

XXX Semana de Estudos da Ecologia

Introdução à linguagem R: manipulação e visualização de dados

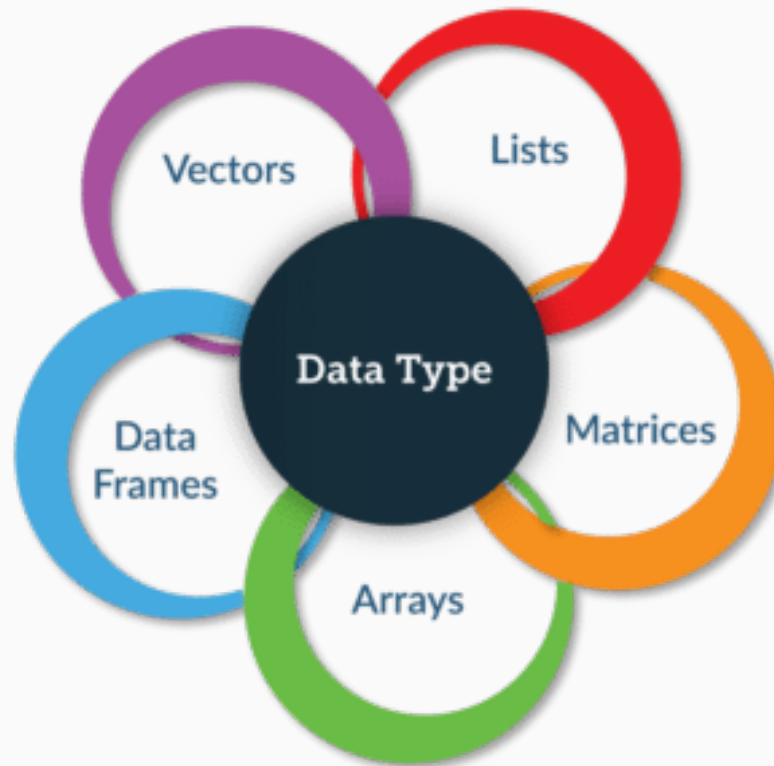
3 Estrutura e manipulação de dados

Maurício Vancine

Helena Oliveira

Lucas Almeida

06/11/2019



3 Estrutura e manejo de dados

Tópicos

- 3.1 Atributos dos objetos
- 3.2 Modos dos objetos (*numeric, character e logical*)
- 3.3 Estrutura dos objetos (*vector, factor, matrix, array, data frame e list*)
- 3.4 Manejo de dados unidimensionais
- 3.5 Manejo de dados bidimensionais
- 3.6 Valores faltantes e especiais
- 3.7 Diretório de trabalho
- 3.8 Importar dados
- 3.9 Conferir e manejar dados importados
- 3.10 Exportar dados

3 Estrutura e manejo de dados

Script

```
script_aula_03.R
```

3.1 Atributos dos objetos

Atribuição

palavra <- dados

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

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

3.1 Atributos dos objetos

Atributos dos objetos no R

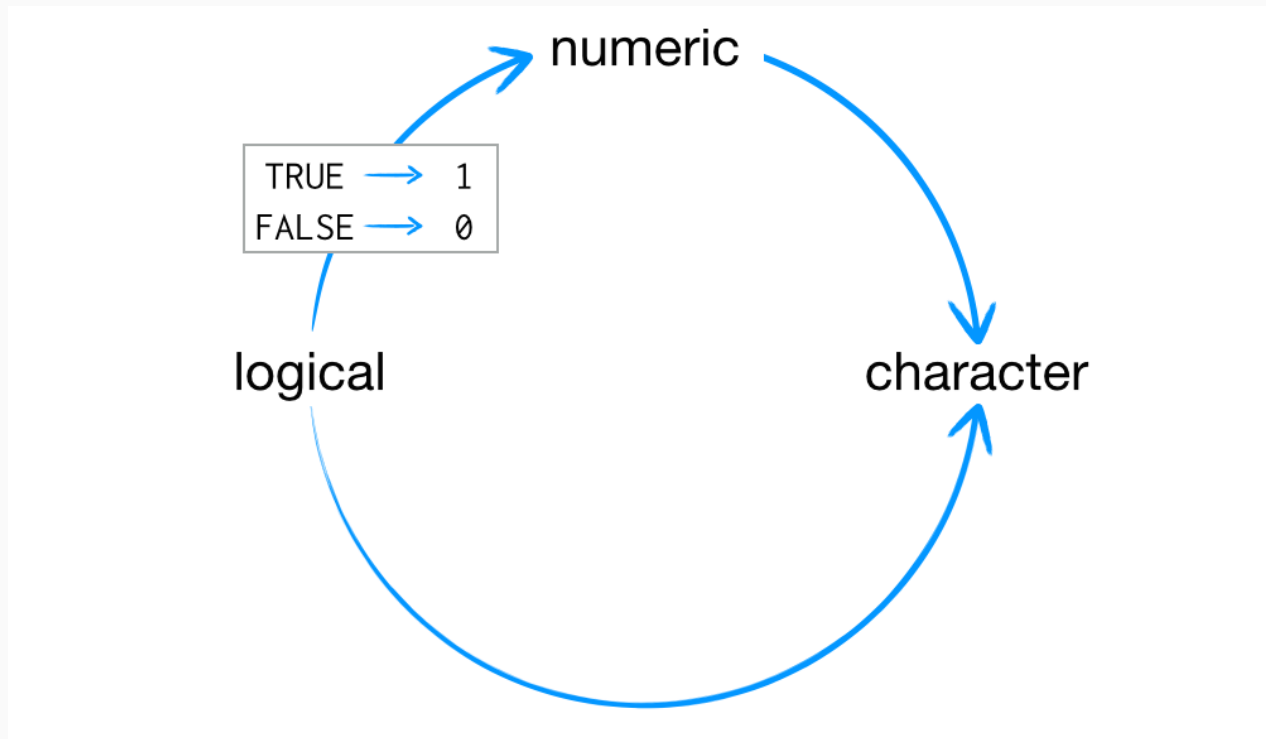
Objetos possuem **três características**:

