A grayscale photograph of a person's hands typing on a laptop keyboard. A large, semi-transparent blue circle is centered over the keyboard area. Inside this circle, the text "Introdução ao software livre R" is written in white. The background shows a wooden desk with a pair of white earbuds, a small potted plant, and a stack of books.

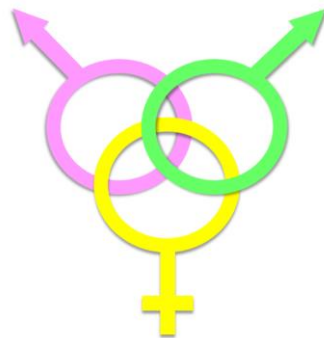
Introdução ao software livre R

Olá!

Amanda Vieira
(Doutoranda EVD - UFABC)

Stefânia Ventura
(Doutoranda ECMVS-UFMG)

LaSexia





Ambiente R



Ross Ihaka




Robert Gentleman



Vantagens x desvantagens

R livre distribuição e gratuito

Minitab  | Loja

Versão completa, licença individual

Analise seus dados e aprimore seus produtos e serviços com o líder dos softwares estatísticos, usado para implementar projetos de aprimoramento de qualidade em todo o mundo.

O preço de desconto por volume de compra é aplicado no carrinho de compras.

Sistema Operacional Windows (32-bit) ▼

Quantidade 1 ▼

Preço por unidade R\$ 7.508,01

COMPRAR AGORA

Vantagens x desvantagens

R livre distribuição e gratuito

Rápida atualização dos pacotes

lme4 News

CHANGES IN VERSION 1.1-26

BUG FIXES

- `predict, model.frame(., fixed.only=TRUE)` work with variable names containing spaces (GH #605)
- `simulate` works when original response variable was logical
- `densityplot` handles partly broken profiles more robustly

NEW FEATURES

- `thpr` method for `densityplot()` (for plotting profiles scaled as densities) gets new arguments

CHANGES IN VERSION 1.1-25 (2020-10-23)

- Set more tests to run only if environment variable `LME4_TEST_LEVEL>1`

CHANGES IN VERSION 1.1-24 (never on CRAN)

USER-VISIBLE CHANGES

- `anova()` now returns a p-value of NA if the df difference between two models is 0 (implying they are equivalent models) (GH #598)
- speedup in `coef()` for large models, by skipping conditional variance calculation (Alexander Bauer)
- `simulate.formula` machinery has changed slightly, for compatibility with the `ergm` package (Pavel Krivitsky)
- informational messages about (non-)convergence improved (GH #599)
- improved error messages for 0 non-NA cases in data (GH #533)

NEW FEATURES

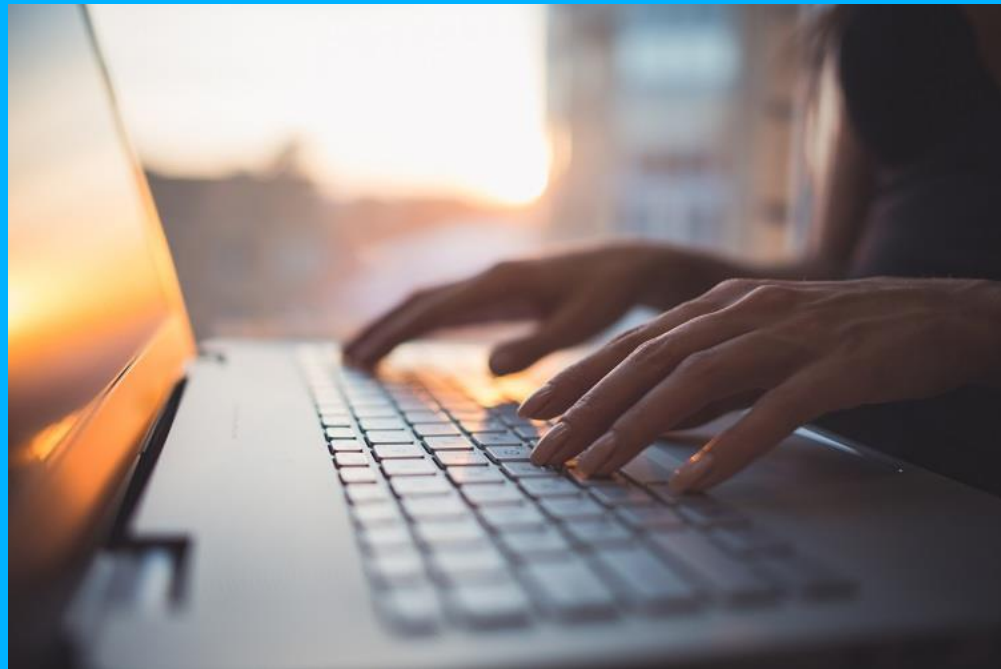
- `getME(., "devfun")` now works for `glmer` objects. Additionally, `profile/confint` for GLMMs no longer depend on objects machinery; results of `glmer` profiling/CIs may not match results from previous versions exactly.

Vantagens x desvantagens

R livre distribuição e gratuito

Rápida atualização dos pacotes

Você programa!



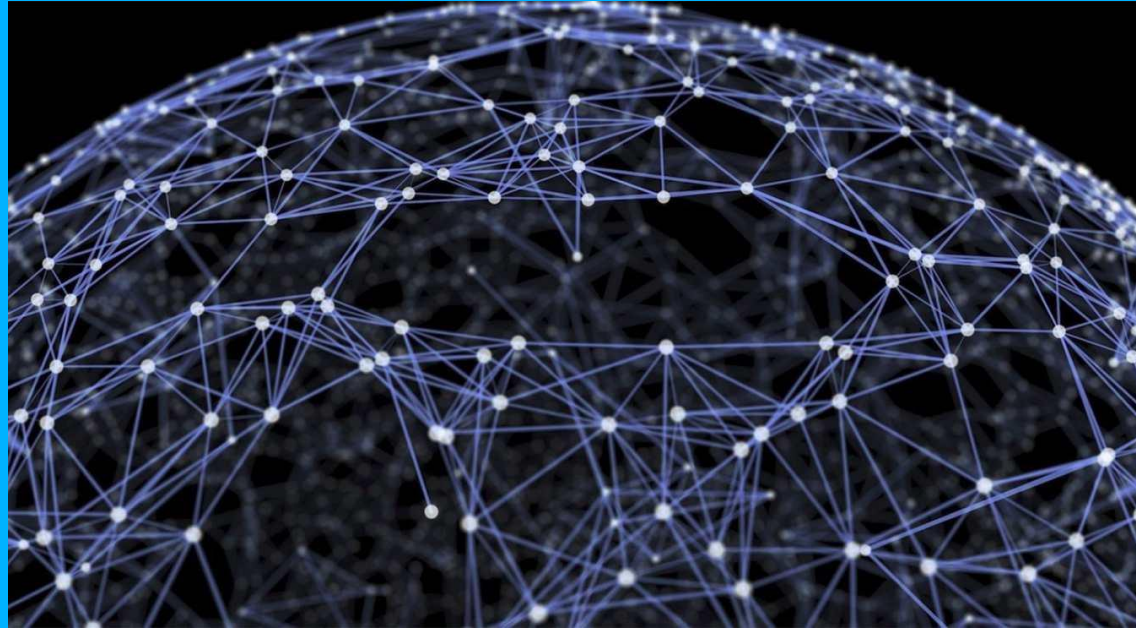
Vantagens x desvantagens

R livre distribuição e gratuito

Rápida atualização dos pacotes

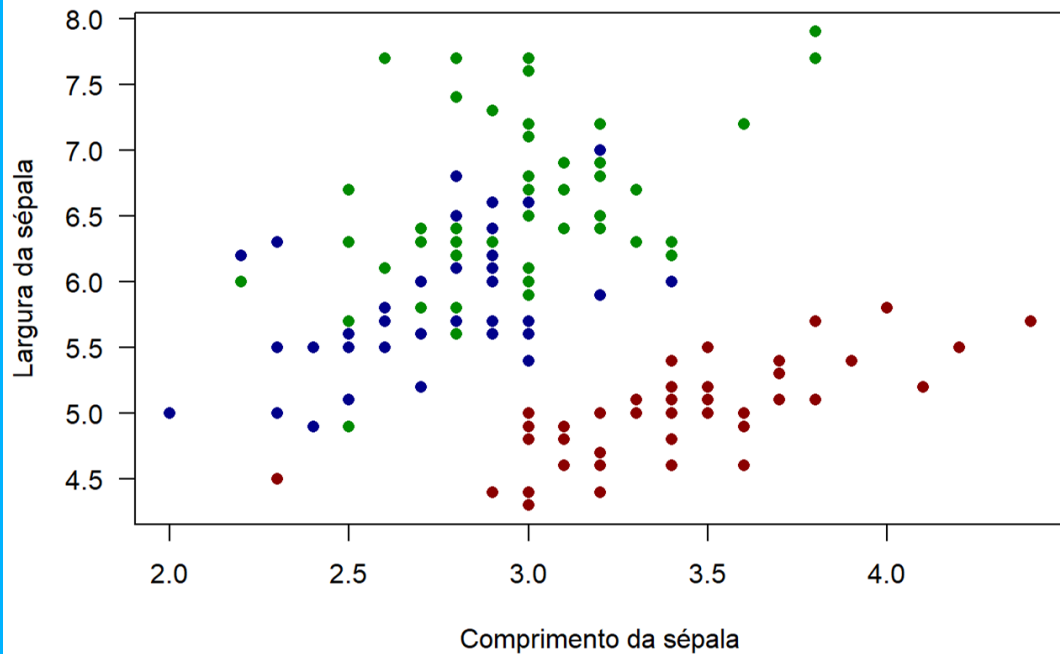
Você programa!

