

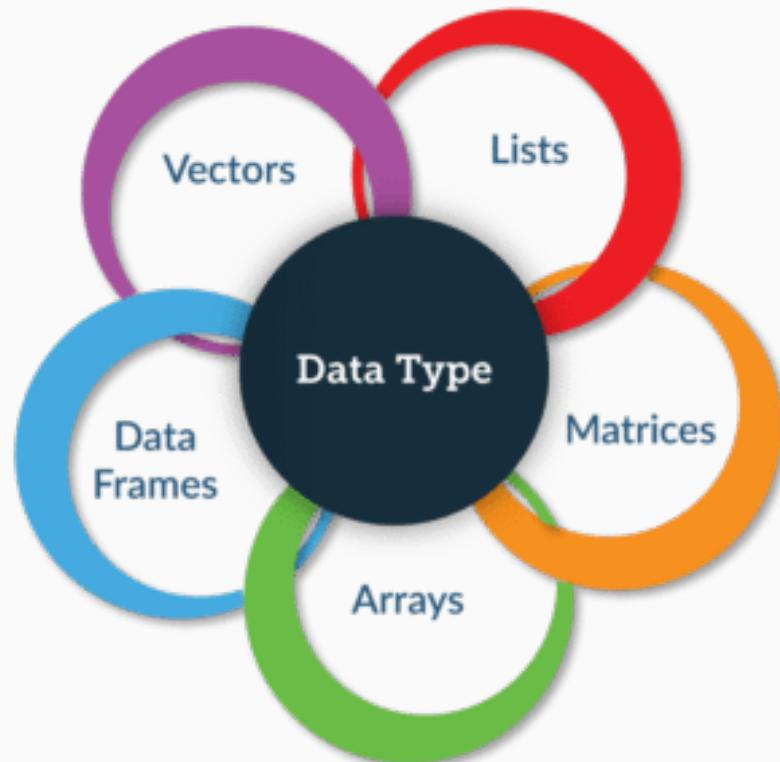
Introdução ao tidyverse

2 Revisão de estrutura de dados

xaringan [presentation ninja]

Maurício Vancine

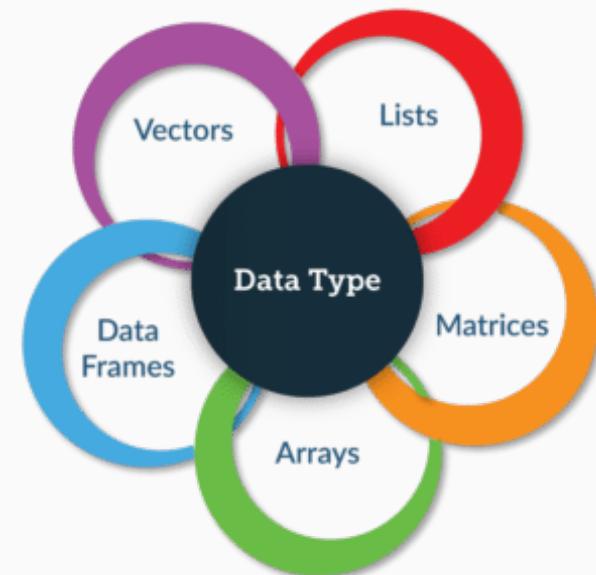
25/04/2019



2 Estrutura e manejo de dados

Conteúdo

- 2.1 Atributos dos objetos
- 2.2 Modos dos objetos (*numeric, character* e *logical*)
- 2.3 Estrutura dos objetos (*vector, factor, matrix, data frame* e *lista*)
- 2.4 Manejo de dados unidimensionais
- 2.5 Manejo de dados bidimensionais
- 2.6 Valores faltantes e especiais
- 2.7 Diretório de trabalho
- 2.8 Importar dados
- 2.9 Conferir e manejear dados importados
- 2.10 Exportar dados



2 Estrutura e manejo de dados

Script

```
script_aula_02.R
```

2.1 Atributos dos objetos

Atribuição

palavra <- dados

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

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

2.1 Atributos dos objetos

Atributos dos objetos no R

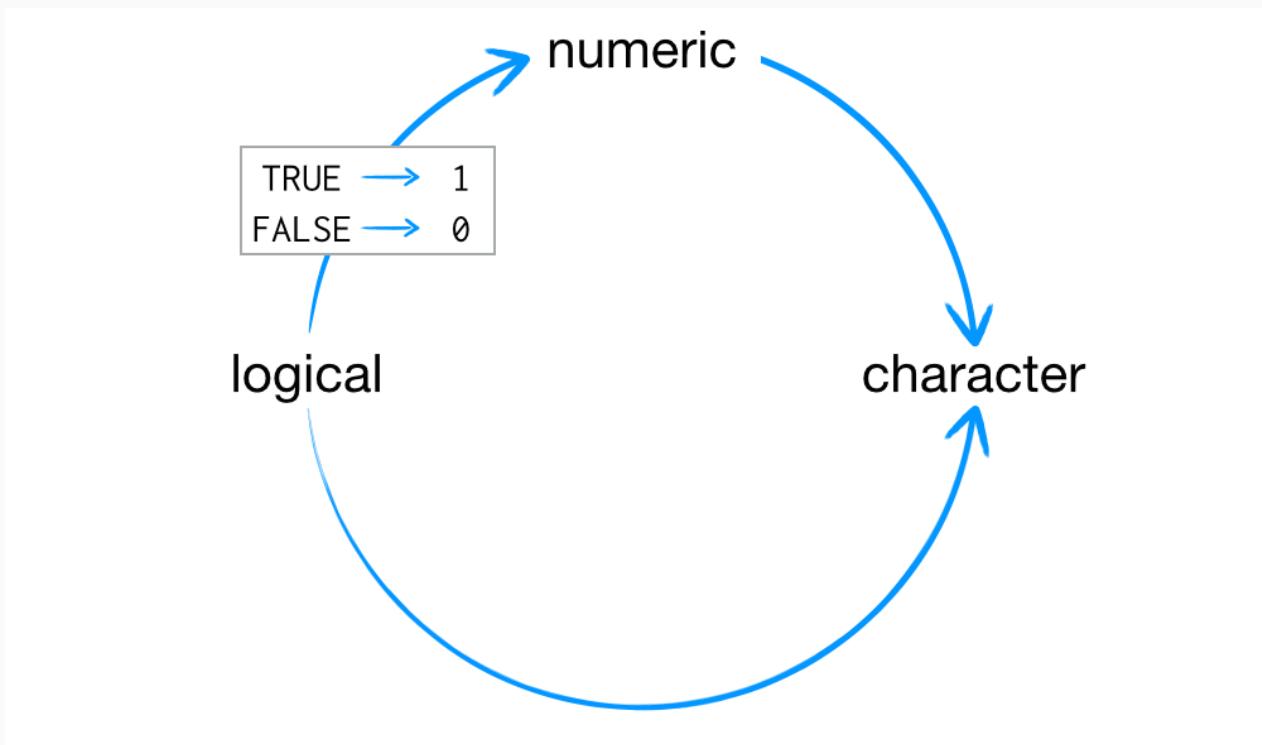
Objetos possuem **três características**:

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

2.2 Modos dos objetos

Modos (*natureza*): numeric, character e logical

Natureza dos **elementos** que compõem os objetos



2.2 Modos dos objetos

1. Numeric: números inteiros ou decimais

```
# numeric  
obj_num <- 1  
obj_num
```

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

```
# mode  
mode(obj_num)
```

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

2.2 Modos dos objetos

2. Character: texto

```
# character  
obj_cha <- "a" # atencao para as aspas  
obj_cha
```

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

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

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

2.2 Modos dos objetos

3. Logical: assume apenas dois valores (TRUE ou FALSE
- booleano)

```
# logical
obj_log <- TRUE # maiusculas e sem aspas
obj_log
```

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

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

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

2.2 Modos dos objetos

Resumindo:

A **natureza** dos **elementos** irá definir os **modos** dos objetos

Modos (*natureza*) são três:

numeric (**número**): *1*

character (**texto**): *"a"*, *"2500"*, *"amostra_01"*

logical (**lógico**): *TRUE* ou *FALSE*

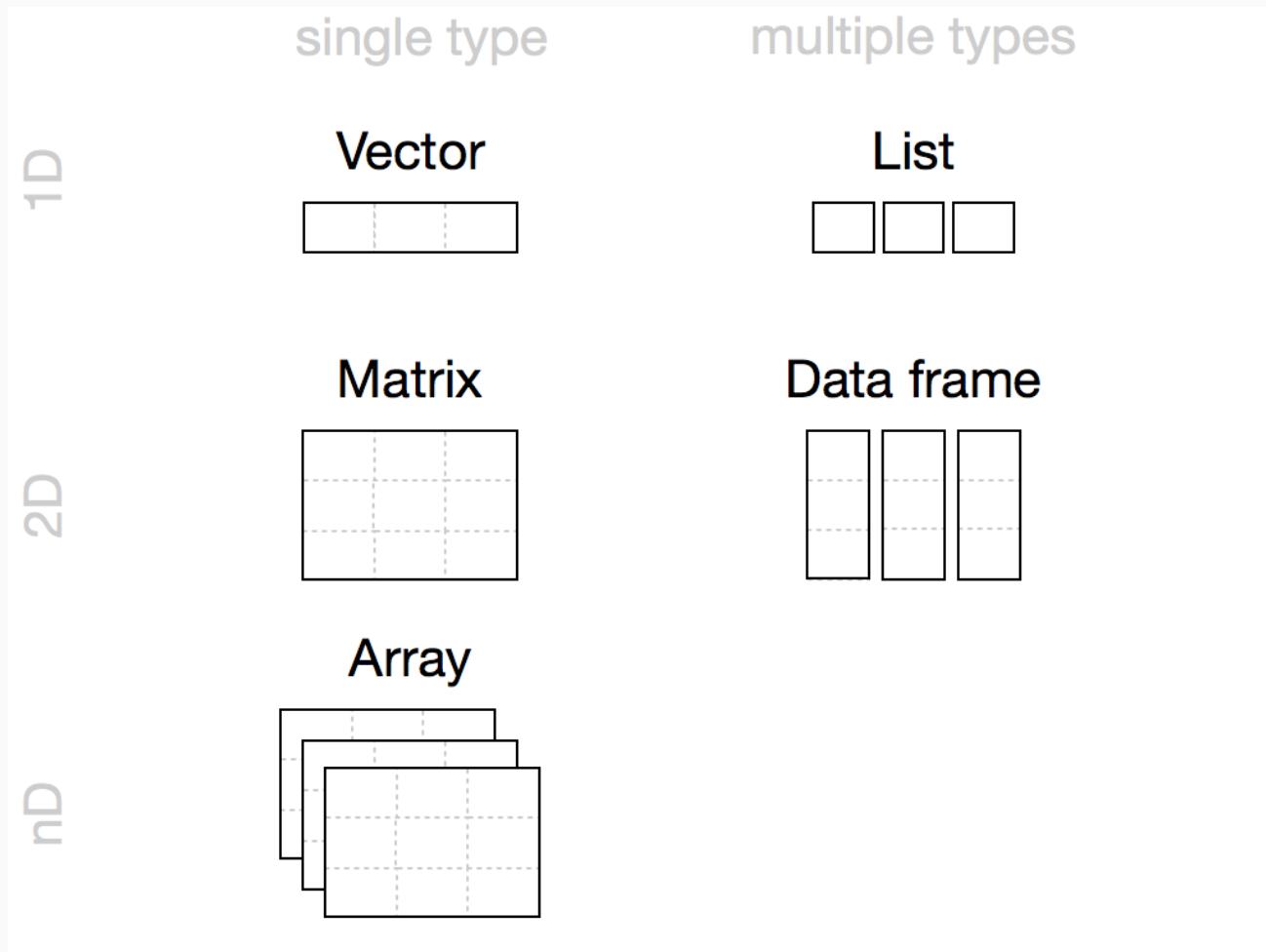
2.3 Estrutura dos objetos

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

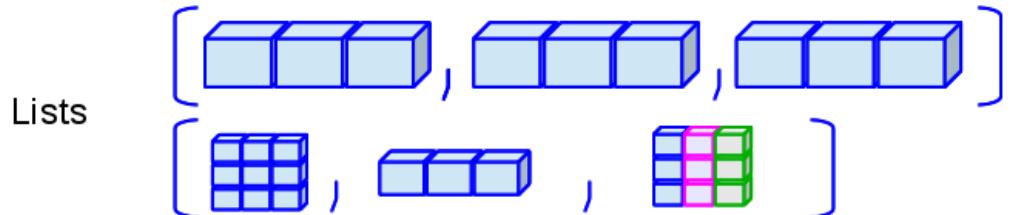
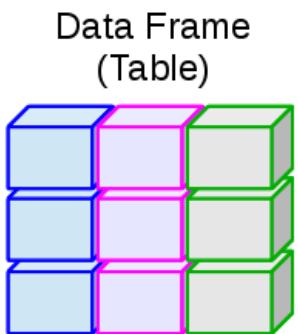
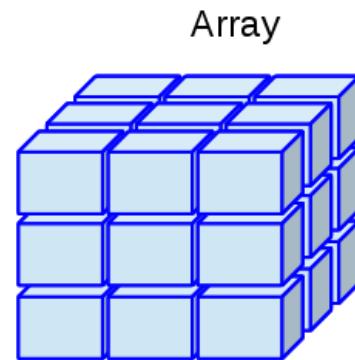
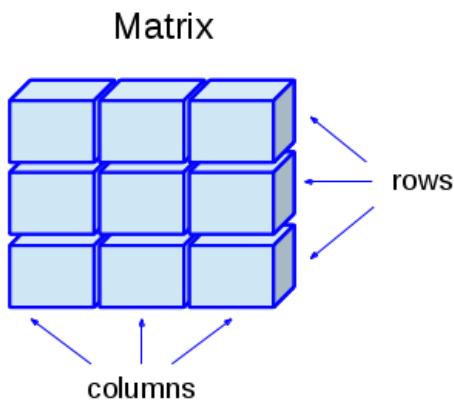
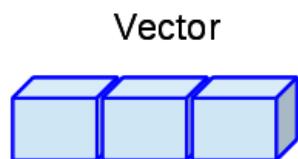
Organização (**modos e dimensionalidade**) dos elementos dos objetos

	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data Frame
nd	Array	

2.3 Estrutura dos objetos



2.3 Estrutura dos objetos



2.3 Estrutura dos objetos

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

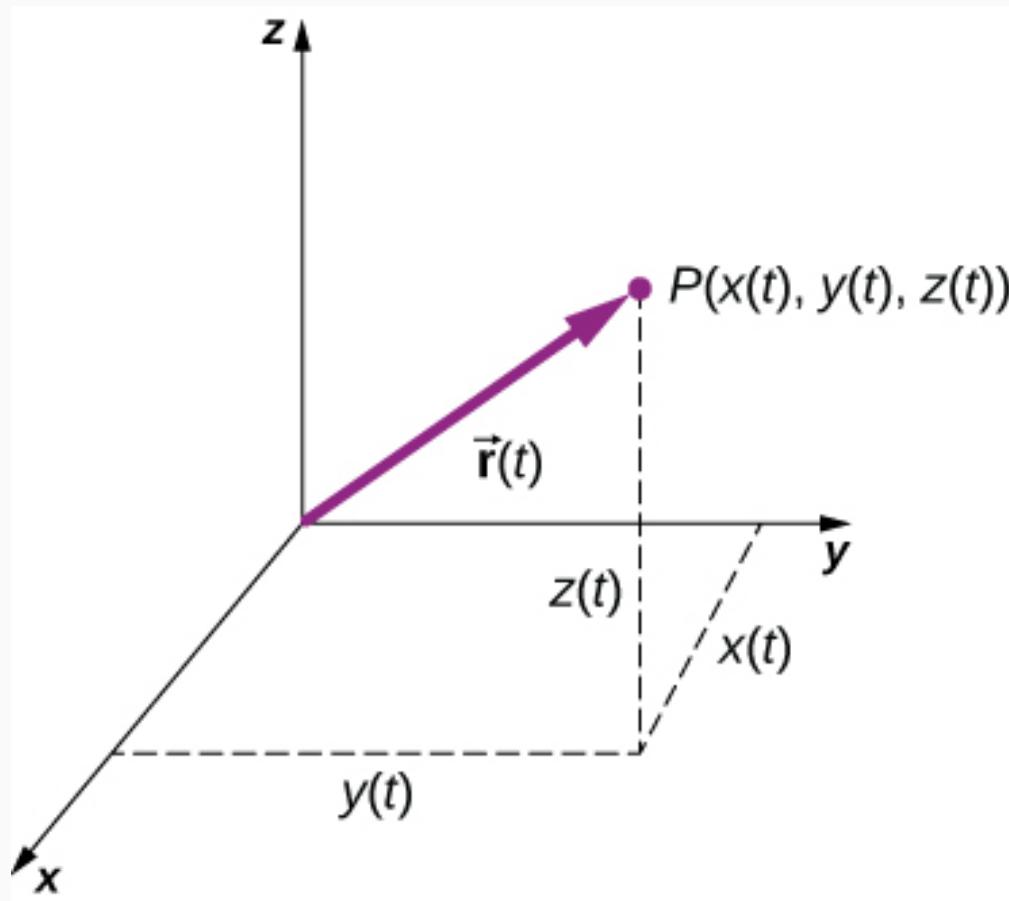
O **vetor** representa medidas de uma **variável quantitativa** (discretas ou contínuas) ou **Descrição** (informações em texto)