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

3.2 Modos dos objetos

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

Natureza dos **elementos** que compõem os objetos



3.2 Modos dos objetos

1. **Numeric**: números inteiros ou decimais

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

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

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

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


3.2 Modos dos objetos

2. Character: texto

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

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

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

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

3.2 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"
```

3.2 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*

Dúvidas?

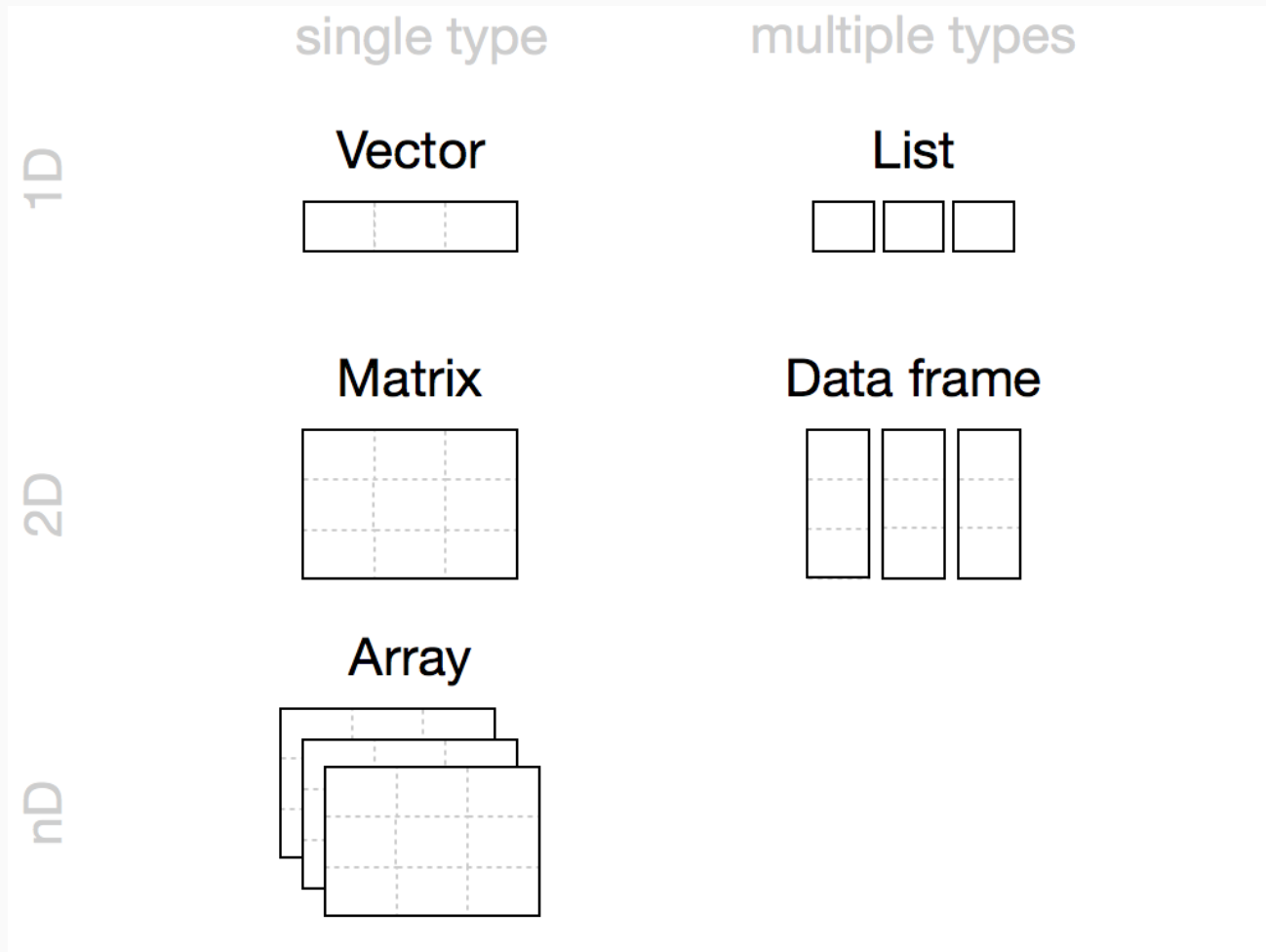
3.3 Estrutura dos objetos

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
<u>nd</u>	Array	

3.3 Estrutura dos objetos

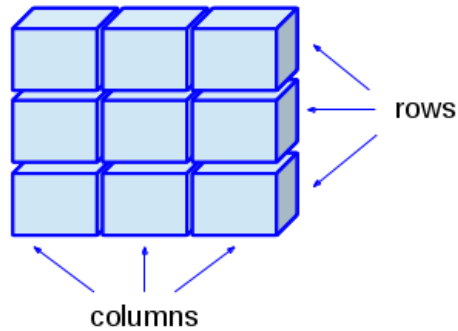


3.3 Estrutura dos objetos

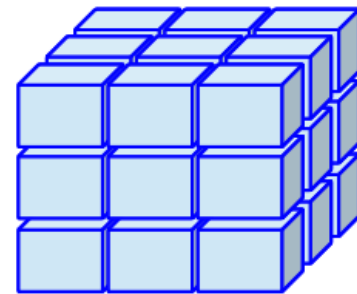
Vector



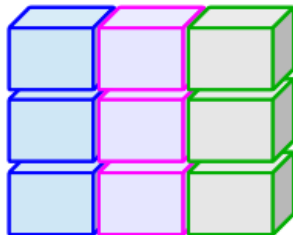
Matrix



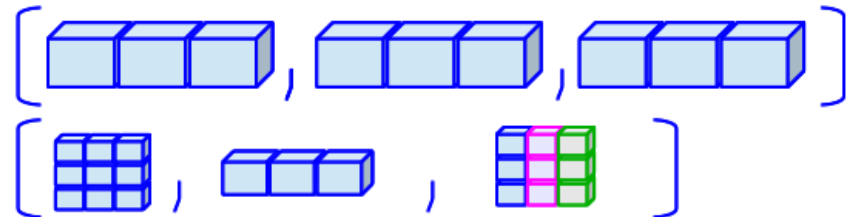
Array



Data Frame
(Table)



Lists



3.3 Estrutura dos objetos

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

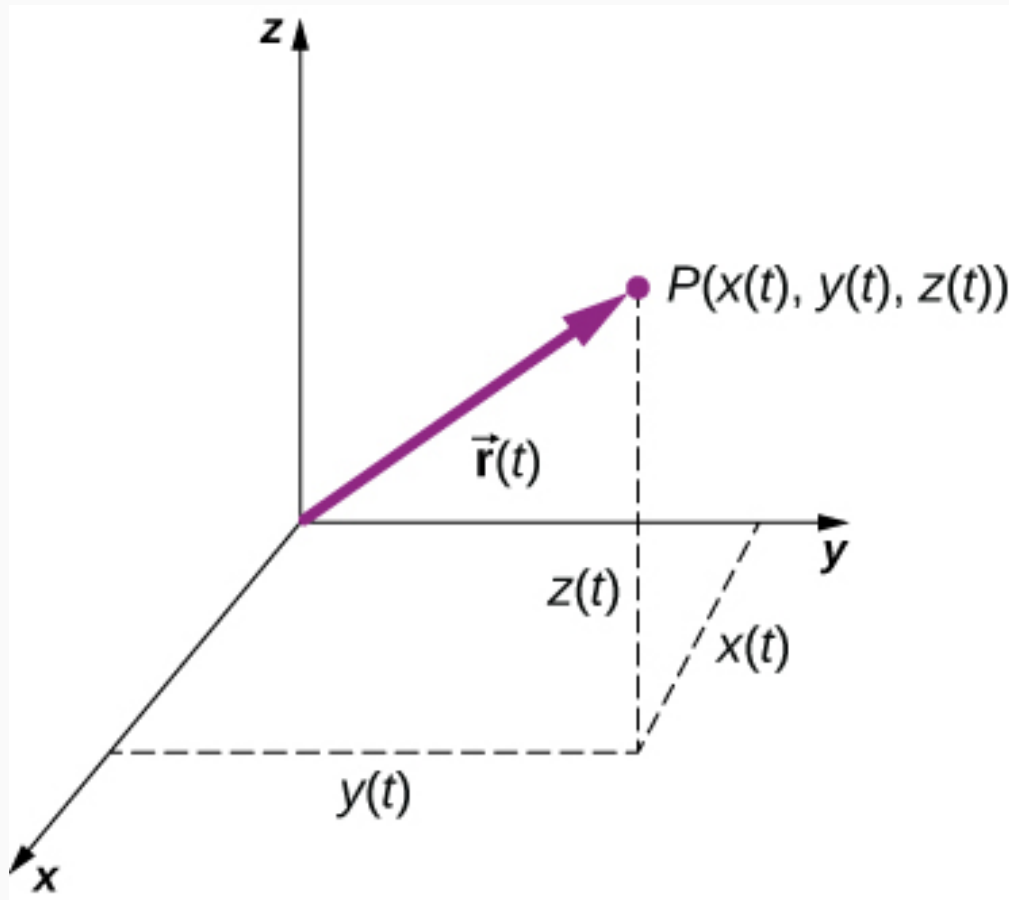
O **vetor** representa medidas de uma **variável quantitativa** (discretas ou contínuas) ou **descrição** (informações em texto)

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}

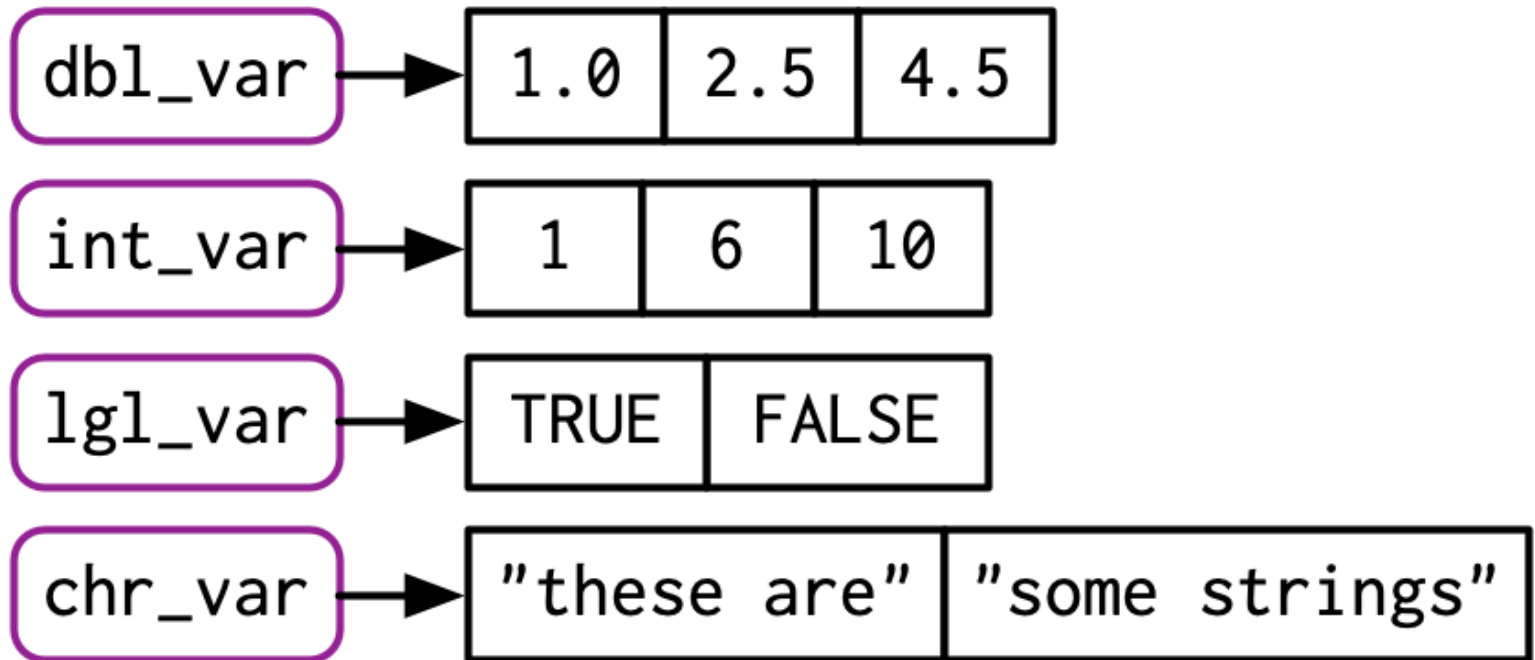
3.3 Estrutura dos objetos

Não me refiro exatamente ao vetor da matemática



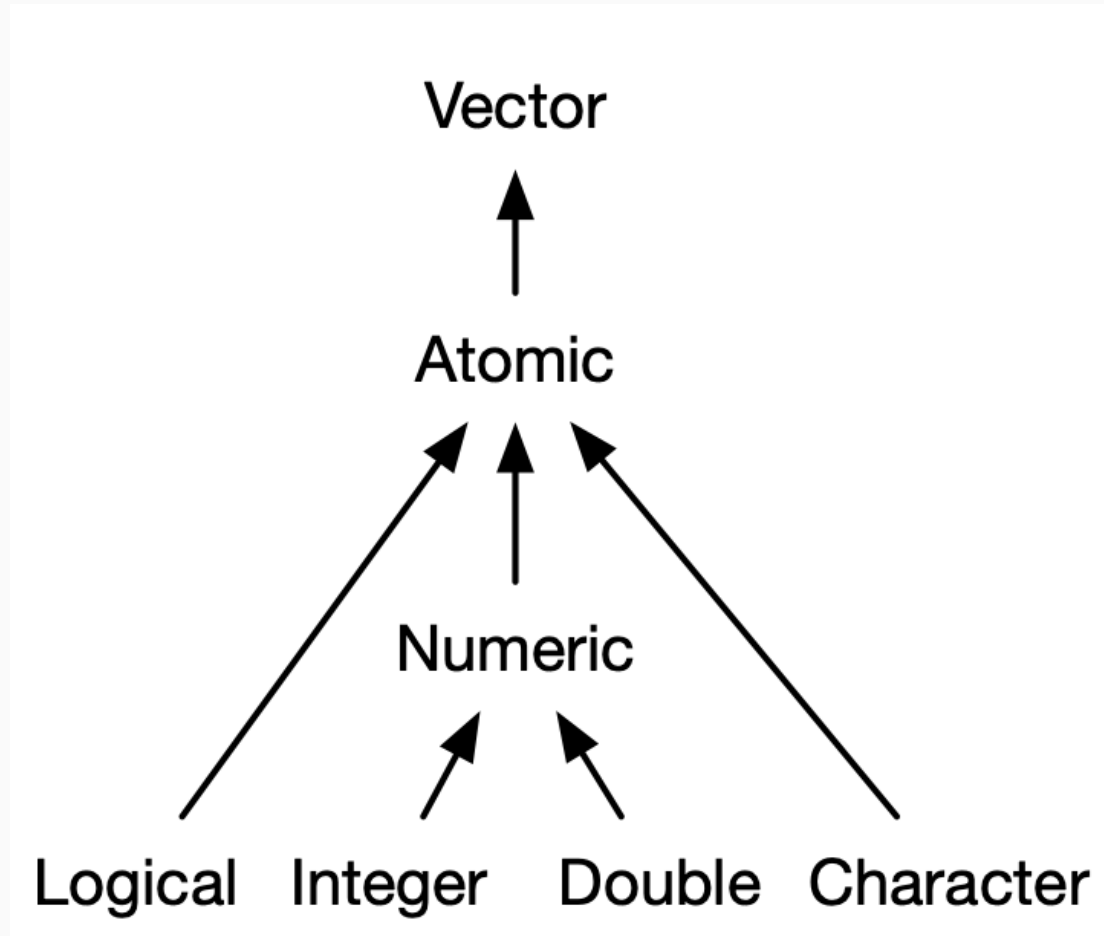
3.3 Estrutura dos objetos

Sequência de elementos



3.3 Estrutura dos objetos

Tipos



3.3 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"
```