Grande rede de comunicação (fóruns)





2. Possibilidades do R

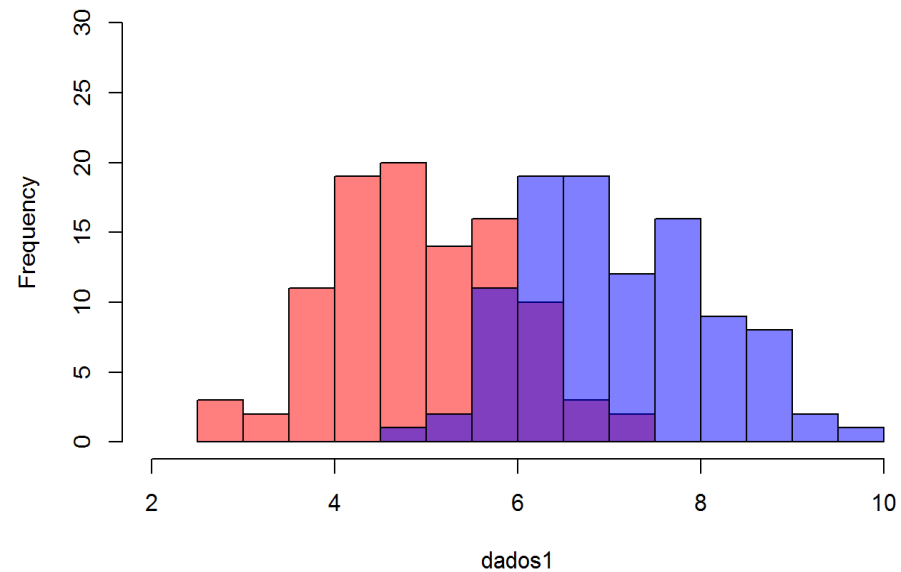
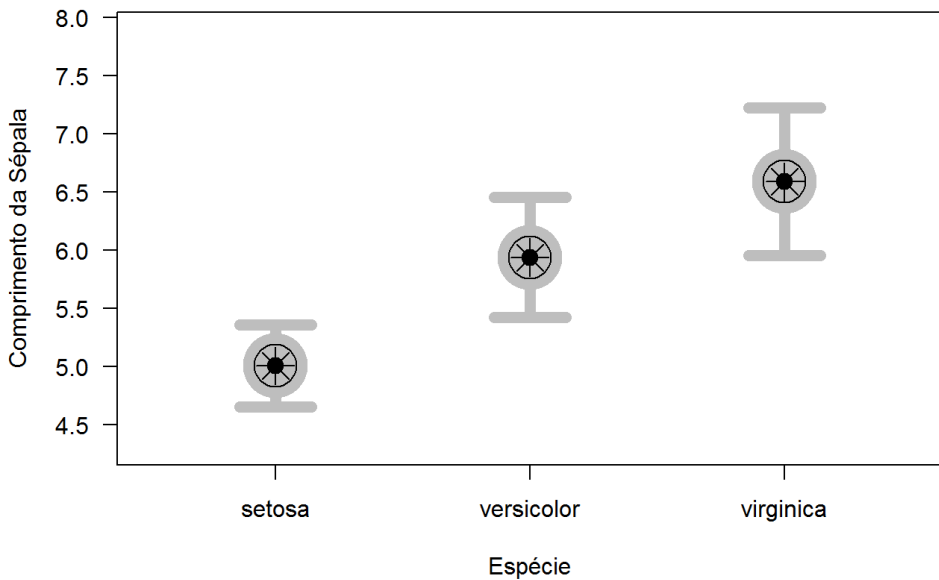


Analysis of Variance Table

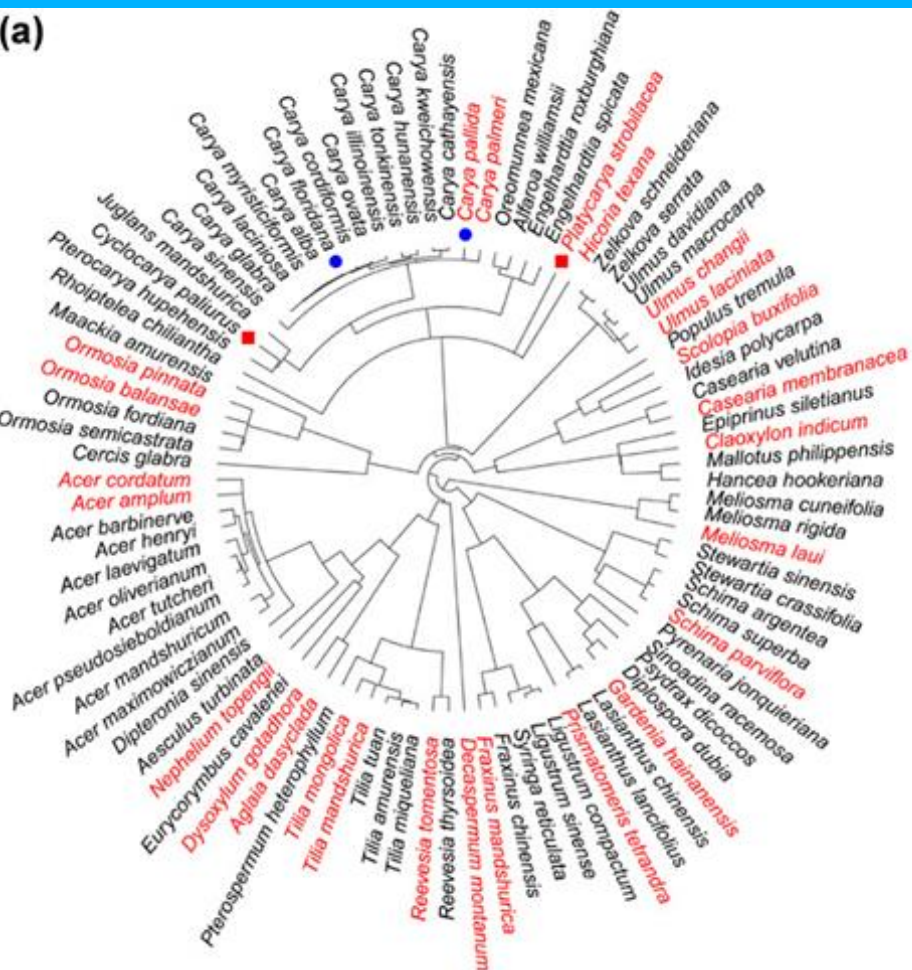
Response: res

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
trat	3	163.750	54.583	7.7976	0.001976 **
Residuals	16	112.000	7.000		

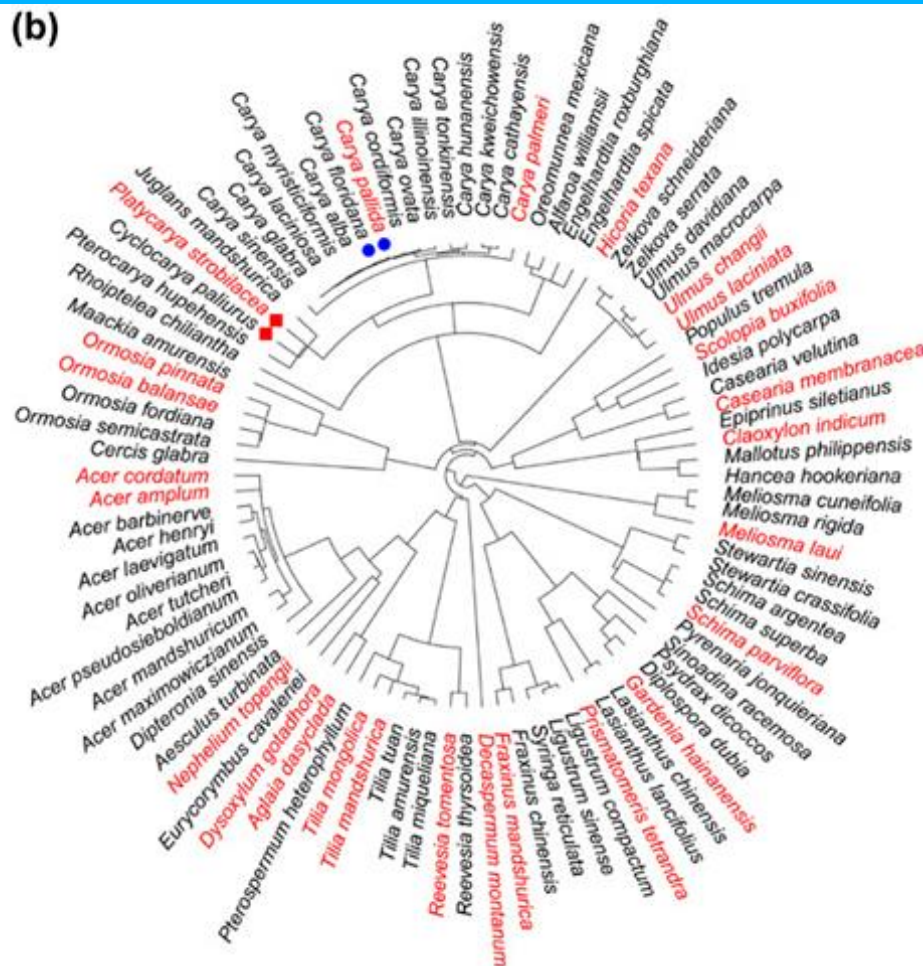
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1



