Introdução ao Tratamento e Análise de Dados em R

Aula 4 - Limpando e organizando seus dados

Sérgio Rivero

PPGE-UFPA

16 de maio de 2019

FACECON Faculdade de Ciências Econômicas Programa de Pós-graduação em Economia da UFPA

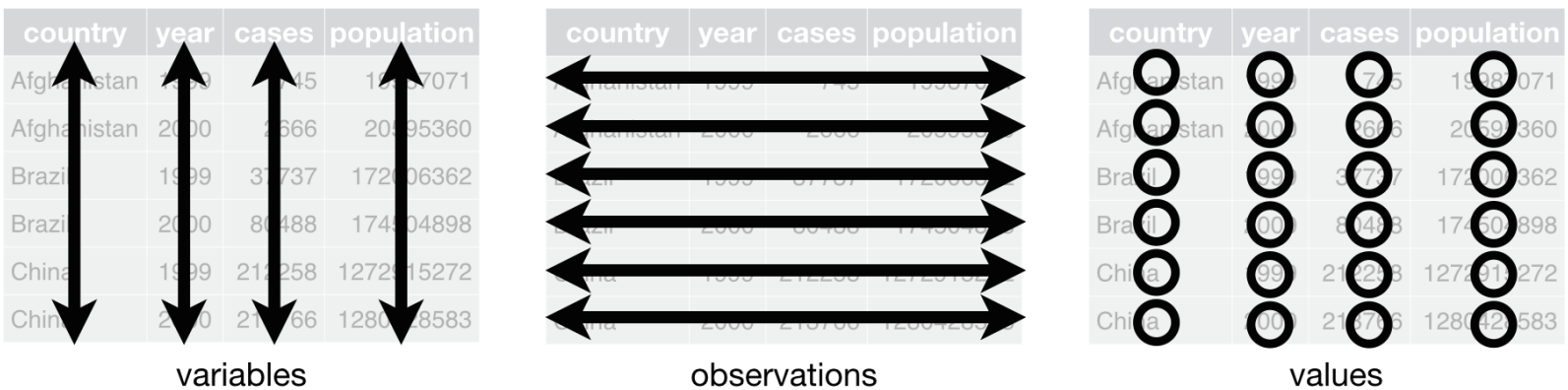
Sumário

Objetivos da Aula

Discutir algumas funções de correção e limpeza de bases de dados em R.

O que é uma boa base de dados e que tipos de bases existem?

Boas bases de dados para trabalhar, sobretudo com tidy, são bases que possuem variáveis nas colunas e observações nas linhas. Podemos identificar tal estrutura na figura 1 abaixo:



Organização ideal para base de dados

Tipos de Bases de Dados

Há três tipos bases de dados principais que comportam a estrutura de variáveis nas colunas e observações nas linhas que são:

- Séries Temporais
- Cortes Transversais
- Dados em Painel

Série Temporal

Séries temporais(ou longitudinais) são identificadas como observações ordenadas ao longo do "tempo". Exemplos:

- PIB distribuído anualmente;
- Arrecadação mensal de um ou mais Estados;
- Taxa de câmbio diária.

A base do BCB

Um bom exemplo de fonte de bases de dados de series temporais é o SGS - Banco Central. Segue o link abaixo:

Parâmetros informados

Séries selecionadas

20783 - Spread médio das operações de crédito - Total

Período	Função
01/03/2018 a 27/03/2019	Linear

Registros encontrados por série: **12**

Lista de valores (Formato numérico: Europeu - 123.456.789,00)

Data mês/AAAA	20783 p.p.
mar/2018	19,93
abr/2018	19,56
mai/2018	18,46
jun/2018	17,68
jul/2018	17,69
ago/2018	17,58
set/2018	17,26
out/2018	17,88
nov/2018	18,08
dez/2018	16,94
jan/2019	18,58
fev/2019	19,04
Fonte	BCB-DSTAT

Exemplo de série temporal

`https://www3.bcb.gov.br/sgspub/localizarseries/localizarSeries.do?method=prepararTelaLocalizarSeries`

Corte Transversal

Dados transversais(ou cross-section) são identificados quando não há importância de ordenação das observações, além de representar dados em um único ponto do "tempo". Exemplos:

- Arrecadação de ICSM para todos Estados do Brasil para o ano de 2018;
- Dados demográficos para todos os municípios do Pará para o ano de 2010.

