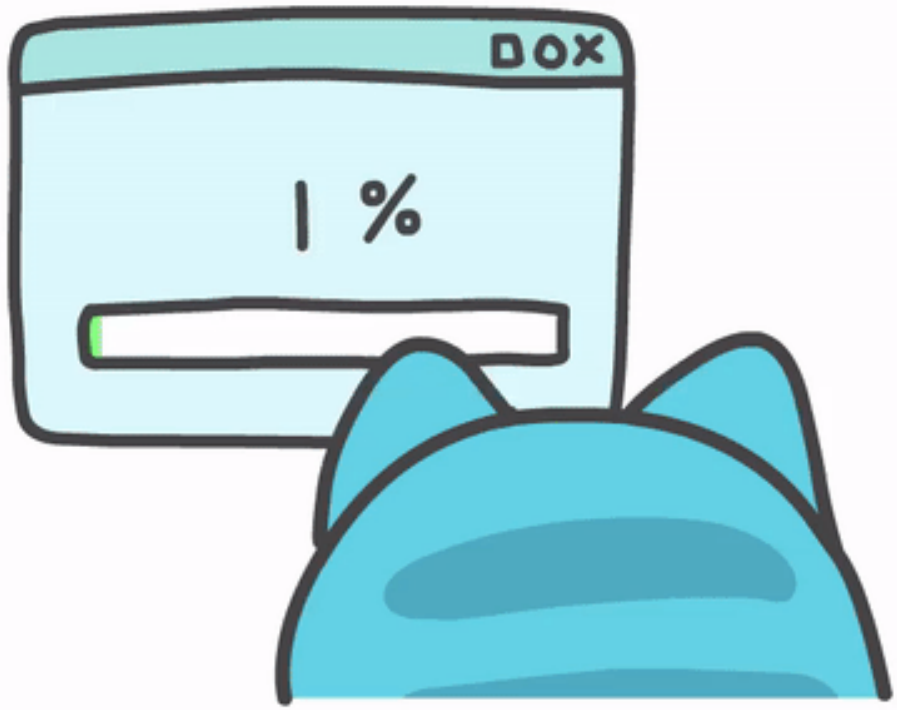
E principalmente, não desistam...



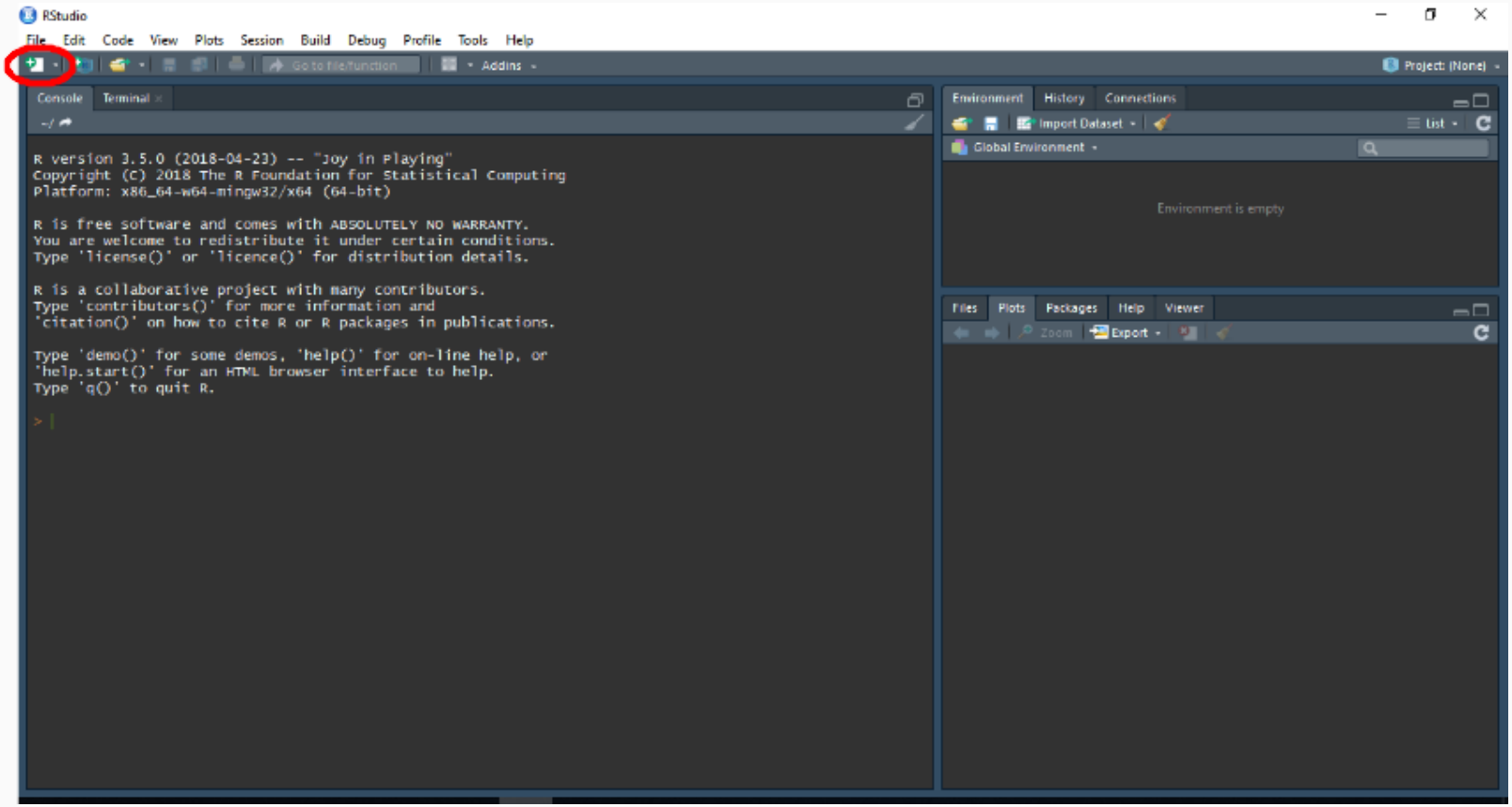
2.1 RStudio

Programar

Principalmente quando acontecer um erro...