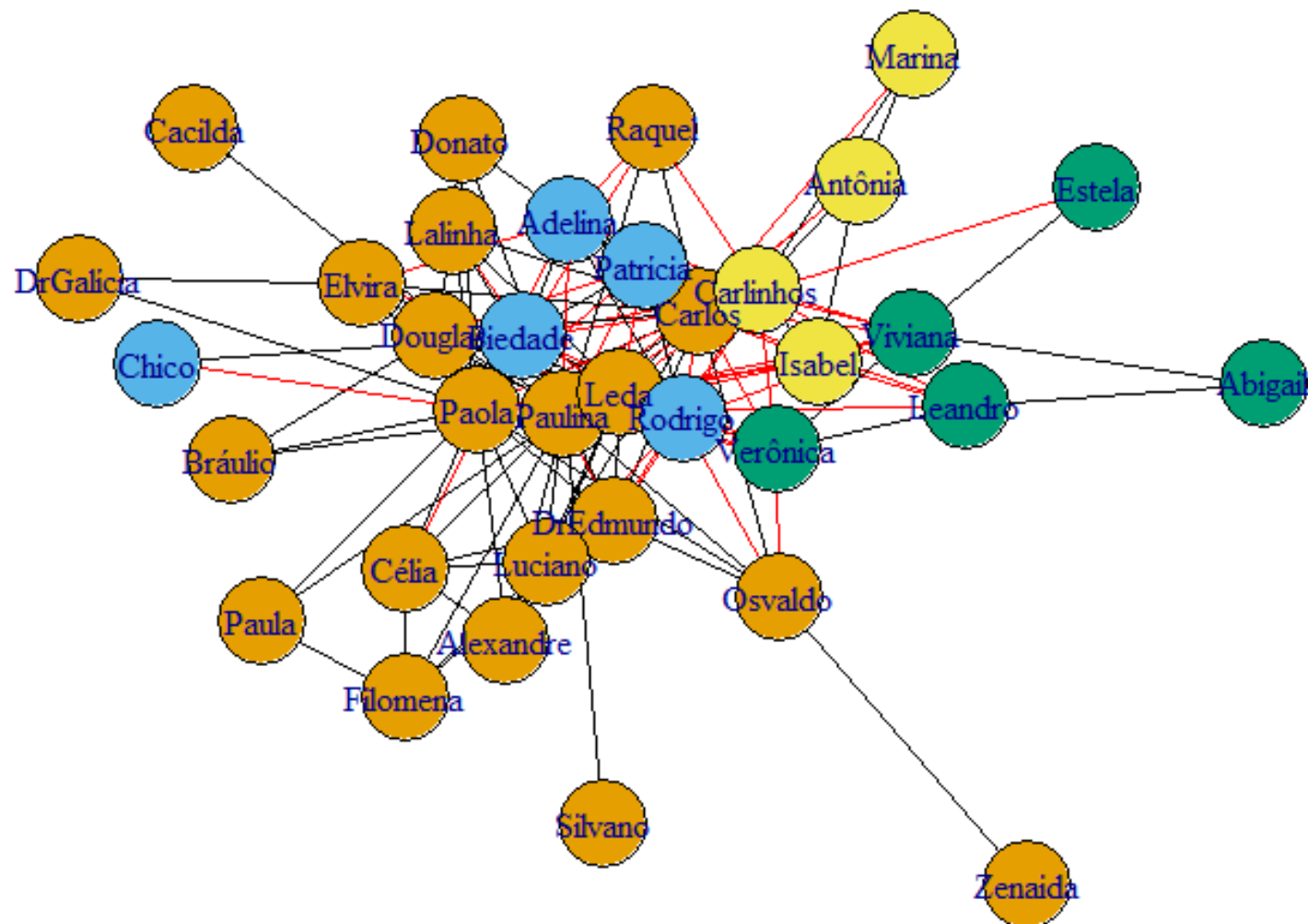
(a)

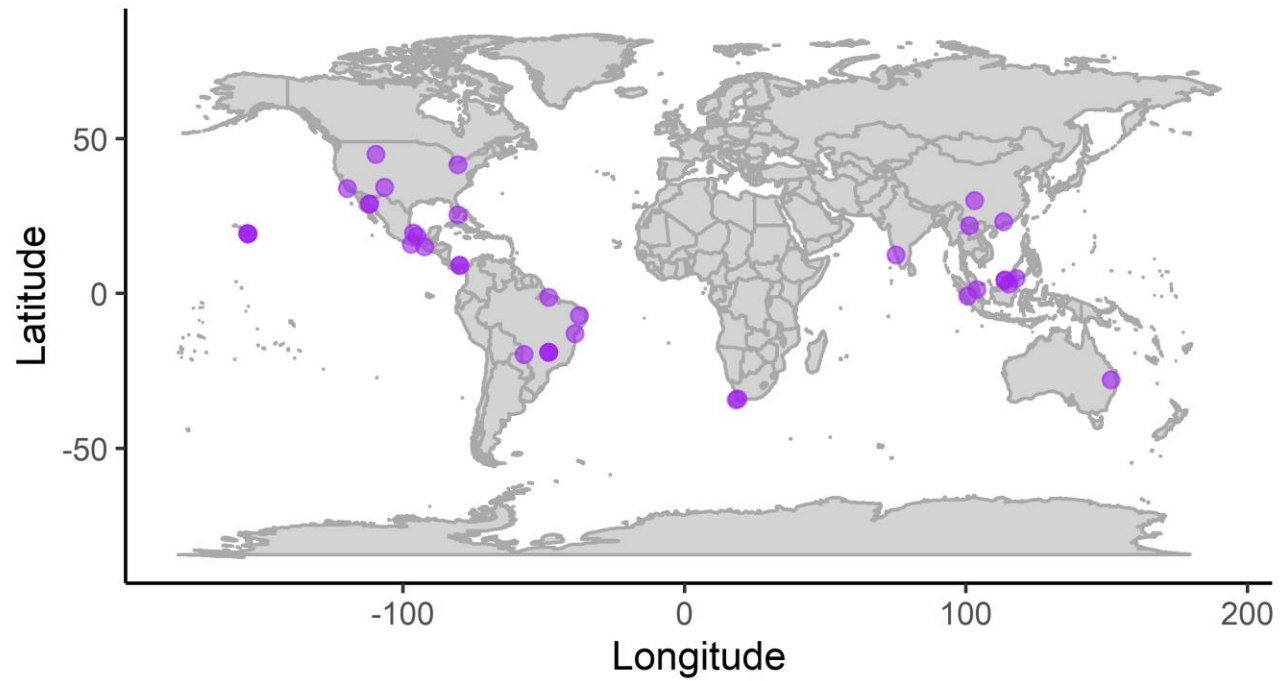


(b)



Rede de personagens de A Usurpadora





always take
look **live** when
end game
your there alone gone
dark and the life
hill hard man **fe** kill
yeah time **ar** begin
run someone wast night
walk

Iron Maiden

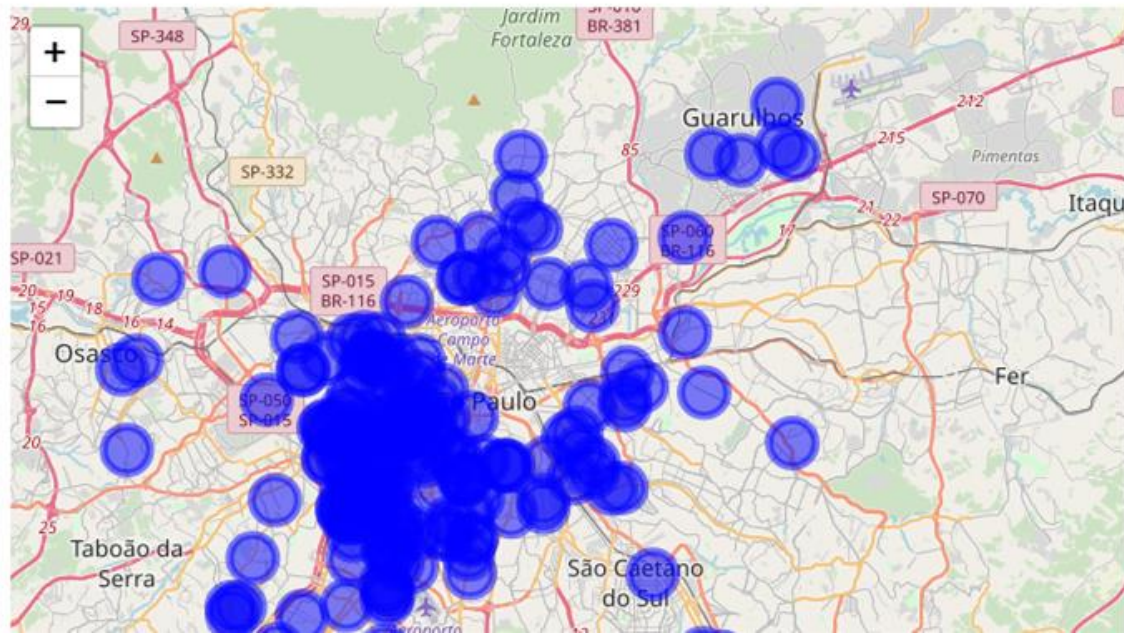
sweat
your dark you
the rictus
play battle beat goin dead
see sword spadewhat
know way
dinsteel shield gonna
forev deaf stale bone
ace back late cut
feel just lose get
move grin make
overkill

Motorhead

> Hamburguerias

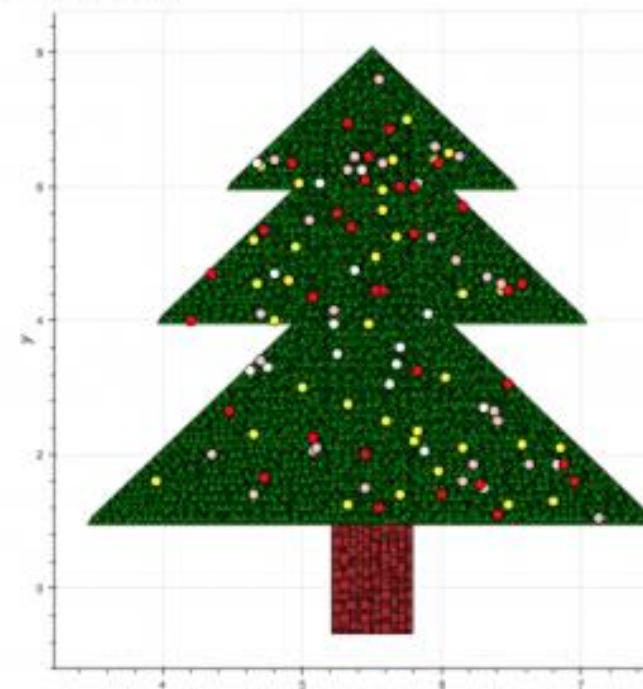
Repare que o único local na região mais central de SP em que você pode ficar a mais de um quilômetro de uma hamburgueria é no meio do parque Ibirapuera.

```
df_places %>%  
  dplyr::filter(place == "hamburguer") %>%  
  leaflet %>%  
  addTiles() %>%  
  addCircles(lng = ~long, lat = ~lat, weight = 5,  
             radius = 1000, color = "blue", fillOpacity = 0.5)
```



