

Universidade Federal do Rio Grande do Norte
Instituto Metr pole Digital
IMD0601 - Bioestat stica

Organiza  o dos dados em R

Prof. Dr. Tetsu Sakamoto
Instituto Metr pole Digital - UFRN
Sala A224, ramal 182
Email: tetsu@imd.ufrn.br



Baixe a aula (e os arquivos)

- Para aqueles que não clonaram o repositório:

```
> git clone https://github.com/tetsufmbio/IMD0601.git
```

- Para aqueles que já tem o repositório local:

```
> cd /path/to/IMD0601
```

```
> git pull
```

iris.data e iris.name

Source: <https://archive.ics.uci.edu/ml/datasets/iris>



Dados de comprimento e largura de pétalas e sépalas de três variedades de íris;

- iris.data → contém os dados;
- Iris.names → contém algumas descrições sobre os dados;

Data Set Characteristics:	Multivariate	Number of Instances:	150	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	4	Date Donated	1988-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	2411032

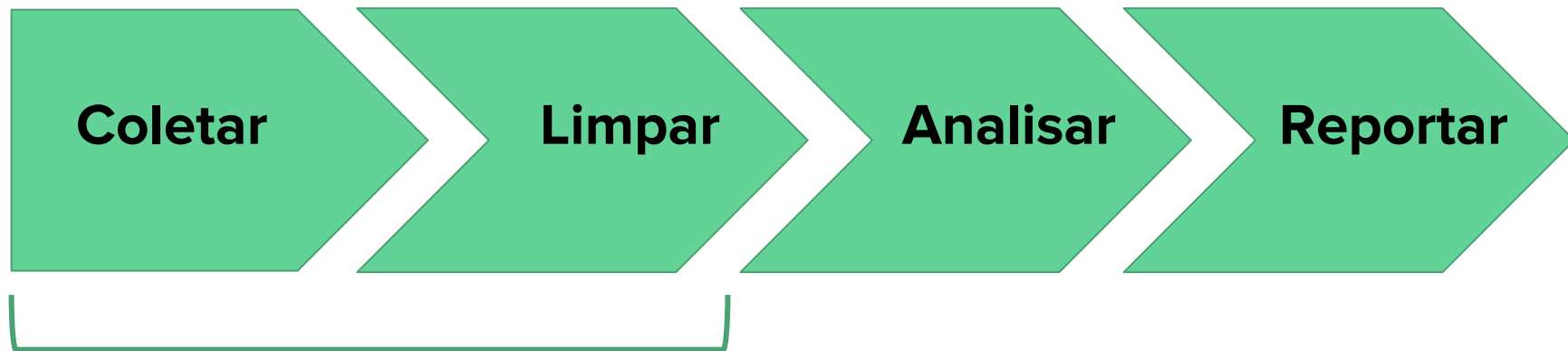
iris.data e iris.name



Exercício:

1. Veja a documentação da função `read.csv()` e carregue os dados do `iris.data`;
2. Veja a descrição de cada coluna dos dados em `iris.names` e nomeie as colunas;
3. Explore os dados utilizando as funções vistas nesta aula e responda:
 - a. Qual a dimensão da tabela?
 - b. Quais são os tipos de dados de cada coluna?
 - c. Quanto estes dados estão ocupando na memória?
 - d. Quantas amostras estão presentes em cada classe de Íris?
 - e. Tem algum dado faltante na tabela?

Procedimentos da análise de dados



50 a 80% do tempo da
análise de dados

Limpendo os dados

- Explorar os dados
 - Entender a sua estrutura;
 - Verificar a qualidade dos dados;
- Tidying data (arrumando os dados)
 - Facilitar a manipulação e a análise dos dados;