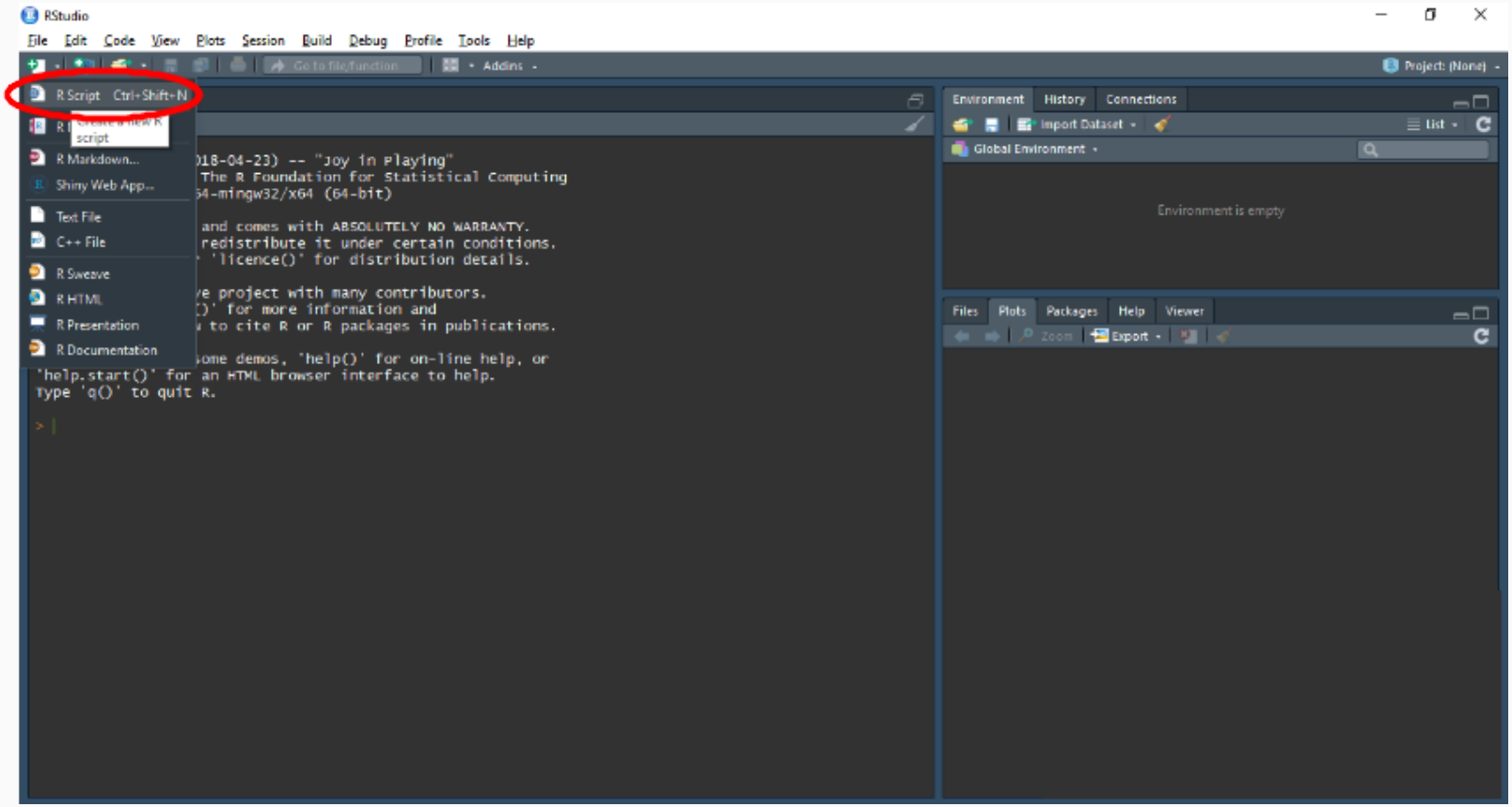


2.1 RStudio

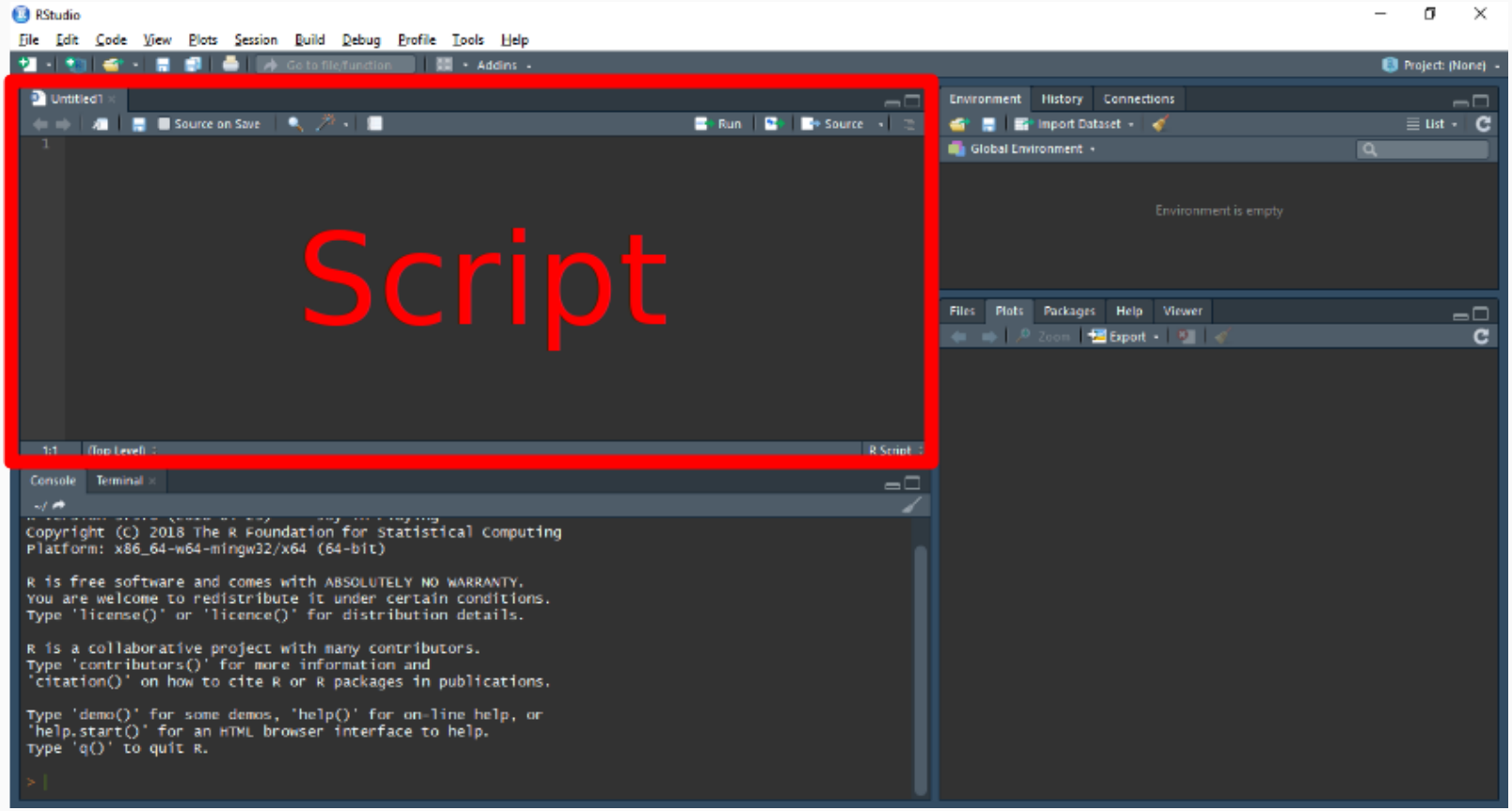

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
Project: (None)
Console Terminal
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Zoom
Export

2.1 RStudio

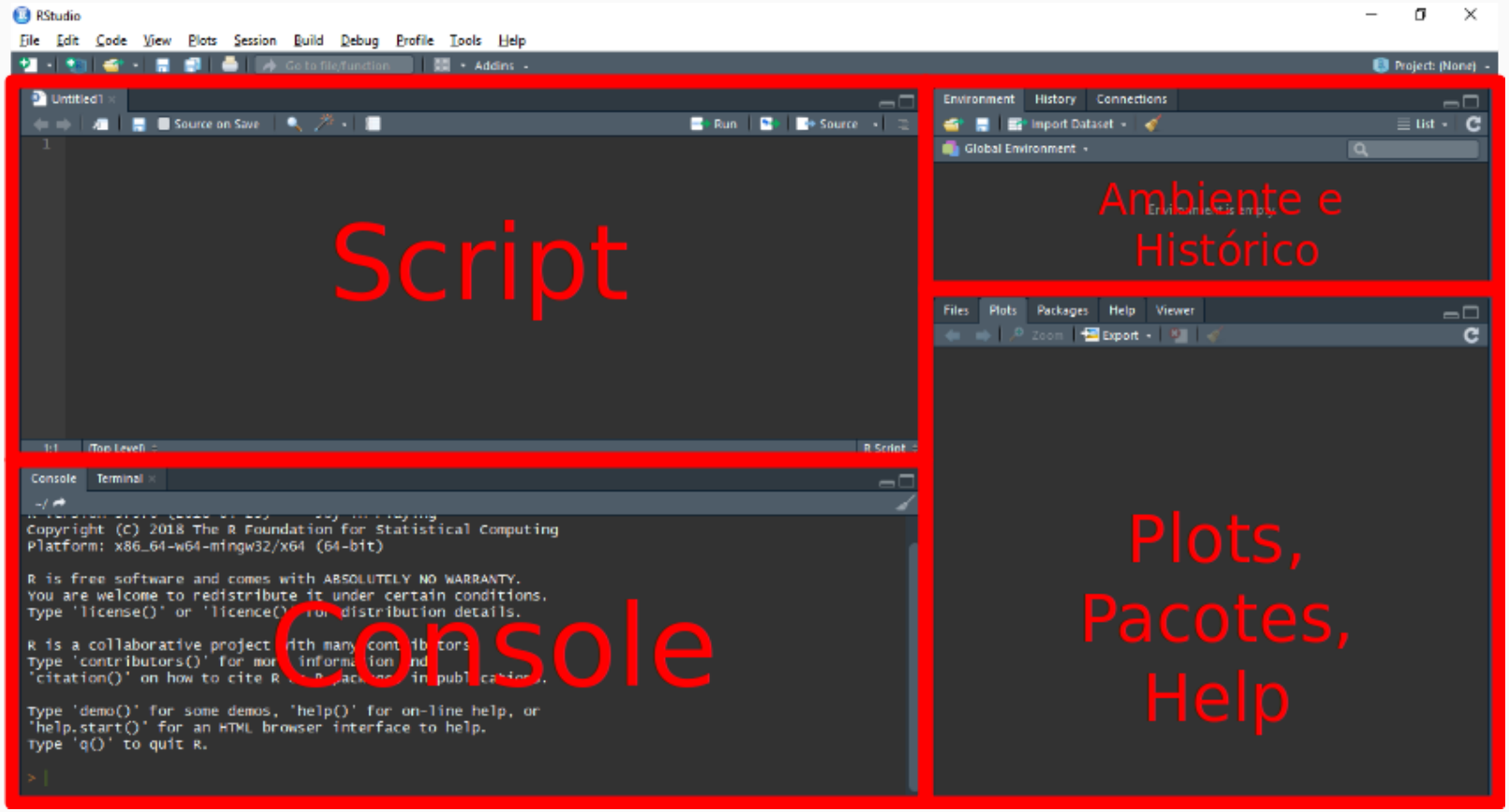
`R Script (Ctrl + Shift + N)`



2.1 RStudio


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1
Source on Save
Run
Source
Script
Environment History Connections
Import Dataset
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Export
Console Terminal
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

