

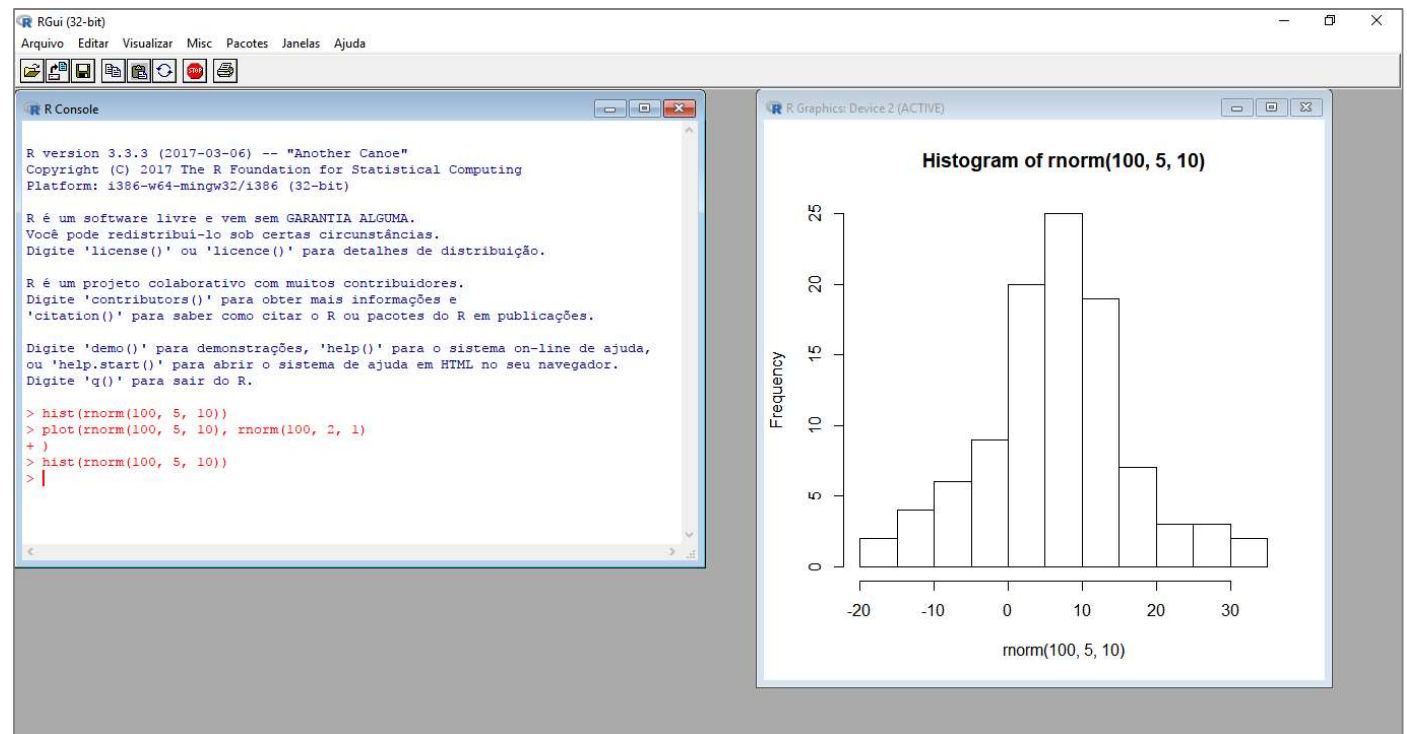


# O ambiente R

Ambiente e linguagem para programação estatística e de gráficos

O termo “ambiente” pretende caracterizar R como um sistema totalmente planejado e coerente

Permite criar ferramentas, montar rotinas de trabalho, flexível.



# Linguagem R

Linguagem de programação

“Gramática”  
Linguagem orientada à objetos

```
x <- 1:10  
x  
[1] 1 2 3 4 5 6 7 8 9 10
```

## Objetos

### VETORES

*Numeric*  
*Integer*  
*Factor*  
*Character*  
*Logical*

1.1, 2, 3.4, 5, 7.8, 10.8  
1, 2, 5, 8, 10, 20  
Categorias/níveis  
Letras, palavras, frases  
Verdadeiro/falso

### Matriz, Lista e *Data Frame*

**Matriz:** possui linhas e colunas onde apenas números são aceitos;  
**Lista:** é uma coleção de elementos que podem ser de diferentes tipos;  
**Data Frame:** Um caso especial de lista onde todos os elementos têm o mesmo comprimento.

