Organizando Datos con tidyr : : GUÍA RÁPIDA

Los datos ordenados son una forma de organizar los datos tabulares en una estructura de datos coherente en todos los paquetes.
Una mesa está ordenada si:

Cada variable está en su propia columna

Cada observación, o caso, está en su propia fila

Accede a **variables** como **vectores**

Conservar casos en operaciones vectorizadas

Tibbles

UN MARCO DE DATOS MEJORADO

Tibbles es un formato de tabla proporcionado por el paquete tibble. Heredan la clase de marco de datos, pero tienen comportamientos mejorados:

- **Subconjunto** de tibble con], un vector con [[y $.
- **No conicidencia parcial** al extraer columnas.
- **Muestra** vistas concisas de los datos en una sola pantalla.

options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf**)** contola la configuración predeterminada de como se representa un tibble.

View() or **glimpse()** Ver todo el conjunto de datos.

CONSTRUYE UN TIBBLE

tibble(…**)** Construir por columnas.
tibble(x = 1:3, y = c("a", "b", "c"))

tribble(…**)** Construir por filas.
tribble(~x, ~y,
1, "a",
2, "b",
3, "c")

Ambos hacen que este tibble

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

as_tibble(x, …**)** Convertir un marco de datos en un tibble.

enframe(x, name = "name", value = "value"**)** Convierte un vector con nombre en un tibble. Además, **deframe()**.

is_tibble(x**)** Comprueba si x es un tibble.

Cambiar la forma de los datos

- Dinamice los datos para reorganizar los valores en un nuevo diseño.

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**)**

"Alargue" los datos colapsando varias columnas en dos. Los nombres de columna se mueven a una nueva columna names_to y los valores a una nueva columna 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"**)**

El inverso de pivot_longer(). "Amplíe" los datos expandiendo dos columnas en varias. Una columna proporciona los nuevos nombres de columna, la otra los valores.

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

Dividir Celdas

- Utilice estas funciones para dividir o combinar celdas en valores individuales y aislados.

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**)** Contrae las celdas de varias columnas en una sola columna.

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_wider_delim(data, cols, delim, ..., names = NULL, names_sep = NULL, names_repair = "check unique", too_few, too_many, cols_remove = TRUE) Separe cada celda de una columna en varias columnas. Además, **separate_wider_regex()** y **separate_wider_position().**

separate_wider_delim(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_longer_delim(data, cols, delim, .., width, keep_eampty) Separe cada celda de una columna en varias filas.

separate_longer_delim(table3, rate, sep = "/")

Expandir Tablas

Cree nuevas combinaciones de variables o identifique valores faltantes implícitos (combinaciones de variables que no están presentes en los datos).

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, …**)** Cree un nuevo tibble con todas las combinaciones posibles de los valores de las variables enumeradas en ...
Descarte otras 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()**)** Agregue las combinaciones posibles de valores de las variables enumeradas en ... Rellene las variables restantes con NA.
complete(mtcars, cyl, gear, carb)

Controlar Valores Faltantes

Quite o reemplace los valores faltantes explícitos (NA).

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

→

x1	x2
A	1
D	3

drop_na(data, …**)** Elimine filas que contengan NA en ... columnas.
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"**)** Rellene los NA en ... columnas que usan el valor siguiente o anterior.
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**)** Especifique un valor para reemplazar NA en las columnas seleccionadas.
replace_na(x, list(x2 = 2))

Datos Anidados

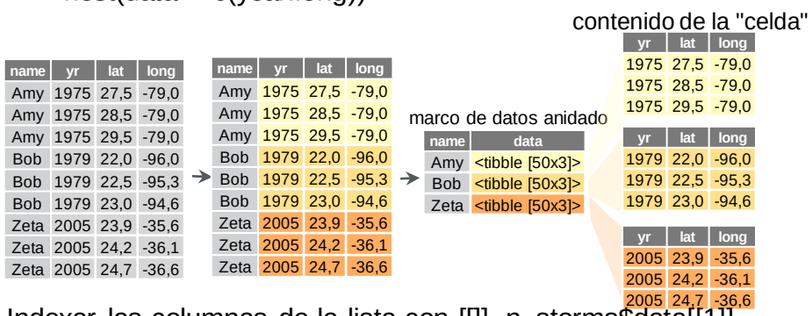
Un marco de datos anidado almacena tablas individuales como una columna de lista de marcos de datos dentro de un marco de datos de organización más grande. Las columnas de lista también pueden ser listas de vectores o listas de diferentes tipos de datos.
Utilice un marco de datos anidado para:

- Conserve las relaciones entre las observaciones y los subconjuntos de datos. Conserve el tipo de las variables que se van a anidar (los factores y las fechas y horas no se obligan a caracteres).
- Manipule muchas subtablas a la vez con funciones purrr como map(), map2() o pmap() o con la agrupación dplyr rowwise().

