

Introdução ao uso de dados geoespaciais no R

3 Estrutura e manipulação de dados

Maurício H. Vancine

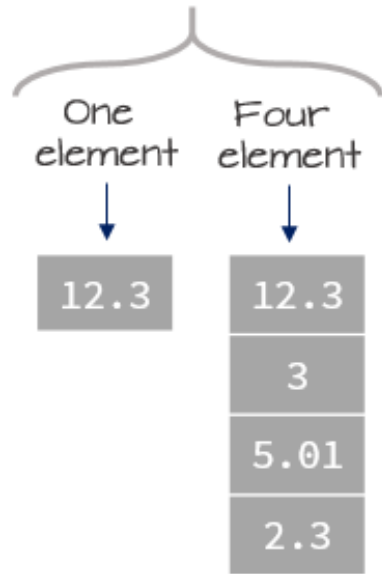
Milton C. Ribeiro

UNESP - Rio Claro

Laboratório de Ecologia Espacial e Conservação (LEEC)

25/10/2021-05/11/2021

vector



dataframe

x	y
12.3	ace
3	tea
5.01	oil
2.3	tree

matrix

12.3	0.1
3.0	5.2
5.01	3.0
2.3	0.1

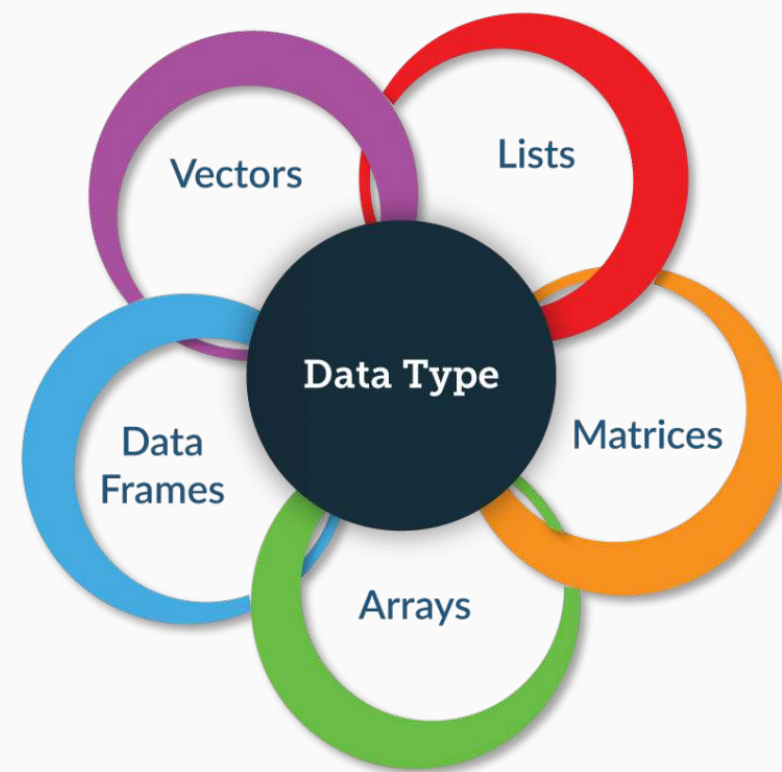
list

x	y
12.3	ace
3	tea
5.01	oil
2.3	tree
3	
$Y \sim x - 1$	
some text	

3 Estrutura e manipulação de dados

Conteúdo

1. Atributos dos objetos
2. Manipulação de dados unidimensionais
3. Manipulação de dados multidimensionais
4. Valores faltantes e especiais
5. Diretório de trabalho
6. Importar dados
7. Conferência de dados importados
8. Exportar dados



3 Estrutura e manipulação de dados

Script

```
03_script_intro_geoespacial_r.R
```

1. Atributos dos objetos

Atribuição

palavra <- dados

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

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

1. Atributos dos objetos

Atributos dos objetos no R

Objetos possuem **três características**:

1. **Nome**: palavra que o R reconhece os dados atribuídos
2. **Conteúdo**: dados em si
3. **Atributos**: modos (*natureza*) e estruturas (*organização*)

1. Atributos dos objetos

Atributos dos objetos no R

```
# atributos  
attributes(dune)
```

```
## $names
```

```
## [1] "Achimill" "Agrostol" "Airaprae" "Alopgeni" "Anthodor" "Bellpere" "Bromhord" "Chenalbu" "Cirsarve" "Comapalu"
```

```
## [13] "Empenigr" "Hyporadi" "Juncarti" "Juncbufo" "Lolipere" "Planlanc" "Poaprat" "Poatriv" "Ranuflam" "Rumeacet"
```

```
## [25] "Scorautu" "Trifprat" "Trifrepe" "Vicilath" "Bracruta" "Callcusp"
```

```
##
```

```
## $row.names
```

```
## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12" "13" "14" "15" "16" "17" "18" "19" "20"
```

```
##
```

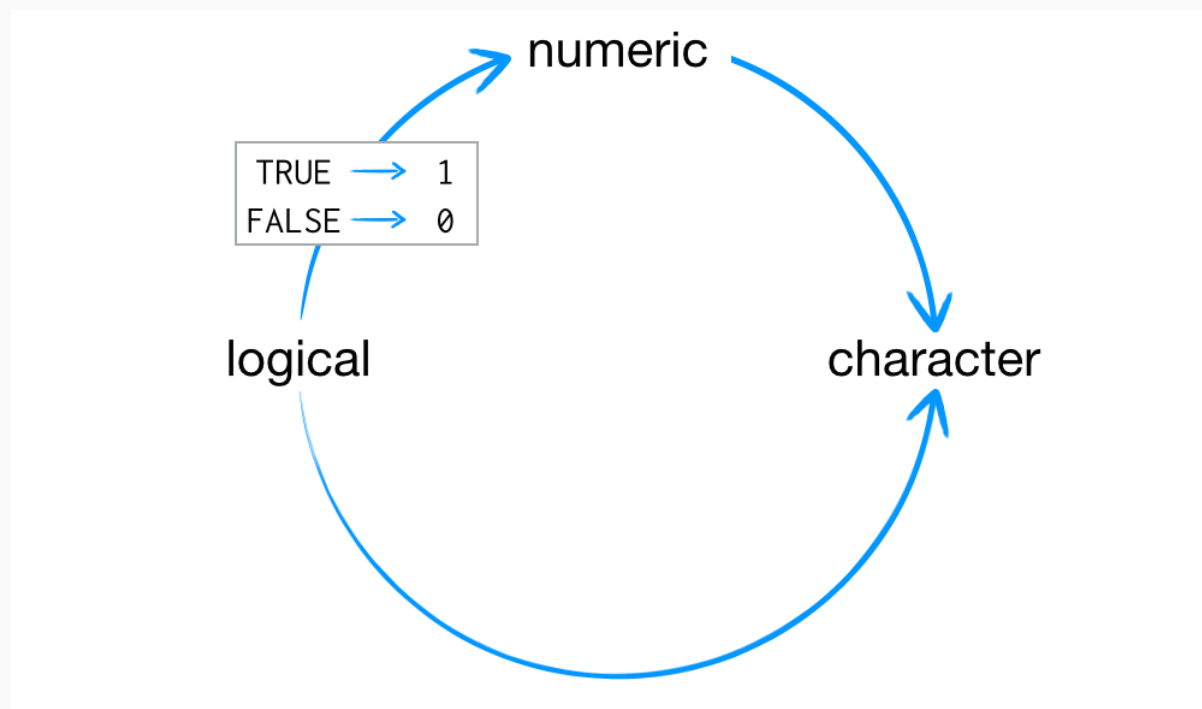
```
## $class
```

```
## [1] "data.frame"
```

1. Atributos dos objetos

Modos (*natureza*): numeric (double), numeric (integer), character, logical e complex

Natureza dos **elementos** que compõem os objetos



1. Atributos dos objetos

1. Numeric (double): números decimais

```
# numeric double  
obj_num_dou ← 1  
obj_num_dou
```

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

```
# mode  
mode(obj_num_dou)
```

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

```
# type  
typeof(obj_num_dou)
```

```
## [1] "double"
```

1. Atributos dos objetos

2. Numeric (integer): números inteiros

```
# numeric integer  
obj_num_int <- 1L  
obj_num_int
```

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

```
# mode  
mode(obj_num_int)
```

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

```
# type  
typeof(obj_num_int)
```

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

1. Atributos dos objetos

3. Character: texto ou caracteres

```
# character  
obj_cha ← "a" # atencao para as aspas  
obj_cha
```

```
## [1] "a"
```

```
# mode  
mode(obj_cha)
```

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

1. Atributos dos objetos

4. **Logical:** ocorrência ou não de um evento (TRUE ou FALSE)

```
# logical  
obj_log ← TRUE # maiusculas e sem aspas  
obj_log
```

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

```
# mode  
mode(obj_log)
```

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

1. Atributos dos objetos

5. **Complexo:** números complexos

```
# complex  
obj_com ← 1+1i # parte imaginaria  
obj_com
```

```
## [1] 1+1i
```

```
# mode  
mode(obj_com)
```

```
## [1] "complex"
```

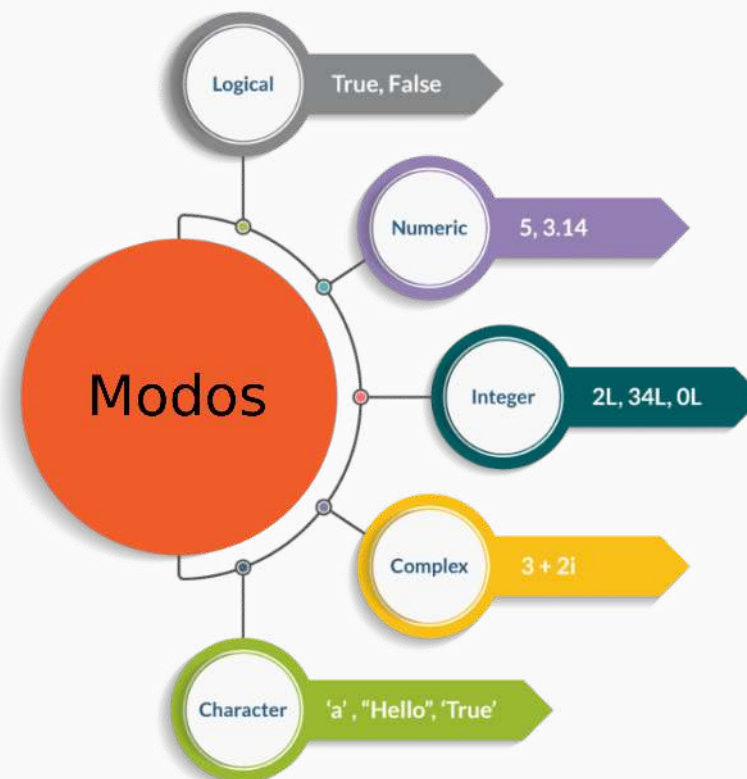
1. Atributos dos objetos

Resumindo:

A **natureza** dos **elementos** definirá os **modos** dos objetos

Modos:

1. Numeric double (**número decimal**): 1
2. Numeric integer (**número inteiro**): 1L
3. Character (**texto**): "a", "2500", "amostra_01"
4. Logical (**lógico**): TRUE ou FALSE
5. Complexo (**complex**): 1+1i



1. Atributos dos objetos

Verificar o modo dos objetos ou fazer a conversão entre os modos

```
# verificar o modo dos objetos
```

```
is.numeric()
```

```
is.integer()
```

```
is.character()
```

```
is.logical()
```

```
is.complex()
```

```
# conversoes entre modos
```

```
as.numeric()
```

```
as.integer()
```

```
as.character()
```

```
as.logical()
```

```
as.complex()
```

```
# exemplo
```

```
as.character(obj_num_dou)
```

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

1. Atributos dos objetos

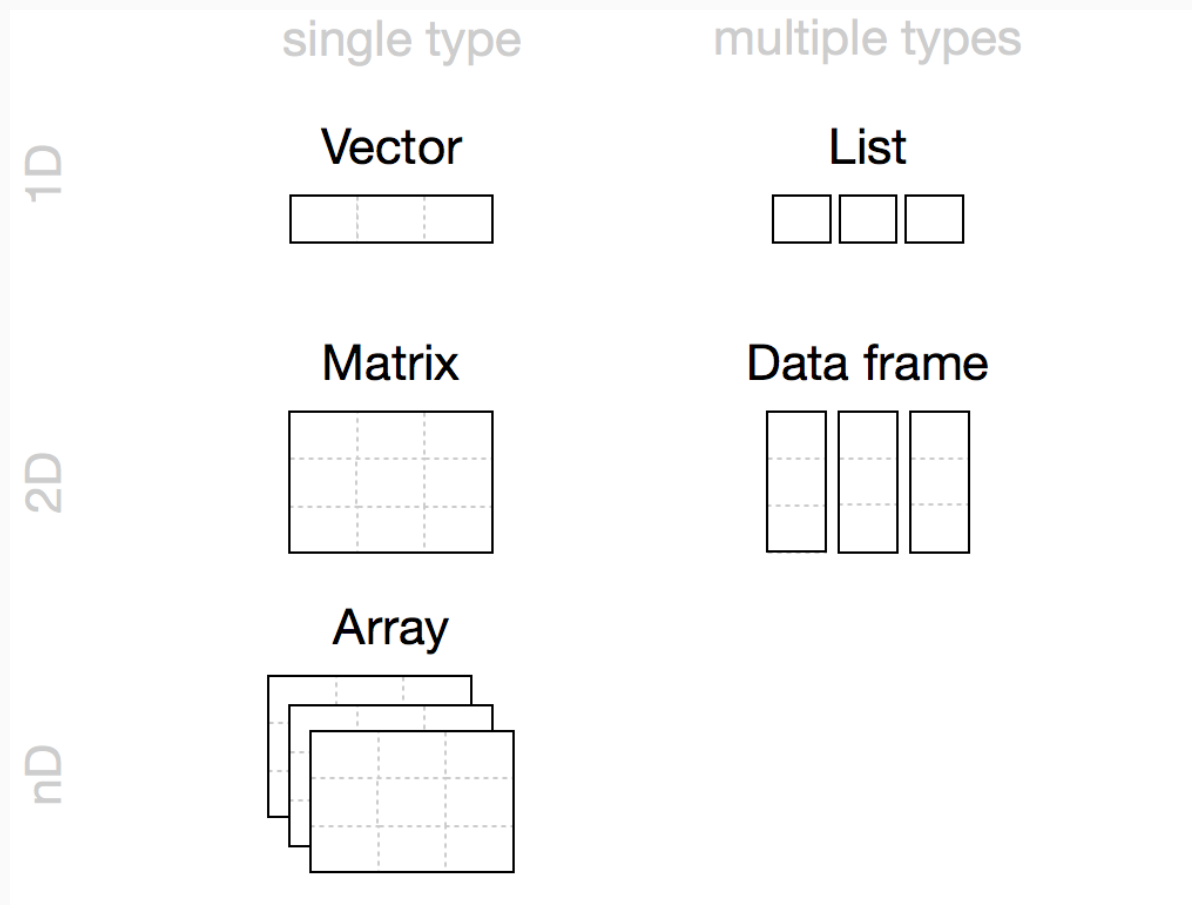
Estruturas (*organização*): vector, factor, matrix, array, data frame e list