3.3 Estrutura dos objetos

Há diversas formas de se criar um **v 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
```

3.3 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"
```

3.3 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"
```

3.3 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] 94 6 72 86 97 39 31 81 50 34
```

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

```
## [1] 4 5 6 3 9 4 6 9 9 7 3 8 9 3 7 3 7 6 10 5 5 8 3  
## [24] 10 2 10 2 10 6 4 1 6 3 8 3 8 1 7 7 7 10 6 7 10 5 6  
## [47] 8 5 7 4 3 9 7 6 10 9 7 2 3 8 4 7 4 1 8 4 9 8 6  
## [70] 4 8 3 4 4 6 1 10 4 9 7 8 5 2 6 9 8 10 4 5 7 1 8  
## [93] 8 10 9 8 2 5 9 7
```


Exercícios

Exercício 06

Vector

Criem uma sequência de 0 à 50, espaçada de 5 em 5.

Atribuem à um objeto chamado "seq_50"

Repitam os elementos desse objeto 10 vezes sequencialmente, atribuindo ao objeto "seq_50_rep_times"

```
# solucao  
seq_50 <- seq(0, 50, by = 5)  
seq_50
```

```
## [1] 0 5 10 15 20 25 30 35 40 45 50
```

```
seq_50_rep_times <- rep(seq_50, times = 10)  
seq_50_rep_times
```

Exercício 07

Vector

Escolham números para jogar na mega-sena

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

```
mega_num <- sample(1:60, 6, replace = FALSE)  
mega_num
```

```
## [1]  7 58 10 24 54 23
```

Exercício 08

Vector

Einstein disse que Deus não joga dados, mas o R joga!

Simulem o resultado de 25 jogadas de um dado e atribuem a um objeto

```
dado_25 <- sample(1:6, 25, rep = TRUE)
dado_25
```

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

Dúvidas?

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

3.3 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
```

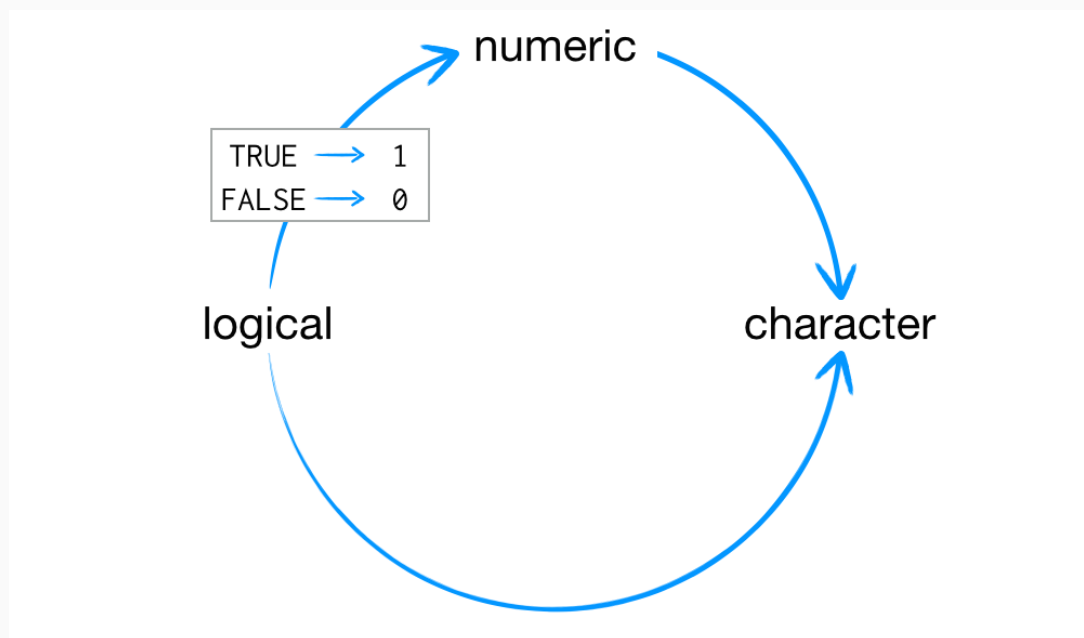
```
## [1] "1"      "a"      "TRUE"
```

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()
```

Dúvidas?

3.3 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. 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}

3.3 Estrutura dos objetos

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



3.3 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"
```

3.3 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"
```

3.3 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"
```

3.3 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 09

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:

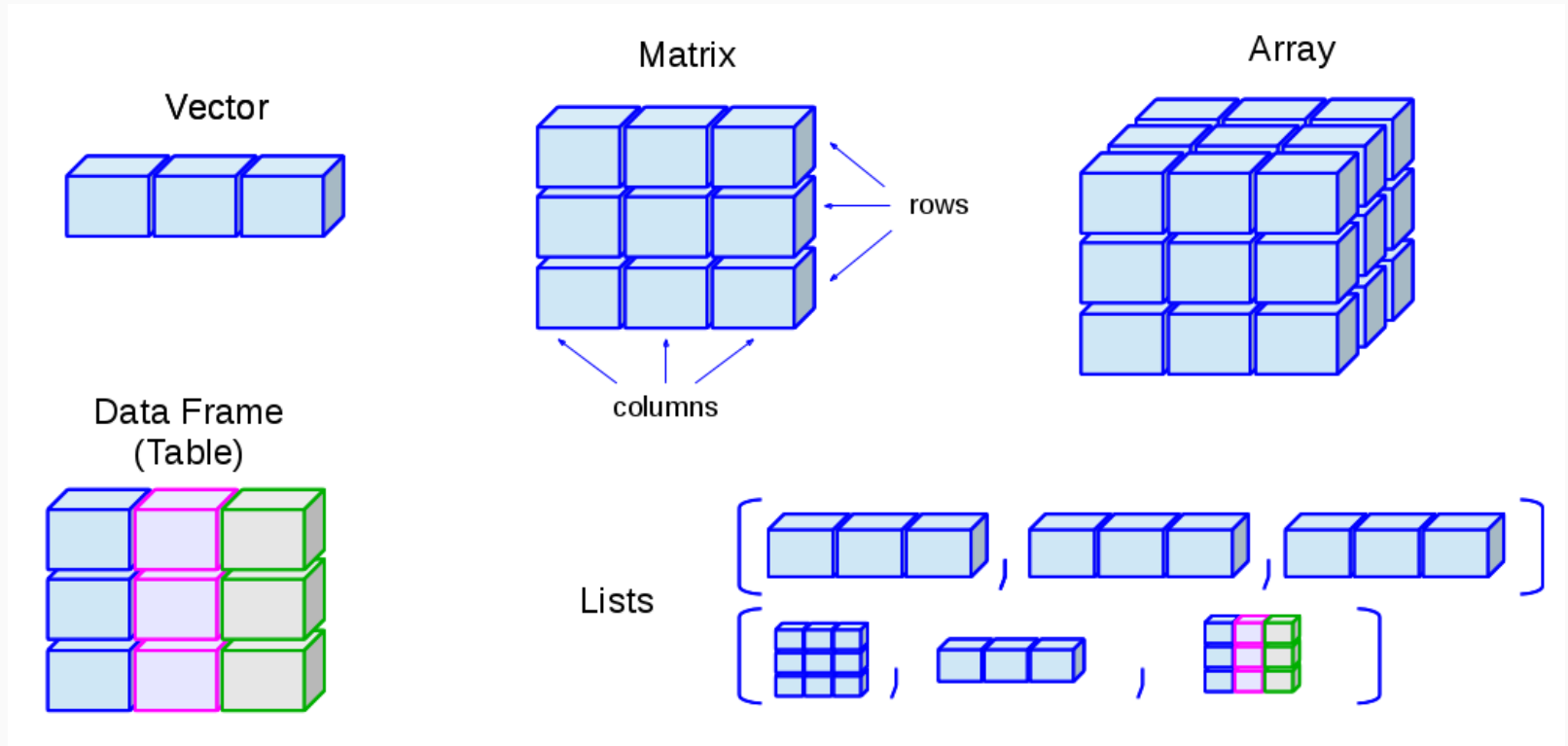
```
cont, cont, cont, . . . . , cont, trat, trat, . . . . , trat
```