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

1. Amostragens: {"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}

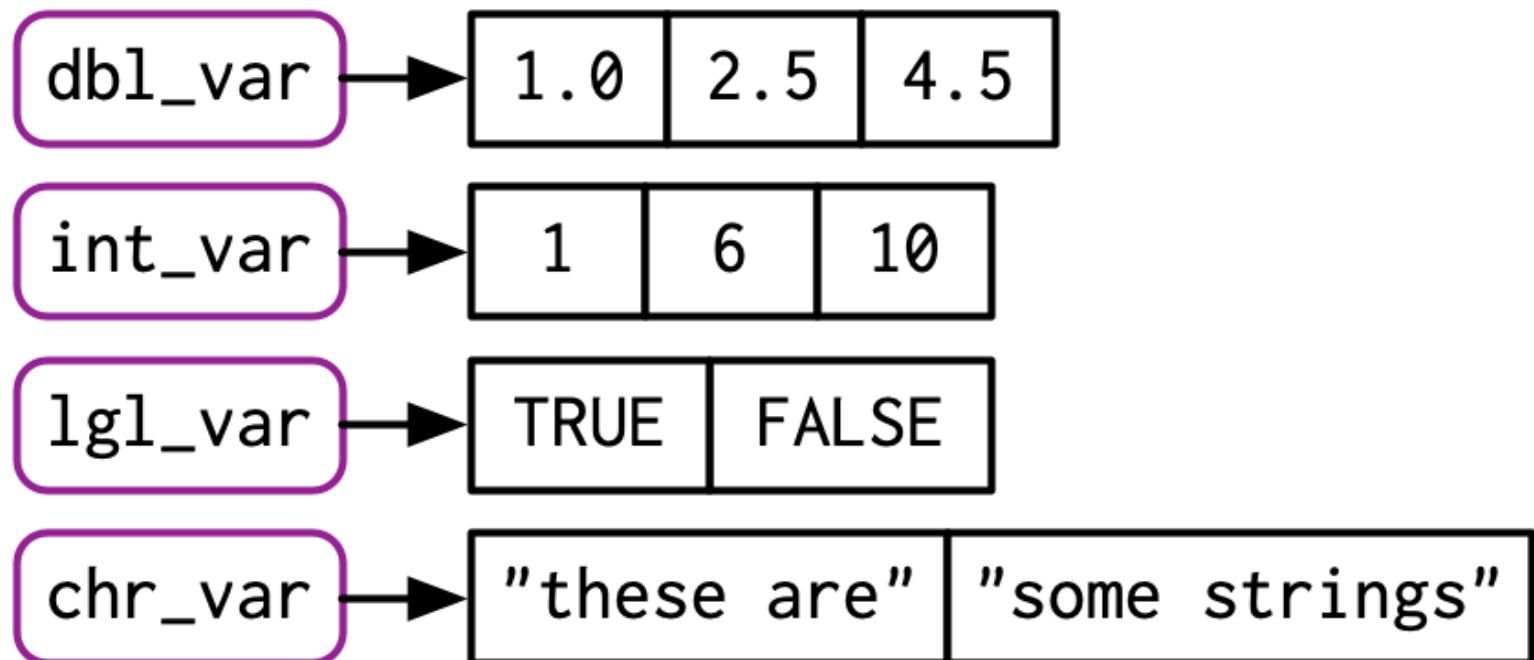
2.3 Estrutura dos objetos

Não me refiro exatamente ao vetor da matemática



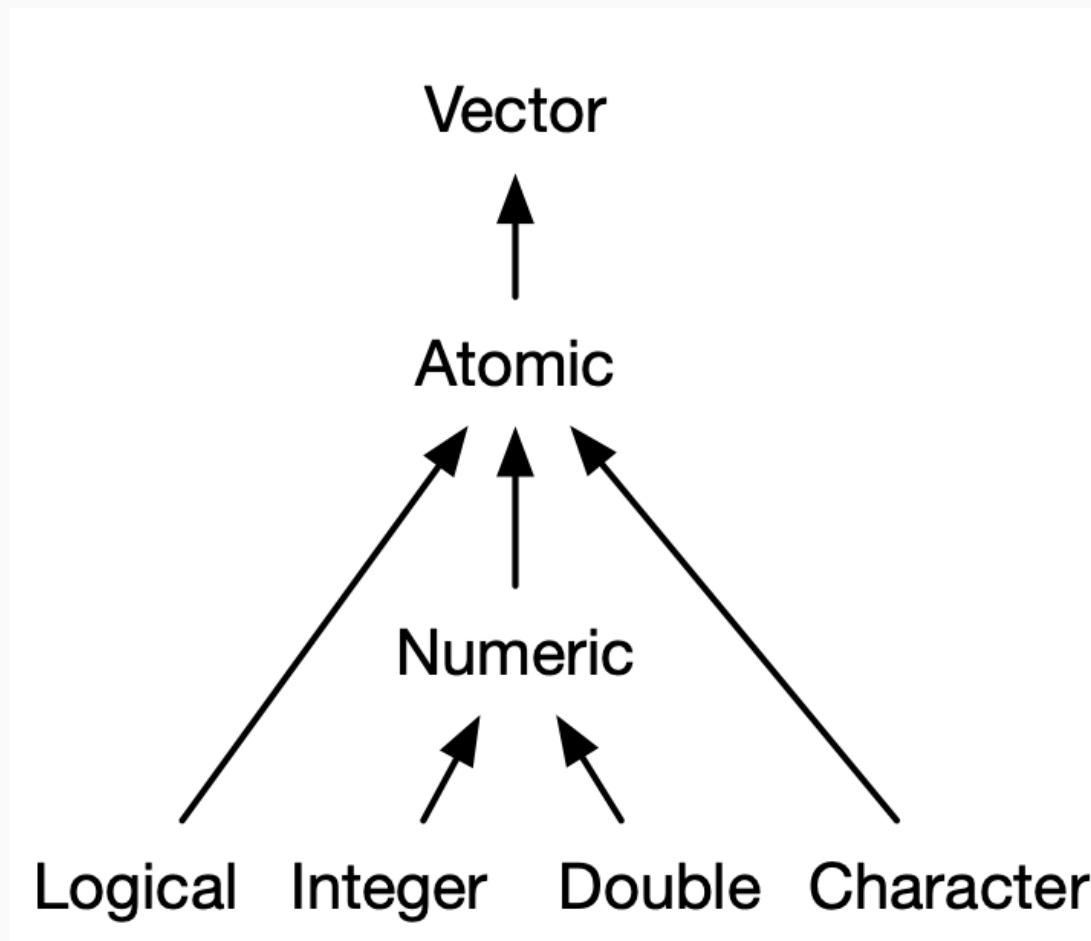
2.3 Estrutura dos objetos

Sequência de elementos



2.3 Estrutura dos objetos

Tipos



2.3 Estrutura 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"
```

2.3 Estrutura dos objetos

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

2. Sequência

```
# sequencia unitaria (x1:x2)
se_e <- 1:10
se_e
```

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

```
# sequencia com diferentes espaçamentos
se_e <- seq(from = 0, to = 100, by = 10)
se_e
```

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

2.3 Estrutura dos objetos

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

3. Repetição

```
# repeticao
# 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"
```

2.3 Estrutura 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"
```

2.3 Estrutura dos objetos

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

5. Amostrando aleatoriamente elementos

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

```
## [1] 85 67 44 42 12 26 23 33 19 75
```

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

```
## [1] 6 9 8 2 5 9 10 4 7 3 4 2 4 8 4 9 5 3 5 6 1 1 2 1  
## [41] 3 4 4 5 7 2 4 7 4 3 1 6 5 8 3 3 7 3 2 4 9 1 5 4  
## [81] 8 7 1 7 6 9 3 8 2 2 10 9 7 2 3 7 8 5 3 2
```

Exercícios

Exercício 05

Vector

Escolham números para jogar na mega-sena

Lembrando: são 6 valores de 1 a 60 e atribuam a um objeto

02 : 00

Exercício 05

Resposta

```
# solucao  
mega_num <- sample(1:60, 6, replace = FALSE)  
mega_num  
  
## [1] 43 47 22 53 2 16
```

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

2.3 Estrutura dos objetos

Vetor com elementos de **modos diferentes**:

```
ve <- c(1, "a", 3)
```

```
ve
```

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

```
ve <- c(1, "a", TRUE)
```

```
ve
```

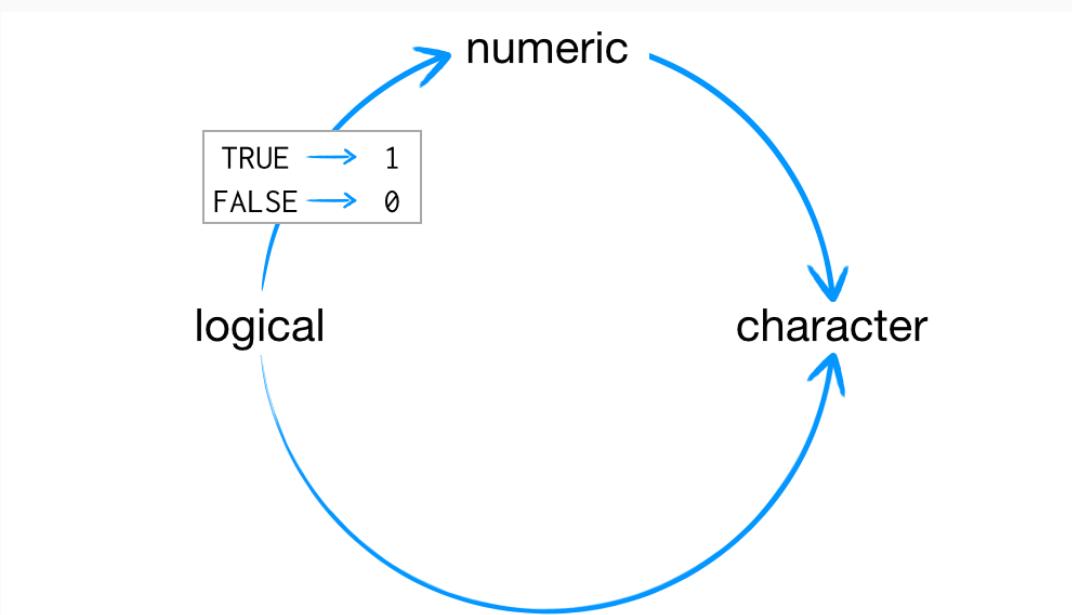
```
## [1] "1"      "a"      "TRUE"
```

Coerção

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

Essa mudança segue essa ordem:

DOMINANTE character >> numeric >> logical RECESSIVO



Conversão

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

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

Conversão

```
# funcoes  
as.character()  
as.integer()  
as.numeric()  
as.logical()
```

2.3 Estrutura 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. Amostragens: {"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}

2.3 Estrutura dos objetos

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



2.3 Estrutura dos objetos

2. Factor nominal: variáveis nominais

```
fa_no <- factor(x = c("fechada", "fechada", "aberta", "aberta", "aberta"),  
                  levels = c("aberta", "fechada"))  
fa_no
```

```
## [1] fechada fechada aberta  aberta  aberta  
## Levels: aberta fechada
```

```
levels(fa_no)
```

```
## [1] "aberta"  "fechada"
```

2.3 Estrutura dos objetos

2. Factor ordinal: variáveis ordinais

```
fa_or <- factor(x = c("alta", "media", "baixa", "baixa", "media"),
                  levels = c("baixa", "media", "alta"), ordered = TRUE)
fa_or
```

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

```
levels(fa_or)
```

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

2.3 Estrutura dos objetos

2. Factor: conversão

Criar um vetor **character**

```
ve_ch <- c("alta", "media", "baixa", "baixa", "media")
ve_ch
```

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

```
mode(ve_ch)
```

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

```
class(ve_ch)
```

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

2.3 Estrutura dos objetos

2. Factor: conversão

Forçar a ser **factor nominal**

```
fa_no <- as.factor(ve_ch)  
fa_no
```

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

```
levels(fa_no)
```

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

```
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
```