Organização (**modos** e **dimensionalidade**) dos elementos dos objetos

	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data Frame
<u>nd</u>	Array	

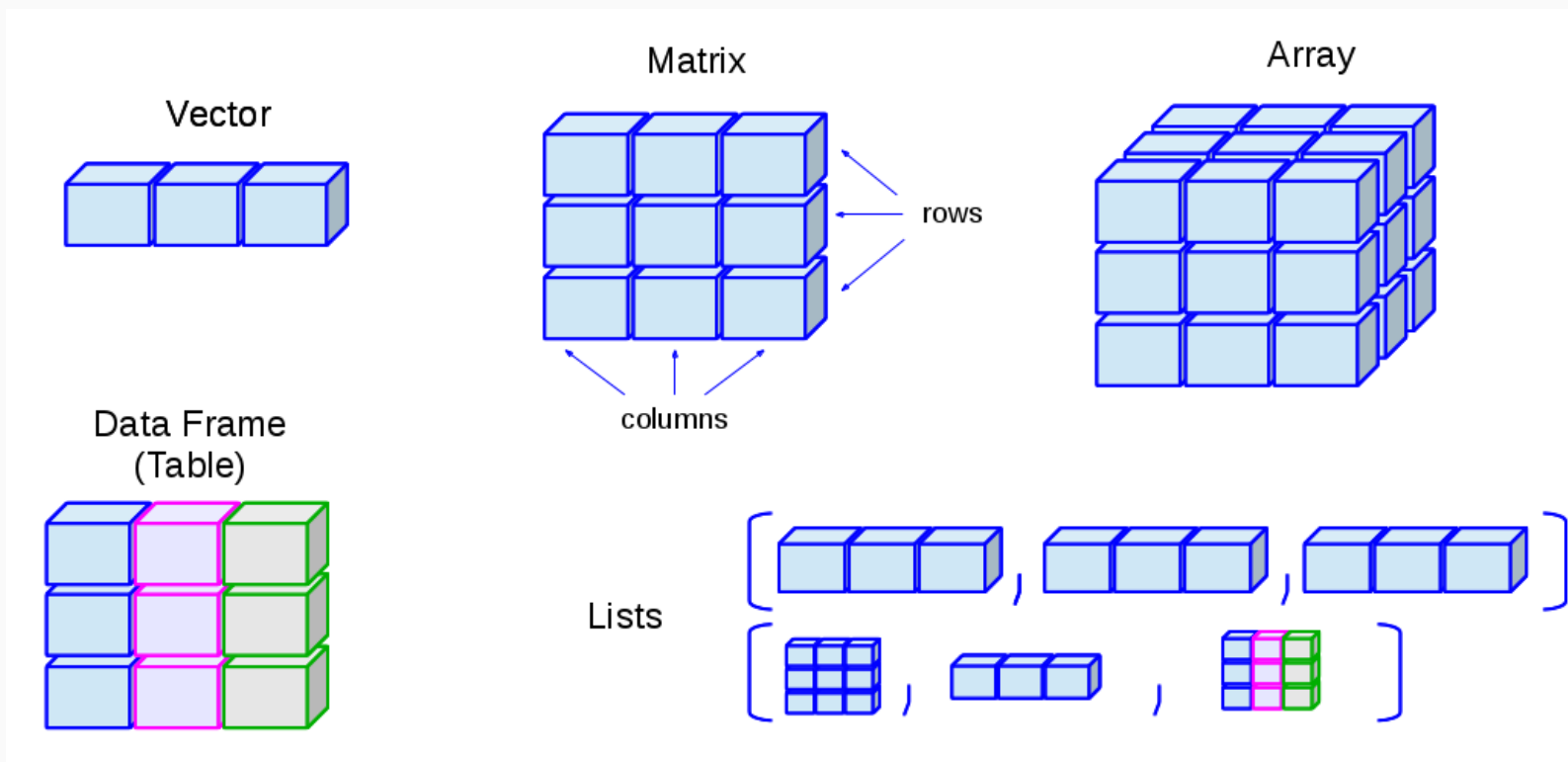
1. Atributos dos objetos

Estruturas (*organização*): vector, factor, matrix, array, data frame e list



1. Atributos dos objetos

Estruturas (*organização*): vector, factor, matrix, array, data frame e list



1. Atributos dos objetos

1. Vector: homogêneo (*um modo*) e unidimensional (*uma dimensão*)

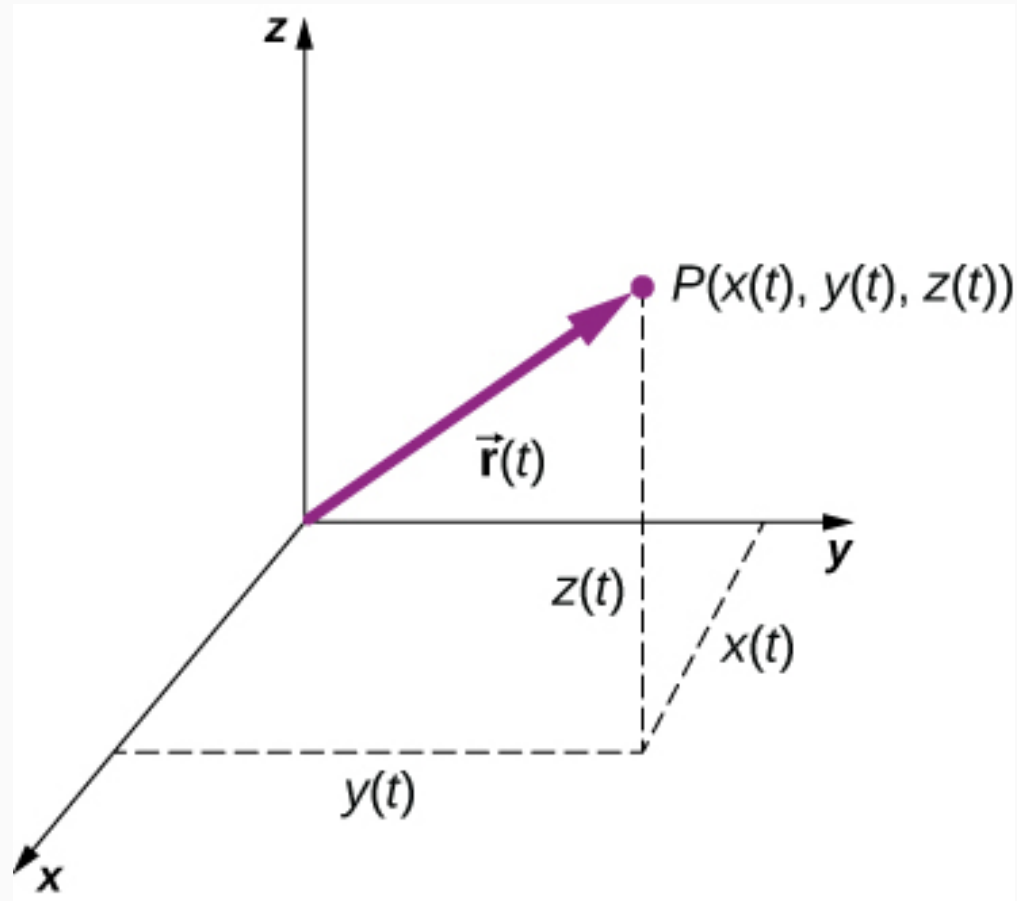
O **vetor** representa medidas de uma **variável quantitativa** (discretas ou contínuas) ou **descrição** (informações em texto)

Ex.: medidas tomadas em campo ao longo de uma amostragem de 5 meses

1. Amostras: {"amostra_01", "amostra_02", "amostra_03", "amostra_04", "amostra_05"}
2. Temperatura: {15, 18, 20, 22, 18}
3. Abertura do dossel: {0.37, 0.45, 0.65, 0.75, 0.40}
4. Abundância de uma espécie: {6, 3, 0, 0, 2}

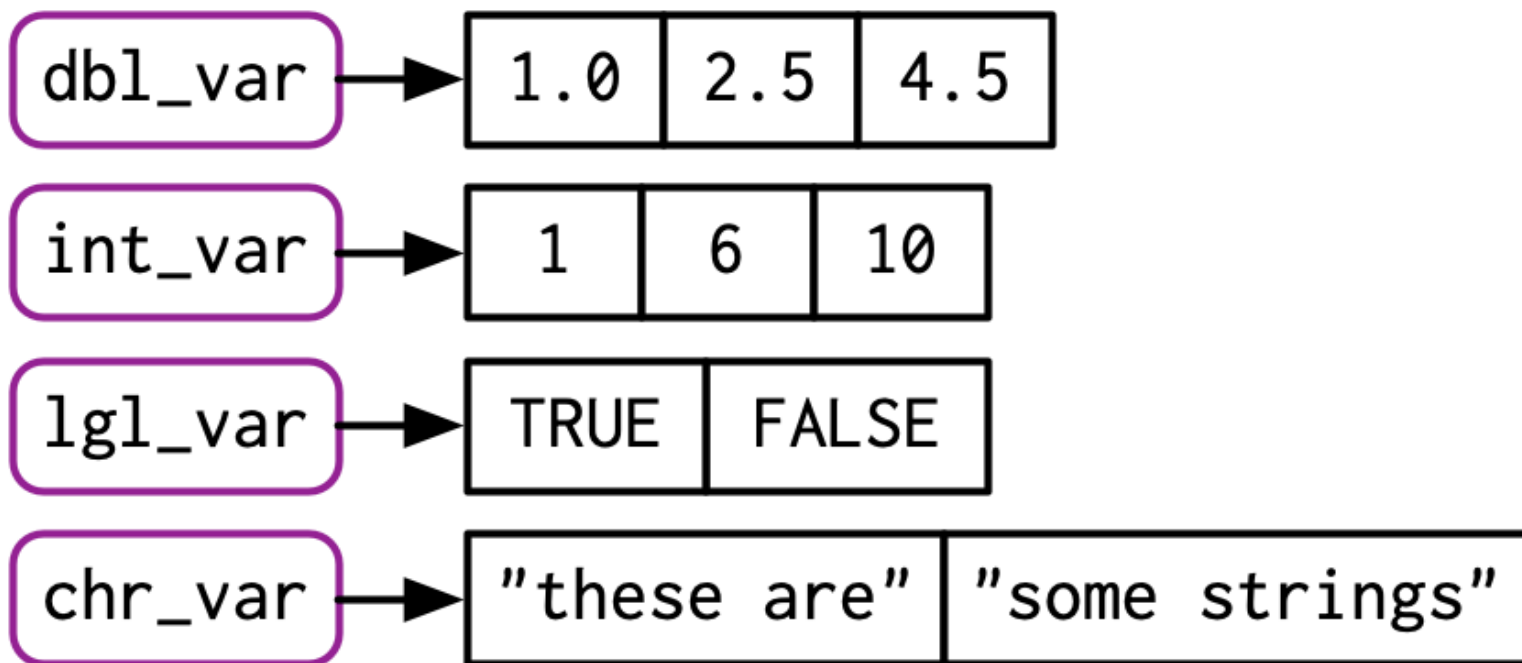
1. Atributos dos objetos

Não me refiro exatamente ao vetor da matemática



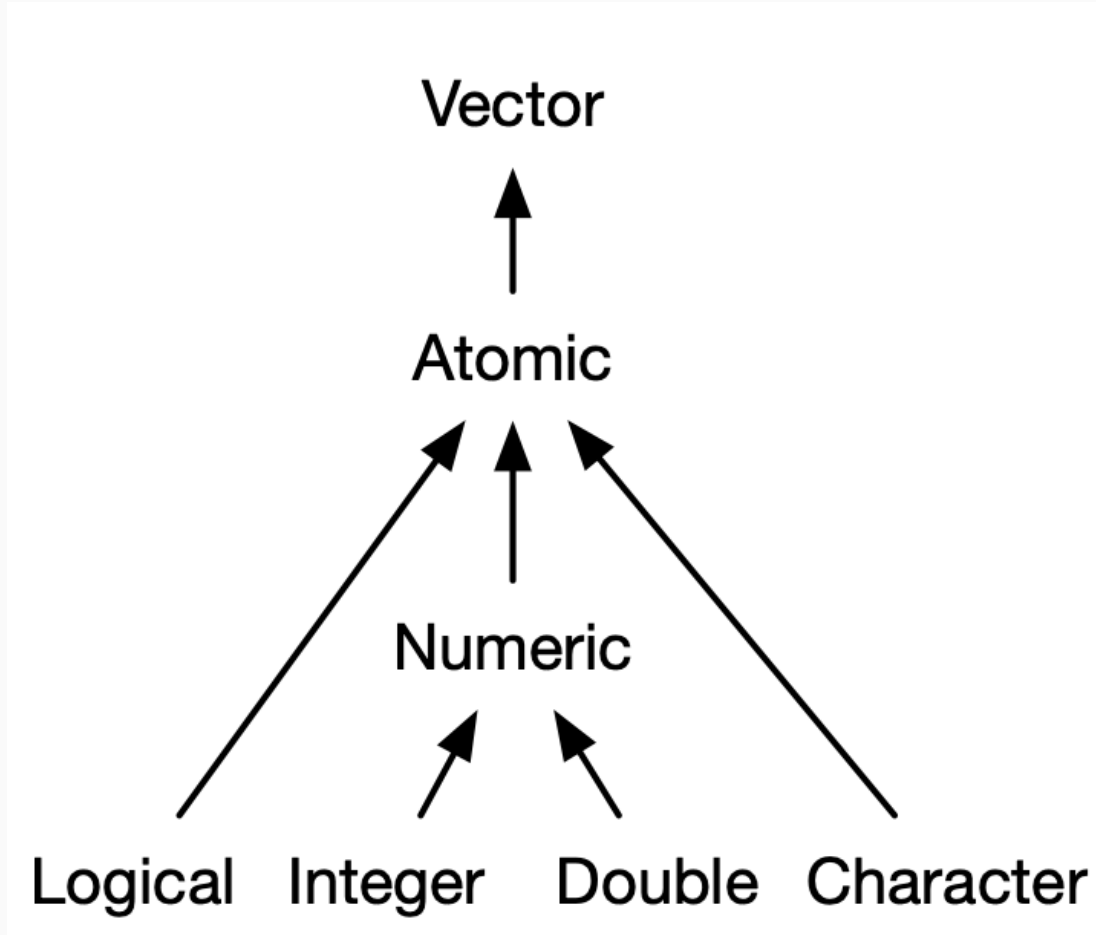
1. Atributos dos objetos

Sequência de elementos



1. Atributos dos objetos

Tipos



1. Atributos dos objetos

Há diversas formas de se criar um **vetor**:

1. Concatenar elementos

```
# concatenar elementos numericos  
temp ← c(15, 18, 20, 22, 18)  
temp
```

```
## [1] 15 18 20 22 18
```

```
# concatenar elementos de texto  
amos ← c("amostra_01", "amostra_02", "amostra_03", "amostra_04")  
amos
```

```
## [1] "amostra_01" "amostra_02" "amostra_03" "amostra_04"
```

1. Atributos dos objetos

Há diversas formas de se criar um **vetor**:

2. Sequência

```
# sequencia unitaria (x1:x2)
se ← 1:10
se
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
# sequencia com diferentes espacamentos
se_e ← seq(from = 0, to = 100, by = 10)
se_e
```

```
## [1] 0 10 20 30 40 50 60 70 80 90 100
```


1. Atributos dos objetos

Há diversas formas de se criar um **vetor**:

3. Repetição

```
# rep(x, times) # repete x tantas vezes
rep_times <- rep(x = c(1, 2), times = 5)
rep_times
```

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

```
# rep(x, each) # retete x tantas vezes de cada
rep_each <- rep(x = c("a", "b"), each = 5)
rep_each
```

```
## [1] "a" "a" "a" "a" "a" "b" "b" "b" "b" "b"
```

1. Atributos dos objetos

Há diversas formas de se criar um **vetor**:

4. "Colar" palavras com uma sequência numérica

```
# palavra e sequencia numerica - sem separacao definida (" ")  
am1 ← paste("amostra", 1:5)  
am1
```

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

```
# palavra e sequencia numerica - separacao por "_0"  
am2 ← paste("amostra", 1:5, sep = "_0")  
am2
```

```
## [1] "amostra_01" "amostra_02" "amostra_03" "amostra_04" "amostra_05"
```

1. Atributos dos objetos

Há diversas formas de se criar um **v vetor**:

5. Amostrando aleatoriamente elementos

```
# amostragem aleatória - sem reposição  
sa_sem_rep ← sample(1:100, 10)  
sa_sem_rep
```

```
## [1] 22 44 64 7 5 31 32 82 39 13
```

```
# amostragem aleatória - com reposição  
sa_com_rep ← sample(1:10, 100, replace = TRUE)  
sa_com_rep
```

```
## [1] 10 6 2 3 1 4 9 6 10 7 3 2 9 10 9 3 7 3 6 4 10 2 7 1 2 3 1 10 5 2 3 3 10 2 10 4  
## [47] 10 7 3 9 2 6 9 8 6 5 10 1 9 10 1 6 1 9 4 3 5 6 5 10 4 2 7 4 9 9 4 9 3 2 10 1  
## [93] 10 7 7 6 9 2 9 1
```

Exercícios

Exercício 05

Vector

Escolham números para jogar na mega-sena

Lembrando: são 6 valores de 1 a 60 e atribuem a um objeto "agora_vai"

04 : 00

Exercício 05

Resposta

```
# exercicio 05  
agora_vai <- sample(1:60, 6, replace = FALSE)  
agora_vai
```

```
## [1] 47  8 36  6 10 27
```

E se eu criar um vetor com elementos de **modos diferentes?**

1. Atributos dos objetos

Vetor com elementos de **modos diferentes**:

```
# vetor com modos diferentes  
ve ← c(1, "a", 3)  
ve
```

```
## [1] "1" "a" "3"
```

```
# vetor com modos diferentes  
ve ← c(1, "a", TRUE)  
ve
```

```
## [1] "1"      "a"      "TRUE"
```