# Linguagem R

## Funções e Argumentos

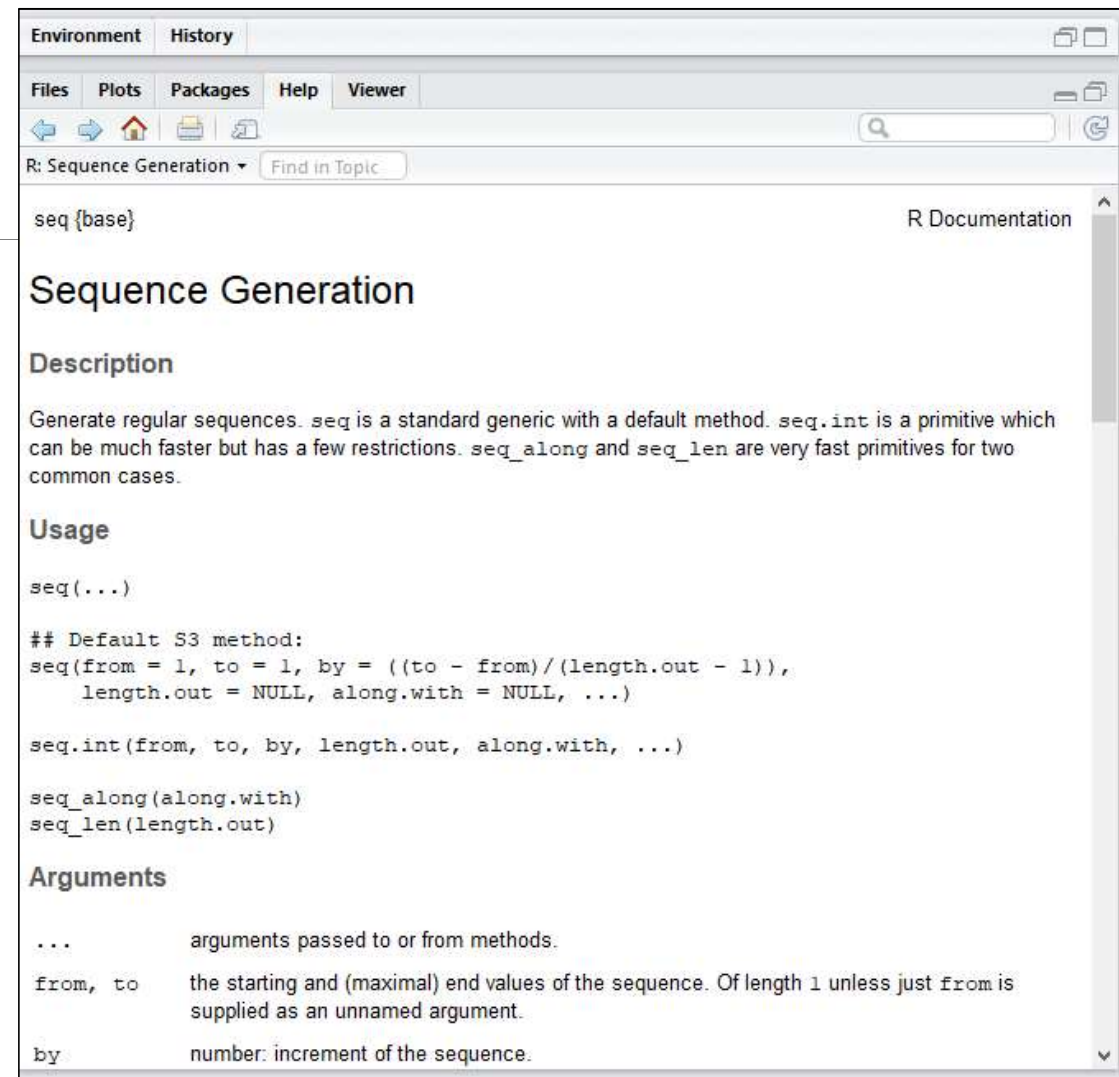
`seq(From = 1, to = 1000, by=100)`

Argumento

Cada argumento é separado por uma vírgula dentro da função

Ler a Ajuda da função dá detalhes de como utilizar a função, do que significa cada argumento, assim como detalhes e exemplos de uso