```
# 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
```

3.3 Estrutura dos objetos

3. Matrix



3.3 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)

3.3 Estrutura dos objetos

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

Ex.: espécies amostradas 5 locais

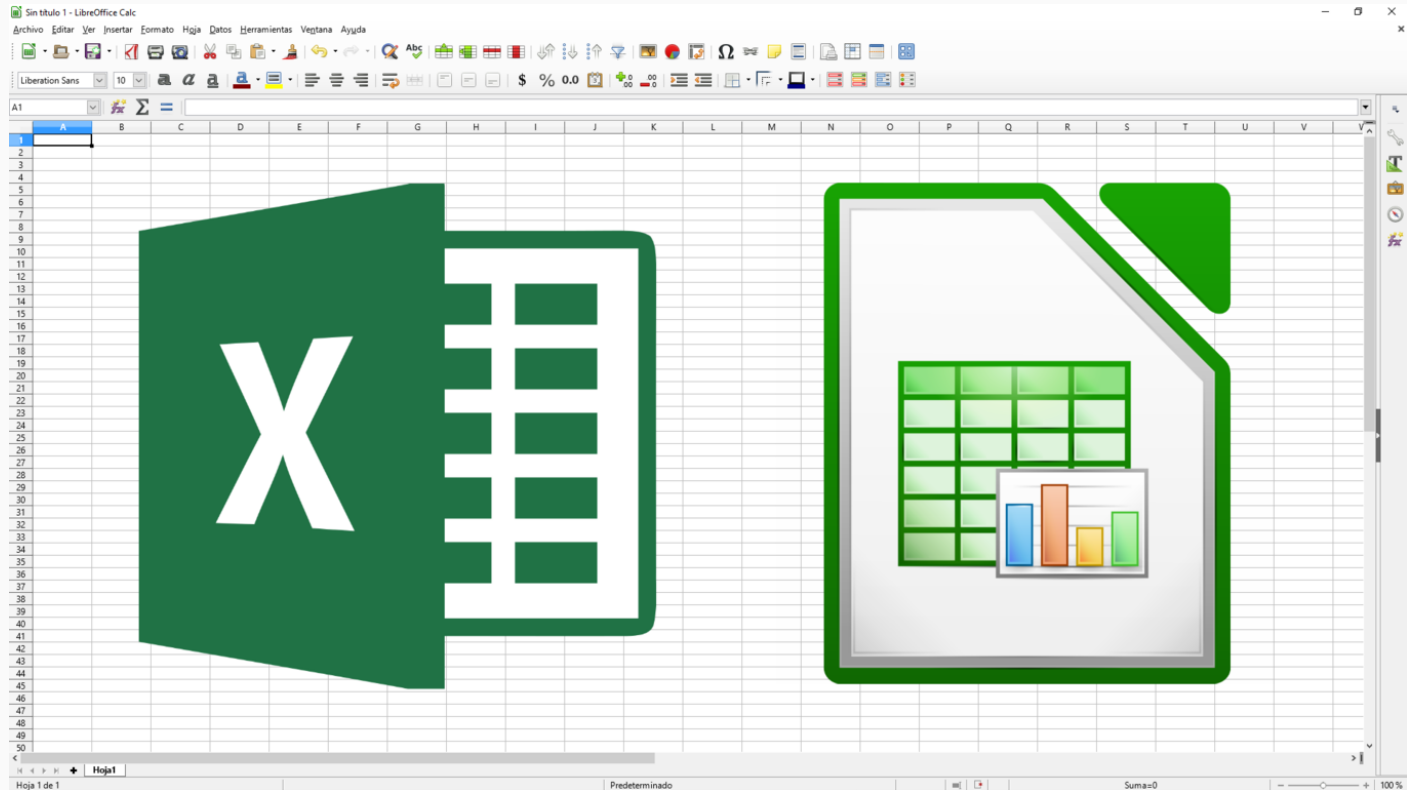
Matrix

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

Esse formato lembra algo?

3.3 Estrutura dos objetos

3. Matrix: planilhas eletrônicas





3.3 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 - 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
```

3.3 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 - 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
```

3.3 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
```

