



1



2

## Data Frames

- São como tabelas do SQL ou Planilhas do Excel
- Geralmente é como seus dados serão representados
- Listas especiais onde todos os elementos possuem o mesmo comprimento
- Cada elemento é uma coluna, e seu comprimento é a quantidade de linhas
  - Então é uma lista de colunas
- Cada coluna pode ter tipos diferentes

Prof. Dr. Razer A N R Montão

SEPT/UFPR

3

3

## Data Frames: Criação

- Transformação de Lista em Data Frame
 

```
> lista <- list(x=c(1, 2, 3), y=c("a", "b", "c"))
> lista
$x
[1] 1 2 3

$y
[1] "a" "b" "c"

> df <- as.data.frame(lista)
> df
  x y
1 1 a
2 2 b
3 3 c
```

Prof. Dr. Razer A N R Montão

SEPT/UFPR

4

4

## Data Frames: Criação

- Criação de Data Frame Vazio: `data.frame()`  

```
> df <- data.frame()
> df
data frame with 0 columns and 0 rows
```
- Adição de colunas: operador `$`  

```
> df$x <- integer()
> df$nome <- character()
> df
[1] x      nome
<0 linhas> (ou row.names de comprimento 0)
```
- Adição de linhas  

```
> df[nrow(df)+1,] <- c(10, "razer")
> df
      x  nome
1 10 razer
```

**Nota**  
*integer(x)* cria um vetor de inteiros de comprimento x (default 0). Todos recebem o valor 0  
*character(x)* cria um vetor de caracteres de comprimento x (default 0). Todos recebem o valor ""

Prof. Dr. Razer A N R Montão

SEPT/UFPR

5

5

## Data Frames: Criação

- Criação de Data Frame Vazio com Colunas: `data.frame()`  

```
> df <- data.frame(x=integer(), nome=character())
> df
[1] x      nome
<0 linhas> (ou row.names de comprimento 0)
```
- Função `str()` mostra a estrutura de um objeto qualquer  

```
> str(df)
'data.frame': 0 obs. of 2 variables:
 $ x      : int
 $ nome   : Factor w/ 0 levels:
```

Prof. Dr. Razer A N R Montão

SEPT/UFPR

6

6

## Data Frames: Criação

- Criação de Data Frame com Dados: `data.frame()`

```
> df <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
> df
```

	id	x	y
1	a	1	11
2	b	2	12
3	c	3	13
4	d	4	14
5	e	5	15
6	f	6	16
7	g	7	17
8	h	8	18
9	i	9	19
10	j	10	20

Prof. Dr. Razer A N R Montaña

SEPT/UFR

7

7

## Data Frames: Criação

- Criação de Data Frame com Dados: `data.frame()`

```
> df <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas",
"Montaña"), idade=c(20, 18, 26, 32, 15))
> df
```

	nome	idade
1	Razer	20
2	Anthom	18
3	Nizer	26
4	Rojas	32
5	Montaña	15

Prof. Dr. Razer A N R Montaña

SEPT/UFR

8

8

## Data Frames: Criação

- Criação de Data Frame com Dados, Sem Factors: `data.frame()`

- Parâmetro booleano: `stringsAsFactors`

```
> df <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas",
"Montaño"), idade=c(20, 18, 26, 32, 15))
> str(df)
'data.frame': 5 obs. of 2 variables:
 $ nome : Factor w/ 5 levels "Anthom","Montaño",...: 4 1 3 5 2
 $ idade: num 20 18 26 32 15
> df <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas",
"Montaño"), idade=c(20, 18, 26, 32, 15), stringsAsFactors=FALSE)
> str(df)
'data.frame': 5 obs. of 2 variables:
 $ nome : chr "Razer" "Anthom" "Nizer" "Rojas" ...
 $ idade: num 20 18 26 32 15
```

Prof. Dr. Razer A N R Montaño

SEPT/UFRP

9

9

## Data Frames: Criação.

- Leitura de um arquivo CSV: `read.csv()`

- Seja o seguinte arquivo: `exemplo.csv`

```
"nome","idade"
"Razer",18
"Anthom",20
"Nizer",15
"Rojas",30
"Montaño",26
```

- Carrega-se com: `read.csv`

```
> df <- read.csv("exemplo.csv", fileEncoding="UTF-8")
> str(df)
'data.frame': 5 obs. of 2 variables:
 $ nome : Factor w/ 5 levels "Anthom","Montaño",...: 4 1 3 5 2
 $ idade: int 18 20 15 30 26
> df
      nome idade
1   Razer    18
2 Anthom    20
3   Nizer    15
4   Rojas    30
5 Montaño    26
```