Kamil Romaszko & Mateusz Urbański

December 20, 2017



Paweł Pollak

21 grudnia 2017

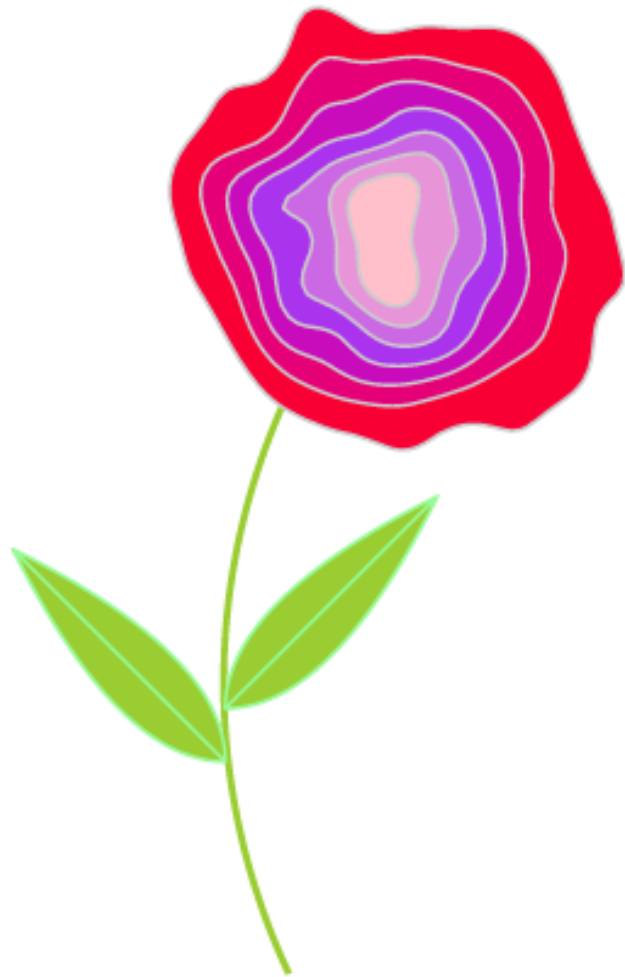
"Wykres" został wykonany przy użyciu pakietu rbokeh.



Krzeszewski Piotr

December 21, 2017





O QUE NÓS QUEREMOS?



```

var evts = 'contextmenu dblclick drag dragenter
dragleave dragstart dragstop drop';
var logHuman = function() {
if (window.wflogHumanRan) { return; }
window.wflogHumanRan = true;
var wfscr = document.createElement('script');
wfscr.type = 'text/javascript';
wfscr.async = true;
wfscr.src = url + '&r=' + Math.random() +
(document.getElementsByTagName('head')
for (var i = 0; i < evts.length; i++)
removeEvent(evts[i], logHuman);
}
};
for (var i = 0; i < evts.length; i++)
addEvent(evts[i], logHuman);

```





Primeiro Contato

Explorando o RStudio

Ambiente: lista dos objetos carregados

Editor: onde escrevemos o código

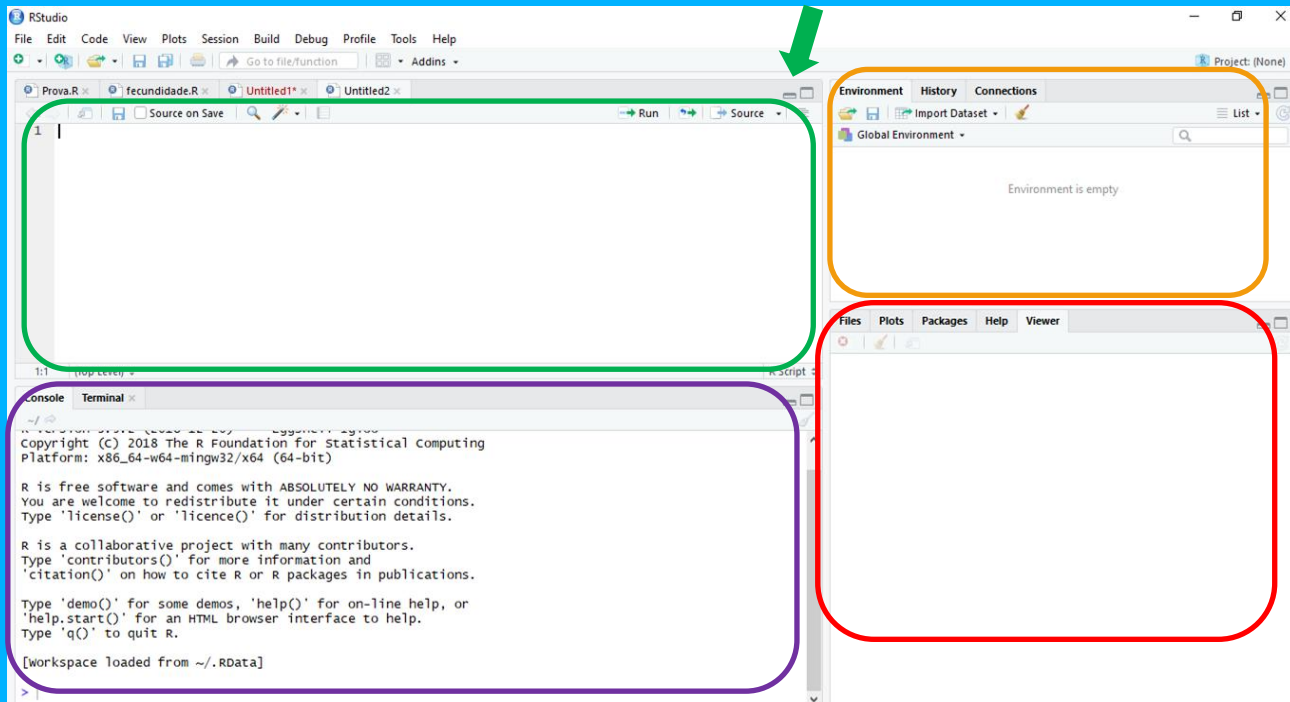
Console: onde executamos o código

Histórico: lista do que foi feito

Files: arquivos no diretório de trabalho

Plots: visualizar gráficos

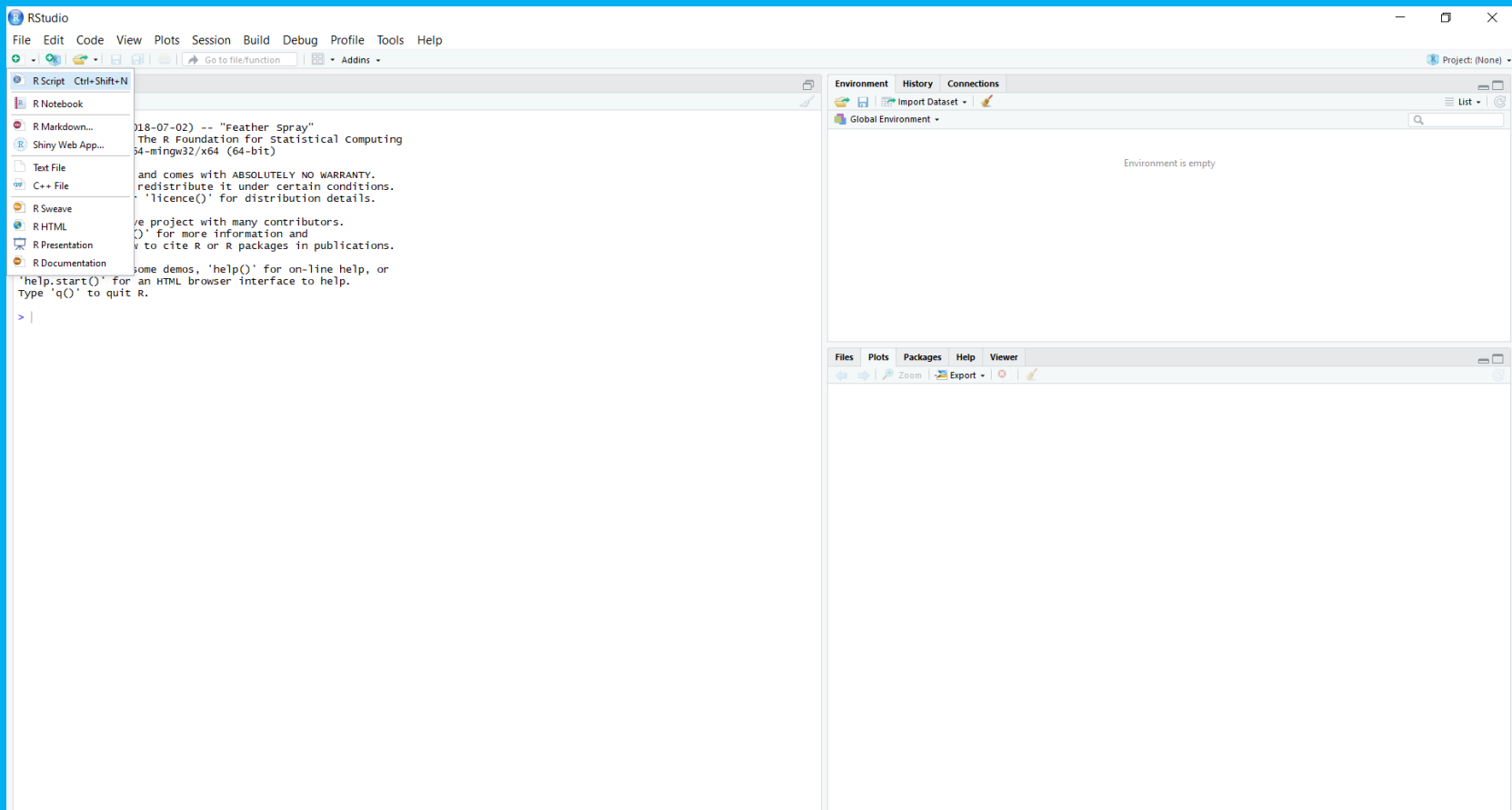
Help: descrição das funções





Usando um script, inserindo comentários e salvando tudo

- São sequências de comandos que podem ser armazenados e usados diversas vezes
- Não acumulam informações de resultados ou mensagens de erros



The image shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for saving, running, and other functions. The main editor area on the left is titled 'Untitled1' and contains a single line of code: `1`. The right-hand side of the window is divided into two panes. The top pane is the 'Environment' pane, which shows 'Global Environment' and states 'Environment is empty'. The bottom pane is the 'Console' pane, which displays the R version 3.5.1 (2018-07-02) -- "Feather Spray" and copyright information. The console also shows the R logo and a prompt `> |`. A red rectangular box is drawn over the console area, containing the text `# comentário`.

```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
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.

> |
```

comentário

RStudio interface showing a script named "Script prova.R" and the console output.