3.3 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 10

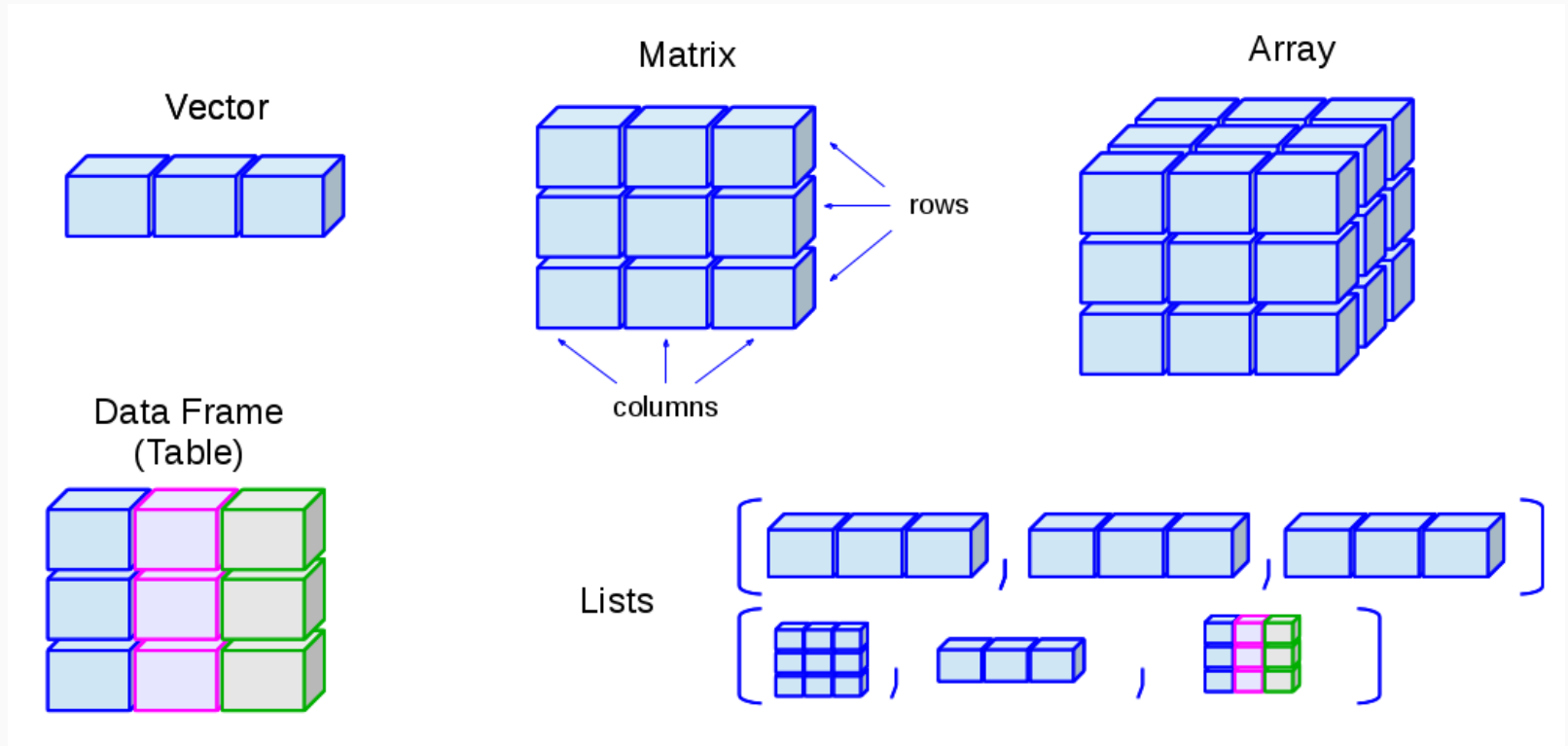
Matrix

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

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

3.3 Estrutura dos objetos

4. Array

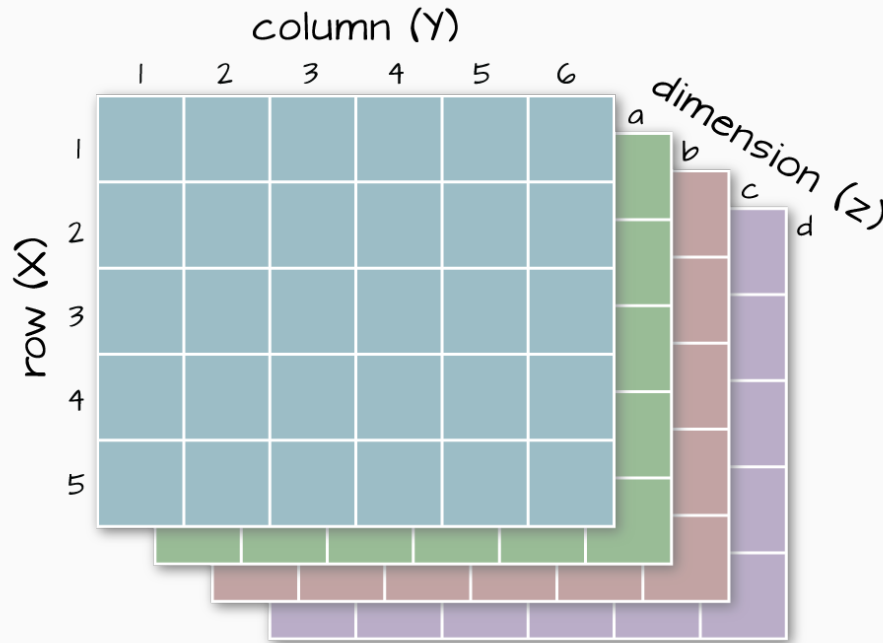


3.3 Estrutura dos objetos

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

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

Array





3.3 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
```

3.3 Estrutura dos objetos

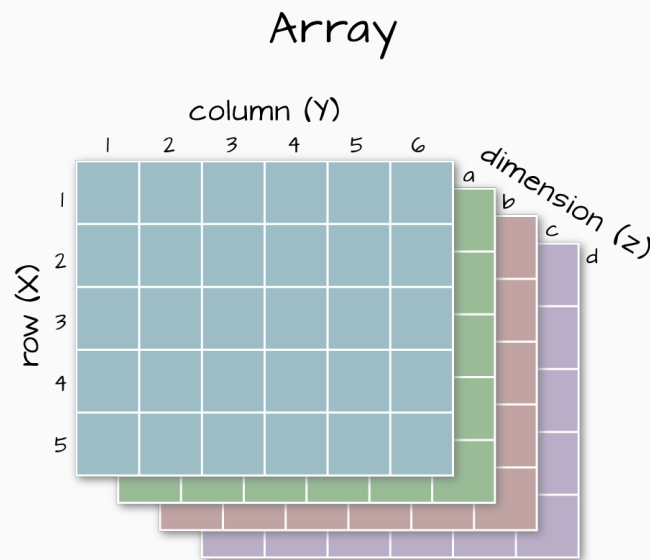
Há **uma forma** de se construir um **array** no R:

1 Dispondo elementos

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

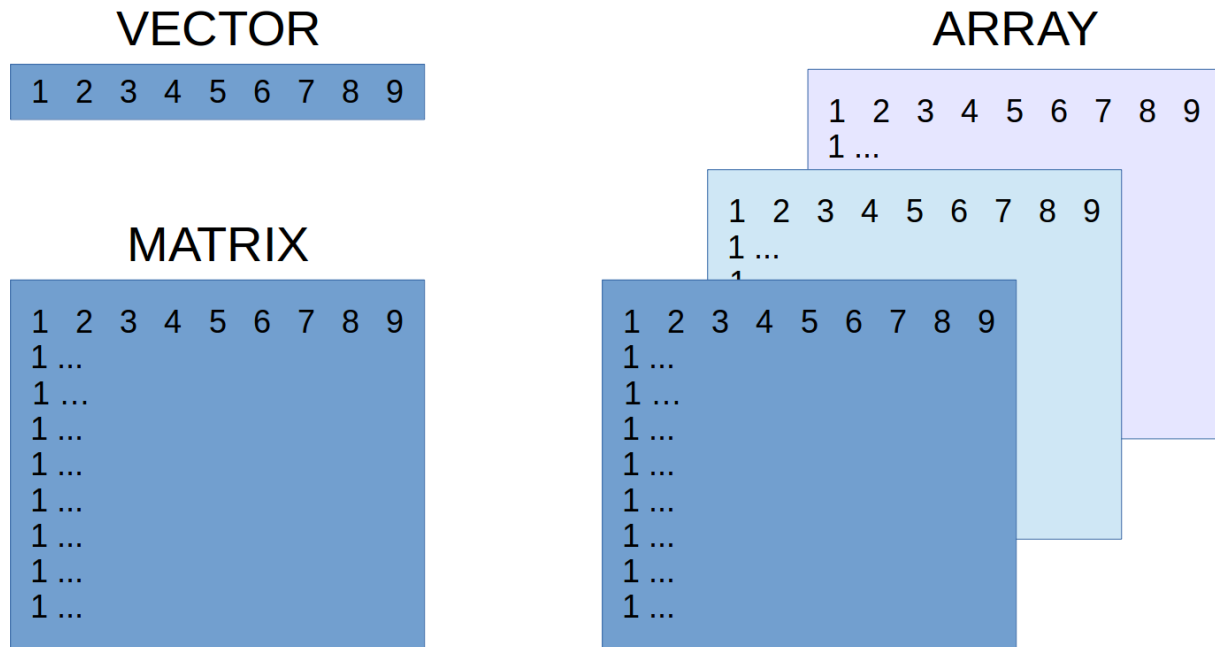
```
ar <- array(data = ve, dim = c(2, 2, 2))  
ar
```

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



3.3 Estrutura dos objetos

Até o momento vimos **estruturas homogêneas**



3.3 Estrutura dos objetos

Agora veremos as **estruturas heterogêneas**

HOMOGENEOUS
(elements are only 1 type)

Vector

Matrix

Array

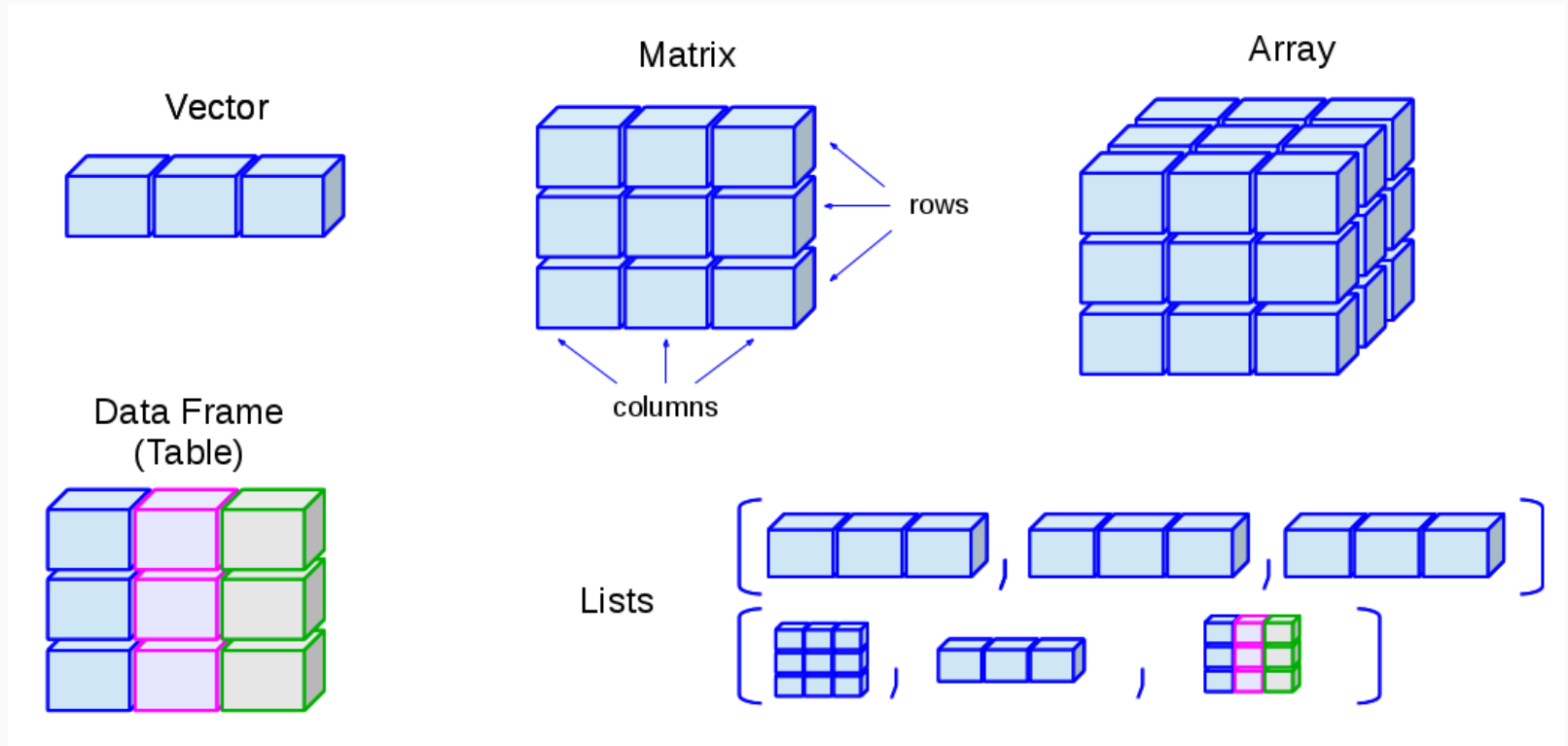
HETEROGENEOUS
(elements can be different)

Dataframe

List

3.3 Estrutura dos objetos

5. Data frame



3.3 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)

3.3 Estrutura dos objetos

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

Vector



- 1 column or row of data
- 1 type (numeric or text)

Matrix



- multiple columns and/or rows of data
- 1 type (numeric or text)

Data Frame

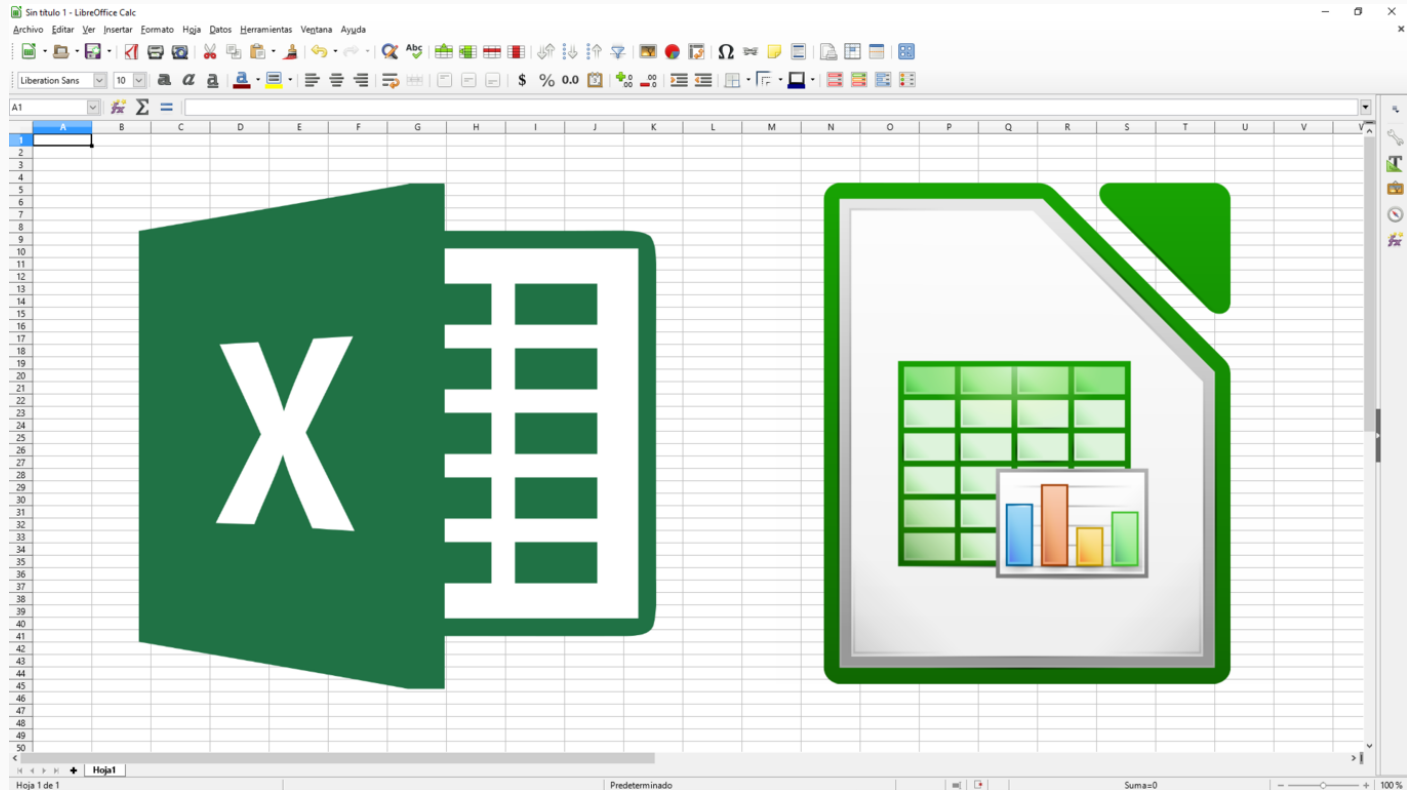


- multiple columns and/or rows of data
- multiple types

Esse formato também lembra algo?

3.3 Estrutura dos objetos

5. Data frame: planilhas eletrônicas



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



3.3 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
```