Prof. Dr. Razer A N R Montaño

SEPT/UFRP

10

10

## Data Frames: Funções Úteis

```
> df <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
```

- Funções úteis

- **dim()** : número de linhas e colunas

```
> dim(df)
[1] 10  3
```

- **nrow()** : número de linhas

```
> nrow(df)
[1] 10
```

- **ncol()** : número de colunas

```
> ncol(df)
[1] 3
```

## Data Frames: Funções Úteis

```
> df <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
```

- Funções úteis

- **head()** : mostra as 6 primeiras linhas

```
> head(df)
  id x  y
1  a 1 11
2  b 2 12
3  c 3 13
4  d 4 14
5  e 5 15
6  f 6 16
```

- **tail()** : mostra as 6 últimas linhas

```
> tail(df)
  id x  y
5  e 5 15
6  f 6 16
7  g 7 17
8  h 8 18
9  i 9 19
10 j 10 20
```

## Data Frames: Funções Úteis

```
> df <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
```

- Funções úteis

- `names()` : mostra o nome das colunas

```
> names(df)
[1] "id" "x"  "y"
```

- `rownames()` : mostra o nome das linhas

```
> rownames(df)
[1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"
```

- `str()` : mostra a estrutura do data frame, amostra de dados, e a classe de cada coluna

```
> str(df)
'data.frame': 10 obs. of  3 variables:
 $ id: Factor w/ 10 levels "a","b","c","d",...: 1 2 3 4 5 6 7 8 9 10
 $ x : int  1 2 3 4 5 6 7 8 9 10
 $ y : int 11 12 13 14 15 16 17 18 19 20
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

13

13

## Data Frames: Funções Úteis

```
> df <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
```

```
> df2 <- data.frame(z = 111:120)
```

- Funções úteis

- `cbind()` : acopla dois data frames lado a lado (adiciona colunas)

```
> cbind(df, df2)
```

```
   id  x  y  z
1  a  1 11 111
2  b  2 12 112
3  c  3 13 113
4  d  4 14 114
5  e  5 15 115
6  f  6 16 116
7  g  7 17 117
8  h  8 18 118
9  i  9 19 119
10 j 10 20 120
```

Coluna Adicionada

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

14

14



## Data Frames: Funções Úteis

```
> df <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
```

```
> df3 <- data.frame(id="r", x=200, y=300)
```

- Funções úteis

- `rbind()` : empilha duas tabelas (adiciona linhas)

```
> rbind(df, df3)
```

	id	x	y
1	a	1	11
2	b	2	12
3	c	3	13
4	d	4	14
5	e	5	15
6	f	6	16
7	g	7	17
8	h	8	18
9	i	9	19
10	j	10	20
11	r	200	300

Linha Adicionada

Prof. Dr. Razer A N R Montão

SEPT/UFPR

15

15

## Data Frames: Funções Úteis.

```
> lista <- list(nome=c("TADS", "TGQ", "TCI"), alunos=c(45, 50, 40))
```

```
> lista
```

```
$nome
```

```
[1] "TADS" "TGQ" "TCI"
```

```
$alunos
```

```
[1] 45 50 40
```

- Funções úteis

- `as.data.frame()` : transforma uma lista em data frame

```
> as.data.frame(lista)
```

	nome	alunos
1	TADS	45
2	TGQ	50
3	TCI	40

Prof. Dr. Razer A N R Montão

SEPT/UFPR

16

16



## Data Frames: Acesso a Elementos

- Acesso a elementos como em uma matriz
 

```
> lista <- list(x=c(1, 2, 3), y=c("a", "b", "c"))
> df <- as.data.frame(lista)
> str(df)
'data.frame':      3 obs. of  2 variables:
 $ x: num  1 2 3
 $ y: Factor w/ 3 levels "a","b","c": 1 2 3
> df
  x y
1 1 a
2 2 b
3 3 c
> df[3, 2]
[1] c
Levels: a b c
> df[3, 1]
[1] 3
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

17

17

## Data Frames: Acesso a Elementos

- Retornar uma coluna individualmente
- Retorna um Data Frame com a coluna escolhida
 

```
> df[1]
  x
1 1
2 2
3 3
> df[2]
  y
1 a
2 b
3 c
```
- Retorna um Vetor com a coluna escolhida (dados do mesmo tipo, [ , ] retorna vetor quando é uma coluna)
 

```
> df[,1]
[1] 1 2 3
> df[,2]
[1] a b c
Levels: a b c
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

18

18

## Data Frames: Acesso a Elementos

- Retornar uma linha individualmente
- Retorna um Data Frame com a linha escolhida
 