Script prova.R

```
1 ##### Prova #####
2 ##Exercício 1##
3 ativ1=read.table("Exercício 1.txt", header=T, dec=",")
4 ativ1
5 library(lme4)
6 test1=lm(biomassa-competicao-densidade, ativ1)
7 plot(test1)
8 anova(test1)
9 summary(test1)
10 library(sciplot)
11 tiff("Fig.1.tif", w=2400,h=1800, res=300)
12 par(mar=c(4,4,2,2),mgp=c(2.5,1,0))
13 lineplot.ci(competicao, biomassa, bty="l",type="p", xlab="Competição", ylab="Biomassa", xaxt = "n", data=ativ1)
14 axis(1,at=c(1,2), labels = c("Interespecifica", "Intraespecifica"))
15 dev.off()
16 ##Exercício 2##
17 ativ2=read.table("Exercício 2.txt", header=T, dec=",", sep="\t")
18 ativ2
19 test2=chisq.test(ativ2$plantas.embaixo, ativ2$arvore.maior,simulate.p.value = T)
20 test2$expected
21 test2$observed
22 500/15
23 1000/14
24 test2
25 # Como o valor de p deu marginal, tentei aumentar o numero de simulações pra ver se altera o resultado
26 test2=chisq.test(ativ2$plantas.embaixo, ativ2$arvore.maior,simulate.p.value = T,B=50000 )# mas não alterou
27 ##Exercício 3##
28 ativ3=read.table("Exercício 3.txt", header=T, dec=",")
29 ativ3
30 summary(ativ3)
31 <
32 >
```

Console

```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

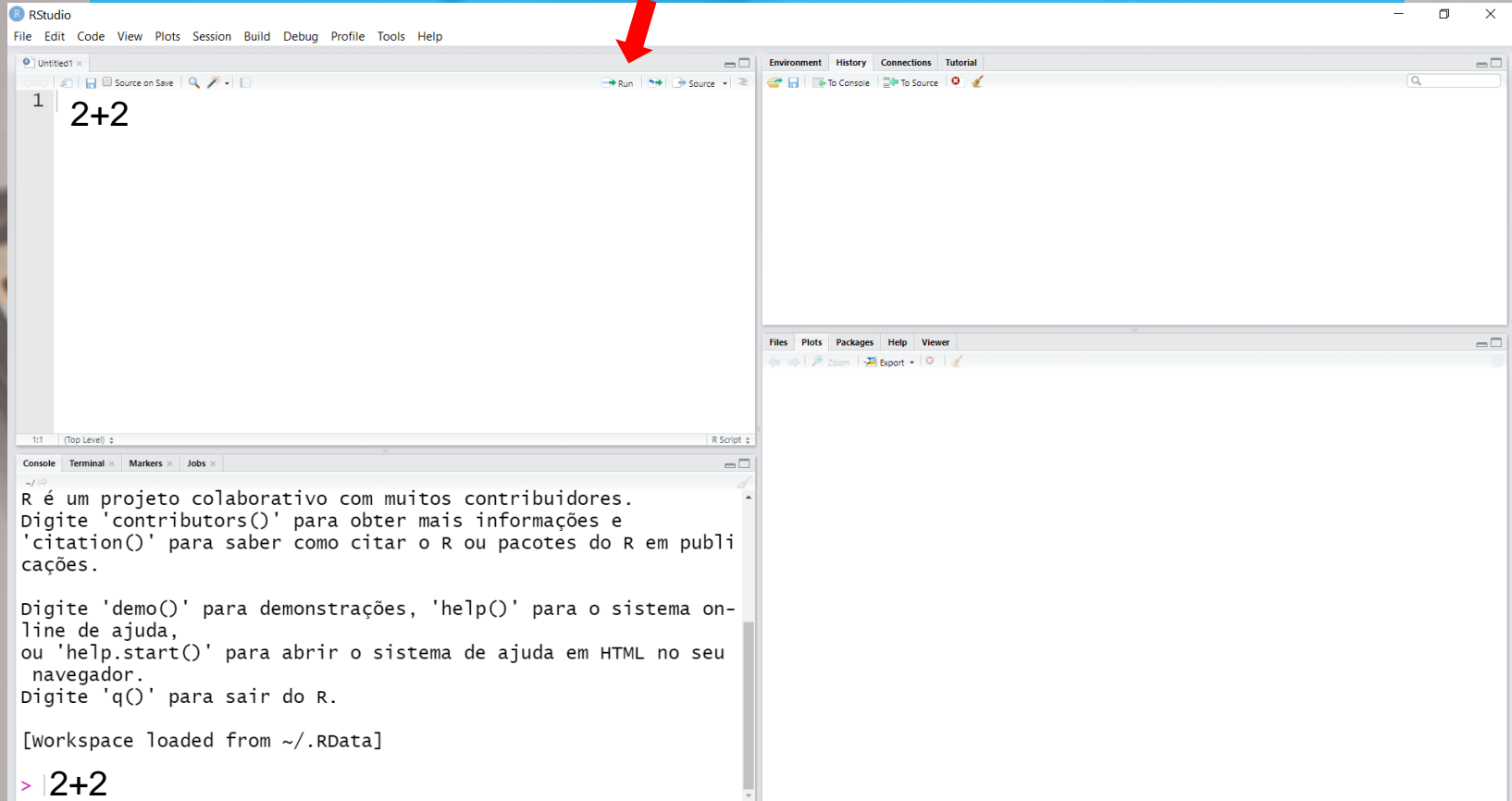
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

> setwd("C:/Users/stefa/OneDrive/Stefânia/Disiplina Ecologia Quantitativa/Prova")
> |
```

R como calculadora



R como calculadora

The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The main editor area shows a script with the following content:

```
1 2+2  
2
```

The console at the bottom left shows the following output:

```
'citation()' para saber como citar o R ou pacotes do R em publi  
cações.  
  
Digite 'demo()' para demonstrações, 'help()' para o sistema on-  
line de ajuda,  
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu  
navegador.  
Digite 'q()' para sair do R.  
  
[workspace loaded from ~/.RData]  
  
> 2+2  
[1] 4  
>
```

The Environment pane on the right side of the window shows the Global Environment, which is currently empty. The bottom right pane shows the Files, Plots, Packages, Help, and Viewer tabs, with the Files tab selected.

Operadores Aritméticos

Arithmetic Operators in R

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus (Remainder from division)
%/%	Integer Division

<https://www.datamentor.io/r-programming/operator/>

Operadores de atribuição

Assignment Operators in R

Operator	Description
<code><-</code> , <code><<-</code> , <code>=</code>	Leftwards assignment
<code>-></code> , <code>->></code>	Rightwards assignment

Operadores Relacionais

Relational Operators in R

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Operadores lógicos

Logical Operators in R

Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
	Element-wise logical OR
	Logical OR



Criando objetos

- Programação orientada a objetos
- Objeto guardará “dentro” dele um conjunto de informações (valores, nomes, etc.)

- ✓ Para atribuir valores a objetos, basta usar os operadores de atribuição

```
> objeto1=3*8  
> objeto2<-2+3  
> 5^7->objeto3
```

- ✓ Para visualizar o valor armazenado no objeto:

```
> objeto1  
[1] 24
```

```
> (objeto2<-2+3)  
[1] 5  
> (5^7->objeto3)  
[1] 78125
```

- ✓ Nomeando os objetos no R:
 - letras
 - números
 - ponto
 - underline
- ✓ Evite nomes muito longos
- ✓ Não use espaço
- ✓ Palavras reservadas

Variable Name	Valid?
variable_name1	yes
variable_name%	no (contains a special character)
1variable_name	no (starts with a number)
.variable_name	yes
.1variable_name	no (dot is followed by a number)
_1variable_name	no (starts with an underscore)

Classe do objeto/ Tipo de variáveis

Função:
class()

Texto (character)

- Características puramente individuais que não podem ser utilizadas para categorizar os indivíduos. Geralmente aparecem nos bancos de dados apenas para ajudar em análises qualitativas e não estatísticas. Exemplo: o nome dos candidatos num banco de dados de resultados eleitorais.

Numéricas (numeric)

- Números inteiros ou reais, como idade, renda, número de filhos.

Datas (Date)

- São um tipo especial de variável numérica.

Catégoricas (factor)

- Variáveis qualitativas, ou seja, características dos indivíduos para as quais não é possível atribuir um valor numérico
- sexo
- religião
- estado civil

Catégoricas ordenáveis (factor)

- Tipo de variável catégorica cujas categorias podem ser hierarquizáveis, como grau de escolaridade.

Booleanas (logical)

- Variáveis cujos valores podem ser VERDADEIRO ou FALSO; no R, TRUE ou FALSE.

Vamos brincar?



Mão na massa 1


São 12 pessoas para escutar e cada depoimento demora, em média, 1h30. Um único delegado e escrivão devem colher todos os depoimentos e ambos trabalham 6 horas por dia.

Crie objetos para guardar as quantidades mais importantes nesse problema. Quantos objetos são? Quais são?

Calcule o tempo necessário para interrogar todas as testemunhas

- A) em horas
- B) em minutos
- C) em dias de trabalho da dupla delegado/escrivão





Diferentes estruturas para armazenar o dados

- Vetores
- Matriz
- Listas
- Data frames

Vetores

- ✓ Sequência de dados do mesmo tipo de classe
- ✓ `c()`- função para combinar argumentos

```
> ?c()
```

The screenshot displays the RStudio application window. At the top, the title bar reads "Curso basico - RStudio". Below it is a menu bar with options: File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. A red arrow points to the "Tools" menu.

The main workspace is divided into three panes:

- Script Editor (Top Left):** Contains two lines of code:

```
1 ?c()  
2 |
```
- Environment Pane (Top Right):** Shows the "Global Environment" with a table header: Name, Type, Length, Size, Value. The content area states "Environment is empty".
- Help Pane (Bottom Right):** Displays the documentation for the `c()` function, titled "Combine Values into a Vector or List". It includes a description, usage, arguments, and details. A red arrow points to the top right corner of this pane.