?seq



The screenshot shows the R help window for the 'seq' function. The window has a menu bar with 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. Below the menu bar is a search bar and a 'Find in Topic' button. The main content area is titled 'Sequence Generation' and includes a 'Description' section, a 'Usage' section, and an 'Arguments' section. The 'Description' section explains that 'seq' is a standard generic with a default method, and 'seq.int' is a primitive which can be much faster but has a few restrictions. The 'Usage' section shows the syntax for 'seq(...)', 'seq.int(from, to, by, length.out, along.with, ...)', 'seq\_along(along.with)', and 'seq\_len(length.out)'. The 'Arguments' section lists the arguments: '...' (arguments passed to or from methods), 'from, to' (the starting and (maximal) end values of the sequence), and 'by' (number: increment of the sequence).

Environment History

Files Plots Packages Help Viewer

R: Sequence Generation Find in Topic

seq {base}

R Documentation

### Sequence Generation

#### Description

Generate regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

#### Usage

```
seq(...)  
  
## Default S3 method:  
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),  
    length.out = NULL, along.with = NULL, ...)  
  
seq.int(from, to, by, length.out, along.with, ...)  
  
seq_along(along.with)  
seq_len(length.out)
```

#### Arguments

...	arguments passed to or from methods.
from, to	the starting and (maximal) end values of the sequence. Of length 1 unless just <code>from</code> is supplied as an unnamed argument.
by	number: increment of the sequence.



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

exemplo 1 curso R.R x

Source on Save Run Source

```
1 a <- 1:10
2 x <- c("a", "b")
3 z <- rep(x, 5)
4 z <- as.factor(z)
5 numeros <- rnorm(10, 5, 20)
6 b <- numeros == a
7 dados <- cbind(a,b,numeros)
8 dados2 <- as.data.frame(cbind(a, b, z))
9
```

Script

Environment History

Import Dataset

Global Environment

Data

dados	num [1:10, 1:3]	1 2 3 4 5 6 7 8 9 10 ...
dados2		10 obs. of 3 variables
values:		
a	int [1:10]	1 2 3 4 5 6 7 8 9 10
b	logi [1:10]	FALSE FALSE FALSE FALSE FALSE ...
numeros	num [1:10]	27.738 -0.921 9.697 -5.737 10.418 ...
x	chr [1:2]	"a" "b"
z	chr [1:10]	"a" "b" "a" "b" "a" "b" "a" "b" "a" "b"

Objetos e histórico

Console

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> b <- numeros == a
> dados <- cbind(a,b,numeros)
> view(dados)
> class(dados)
[1] "matrix"
> length(z)
[1] 30
> x <- c("a", "b")
> z <- rep(x, 5)
> dados2 <- cbind(a, b, z)
> view(dados2)
> view(dados)
> view(dados2)
> class(dados2)
[1] "matrix"
> dados2 <- as.data.frame(cbind(a, b, z))
>
```

Console

Files Plots Packages Help Viewer

New Folder Delete Rename More

Home

	Name	Size	Modified
<input type="checkbox"/>	.Rhistory	942 B	Apr 24, 2017, 4:03 PM
<input type="checkbox"/>	042-047_Botanica_241.jpg	5.1 MB	Mar 22, 2016, 8:15 PM
<input type="checkbox"/>	A. th. V. Rodrigues_1.jpg	76.6 KB	Jul 7, 2015, 10:34 PM
<input type="checkbox"/>	Audacity.auf	51 Kb	Apr 8, 2017, 11:52 PM
<input type="checkbox"/>	Audacity_data		
<input type="checkbox"/>	CIMG6124.JPG	1.5 MB	Jul 28, 2015, 12:44 PM
<input type="checkbox"/>	CIMG6250.JPG	1.5 MB	Apr 4, 2017, 6:56 PM
<input type="checkbox"/>	copia		
<input type="checkbox"/>	CyberLink		

Ajuda, gráficos, arquivos e etc..

# O Script

---

Sua rotina de trabalho pode ser reproduzível com um Script;

Ctrl + R (Windows) ou Ctrl + ENTER (Mac ou Linux) executa o comando no Script;

Use apenas o ENTER para executar um comando no Console;

Use # para comentar e anotar informações no Script ;

Mantenha seu script organizado;

Divida os códigos mais longos em mais de uma linha;

Use o Tab e linhas em branco para dar noção de emblocamento dos códigos