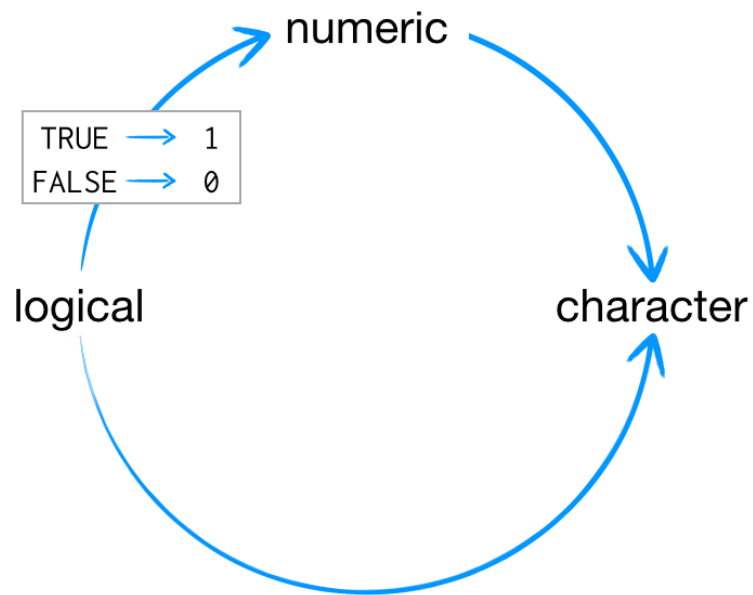

1. Atributos dos objetos

Coerção

Mudança do **modo** dos elementos para um **mesmo modo**

Essa mudança segue essa ordem:

DOMINANTE **character >> numeric >> logical** RECESSIVO



1. Atributos dos objetos

Conversão

Podemos **forçar** um vetor a ter um **modo específico**

Ideia semelhante: mudar o **tipo da célula** numa planilha eletrônica

Conversão

```
# funcoes de conversao  
as.character()  
as.integer()  
as.numeric()  
as.double()  
as.integer()  
as.logical()
```

1. Atributos dos objetos

2. Factor: homogêneo (*um modo* - sempre *numeric*), unidimensional (*uma dimensão*) e possui ainda **levels** (níveis)

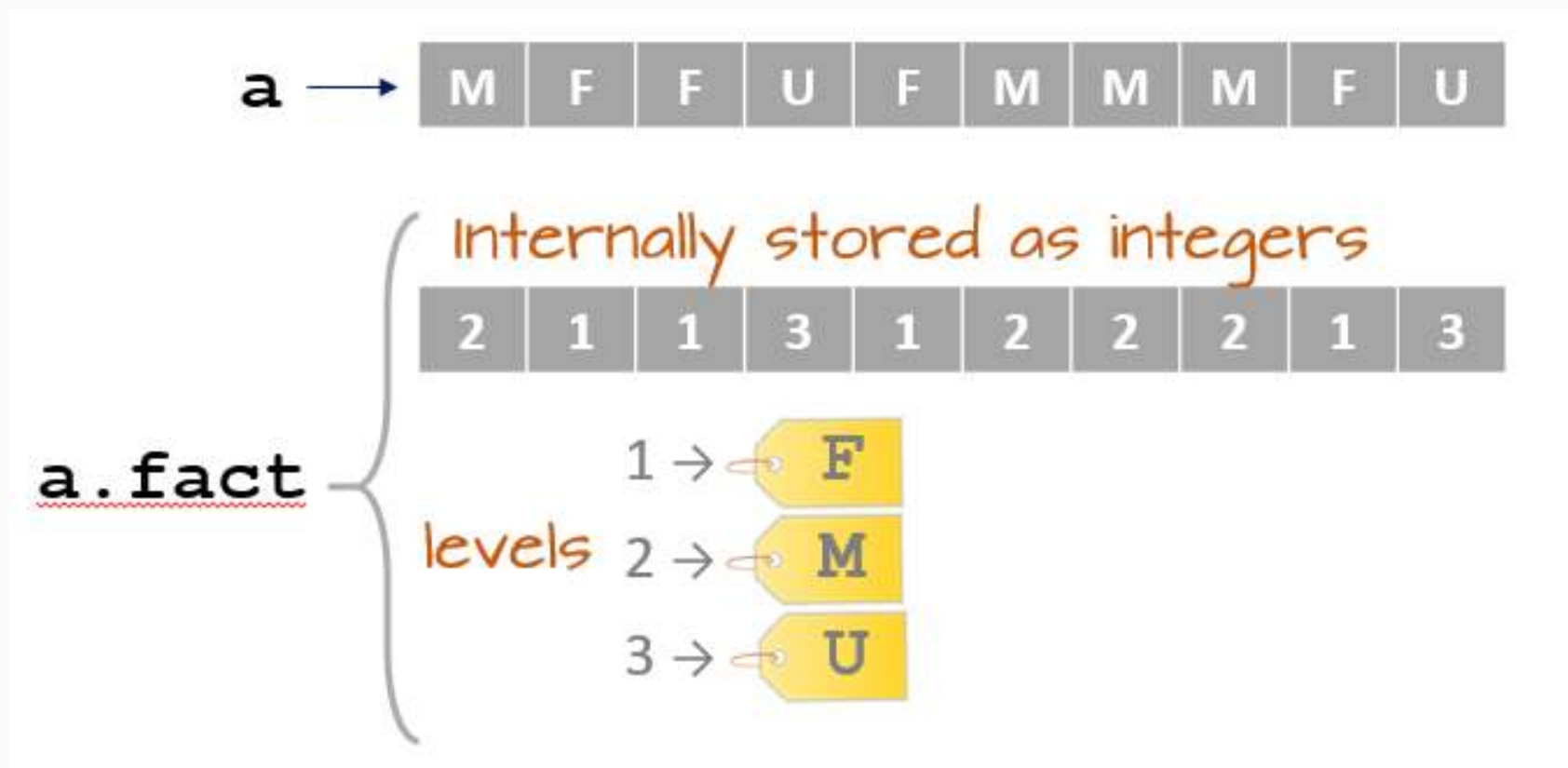
O **factor** representa medidas de uma **variável qualitativa**, podendo ser **nominal** ou **ordinal**

Ex.: medidas tomadas em campo ao longo de uma amostragem de 6 meses

1. Amostras: {"amostra_01", "amostra_02", "amostra_03", "amostra_04", "amostra_05"}
2. Tipo de floresta: {fechada, fechada, aberta, aberta, aberta}
3. Abundância de uma espécie: {alta, media, baixa, baixa, media}

1. Atributos dos objetos

2. Factor: homogêneo (*um modo* - sempre *numeric*), unidimensional (*uma dimensão*) e possui ainda **levels** (níveis)



1. Atributos dos objetos

2. Factor (nominal): variáveis nominais

```
# fator nominal
fa_no <- factor(x = sample(x = c("floresta", "pastagem", "cerrado"),
                           size = 20, replace = TRUE),
               levels = c("floresta", "pastagem", "cerrado"))
fa_no
```

```
## [1] floresta cerrado cerrado cerrado floresta cerrado cerrado pastagem floresta pastagem cerrado cerrado f
## [16] cerrado pastagem floresta pastagem cerrado
## Levels: floresta pastagem cerrado
```

```
levels(fa_no)
```

```
## [1] "floresta" "pastagem" "cerrado"
```

1. Atributos dos objetos

2. Factor (ordinal): variáveis ordinais

```
# fator ordinal
fa_or <- factor(x = sample(x = c("baixa", "media", "alta"),
                           size = 20, replace = TRUE),
               levels = c("baixa", "media", "alta"), ordered = TRUE)
fa_or
```

```
## [1] baixa alta media alta media alta baixa baixa alta media alta media media baixa baixa media media media a
## Levels: baixa < media < alta
```

```
levels(fa_or)
```

```
## [1] "baixa" "media" "alta"
```

1. Atributos dos objetos

2. Factor: conversão

Criar um vetor **character**

```
# conversao de fatores  
ve_ch <- c("alta", "media", "baixa", "baixa", "media")  
ve_ch
```

```
## [1] "alta" "media" "baixa" "baixa" "media"
```

```
# modo  
mode(ve_ch)
```

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

```
# classe  
class(ve_ch)
```

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

1. Atributos dos objetos

2. Factor: conversão

Forçar a ser **factor nominal**

```
# conversao de fatores  
fa_no <- as.factor(ve_ch)  
fa_no
```

```
## [1] alta  media baixa baixa media  
## Levels: alta baixa media
```

```
# niveis  
levels(fa_no)
```

```
## [1] "alta"  "baixa" "media"
```

```
# classe  
class(fa_no)
```

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


Exercícios

Exercício 06

Factor

Criem um fator chamado "tr", com dois níveis ("cont" e "trat") para descrever 100 locais de amostragem, 50 de cada tratamento. O fator deve ser dessa forma:

```
cont, cont, cont, ....., cont, trat, trat, ....., trat
```

04:00

Exercício 06

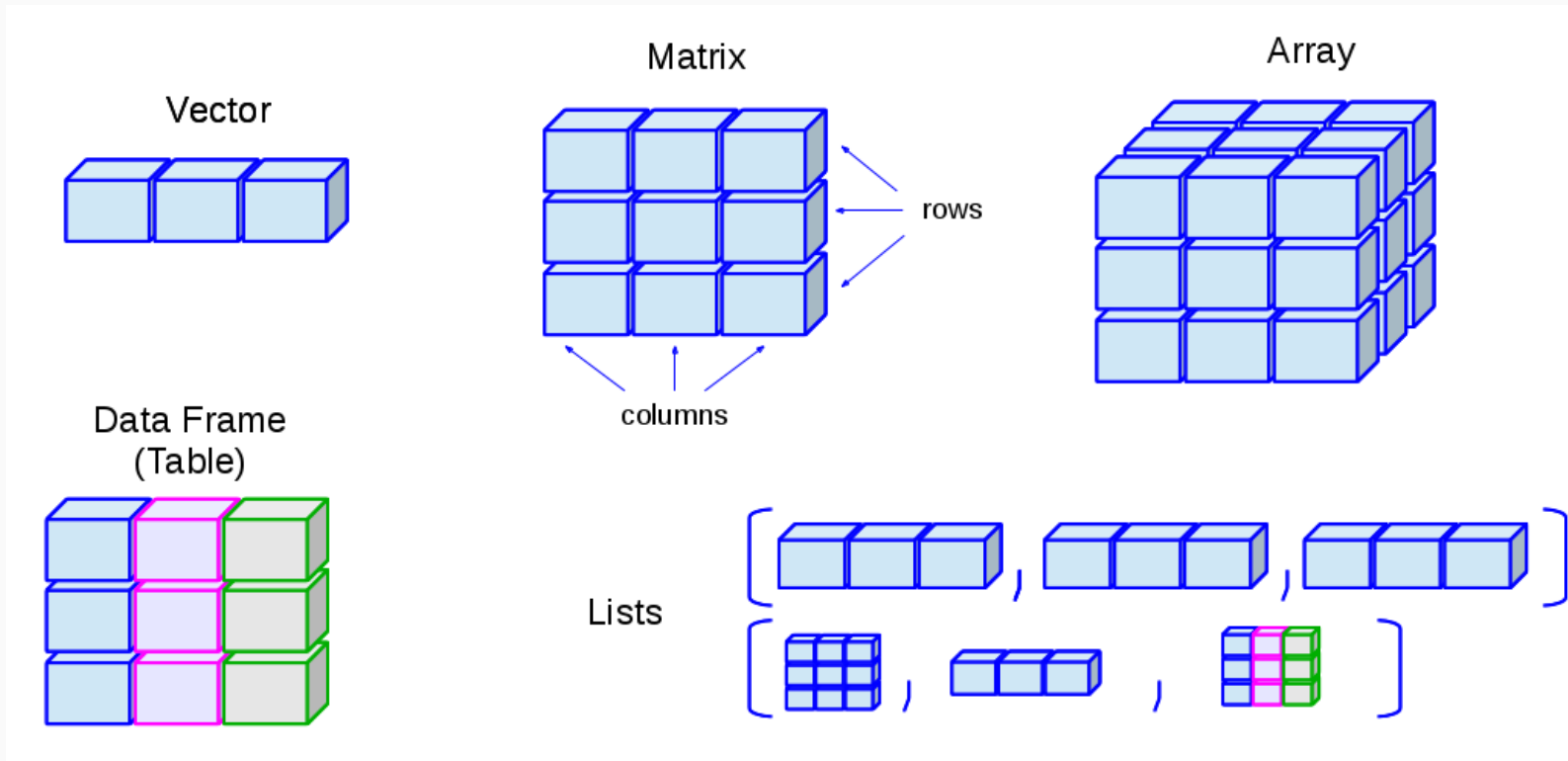
Resposta

```
# exercicio 06  
# solucao 1  
ch ← rep(c("cont", "trat"), each = 50)  
ch  
  
tr ← as.factor(ch)  
tr
```

```
# solucao 2  
tr ← as.factor(rep(c("cont", "trat"), each = 50))  
tr
```

1. Atributos dos objetos

3. Matrix



1. Atributos dos objetos

3. Matrix: homogêneo (*um modo*) e bidimensional (*duas dimensões*)

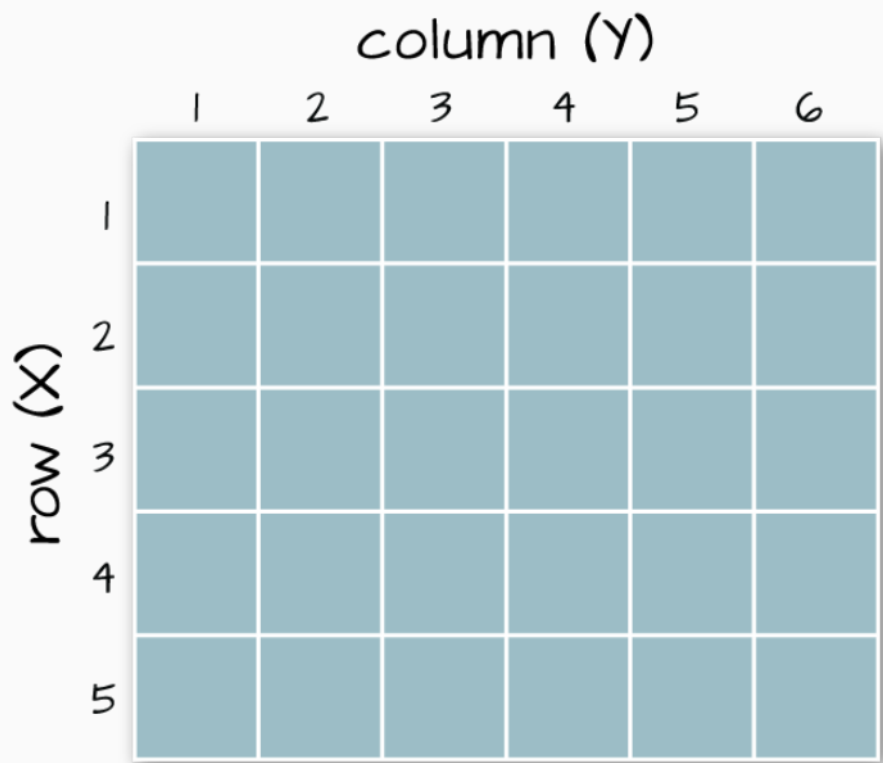
A **matrix** representa os dados no formato de **tabela**, com **linhas** e **colunas**

As **linhas** representam **unidades amostrais** (locais, transectos, parcelas) e as **colunas** representam **variáveis quantitativas** (discretas ou contínuas) ou **descrições** (informações em texto)