02 : 00

Exercício 06

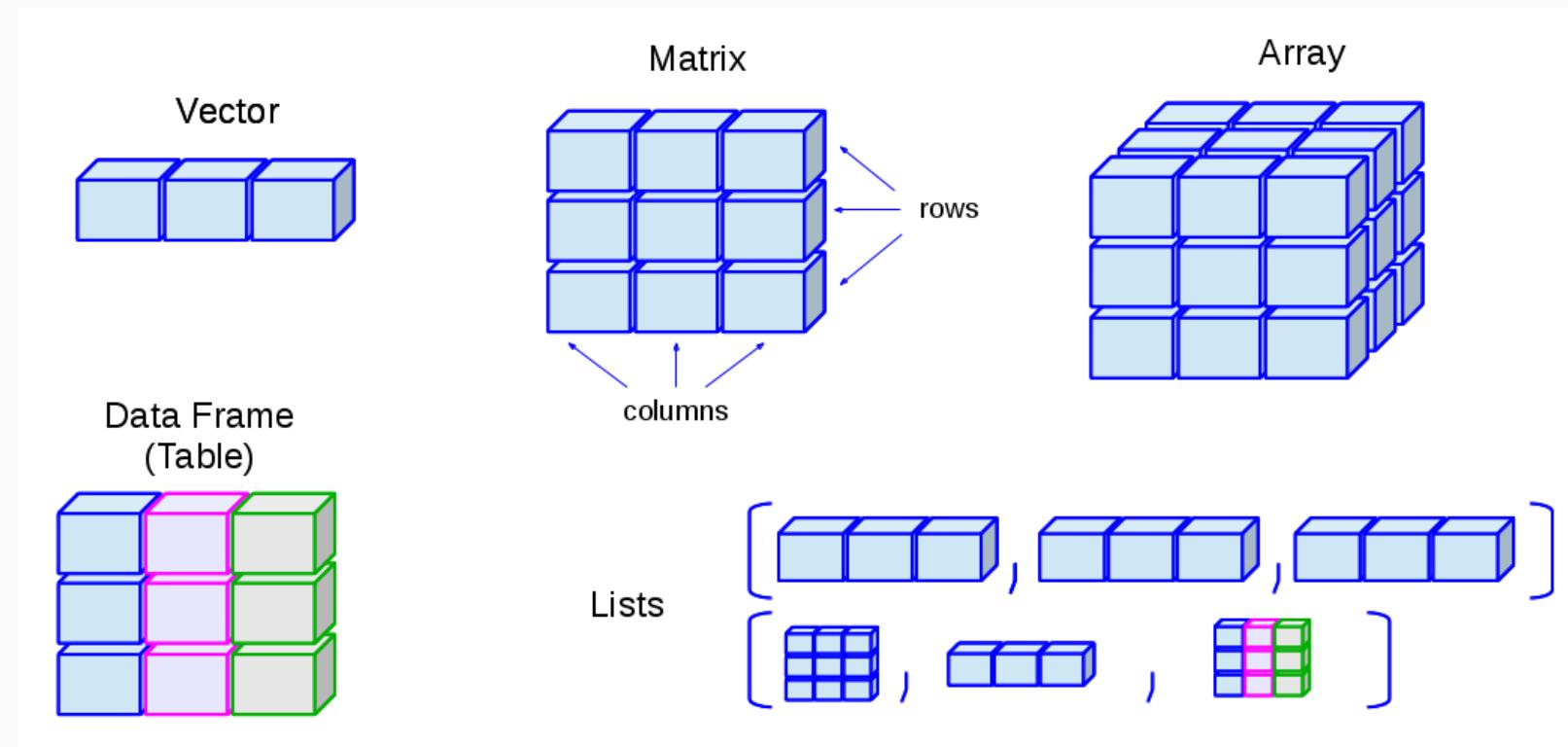
Resposta

```
# 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
```

2.3 Estrutura dos objetos

3. Matrix



2.3 Estrutura dos objetos

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

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)

2.3 Estrutura dos objetos

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

Ex.: espécies amostradas 5 locais

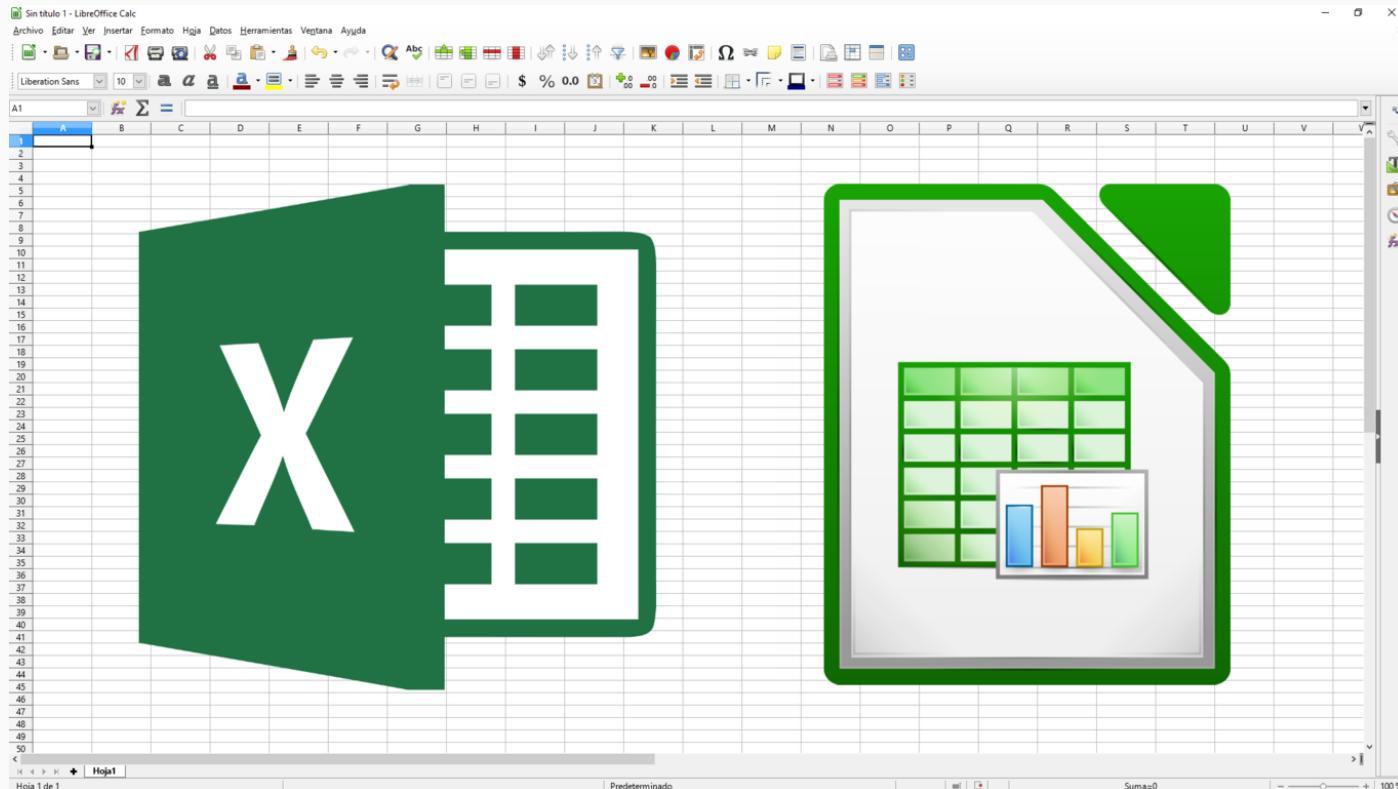
Matrix

		column (Y)					
		1	2	3	4	5	6
row (X)	1						
	2						
	3						
	4						
	5						

Esse formato lembra algo?

2.3 Estrutura dos objetos

3. Matrix: planilhas eletrônicas





2.3 Estrutura 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 - função 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
```

2.3 Estrutura 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 - função 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
```

2.3 Estrutura 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
```

2.3 Estrutura 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

02 : 00

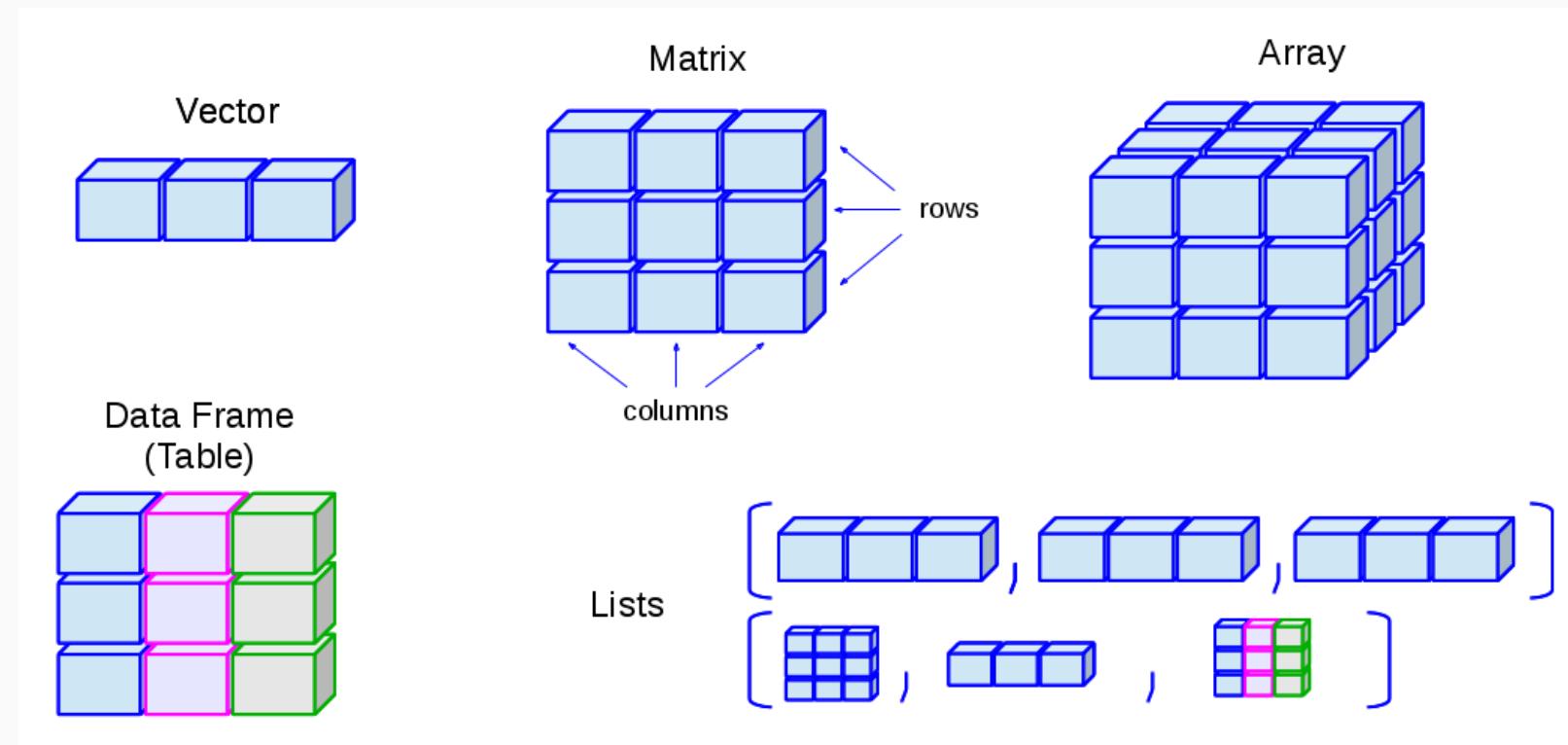
Exercício 07

Resposta