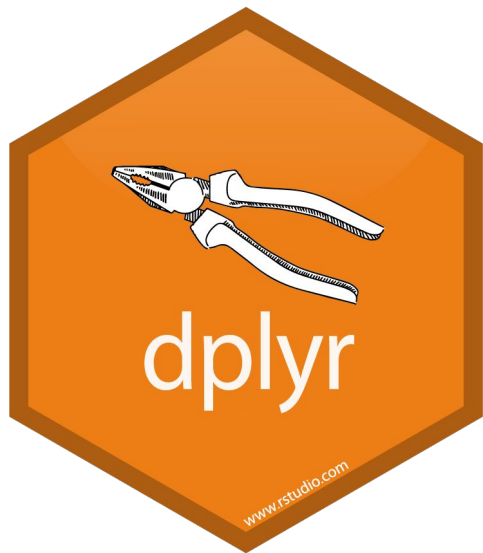
Explorando os dados

<code>dim(v)</code>	Mostra as dimensões de uma tabela v (linha x coluna);
<code>nrow(v)</code>	Mostra o número de linhas em v;
<code>ncol(v)</code>	Mostra o número de colunas em v;
<code>names(v)</code>	Mostra os nomes das colunas;
<code>object.size(v)</code>	Mostra o espaço ocupado na memória por v;
<code>head(v)</code>	Mostra as 6 primeiras linhas de v;
<code>tail(v)</code>	Mostra as 6 últimas linhas de v;
<code>summary(v)</code>	Mostra um breve sumário de v;
<code>table(v)</code>	Conta o número de ocorrência de cada categoria em v;
<code>str(v)</code>	Mostra a estrutura de v;

Explorando os dados

Pacote dplyr

- <https://dplyr.tidyverse.org/>
- Escrito por Hadley Wickham e Romain Francois;
- Oferece funções consistentes e concisas para manipular dados tabulares.



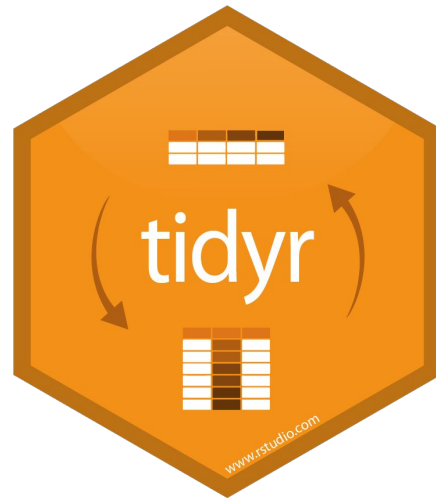
Pacote dplyr

- `select()` → seleciona as variáveis (colunas) desejadas;
 - `select(<tbl_df>, <var1>, <var2>, ...)`
 - `select(cran, ip_id, package, country)`
- `filter()` → filtra as linhas de acordo com as condições especificadas;
 - `filter(<tbl_df>, <condition1>, <condition2>, ...)`
 - `filter(cran, r_version == "3.1.1", country == "US")`
- `arrange()` → ordena as linhas na ordem crescente ou decrescente;
 - `arrange(<tbl_df>, <var1>, desc(<var2>), ...)`
 - `arrange(cran2, country, desc(r_version), ip_id)`
- `mutate()` → adiciona colunas;
 - `mutate(<tbl_df>, <new_var1> = <value>, <new_var2> = <value>, ...)`
 - `mutate(cran3, size_mb = size / 2^20, size_gb = size_mb / 2^10)`
- `summarize()` → reduz múltiplos valores a um sumário;
 - `summarize(<tbl_df>, <column> = <function>)`
 - `summarize(cran, avg_bytes = mean(size))`

Tidying data

Pacote tidyr

- Hadley Wickham
- Paper: “Tidy data” (<http://vita.had.co.nz/papers/tidy-data.pdf>)
- Tidy data são dados formatados de forma padronizada que facilita a exploração e a análise dos dados. Um tidy data satisfaz as seguintes condições:
 - Cada variável deve estar em uma coluna;
 - Cada observação forma uma linha;
 - Cada tipo de unidade observacional forma uma tabela.