At the bottom, the **Console** pane shows the command prompt with the text `> ?c()` and a cursor on the next line.

✓ Criando os vetores no R:

```
> (vetor1=c(10,20,30,40,50))  
[1] 10 20 30 40 50  
> (vetor2<- c(100,200,300,400))  
[1] 100 200 300 400
```

✓ Combinando vetores

```
> (vetor3=c(vetor1,vetor2))  
[1] 10 20 30 40 50 100 200 300 400
```

✓ Coerção de vetores

Como os vetores só podem pertencer a uma classe de variável, sempre que você misturar diferentes classes em um vetor, haverá coerção de uma das variáveis!

```
2 numeros=c(1,2,3,"a")
3 logico=c(1,2,3,TRUE)
4 letras=c("a","b","c",TRUE)
5 class(letras)
6
```

letras	character	4	304 B	chr [1:4]	"a" "b" "c" "T..."
logico	numeric	4	80 B	num [1:4]	1 2 3 1
numeros	character	4	304 B	chr [1:4]	"1" "2" "3" "a"

```
> class(numeros)
[1] "character"
> class(logico)
[1] "numeric"
> class(letras)
[1] "character"
```


✓ Operações com vetores:

```
> (operacao=vetor1*5)
[1] 50 100 150 200 250
```

✓ Operações com dois vetores

```
> somav1v2=(vetor1+vetor2)
Warning message:
In vetor1 + vetor2 :
  comprimento do objeto maior não é múltiplo do comprimento
do objeto menor
```

```
> (vetor1=c(10,20,30,40,50))
[1] 10 20 30 40 50
> (vetor2<- c(100,200,300,400))
[1] 100 200 300 400
```



Reciclagem de valores

✓ Operações com vetores:

```
> (operacao=vetor1*5)  
[1] 50 100 150 200 250
```

✓ Operações com dois vetores

```
> soma v1v2  
[1] 110 220 330 440 150
```

- ✓ Criando os vetores longos:
 - seq() cria uma sequência

```
> (vetor4=seq(from=20,to=200, by=15))  
[1] 20 35 50 65 80 95 110 125 140 155 170 185 200  
> (vetor5=seq(20,200, by=15))  
[1] 20 35 50 65 80 95 110 125 140 155 170 185 200  
>
```

- rep() repete valores passados

```
> (vetor6=rep(x=c(2,4,6,8), each=10))  
[1] 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6  
[28] 6 6 6 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
```

- ✓ Fazendo sorteio no R:
 - sample()

```
> nomes=c("João", "Amanda", "Douglas", "Stefania")  
> sample(nomes, 1, replace=F)  
[1] "Stefania"
```

- ✓ Acessando elementos de um vetor:
 - operadores lógicos

```
> peso=c(46,52,56,60,73,78,80,82,100)
> peso>70
[1] FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE
```

- [] selecionando elementos

```
> peso[peso>70]
[1] 73 78 80 82 100
```

```
> peso[peso>100]
numeric(0)
```

Funções aplicadas a vetores

Função	Descrição
<code>sum()</code>	Retorna a soma
<code>mean()</code>	Retorna a média
<code>sd()</code>	Retorna o desvio padrão
<code>median()</code>	Retorna a mediana
<code>var()</code>	Retorna a variância
<code>cor()</code>	Retorna a correlação entre dois vetores
<code>min()</code>	Retorna o mínimo
<code>max()</code>	Retorna o máximo
<code>range()</code>	Retorna o mínimo e o máximo
<code>summary()</code>	Retorna um sumário dos dados
<code>quantile()</code>	Retorna os quantis do conjunto numérico

Mão na massa 2

-Crie dois vetores:

v1 com valores 2 3 5 10 9 15 51 5

v2 com valores 16 17.7 20 23.1 27 16.2 18 22

-Obtenha a soma dos valores de cada vetor

-Crie um novo vetor que seja a união de v1 e v2

- Multiplique por 100 o vetor anterior resguardando os valores em outro vetor



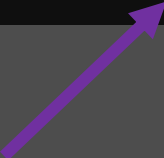
Matriz

Função `matrix()`


Bidimensional e uma única classe

```
?matrix
```

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,  
        dimnames = NULL)
```




Nome nas
linhas e
colunas



Número de
linhas



Número de
colunas



Preencher
por linha?

Matriz

Função `matrix()`

Bidimensional e uma única classe

```
matrix(1:6, ncol = 2, nrow = 3)  
|
```

	[,1]	[,2]
[1,]	1	4
[2,]	2	5
[3,]	3	6

Matriz

Função `matrix()`

Bidimensional e uma única classe

```
matrix(c("a", 1, 2, TRUE), ncol = 2)  
|
```

O que aconteceu com o que está dentro da matriz?

```
> matrix(c("a", 1, 2, TRUE), ncol = 2)  
      [,1] [,2]  
[1,] "a"  "2"  
[2,] "1"  "TRUE"
```

Matriz

Função `matrix()`

Bidimensional e uma única classe

E se eu especificar apenas o número de colunas?

```
matrix(1:6, ncol = 2)
```

```
> matrix(1:6, ncol = 2)
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Matriz

Função `matrix()`

Bidimensional e uma única classe

Usando o `byrow`

```
> matrix(1:6, ncol = 2, byrow = TRUE)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

```
> matrix(1:6, ncol = 2)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Matriz

Acessando elementos da matriz

```
> matriz
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

LINHA

```
> matriz[1,]
[1] 1 2
```

COLUNA

```
> matriz[,1]
[1] 1 3 5
```

ELEMENTO

```
> matriz[1,1]
[1] 1
```

Matriz

Operações com matrizes

SOMA

```
> matriz
      [,1] [,2]
[1,]     1     2
[2,]     3     4
[3,]     5     6
```

```
> sum(matriz)
[1] 21
```

```
> sum(1:6)
[1] 21
```

PRODUTO

```
> prod(matriz)
[1] 720
```

```
> prod(1:6)
[1] 720
```


Matriz

Operações com matrizes

SOMA

```
> matriz
      [,1] [,2]
[1,]     1     2
[2,]     3     4
[3,]     5     6
```

```
> matriz + 5
      [,1] [,2]
[1,]     6     7
[2,]     8     9
[3,]    10    11
```

PRODUTO

```
> matriz * 5
      [,1] [,2]
[1,]     5    10
[2,]    15    20
[3,]    25    30
```

Matriz

Operações com matrizes: CUIDADO COM A MULTIPLICAÇÃO DE MATRIZES!

PRODUTO

```
> matriz
      [,1] [,2]
[1,]     1     2
[2,]     3     4
[3,]     5     6
```

```
> matriz2
      [,1] [,2]
[1,]     8    11
[2,]     9    12
[3,]    10    13
```

```
> matriz * matriz2
      [,1] [,2]
[1,]     8    22
[2,]    27    48
[3,]    50    78
> |
```

Matriz

Operações com matrizes: CUIDADO COM A MULTIPLICAÇÃO DE MATRIZES!

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a \cdot e + b \cdot g & a \cdot f + b \cdot h \\ c \cdot e + d \cdot g & c \cdot f + d \cdot h \end{pmatrix}$$

PRODUTO MATRICIAL

```
> matriz2
      [,1] [,2]
[1,]     1     3
[2,]     2     4
```

```
> matriz3
      [,1] [,2]
[1,]     5     7
[2,]     6     8
```

```
> matriz2 %*% matriz3
      [,1] [,2]
[1,]    23    31
[2,]    34    46
```

Matriz

Funções aplicadas a matrizes

Função	Descrição
<code>t()</code>	Retorna a matriz transposta
<code>diag(k)</code>	Cria uma matriz identidade $k \times k$
<code>det()</code>	Calcula o determinante da matriz
<code>diag()</code>	Retorna os elementos da diagonal principal
<code>dim()</code>	Retorna a dimensão da matriz
<code>ncol</code>	Retorna o número de colunas da matriz
<code>nrow()</code>	Retorna o número de linhas da matriz
<code>rowSums()</code>	Retorna a soma das linhas da matriz
<code>rowMeans()</code>	Retorna a média das linhas da matriz
<code>colSums()</code>	Retorna a soma das colunas da matriz
<code>colMeans()</code>	Retorna a média das colunas da matriz

Matriz

Funções aplicadas a matrizes

SOMA DAS COLUNAS E LINHAS

```
> matriz
      [,1] [,2]
[1,]     1     2
[2,]     3     4
[3,]     5     6
```

```
> colSums(matriz)
[1]  9 12
> rowSums(matriz)
[1]  3  7 11
>
```

MÉDIA DAS COLUNAS E LINHAS

```
> colMeans(matriz)
[1] 3 4
> rowMeans(matriz)
[1] 1.5 3.5 5.5
```

Matriz

Concatenando matrizes

POR COLUNA

`cbind()`

```
cafe <- c("cha", "pao")
almoco <- c("suco", "salada")
janta <- c("agua", "macarrao")

refeicao <- cbind(cafe, almoco, janta)
refeicao
```

```
> refeicao
      cafe almoco  janta
[1,] "cha" "suco"  "agua"
[2,] "pao" "salada" "macarrao"
```

POR LINHA

`rbind()`

```
refeicao2 <- rbind(cafe, almoco, janta)
refeicao2
```

```
> refeicao2
      [,1] [,2]
cafe    "cha" "pao"
almoco  "suco" "salada"
janta   "agua" "macarrao"
```

Mão na massa

- 1) Crie uma matriz 1 a 6 com 3 colunas. Salve num objeto chamado `matriz`
- 2) Multiplique a matriz acima por 5 e salve em `matriz2`
- 3) Multiplique a matriz acima por uma `matriz3` de que vai de 2 a 12 de 2 em 2 (3 col)
- 4) Pegue a matriz e adicione uma linha com os seguintes elementos 0,1,2 no começo e 8, 10, 12 no final da matriz
- 5) Faça as seguintes operações e diga o que essas funções retornam
 - `matriz > 4`
 - `matriz == 5`

Listas

Função list()

Permite armazenar vetores de classes distintas

```
animais <- c("gato", "cachorro", "peixe")
quantidade <- c(1, 3, 7, 9, 10)
logico <- c(TRUE, FALSE)
lista <- list(animais, quantidade, logico)
```