```

1  ### Montagem da matriz de média ponderada da comunidade (CWM)
2
3  install.packages("FD")
4  library(FD)
5  library(dplyr)
6
7  ## Matriz de abundância
8
9  DA <- read.csv2("Densidade de Indivíduos por UA.csv", row.names = 1)
10
11  ## os nomes da matriz de abundancia e a ordem devem ser identicos ao da matriz de a
12  ## Usar script PARTE 2 para gerar a matriz TRAIT
13
14  rownames(TRAIT) <- gsub(" ", ".", rownames(TRAIT))
15  spp <- rownames(TRAIT)
16
17  ## matriz de abundancia somente para as espécies presentes em TRAIT
18  a <- DA[,spp]
19
20  ### CWM
21
22  cwm <- functcomp(TRAIT, as.matrix(a))
23
24  ### Exportar matrix
25  write.table(cwm, "CWM 02-05-2017.csv", sep = ";", dec = ",")
26
27
28  ##### Explorar relações com a altitude
29
30  Uas <- read.csv2("Uas da Floresta Pluvial - Vale do Itajai.csv", row.names = 1)
31
32  plot(Uas$Altitude, log10(cwm$Peso.semente + 1))
33  cor.test(Uas$Altitude, log10(cwm$Peso.semente))
34  abline(lm(log10(cwm$Peso.semente + 1) ~ Uas$Altitude))
35

```

Agrupe a abertura de pacotes no início do Script.

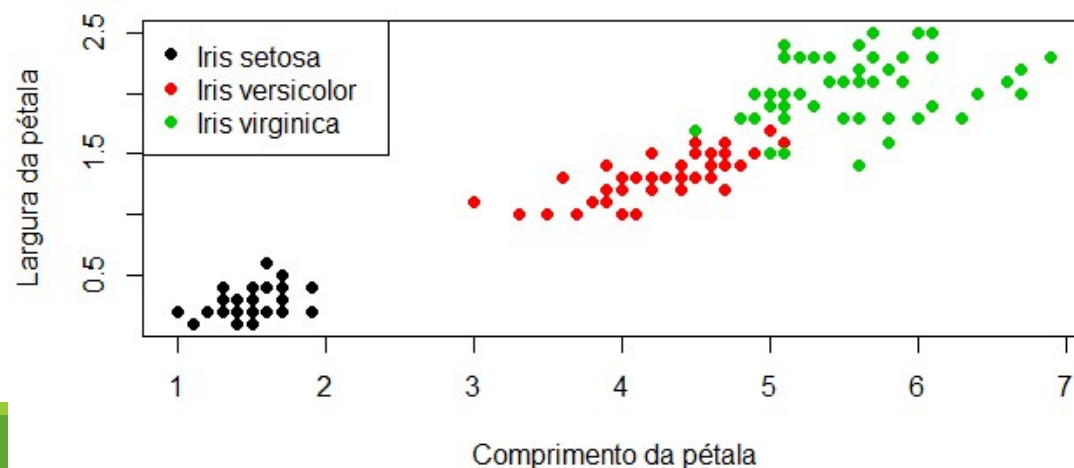
Sequencias de trabalho em bloco.

Espaço antes e depois da seta.  
E após virgulas.