Messy data 1

```
> students
```

	grade	male	female
1	A	1	5
2	B	5	0
3	C	5	2
4	D	5	5
5	E	7	4

Problema: Colunas **male** e **female** representam valores de uma variável

```
gather(<tbl>, <key>, <value>,  
<columns>)
```

```
> gather(students, sex, count, -grade)
```

	grade	sex	count
1	A	male	1
2	B	male	5
3	C	male	5
4	D	male	5
5	E	male	7
6	A	female	5
7	B	female	0
8	C	female	2
9	D	female	5
10	E	female	4

Messy data 2

```
> students2
  grade male_1 female_1 male_2 female_2
1     A      3       4      3       4
2     B      6       4      3       5
3     C      7       4      3       8
4     D      4       0      8       1
5     E      1       1      2       7
```

Problema: Múltiplas variáveis armazenadas em uma coluna (sexo e classe)

Messy data 2

```
> res <- gather(students2, sex_class, count, -grade)
> res
```

	grade	sex_class	count
1	A	male_1	3
2	B	male_1	6
3	C	male_1	7
4	D	male_1	4
5	E	male_1	1
6	A	female_1	4
7	B	female_1	4
8	C	female_1	4
...			

Problema: Duas variáveis distintas em uma coluna (sex_class)

Messy data 2

```
> separate(data = res, col = sex_class, into = c("sex", "class"))
```

	grade	sex	class	count
1	A	male	1	3
2	B	male	1	6
3	C	male	1	7
4	D	male	1	4
5	E	male	1	1
6	A	female	1	4
7	B	female	1	4
8	C	female	1	4
9	D	female	1	0

...

Messy data 3

Problema: Variáveis armazenadas tanto nas colunas quanto nas linhas.

```
> students3
```

	name	test	class1	class2	class3	class4	class5
1	Sally	midterm	A	<NA>	B	<NA>	<NA>
2	Sally	final	C	<NA>	C	<NA>	<NA>
3	Jeff	midterm	<NA>	D	<NA>	A	<NA>
4	Jeff	final	<NA>	E	<NA>	C	<NA>
5	Roger	midterm	<NA>	C	<NA>	<NA>	B
6	Roger	final	<NA>	A	<NA>	<NA>	A
7	Karen	midterm	<NA>	<NA>	C	A	<NA>
8	Karen	final	<NA>	<NA>	C	A	<NA>
9	Brian	midterm	B	<NA>	<NA>	<NA>	A
10	Brian	final	B	<NA>	<NA>	<NA>	C

Deveriam ser valores da variável class

Cada um deveria ser uma variável tendo como valor a nota

Messy data 3

```
> res <- gather(students3, class, grade, class1:class5, na.rm=TRUE)
> res
```

	name	test	class	grade
1	Sally	midterm	class1	A
2	Sally	final	class1	C
9	Brian	midterm	class1	B
10	Brian	final	class1	B
13	Jeff	midterm	class2	D
14	Jeff	final	class2	E
15	Roger	midterm	class2	C
16	Roger	final	class2	A
...				

Juntando as colunas class1 a class5 de forma que eles se tornem valores da coluna **class**, e seus valores da tabela antiga fiquem na coluna **grade**.

Messy data 3

```
> res <- spread(res, test, grade)
```

```
> res
```

	name	class	final	midterm
1	Brian	class1	B	B
2	Brian	class5	C	A
3	Jeff	class2	E	D
4	Jeff	class4	C	A
5	Karen	class3	C	C
6	Karen	class4	A	A
7	Roger	class2	A	C
8	Roger	class5	A	B
...				

Pega os valores distintos da coluna **test** e transforma cada uma em uma coluna (**final**, **midterm**). Seus valores são preenchidos pelos valores da coluna **grade**.

Messy data 3

```
> library(readr)
> mutate(res, class = parse_number(class))
```

	name	class	final	midterm
1	Brian	1	B	B
2	Brian	5	C	A
3	Jeff	2	E	D
4	Jeff	4	C	A
5	Karen	3	C	C
6	Karen	4	A	A
7	Roger	2	A	C
8	Roger	5	A	B
...				

parse_number() retira qualquer caracteres não numéricos que esteja antes ou depois do primeiro número. Uso do **mutate()** para transformar os valores da coluna **class** em apenas número.