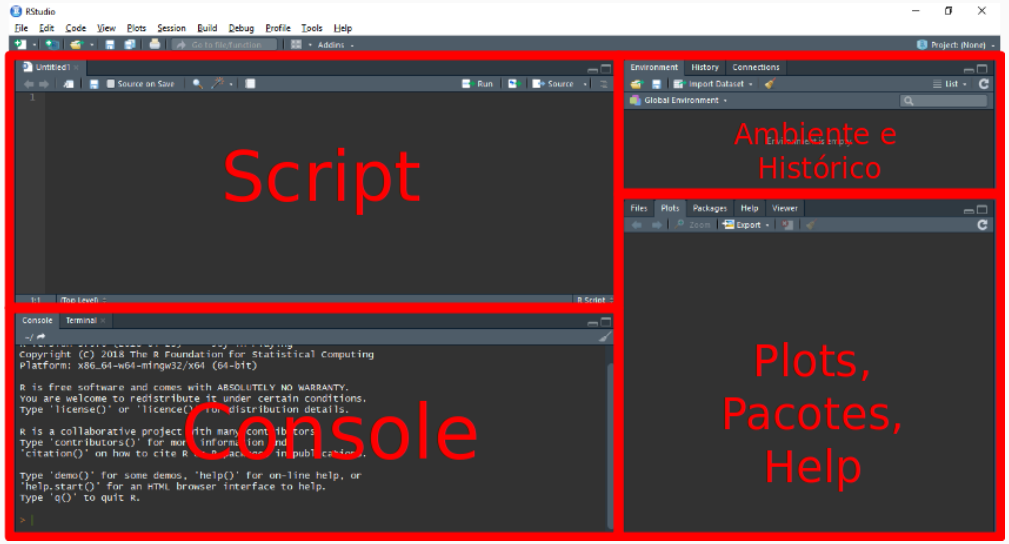
2.1 RStudio



2.1 RStudio

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



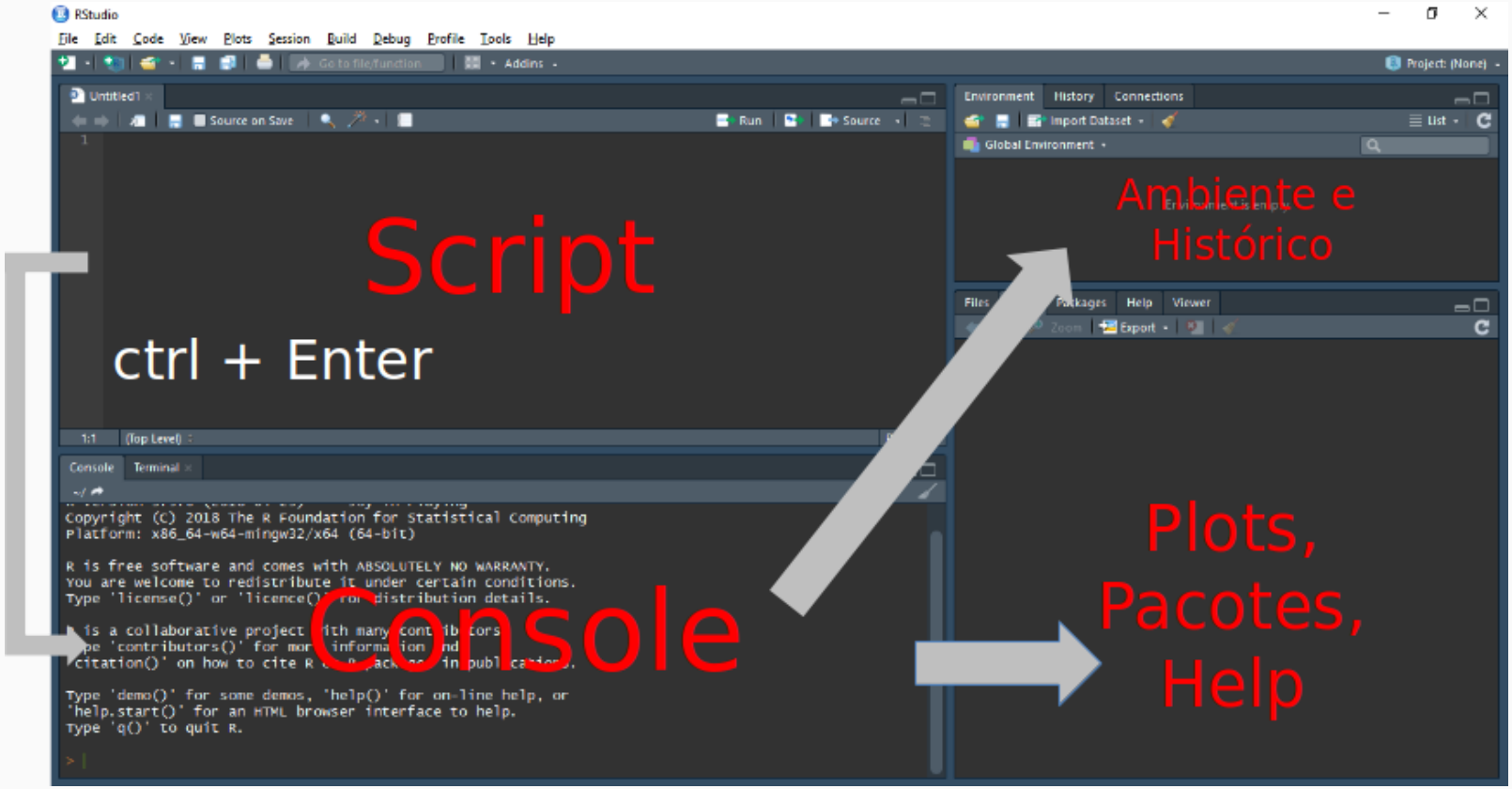
2.1 RStudio

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

2.1 RStudio

Funcionamento

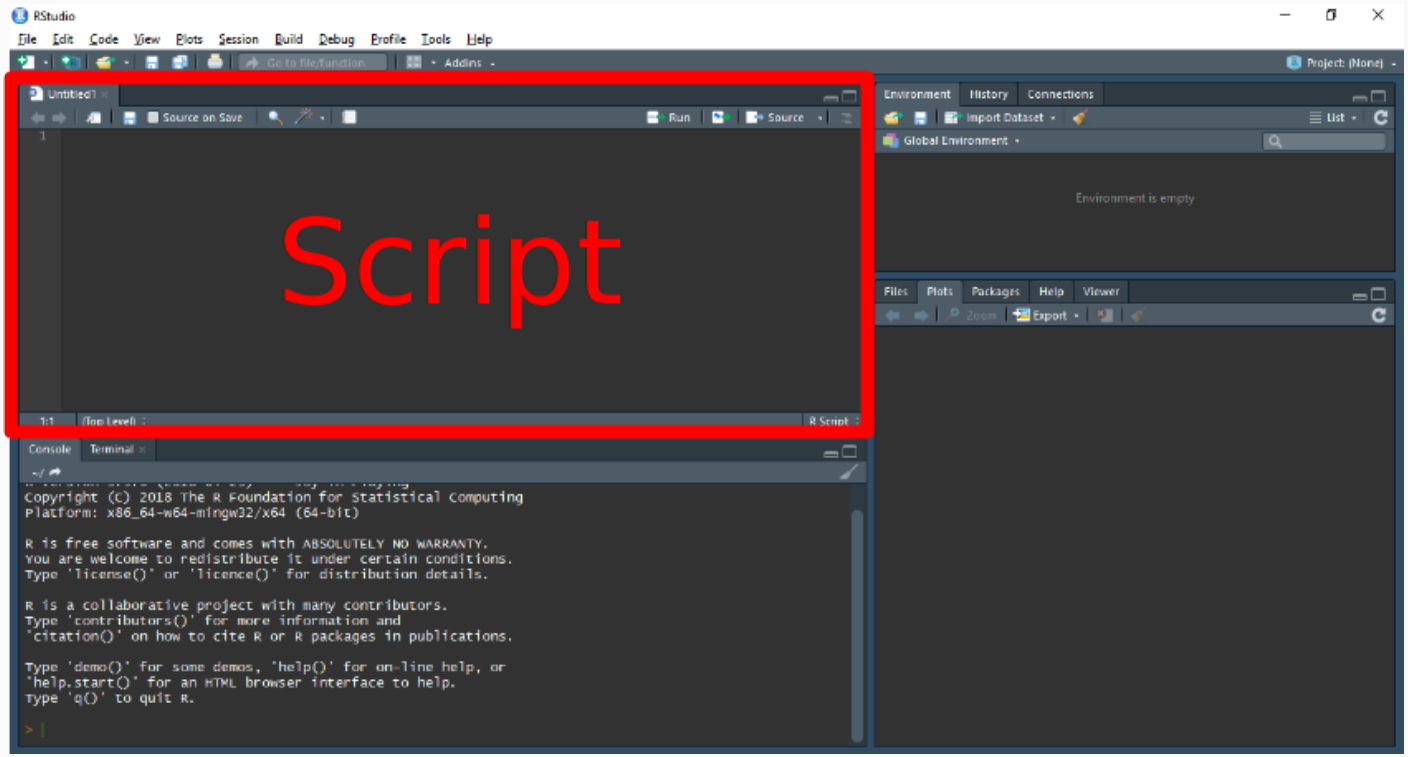


Dúvidas?