3.3 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
```

3.3 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] 1 1 0 0
```

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

```
## [1] trat trat cont cont
```

```
## Levels: cont trat
```

3.3 Estrutura dos objetos

Qual a diferença?

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

```
##           pa pe  tr
## 1 parcela_1  1 trat
## 2 parcela_2  1 trat
## 3 parcela_3  0 cont
## 4 parcela_4  0 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  1 1 0 0
## $ tr: Factor w/ 2 levels "cont","trat": 2 2 1 1
```


3.3 Estrutura dos objetos

Qual a diferença?

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

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

```
str(df_c)
```

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

Exercícios

Exercício 11

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

Exercício 11

Data frame

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

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

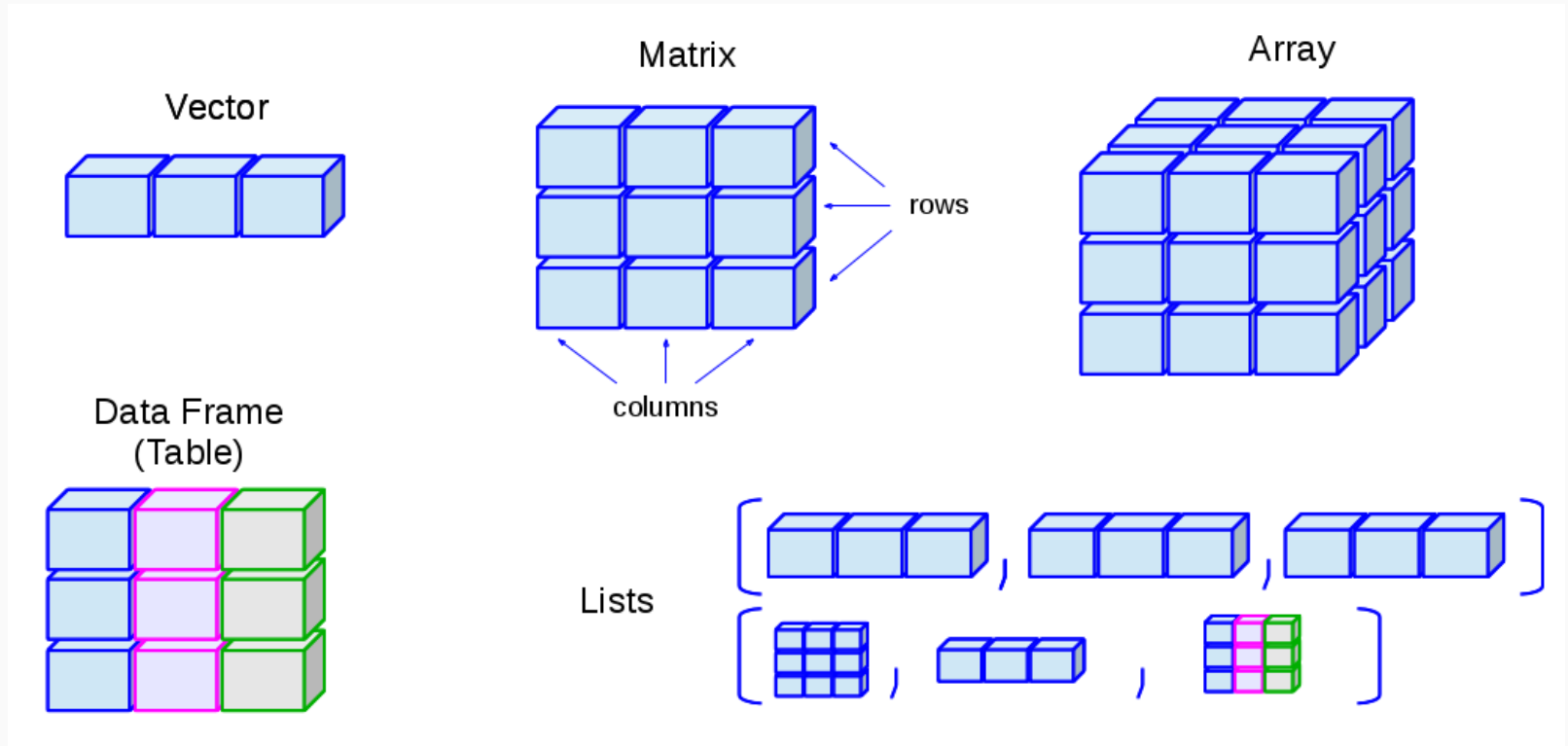
sp <- c(paste("sp", 1:9, sep = "0"), paste("sp", 10:50, sep = ""))
sp

ab <- sample(0:5, 50)
ab

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

3.3 Estrutura dos objetos

6. List



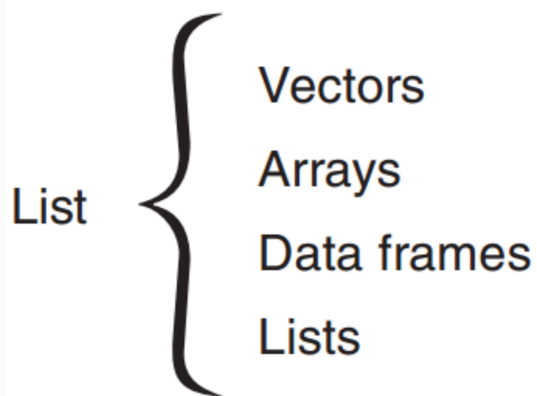
3.3 Estrutura dos objetos

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**



3.3 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
```

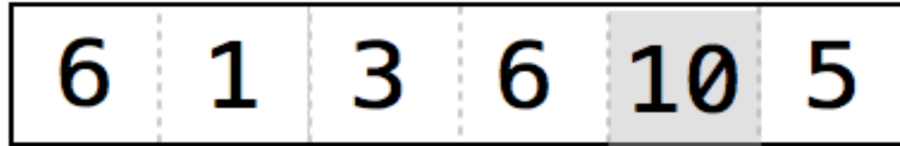
```
## [[1]]
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## [[2]]
##  [1] 1
## Levels: 1
##
## [[3]]
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
```

Dúvidas?

Bora manejar isso tudo?

3.4 Manejo de dados unidimensionais

Indexação []: acessa elementos de objetos



`vec[5]`

3.4 Manejo de dados unidimensionais

Indexação []: acessa elementos de objetos

Vetor e fator