```
> df[1,]
  x y
1 1 a
> df[2,]
  x y
2 2 b
```
- Retorna uma Lista com a linha escolhida (dados diferentes, precisa ser lista, usar **drop**)
 

```
> df[1, ,drop=TRUE]
$x
[1] 1

$y
[1] a
Levels: a b c

> df[2, ,drop=TRUE]
$x
[1] 2

$y
[1] b
Levels: a b c
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

19

19

## Data Frames: Acesso a Elementos

- Acesso a elementos pelo nome da coluna
 

```
> lista <- list(x=c(1, 2, 3), y=c("a", "b", "c"))
> df <- as.data.frame(lista)
> df
  x y
1 1 a
2 2 b
3 3 c
> df$x
[1] 1 2 3
> df$y
[1] "a" "b" "c"
> df$y[3]
[1] c
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

20

20

## Data Frames: Acesso a Elementos

- Diferença entre `[]` e `[[ ]]`
  - `[]` fatia o data frame, retornando outro data frame, `[[ ]]` retorna o conteúdo
  - `[]` mantém o tipo do dado, `[[ ]]` extrai o elemento de dentro

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
                   valores=seq(1.5, by=0.5, length.out=10))
```

```
> df[1]
  letras
1      a
2      b
3      c
4      d
5      e
6      f
7      g
8      h
9      i
10     j

> df[[1]]
[1] a b c d e f g h i j
```

Obtém um data frame com a coluna 1

Obtém um vetor com os dados da coluna 1

Prof. Dr. Razer A N R Montaña

SEPT/UFRP

21

21

## Data Frames: Acesso a Elementos

- Diferença entre `[]` e `[[ ]]`
  - Vetores:** Retornam o mesmo tipo de dado
  - Matrizes:** Como é um vetor com dimensões, retornam o mesmo tipo de dado

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

```
> mat <- matrix(1:10, nrow=5)
> mat
      [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> mat[5]
[1] 5
> mat[[5]]
[1] 5
```

Prof. Dr. Razer A N R Montaña

SEPT/UFRP

22

22

## Data Frames: Acesso a Elementos

- Diferença entre `[]` e `[[ ]]`

- Listas:** Retornam tipos diferentes. `[]` retorna uma sublista e `[[ ]]` um vetor

```
> lista <- list(letras=letters[1:5], numeros=10:1, valores=rnorm(3))
> lista
$letras
[1] "a" "b" "c" "d" "e"

$numeros
[1] 10  9  8  7  6  5  4  3  2  1

$valores
[1] -0.4997171 -0.7824937 -0.5532524
> lista[2]
$numeros
[1] 10  9  8  7  6  5  4  3  2  1

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

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

23

23

## Data Frames: Acesso a Elementos.

- Diferença entre `[]` e `[[ ]]`

- Data Frames:** Como são listas, retornam tipos diferentes. `[]` retorna um sub-data-frame e `[[ ]]` um vetor

```
> df <- data.frame(numero=1:5, letras=letters[1:5])
> df
  numero letras
1      1     a
2      2     b
3      3     c
4      4     d
5      5     e
> df[2]
  letras
1     a
2     b
3     c
4     d
5     e
> df[[2]]
[1] a b c d e
Levels: a b c d e
```

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

24

24

## Data Frames: Cortando Data Frames

- Cortando Linhas:

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
```

```
> teste <- df[3,]
```

```
> teste
  letras numeros valores
c      23      2.5
```

Obtém um data frame contendo a linha 3

```
> teste <- df[c(4:6),]
```

```
> teste
  letras numeros valores
4      d      24      3.0
5      e      25      3.5
6      f      26      4.0
```

Obtém um data frame contendo as linhas de 4 a 6

```
> teste <- df[c(2,5,8),]
```

```
> teste
  letras numeros valores
2      b      22      2.0
5      e      25      3.5
8      h      28      5.0
```

Obtém um data frame contendo as linhas 2, 5 e 8

Prof. Dr. Razer A N R Montaña

SEPT / UFPR

25

25

## Data Frames: Cortando Data Frames

- Cortando Colunas

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
```

```
> teste <- df[,2]
```

```
> teste
[1] 21 22 23 24 25 26 27 28 29 30
```

Obtém um vetor com a coluna 2 (como é só uma coluna transforma em vetor)