2.2 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



2.2 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"
```

2.2 Editor/Roteiro (*code*/*script*)

Escolham uma linha e digitem no *script*:

```
"Meu primeiro script no R."
"Meu segundo script no R."
"Meu milésimo script no R."
"Meu n-ésimo script no R."
```

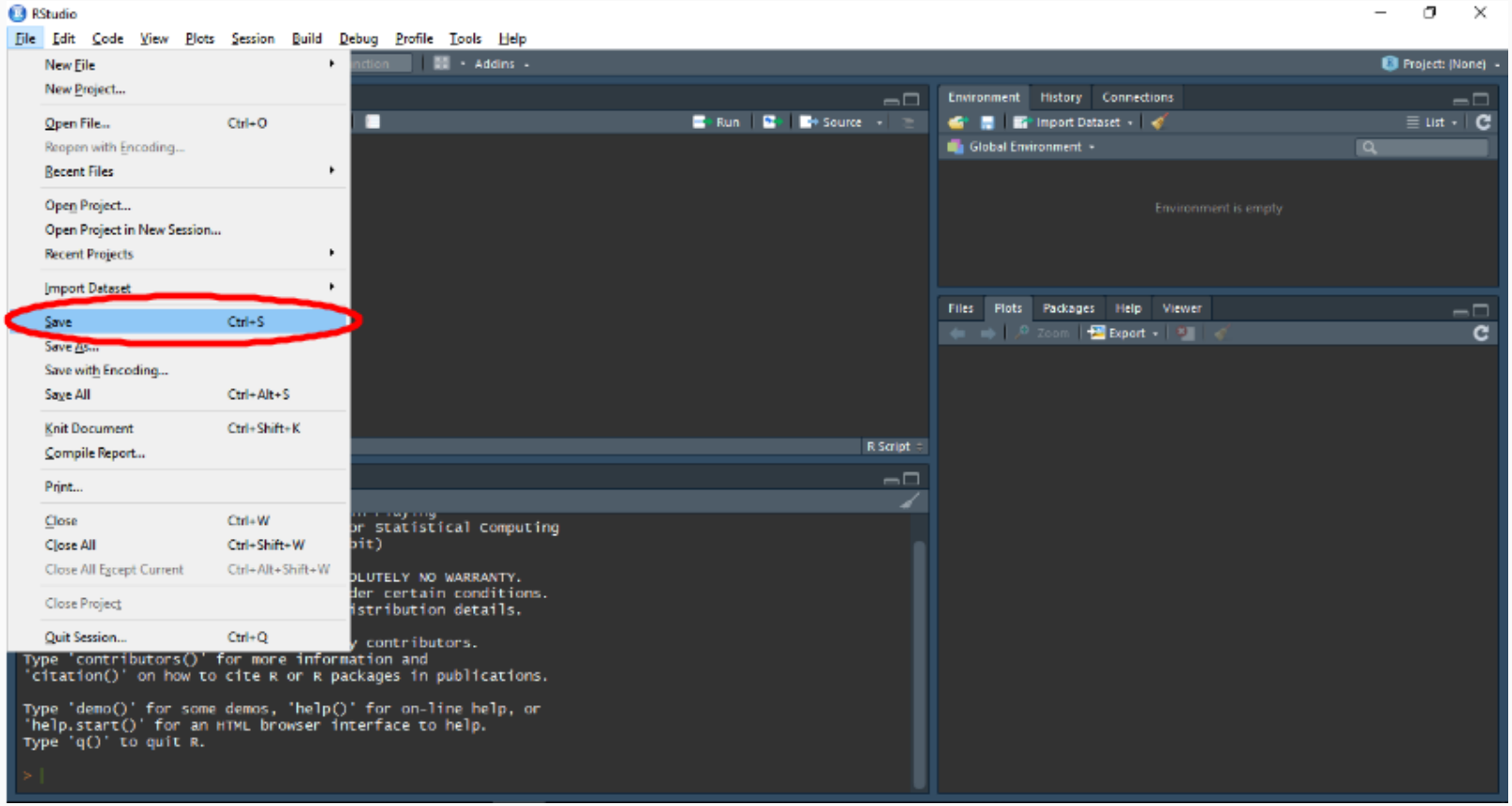
2.2 Editor/Roteiro (*code/script*)

Escolher UTF-8

`Ferramentas (Tools) -> Opções Globais (Global Options) -> (Code) -> s (Saving) -> Default text endcoding (UTF-8)`

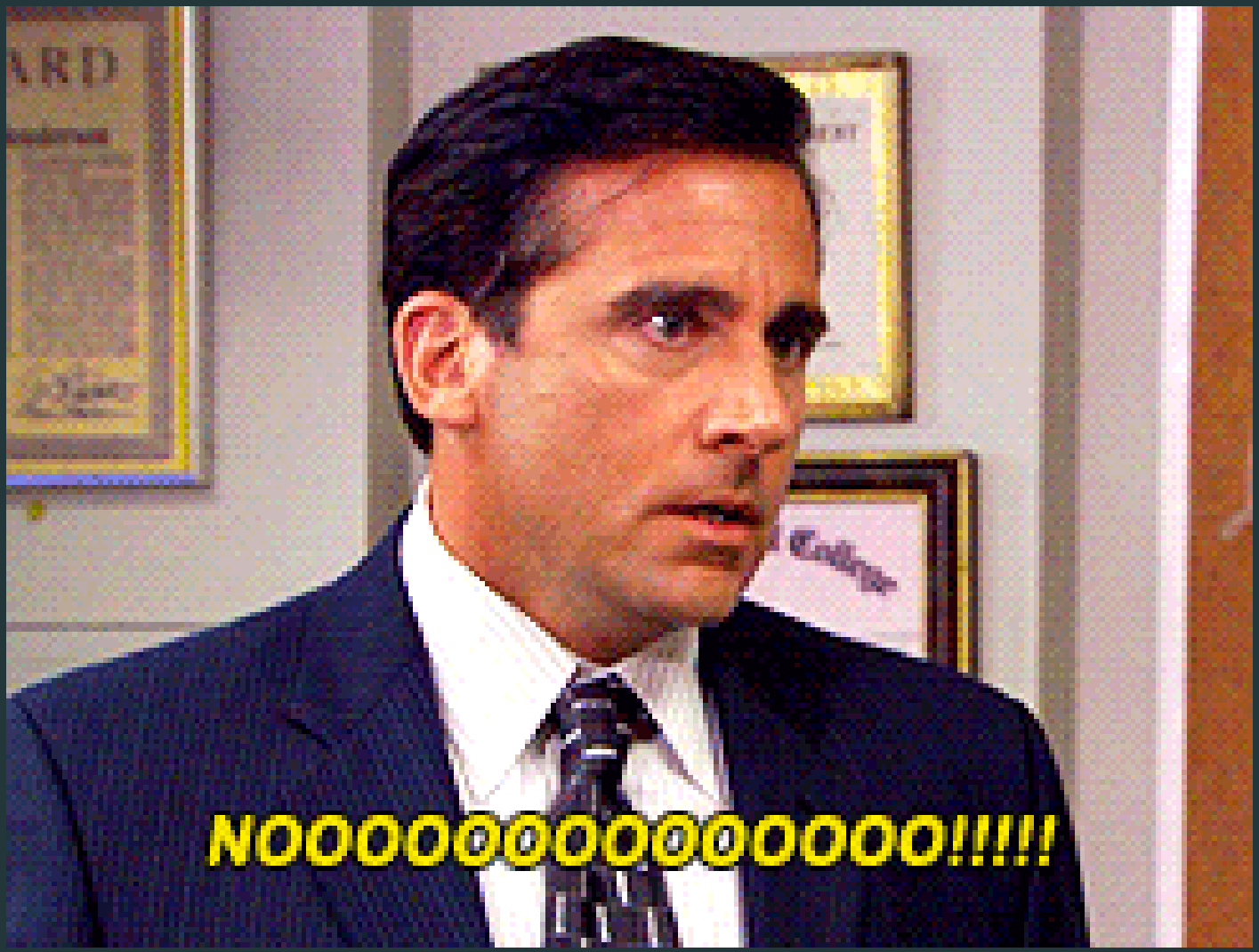
2.2 Editor/Roteiro (*code/script*)

`ctrl + S`



Calma!!!

Onde vocês iam salvar?!


NOOOOOOOOOOOOOO!!!!!

Vamos padronizar:

Pasta do diretório `/minicurso-r-sebio-2019/`:

00_ementa

01_aulas

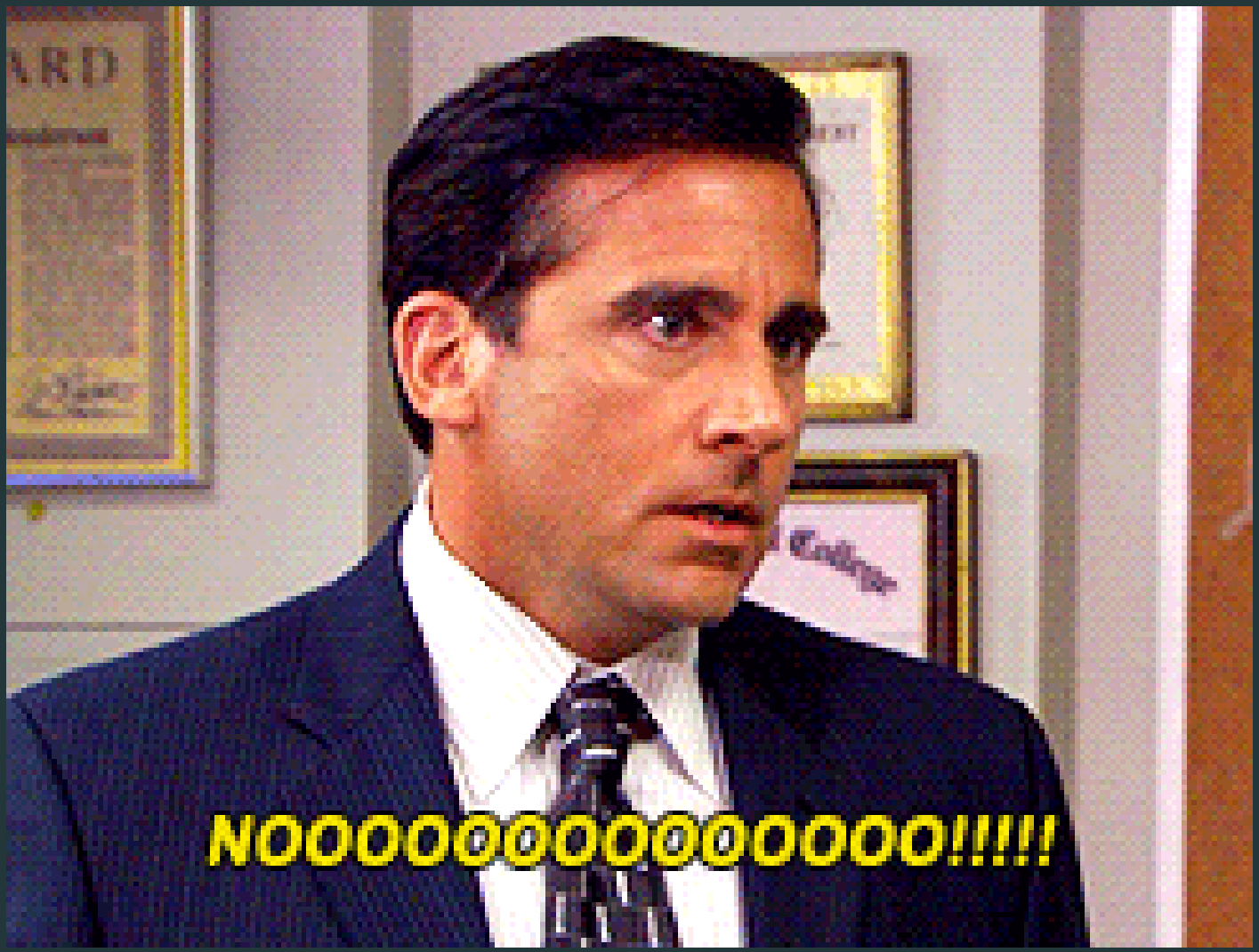
02_scripts

03_dados

04_alunos

Calma!!!

Que nome vocês iam salvar?!


NOOOOOOOOOOOOOO!!!!!

Vamos padronizar

`script_aula_02.R`

Sigam as instruções

1. Fechem o RStudio
2. Abram o RStudio
3. Abram o script salvo `script_aula_02.R`
4. Apaguem seu conteúdo

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
```