Messy data 4

```
> students4
```

	id	name	sex	class	midterm	final
1	168	Brian	F	1	B	B
2	168	Brian	F	5	A	C
3	588	Sally	M	1	A	C
4	588	Sally	M	3	B	C
5	710	Jeff	M	2	D	E
6	710	Jeff	M	4	A	C
7	731	Roger	F	2	C	A
8	731	Roger	F	5	B	A
9	908	Karen	M	3	C	C
10	908	Karen	M	4	A	A

Problema: Múltiplas unidades observacionais em uma única tabela. Dados redundantes na tabela.

Messy data 4

```
> student_info <- select(students4, id, name, sex)
```

```
> student_info
```

	id	name	sex
1	168	Brian	F
2	168	Brian	F
3	588	Sally	M
4	588	Sally	M
5	710	Jeff	M
6	710	Jeff	M
7	731	Roger	F
8	731	Roger	F
...			

Selecionando as colunas **id**,
name e **sex** da tabela
students4.

Messy data 4

```
> unique(student_info)
```

	id	name	sex
1	168	Brian	F
3	588	Sally	M
5	710	Jeff	M
7	731	Roger	F
9	908	Karen	M

Usando a função **unique()**
para retirar as linhas
duplicadas.

Messy data 4

```
> gradebook <- select(students4, id, class, midterm, final)
```

```
> gradebook
```

	id	class	midterm	final
1	168	1	B	B
2	168	5	A	C
3	588	1	A	C
4	588	3	B	C
5	710	2	D	E
6	710	4	A	C
7	731	2	C	A
8	731	5	B	A

...

Retirar as colunas incorporadas na outra tabela usando o **select()**, mas mantendo a coluna **id**, pois ele funcionaria como uma chave primária em um banco de dados relacional.

Messy data 5

> passed

	name	class	final
1	Brian	1	B
2	Roger	2	A
3	Roger	5	A
4	Karen	4	A

> failed

	name	class	final
1	Brian	5	C
2	Sally	1	C
3	Sally	3	C
4	Jeff	2	E
5	Jeff	4	C
6	Karen	3	C

Problema: Uma unidade observacional em duas tabelas separadas.

Messy data 5

```
> passed <- mutate(passed,  
status = "passed")
```

```
> passed
```

	name	class	final	status
1	Brian	1	B	passed
2	Roger	2	A	passed
3	Roger	5	A	passed
4	Karen	4	A	passed

Usando o **mutate()** para adicionar uma coluna **status**.

```
> failed <- mutate(failed,  
status = "failed")
```

```
> failed
```

	name	class	final	status
1	Brian	5	C	failed
2	Sally	1	C	failed
3	Sally	3	C	failed
4	Jeff	2	E	failed
5	Jeff	4	C	failed
6	Karen	3	C	failed

Messy data 5

```
> bind_rows(passed, failed)
```

	name	class	final	status
1	Brian	1	B	passed
2	Roger	2	A	passed
3	Roger	5	A	passed
4	Karen	4	A	passed
5	Brian	5	C	failed
6	Sally	1	C	failed
7	Sally	3	C	failed
8	Jeff	2	E	failed
9	Jeff	4	C	failed
10	Karen	3	C	failed

Usando a função **bind_rows()** para juntar duas tabelas.

Exercício

Veja os dados dentro da variável **sat**. Tente realizar as seguintes tarefas:

1. Selecione todas as colunas que não contenha “total” (veja a função `contain()`);
2. Junte todas as colunas, exceto a coluna `score_range`. Utilize `key = part_sex` e `value = count`;
3. Separe a coluna `part_sex` em duas variáveis chamados de “part” e “sex”.
4. Use a função `group_by()` para agrupar as amostras pela coluna “part” e “sex” nesta ordem.
5. Utilize `mutate()` para adicionar duas novas colunas cujo os valores serão:
 - a. `total = sum(count)`
 - b. `prop = count / total`

Referência

Esta aula foi baseada no curso **Getting and Cleaning Data** do grupo **Swirl**.

Mais informações em: <https://swirlstats.com/scn/getclean.html>