```
# solucao  
ma <- matrix(sample(0:10, 10000, rep = TRUE), nrow = 100, byrow = FALSE)  
ma
```

2.3 Estrutura dos objetos

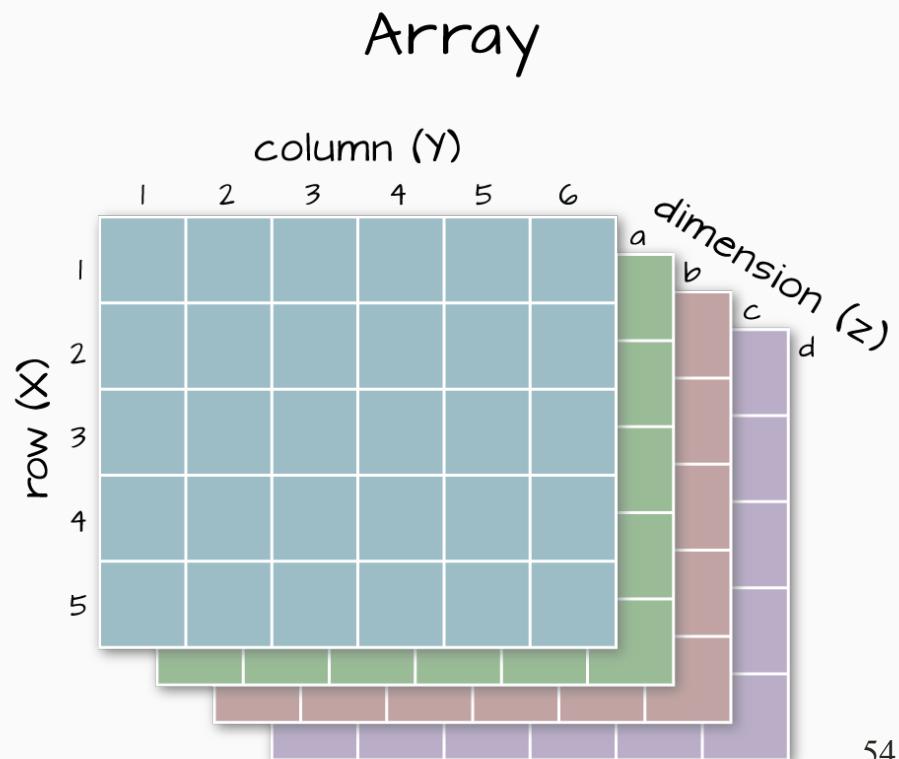
4. Array



2.3 Estrutura dos objetos

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

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





2.3 Estrutura dos objetos

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

1 Dispondo elementos

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
```

2.3 Estrutura dos objetos

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

1 Dispondo elementos

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

```
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,]
```

2.3 Estrutura dos objetos

Até o momento vimos **estruturas homogêneas**

VECTOR

```
1 2 3 4 5 6 7 8 9
```

ARRAY

```
1 2 3 4 5 6 7 8 9  
1 ...
```

```
1 2 3 4 5 6 7 8 9  
1 ...  
1
```

MATRIX

```
1 2 3 4 5 6 7 8 9  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...
```

```
1 2 3 4 5 6 7 8 9  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...  
1 ...
```

2.3 Estrutura dos objetos

Agora veremos as **estruturas heterogêneos**

HOMOGENEOUS

(elements are only 1 type)

Vector

Matrix

Array

HETEROGENEOUS

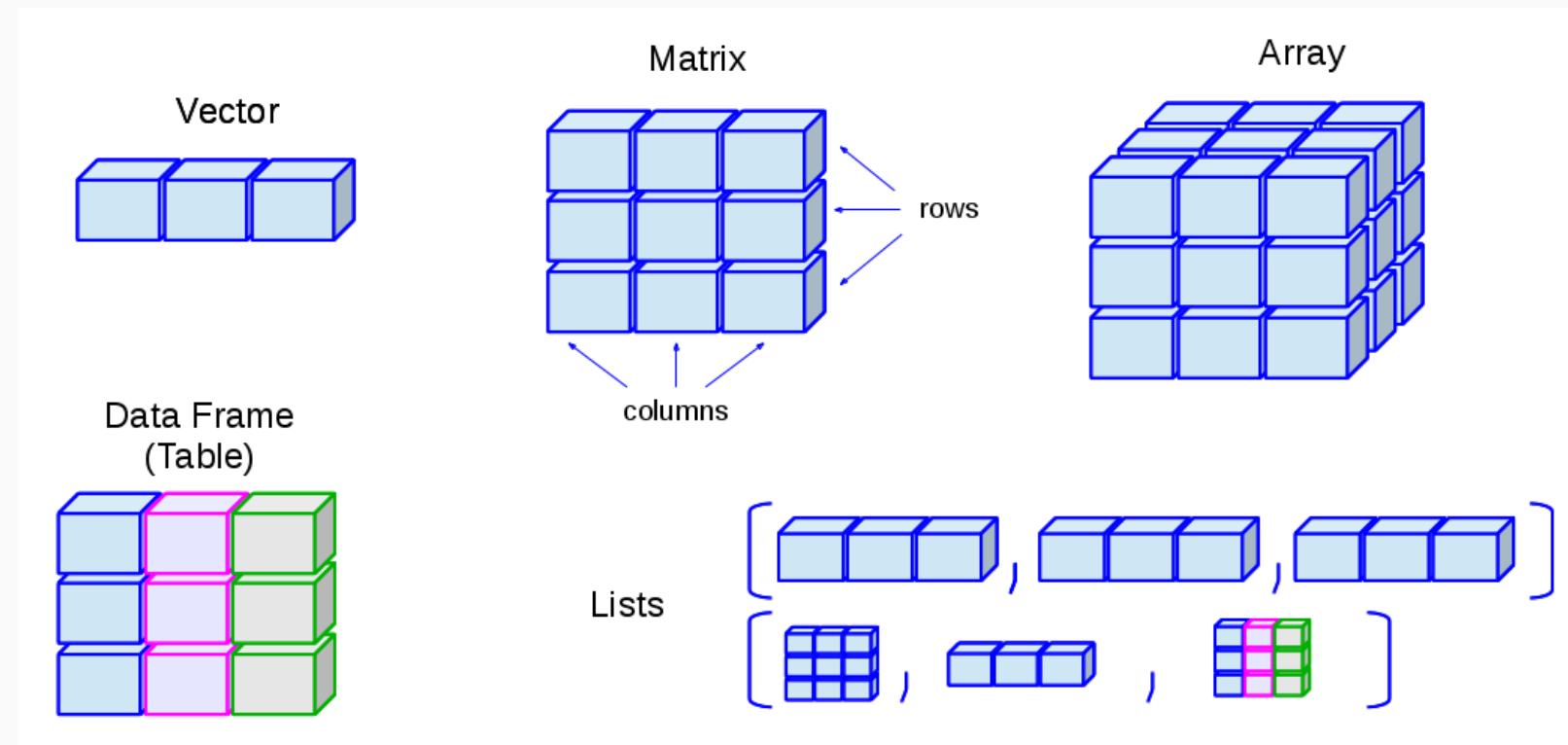
(elements can be different)

Dataframe

List

2.3 Estrutura dos objetos

5. Data frame



2.3 Estrutura dos objetos

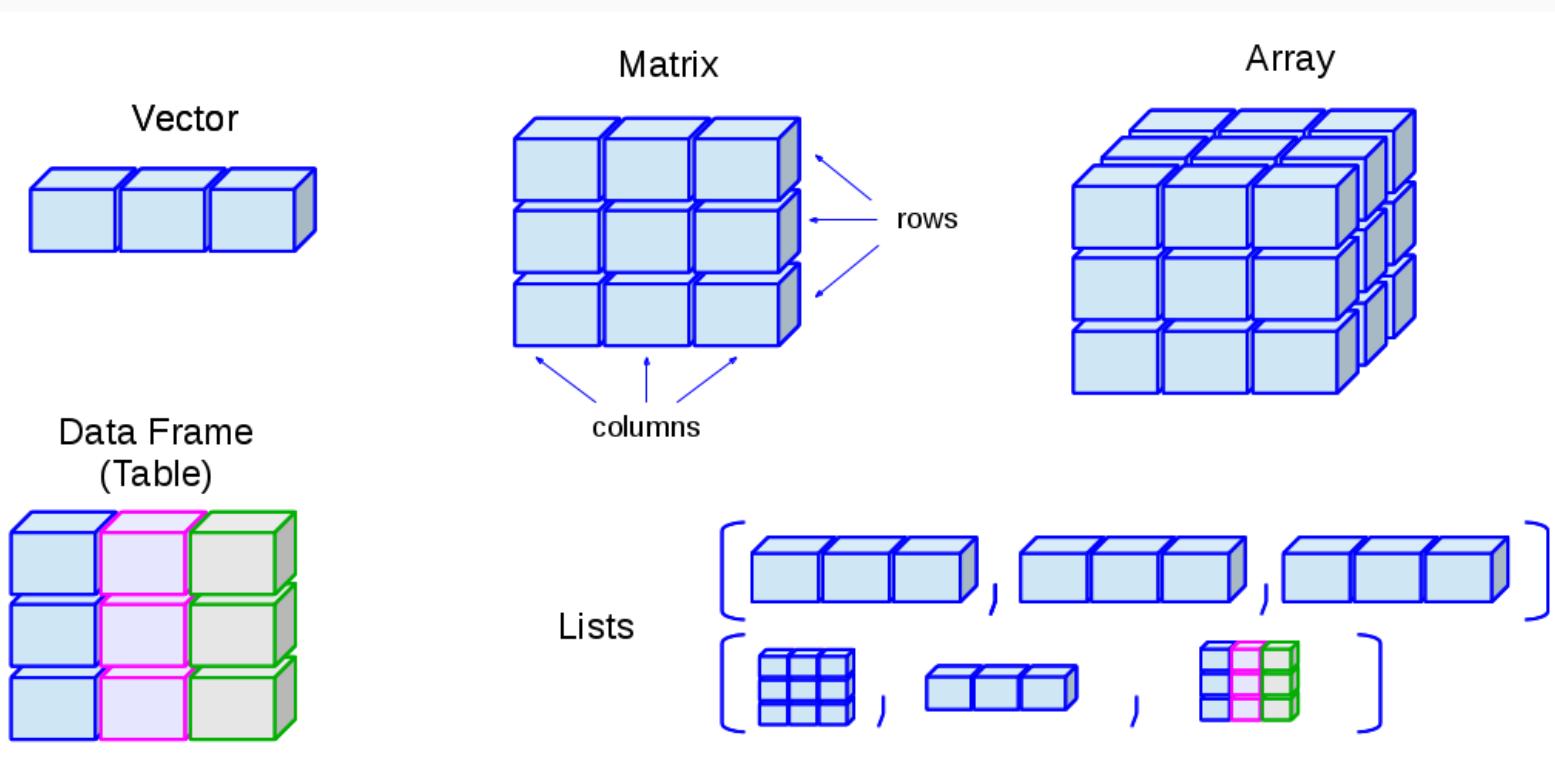
5. Data frame: heterogêneo (*mais de um modo*) e bidimensional (*duas dimensões*)