Todos entenderam como iremos fazer até o final desse minicurso?

Muito bem, apaguem todo o conteúdo desse script

Dúvidas?

2.3 Comentários (#)

Comentários **não são lidos** pelo R

São úteis para **descrever informações** sobre o **script** ou **comandos**

Cabeçalho

```
### aula 02 - funcionamento do r ###

# seu nome
# 22-10-2019
```

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
```

Vocês são organizados?

Aqui organização funciona como seleção natural... =]


BIRD

Organização dos scripts

Se você vai usar o R para fazer suas análises, a primeira coisa é ser organizado

Separe as linhas das análises e comente cada comando

Você vai se agradecer no futuro =]

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
```

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 treta...

$$9 - 3 \div \frac{1}{3} + 1 = ?$$

Exercício 01

Resposta

```
# exercicio 01
9 - 3 / 1/3 + 1
```

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

Alguém notou esses 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
## [24] 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
## [47] 56 57 58 59 60
```

Fácil até aqui? Então vamos complicar...

2.4 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 + -`

2.4 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....

2.4 Atribuição (<-)

Sempre **confira** a atribuição!!!

Chame o objeto **novamente**!!!

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

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

Outro exemplo:

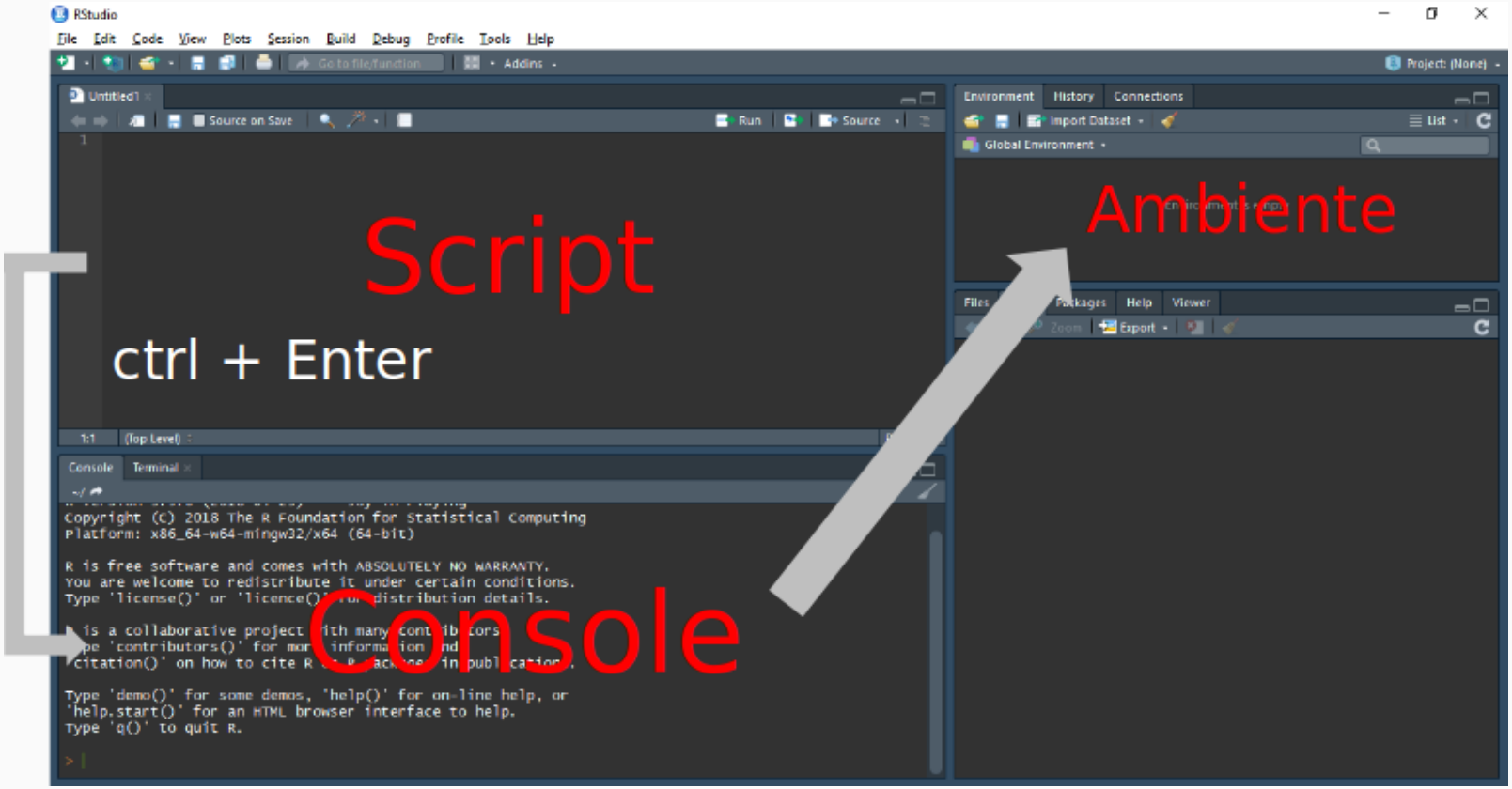
```
## atribuicao - simbolo (<-)
obj_2 <- 2
obj_2
```

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

Os objetos podem ser visualizados no painel *Environment*

2.4 Atribuição (<-)

Ambiente (*Environment*)



2.4 Atribuição (<-)

CUIDADO 1!

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
```

2.4 Atribuição (<-)

CUIDADO 2!