```
> teste <- df[,2, drop=FALSE]
```

```
> teste
  numeros
1      21
2      22
3      23
4      24
5      25
6      26
7      27
8      28
9      29
10     30
```

Obtém um data frame contendo a coluna 2 (drop=FALSE)

Prof. Dr. Razer A N R Montaña

SEPT / UFPR

26

26

## Data Frames: Cortando Data Frames

- Cortando Colunas

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
> teste <- df[,2:3]
> teste
```

	numeros	valores
1	21	1.5
2	22	2.0
3	23	2.5
4	24	3.0
5	25	3.5
6	26	4.0
7	27	4.5
8	28	5.0
9	29	5.5
10	30	6.0

Obtém um data frame com as colunas de 2 a 3 (como retorna mais de uma coluna, mantém como data frame)

Prof. Dr. Razer A N R Montaña

SEPT / UFPR

27

27

## Data Frames: Cortando Data Frames

- Cortando Colunas

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
> teste <- df[,c(1,3)]
> teste
```

	letras	valores
1	a	1.5
2	b	2.0
3	c	2.5
4	d	3.0
5	e	3.5
6	f	4.0
7	g	4.5
8	h	5.0
9	i	5.5
10	j	6.0

Obtém um data frame com as colunas de 1 e 3 (como retorna mais de uma coluna, mantém como data frame)

Prof. Dr. Razer A N R Montaña

SEPT / UFPR

28

28

## Data Frames: Cortando Data Frames

- Cortando Linhas e Colunas

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
> teste <- df[c(1,6,10), c("letras", "valores")]
> teste
```

	letras	valores
1	a	1.5
6	f	4.0
10	j	6.0

Obtém um data frame contendo as linhas de 1, 6 e 10, e as colunas "letras" e "valores"

## Data Frames: Cortando Data Frames

- Exclusão de Linhas

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
> teste <- df[-c(2:9), ]
> teste
```

	letras	numeros	valores
1	a	21	1.5
10	j	30	6.0

Obtém um data frame contendo tudo menos as linhas de 2 a 9

```
> teste <- df[-c(1,3,5,7,9), ]
> teste
```

	letras	numeros	valores
2	b	22	2
4	d	24	3
6	f	26	4
8	h	28	5
10	j	30	6

Obtém um data frame contendo tudo menos as linhas de 1, 3, 5, 7 e 9



## Data Frames: Cortando Data Frames

- Exclusão de Colunas

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
> teste <- df[, -1]
> teste
```

	numeros	valores
1	21	1.5
2	22	2.0
3	23	2.5
4	24	3.0
5	25	3.5
6	26	4.0
7	27	4.5
8	28	5.0
9	29	5.5
10	30	6.0

Retorna um dataframe contendo tudo de "df" menos a coluna 1

Prof. Dr. Razer A N R Montañó

SEPT / UFPR

31

31

## Data Frames: Cortando Data Frames

- Exclusão de Colunas

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
> teste <- df[, -c(1:2)]
> teste
```

[1]	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

```
> teste <- df[, -c(1,3)]
> teste
```

[1]	21	22	23	24	25	26	27	28	29	30
-----	----	----	----	----	----	----	----	----	----	----

Obtém um vetor contendo tudo menos as colunas de 1 a 2 (ficam os dados da coluna 3). Como sobra só uma coluna retorna um vetor.

Obtém um vetor contendo tudo menos as colunas de 1 e 3 (sobram os dados da coluna 2). Como sobra só uma coluna, retorna um vetor.

Prof. Dr. Razer A N R Montañó

SEPT / UFPR

32

32

## Data Frames: Cortando Data Frames.

- Exclusão de Colunas

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
> teste <- df[, -c(1:2), drop=FALSE]
```

```
> teste
  valores
1      1.5
2      2.0
3      2.5
4      3.0
5      3.5
6      4.0
7      4.5
8      5.0
9      5.5
10     6.0
```

Nesse caso o “drop=FALSE” indica que o retorno não será transformado em vetor, mantendo como dataframe

Prof. Dr. Razer A N R Montaña

SEPT/UFRP

33

33

## Data Frames: Seleção de Elementos

- Seleção condicional de dados com `which()`
  - `which()` : recebe um condicional e retorna a posição dos elementos que atendem esta condição (Verdadeiro - TRUE)

```
> lista <- c(TRUE, FALSE, FALSE, TRUE)
> which(lista)
[1] 1 4
> lista <- c(10:1)
> lista
[1] 10 9 8 7 6 5 4 3 2 1
> lista==3
[1] FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE
> which( lista==3 )
[1] 8
> which( lista%%2==0 )
[1] 1 3 5 7 9
> which(lista%%2==0 & lista>5)
[1] 1 3 5
```