1. Atributos dos objetos

3. Matrix: homogêneo (*um modo*) e bidimensional (*duas dimensões*)

Ex.: cinco espécies amostradas em cinco locais



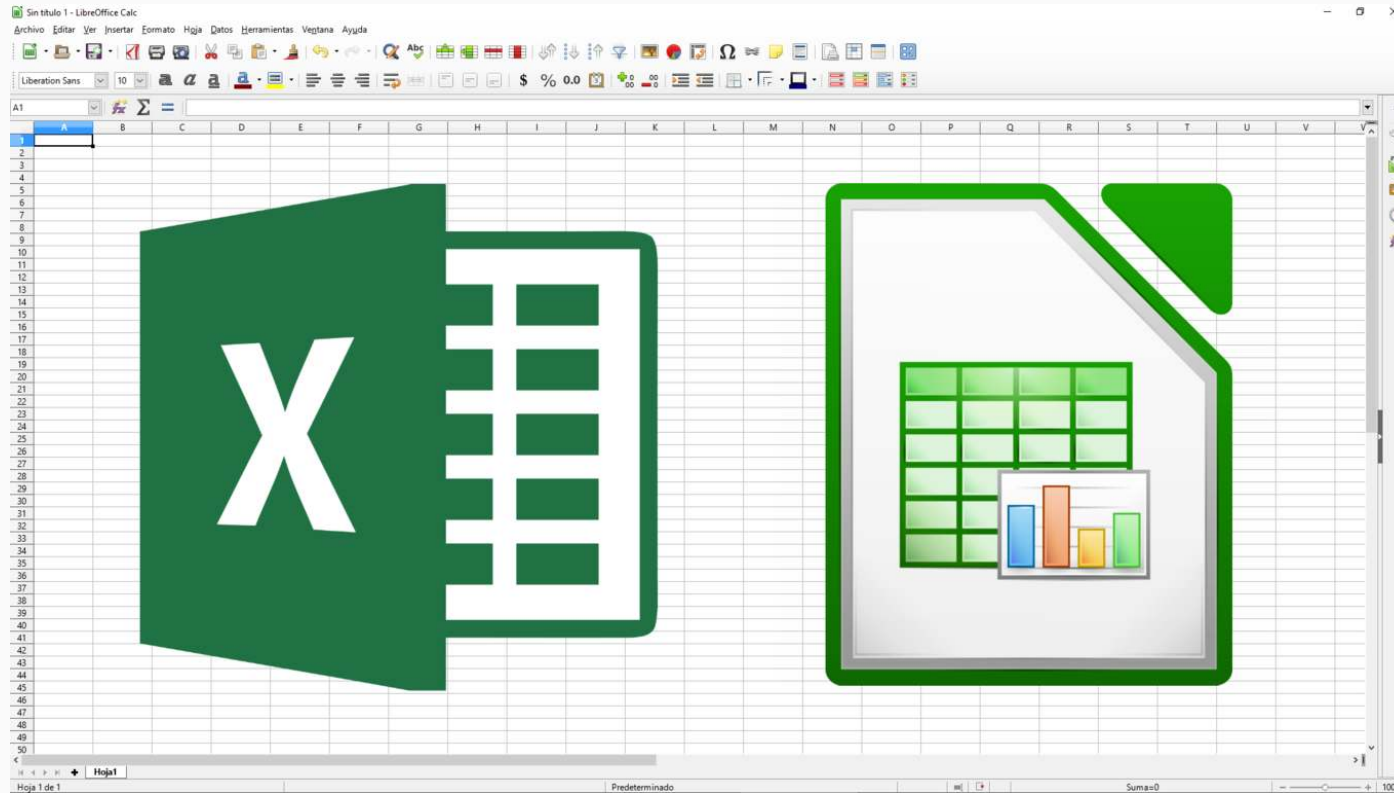
A 5x6 matrix grid with row and column labels. The columns are labeled 1 through 6, and the rows are labeled 1 through 5. The grid consists of 30 empty cells.

	1	2	3	4	5	6
1						
2						
3						
4						
5						

Esse formato lembra algo?

1. Atributos dos objetos

3. **Matrix:** planilhas eletrônicas





1. Atributos dos objetos

Há **duas formas** de se construir uma **matrix** no R:

1. Dispondo elementos

`matrix()`: dispõem um vetor em um certo número de linhas e colunas

```
# matriz - funcao matrix
# vetor
ve <- 1:12
```

```
# matrix - preenchimento por linhas - horizontal
ma_row <- matrix(data = ve, nrow = 4, ncol = 3, byrow = TRUE)
ma_row
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

1. Atributos dos objetos

Há **duas formas** de se construir uma **matrix** no R:

1. Dispondo elementos

`matrix()`: dispõem um vetor em um certo número de linhas e colunas

```
# matriz - funcao matrix  
# vetor  
ve ← 1:12
```

```
# matrix - preenchimento por colunas - vertical  
ma_col ← matrix(data = ve, nrow = 4, ncol = 3, byrow = FALSE)  
ma_col
```

```
##      [,1] [,2] [,3]  
## [1,]    1    5    9  
## [2,]    2    6   10  
## [3,]    3    7   11  
## [4,]    4    8   12
```

1. Atributos dos objetos

Há **duas formas** de se construir uma **matrix** no R:

2. Combinando vetores

`rbind()`: combina vetores por linha, i.e., vetor embaixo do outro

`cbind()`: combina vetores por coluna, i.e., vetor ao lado do outro

```
# criar dois vetores
vec_1 <- c(1, 2, 3)
vec_2 <- c(4, 5, 6)
```

```
# combinar por linhas - vertical - um embaixo do outro
ma_rbind <- rbind(vec_1, vec_2)
ma_rbind
```

```
##      [,1] [,2] [,3]
## vec_1   1   2   3
## vec_2   4   5   6
```

1. Atributos dos objetos

Há **duas formas** de se construir uma **matrix** no R:

2. Combinando vetores

`rbind()`: combina vetores por linha, i.e., vetor embaixo do outro

`cbind()`: combina vetores por coluna, i.e., vetor ao lado do outro

```
# criar dois vetores
vec_1 <- c(1, 2, 3)
vec_2 <- c(4, 5, 6)
```

```
# combinar por colunas - horizontal - um ao lado do outro
ma_cbind <- cbind(vec_1, vec_2)
ma_cbind
```

```
##      vec_1 vec_2
## [1,]     1     4
## [2,]     2     5
## [3,]     3     6
```

Exercícios

Exercício 07

Matrix

Criem uma matriz chamada "ma", resultante da disposição de um vetor composto por 10000 valores aleatórios entre 0 e 10. A matriz deve conter 100 linhas e ser disposta por colunas

04:00

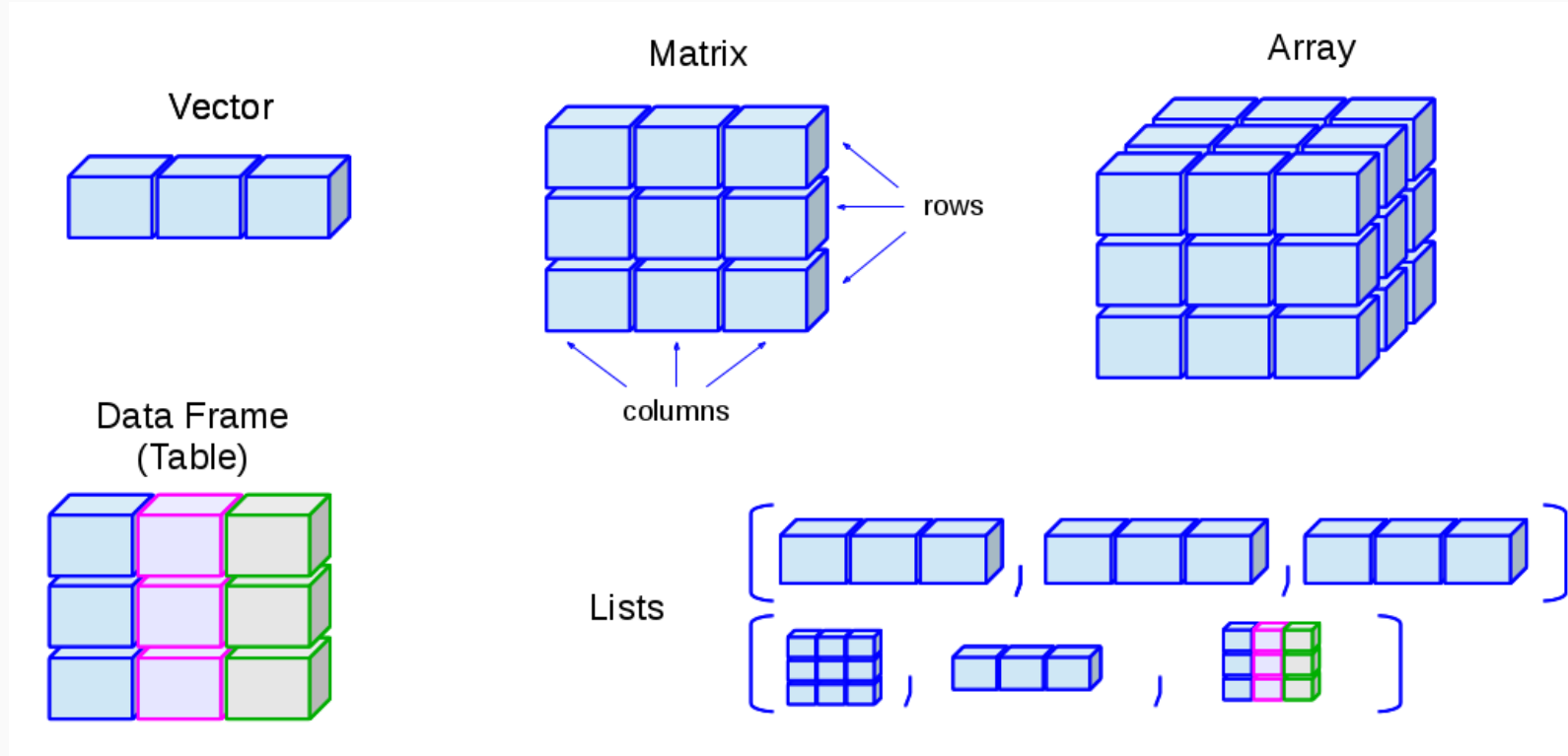
Exercício 07

Resposta

```
# exercicio 07  
ma ← matrix(sample(0:10, 10000, rep = TRUE), nrow = 100, byrow = FALSE)  
ma
```


1. Atributos dos objetos

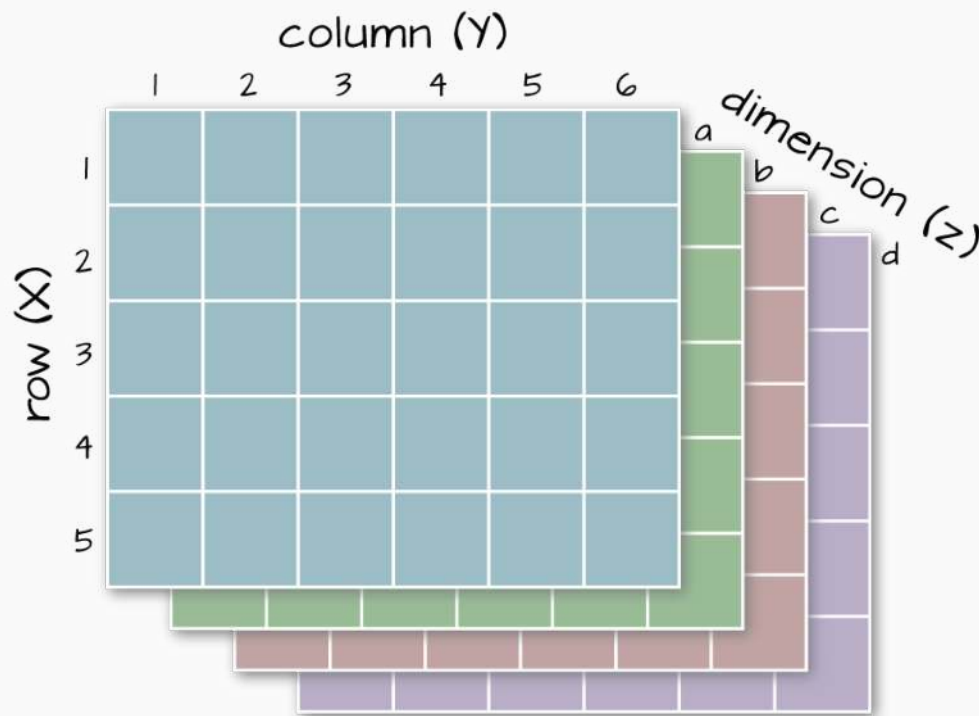
4. Array



1. Atributos dos objetos

4. Array: homogêneo (*um modo*) e multidimensional (*mais de duas dimensões*)

O **array** representa combinação de **tabelas**, com **linhas**, **colunas** e **dimensões**





1. Atributos dos objetos

Há **uma forma** de se construir um **array** no R:

1. Dispondo elementos em dimensões

`array()`: dispõem um vetor em um certo número de linhas, colunas e dimensões....

```
# vetor  
ve ← 1:8  
ve
```

```
## [1] 1 2 3 4 5 6 7 8
```

1. Atributos dos objetos

Há **uma forma** de se construir um **array** no R:

1. Dispondo elementos em dimensões

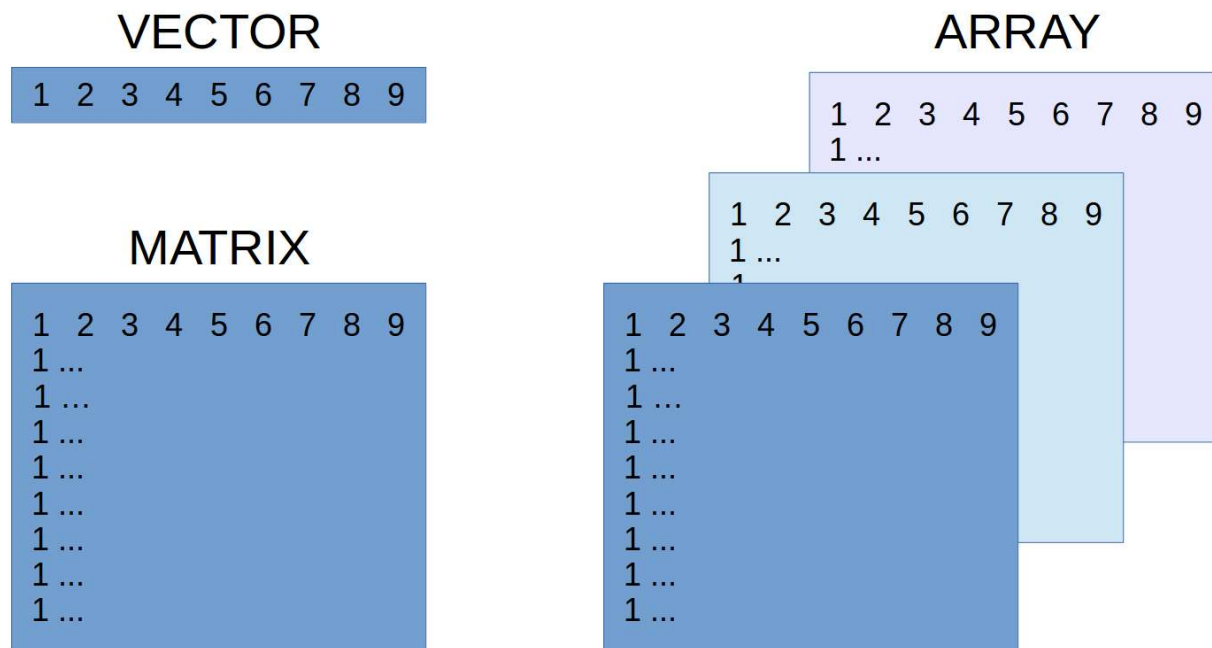
`array`: dispõem um vetor em um certo número de linhas, colunas e dimensões....