Um bom exemplo de fonte de bases de dados de cortes transversais é o Sidra - IBGE. Segue o link abaixo:
`https://sidra.ibge.gov.br/home`

A Série

Tabela 3939 - Efetivo dos rebanhos, por tipo de rebanho	
Variável - Efetivo dos rebanhos (Cabeças)	
Ano - 2017	
Tipo de rebanho - Bovino	
Município	
Abaetetuba (PA)	2.800
Abel Figueiredo (PA)	46.364
Acará (PA)	10.740
Afuá (PA)	1.571
Água Azul do Norte (PA)	643.946
Alenquer (PA)	188.400
Almeirim (PA)	23.414
Altamira (PA)	656.430
Anajás (PA)	960

Exemplo de corte transversal

Operando com os Dataframes

Nesta seção utilizaremos a base de dados "starwars", que é uma base nativa do pacote dplyr. Para que possamos ter acesso a base, devemos utilizar a função a seguir:

```
library(dplyr)
data("starwars")
```

Funções de Remodelagem

A primeira função apresentada é a função **arrange**, que consiste em ordenar uma coluna(variável) de forma crescente ou decrescente. As função é escrita da seguinte forma:

```
starwars11 <- arrange(starwars, mass)

# Para ordenamento crescente

starwars12 <- arrange(starwars, desc(height))

#Para ordenamento decrescente
```

Funções de Remodelagem

Estão apresentadas a seguir as funções utilizadas para renomear rótulos(nomes) de colunas(variáveis).

```
starwars21 <- rename(starwars, nome = name)
starwars22 <- rename(starwars, altura = height)
starwars23 <- rename(starwars, massa = mass)
```

Funções de Filtragem

A função **filter** extrai linhas de uma base de dados a partir de um critério lógico. Essa função é bastante útil quando é preciso extrair um conjunto de dados que apresentam determinadas características. Por exemplo quando é preciso criar um novo objeto sem os outliers. A função está descrita abaixo:

```
starwars31 <- filter(starwars, mass < 70)
```

Funções de Filtragem

A função **distinct** remove todas as linhas que se encontram duplicadas na base de dados, permanecendo apenas uma de cada linha duplicada. Utilizamos a função da seguinte forma:
Observação: Para que possamos usar a função **distinct**, criamos o objeto *starwars32* com as linhas 31 até a 60 duplicadas.

```
starwars32 <- rbind(starwars, starwars[31:60,])
starwars32 <- distinct(starwars32)
```

distinct, sample

distinct remove todas as linhas que se encontram duplicadas na base de dados, permanecendo apenas uma de cada linha duplicada. Utilizamos a função da seguinte forma:

```
starwars32 <- rbind(starwars, starwars[31:60,])
starwars32 <- distinct(starwars32)
```

sample_frac seleciona aleatoriamente o número de linhas correspondentes a fração determinada na função.
sample_n seleciona um número *N* de linhas. Onde *N* é o número de linhas que devem ser extraídas.

```
starwars41 <- sample_frac(starwars, 0.4, replace = TRUE)
starwars42 <- sample_n(starwars, 25, replace = TRUE)
```

Selecionando colunas de seus dados

A função **select** seleciona colunas a partir dos rótulos especificados na função.

```
starwars61 <- select(starwars, name, height, mass)
starwars62 <- select(starwars, name, gender, species)
```

Resumindo

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**
Extract rows that meet logical criteria.

dplyr::**distinct(iris)**
Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**
Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**
Randomly select n rows.

dplyr::**slice(iris, 10:15)**
Select rows by position.

dplyr::**top_n(storms, 2, date)**
Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**
Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains(".")**)
Select columns whose name contains a character string.

select(iris, **ends_with("Length")**)
Select columns whose name ends with a character string.

select(iris, **everything()**)
Select every column.

select(iris, **matches(".t.")**)
Select columns whose name matches a regular expression.

select(iris, **num_range("x", 1:5)**)
Select columns named x1, x2, x3, x4, x5.

select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.

select(iris, **starts_with("Sepal")**)
Select columns whose name starts with a character string.

select(iris, **Sepal.Length:Petal.Width**)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, **-Species**)
Select all columns except Species.

Algumas funçoes para alteração de dataframes

Combinando Datasets

a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

dplyr::**left_join(a, b, by = "x1")**

Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

dplyr::**right_join(a, b, by = "x1")**

Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

dplyr::**inner_join(a, b, by = "x1")**

Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

dplyr::**full_join(a, b, by = "x1")**

Join data. Retain all values, all rows.

Pipes

Além de todas essas funções apresentadas anteriormente, o dplyr permite concatenar funções por meio de um mecanismo chamado **pipe**. Para utilizar o pipe é preciso colocar tais símbolos:

- %>%

Abaixo temos um exemplo da utilização do pipe. O exemplo consiste em calcular média e desvio padrão a partir de um agrupamento por espécie da base de dados starwars.

```
starwars81 <- starwars %>%
group_by(species) %>%
          summarise(desvpad = sd(birth_year, na.rm = TRUE),
          avg = mean(birth_year, na.rm = TRUE)
         )
```

Gather

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

Exemplo da função gather: gather(table4a, ‘1999‘, ‘2000‘,key = ”year”, value = ”cases”)

FACECON Faculdade de Ciências Econômicas Programa de Pós-graduação em Economia da UFPA

gather

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

key value

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

Exemplo da função spread: spread(table2, type, count)

unite

table5

country	century	year
Afghan	19	99
Afghan	20	0
Brazil	19	99
Brazil	20	0
China	19	99
China	20	0

→

country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

Exemplo da função unite: unite(table5, century, year, col = ”year”, sep = ““)

separate

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174
C	1999	212K	1T
C	2000	213K	1T