Prof. Dr. Razer A N R Montaña

SEPT/UFRP

34

34

## Data Frames: Seleção de Elementos

- Seleção condicional de dados com `which()`

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
```

```
> which(df$numeros>=28)
```

```
[1] 8 9 10
```

```
> df[ which(df$numeros>=28), ]
```

```
  letras numeros valores
8      h      28     5.0
9      i      29     5.5
10     j      30     6.0
```

```
> teste <- df[ which(df$numeros>=28), ]
```

```
> df[ which(df$numeros>=28 & df$numero%%2==1), ]
```

```
  letras numeros valores
9      i      29     5.5
```

Obtém a posição dos elementos cujos números são maiores ou iguais a 28

Retorna um data frame com as linhas cujos números são maiores ou iguais a 28

Prof. Dr. Razer A N R Montañó

SEPT/UFPR

35

35

## Data Frames: Seleção de Elementos

- Seleção condicional de dados SEM `which()`

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
```

```
> df[ df$numeros>=28, ]
```

```
  letras numeros valores
8      h      28     5.0
9      i      29     5.5
10     j      30     6.0
```

```
> teste <- df[ df$numeros>=28, ]
```

```
> df[ df$numeros>=28 & df$numero%%2==1, ]
```

```
  letras numeros valores
9      i      29     5.5
```

Prof. Dr. Razer A N R Montañó

SEPT/UFPR

36

36

## Data Frames: Seleção de Elementos

- Seleção condicional de dados com `%in%`
  - `%in%` verifica se um elemento está dentro do vetor
- Quando aplicado a um vetor, verifica cada um dos elementos
  - Retorna um vetor com o resultado de cada teste, para cada elemento

```
> 10 %in% c(5:50)
```

```
[1] TRUE
```

```
> 1 %in% c(5:50)
```

```
[1] FALSE
```

```
> c(1:10) %in% c(5:50)
```

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

Prof. Dr. Razer A N R Montaña

SEPT/UFR

37

37

## Data Frames: Seleção de Elementos

- Quando aplicado a um data frame, também verifica cada um dos elementos

```
> df <- data.frame(letras=letters[1:10], numeros=21:30,
  valores=seq(1.5, by=0.5, length.out=10))
```

```
> df
```

	letras	numeros	valores
1	a	21	1.5
2	b	22	2.0
3	c	23	2.5
4	d	24	3.0
5	e	25	3.5
6	f	26	4.0
7	g	27	4.5
8	h	28	5.0
9	i	29	5.5
10	j	30	6.0

```
> df$letras %in% c("a", "d", "i")
```

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

Prof. Dr. Razer A N R Montaña

SEPT/UFR

38

38

## Data Frames: Seleção de Elementos.

- Seleção condicional de dados com

```
> df[df$letras %in% c("a", "h", "j"), ]
```

	letras	numeros	valores
1	a	21	1.5
8	h	28	5.0
10	j	30	6.0

Obtém um data frame com todas as linhas cuja coluna "letras" tenha os elementos do vetor



## Exercícios.

- Crie o seguinte Data Frame

```
> df <- data.frame(letras=letters[1:10], numeros=21:30, valores=rnorm(10))
```

- Faça as seguintes pesquisas

- Retorne a linha 5
- Retorna a coluna 2 (como vetor e como data frame – drop=FALSE)
- Retorne as colunas 2 e 3
- Retorne os elementos da linha 6, mas somente as colunas 1 e 3
- Retorne os elementos que possuem na coluna "valores" um valor maior que zero
- Retorne os elementos que possuem na coluna "numeros" um valor ímpar
- Retorne os elementos que possuem na coluna "valores" um valor maior que zero e na coluna "numeros" um valor par
- Retorne os elementos que possuem na coluna "letras" somente os seguintes valores "b", "g", "h"

## Data Frames: Merge

- Usado para unir data frames de forma horizontal
  - CROSS JOIN : produto cartesiano
  - INNER JOIN
    - Une fazendo o casamento de atributos chave
    - Só une se todos os atributos casarem
  - OUTER JOIN
    - Une fazendo o casamento de atributos chave
    - Une os dados mesmo se o atributo chave não casar nos dois data frames
  - LEFT OUTER JOIN
    - Une fazendo o casamento de atributos chave
    - Une todos os dados do data frame esquerdo (primeiro), mesmo se não houver correspondência no segundo
  - RIGHT OUTER JOIN
    - Une fazendo o casamento de atributos chave
    - Une todos os dados do data frame direito (segundo), mesmo se não houver correspondência no primeiro