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)

2.3 Estrutura dos objetos

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



Esse formato também lembra algo?

2.3 Estrutura dos objetos

5. Data frame: planilhas eletrônicas



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



2.3 Estrutura 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 três 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
```

2.3 Estrutura 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
```

2.3 Estrutura 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 1
```

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

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

2.3 Estrutura 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  1  cont
```

```
str(df)
```

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

2.3 Estrutura 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" "1"  "1"
```

```
str(df_c)
```

```
##  chr [1:4, 1:3] "parcela_1" "parcela_2" "parcela_3" "parcela_4" "0"  "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

02:00

Exercício 08

Resposta

```
# solucao
id <- 1:50
id

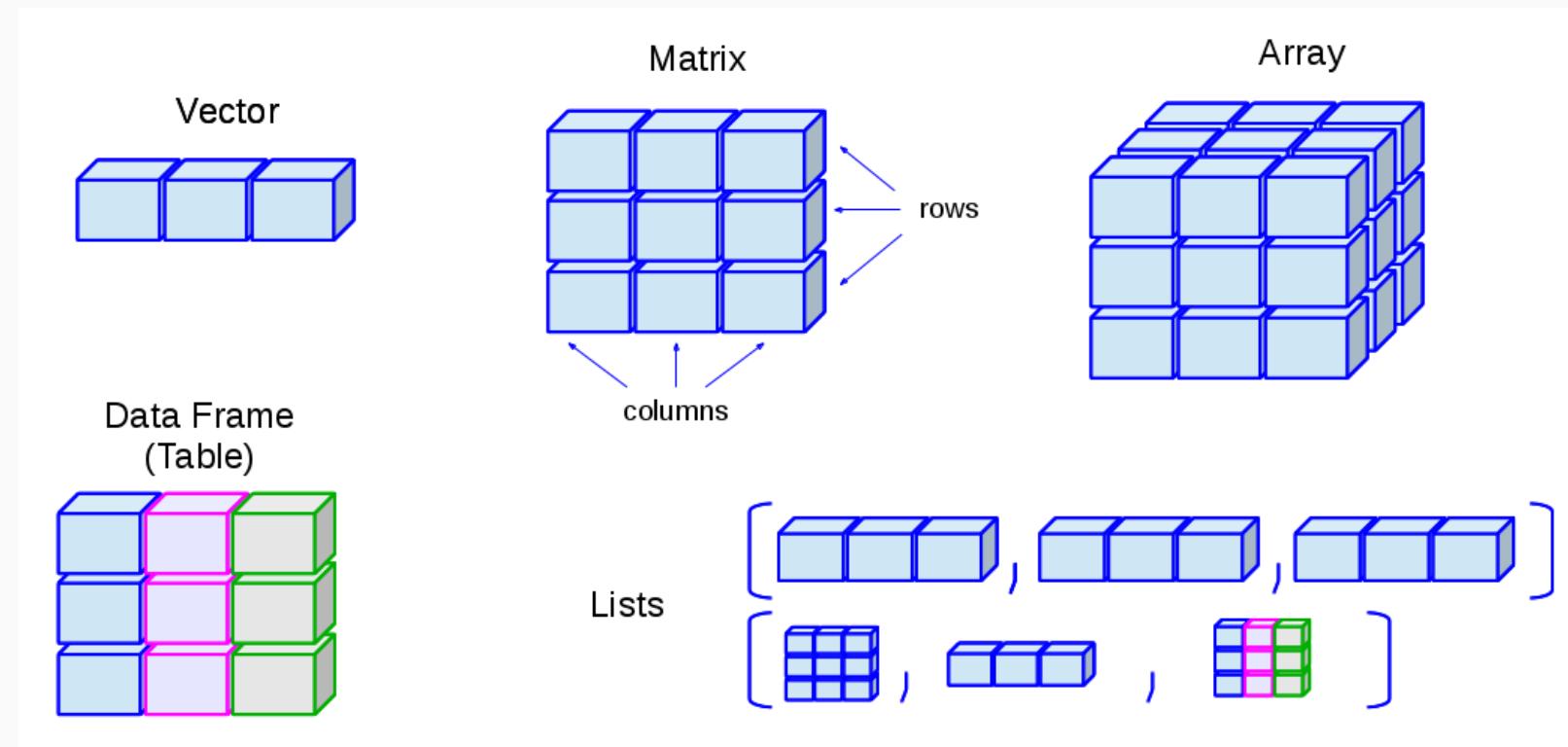
sp <- c(paste("sp", 1:9, sep = "0"), paste("sp", 10:50, sep = ""))
sp

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

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

2.3 Estrutura dos objetos

6. List



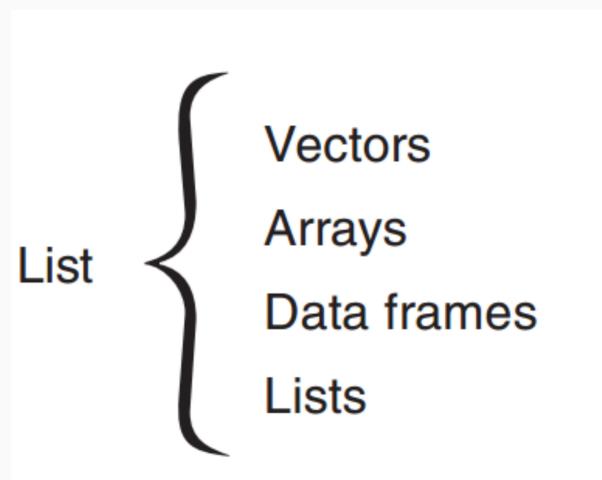
2.3 Estrutura dos objetos

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**



2.3 Estrutura dos objetos

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

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

Dúvidas?

Bora manejar isso tudo?

2.4 Manejo de dados (1D)

Vetor e fator

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

```
## indexacao []
# fixar a amostragem
set.seed(42)

# vetor
se <- seq(0, 5, .01)
se

## [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13 0
## [25] 0.24 0.25 0.26 0.27 0.28 0.29 0.30 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0
## [49] 0.48 0.49 0.50 0.51 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59 0.60 0.61 0
## [73] 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.80 0.81 0.82 0.83 0.84 0.85 0
## [97] 0.96 0.97 0.98 0.99 1.00 1.01 1.02 1.03 1.04 1.05 1.06 1.07 1.08 1.09 1
## [121] 1.20 1.21 1.22 1.23 1.24 1.25 1.26 1.27 1.28 1.29 1.30 1.31 1.32 1.33 1
## [145] 1.44 1.45 1.46 1.47 1.48 1.49 1.50 1.51 1.52 1.53 1.54 1.55 1.56 1.57 1
## [169] 1.68 1.69 1.70 1.71 1.72 1.73 1.74 1.75 1.76 1.77 1.78 1.79 1.80 1.81 1
## [183] 1.82 1.83 1.84 1.85 1.86 1.87 1.88 1.89 1.80 1.81 1.82 1.83 1.84 1.85 1
```

2.4 Manejo de dados (1D)

Vetor e fator

1. Indexação []

Selecionar elementos

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

```
## [1] 0.73
```

```
# seleciona os elementos de 10 a 50  
ve[1:5]
```

```
## [1] 0.48 4.84 3.20 1.52 0.73
```

```
# seleciona os elementos 10 e 100 e atribuir  
ve_sel <- ve[c(1, 10)]  
ve_sel
```

2.4 Manejo de dados (1D)

Vetor e fator

1. Indexação []

Retirar elementos

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

```
## [1] 0.48 4.84 3.20 1.52 0.73 2.27 1.45 1.21 5.00
```

```
# retirar os elementos 20 a 90  
ve[-(2:9)]
```

```
## [1] 0.48 1.27
```

```
# retirar os elementos 50 e 100 e atribuir  
ve_sub <- ve[-c(5, 10)]  
ve_sub
```

2.4 Manejo de dados (1D)

Vetor e fator

2 Seleção condicional: selecionar elementos por condições

```
# dois vetores
foo <- 42
bar <- 23
```

```
# operadores relacionais - saídas booleanas (TRUE ou FALSE)
foo == bar # igualdade
foo != bar # diferença
foo > bar # maior
foo >= bar # maior ou igual
foo < bar # menor
foo <= bar # menor ou igual
```

2.4 Manejo de dados (1D)

Vetor e fator

2 Seleção condicional

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

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

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

```
## [1] 4.84 3.20 1.52 2.27 1.45 1.21 5.00 1.27
```

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

```
## [1] 4.84 3.20 1.52 2.27 1.45 1.21 5.00 1.27
```

2.4 Manejo de dados (1D)

Vetor e fator

3 Funções de manejo: *max, min, range, length, sort, e round*