```
# array
ar <- array(data = ve, dim = c(2, 2, 2))
ar
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

1. Atributos dos objetos

Até o momento vimos **estruturas homogêneas**



1. Atributos dos objetos

Agora veremos as **estruturas heterogêneas**

HOMOGENEOUS
(elements are only 1 type)

Vector

Matrix

Array

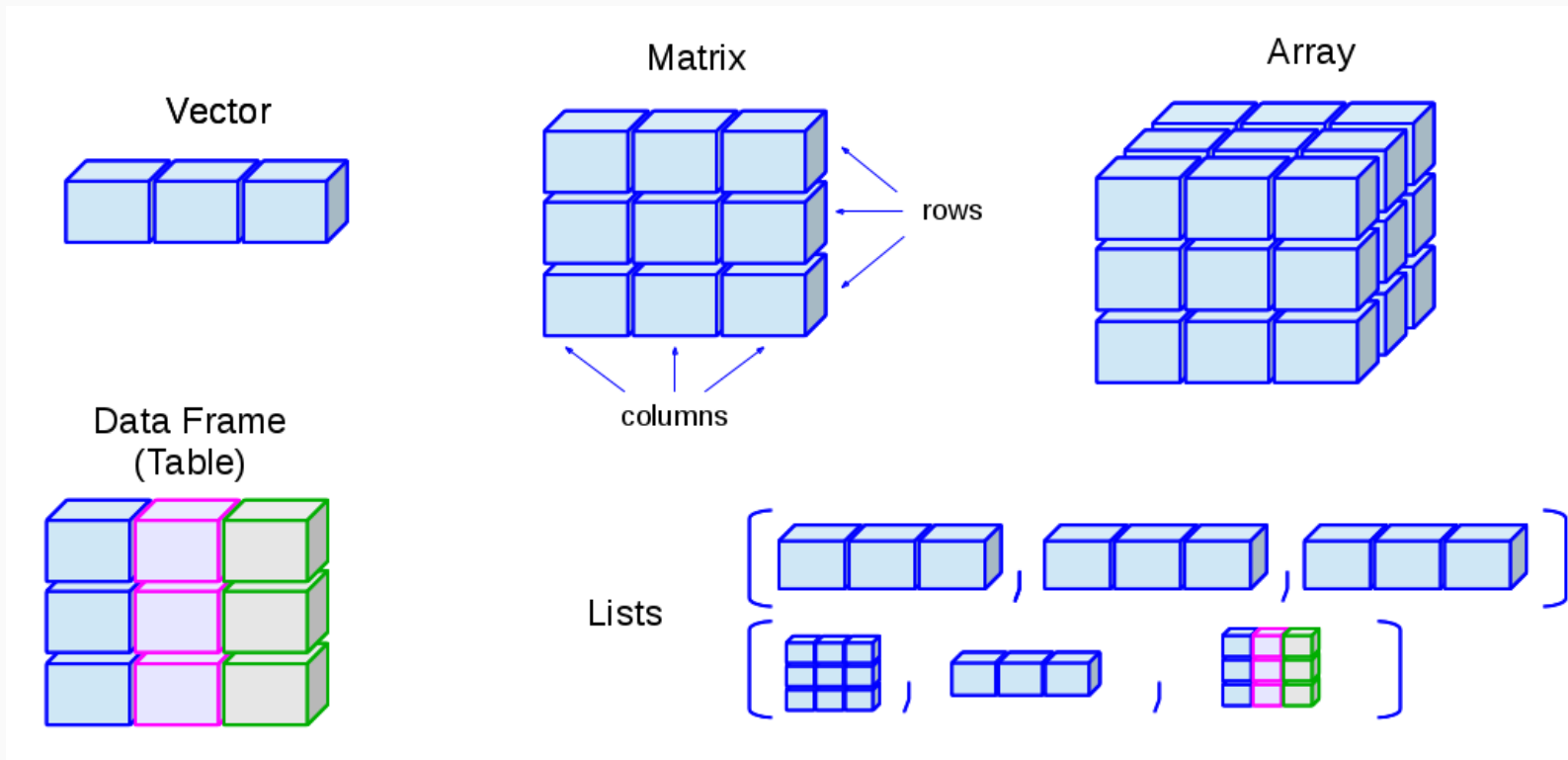
HETEROGENEOUS
(elements can be different)

Dataframe

List

1. Atributos dos objetos

5. Data frame



1. Atributos dos objetos

5. Data frame: heterogêneo (*mais de um modo*) e bidimensional (*duas dimensões*)

O **data frame** representa dados no formato de **tabela**, com **linhas** e **colunas**

As **linhas** representam **unidades amostrais** (locais, transectos, parcelas) e as **colunas** representam **descrições** (informações em texto), **variáveis quantitativas** (discretas ou contínuas) e/ou **variáveis qualitativas** (nominais ou ordinais)

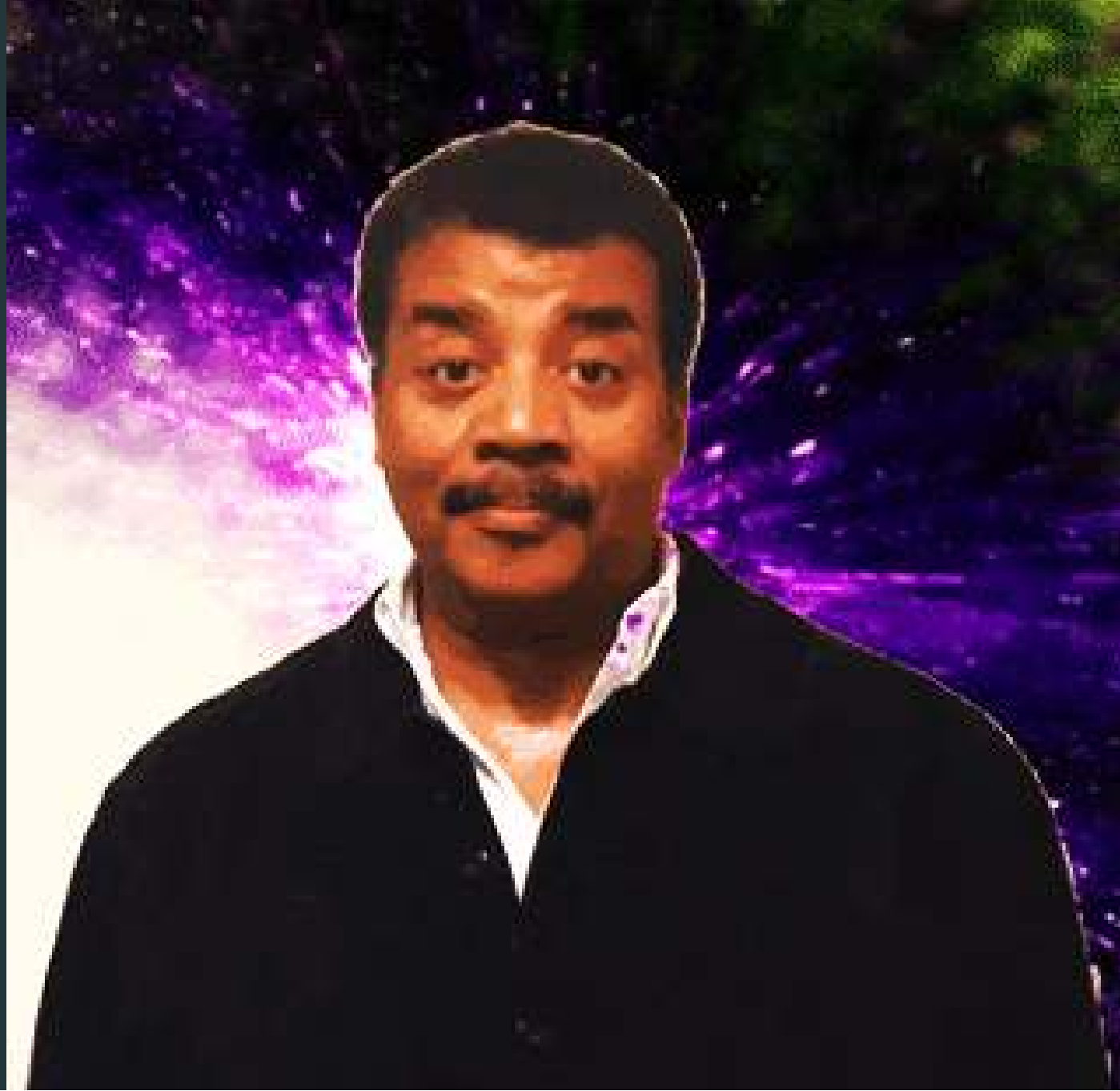
Esse formato também lembra algo?

1. Atributos dos objetos

5. Data frame: planilhas eletrônicas



Esse é justamente o formato de entrada dos dados de planilhas eletrônicas!



1. Atributos dos objetos

Há **uma forma** de se construir um **data frame** no R:

1. Combinando vetores horizontalmente

`data.frame()`: combina vetores horizontalmente, um ao lado do outro. Semelhante à função `cbind()`

```
# criar tres vetores
vec_ch <- c("sp1", "sp2", "sp3")
vec_nu <- c(4, 5, 6)
vec_fa <- factor(c("campo", "floresta", "floresta"))
```

```
# data.frame - combinar por colunas - horizontal - um ao lado do outro
df <- data.frame(vec_ch, vec_nu, vec_fa)
df
```

```
##   vec_ch vec_nu  vec_fa
## 1   sp1      4   campo
## 2   sp2      5 floresta
## 3   sp3      6 floresta
```

1. Atributos dos objetos

Há **uma forma** de se construir um **data frame** no R:

1. Combinando vetores horizontalmente

Também podemos informar o nome das colunas

```
# data frame
df <- data.frame(especies = vec_ch,
                 abundancia = vec_nu,
                 vegetacao = vec_fa)

df
```

```
##   especies abundancia vegetacao
## 1      sp1          4      campo
## 2      sp2          5  floresta
## 3      sp3          6  floresta
```

1. Atributos dos objetos

data frame vs cbind

Criação dos vetores

```
# vetores
pa <- paste("parcela", 1:4, sep = "_")
pa
```

```
## [1] "parcela_1" "parcela_2" "parcela_3" "parcela_4"
```

```
pe <- sample(0:1, 4, rep = TRUE)
pe
```

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

```
tr <- factor(rep(c("trat", "cont"), each = 2))
tr
```

```
## [1] trat trat cont cont
## Levels: cont trat
```


1. Atributos dos objetos

Qual a diferença?

```
# uniao de vetores  
df ← data.frame(pa, pe, tr)  
df
```

```
##           pa pe  tr  
## 1 parcela_1  0 trat  
## 2 parcela_2  1 trat  
## 3 parcela_3  1 cont  
## 4 parcela_4  0 cont
```

```
# estrutura  
str(df)
```

```
## 'data.frame':    4 obs. of  3 variables:  
## $ pa: chr  "parcela_1" "parcela_2" "parcela_3" "parcela_4"  
## $ pe: int   0  1  1  0  
## $ tr: Factor w/ 2 levels "cont","trat": 2 2 1 1
```

1. Atributos dos objetos

Qual a diferença?

```
# uniao de vetores
df_c <- cbind(pa, pe, tr)
df_c
```

```
##      pa      pe  tr
## [1,] "parcela_1" "0" "2"
## [2,] "parcela_2" "1" "2"
## [3,] "parcela_3" "1" "1"
## [4,] "parcela_4" "0" "1"
```

```
# estrutura
str(df_c)
```

```
##  chr [1:4, 1:3] "parcela_1" "parcela_2" "parcela_3" "parcela_4" "0" "1" "1" "0" "2" "2" "1" "1"
##  - attr(*, "dimnames")=List of 2
##    ..$ : NULL
##    ..$ : chr [1:3] "pa" "pe" "tr"
```

Exercícios

Exercício 08

Data frame

Criem um data frame "df", resultante da composição desses vetores:

id: 1:50

sp: sp01, sp02, ... , sp49, sp50

ab: 50 valores aleatórios entre 0 a 5

05:00

Exercício 08

Resposta

```
# exercicio 08
id <- 1:50
id

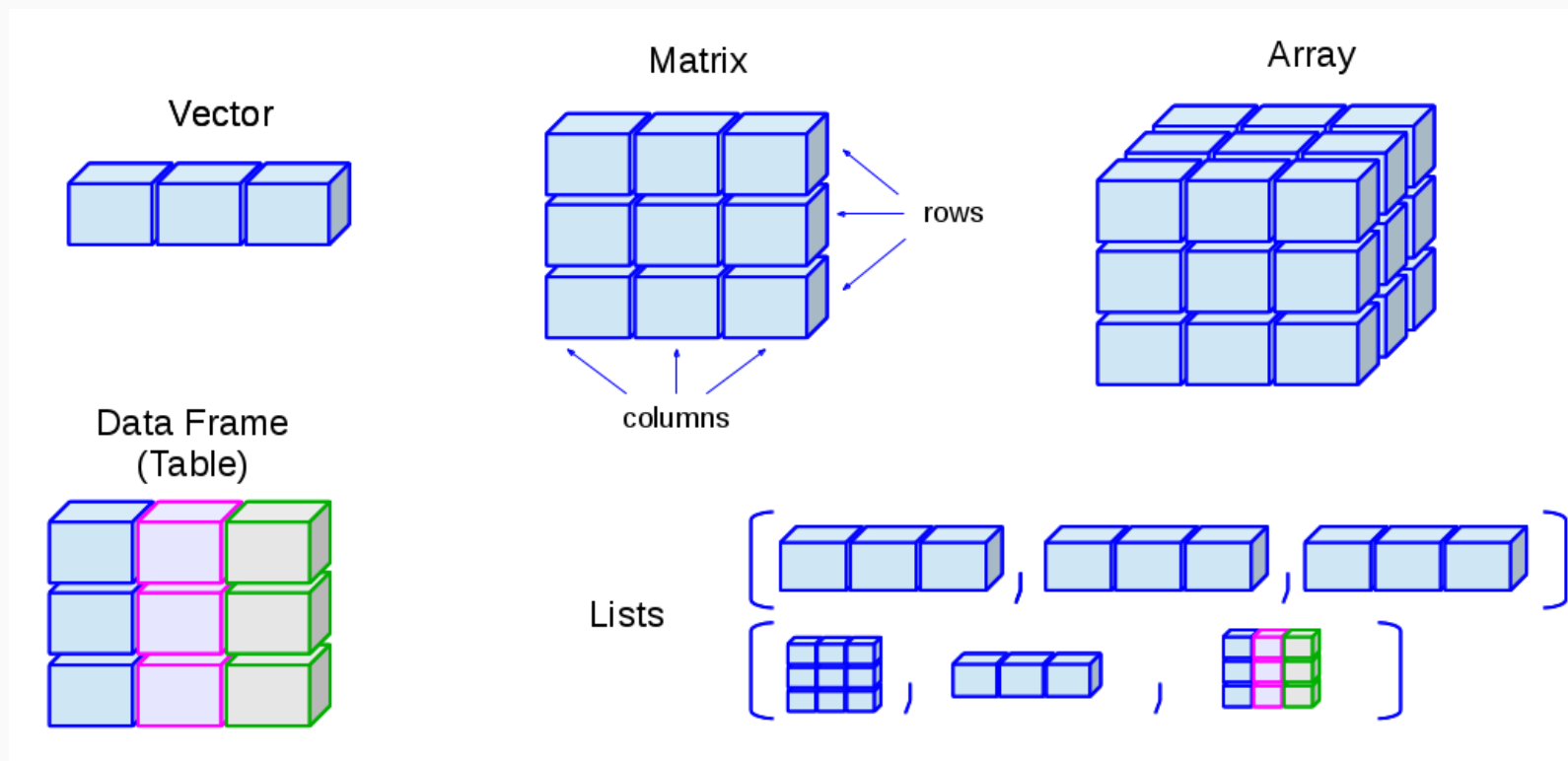
sp <- c(paste("sp", 1:9, sep = "0"), paste("sp", 10:50, sep = ""))
sp

ab <- sample(0:5, 50, replace = TRUE)
ab

df <- data.frame(id, sp, ab)
df
```

1. Atributos dos objetos

6. List



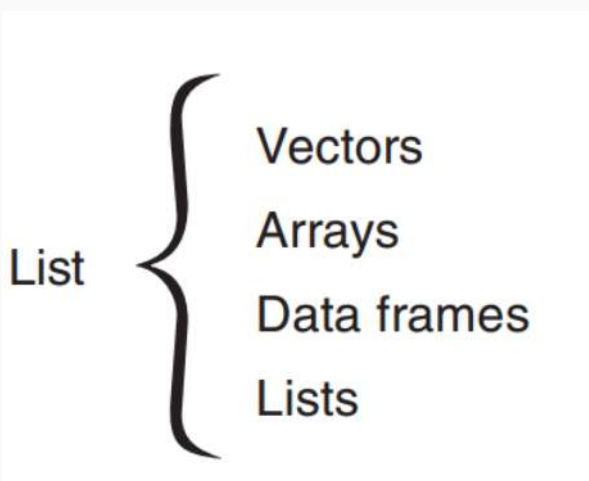
1. Atributos dos objetos

6. List: heterogêneo (*mais de um modo*) e unidimensional (*uma dimensão*)

Tipo **especial de vetor** que aceita **objetos** como **elementos**

Estrutura de dados utilizado para **agrupar objetos**

É a **saída** de muitas funções que fazem **análises estatísticas**



1. Atributos dos objetos

Há **uma forma** de se construir um **listas** no R:

1. Combinando objetos horizontalmente

`list()`: combina objetos horizontalmente, semelhante à função `c()`

```
# lista
li <- list(rep(1, 20), # vector
           factor(1, 1), # factor
           cbind(c(1, 2), c(1, 2))) # matrix
li
```

```
## [[1]]
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## [[2]]
## [1] 1
## Levels: 1
##
## [[3]]
##      [,1] [,2]
## [1,]    1    1
```


1. Atributos dos objetos

Há **uma forma** de se construir um **listas** no R:

1. Combinando objetos horizontalmente

Também podemos **nomear** os elementos

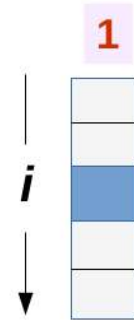
```
# lista com nomes
li <- list(vector = rep(1, 20), # vector
          factor = factor(1, 1), # factor
          matrix = cbind(c(1, 2), c(1, 2))) # matrix
li
```

```
## $vector
## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## $factor
## [1] 1
## Levels: 1
##
## $matrix
##      [,1] [,2]
## [1,]    1    1
```

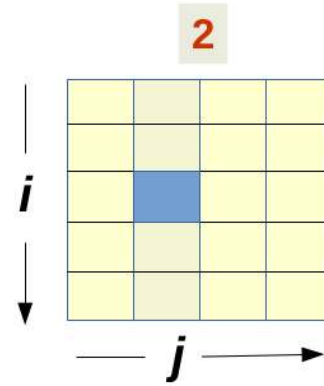
Dúvidas?

