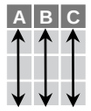
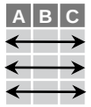
Organizando dados com tidyr : : **Folha de Referência**

Dados organizados (tidy) é uma forma de estruturar dados tabulares em uma forma consistente através dos pacotes.
Uma tabela é dita organizada (tidy) se:



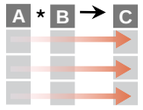
Cada variável está em sua própria coluna

&



Cada observação, ou caso, está em sua própria linha

Acessar variáveis como vetores



Preservar observações em operações vetorizadas

Tibbles

UM DATA FRAME MELHORADO

Tibbles são formatos de tabelas fornecidos pelo pacote tibble.
Eles herdam a classe data frame, mas com comportamentos melhorados:

- Gera uma parte com], um vetor com [[and $.
- Sem encontro parcial quando extrai colunas.
- Mostra os dados de maneira concisa na tela.

options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf) Alterar os parâmetros padrão para mostrar na tela.

View() ou glimpse() para ver todo o conjunto de dados.

CONSTRUINDO UMA TIBBLE

tibble(…) Cria por colunas.

```
tibble(x = 1:3, y = c("a", "b", "c"))
```

tribble(…) Cria por linhas.

```
tribble(~x,  ~y,
        1, "a",
        2, "b",
        3, "c")
```

Ambos criam esta tibble

```
A tibble: 3 × 2
      x     y
  <int> <chr>
1     1     a
2     2     b
3     3     c
```

as_tibble(x, …**)** Converte um data frame para tibble.

enframe(x, name = "name", value = "value"**)** Converte um vertor nomeado para uma tibble. Ver também **deframe()**.

is_tibble(x**)** Testa se x é uma tibble.

Reformatando Dados - Inverte dados para reorganizar valores em um novo layout.

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

pivot_longer(data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE**)**

"Alonga" dados juntando várias colunas em duas. Nomes das colunas vão para coluna names_to e valores vão para a nova coluna values_to.

```
pivot_longer(table4a, cols = 2:3, names_to ="year",
  values_to = "cases")
```

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

→

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

pivot_wider(data, names_from = "name", values_from = "value"**)**

Inverso da pivot_longer(). "Expande" dados de duas colunas em várias. Uma coluna fornece os nomes para as novas colunas e outra os valores.

```
pivot_wider(table2, names_from = type,
  values_from = count)
```

Dividindo Células - Funções para dividir ou combinar células em valores individuais

table5

country	century	year
A	19	99
A	20	00
B	19	99
B	20	00

→

country	year
A	1999
A	2000
B	1999
B	2000

unite(data, col, …, sep = "_", remove = TRUE, na.rm = FALSE**) Combina células de várias colunas em um única coluna**.

```
unite(table5, century, year, col = "year", sep = "")
```

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M

separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", …**)** Separa cada célula de uma coluna em várias colunas. Ver também **extract()**.

```
separate(table3, rate, sep = "/",
  into = c("cases", "pop"))
```

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

→

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

separate_rows(data, …, sep = "[^[:alnum:].]+", convert = FALSE**)** Separa cada célula de uma coluna em várias linhas.

```
separate_rows(table3, rate, sep = "/")
```

Expandir Tabelas

Create new combinations of variables or identify implicit missing values (combinations of variables not present in the data).

x

x1	x2	x3
A	1	3
B	1	4
B	2	3

→

x1	x2
A	1
A	2
B	1
B	2

expand(data, …**)** Create a new tibble with all possible combinations of the values of the variables listed in … Drop other variables.

```
expand(mtcars, cyl, gear,
carb)
```

x

x1	x2	x3
A	1	3
B	1	4
B	2	3

→

x1	x2	x3
A	1	3
A	2	NA
B	1	4
B	2	3

complete(data, …, fill = list()**)** Add missing possible combinations of values of variables listed in … Fill remaining variables with NA.

```
complete(mtcars, cyl, gear,
carb)
```

Valores Ausentes

Ignora ou substitui valores ausentes (NA).