```
1 data("iris")
2
3 iris
4 plot(iris$Petal.Length, iris$Petal.width, col = c(1:3)[iris$Species],
5      pch = 16, xlab = "Comprimento da pétala", ylab = "Largura da pétala",
6      main = "Relação Comprimento:Largura de Pétalas
7      para 3 espécies do genero Iris")
8 legend("topleft", legend = paste("Iris", levels(iris$Species)), pch = 16,
9      col = c(1:3))
10 |
```

**Relação Comprimento:Largura de Pétalas  
para 3 espécies do genero Iris**



Note que há muitos argumentos utilizados nas funções `plot` e `legend`

Para deixar o código um pouco mais agradável visualmente, pula-se para outra linha após uma vírgula.

A vírgula deve estar no fim de uma linha para que a função seja executada corretamente.

*Tools -> Global Options -> Code -> Display*

☒ Show margin

Margin column 80

# Pacotes

---

`install.packages("nome_do_pacote")` - Baixa e instala o pacote

`library(nome_do_pacote)` - Carrega o pacote

Os pacotes contém funções que não existem no base R.

Assim, quando for utilizar uma função no R o pacote que contem a função deve estar carregado. Por isso é comum ver uma lista de pacotes sendo carregados no inicio de um script.

A instalação é feita apenas uma vez.



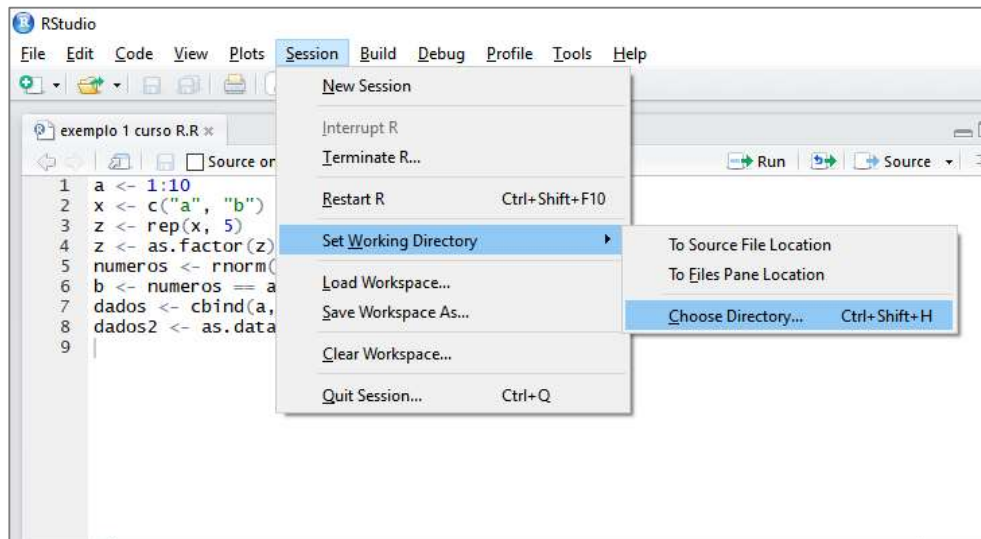


# Iniciando os trabalhos

Sempre confira a pasta de trabalho que o R está:

`getwd()` – retorna qual é a pasta de trabalho que está sendo utilizada

`setwd("Documentos/R")` – configura qual a pasta de trabalho você deseja usar



Evite abusar dos atalhos de mouse.

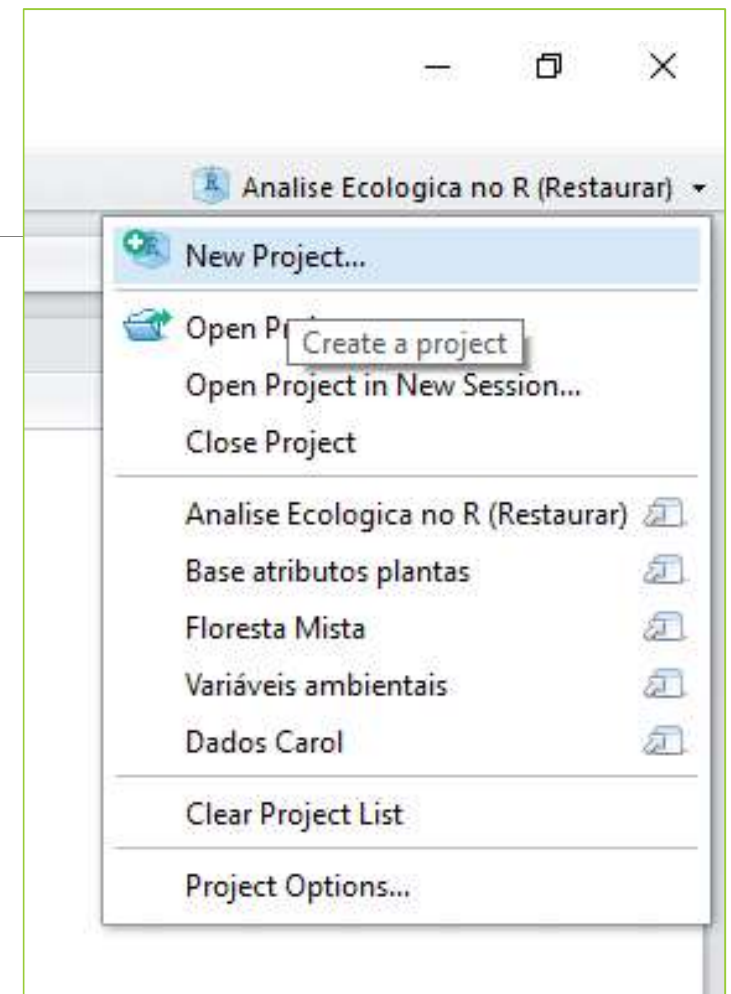
Manter tudo que você faz em códigos permite que você possa saber exatamente o que fez quando usou da última vez.

# Criar um projeto

Ajuda a manter seus diferentes trabalhos organizados.

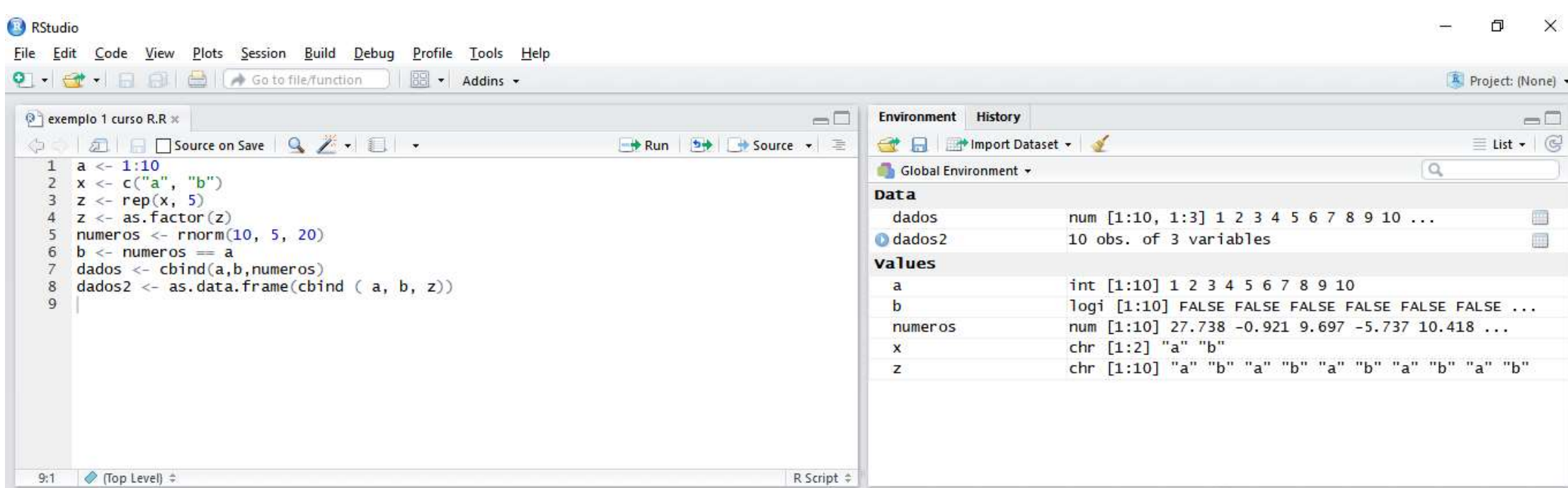
Ao criar um projeto você associa ele a uma pasta de trabalho

Diminuindo a necessidade de utilizar as funções `getwd()` e `setwd()`



# Objetos e o ambiente global

Objetos são gravados na memória RAM (ambiente global)



The screenshot displays the RStudio interface. The main editor window on the left contains the following R code:

```
1 a <- 1:10
2 x <- c("a", "b")
3 z <- rep(x, 5)
4 z <- as.factor(z)
5 numeros <- rnorm(10, 5, 20)
6 b <- numeros == a
7 dados <- cbind(a,b,numeros)
8 dados2 <- as.data.frame(cbind ( a, b, z))
9
```

The Environment pane on the right shows the state of the global environment. It lists the objects created by the code:

Global Environment	
<b>Data</b>	
dados	num [1:10, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
dados2	10 obs. of 3 variables
<b>values</b>	
a	int [1:10] 1 2 3 4 5 6 7 8 9 10
b	logi [1:10] FALSE FALSE FALSE FALSE FALSE FALSE ...
numeros	num [1:10] 27.738 -0.921 9.697 -5.737 10.418 ...
x	chr [1:2] "a" "b"
z	chr [1:10] "a" "b" "a" "b" "a" "b" "a" "b" "a" "b"

# Criando vetores

---

Para criar um objeto você deve assinalar um valor à um nome:

```
obj <- 5
```

para mais de um valor utilize a função `c()` ou valores numéricos sequencias são obtidos com símbolo `":"`.

```
obj <- c(2, 3, 4, 5)
```

ou

```
obj <- 2:5
```

# Criando vetores

---

Vetores também podem conter caracteres, eles sempre devem estar entre áspas.

```
obj <- c("a","b","c")
```

# Crie repetições - Função rep()

---

Cria repetições

```
rep(4, 3)
```

```
rep(x, 4)
```

```
rep(x, each = 4)
```

```
rep(c(4, 3), each = 3)
```

# Crie sequencias - Função seq()

---

Cria sequencias padronizadas.

Sequencia de 1 a 100

```
seq(1, 100)
```

Sequencia de 1 a 100 de 10 em 10.

```
seq(1, 100, by = 10)
```



# Crie valores aleatório (uniformes ou normais)

---

**Para valores aleatórios uniformes:**

```
runif(n, min, max)
```

```
runif(10, 0, 2) # dez valores aleatórios entre 0 e 2
```

**Para valores aleatórios de distribuição normal:**

```
rnorm(n, mean, sd)
```

```
rnorm (10, 5, 1) # dez valores aleatórios com média 5 e desvio padrão 1
```

# Fixar aleatorização - Função set.seed()


---

```
set.seed(12)
```

```
rnorm (10, 5, 1)
```

```
[1] 3.519432 6.577169 4.043256 4.079995 3.002358 4.727704 4.684651 4.371745
```

```
[9] 4.893536 5.428015
```



# Extrair valores aleatórios - Função `sample()`

---

Criando valores aleatórios:

```
random <- rnorm(10, 5, 1)
```

Extraindo valores aleatórios:

```
sample(random, 5)
```

```
sample(random, 15)
```

```
sample(random, 15, replace = TRUE) #Faz amostras com repetição.
```

# Crie matrizes - função `matrix()`

---

Matrizes aceitam apenas valores numéricos e possuem duas dimensões (linhas e colunas)

```
matrix(data, nrow, ncol)
```

```
matrix(1:20, 10, 2)
```

```
matrix(c(a, 20:29), 10, 2)
```

# Crie data frames - função data.frame()

---

Data frames aceitam qualquer tipo de valor (numérico, caracter, lógico). Os valores devem ter o mesmo comprimento

```
data.frame (a, b, z) #utilizou os objetos já criados para criar um data frame
```

```
data.frame (c(1:20), a, b) # identifique o erro
```

# Crie listas - Função list()

---

Listas podem ser compostas por elementos de diferentes tamanhos. É a maneira mais usual de funções retornarem valores.

```
list(dados, b, x) # utilizou objetos já criados para gerar uma lista
```

# Selecionando valores de objetos vetores

Conchetes [] são usados para selecionar valores de um objeto.

Execute as funções

abaixo:

```
dim(a)
```

```
class(a)
```

```
length(a)
```

```
a[5] # Para selecionar o quinto valor
```

```
a[a>2] #Selecionar de acordo com uma condição lógica  
(valores contidos em a maior que 2)
```

## Base R

Cheat Sheet

Conditions	a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
	a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

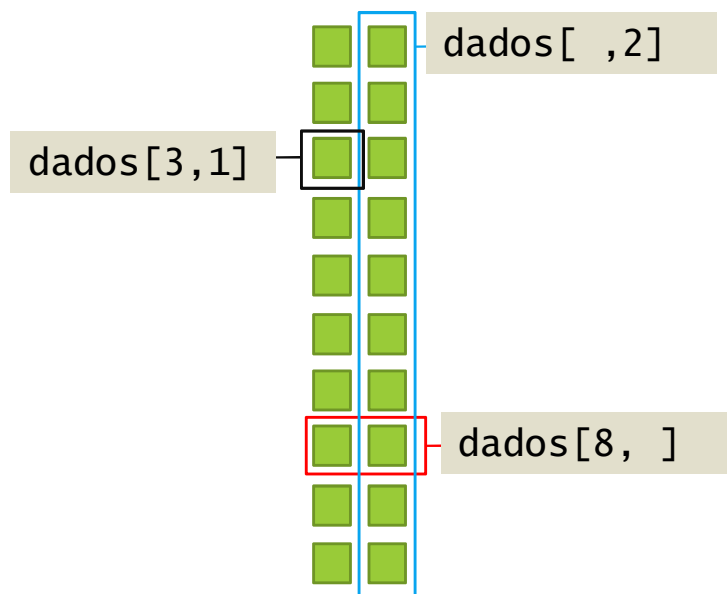
Compartilhado no Classroom



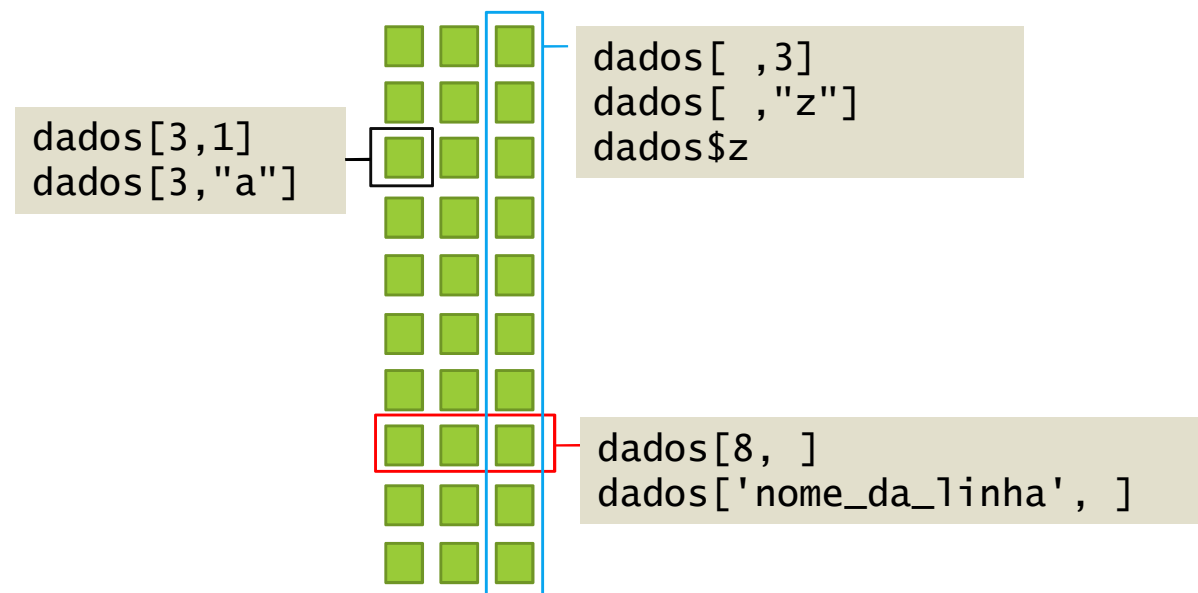
# Selecionando valores em matrizes e data frames

`objeto[linha,coluna]`

Objeto Matriz



Objeto Data frame



# Selecionando valores em listas

---

```
objeto$nome_elemento  
objeto[[1]]  
objeto[[1]][2]
```

Exemplo:

```
lista <- list(dados, dados1, b, x)  
names(lista)  
names(lista) <- c("dados", "dados1", "b", "x")
```

# Símbolos

---

`<-`

Assinala

`[ ]`

Seleciona

`$`

Seleciona

`( )`

Indica argumentos de uma função

`{ }`

Indica o conteúdo de uma função

```
c(0,1,5)[2]
```

```
l <- list(dados, b, x)
```

```
l[[1]][2]
```

```
Iris$Sepal.Length[2:4]
```

```
Media <- function(x) {  
    mean(x)
```

```
}
```

```
Media(x)
```

# Comandos úteis

---

```
class(x) # classe do objeto  
str(x) # Estrutura do objeto  
dim(x) # Dimensões do objeto
```

```
length(x) # Comprimento do objeto  
names(x) # Nomes contidos num objeto vetor, data.frame ou lista  
rownames(x) # Nomes das linhas em um data frame
```

```
ncol(x) # Número de linhas  
nrow(x) # Número de colunas
```