```
## indexacao []  
# vetor  
vec <- c(6, 1, 3, 6, 10, 5)  
vec
```

```
## [1] 6 1 3 6 10 5
```

3.4 Manejo de dados unidimensionais

Indexação []: acessa elementos de objetos

Vetor e fator

Selecionar elementos

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

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

```
# seleciona os elementos do primeiro ao quinto  
vec[1:5]
```

```
## [1] 6 1 3 6 10
```

```
# seleciona o primeiro e o sexto elemento  
vec_sel <- vec[c(1, 6)]  
vec_sel
```

3.4 Manejo de dados unidimensionais

Indexação []: acessa elementos de objetos

Vetor e fator

Retirar elementos

```
# retira o quinto elemento  
vec[-5]
```

```
## [1] 6 1 3 6 5
```

```
# retira os elementos do primeiro ao quinto  
vec[-(1:5)]
```

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

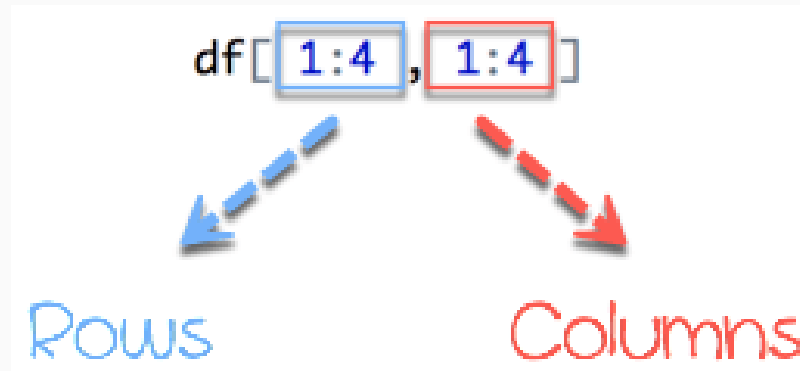
```
# retira o primeiro e o sexto elemento  
vec_sub <- vec[-c(1, 6)]  
vec_sub
```

Dúvidas?

3.5 Manejo de dados bidimensionais

Indexação []

Matrizes e Data Frames



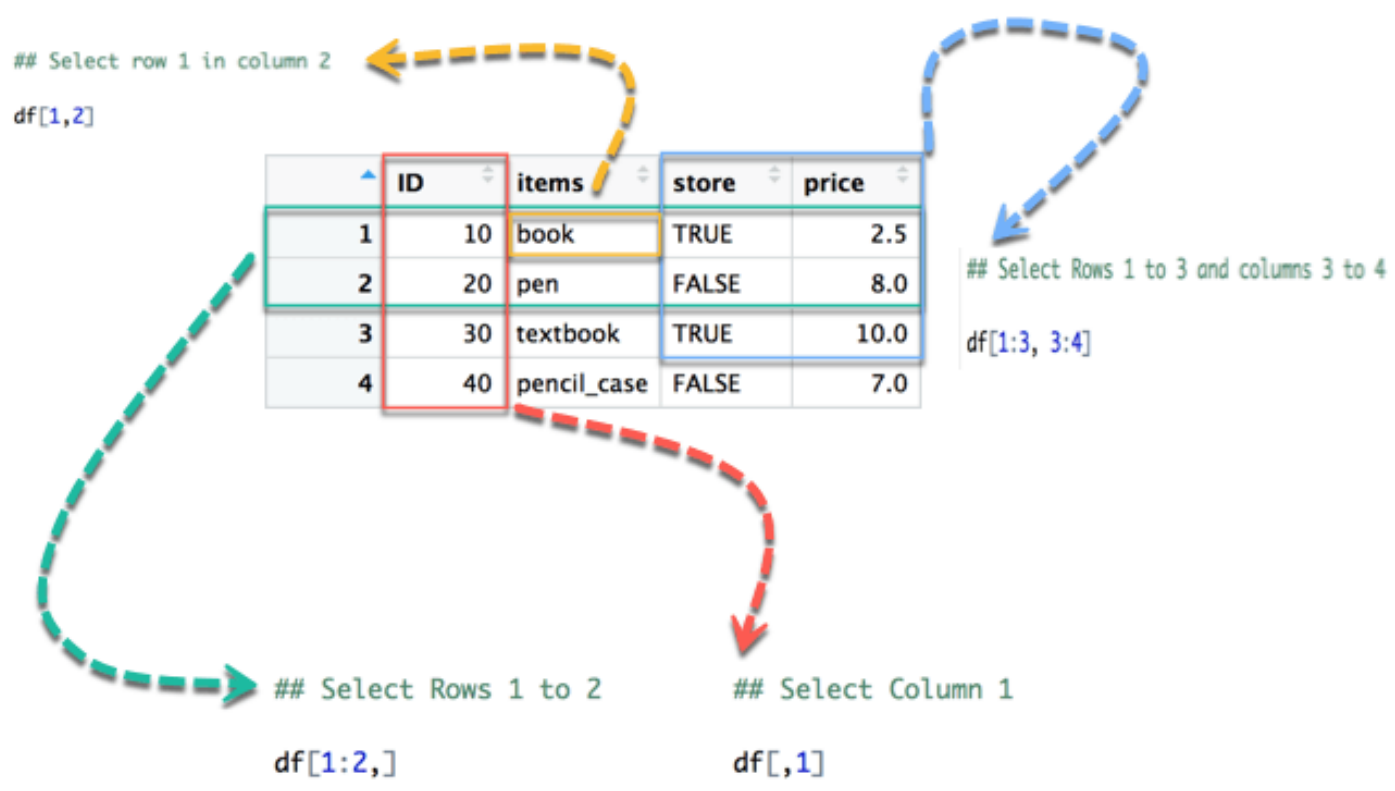
3.5 Manejo de dados bidimensionais

Indexação []

Matrizes e Data Frames

```
## Select row 1 in column 2
```

```
df[1,2]
```



	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

The diagram illustrates indexing on a data frame. A central table with columns ID, items, store, and price is shown. Four colored dashed arrows originate from code snippets and point to specific parts of the table: a yellow arrow from 'df[1,2]' points to the 'book' cell; a blue arrow from 'df[1:3, 3:4]' points to the 'store' and 'price' columns; a green arrow from 'df[1:2,]' points to the first two rows; and a red arrow from 'df[,1]' points to the 'ID' column.

```
## Select Rows 1 to 3 and columns 3 to 4
```

```
df[1:3, 3:4]
```

```
## Select Rows 1 to 2
```

```
df[1:2,]
```

```
## Select Column 1
```

```
df[,1]
```


3.5 Manejo de dados bidimensionais

Indexação []

Matrizes 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
```

3.5 Manejo de dados bidimensionais

Indexação []

Matrizes e Data Frames

```
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
```

3.5 Manejo de dados bidimensionais

Indexação []

Matrizes e Data Frames

```
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
```

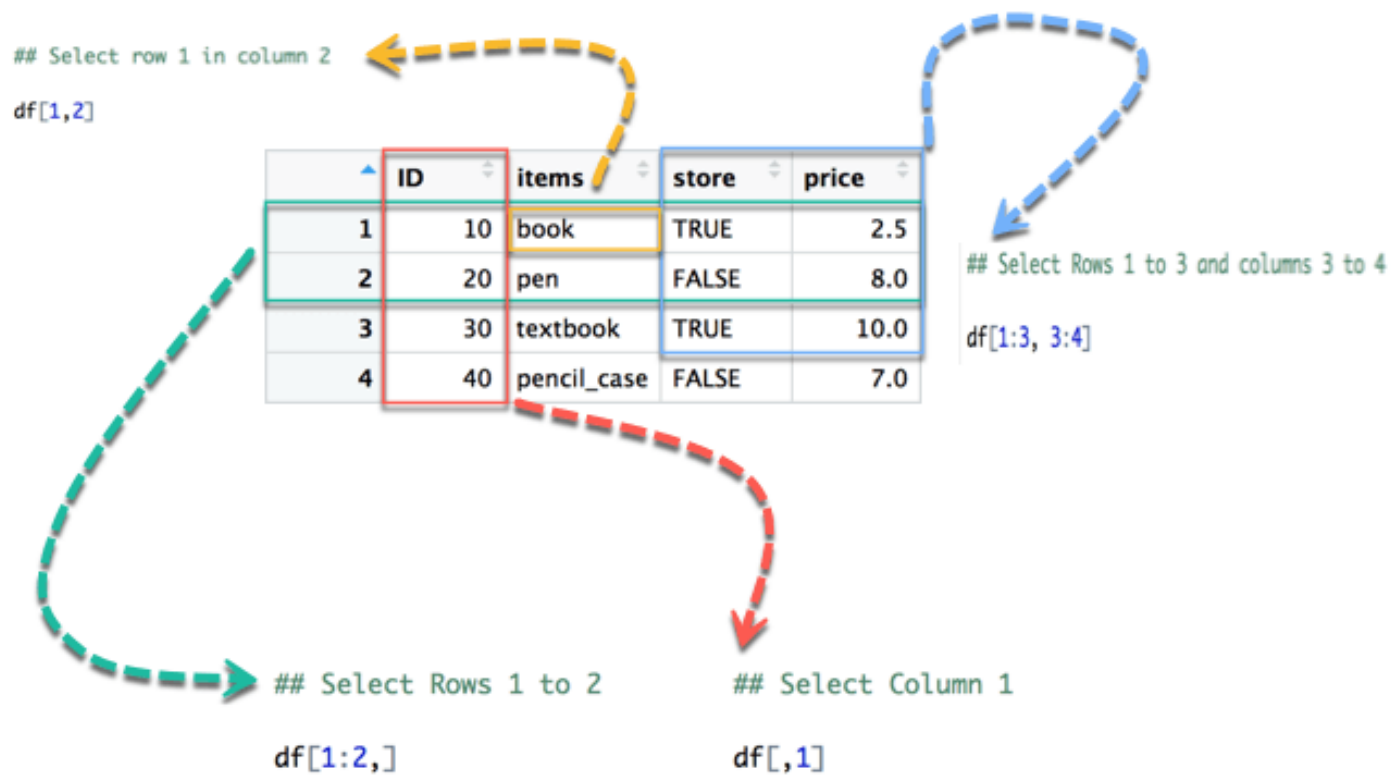
3.5 Manejo de dados bidimensionais

Indexação []