Bora manejar isso tudo?

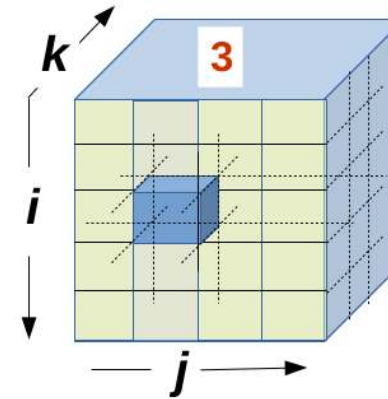
vector



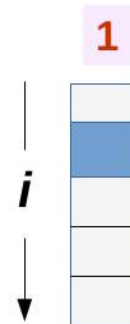
matrix



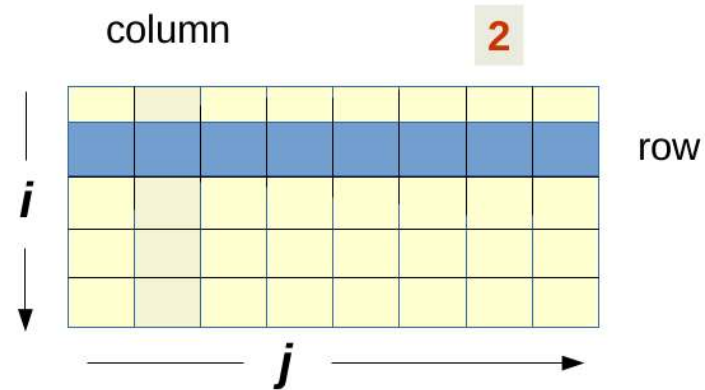
array (dim = 3)



list



data frame



2. Manipulação de dados (1D)

Vetor e Fator

1. Indexação []: acessa elementos de vetores e fatores

```
# indexacao []  
  
# fixar a amostragem  
set.seed(42)  
  
# amostrar 10 elementos de uma sequencia  
ve ← sample(x = seq(0, 2, .05), size = 10)  
ve
```

```
## [1] 1.80 0.00 1.20 0.45 1.75 0.85 1.15 0.30 1.90 0.20
```

2. Manipulação de dados (1D)

Vetor e Fator

1. Indexação []: acessa elementos de vetores e fatores

Selecionar elementos

```
# seleciona o quinto elemento  
ve[5]
```

```
## [1] 1.75
```

```
# seleciona os elementos de 1 a 5  
ve[1:5]
```

```
## [1] 1.80 0.00 1.20 0.45 1.75
```

```
# seleciona os elementos 1 e 10 e atribui  
ve_sel ← ve[c(1, 10)]  
ve_sel
```

2. Manipulação de dados (1D)

Vetor e Fator

1. Indexação []: acessa elementos de vetores e fatores

Retirar elementos

```
# retira o decimo elemento  
ve[-10]
```

```
## [1] 1.80 0.00 1.20 0.45 1.75 0.85 1.15 0.30 1.90
```

```
# retira os elementos 2 a 9  
ve[-(2:9)]
```

```
## [1] 1.8 0.2
```

```
# retira os elementos 5 e 10 e atribui  
ve_sub <- ve[-c(5, 10)]  
ve_sub
```

2. Manipulação de dados (1D)

Vetor e Fator

2. Seleção condicional: selecionar elementos por condições

```
# dois vetores  
foo ← 42  
bar ← 23
```

```
# operadores relacionais - saidas booleanas (TRUE ou FALSE)  
foo = bar # igualdade  
foo ≠ bar # diferenca  
foo > bar # maior  
foo ≥ bar # maior ou igual  
foo < bar # menor  
foo ≤ bar # menor ou igual
```


2. Manipulação de dados (1D)

Vetor e Fator

2. Seleção condicional: selecionar elementos por condições

```
# quais valores são maiores que 1?  
ve > 1
```

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

```
# valores acima de 1  
ve[ve > 1]
```

```
## [1] 1.80 1.20 1.75 1.15 1.90
```

```
# atribuir valores maiores que 1  
ve_maior1 <- ve[ve > 1]  
ve_maior1
```

```
## [1] 1.80 1.20 1.75 1.15 1.90
```

2. Manipulação de dados (1D)

Vetor e Fator

3. Funções de manipulação: `max()`, `min()`, `range()`, `length()`, `sort()` e `round()`

```
# maximo  
max(ve)
```

```
## [1] 1.9
```

```
# minimo  
min(ve)
```

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

2. Manipulação de dados (1D)

Vetor e Fator

3. Funções de manipulação: `max()`, `min()`, `range()`, `length()`, `sort()` e `round()`

```
# amplitude  
range(ve)
```

```
## [1] 0.0 1.9
```

```
# comprimento  
length(ve)
```

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

2. Manipulação de dados (1D)

Vetor e Fator

3. Funções de manipulação: `max()`, `min()`, `range()`, `length()`, `sort()` e `round()`

```
# ordenar crescente  
sort(ve)
```

```
## [1] 0.00 0.20 0.30 0.45 0.85 1.15 1.20 1.75 1.80 1.90
```

```
# ordenar decrescente  
sort(ve, dec = TRUE)
```

```
## [1] 1.90 1.80 1.75 1.20 1.15 0.85 0.45 0.30 0.20 0.00
```

2. Manipulação de dados (1D)

Vetor e Fator

3. Funções de manipulação: `max()`, `min()`, `range()`, `length()`, `sort()` e `round()`

```
# arredondamento  
ve
```

```
## [1] 1.80 0.00 1.20 0.45 1.75 0.85 1.15 0.30 1.90 0.20
```

```
# arredondamento  
round(ve, digits = 1)
```

```
## [1] 1.8 0.0 1.2 0.4 1.8 0.9 1.2 0.3 1.9 0.2
```

```
# arredondamento  
round(ve, digits = 0)
```

```
## [1] 2 0 1 0 2 1 1 0 2 0
```

2. Manipulação de dados (1D)

Vetor e Fator

3. Funções de manipulação: `any()`, `all()` e `which()`

```
# algum?  
any(ve > 1)
```

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

```
# todos?  
all(ve > 1)
```

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

```
# qual(is)?  
which(ve > 1)
```

```
## [1] 1 3 5 7 9
```

2. Manipulação de dados (1D)

Vetor e Fator

3. Funções de manipulação: `subset()` e `ifelse()`

```
# subconjunto  
subset(ve, ve > 1)
```

```
## [1] 1.80 1.20 1.75 1.15 1.90
```

```
# condicao para uma operacao  
ifelse(ve > 1, 1, 0)
```

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

2. Manipulação de dados (1D)

Listas

1. Indexação []: acessa elementos de listas

```
## indexacao []  
# lista  
li ← list(elem1 = 1, elem2 = 2, elem3 = 3)  
li
```

```
## $elem1  
## [1] 1  
##  
## $elem2  
## [1] 2  
##  
## $elem3  
## [1] 3
```


2. Manipulação de dados (1D)

Listas

1. Indexação []: acessa elementos de listas

Selecionar elementos

```
# acessar o primeiro elemento  
li[1]
```

```
## $elem1  
## [1] 1
```

```
# acessar o primeiro e o terceiro elementos e atribuir  
li2 ← li[c(1, 3)]  
li2
```

```
## $elem1  
## [1] 1  
##  
## $elem3  
## [1] 3
```

2. Manipulação de dados (1D)

Listas

1. Indexação []: acessa elementos de listas

Retirar elementos

```
# retirar o primeiro elemento  
li[-1]
```

```
## $elem2  
## [1] 2  
##  
## $elem3  
## [1] 3
```

```
# retirar o segundo elemento e atribuir  
li_13 ← li[-2]  
li_13
```

```
## $elem1  
## [1] 1
```

2. Manipulação de dados (1D)

Listas

2. Indexação [[]]: acessa valores dos elementos de listas

Retirar elementos

```
# valor do primeiro elemento  
li[[1]]
```

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

```
# valor do segundo elemento e atribuir  
li2_val ← li[[2]]  
li2_val
```

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

2. Manipulação de dados (1D)

Listas

3. Indexação \$: acessa elementos pelo nome

Selecionar elementos

```
# acessar o primeiro elemento  
li$elem1
```

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

```
# acessar o primeiro e o terceiro elementos e atribuir  
li1 ← li$elem1  
li1
```

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

2. Manipulação de dados (1D)

Listas

4. Funções: `length()` e `names()`

```
# comprimento  
length(li)
```

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

```
# names  
names(li)
```

```
## [1] "elem1" "elem2" "elem3"
```

2. Manipulação de dados (1D)

Listas

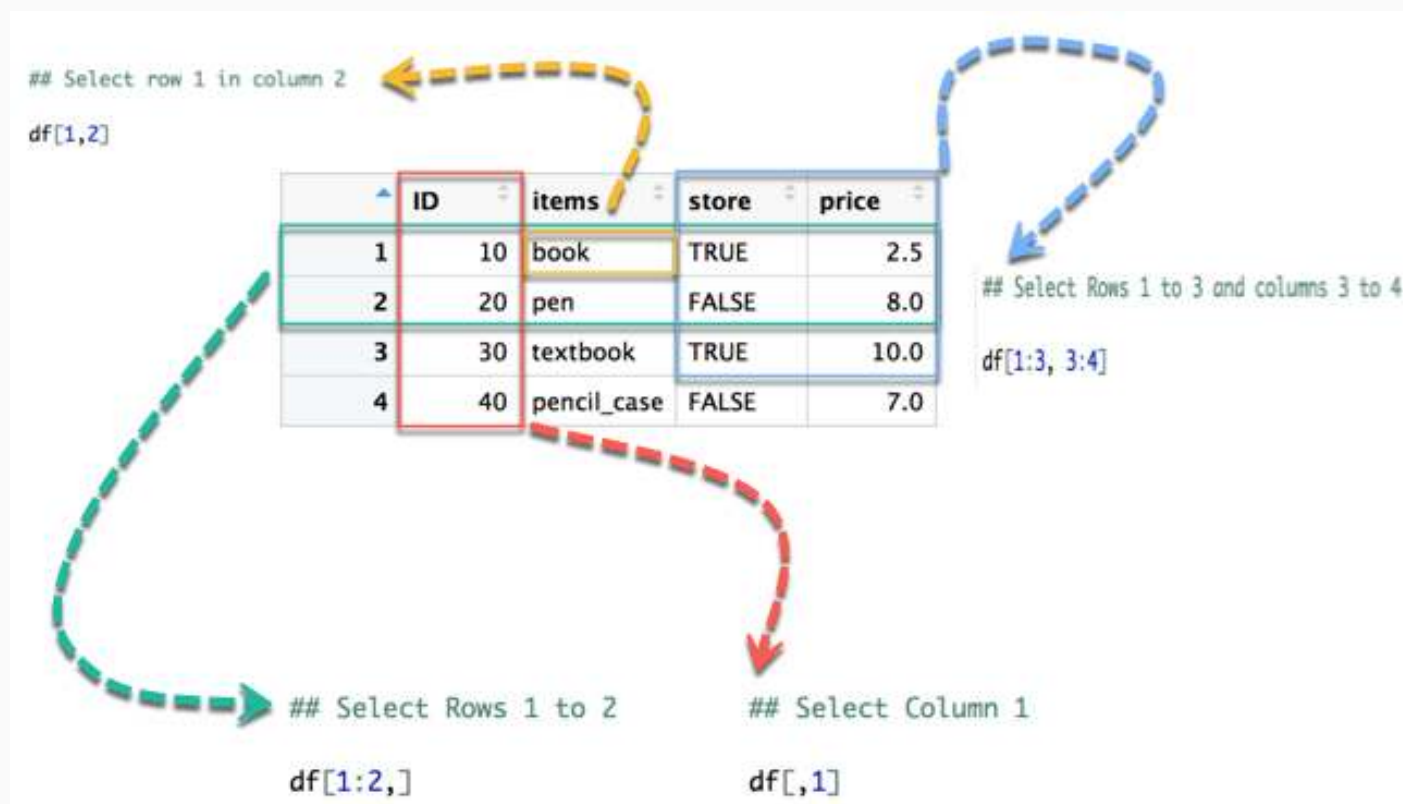
4. Funções: `length()` e `names()`

```
# renomear  
names(li) ← paste0("elemento0", 1:3)  
li
```

```
## $elemento01  
## [1] 1  
##  
## $elemento02  
## [1] 2  
##  
## $elemento03  
## [1] 3
```

3. Manipulação de dados (2D ou nD)

Matrizes, Arrays e Data Frames



3. Manipulação de dados (2D ou nD)

Matrizes, Arrays e Data Frames

1. Indexação []: acessa elementos de matrizes, arrays e data frames

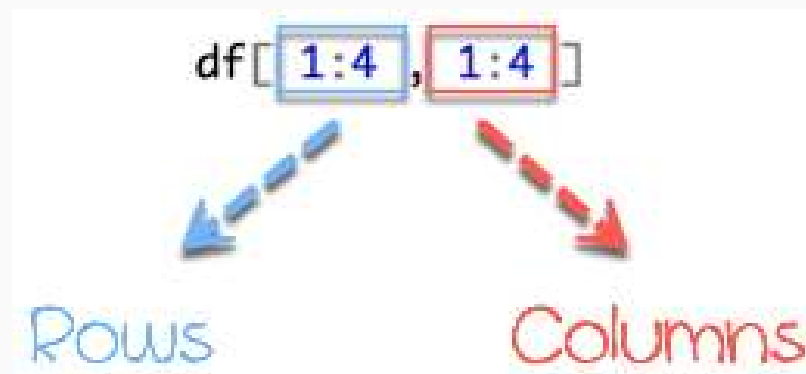
```
# matriz  
ma <- matrix(1:12, 4, 3)  
ma
```

```
##      [,1] [,2] [,3]  
## [1,]    1    5    9  
## [2,]    2    6   10  
## [3,]    3    7   11  
## [4,]    4    8   12
```


3. Manipulação de dados (2D ou nD)

Matrizes, Arrays e Data Frames

1. Indexação []: acessa elementos de matrizes, arrays e data frames



3. Manipulação de dados (2D ou nD)

Matrizes, Arrays e Data Frames

1. Indexação []: acessa elementos de matrizes, arrays e data frames

```
# linha 3  
ma[3, ]
```

```
## [1]  3  7 11
```

```
# coluna 2  
ma[, 2]
```

```
## [1] 5 6 7 8
```

```
# elemento da linha 1 e coluna 2  
ma[1, 2]
```

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

```
# elementos da linha 1 e coluna 1 e 2  
ma[1, 1:2]
```

3. Manipulação de dados (2D ou nD)

Matrizes, Arrays e Data Frames

1. Indexação []: acessa elementos de matrizes, arrays e data frames

```
# elementos da linha 1 e coluna 1 e 3  
ma[1, c(1, 3)]
```