```
> lista
[[1]]
[1] "gato"      "cachorro" "peixe"

[[2]]
[1] 1 3 7 9 10

[[3]]
[1] TRUE FALSE
```


Listas

Acessando elementos de uma lista

```
> lista
[[1]]
[1] "gato"      "cachorro" "peixe"

[[2]]
[1] 1 3 7 9 10

[[3]]
[1] TRUE FALSE
```

```
> lista[1]
[[1]]
[1] "gato"      "cachorro" "peixe"

> lista[2]
[[1]]
[1] 1 3 7 9 10

> lista[3]
[[1]]
[1] TRUE FALSE
```

Listas

Acessando elementos específicos de uma lista

```
> lista
[[1]]
[1] "gato"      "cachorro" "peixe"

[[2]]
[1] 1 3 7 9 10

[[3]]
[1] TRUE FALSE
```

```
> lista[[1]][3]
[1] "peixe"
> lista[[2]][1]
[1] 1
> lista[[3]][2]
[1] FALSE
```

Listas

Adicionando e modificando elementos numa lista

```
lista[[1]][4] <- "papgaio"
```

```
> lista[[1]]  
[1] "gato"      "cachorro" "peixe"     "papgaio"
```

```
lista[[1]][4] <- "papagaio"  
lista[[1]]
```

Listas

Operações com listas

```
> sum(lista)
Error in sum(lista) : 'type' inválido (list) do argumento
```

```
> lista
[[1]]
[1] "gato"      "cachorro" "peixe"     "papagaio"

[[2]]
[1] 1 3 7 9 10

[[3]]
[1] TRUE FALSE

> sum(lista[[2]])
[1] 30
```

Data frames

Estruturas bidimensionais e parecidas com planilhas

Coluna = variável

Linha = observação

“Conjunto de vetores de mesmo tamanho”

São flexíveis, porque cada coluna pode ser uma classe diferente

O R disponibiliza alguns dataframes no datasets

Data frames

```
mtcars
```

```
mtcars {datasets}
```

R Documentation

Motor Trend Car Road Tests

Description

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

Usage

```
mtcars
```

Format

A data frame with 32 observations on 11 (numeric) variables.

```
[, 1] mpg  Miles/(US) gallon  
[, 2] cyl   Number of cylinders  
[, 3] disp  Displacement (cu.in.)  
[, 4] hp    Gross horsepower  
[, 5] drat  Rear axle ratio  
[, 6] wt    Weight (1000 lbs)  
[, 7] qsec  1/4 mile time  
[, 8] vs    Engine (0 = V-shaped, 1 = straight)  
[, 9] am    Transmission (0 = automatic, 1 = manual)  
[,10] gear  Number of forward gears  
[,11] carb  Number of carburetors
```

Data frames

Acessando colunas

```
#Acessando 1ª coluna  
mtcars[,1]  
mtcars$mpg  
mtcars[['mpg']]  
mtcars[, 'mpg']  
mtcars['mpg']
```

```
> #Acessando 1ª coluna  
> mtcars[,1]  
[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4  
[13] 17.3 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3  
[25] 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4  
> mtcars$mpg  
[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4  
[13] 17.3 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3  
[25] 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4  
> mtcars[['mpg']]  
[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4  
[13] 17.3 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3  
[25] 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4  
> mtcars[, 'mpg']  
[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4  
[13] 17.3 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3  
[25] 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4  
> mtcars['mpg']  
  
                mpg  
Mazda RX4      21.0  
Mazda RX4 Wag  21.0  
Datsun 710     22.8  
Hornet 4 Drive 21.4  
Hornet Sportabout 18.7  
Valiant        18.1  
Duster 360     14.3  
Merc 240D      24.4
```

Data frames

Acessando linhas

```
#Acessando primeira linha  
mtcars[1,]  
|
```

```
> mtcars[1,]  
      mpg  cyl  disp  hp  drat    wt   qsec  vs  am  gear  carb  
Mazda RX4   21   6  160 110   3.9  2.62 16.46  0   1     4     4
```


Data frames

Acessando várias linhas e várias colunas

```
mtcars[1:5, 3:6]
```

```
> mtcars[1:5, 3:6]
```

	disp	hp	drat	wt
Mazda RX4	160	110	3.90	2.620
Mazda RX4 Wag	160	110	3.90	2.875
Datsun 710	108	93	3.85	2.320
Hornet 4 Drive	258	110	3.08	3.215
Hornet Sportabout	360	175	3.15	3.440

Data frames

Checando um dataframe: str()

```
> str(mtcars)
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num   6  6  4  6  8  6  8  4  4  6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110  93 110 175 105 245  62  95 123 ...
 $ drat: num   3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num   2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num   0  0  1  1  0  1  0  1  1  1 ...
 $ am  : num   1  1  1  0  0  0  0  0  0  0 ...
 $ gear: num   4  4  4  3  3  3  3  4  4  4 ...
 $ carb: num   4  4  1  1  2  1  4  2  2  4 ...
```

Data frames

Checando um dataframe: summary()

```
> summary(mtcars)
```

mpg	cyl	disp	hp	drat	wt
Min. :10.40	Min. :4.000	Min. : 71.1	Min. : 52.0	Min. :2.760	Min. :1.513
1st Qu.:15.43	1st Qu.:4.000	1st Qu.:120.8	1st Qu.: 96.5	1st Qu.:3.080	1st Qu.:2.581
Median :19.20	Median :6.000	Median :196.3	Median :123.0	Median :3.695	Median :3.325
Mean :20.09	Mean :6.188	Mean :230.7	Mean :146.7	Mean :3.597	Mean :3.217
3rd Qu.:22.80	3rd Qu.:8.000	3rd Qu.:326.0	3rd Qu.:180.0	3rd Qu.:3.920	3rd Qu.:3.610
Max. :33.90	Max. :8.000	Max. :472.0	Max. :335.0	Max. :4.930	Max. :5.424

qsec	vs	am	gear	carb
Min. :14.50	Min. :0.0000	Min. :0.0000	Min. :3.000	Min. :1.000
1st Qu.:16.89	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:3.000	1st Qu.:2.000
Median :17.71	Median :0.0000	Median :0.0000	Median :4.000	Median :2.000
Mean :17.85	Mean :0.4375	Mean :0.4062	Mean :3.688	Mean :2.812
3rd Qu.:18.90	3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.:4.000	3rd Qu.:4.000
Max. :22.90	Max. :1.0000	Max. :1.0000	Max. :5.000	Max. :8.000

```
> |
```

Data frames

Checando um dataframe: head()

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Data frames

Checando um dataframe: head()

```
> head(mtcars, 10)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

Data frames

Checando um dataframe: tail()

```
> tail(mtcars, 3)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Ferrari Dino	19.7	6	145	175	3.62	2.77	15.5	0	1	5	6
Maserati Bora	15.0	8	301	335	3.54	3.57	14.6	0	1	5	8
Volvo 142E	21.4	4	121	109	4.11	2.78	18.6	1	1	4	2

Data frames

Funções aplicadas a dataframes

→ Exemplo iris

iris {datasets}

R Documentation

Edgar Anderson's Iris Data

Description

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

Usage

```
iris  
iris3
```

Format

`iris` is a data frame with 150 cases (rows) and 5 variables (columns) named `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, and `Species`.

`iris3` gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the measurements with names `Sepal L.`, `Sepal W.`, `Petal L.`, and `Petal W.`, and the third the species.

Source

Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7, Part II, 179–188.

Data frames

Funções aplicadas a dataframes

→ Exemplo iris

```
> head(iris,3)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1           3.5           1.4           0.2   setosa
2          4.9           3.0           1.4           0.2   setosa
3          4.7           3.2           1.3           0.2   setosa

> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1
 1 ...

> summary(iris)
   Sepal.Length      Sepal.Width      Petal.Length      Petal.Width
Min.   :4.300    Min.   :2.000    Min.   :1.000    Min.   :0.100
1st Qu.:5.100    1st Qu.:2.800    1st Qu.:1.600    1st Qu.:0.300
Median :5.800    Median :3.000    Median :4.350    Median :1.300
Mean   :5.843    Mean   :3.057    Mean   :3.758    Mean   :1.199
3rd Qu.:6.400    3rd Qu.:3.300    3rd Qu.:5.100    3rd Qu.:1.800
Max.   :7.900    Max.   :4.400    Max.   :6.900    Max.   :2.500

   Species
setosa   :50
versicolor:50
virginica :50
```


(O que é um fator ?

Fatores são dados categóricos que podem ser exibidos em níveis.

No R, eles são armazenados “internamente” como números inteiros)

```
> levels(iris$Species)
[1] "setosa"      "versicolor" "virginica"
```

Data frames

Funções aplicadas a dataframes

```
> sum(iris$Sepal.Length)
[1] 876.5
> mean(iris$Sepal.Length)
[1] 5.843333
```

Data frames

E se eu quiser saber a média do comprimento da sépala só da espécie versicolor?

```
> mean(iris$Sepal.Length[iris$Species == "versicolor"])  
[1] 5.936
```

Data frames

E se eu quiser
saber a
média do
comprimento
da sépala
das 3
espécies...
Preciso
escrever uma
por uma?

`tapply {base}`

R Documentation

Apply a Function Over a Ragged Array

Description

Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

Usage

```
tapply(X, INDEX, FUN = NULL, ..., default = NA, simplify = TRUE)
```

Arguments

- | | |
|----------------|--|
| X | an R object for which a <u>split</u> method exists. Typically vector-like, allowing subsetting with <code>[]</code> . |
| INDEX | a <u>list</u> of one or more <u>factors</u> , each of same length as X. The elements are coerced to factors by <u>as.factor</u> . |
| FUN | a function (or name of a function) to be applied, or <code>NULL</code> . In the case of functions like <code>+</code> , <code>%*%</code> , etc., the function name must be backquoted or quoted. If <code>FUN</code> is <code>NULL</code> , <code>tapply</code> returns a vector which can be used to subscript the multi-way array <code>tapply</code> normally produces. |
| ... | optional arguments to <code>FUN</code> : the Note section. |
| default | (only in the case of simplification to an array) the value with which the array is initialized as <code>array(default, dim = ..)</code> . Before R 3.4.0, this was hard coded to <code>array()</code> 's default <code>NA</code> . If it is <code>NA</code> (the default), the missing value of the answer type, e.g. <u><code>NA_real_</code></u> , is chosen (<u><code>as.raw(0)</code></u> for "raw"). In a numerical case, it may be set, e.g., to <code>FUN(integer(0))</code> , e.g., in the case of <code>FUN = sum</code> to 0 or 0L. |

Data frames

E se eu quiser saber a média do comprimento da sépala das 3 espécies... Preciso escrever uma por uma?