Prof. Dr. Razer A N R Montaña

SEPT / UFPR

41

41

## Data Frames: Merge

- Sejam 2 data frames: df1 e cidades

df1	
<nome>	<cidadeId>
"Razer"	3
"Anthom"	10
"Nizer"	2
"Rojas"	3
"Montaña"	1

cidades	
<cidadeId>	<cidade>
1	"Curitiba"
2	"SJP"
3	"Pinhais"
4	"Colombo"

```
> df1 <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas", "Montaña"),
  cidadeId=c(3, 10, 2, 3, 1))
```

```
> cidades <- data.frame(cidadeId=c(1, 2, 3, 4), cidade=c("Curitiba", "SJP",
  "Pinhais", "Colombo"))
```

Prof. Dr. Razer A N R Montaña

SEPT / UFPR

42

42

## Data Frames: Merge

- Uso:
  - CROSS JOIN :
 

```
merge(df1, cidades, by=NULL)
```
  - INNER JOIN
 

```
merge(df1, cidades, by="cidadeId")
```
  - OUTER JOIN
 

```
merge(df1, cidades, by="cidadeId", all=TRUE)
```
  - LEFT OUTER JOIN
 

```
merge(df1, cidades, by="cidadeId", all.x=TRUE)
```
  - RIGHT OUTER JOIN
 

```
merge(df1, cidades, by="cidadeId", all.y=TRUE)
```

Prof. Dr. Razer A N R Montaña

SEPT / UFPR

43

43

## Data Frames: Merge

- CROSS JOIN : produto cartesiano

```
> df1 <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas", "Montaña"), cidadeId=c(3, 10, 2, 3, 1))
> df1
  nome cidadeId
1 Razer        3
2 Anthom       10
3 Nizer        2
4 Rojas        3
5 Montaña      1
> cidades <- data.frame(cidadeId=c(1, 2, 3, 4), cidade=c("Curitiba", "SJP", "Pinhais", "Colombo"))
> cidades
  cidadeId cidade
1         1 Curitiba
2         2      SJP
3         3 Pinhais
4         4  Colombo
> merge(df1, cidades, by=NULL)
  nome cidadeId.x cidadeId.y cidade
1 Razer          3          1 Curitiba
2 Anthom         10          1 Curitiba
3 Nizer           2          1 Curitiba
4 Rojas           3          1 Curitiba
5 Montaña         1          1 Curitiba
6 Razer           3          2      SJP
7 Anthom          10          2      SJP
8 Nizer            2          2      SJP
9 Rojas            3          2      SJP
10 Montaña         1          2      SJP
11 Razer           3          3 Pinhais
12 Anthom          10          3 Pinhais
13 Nizer            2          3 Pinhais
14 Rojas            3          3 Pinhais
15 Montaña         1          3 Pinhais
16 Razer           3          4  Colombo
17 Anthom          10          4  Colombo
18 Nizer            2          4  Colombo
19 Rojas            3          4  Colombo
20 Montaña         1          4  Colombo
```

Prof. Dr. Razer A N R Montaña

SEPT / UFPR

44

44



## Data Frames: Merge

- INNER JOIN
  - Une fazendo o casamento de atributos chave
  - Só une se todos os atributos casarem

```
> df1 <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas", "Montaño"), cidadeId=c(3, 10, 2, 3, 1))
> df1
  nome cidadeId
1  Razer      3
2 Anthom     10
3  Nizer      2
4  Rojas      3
5 Montaño     1
> cidades <- data.frame(cidadeId=c(1, 2, 3, 4), cidade=c("Curitiba", "SJP", "Pinhais", "Colombo"))
> cidades
  cidadeId cidade
1         1 Curitiba
2         2      SJP
3         3 Pinhais
4         4  Colombo
> merge(df1, cidades, by="cidadeId")
  cidadeId nome cidade
1         1 Montaño Curitiba
2         2  Nizer      SJP
3         3  Razer Pinhais
4         3  Rojas Pinhais
```

Prof. Dr. Razer A N R Montaño

SEPT / UFPR

45

45

## Data Frames: Merge

- OUTER JOIN
  - Une fazendo o casamento de atributos chave
  - Une os dados mesmo se o atributo chave não casar nos dois data frames

```
> df1 <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas", "Montaño"), cidadeId=c(3, 10, 2, 3, 1))
> df1
  nome cidadeId
1  Razer      3
2 Anthom     10
3  Nizer      2
4  Rojas      3
5 Montaño     1
> cidades <- data.frame(cidadeId=c(1, 2, 3, 4), cidade=c("Curitiba", "SJP", "Pinhais", "Colombo"))
> cidades
  cidadeId cidade
1         1 Curitiba
2         2      SJP
3         3 Pinhais
4         4  Colombo
> merge(df1, cidades, by="cidadeId", all=TRUE)
  cidadeId nome cidade
1         1 Montaño Curitiba
2         2  Nizer      SJP
3         3  Razer Pinhais
4         3  Rojas Pinhais
5         4 <NA>  Colombo
6        10 Anthom  <NA>
```