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

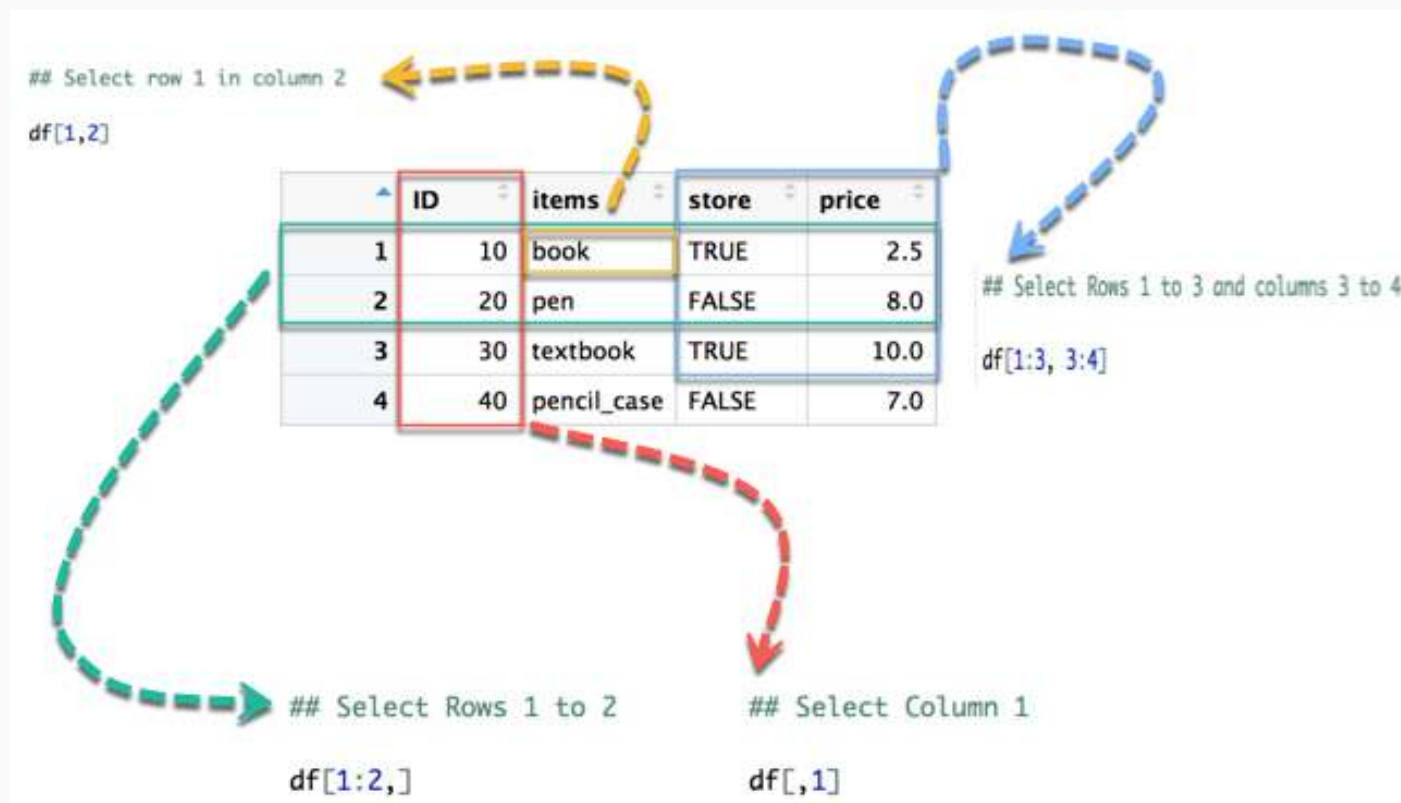
```
# elementos da linha 1 e coluna 1 e 3 e atribuir  
ma_sel ← ma[1, c(1, 3)]  
ma_sel
```

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

3. Manipulação de dados (2D ou nD)

Matrizes, Arrays e Data Frames

1. Indexação []: acessa elementos de matrizes, arrays e data frames



3. Manipulação de dados (2D ou nD)

Data Frames

2. Indexação \$: acessa elementos de data frames

```
# criar tres vetores
sp <- paste("sp", 1:10, sep = "")
abu <- 1:10
flo <- factor(rep(c("campo", "floresta"), each = 5))
```

```
# data frame
df <- data.frame(sp, abu, flo)
df
```

```
##      sp abu   flo
## 1  sp1   1  campo
## 2  sp2   2  campo
## 3  sp3   3  campo
## 4  sp4   4  campo
## 5  sp5   5  campo
## 6  sp6   6 floresta
## 7  sp7   7 floresta
```

3. Manipulação de dados (2D ou nD)

Data Frames

2. Indexação \$: acessa colunas de data frames

```
# $ funciona apenas para data frame  
df$sp
```

```
## [1] "sp1" "sp2" "sp3" "sp4" "sp5" "sp6" "sp7" "sp8" "sp9" "sp10"
```

```
df$abu
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
df$flo
```

```
## [1] campo campo campo campo campo floresta floresta floresta floresta floresta  
## Levels: campo floresta
```

3. Manipulação de dados (2D ou nD)

Data Frames

2. Indexação \$: acessa colunas de data frames

```
# funcoes para uma coluna indexada por $  
length(df$abu)
```

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

```
max(df$abu)
```

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

```
min(df$abu)
```

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

```
range(df$abu)
```

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

3. Manipulação de dados (2D ou nD)

Data Frames

3. Indexação \$ e mudanças de colunas: acessa e modifica colunas em data frames

```
# modo  
mode(df$abu)
```

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

```
# converter colunas para caracter  
df$abu ← as.character(df$abu)
```

```
# modo  
df$abu
```

```
## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"
```

```
mode(df$abu)
```

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


3. Manipulação de dados (2D ou nD)

Data Frames

3. Indexação \$ e mudanças de colunas: acessa e modifica colunas em data frames

```
# converter colunas para numerico  
df$abu ← as.numeric(df$abu)
```

```
# modo  
df$abu
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
mode(df$abu)
```

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

3. Manipulação de dados (2D ou nD)

Data Frames

4. Indexação \$ e adicionar colunas: acessa e adiciona colunas em data frames

```
# adicionar coluna
set.seed(42)
df$abu2 ← sample(0:1, nrow(df), rep = TRUE)
```

```
df$abu2
```

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

```
df
```

```
##      sp abu      flo abu2
## 1  sp1   1  campo     0
## 2  sp2   2  campo     0
## 3  sp3   3  campo     0
## 4  sp4   4  campo     0
## 5  sp5   5  campo     1
## 6  sp6   6 floresta    1
```

3. Manipulação de dados (2D ou nD)

Data Frames

5. Seleção condicional: filtro de linhas

Selecionar linhas = filtro da planilha eletrônica

```
# selecionar linhas de uma matriz ou data frame  
df[df$abu > 4, ]
```

```
##      sp abu      flo abu2  
## 5   sp5   5      campo   1  
## 6   sp6   6 floresta   1  
## 7   sp7   7 floresta   1  
## 8   sp8   8 floresta   1  
## 9   sp9   9 floresta   0  
## 10  sp10  10 floresta   1
```

3. Manipulação de dados (2D ou nD)

Data Frames

5. Seleção condicional: filtro de linhas

Selecionar linhas = filtro da planilha eletrônica

```
# selecionar linhas de uma matriz ou data frame  
df[df$abu2 == 0, ]
```

```
##      sp abu      flo abu2  
## 1 sp1    1   campo    0  
## 2 sp2    2   campo    0  
## 3 sp3    3   campo    0  
## 4 sp4    4   campo    0  
## 9 sp9    9 floresta    0
```

3. Manipulação de dados (2D ou nD)

Data Frames

5. Seleção condicional: filtro de linhas

Selecionar linhas = filtro da planilha eletrônica

```
# selecionar linhas de uma matriz ou data frame  
df[df$flo == "floresta", ]
```

```
##      sp abu      flo abu2  
## 6   sp6   6 floresta   1  
## 7   sp7   7 floresta   1  
## 8   sp8   8 floresta   1  
## 9   sp9   9 floresta   0  
## 10 sp10  10 floresta   1
```

3. Manipulação de dados (2D ou nD)

Matrizes, Arrays e Data Frames

6. Funções de conferência e manipulação

`head()`: mostra as primeiras 6 linhas

`tail()`: mostra as últimas 6 linhas

`nrow()`: mostra o número de linhas

`ncol()`: mostra o número de colunas

`dim()`: mostra o número de linhas e de colunas

`rownames()`: mostra os nomes das linhas (locais)

`colnames()`: mostra os nomes das colunas (variáveis)

`str()`: mostra as classes de cada coluna (estrutura)

`summary()`: mostra um resumo dos valores de cada coluna

`rowSums()`: calcula a soma das linhas (horizontal)

`colSums()`: calcula a soma das colunas (vertical)

`rowMeans()`: calcula a média das linhas (horizontal)

`colMeans()`: calcula a média das colunas (vertical)

4. Valores faltantes e especiais

São **valores reservados** que representam *dados faltantes, indefinições matemáticas, infinitos e objetos nulos*

1. NA (Not Available)

2. NaN (Not a Number)

3. Inf (Infinito)

4. NULL

4. Valores faltantes e especiais

1. NA (Not Available)

Significa dado faltante/indisponível

NA deve ser maiúsculo

```
# na - not available  
foo_na <- NA  
foo_na
```

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


4. Valores faltantes e especiais

1. NA (Not Available)

Criar um data frame com NA

```
# data frame  
df <- data.frame(var1 = c(1, 4, 2, NA), var2 = c(1, 4, 5, 2))  
df
```

```
##   var1 var2  
## 1    1    1  
## 2    4    4  
## 3    2    5  
## 4   NA    2
```

4. Valores faltantes e especiais

1. NA (Not Available)

Função para verificar a **presença/ausência** de NA's

```
# possui nas?  
is.na(df)
```

```
##      var1  var2  
## [1,] FALSE FALSE  
## [2,] FALSE FALSE  
## [3,] FALSE FALSE  
## [4,]  TRUE FALSE
```

Função para verificar a **presença de algum** NA's

```
# algum é na?  
any(is.na(df))
```

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

4. Valores faltantes e especiais

1. NA (Not Available)

Vamos retirar as linhas que possuem NA's

```
# retirar linhas com na
df_sem_na <- na.omit(df)
df_sem_na
```

```
##   var1 var2
## 1    1    1
## 2    4    4
## 3    2    5
```

```
# numero de linhas com e sem na
nrow(df)
```

```
## [1] 4
```

```
nrow(df_sem_na)
```

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

4. Valores faltantes e especiais

1. NA (Not Available)

Vamos substituir os NA's por 0

```
# substituir na por 0
df[is.na(df)] ← 0
df
```

```
##   var1 var2
## 1    1    1
## 2    4    4
## 3    2    5
## 4    0    2
```

4. Valores faltantes e especiais

2. NaN (Not a Number)

Representa indefinições matemáticas como $0/0$ e $\log(-1)$

```
# nan - not a number  
0/0
```

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

```
# nan - not a number  
log(-1)
```

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

4. Valores faltantes e especiais

2. NaN (Not a Number)

Um **NaN** é um **NA**, mas o **NA** não é um **NaN**

```
# criar um vetor
ve ← c(1, 2, 3, NA, NaN)
ve
```

```
## [1] 1 2 3 NA NaN
```

```
# verificar a presença de na
is.na(ve)
```

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

```
# verificar a presença de nan
is.nan(ve)
```

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

4. Valores faltantes e especiais

3. Inf (Infinito)

É um número muito grande ou um limite matemático, e.g., 10^{310} e $1/0$

```
# limite matematico  
1/0
```

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

```
# numero muito grande  
10^310
```

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

4. Valores faltantes e especiais

4. NULL

Representa um objeto nulo

Útil para preenchimento de laços e outras aplicações de programação

```
# objeto nulo  
nulo ← NULL  
nulo
```

```
## NULL
```


5. Diretório de trabalho

Endereço da pasta onde o R irá **importar e exportar** os dados

Atalho: `ctrl + shift + H`

Windows: inverter as barras ("`\`" por "`/`")!

```
## diretorio de trabalho
# pasta onde o r ira importar e exportar os arquivos

# definir o diretorio de trabalho
setwd("/home/mude/data/github/course-geospatial-data-r/03_dados/tabelas")
```

```
# verificar o diretorio
getwd()
```

```
# listar os arquivos
dir()
```

Vamos trabalhar com dados reais?



*Real world
data*

*iris &
mtcars*

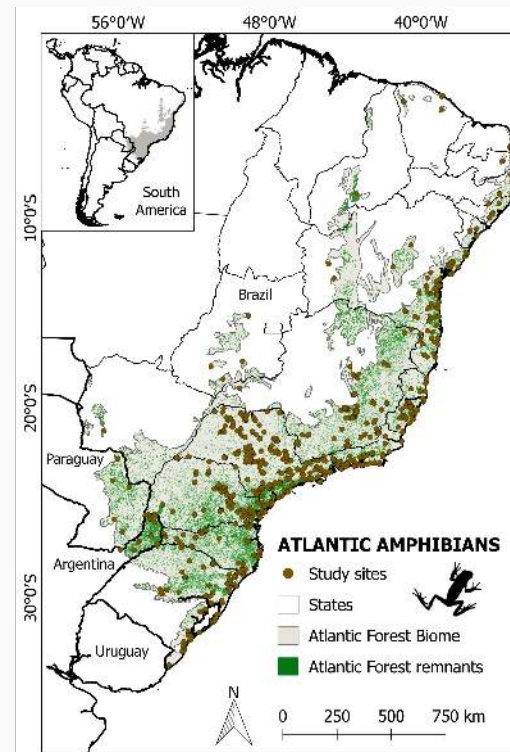
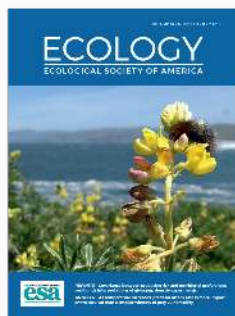
6. Importar dados

ATLANTIC AMPHIBIANS: a dataset of amphibian communities from the Atlantic Forests of South America

Eu mesmo et al. (2018)



[Vancine et al. \(2018\)](#)



6. Importar dados

Os arquivos de tabelas geralmente estão num desses **três** formatos:

1. csv

2. txt

3. xlsx



6. Importar dados

Ler uma planilha eletrônica (.csv)

```
# ler uma planilha eletrônica (.csv)
read.csv("ATLANTIC_AMPHIBIANS_sites.csv", encoding = "latin1")
```

	id	reference_number	species_number	record	sampled_habitat	active_methods	passive_methods	complementary_met
## 1	amp1001	1001	19	ab	fo,ll	as	pt	
## 2	amp1002	1002	16	co	fo,la,ll	as	pt	
## 3	amp1003	1002	14	co	fo,la,ll	as	pt	
## 4	amp1004	1002	13	co	fo,la,ll	as	pt	
## 5	amp1005	1003	30	co	fo,ll,br	as	<NA>	
## 6	amp1006	1004	42	co	tp,pp,la,ll,is	<NA>	<NA>	
## 7	amp1007	1005	23	co	sp	as	<NA>	
## 8	amp1008	1005	19	co	sp,la,sw	as,sb,tr	<NA>	
## 9	amp1009	1005	13	ab	fo	<NA>	pt	
## 10	amp1010	1006	1	ab	fo	<NA>	pt	
## 11	amp1011	1006	1	ab	fo	<NA>	pt	
## 12	amp1012	1006	2	ab	fo	<NA>	pt	
## 13	amp1013	1006	4	ab	fo	<NA>	pt	
## 14	amp1014	1006	4	ab	fo	<NA>	pt	
## 15	amp1015	1006	6	ab	fo	<NA>	pt	
## 16	amp1016	1006	5	ab	fo	<NA>	pt	

6. Importar dados

Ler e atribuir uma planilha eletrônica (.csv) a um objeto

```
# ler e atribuir uma planilha eletrônica (.csv) a um objeto  
da ← read.csv("ATLANTIC_AMPHIBIANS_sites.csv", encoding = "latin1")
```

```
# ver os dados  
da
```

```
# conferir a classe  
class(da)
```

```
## [1] "data.frame"
```

IMPORTANTE: a tabela importada para o R sempre será
um **data frame**!