Matrizes e Data Frames: resumindo

```
## Select row 1 in column 2
```

```
df[1,2]
```



	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 Rows 1 to 3 and columns 3 to 4
```

```
df[1:3, 3:4]
```

```
## Select Rows 1 to 2
```

```
df[1:2,]
```

```
## Select Column 1
```

```
df[:,1]
```

3.5 Manejo de dados bidimensionais

Indexação \$

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

3.5 Manejo de dados bidimensionais

Indexação \$

Data Frames

```
# $ 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  
## [8] floresta floresta floresta  
## Levels: campo floresta
```

3.5 Manejo de dados bidimensionais

Indexação \$ e mudanças de colunas

Data Frames

```
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"
```

3.5 Manejo de dados bidimensionais

Indexação \$ e mudanças de colunas

Data Frames

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

```
df$abu
```

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

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

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


3.5 Manejo de dados bidimensionais

Indexação \$ e mudanças de colunas

Data Frames

```
# 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
```

Dúvidas?

3.6 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

3.6 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
```

3.6 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
```

3.6 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
```

3.6 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
```

3.6 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
```


3.6 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
```

3.6 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

Dúvidas?

3.7 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-r-see-2019/03_dados")
```

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

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

Vamos trabalhar com dados reais?



Dados reais

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

Eu mesmo et al. (2018)



3.8 Importar dados

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

1. csv
2. txt
3. xlsx

3.8 Importar dados

Ler uma planilha eletrônica (.csv)

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

3.8 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)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

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

3.8 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
```

3.8 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
```

3.9 Conferir e manejar dados importados

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

3.9 Conferir e manejar dados importados

head(): mostra as primeiras 6 linhas

```
head(da)
```

```
##           id reference_number species_number record sampled_habitat
## 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
## active_methods passive_methods complementary_methods      period
## 1             as              pt              <NA> mo,da,tw,ni
## 2             as              pt              <NA> mo,da,tw,ni
## 3             as              pt              <NA> mo,da,tw,ni
## 4             as              pt              <NA> mo,da,tw,ni
## 5             as             <NA>              <NA> mo,da,ni
## 6             <NA>             <NA>              <NA>      <NA>
## month_start year_start month_finish year_finish effort_months country
## 1             9       2000             1       2002             16  Brazil
```

3.9 Conferir e manejar dados importados

tail(): mostra as últimas 6 linhas

```
tail(da)
```

```
##           id reference_number species_number record sampled_habitat
## 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>
##      active_methods passive_methods complementary_methods period
## 1158          <NA>          <NA>          <NA>          <NA>
## 1159          <NA>          <NA>          <NA>          <NA>
## 1160          <NA>          <NA>          <NA>          <NA>
## 1161          <NA>          <NA>          <NA>          <NA>
## 1162          <NA>          <NA>          <NA>          <NA>
## 1163          <NA>          <NA>          <NA>          <NA>
##      month_start year_start month_finish year_finish effort_months
## 1158          NA          NA          NA          NA          NA
```


3.9 Conferir e manejar dados importados

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
```

3.9 Conferir e manejar dados importados

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

```
rownames(da)
```

```
##      [1] "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"
##     [10] "10"   "11"   "12"   "13"   "14"   "15"   "16"   "17"   "18"
##     [19] "19"   "20"   "21"   "22"   "23"   "24"   "25"   "26"   "27"
##     [28] "28"   "29"   "30"   "31"   "32"   "33"   "34"   "35"   "36"
##     [37] "37"   "38"   "39"   "40"   "41"   "42"   "43"   "44"   "45"
##     [46] "46"   "47"   "48"   "49"   "50"   "51"   "52"   "53"   "54"
##     [55] "55"   "56"   "57"   "58"   "59"   "60"   "61"   "62"   "63"
##     [64] "64"   "65"   "66"   "67"   "68"   "69"   "70"   "71"   "72"
##     [73] "73"   "74"   "75"   "76"   "77"   "78"   "79"   "80"   "81"
##     [82] "82"   "83"   "84"   "85"   "86"   "87"   "88"   "89"   "90"
##     [91] "91"   "92"   "93"   "94"   "95"   "96"   "97"   "98"   "99"
##    [100] "100"  "101"  "102"  "103"  "104"  "105"  "106"  "107"  "108"
##    [109] "109"  "110"  "111"  "112"  "113"  "114"  "115"  "116"  "117"
##    [118] "118"  "119"  "120"  "121"  "122"  "123"  "124"  "125"  "126"
##    [127] "127"  "128"  "129"  "130"  "131"  "132"  "133"  "134"  "135"
##    [136] "136"  "137"  "138"  "139"  "140"  "141"  "142"  "143"  "144"
```

3.9 Conferir e manejar dados importados

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

```
colnames(da)
```

```
## [1] "id" "reference_number"
## [3] "species_number" "record"
## [5] "sampled_habitat" "active_methods"
## [7] "passive_methods" "complementary_methods"
## [9] "period" "month_start"
## [11] "year_start" "month_finish"
## [13] "year_finish" "effort_months"
## [15] "country" "state"
## [17] "state_abbreviation" "municipality"
## [19] "site" "latitude"
## [21] "longitude" "coordinate_precision"
## [23] "altitude" "temperature"
## [25] "precipitation"
```

3.9 Conferir e manejar dados importados

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 10
## $ 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 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",...: 2 2 2 2
```

3.9 Conferir e manejar dados importados

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

```
summary(da)
```

```
##          id          reference_number species_number record sampled_habitat
## amp1001:    1    Min.      :1001      Min.      : 1.00  ab:346    fo      :114
## amp1002:    1    1st Qu.:1096      1st Qu.:  7.00  co:817    pp      :108
## amp1003:    1    Median :1204      Median :13.00                tp      : 74
## amp1004:    1    Mean   :1196      Mean   :15.17                la      : 51
## amp1005:    1    3rd Qu.:1295      3rd Qu.:21.00                sw      : 49
## amp1006:    1    Max.    :1389      Max.    :80.00                (Other):666
## (Other):1157                NA's      :101
##   active_methods passive_methods complementary_methods      period
## as          :573    pt          :270    ae          :187                tw,ni      :269
## as,sb       :175    dr          : 21    tp          : 11    mo,da,tw,ni:256
## as,tr       : 82    pt,ar       :  7    ae,tp       :  8    da,tw,ni  :207
## as,sb,tr    : 36    ft          :  6    ae,in       :  6    ni        :142
## sb          : 36    pt,ft       :  6    ae,rr       :  6    mo,ni     : 20
```

3.9 Conferir e manejar dados importados

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  
##     [14] 4780 4781 4782 4783 4784 4785 4786 4787 4788 4789 4790 4791 4792  
##     [27] 4793 4794 4795 4796 4797 4798 4799 4852 4853 4854 4855 4873 4874  
##     [40] 4965 5005 5020 5026 5027 5028 5034 5035 5036 5048 5055 5056 5057  
##     [53] 5063 5211 5223 5224 5225 5227 5249 5326 5327 5374 5469 5480 5518  
##     [66] 5519 5526 5529 5530 5531 5532 5533 5534 5566 5567 5568 5638 5639  
##     [79] 5640 5641 5642 5665 5690 5717 5728 5729 5736 5738 5744 5798 5801  
##     [92] 5806 5807 5808 5809 5810 5811 5812 5813 5814 5815 5821 5824 5825  
##    [105] 5826 5827 5828 5829 5830 5831 5832 5833 5834 5835 5836 5837 5838  
##    [118] 5839 5840 5841 5842 5843 5844 5845 5846 5847 5848 5849 5850 5851
```

3.9 Conferir e manejar dados importados

Retirar os NAs

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

```
nrow(da)
```

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

```
nrow(da_na)
```

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

Dúvidas?

3.10 Exportar dados

Exportar uma tabela de dados na pasta do diretório

Planilha eletrônica (.csv)

```
write.csv(da_na, "ATLANTIC_AMPHIBIAN_sites_sem_na.csv",  
          row.names = FALSE, quote = FALSE)
```

Planilha de texto (.txt)

```
write.table(da_na, "ATLANTIC_AMPHIBIAN_sites_sem_na.txt",  
            row.names = FALSE, quote = FALSE)
```

Planilha eletrônica (.xlsx)

```
openxlsx::write.xlsx(da_na, "ATLANTIC_AMPHIBIAN_sites_sem_na.xlsx",  
                     row.names = FALSE, quote = FALSE)
```

Dúvidas?

Maurício Vancine

Contatos:

 mauricio.vancine@gmail.com

 mauriciovancine.netlify.com

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

 [@mauriciovancine](https://github.com/mauriciovancine)

 [@mauriciovancine](https://discord.com/invite/mauriciovancine)

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