CREACIÓN DE DATOS ANIDADOS

nest(data, …**)** Mueve grupos de celdas a una columna de lista de un marco de datos. Úselo solo o con dplyr::**group_by()**:

1. Agrupe el marco de datos con **group_by()** y use **nest()** para mover los grupos a una columna de lista.
   n_storms <- storms |>
   group_by(name) |>
   nest()
2. Use **nest(new_col = c(x, y))** Para especificar las columnas que se van a agrupar
   usando la sintaxis dplyr::**select()**.
   n_storms <- storms |>
   nest(data = c(year:long))



Indexar las columnas de la lista con [[]]. n_storms$data[[1]]

CREAR TIBBLES CON LISTAS-COLUMNAS

tibble::**tribble(**…**)** Crea columnas de lista cuando es necesario.

```
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(**…**)** Guarda la entrada de la lista como columnas de lista.
tibble(max = c(3, 4, 5), seq = list(1:3, 1:4, 1:5))

tibble::**enframe(**x, name="name", value="value"**)**
Convierte una lista de varios niveles en un tibble con columnas de listas.
enframe(list('3'=1:3, '4'=1:4, '5'=1:5), 'max', 'seq')

COLUMNAS DE LISTA DE SALIDA DE OTRAS FUNCIONES

dplyr::**mutate()**, **transmute()**, y **summarise()** generarán columnas de lista si devuelven una lista.

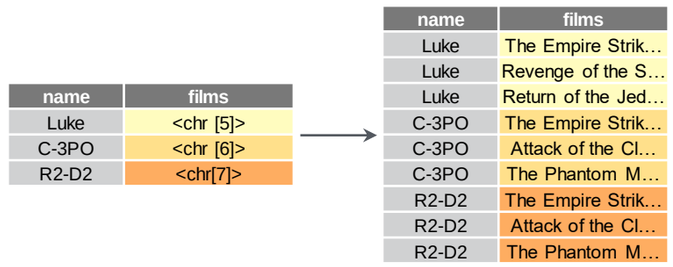
```
mtcars |>
  group_by(cyl) |>
  summarise(q = list(quantile(mpg)))
```

CAMBIAR LA FORMA DE LOS DATOS ANIDADOS

unnest(data, cols, ..., keep_empty = FALSE**)** Vuelva a aplanar las columnas anidadas a columnas normales. El inverso de nest().
n_storms |> unnest(data)

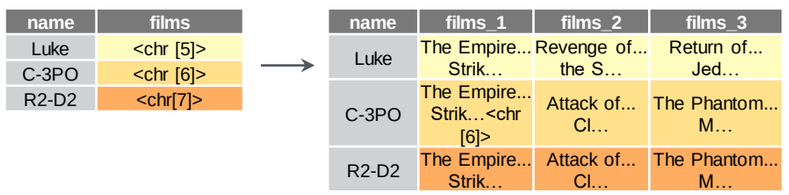
unnest_longer(data, col, values_to = NULL, indices_to = NULL**)** Convertir cada elemento de una columna de lista en una fila.

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



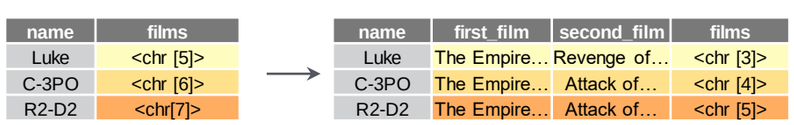
unnest_wider(data, col**)** Convierta cada elemento de una columna de lista en una columna normal.

```
starwars |>
  select(name, films) |>
  unnest_wider(films, names_sep =
"_")
```



hoist(.data, .col, ..., .remove = TRUE) Extraiga selectivamente los componentes de la lista en sus propias columnas de nivel superior. Utiliza la sintaxis purrr::pluck() para seleccionar de las listas.

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



TRANSFORMACIÓN DE DATOS ANIDADOS

Una función vectorizada toma un vector, transforma cada elemento en paralelo y devuelve un vector de la misma longitud. Por sí mismas, las funciones vectorizadas no pueden trabajar con listas, como las columnas de lista.

dplyr::**rowwise(**.data, …**)** Agrupe los datos de modo que cada fila sea un grupo, y dentro de los grupos, los elementos de las columnas-lista aparecen directamente (a los que se accede con [[ ), no como listas de longitud uno. **Cuando usas rowwise(), las funciones dplyr parecerán aplicar funciones a las columnas de la lista de forma vectorizada.**

Aplique una función a una columna de lista y **cree una nueva columna de listas**

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

dim() devuelve dos valores por fila

wrap con list para decirle mutar para crear una columna de lista

Aplique una función a una columna de lista y **crea una columna regular.**

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

nrow() devuelve un entero por fila

Contrae **múltiples columnas de listas** en una simple columna lista**.**

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

append() devuelve una lista para cada fila, por lo que el tipo col debe ser list

Aplicar una función a **múltiples columnas de listas.**

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

length() devuelve un entero por fila

vea **purrr** paquete para más funciones de lista.