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, números, 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**
5. Nome de objetos não podem ser iguais a **nomes especies**:

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

Dúvidas?

2.5 Objetos

Podemos utilizar **objetos** para fazer operações

```
# definir dois objetos
va1 <- 10
va1
```

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

```
va2 <- 2
va2
```

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

2.5 Objetos

Podemos utilizar **objetos** para fazer operações

```
# operacoes com objetos
va1 + va2 # adicao
```

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

```
va1 - va2 # subtracao
```

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

2.5 Objetos

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

```
# operacoes com objetos e atribuicao
adi <- va1 + va2 # adicao
adi
```

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

```
sub <- va1 - va2 # subtracao
sub
```

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

2.6 Operadores

Operadores aritméticos

Resultados numéricos

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

2.6 Operadores

Operadores relacionais

Resultados Booleanos (TRUE ou FALSE)

Operador	Descrição	Uso
<	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 $3x2^3$ é maior que $2x3^2$

Resposta

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

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

Exercício 03

Verifique se o resto da divisão de 12567/34 é menor ou igual ao quociente da divisão 3213/123

Resposta

```
# exercicio 03
12567%%34 <= 3213%/%123
```

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

Dúvidas até aqui?

2.7 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
```

2.7 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
```

Argumentos como **valores**

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

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

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

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

Argumentos como **objetos**

```
# funcoes - argumentos como objetos
# soma
sum(va1, va2)
```

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

```
# produto
prod(va1, va2)
```

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

Argumentos como **parâmetros**

```
# funcoes - nome dos argumentos
# repeticao - todos
rep(x = 1:5, times = 10)
```

```
##  [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
## [36] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
# 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 4 4 4 4 4
## [36] 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5
```

Atribuir resultados das funções à objetos

```
## atribuicao dos resultados
# repeticao - todos
rep_times <- rep(x = 1:5, times = 10)
rep_times
```

```
##  [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 3 4 5
## [36] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
## atribuicao dos resultados
# repeticao - todos
rep_each <- rep(x = 1:5, each = 10)
rep_each
```

```
##  [1] 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
## [36] 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5
```

Nesse momento, vocês já foram apresentados à 50% do funcionamento do R

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 04

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 *mult* e atribuam ao objeto *loge*

Exercício 04

Resposta

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

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

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

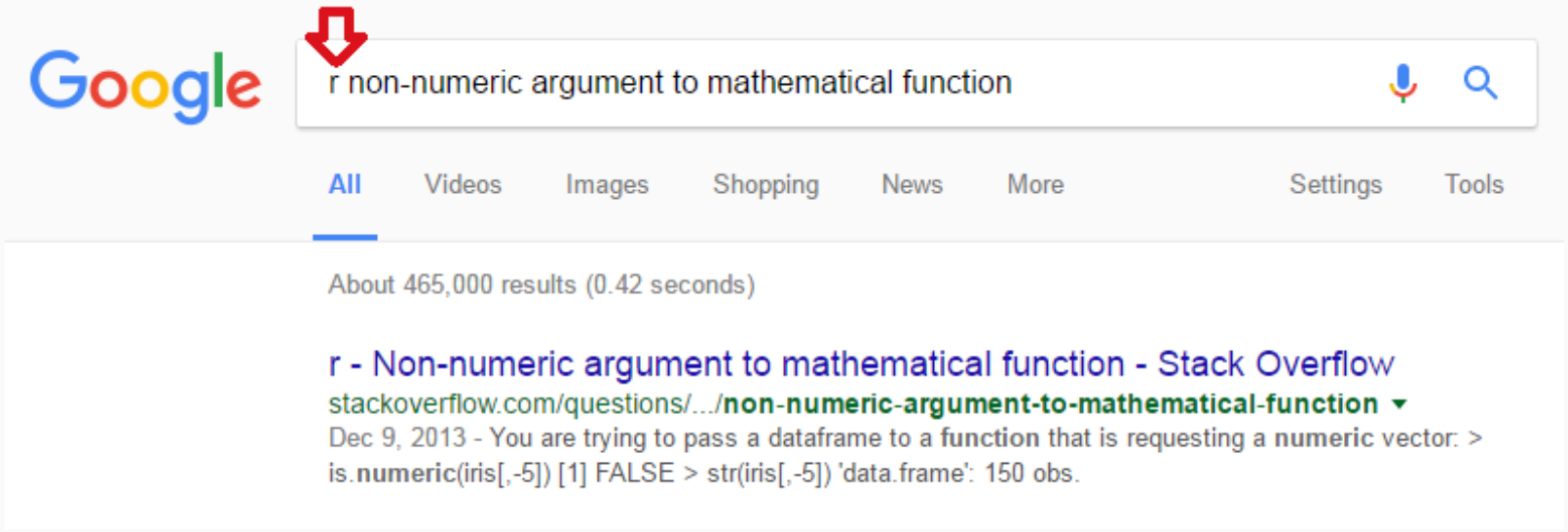
```
# raiz e atribuir
loge <- log(mult)
loge
```

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

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


GOOGLE IT! GOOGLE IT!

Uma **maracutaia** para ajudar!



E de onde vêm as funções?!

2.7 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?!

2.8 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:forcats"   "package:stringr"
##  [4] "package:dplyr"     "package:purrr"     "package:readr"
##  [7] "package:tidyr"     "package:tibble"    "package:ggplot2"
## [10] "package:tidyverse" "package:xaringan"  "package:pagedown"
## [13] "tools:rstudio"     "package:stats"     "package:graphics"
## [16] "package:grDevices" "package:utils"     "package:datasets"
## [19] "package:methods"   "Autoloads"         "package:base"
```

2.8 Pacotes

Coleção de funções para executar **tarefas específicas**

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

Verificar **pacotes instalados**

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

2.8 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

2.8 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")
```

2.8 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)
```

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")
```

2.8 Pacotes

Atualização de pacotes

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

Pacotes **não atualizam sozinhos**

É uma função que **demora** para rodar

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

E onde ficam esses pacotes no meu notebook?

2.8 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**

2.8 Pacotes

Exemplos:

vegan – análises de comunidades

raster – manejo de rasters

ggplot2 – gráficos

bblme – seleção de modelos (AIC)

dismo – modelos de distribuição de espécies

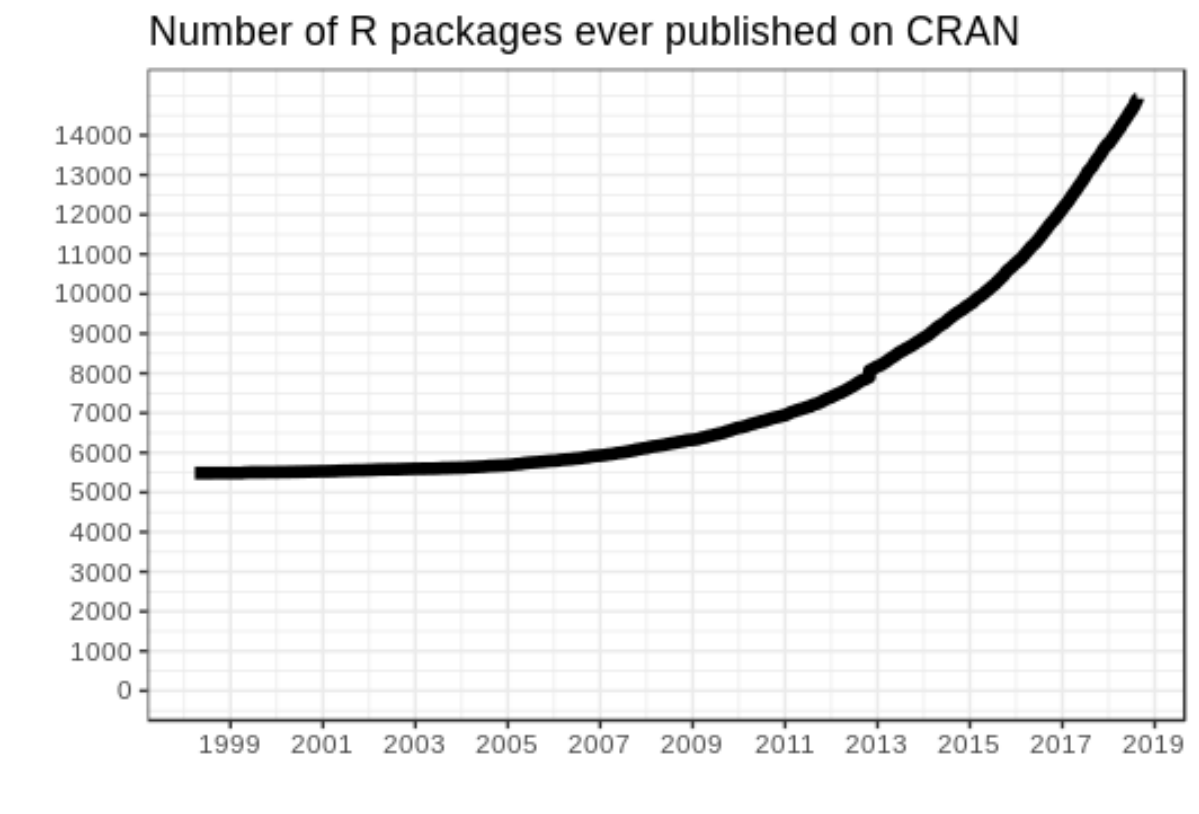
tidyverse – data science

E quantos pacotes existem?