```
> tapply(iris$Sepal.Length, iris$Species, FUN = mean)
      setosa versicolor  virginica 
      5.006      5.936      6.588
```

Data frames

Usando o tapply para outras funções

```
> tapply(iris$Sepal.Length, iris$Species, FUN = median)
      setosa versicolor  virginica 
        5.0         5.9         6.5 
> tapply(iris$Sepal.Length, iris$Species, FUN = sd)
      setosa versicolor  virginica 
0.3524897  0.5161711  0.6358796 
> tapply(iris$Sepal.Length, iris$Species, FUN = sum)
      setosa versicolor  virginica 
    250.3     296.8     329.4
```

Mão na massa

1) Abra o data frame IRIS

- Solicite a primeira coluna do IRIS
- Solicite a ultima coluna do IRIS
- Solicite as 5 primeiras linhas do IRIS
- Solicite as linhas de 3 a 7 e de 12 a 19

2) Abra o data frame mtcars

- Selecione as 3 primeiras colunas
- Selecione os valores de mpg acima de 15
- Selecione os valores de cyl \geq 5 e carb $>$ 2

Tibbles

Um dataframe mais moderno

```
> library(tidyverse)
> starwars
# A tibble: 87 x 14
   name      height    mass hair_color skin_color eye_color birth_year sex
   <chr>    <int>    <dbl> <chr>      <chr>      <chr>      <dbl> <chr>
1 Luke~      172      77 blond      fair        blue        19    male
2 C-3PO      167      75 NA         gold        yellow     112    none
3 R2-D2       96      32 NA         white, bl~  red        33    none
4 Dart~      202     136 none       white       yellow     41.9   male
5 Leia~      150      49 brown      light       brown       19    fema~
6 Owen~      178     120 brown, gr~ light       blue       52    male
7 Beru~      165      75 brown      light       blue       47    fema~
8 R5-D4       97      32 NA         white, red red        NA     none
9 Bigg~      183      84 black      light       brown       24    male
10 Obi-~      182      77 auburn, w~ fair        blue-gray   57    male
# ... with 77 more rows, and 6 more variables: gender <chr>,
#   homeworld <chr>, species <chr>, films <list>, vehicles <list>,
#   starships <list>
> |
```


Tibbles

Um dataframe mais moderno

```
# A tibble: 87 x 14
  name height mass hair_color skin_color eye_color birth_year sex gender homeworld species
  <chr> <int> <dbl> <chr> <chr> <chr> <dbl> <chr> <chr> <chr> <chr>
1 Luke~ 172 77 blond fair blue 19 male mascu~ Tatooine Human
2 C-3PO 167 75 NA gold yellow 112 none mascu~ Tatooine Droid
3 R2-D2 96 32 NA white, bl~ red 33 none mascu~ Naboo Droid
4 Dart~ 202 136 none white yellow 41.9 male mascu~ Tatooine Human
5 Leia~ 150 49 brown light brown 19 fema~ femin~ Alderaan Human
6 Owen~ 178 120 brown, gr~ light blue 52 male mascu~ Tatooine Human
7 Beru~ 165 75 brown light blue 47 fema~ femin~ Tatooine Human
8 R5-D4 97 32 NA white, red red NA none mascu~ Tatooine Droid
9 Bigg~ 183 84 black light brown 24 male mascu~ Tatooine Human
10 Obi-~ 182 77 auburn, w~ fair blue-gray 57 male mascu~ Stewjon Human
# ... with 77 more rows, and 3 more variables: films <list>, vehicles <list>, starships <list>
> |
```



Funções

Funções

Embora existam MUUUUUUUUUUUUITOS pacotes e MUUUUITAS funções disponíveis, eventualmente você pode desejar fazer alguma manipulação específica nos seus dados

Se você precisar fazer essa manipulação diversas vezes, você pode criar uma função para automatizar o processo

Funções

Função é um código para executar uma determinada tarefa

Usamos várias funções hoje

- mean
- sum
- prod
- tapply

Funções

Uma função é composta de parâmetros e um retorno

```
nome_funcao <- function(par1, par2){ #parametros da função
  instrucoes #instruções que serão executadas
  saida <- intrucoes #definindo a saida
  return(saida) #definindo o retorno
}
```

Funções

Criando uma função que soma 2 valores

```
▼ soma <- function(a, b) {  
  resultado = a + b  
  return(resultado)  
▲ }
```

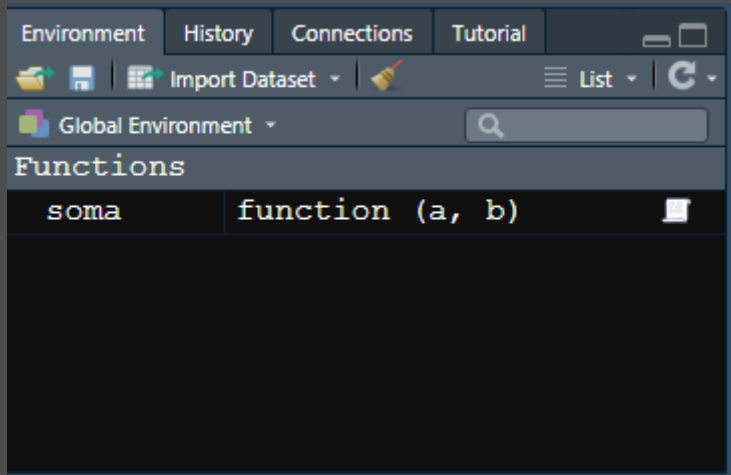
Funções

Como usar a função?

```
> soma(1,2)
[1] 3
> soma(5,7)
[1] 12
```

Funções

Quando você cria uma função, ela vai para o ambiente global, mas as variáveis DENTRO de uma função não vão!



```
> a
Erro: objeto 'a' não encontrado
> |
```

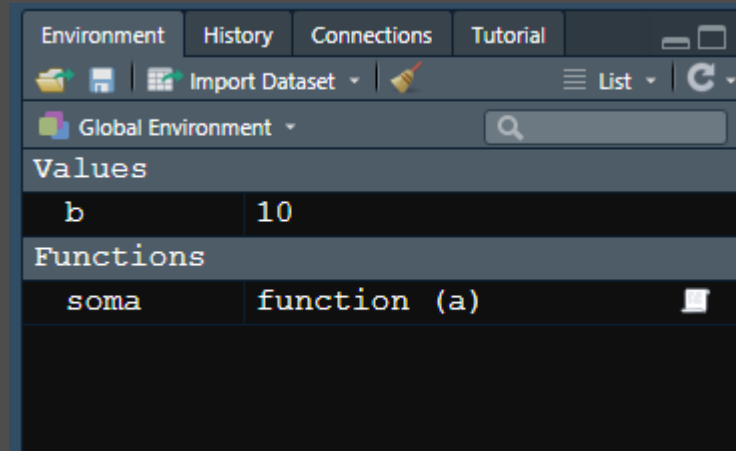

Funções

Cuidado ao usar objetos fora da função

```
b <- 10
soma <- function(a) {
  resultado = a + b

  return(resultado)
}

soma(1)
```



The screenshot shows the R Studio Environment pane. The 'Global Environment' is selected, and the search bar is empty. The 'Values' section displays the variable 'b' with the value '10'. The 'Functions' section displays the function 'soma' defined as 'function (a)'. The function body is not visible in the screenshot.

Values	
b	10
Functions	
soma	function (a)

```
> soma(1)
[1] 11
```

Funções

E se o objeto b não existisse?

```
> rm("b")
> soma <- function(a){
+   resultado = a + b
+
+   return(resultado)
+ }
> soma(1)
Error in soma(1) : objeto 'b' não encontrado
> |
```

Funções

Argumentos padrões:

```
soma <- function(a = 3,b = 5) {  
  soma = a + b  
  return(soma)  
}
```

```
> soma()  
[1] 8
```

Funções

Argumentos padrões:

```
soma <- function(a, b) {  
  soma = a + b  
  return(soma)  
}
```

```
> soma()  
Error in soma() : argumento "a" ausente, sem padrão
```

Funções

Queria fazer uma função que retornasse a soma e a subtração de dois números.. E agora?

```
▼ somasub <- function(a,b) {  
  soma = a + b  
  sub = a - b  
  return(soma)  
  return(sub)  
▲ }
```

```
> somasub(1,2)  
[1] 3
```

O que aconteceu?

Funções

Queria fazer uma função que retornasse a soma e a subtração de dois números.. E agora? Lembra das listas?

```
1 somasub <- function(a,b) {  
2   soma = a + b  
3   sub = a - b  
4   resultado = list(soma, sub)  
5   return(resultado)  
6 }  
7  
8 somasub(1,2)
```

```
> somasub(1,2)  
[[1]]  
[1] 3  
  
[[2]]  
[1] -1
```

Funções

Queria fazer uma função que retornasse a soma e a subtração de dois números.. E agora? Lembra das listas?

```
1 somasub <- function(a,b) {  
2   soma = a + b  
3   sub = a - b  
4   resultado = list(soma = soma, subtracao = sub)  
5   return(resultado)  
6 }  
7  
8 somasub(1,2)
```

```
1  
2  
3 > somasub(1,2)  
4 $soma  
5 [1] 3  
6  
7 $subtracao  
8 [1] -1
```

Mão na massa

- 1) Crie uma função que calcule o quadrado de um número
- 2) Crie uma função que retorne o máximo e o mínimo de um vetor

Obrigada!

Questões?

Amanda Vieira

@mandybouvier

vs.amanda@yahoo.com.br

Stefânia Ventura

Stefania.ventura2@gmail.com

LaSexia



MAIS COISAS SOBRE R

→ Ecologia USP:

<http://ecovirtual.ib.usp.br/doku.php?id=ecovirt:roteiro:soft:rprincip>

→ Cantinho do R:

<https://cantinhodor.wordpress.com/page/1/>