Exemplo da função separate: separate(table3, rate, into = c("cases", "pop"))

separate_rows

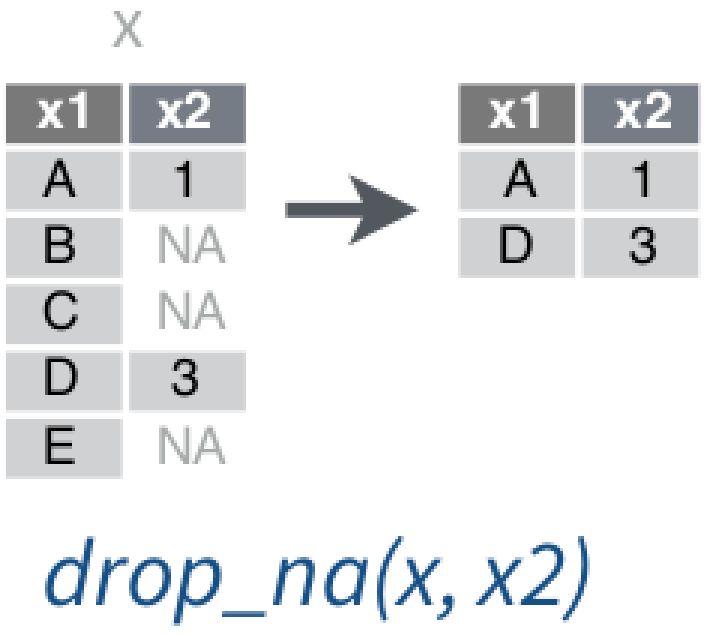
table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M
C	1999	212K
C	1999	1T
C	2000	213K
C	2000	1T

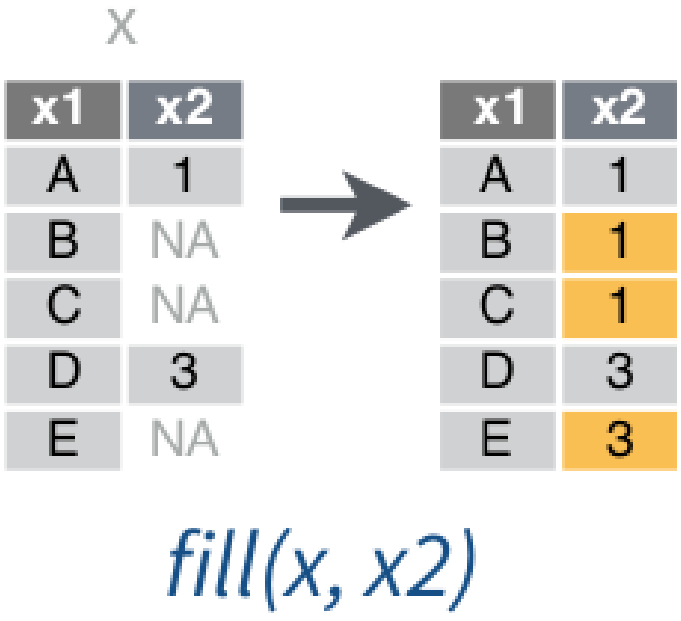
Exemplo da função separate_rows: separate_rows(table3, rate)

Valores Omissos (NA)



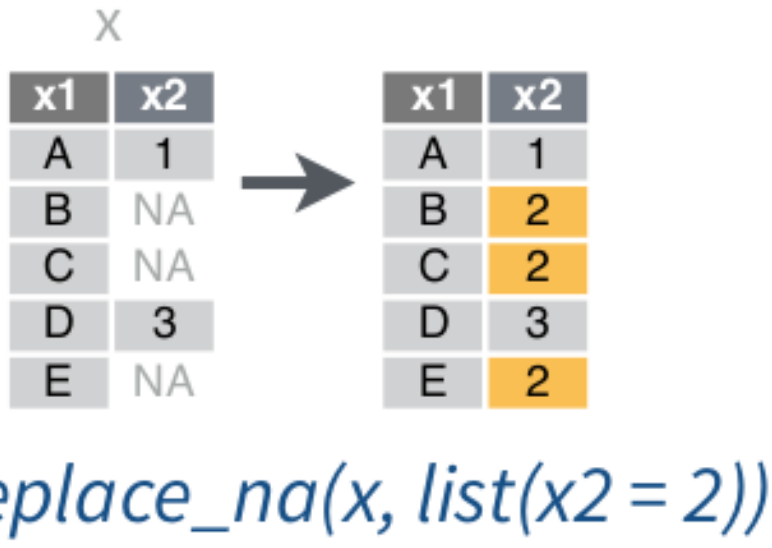
Exemplo da função drop_na: drop_na(x, x2)

Valores Omissos (NA)



Exemplo da função fill: fill(x,x2)

Valores Omissos (NA)



Exemplo da função replace_na: replace_na(x, list(x2 = 2))

Dúvidas?????

Esclarecendo as Dúvidas

Avaliação

Exercícios