2.8 Pacotes

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

```
## [1] 15128
```



https://cran.r-project.org/web/packages/available_packages_by_name.html

Exercícios

Exercício 05

Instalem o pacote **tidyverse** do CRAN

Resposta

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

Alguém aqui lê o manual de alguma coisa?

Helena não vale...

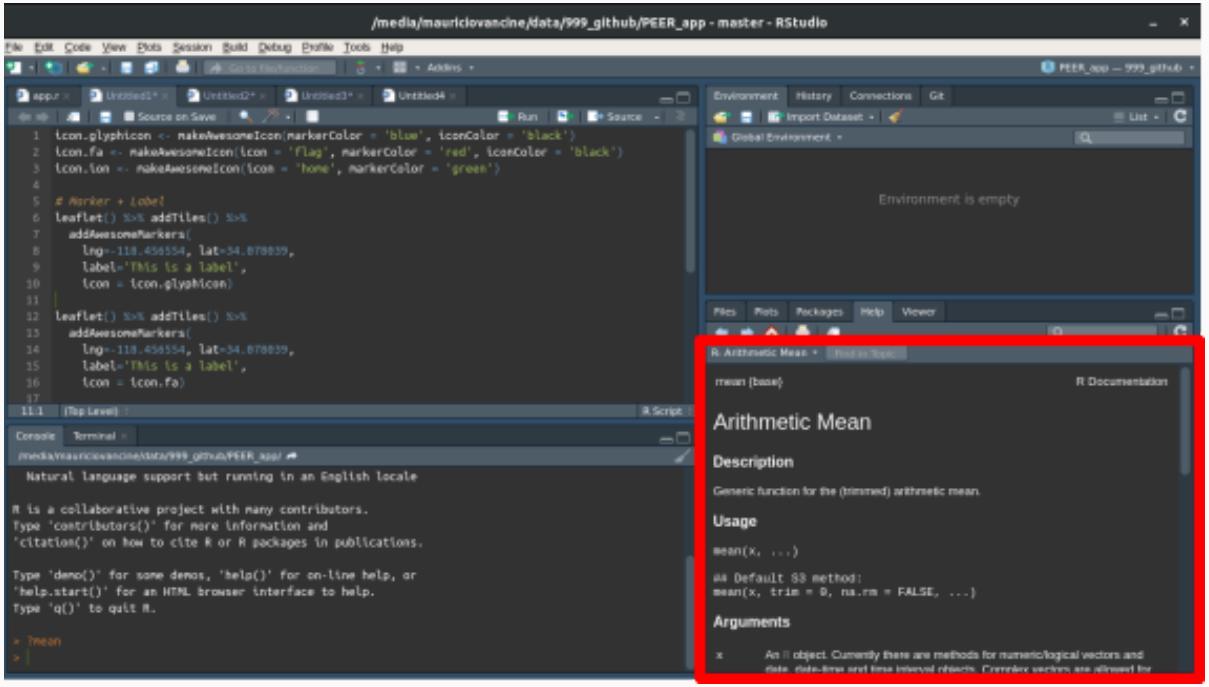
2.9 Ajuda (*help*)

Descreve as informações de uma função

```
## ajuda
# descreve as informacoes de uma funcao

help("mean") # arquivo .html

?mean
```



2.9 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))
```

2.9 Ajuda (*help*)

Resumo do *help*

1. **Description**: faz um resumo geral sobre o uso da função
2. **Usage**: mostra como a função deve ser utilizada e quais argumentos podem ser especificados
3. **Arguments**: explica o que é cada um dos argumentos
4. **Details**: explica alguns detalhes sobre o uso e aplicação da função (geralmente poucos)
5. **Value**: mostra o que sai no output após usar a função (os resultados)
6. **Note**: notas sobre a função
7. **Authors**: lista os autores da função (quem escreveu os códigos em R)
8. **References**: referências para os métodos usados
9. **See also**: mostra outras funções relacionadas que podem ser consultadas
10. **Examples**: exemplos do uso da função. Copie e cole os exemplos no R para ver como funciona

2.9 Ajuda (*help*)

Detalhes de um pacote

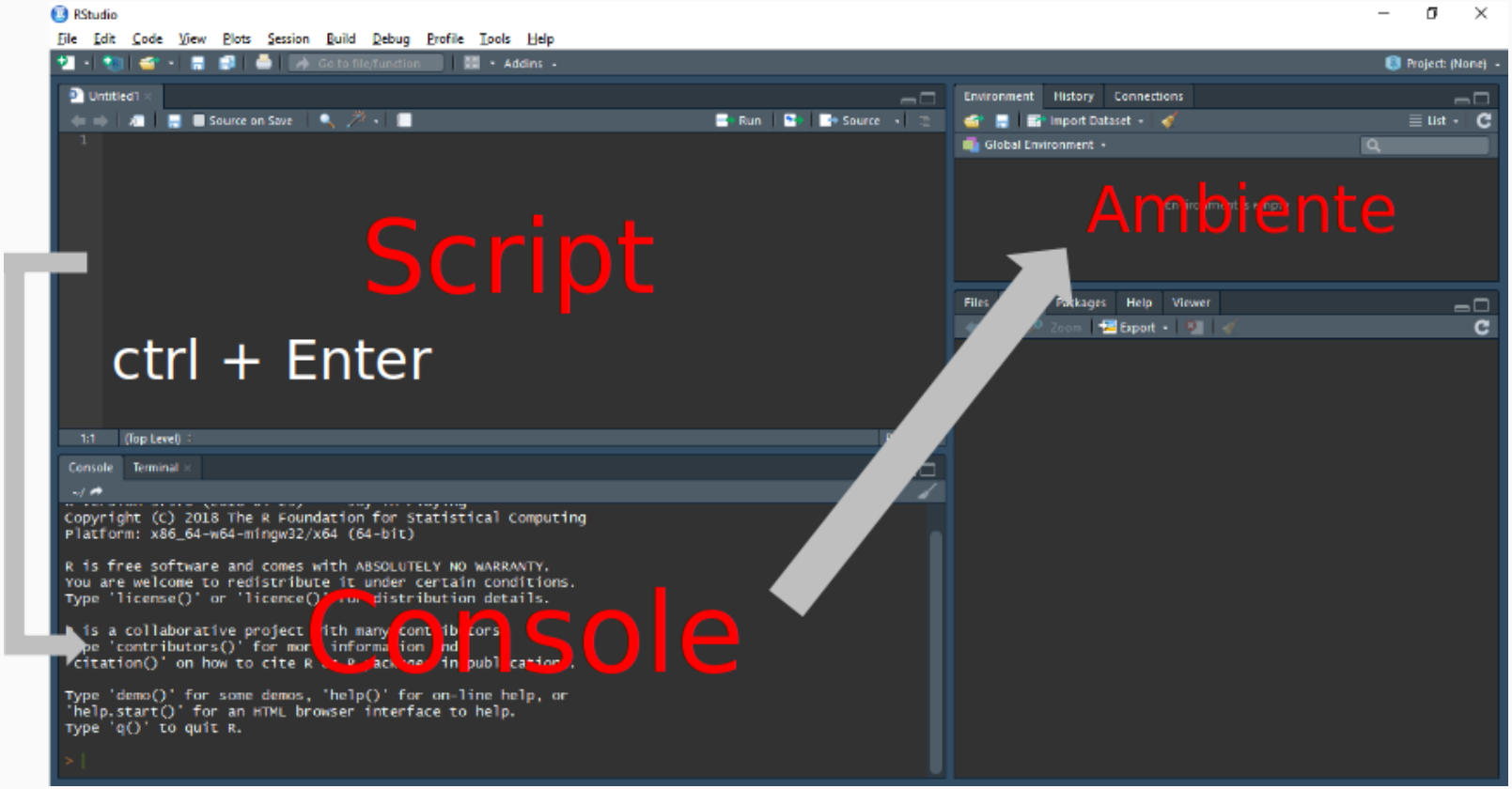
```
library(help = "vegan")
```

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

Todos se lembram da atribuição e criação de objetos?

palavra <- dados

2.10 Ambiente (*environment/workspace*)



Não seria legal se pudéssemos listar ou remover esses objetos?

2.10 Ambiente (*environment/workspace*)

Listar todos os objetos criados

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