```
# maximo  
max(vé)
```

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

```
# minimo  
min(vé)
```

```
## [1] 0.48
```

2.4 Manejo de dados (1D)

Vetor e fator

3 Funções de manejo: *max, min, range, length, sort, e round*

```
# amplitude  
range(vé)
```

```
## [1] 0.48 5.00
```

```
# comprimento  
length(vé)
```

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

2.4 Manejo de dados (1D)

Vetor e fator

3 Funções de manejo: *max, min, range, length, sort e round*

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

```
## [1] 0.48 0.73 1.21 1.27 1.45 1.52 2.27 3.20 4.84 5.00
```

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

```
## [1] 5.00 4.84 3.20 2.27 1.52 1.45 1.27 1.21 0.73 0.48
```

2.4 Manejo de dados (1D)

Vetor e fator

3 Funções de manejo: *max, min, range, length, sort e round*

```
# arredondamento  
ve
```

```
## [1] 0.48 4.84 3.20 1.52 0.73 2.27 1.45 1.21 5.00 1.27
```

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

```
## [1] 0.5 4.8 3.2 1.5 0.7 2.3 1.4 1.2 5.0 1.3
```

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

```
## [1] 0 5 3 2 1 2 1 1 5 1
```

2.4 Manejo de dados (1D)

Vetor e fator

3 Funções de manejo: *any()*, *all()* e *which()*

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

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

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

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

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

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

2.4 Manejo de dados (1D)

Vetor e fator

3 Funções de manejo: *subset* e *ifelse*

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

```
## [1] 4.84 3.20 1.52 2.27 1.45 1.21 5.00 1.27
```

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

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

2.5 Manejo de dados (2D)

Matrizes e Data Frames

The diagram shows a 4x4 DataFrame with columns ID, items, store, and price. It highlights different selection methods:

- df[1,2]**: Selects row 1 in column 2 (the value "book"). A yellow dashed arrow points from the code to the cell.
- df[1:3, 3:4]**: Selects rows 1 to 3 and columns 3 to 4. A blue dashed arrow points from the code to the highlighted submatrix.
- df[1:2,]**: Selects rows 1 to 2. A green dashed arrow points from the code to the first two rows.
- df[,1]**: Selects Column 1. A red dashed arrow points from the code to the first column.

	ID	items	store	price
1	10	book	TRUE	2.5
2	20	pen	FALSE	8.0
3	30	textbook	TRUE	10.0
4	40	pencil_case	FALSE	7.0

2.5 Manejo de dados (2D)

Matrizes e Data Frames

1 Indexação []

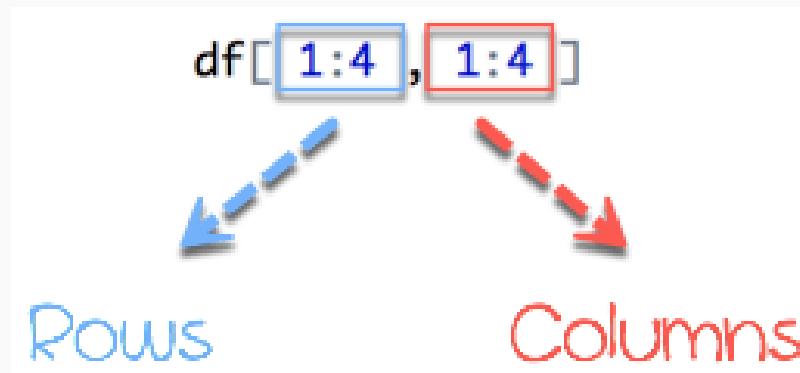
```
# matriz  
ma <- matrix(1:12, 4, 3)  
ma
```

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

2.5 Manejo de dados (2D)

Matrizes e Data Frames

1 Indexação []



2.5 Manejo de dados (2D)

Matrizes e Data Frames

1 Indexação []

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

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

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

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

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

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

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

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

2.5 Manejo de dados (2D)

Matrizes e Data Frames

1 Indexação []

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

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

```
ma_sel <- ma[1, c(1, 3)]  
ma_sel
```

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

2.5 Manejo de dados (2D)

Matrizes e Data Frames

1 Indexação []: resumindo

The diagram illustrates various ways to index a data frame with 4 rows and 5 columns. A red box highlights row 1, a yellow box highlights column 2, and a blue box highlights columns 3 to 4. Dashed arrows point from the code examples to the corresponding parts of the data frame.

	ID	items	store	price
1	10	book	TRUE	2.5
2	20	pen	FALSE	8.0
3	30	textbook	TRUE	10.0
4	40	pencil_case	FALSE	7.0

Select row 1 in column 2
df[1,2]

Select Rows 1 to 2
df[1:2,]

Select Column 1
df[,1]

Select Rows 1 to 3 and columns 3 to 4
df[1:3, 3:4]

2.5 Manejo de dados (2D)

Data Frames

1 Indexação \$

```
# 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
```

2.5 Manejo de dados (2D)

Data Frames

1 Indexação \$

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

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

```
df$abu
```

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

```
df$flo
```

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

2.5 Manejo de dados (2D)

Data Frames

1 Indexação \$

```
length(df$abu)
```

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

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

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

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

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

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

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

2.5 Manejo de dados (2D)

Data Frames

1 Indexação \$ e mudanças de colunas

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

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

```
# converter colunas
```

```
df$abu <- as.character(df$abu)
```

```
df$abu
```

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

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

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

2.5 Manejo de dados (2D)

Data Frames

1 Indexação \$ e mudanças de colunas

```
df$abu <- as.numeric(df$abu)
```

```
df$abu
```

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

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

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

2.5 Manejo de dados (2D)

Data Frames

1 Indexação \$ e adicionar uma coluna

```
set.seed(1)
df$abu2 <- sample(0:1, nrow(df), rep = TRUE)
```

```
df$abu2
```

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

```
df
```

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

2.5 Manejo de dados (2D)

Data Frames

2 Seleção condicional

Selecionar linhas = filtro do 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    0  
## 7      sp7   7 floresta    0  
## 8      sp8   8 floresta    0  
## 9      sp9   9 floresta    1  
## 10     sp10 10 floresta    1
```

2.5 Manejo de dados (2D)

Data Frames

2 Seleção condicional

Selecionar linhas = filtro do planilha eletrônica

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

```
##      sp abu        flo abu2  
## 1  sp1   1     campo     0  
## 3  sp3   3     campo     0  
## 4  sp4   4     campo     0  
## 6  sp6   6 floresta    0  
## 7  sp7   7 floresta    0  
## 8  sp8   8 floresta    0
```

2.5 Manejo de dados (2D)

Data Frames

2 Seleção condicional

Selecionar linhas = filtro do planilha eletrônica

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

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

2.5 Manejo de dados (2D)

Matrizes e Data Frames

3 Funções de visualização e manejo

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)

2.6 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

2.6 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
```

2.6 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
```

2.6 Valores faltantes e especiais

1 NA (Not Available)

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

```
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

```
any(is.na(df))
```

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

2.6 Valores faltantes e especiais

1 NA (Not Available)

Vamos retirar as linhas que possuem NA's

```
df_sem_na <- na.omit(df)  
df_sem_na
```

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

```
nrow(df)
```

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

```
nrow(df_sem_na)
```

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

2.6 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
```

2.6 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
```

```
log(-1)
```

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

2.6 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 presenca de na
is.na(ve)
```

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

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

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

2.6 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 grande  
10^310
```

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

2.6 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
```

2.7 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/minicurso-tidyverse/03_dados")
```

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

```
# verificar os arquivos
dir()
```

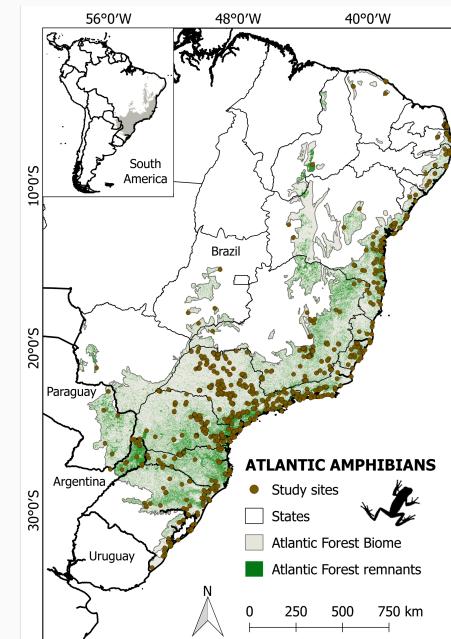
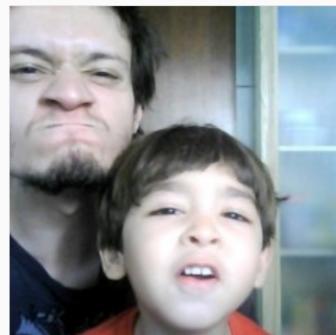
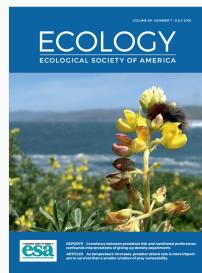
Vamos trabalhar com dados reais?



3.3 readr

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

Eu mesmo et al. (2018)



2.8 Importar dados

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

1. csv

2. txt

3. xlsx

2.8 Importar dados

Ler uma planilha eletrônica (.csv)

```
# ler uma planilha eletronica (.csv)
read.csv("ATLANTIC_AMPHIBIANS_sites.csv")
```

2.8 Importar dados

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

```
# ler e atribuir uma planilha eletronica (.csv) a um objeto  
da <- read.csv("ATLANTIC_AMPHIBIANS_sites.csv")
```

```
# ver os dados  
da
```

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

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

2.8 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
```

2.8 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)
da
```

2.9 Conferir e manejlar dados

Conjunto de funções para conferir os dados

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

2.9 Conferir e manejardados

head(): mostra as primeiras 6 linhas