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3

drop_na(data, …**) Ignora linhas contendo** NA's nas colunas.

```
drop_na(x, x2)
```

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	1
C	1
D	3
E	3

fill(data, …, .direction = "down"**)** Preenche os NA's nas colunas usando o valore anterior ou seguinte.

```
fill(x, x2)
```

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	2
C	2
D	3
E	2

replace_na(data, replace**)** Especifica um valor para substituir NA nas colunas selectionadas.

```
replace_na(x, list(x2 = 2))
```

RStudio

Dados Aninhados

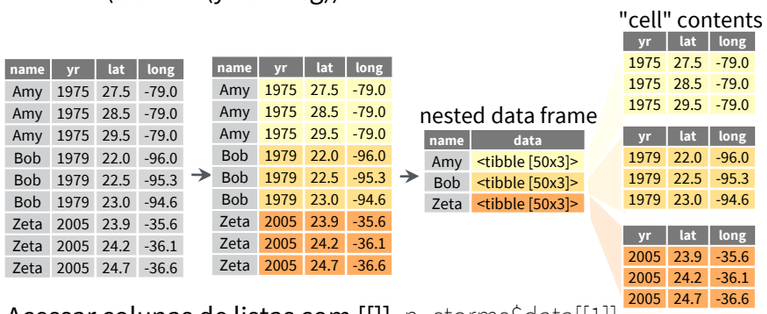
Um **data frame aninhado** armazena tabelas completas em colunas do tipo lista (colunas de lista) dentro de outro data frame maior e organizado. Uma coluna de lista, pode ter também listas de vetores ou listas de vários tipos de dados.
Use um data frame aninhado para:

- Preservar o relacionamento entre observações e sub-grupos de dados. Preservar os tipos de dados das variáveis aninhadas (fatores e datahora são sofrem coerção para caracteres).
- Manipular várias sub-tabelas de uma vez usando funções **purrr** com map(), map2(), ou pmap() ou com grupos rowwise() do **dplyr**.

CRIANDO DADOS ANINHADOS

nest(data, …**)** Move grupos de células para uma coluna de lista de um data frame. Use sozinho ou com dplyr::**group_by()**:

1. Agrupe o data frame com **group_by()** e use **nest()** para mover os grupos para a coluna de lista.
```
n_storms <- storms %>%
  group_by(name) %>%
  nest()
```
2. Use **nest(new_col = c(x, y))** para definir as colunas que serão agrupadas
   using dplyr::**select()** syntax.
```
n_storms <- storms %>%
  nest(data = c(year:long))
```



Acessar colunas de listas com [[]]. n_storms$data[[1]]

CRIAR TIBBLES COM COLUNA DE LISTA

tibble::**tribble(**…**)** Para criar uma coluna de lista quando precisar.

```
tribble( ~max, ~seq,
          3,   1:3,
          4,   1:4,
          5,   1:5)
```

max	seq
3	<int [3]>
4	<int [4]>
5	<int [5]>

tibble::**tibble(**…**)** Salva lista como uma colunas de lista.
tibble(max = c(3, 4, 5), seq = list(1:3, 1:4, 1:5))

tibble::**enframe(**x, name="name", value="value"**)**
Converte listas multinível em um tibble com coluna de lista.
enframe(list('3'=1:3, '4'=1:4, '5'=1:5), 'max', 'seq')

GERAR COLUNA DE LISTA DE OUTRAS FUNÇÕES

dplyr::**mutate()**, **transmute()**, e **summarise()** irão gerar uma coluna de lista se retornarem uma lista.