Prof. Dr. Razer A N R Montaño

SEPT / UFPR

46

46

## Data Frames: Merge

### • LEFT OUTER JOIN

- Une fazendo o casamento de atributos chave
- Une todos os dados do data frame esquerdo (primeiro), mesmo se não houver correspondência no segundo

```
> df1 <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas", "Montaño"), cidadeId=c(3, 10, 2, 3, 1))
> df1
  nome cidadeId
1  Razer      3
2 Anthom     10
3  Nizer      2
4  Rojas      3
5 Montaño     1
> cidades <- data.frame(cidadeId=c(1, 2, 3, 4), cidade=c("Curitiba", "SJP", "Pinhais", "Colombo"))
> cidades
  cidadeId cidade
1         1 Curitiba
2         2      SJP
3         3  Pinhais
4         4  Colombo
> merge(df1, cidades, by="cidadeId", all.x=TRUE)
  cidadeId nome cidade
1         1 Montaño Curitiba
2         2   Nizer      SJP
3         3   Razer  Pinhais
4         3   Rojas  Pinhais
5        10 Anthom    <NA>
```

Prof. Dr. Razer A N R Montaño

SEPT/UFRP

47

47

## Data Frames: Merge.

### • RIGHT OUTER JOIN

- Une fazendo o casamento de atributos chave
- Une todos os dados do data frame direito (segundo), mesmo se não houver correspondência no primeiro

```
> df1 <- data.frame(nome=c("Razer", "Anthom", "Nizer", "Rojas", "Montaño"), cidadeId=c(3, 10, 2, 3, 1))
> df1
  nome cidadeId
1  Razer      3
2 Anthom     10
3  Nizer      2
4  Rojas      3
5 Montaño     1
> cidades <- data.frame(cidadeId=c(1, 2, 3, 4), cidade=c("Curitiba", "SJP", "Pinhais", "Colombo"))
> cidades
  cidadeId cidade
1         1 Curitiba
2         2      SJP
3         3  Pinhais
4         4  Colombo
> merge(df1, cidades, by="cidadeId", all.y=TRUE)
  cidadeId nome cidade
1         1 Montaño Curitiba
2         2   Nizer      SJP
3         3   Razer  Pinhais
4         3   Rojas  Pinhais
5         4    <NA>  Colombo
```

Prof. Dr. Razer A N R Montaño

SEPT/UFRP

48

48



## Exercícios.

- Criar os data frames df1 e cidades

```
> df1 <- data.frame(nome=c("Razer", "Anthon", "Nizer", "Rojas", "Montaño"),
  cidadeId=c(3, 10, 2, 3, 1))
```

```
> cidades <- data.frame(cidadeId=c(1, 2, 3, 4), cidade=c("Curitiba", "SJP",
  "Pinhais", "Colombo"))
```

- Executar os merges apresentados

- a) Cross Join
- b) Inner Join
- c) Outer Join
- d) Left Outer Join
- e) Right Outer Join

Prof. Dr. Razer A N R Montaño

SEPT / UFPR

49

49

## Criação de dados

- Função `sample()` : obtém uma amostra de elementos, de um determinado tamanho

**`sample(x, tamanho, replace=FALSE)`**

- `x` : elementos para escolher
- `tamanho` : quantidade de itens
- `replace` : amostragem com replacement, pode repetir os dados

- Exemplo

```
> idade <- sample( 14:102, 10)
```

```
> idade
```

```
[1] 33 91 19 70 84 64 56 85 16 24
```

Prof. Dr. Razer A N R Montaño

SEPT / UFPR

50

50

## Criação de dados

- Exemplo de criação de dados de pessoas usando `sample()` e `data.frame()`

```
> n <- 10
> sexo <- sample( c("masculino", "feminino"), n, replace=TRUE)
> idade <- sample( 14:102, n, replace=TRUE)
> peso <- sample( 50:90, n, replace=TRUE)
> menor <- idade<18
> pessoas <- data.frame(sexo=sexo, idade=idade, peso=peso, menor=menor)
> pessoas
```

	sexo	idade	peso	menor
1	feminino	44	71	FALSE
2	feminino	81	70	FALSE
3	masculino	14	62	TRUE
4	feminino	22	79	FALSE
5	masculino	49	61	FALSE
6	masculino	16	82	TRUE
7	feminino	72	83	FALSE
8	masculino	80	79	FALSE
9	masculino	31	81	FALSE
10	feminino	57	66	FALSE