```
##  [1] "a"         "adi"       "b"         "bar"       "foo"
##  [6] "i"         "loge"      "mult"      "obj"       "obj_10"
## [11] "obj_2"     "rep_each"  "rep_times" "sq"        "su"
## [16] "sub"       "va1"       "va2"
```

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

```
##  [1] "a"         "adi"       "b"         "bar"       "foo"
##  [6] "i"         "loge"      "mult"      "obj"       "obj_10"
## [11] "obj_2"     "rep_each"  "rep_times" "sq"        "su"
## [16] "sub"       "va1"       "va2"
```

2.10 Ambiente (*environment*/*workspace*)

CUIDADO 3!

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

2.10 Ambiente (*environment*/*workspace*)

Salvar todos os objetos criados

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

Carregar os objetos criados e salvos

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

2.10 Ambiente (*environment/workspace*)

Remover um objeto

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

```
##  [1] "a"         "adi"       "b"         "bar"       "foo"
##  [6] "i"         "loge"      "mult"      "obj"       "obj_10"
## [11] "obj_2"     "rep_each"  "rep_times" "sq"        "su"
## [16] "sub"       "va1"       "va2"
```

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

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

```
##  [1] "a"         "adi"       "b"         "foo"       "i"
##  [6] "loge"      "mult"      "obj"       "obj_10"    "obj_2"
## [11] "rep_each"  "rep_times" "sq"        "su"        "sub"
## [16] "va1"       "va2"
```

2.10 Ambiente (*environment/workspace*)

Remover todos os objetos

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

```
##  [1] "a"         "adi"       "b"         "foo"       "i"
##  [6] "loge"      "mult"      "obj"       "obj_10"    "obj_2"
## [11] "rep_each"  "rep_times" "sq"        "su"        "sub"
## [16] "va1"       "va2"
```

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

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

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

2.10 Ambiente (*environment*/*workspace*)

Carregar os objetos criados e salvos

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

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

```
##  [1] "adi"       "bar"       "foo"       "lo"        "mu"
##  [6] "obj"       "obj_10"    "obj_2"     "rep_each"  "rep_times"
## [11] "sq"        "su"        "sub"       "va1"       "va2"
```


CITATION?

2.11 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 (2019). R: A language and environment for
##   statistical computing. R Foundation for Statistical 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},
```

2.11 Citações

Como citar o R e os pacotes em trabalhos?

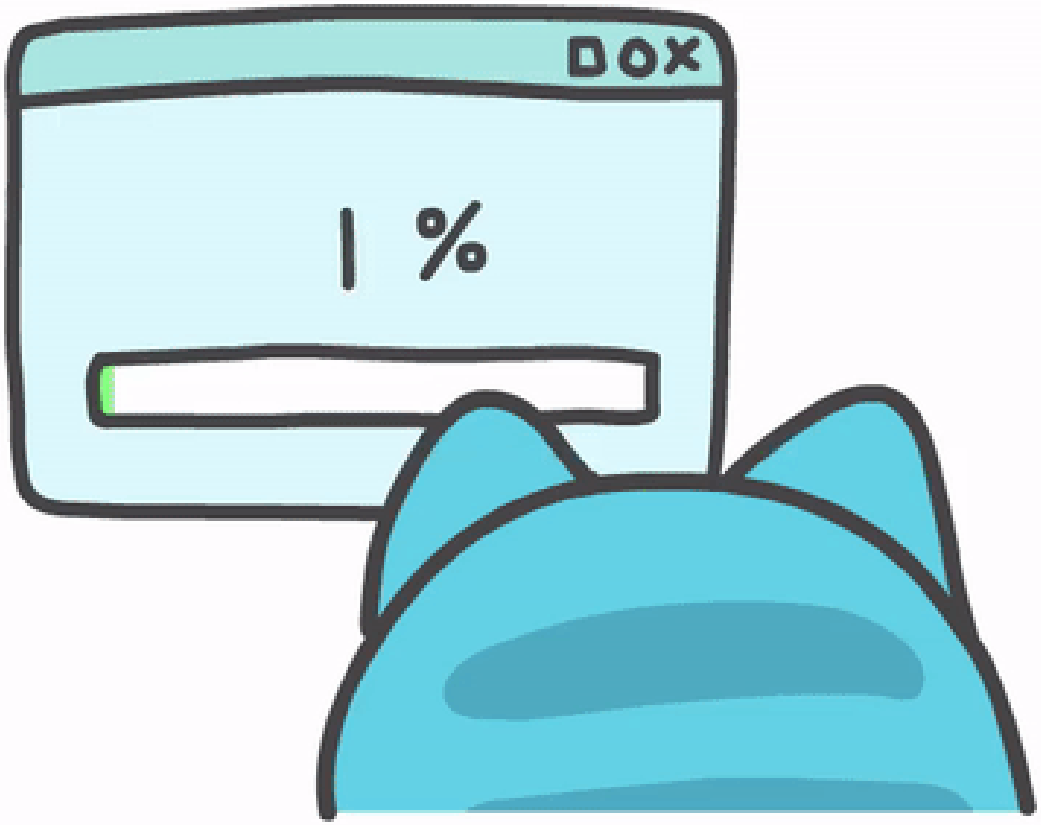
```
# citacao dos pacotes
citation("vegan")
```

```
##
## To cite package 'vegan' in publications use:
##
##   Jari Oksanen, F. Guillaume Blanchet, Michael Friendly, Roeland
##   Kindt, Pierre Legendre, Dan McGlinn, Peter R. Minchin, R. B.
##   O'Hara, Gavin L. Simpson, Peter Solymos, M. Henry H. Stevens,
##   Eduard Szoecs and Helene Wagner (2019). vegan: Community Ecology
##   Package. R package version 2.6-0.
##   https://github.com/vegandevs/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
```

Dúvidas?

Erros!!!

Se seu script rodou sem erros, tem algo errado...


DOX
1 %


I'm fine
BugCat-CAPOO

2.12 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: +
```

2.12 Principais erros

2. Esquecer da vírgula

```
sum(1 2)
```

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

2.12 Principais erros

3. Chamar um objeto errado

```
obj <- 10
OBJ
```

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

2.12 Principais erros

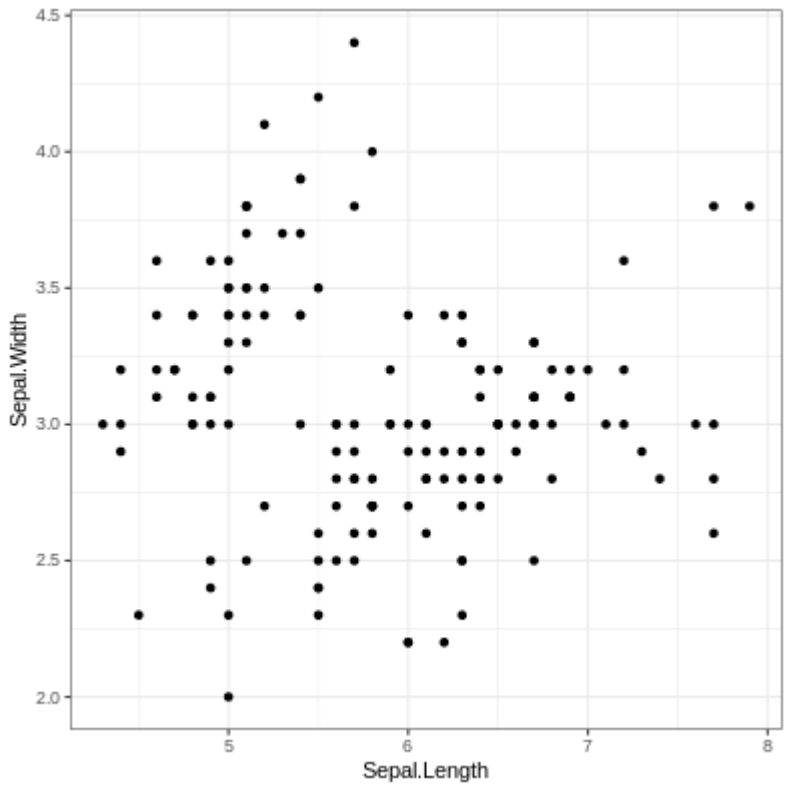
4. Esquecer de carregar um pacote

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

2.12 Principais erros

4. Esquecer de carregar um pacote

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



2.12 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
```

2.12 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"
```

Camel Case vs Snake Case

Dúvidas?

Maurício Vancine

Contatos:

- mauricio.vancine@gmail.com
- mauriciovancine.netlify.com
- @mauriciovancine
- @mauriciovancine
- @mauriciovancine

Slides criados via pacote xaringan e tema Metropolis