```
head(da)
```

```
##           id reference_number species_number record sampled_habitat active_method
## 1 amp1001          1001             19      ab        fo,ll
## 2 amp1002          1002             16      co        fo,la,ll
## 3 amp1003          1002             14      co        fo,la,ll
## 4 amp1004          1002             13      co        fo,la,ll
## 5 amp1005          1003             30      co        fo,ll,br
## 6 amp1006          1004             42      co tp,pp,la,ll,is <NA>
##           period month_start year_start month_finish year_finish effort_months co
## 1 mo,da,tw,ni       9     2000                 1     2002            16 E
## 2 mo,da,tw,ni      12     2007                 5     2009            17 E
## 3 mo,da,tw,ni      12     2007                 5     2009            17 E
## 4 mo,da,tw,ni      12     2007                 5     2009            17 E
## 5 mo,da,ni          7     1988                 8     2001           157 E
## 6 <NA>              NA        NA                NA        NA            NA E
##           municipality                               site latitude longitude
## 1 Canto do Buriti Parque Nacional Serra das Confusões -8.680000  -43.421287 141
```

2.9 Conferir e manejardados

head(): mostra as primeiras 10 linhas

```
head(da, 10)
```

```
##          id reference_number species_number record sampled_habitat active_method
## 1  amp1001           1001             19      ab        fo, ll
## 2  amp1002           1002             16      co        fo, la, ll
## 3  amp1003           1002             14      co        fo, la, ll
## 4  amp1004           1002             13      co        fo, la, ll
## 5  amp1005           1003             30      co        fo, ll, br
## 6  amp1006           1004             42      co tp, pp, la, ll, is <NA>
## 7  amp1007           1005             23      co                  sp
## 8  amp1008           1005             19      co      sp, la, sw as, sb, <NA>
## 9  amp1009           1005             13      ab                  fo <NA>
## 10 amp1010           1006              1      ab                  fo <NA>
##          period month_start year_start month_finish year_finish effort_months
## 1 mo, da, tw, ni       9     2000                 1    2002            16
## 2 mo, da, tw, ni      12     2007                 5    2009            17
## 3 mo, da, tw, ni      12     2007                 5    2009            17
## 4 mo, da, tw, ni      12     2007                 5    2009            17
129 / 141
```

2.9 Conferir e manejear dados

tail(): mostra as últimas 6 linhas

```
tail(da)
```

```
##           id reference_number species_number record sampled_habitat active_mete
## 1158 amp2158                 1389            3      co          <NA>
## 1159 amp2159                 1389            9      co          <NA>
## 1160 amp2160                 1389            6      co          <NA>
## 1161 amp2161                 1389            1      co          <NA>
## 1162 amp2162                 1389            2      co          <NA>
## 1163 amp2163                 1389            2      co          <NA>
##           period month_start year_start month_finish year_finish effort_months co
## 1158 <NA>             NA        NA            NA        NA            NA        NA Arge
## 1159 <NA>             NA        NA            NA        NA            NA        NA Arge
## 1160 <NA>             NA        NA            NA        NA            NA        NA Arge
## 1161 <NA>             NA        NA            NA        NA            NA        NA Arge
## 1162 <NA>             NA        NA            NA        NA            NA        NA Arge
## 1163 <NA>             NA        NA            NA        NA            NA        NA Arge
##           municipality                               site latitude longitude coordinate
## 1158 Manuel Belgrano           Comandante Andresito -25.66944 -54.04556
```

2.9 Conferir e manejear dados

nrow(): mostra o número de linhas

```
nrow(da)
```

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

ncol(): mostra o número de colunas

```
ncol(da)
```

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

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

```
dim(da)
```

```
## [1] 1163    25
```

2.9 Conferir e manejar dados

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

```
rownames(da)
```

```
## [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "  
## [18] "18"  "19"  "20"  "21"  "22"  "23"  "24"  "25"  "26"  "27"  "  
## [35] "35"  "36"  "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "  
## [52] "52"  "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"  "61"  "  
## [69] "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"  "78"  "  
## [86] "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"  "  
## [103] "103" "104" "105" "106" "107" "108" "109" "110" "111" "112" "  
## [120] "120" "121" "122" "123" "124" "125" "126" "127" "128" "129" "  
## [137] "137" "138" "139" "140" "141" "142" "143" "144" "145" "146" "  
## [154] "154" "155" "156" "157" "158" "159" "160" "161" "162" "163" "  
## [171] "171" "172" "173" "174" "175" "176" "177" "178" "179" "180" "  
## [188] "188" "189" "190" "191" "192" "193" "194" "195" "196" "197" "  
## [205] "205" "206" "207" "208" "209" "210" "211" "212" "213" "214" "  
## [222] "222" "223" "224" "225" "226" "227" "228" "229" "230" "231" "  
## [239] "239" "240" "241" "242" "243" "244" "245" "246" "247" "248" "  
## [256] "256" "257" "258" "259" "260" "261" "262" "263" "264" "265" "
```

2.9 Conferir e manejear dados

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

```
colnames(da)
```

```
## [1] "id"                      "reference_number"      "species_number"
## [6] "active_methods"           "passive_methods"        "complementary_methods"
## [11] "year_start"                "month_finish"          "year_finish"
## [16] "state"                     "state_abbreviation"    "municipality"
## [21] "longitude"                 "coordinate_precision"   "altitude"
```

2.9 Conferir e manejear dados

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

```
str(da)
```

```
## 'data.frame':    1163 obs. of  25 variables:  
## $ id                  : Factor w/ 1163 levels "amp1001","amp1002",...: 1 2 3  
## $ reference_number    : int  1001 1002 1002 1002 1003 1004 1005 1005 1005 1005 ...  
## $ species_number      : int  19 16 14 13 30 42 23 19 13 1 ...  
## $ record               : Factor w/ 2 levels "ab","co": 1 2 2 2 2 2 2 2 2 1 1 .  
## $ sampled_habitat     : Factor w/ 210 levels "br","du","eu,la",...: 27 16 16 ...  
## $ active_methods       : Factor w/ 14 levels "as","as,qs","as,qs,tr",...: 1 1 ...  
## $ passive_methods      : Factor w/ 7 levels "ar","dr","ft",...: 4 4 4 4 NA NA ...  
## $ complementary_methods: Factor w/ 7 levels "ae","ae,in","ae,rr",...: NA NA NA ...  
## $ period                : Factor w/ 11 levels "da","da,ni","da,tw,ni",...: 7 7 ...  
## $ 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 ...  
## $ country                : Factor w/ 3 levels "Argentina","Brazil",...: 1 3 4 2 2 2 1 4 1 2 ...
```

2.9 Conferir e manejardados

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

```
summary(da)
```

```
##          id      reference_number species_number record sampled_habitat act
## amp1001: 1      Min.    :1001      Min.    : 1.00 ab:346   fo     :114   as
## amp1002: 1      1st Qu.:1096      1st Qu.: 7.00 co:817   pp     :108   as,s
## amp1003: 1      Median  :1204      Median  :13.00 tp      : 74   as,t
## amp1004: 1      Mean    :1196      Mean    :15.17 la      : 51   as,s
## amp1005: 1      3rd Qu.:1295      3rd Qu.:21.00 sw      : 49   sb
## amp1006: 1      Max.    :1389      Max.    :80.00 (Other) :666   (Oth)
## (Other) :1157                               NA's     :101   NA's
## complementary_methods period month_start year_start mont
## ae      :187      tw,ni    :269      Min.    : 1.000 Min.    :1940 Min.
## tp      : 11       mo,da,tw,ni:256      1st Qu.: 4.000 1st Qu.:2004 1st Q
## ae, tp :  8       da,tw,ni  :207      Median : 8.000 Median :2007 Media
## ae,in :  6       ni        :142      Mean   : 7.058 Mean   :2007 Mean
## ae,rr :  6       mo,ni    : 20      3rd Qu.:10.000 3rd Qu.:2011 3rd Q
## (Other):135
```

2.9 Conferir e manejar dados

Verificar a presença de NAs

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

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

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

```
##      [1] 4685 4686 4687 4688 4689 4690 4691 4692 4693 4711 4744 4749 4751 4780 4  
##     [24] 4790 4791 4792 4793 4794 4795 4796 4797 4798 4799 4852 4853 4854 4855 4  
##     [47] 5035 5036 5048 5055 5056 5057 5063 5211 5223 5224 5225 5227 5249 5326 5  
##     [70] 5531 5532 5533 5534 5566 5567 5568 5638 5639 5640 5641 5642 5665 5690 5  
##     [93] 5807 5808 5809 5810 5811 5812 5813 5814 5815 5821 5824 5825 5826 5827 5  
##    [116] 5837 5838 5839 5840 5841 5842 5843 5844 5845 5846 5847 5848 5849 5850 5  
##    [139] 5914 5925 5926 5966 5967 5968 5969 5970 5971 5972 5973 6022 6023 6024 6  
##    [162] 6125 6142 6162 6183 6186 6187 6189 6190 6236 6237 6238 6239 6240 6241 6  
##    [185] 6258 6259 6274 6275 6276 6277 6278 6279 6280 6281 6282 6283 6284 6285 6  
##    [208] 6309 6310 6311 6312 6313 6314 6315 6326 6366 6367 6368 6369 6370 6366 6371 6
```

2.9 Conferir e manejar dados

Retirar os NAs

```
da_na <- na.omit(da)
```

```
nrow(da)
```

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

```
nrow(da_na)
```

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

2.9 Conferir e manejardados

Subset das linhas

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

	id	reference_number	species_number	record	sampled_habitat	active_me
## 410	amp1410	1173		27	ab	fo, ll
## 411	amp1411	1174		14	ab	<NA>
## 412	amp1412	1175		10	co	pp
## 413	amp1413	1176		53	co	fo, is
## 414	amp1414	1177		24	co	fo, pp, la, sw, is
## 415	amp1415	1178		30	co	tp, pp, sw, ll, is, br
## 416	amp1416	1178		22	co	tp, pp, sw, ll, is, br
## 417	amp1417	1178		32	co	tp, pp, sw, ll, is, br
## 418	amp1418	1178		14	co	tp, pp, sw, ll, is, br
## 419	amp1419	1178		17	co	tp, pp, sw, ll, is, br
## 420	amp1420	1178		17	co	tp, pp, sw, ll, is, br
## 421	amp1421	1179		4	co	fo
## 422	amp1422	1179		4	co	fo

2.10 Exportar dados

Exportar uma tabela de dados na pasta do diretório

Planilha eletrônica (.csv)

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

Planilha de texto (.txt)

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

Planilha eletrônica (.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.netlify.com

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