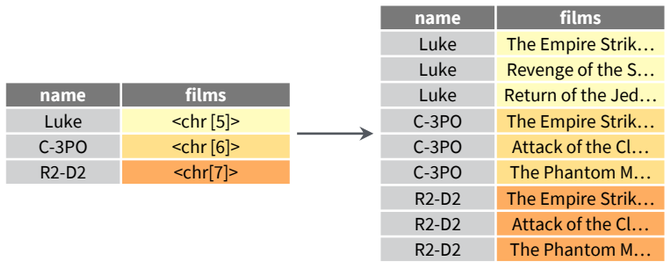
```
mtcars %>%
  group_by(cyl) %>%
  summarise(q = list(quantile(mpg)))
```

REFORMATANDO DADOS ANINHADOS

unnest(data, cols, ..., keep_empty = FALSE**)** Expande colunas anihadas de volta a sua forma original. Inverso da nest().
n_storms %>% unnest(data)

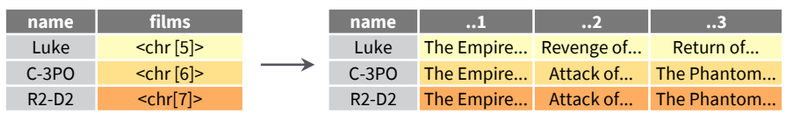
unnest_longer(data, col, values_to = NULL, indices_to = NULL**)**
Move cada elemento da coluna de lista para uma linha.

```
starwars %>%
  select(name, films) %>%
  unnest_longer(films)
```



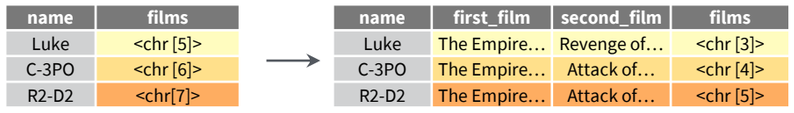
unnest_wider(data, col**)** Move cada elemento da coluna de lista para uma coluna.

```
starwars %>%
  select(name, films) %>%
  unnest_wider(films)
```



hoist(.data, .col, ..., .remove = TRUE) Extrai seletivamente elementos da lista para as colunas de nível superior. Use a sintaxe da purrr::pluck() para selecionar elementos da lista.

```
starwars %>%
  select(name, films) %>%
  hoist(films, first_film = 1, second_film = 2)
```



TRANSFORMANDO DADOS ANINHADOS

Uma função vetorizada recebe um vetor, transforma cada elemento em paralelo e retorna um vetor de mesmo tamanho que o vetor de entrada. Estas funções sozinhas não trabalham com listas, e consequentemente, não trabalham com colunas de listas.

dplyr::**rowwise(**.data, …**)** agrupa cada linha da tabela em um grupo diferente e dentro de cada grupo os elementos da coluna de lista aparecem diretamente (acessados por [[]) e não mais como uma lista e tamanho igual a um. **Quando usamos a rownames(), as funções vetorizadas do pacote dplyr poderão ser aplicadas em uma coluna de lista de uma forma vetorizada**

Aplica uma função a uma coluna de lista e **cria uma nova coluna de lista.**

```
n_storms %>%
  rowwise() %>%
  mutate(n = list(dim(data)))
```

dim() retorna dois valores por linha

Use list() para fazer mutate criar uma coluna de lista

Aplica uma função a uma coluna de lista e **cria uma coluna normal.**

```
n_storms %>%
  rowwise() %>%
  mutate(n = nrow(data))
```

nrow() retorna um inteiro por linha

Combina **colunas de lista multinível** em uma única coluna de lista**.**

```
starwars %>%
  rowwise() %>%
  mutate(transport = list(append(vehicles, starships)))
```

append() retorna uma lista para cada linha, então usamos listy() para criar uma coluna de lista

Aplica uma função a uma **colunas de lista multinível .**

```
starwars %>%
  rowwise() %>%
  mutate(n_transports = length(c(vehicles, starships)))
```

length() retorna um inteiro por linha

See **purrr** package for more list functions.