Prof. Dr. Razer A N R Montão

SEPT/UFRP

51

51

## Criação de dados

- Ao criar o data frame, automaticamente o sexo é transformado em Factor
  - Factor são valores categóricos

```
> str(pessoas)
'data.frame': 10 obs. of 4 variables:
 $ sexo : Factor w/ 2 levels "feminino","masculino": 1 1 2 1 2 2 1 2 2 1
 $ idade: int 44 81 14 22 49 16 72 80 31 57
 $ peso : int 71 70 62 79 61 82 83 79 81 66
 $ menor: logi FALSE FALSE TRUE FALSE FALSE TRUE ...
```

Prof. Dr. Razer A N R Montão

SEPT/UFRP

52

52

## Criação de dados.

- Para evitar este comportamento use o parâmetro `stringsAsFactors`

```
> pessoas <- data.frame(sexo=sexo, idade=idade, peso=peso,
menor=menor, stringsAsFactors=FALSE)
> str(pessoas)
'data.frame': 10 obs. of 4 variables:
 $ sexo : chr  "feminino" "feminino" "masculino" "feminino" ...
 $ idade: int   44  81  14  22  49  16  72  80  31  57
 $ peso : int   71  70  62  79  61  82  83  79  81  66
 $ menor: logi  FALSE FALSE TRUE  FALSE FALSE TRUE  ...
```

## Ordenação do Data Frame

- Ordenação do data frame: `sort()` e `order()`
  - `sort()` retorna o vetor com os valores ordenados. Default ordem crescente (`decreasing=TRUE` para ordem decrescente)

```
> sort(pessoas$idade)
[1] 14 16 22 31 44 49 57 72 80 81

> sort(pessoas$idade, decreasing=TRUE)
[1] 81 80 72 57 49 44 31 22 16 14
```
  - `order()` retorna um vetor com as posições dos elementos ordenados, usado para se criar um data frame ordenado. Default ordem crescente (`decreasing=TRUE` para ordem decrescente)

```
> order(pessoas$idade)
[1] 3 6 4 9 1 5 10 7 8 2
```

## Ordenação do Data Frame

- Para se obter um data frame ordenado por uma determinada coluna

```
> pessoas[order(pessoas$idade), ]
```

	sexo	idade	peso	menor
3	masculino	14	62	TRUE
6	masculino	16	82	TRUE
4	feminino	22	79	FALSE
9	masculino	31	81	FALSE
1	feminino	44	71	FALSE
5	masculino	49	61	FALSE
10	feminino	57	66	FALSE
7	feminino	72	83	FALSE
8	masculino	80	79	FALSE
2	feminino	81	70	FALSE

Prof. Dr. Razer A N R Montão

SEPT/UFRP

55

55

## Ordenação do Data Frame

- Ordenação por duas ou mais colunas

```
> pessoas[order(pessoas$sexo, pessoas$idade), ]
```

	sexo	idade	peso	menor
4	feminino	22	79	FALSE
1	feminino	44	71	FALSE
10	feminino	57	66	FALSE
7	feminino	72	83	FALSE
2	feminino	81	70	FALSE
3	masculino	14	62	TRUE
6	masculino	16	82	TRUE
9	masculino	31	81	FALSE
5	masculino	49	61	FALSE
8	masculino	80	79	FALSE

Prof. Dr. Razer A N R Montão

SEPT/UFRP

56

56

## Ordenação do Data Frame

- Uma crescente e outra decrescente
- Use o operador -

```
> pessoas[order(pessoas$sexo, -pessoas$idade), ]
```

	sexo	idade	peso	menor
3	feminino	89	63	FALSE
9	feminino	85	80	FALSE
10	feminino	79	88	FALSE
2	feminino	58	60	FALSE
1	feminino	46	76	FALSE
4	feminino	28	77	FALSE
8	feminino	21	83	FALSE
6	masculino	71	89	FALSE
7	masculino	51	85	FALSE
5	masculino	15	58	TRUE

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

57

57



## Exercícios

- Crie o data frame anterior e execute os comandos, vendo os resultados
  - Ordene o data frame por peso
  - Ordene o data frame por sexo e peso, decrescentemente
  - Dê a maior idade nos dados (`max`)
  - Dê a média dos pesos (`mean`)
  - Mostrar as pessoas do sexo feminino que estão na base
  - Contar as pessoas do sexo feminino (`nrow`)

Prof. Dr. Razer A N R Montaña

SEPT/UFPR

58

58