6. Importar dados

Ler e atribuir uma planilha simples (.txt) a um objeto

```
# ler e atribuir uma planilha simples (.txt) a um objeto
da <- read.table("ATLANTIC_AMPHIBIANS_sites.txt", header = TRUE, sep = "\t")
da
```

##	id	reference_number	species_number	record	sampled_habitat	active_methods	passive_methods	complementary_met
## 1	amp1001	1001	19	ab	fo,ll	as	pt	
## 2	amp1002	1002	16	co	fo,la,ll	as	pt	
## 3	amp1003	1002	14	co	fo,la,ll	as	pt	
## 4	amp1004	1002	13	co	fo,la,ll	as	pt	
## 5	amp1005	1003	30	co	fo,ll,br	as	<NA>	
## 6	amp1006	1004	42	co	tp,pp,la,ll,is	<NA>	<NA>	
## 7	amp1007	1005	23	co	sp	as	<NA>	
## 8	amp1008	1005	19	co	sp,la,sw	as,sb,tr	<NA>	
## 9	amp1009	1005	13	ab	fo	<NA>	pt	
## 10	amp1010	1006	1	ab	fo	<NA>	pt	
## 11	amp1011	1006	1	ab	fo	<NA>	pt	
## 12	amp1012	1006	2	ab	fo	<NA>	pt	
## 13	amp1013	1006	4	ab	fo	<NA>	pt	
## 14	amp1014	1006	4	ab	fo	<NA>	pt	
## 15	amp1015	1006	6	ab	fo	<NA>	pt	

6. Importar dados

Ler e atribuir uma planilha eletrônica (.xlsx) a um objeto

Pacote **openxlsx**

```
# pacote openxlsx
# install.packages("openxlsx")
library(openxlsx)
```

Importar os dados

```
# ler e atribuir uma planilha eletrônica (.xlsx) a um objeto
da <- openxlsx::read.xlsx("ATLANTIC_AMPHIBIANS_sites.xlsx", sheet = 1, encoding = "latin1")
da
```

##		id	reference_number	species_number	record	sampled_habitat	active_methods	passive_methods	complementary_met
## 1	amp1001		1001	19	ab	fo,ll	as	pt	
## 2	amp1002		1002	16	co	fo,la,ll	as	pt	
## 3	amp1003		1002	14	co	fo,la,ll	as	pt	
## 4	amp1004		1002	13	co	fo,la,ll	as	pt	
## 5	amp1005		1003	30	co	fo,ll,br	as	<NA>	
## 6	amp1006		1004	42	co	tp,pp,la,ll,is	<NA>	<NA>	

7. Conferência de dados importados

Conjunto de funções para conferir os dados

Funções de conferência

`head()`: mostra as primeiras 6 linhas

`tail()`: mostra as últimas 6 linhas

`nrow()`: mostra o número de linhas

`ncol()`: mostra o número de colunas

`dim()`: mostra o número de linhas e de colunas

`rownames()`: mostra os nomes das linhas (locais)

`colnames()`: mostra os nomes das colunas (variáveis)

`str()`: mostra as classes de cada coluna (estrutura)

`summary()`: mostra um resumo dos valores de cada coluna

7. Conferência de dados importados

Conjunto de funções para conferir os dados

`head()`: mostra as primeiras 6 linhas

```
# primeiras linhas
head(da)
```

```
##          id reference_number species_number record sampled_habitat active_methods passive_methods complementary_meth
## 1 amp1001          1001          19      ab          fo,ll          as          pt          <
## 2 amp1002          1002          16      co          fo,la,ll          as          pt          <
## 3 amp1003          1002          14      co          fo,la,ll          as          pt          <
## 4 amp1004          1002          13      co          fo,la,ll          as          pt          <
## 5 amp1005          1003          30      co          fo,ll,br          as          <NA>          <
## 6 amp1006          1004          42      co      tp,pp,la,ll,is          <NA>          <NA>          <
##   year_start month_finish year_finish effort_months country  state state_abbreviation municipality
## 1      2000           1          2002           16  Brazil Piauí          BR-PI          Canto do Buriti
## 2      2007           5          2009           17  Brazil Ceará;          BR-CE  São Gonçalo do Amarante
## 3      2007           5          2009           17  Brazil Ceará;          BR-CE  São Gonçalo do Amarante
## 4      2007           5          2009           17  Brazil Ceará;          BR-CE  São Gonçalo do Amarante
## 5      1988           8          2001          157  Brazil Ceará;          BR-CE          Baturitã
## 6           NA          NA          NA           NA  Brazil Ceará;          BR-CE          Quebrangulo
##          site latitude longitude coordinate_precision altitude temperature precipitation
```

7. Conferência de dados importados

Conjunto de funções para conferir os dados

`head()`: mostra as primeiras 10 linhas

```
# primeiras linhas  
head(da, 10)
```

```
##           id reference_number species_number record sampled_habitat active_methods passive_methods complementary_met  
## 1  amp1001           1001           19    ab           fo,ll           as           pt  
## 2  amp1002           1002           16    co           fo,la,ll           as           pt  
## 3  amp1003           1002           14    co           fo,la,ll           as           pt  
## 4  amp1004           1002           13    co           fo,la,ll           as           pt  
## 5  amp1005           1003           30    co           fo,ll,br           as           <NA>  
## 6  amp1006           1004           42    co  tp,pp,la,ll,is           <NA>           <NA>  
## 7  amp1007           1005           23    co           sp           as           <NA>  
## 8  amp1008           1005           19    co           sp,la,sw       as,sb,tr           <NA>  
## 9  amp1009           1005           13    ab           fo           <NA>           pt  
## 10 amp1010           1006            1    ab           fo           <NA>           pt  
##      year_start month_finish year_finish effort_months country           state state_abbreviation           m  
## 1      2000           1      2002           16  Brazil       Piauí          BR-PI           Canto  
## 2      2007           5      2009           17  Brazil       Ceará          BR-CE  São Gonçalo  
## 3      2007           5      2009           17  Brazil       Ceará          BR-CE  São Gonçalo
```

7. Conferência de dados importados

Conjunto de funções para conferir os dados

`tail()`: mostra as últimas 6 linhas

```
# ultimas linhas
tail(da)
```

```
##          id reference_number species_number record sampled_habitat active_methods passive_methods complementary_m
## 1158 amp2158          1389           3      co          <NA>          <NA>          <NA>
## 1159 amp2159          1389           9      co          <NA>          <NA>          <NA>
## 1160 amp2160          1389           6      co          <NA>          <NA>          <NA>
## 1161 amp2161          1389           1      co          <NA>          <NA>          <NA>
## 1162 amp2162          1389           2      co          <NA>          <NA>          <NA>
## 1163 amp2163          1389           2      co          <NA>          <NA>          <NA>
##          year_start month_finish year_finish effort_months  country  state state_abbreviation  municipality
## 1158          NA          NA          NA          NA Argentina  Misiones          AR-N Manuel Belgrano
## 1159          NA          NA          NA          NA Argentina  Misiones          AR-N          Posadas
## 1160          NA          NA          NA          NA Argentina  Misiones          AR-N Montecarlo
## 1161          NA          NA          NA          NA Argentina  Misiones          AR-N San Pedro
## 1162          NA          NA          NA          NA Argentina  Misiones          AR-N Caingüã's Balne
## 1163          NA          NA          NA          NA Argentina  Misiones          AR-N Oberã;
##          latitude longitude coordinate_precision altitude temperature precipitation
```

7. Conferência de dados importados

Conjunto de funções para conferir os dados

`nrow()`: mostra o número de linhas

```
# numero de linhas  
nrow(da)
```

```
## [1] 1163
```

`ncol()`: mostra o número de colunas

```
# numero de colunas  
ncol(da)
```

```
## [1] 25
```

`dim()`: mostra o número de linhas e de colunas

```
# numero de linhas e de colunas  
dim(da)
```

7. Conferência de dados importados

Conjunto de funções para conferir os dados

`rownames()`: mostra os nomes das linhas (locais)

```
# nome das linhas  
rownames(da)
```

##	[1]	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"	"10"	"11"	"12"	"13"	"14"	"15"	"1"
##	[20]	"20"	"21"	"22"	"23"	"24"	"25"	"26"	"27"	"28"	"29"	"30"	"31"	"32"	"33"	"34"	"3"
##	[39]	"39"	"40"	"41"	"42"	"43"	"44"	"45"	"46"	"47"	"48"	"49"	"50"	"51"	"52"	"53"	"5"
##	[58]	"58"	"59"	"60"	"61"	"62"	"63"	"64"	"65"	"66"	"67"	"68"	"69"	"70"	"71"	"72"	"7"
##	[77]	"77"	"78"	"79"	"80"	"81"	"82"	"83"	"84"	"85"	"86"	"87"	"88"	"89"	"90"	"91"	"9"
##	[96]	"96"	"97"	"98"	"99"	"100"	"101"	"102"	"103"	"104"	"105"	"106"	"107"	"108"	"109"	"110"	"1"
##	[115]	"115"	"116"	"117"	"118"	"119"	"120"	"121"	"122"	"123"	"124"	"125"	"126"	"127"	"128"	"129"	"1"
##	[134]	"134"	"135"	"136"	"137"	"138"	"139"	"140"	"141"	"142"	"143"	"144"	"145"	"146"	"147"	"148"	"1"
##	[153]	"153"	"154"	"155"	"156"	"157"	"158"	"159"	"160"	"161"	"162"	"163"	"164"	"165"	"166"	"167"	"1"
##	[172]	"172"	"173"	"174"	"175"	"176"	"177"	"178"	"179"	"180"	"181"	"182"	"183"	"184"	"185"	"186"	"1"
##	[191]	"191"	"192"	"193"	"194"	"195"	"196"	"197"	"198"	"199"	"200"	"201"	"202"	"203"	"204"	"205"	"2"
##	[210]	"210"	"211"	"212"	"213"	"214"	"215"	"216"	"217"	"218"	"219"	"220"	"221"	"222"	"223"	"224"	"2"
##	[229]	"229"	"230"	"231"	"232"	"233"	"234"	"235"	"236"	"237"	"238"	"239"	"240"	"241"	"242"	"243"	"2"
##	[248]	"248"	"249"	"250"	"251"	"252"	"253"	"254"	"255"	"256"	"257"	"258"	"259"	"260"	"261"	"262"	"2"
##	[267]	"267"	"268"	"269"	"270"	"271"	"272"	"273"	"274"	"275"	"276"	"277"	"278"	"279"	"280"	"281"	"2"

7. Conferência de dados importados

Conjunto de funções para conferir os dados

`colnames()`: mostra os nomes das colunas (variáveis)

```
# nome das colunas  
colnames(da)
```

```
## [1] "id" "reference_number" "species_number" "record" "sampled_habi  
## [6] "active_methods" "passive_methods" "complementary_methods" "period" "month_start"  
## [11] "year_start" "month_finish" "year_finish" "effort_months" "country"  
## [16] "state" "state_abbreviation" "municipality" "site" "latitude"  
## [21] "longitude" "coordinate_precision" "altitude" "temperature" "precipitatio
```

7. Conferência de dados importados

Conjunto de funções para conferir os dados

`str()`: mostra as classes de cada coluna (estrutura)

```
# estrutura dos dados  
str(da)
```

```
## 'data.frame':    1163 obs. of  25 variables:  
##  $ id                : chr  "amp1001" "amp1002" "amp1003" "amp1004" ...  
##  $ reference_number  : int   1001 1002 1002 1002 1003 1004 1005 1005 1005 1006 ...  
##  $ species_number    : int    19 16 14 13 30 42 23 19 13 1 ...  
##  $ record            : chr   "ab" "co" "co" "co" ...  
##  $ sampled_habitat   : chr   "fo,ll" "fo,la,ll" "fo,la,ll" "fo,la,ll" ...  
##  $ active_methods    : chr   "as" "as" "as" "as" ...  
##  $ passive_methods   : chr   "pt" "pt" "pt" "pt" ...  
##  $ complementary_methods: chr  NA NA NA NA ...  
##  $ period            : chr   "mo,da,tw,ni" "mo,da,tw,ni" "mo,da,tw,ni" "mo,da,tw,ni" ...  
##  $ month_start       : int    9 12 12 12 7 NA 4 4 4 5 ...  
##  $ year_start        : int   2000 2007 2007 2007 1988 NA 2007 2007 2007 2011 ...  
##  $ month_finish      : int    1 5 5 5 8 NA 4 4 4 7 ...  
##  $ year_finish       : int   2002 2009 2009 2009 2001 NA 2009 2009 2009 2011 ...  
##  $ effort_months     : int   16 17 17 17 157 NA 24 24 24 2 ...
```

7. Conferência de dados importados

Conjunto de funções para conferir os dados

`summary()`: mostra um resumo dos valores de cada coluna

```
# resumo dos dados  
summary(da)
```

```
##          id          reference_number species_number      record      sampled_habitat      active_methods      pass
## Length:1163      Min.    :1001      Min.    : 1.00      Length:1163      Length:1163      Length:1163      Leng
## Class :character 1st Qu.:1096      1st Qu.: 7.00      Class :character  Class :character  Class :character  Clas
## Mode  :character Median :1204      Median :13.00      Mode  :character  Mode  :character  Mode  :character  Mode
##                      Mean  :1196      Mean   :15.17
##                      3rd Qu.:1295      3rd Qu.:21.00
##                      Max.   :1389      Max.   :80.00
##
## complementary_methods      period          month_start      year_start      month_finish      year_finish      effort_m
## Length:1163          Length:1163      Min.    : 1.000      Min.    :1940      Min.    : 1.000      Min.    :1983      Min.    :
## Class :character      Class :character 1st Qu.: 4.000      1st Qu.:2004      1st Qu.: 3.000      1st Qu.:2006      1st Qu.:
## Mode  :character      Mode  :character Median : 8.000      Median :2007      Median : 6.000      Median :2009      Median :
##                      Mean  : 7.058      Mean   :2007      Mean   : 6.375      Mean   :2008      Mean   :
##                      3rd Qu.:10.000      3rd Qu.:2011      3rd Qu.:10.000      3rd Qu.:2012      3rd Qu.:
##                      Max.   :12.000      Max.   :2015      Max.   :12.000      Max.   :2017      Max.   :
##                      147/153 :
```

7. Conferência de dados importados

Conjunto de funções para conferir os dados

Verificar a presença de NAs

```
# algum nas?  
any(is.na(da))
```

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

```
# quais sao nas?  
which(is.na(da))
```

```
##      [1] 4685 4686 4687 4688 4689 4690 4691 4692 4693 4711 4744 4749 4751 4780 4781 4782 4783 4784 4785 4786 4787 47  
##      [28] 4794 4795 4796 4797 4798 4799 4852 4853 4854 4855 4873 4874 4965 5005 5020 5026 5027 5028 5034 5035 5036 50  
##      [55] 5223 5224 5225 5227 5249 5326 5327 5374 5469 5480 5518 5519 5526 5529 5530 5531 5532 5533 5534 5566 5567 55  
##      [82] 5665 5690 5717 5728 5729 5736 5738 5744 5798 5801 5806 5807 5808 5809 5810 5811 5812 5813 5814 5815 5821 58  
##     [109] 5830 5831 5832 5833 5834 5835 5836 5837 5838 5839 5840 5841 5842 5843 5844 5845 5846 5847 5848 5849 5850 58  
##     [136] 5903 5907 5912 5914 5925 5926 5966 5967 5968 5969 5970 5971 5972 5973 6022 6023 6024 6025 6029 6030 6036 60  
##     [163] 6142 6162 6183 6186 6187 6189 6190 6236 6237 6238 6239 6240 6241 6249 6250 6251 6252 6253 6254 6255 6256 62  
##     [190] 6277 6278 6279 6280 6281 6282 6283 6284 6285 6286 6287 6288 6289 6290 6305 6306 6307 6308 6309 6310 6311 63  
##     [217] 6367 6368 6369 6370 6371 6412 6490 6527 6529 6537 6663 6665 6666 6681 6682 6689 6692 6693 6694 6695 6696 66  
148/153
```

7. Conferência de dados importados

Conjunto de funções para manipular os dados

Retirar os NAs

```
# omitir as linhas com nas e atribuir  
da_na <- na.omit(da)
```

```
# numero de linhas do dado total  
nrow(da)
```

```
## [1] 1163
```

```
# numero de linhas do dado sem nas  
nrow(da_na)
```

```
## [1] 40
```

7. Conferência de dados importados

Conjunto de funções para manipular os dados

Subset das linhas

```
# subset das linhas com amostragens em sao paulo  
da_sp <- da[da$state == "São Paulo", ]  
da_sp
```

```
## [1] id                reference_number    species_number      record              sampled_habitat  
## [7] passive_methods    complementary_methods period              month_start         year_start  
## [13] year_finish        effort_months      country             state               state_abbreviation  
## [19] site              latitude           longitude           coordinate_precision altitude  
## [25] precipitation  
## <0 rows> (or 0-length row.names)
```

8. Exportar dados

Exportar uma tabela de dados na pasta do diretório

Planilha eletrônica (.csv)

```
# exportar csv
write.csv(da_sp, "ATLANTIC_AMPHIBIAN_sites_sao_paulo.csv",
          row.names = FALSE, quote = FALSE)
```

Planilha de texto (.txt)

```
# exportar txt
write.table(da_sp, "ATLANTIC_AMPHIBIAN_sites_sao_paulo.txt",
            row.names = FALSE, quote = FALSE)
```

Planilha eletrônica (.xlsx)

```
# exportar xlsx
openxlsx::write.xlsx(da_sp, "ATLANTIC_AMPHIBIAN_sites_sao_paulo.xlsx",
                     row.names = FALSE, quote = FALSE)
```

Dúvidas?

Maurício Vancine

Contatos:

✉ mauricio.vancine@gmail.com

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

🌐 [mauriciovancine](https://mauriciovancine.github.io)

🔗 mauriciovancine.github.io



Slides criados via pacote [xaringan](#) e tema [Metropolis](#). Animação dos sapos por [@probzz](#).