R para contextos humanitarios de emergencia

Manipulación de datos II

Violeta Roizman

Abrir el archivo 05-EJ-manipulacionII.Rmd

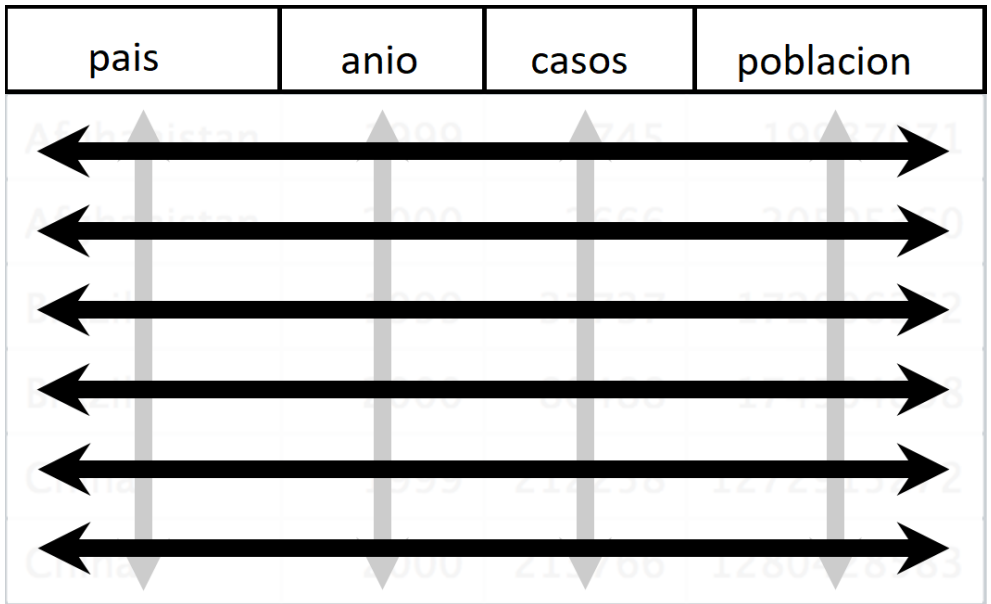
Para ir haciendo los EJ

Datos ordenados

El entorno tidyverse esta basado en el concepto de datos ordenados

Un dataset está ordenado si:

- Cada variable está en una columna
- Cada observacion está una fila
- Cada valor se encuentra en una celda



Transformando a tidy

Veamos la siguiente tabla de la cantidad total de los pedidos de asilo recibidas. Las variables involucradas son país, año y cantidad de cados. Está ordenada la tabla?

Pais	2011	2012	2013
ARG	7000	6900	7000
PER	5800	6000	6200
TUR	15000	14000	13000

Me gustaría que estuviera así:

Pais	Anio	cantidad
ARG	2011	7000
ARG	2012	6900
ARG	2013	7000
PER	2011	5800
PER	2012	6000
PER	2013	6200

Transformando a tidy: `pivot_longer`

La funcion `pivot_longer` se encarga de esto!

Pasa de una tabla ancha a una table mas larga.

Hay que idenfificar cuáles son las columnas que dejarán de ser fila y pasarán a ser los valores de una nueva columna. Estas columnas serán la llave de la transformación

Pais	2011	2012	2013
ARG	7000	6900	7000
PER	5800	6000	6200
TUR	15000	14000	13000

Pais	Anio	cantidad
ARG	2011	7000
ARG	2012	6900
ARG	2013	7000
PER	2011	5800
PER	2012	6000
PER	2013	6200
TUR	2011	15000
TUR	2012	14000
TUR	2013	13000

Transformando a tidy: `pivot_longer`

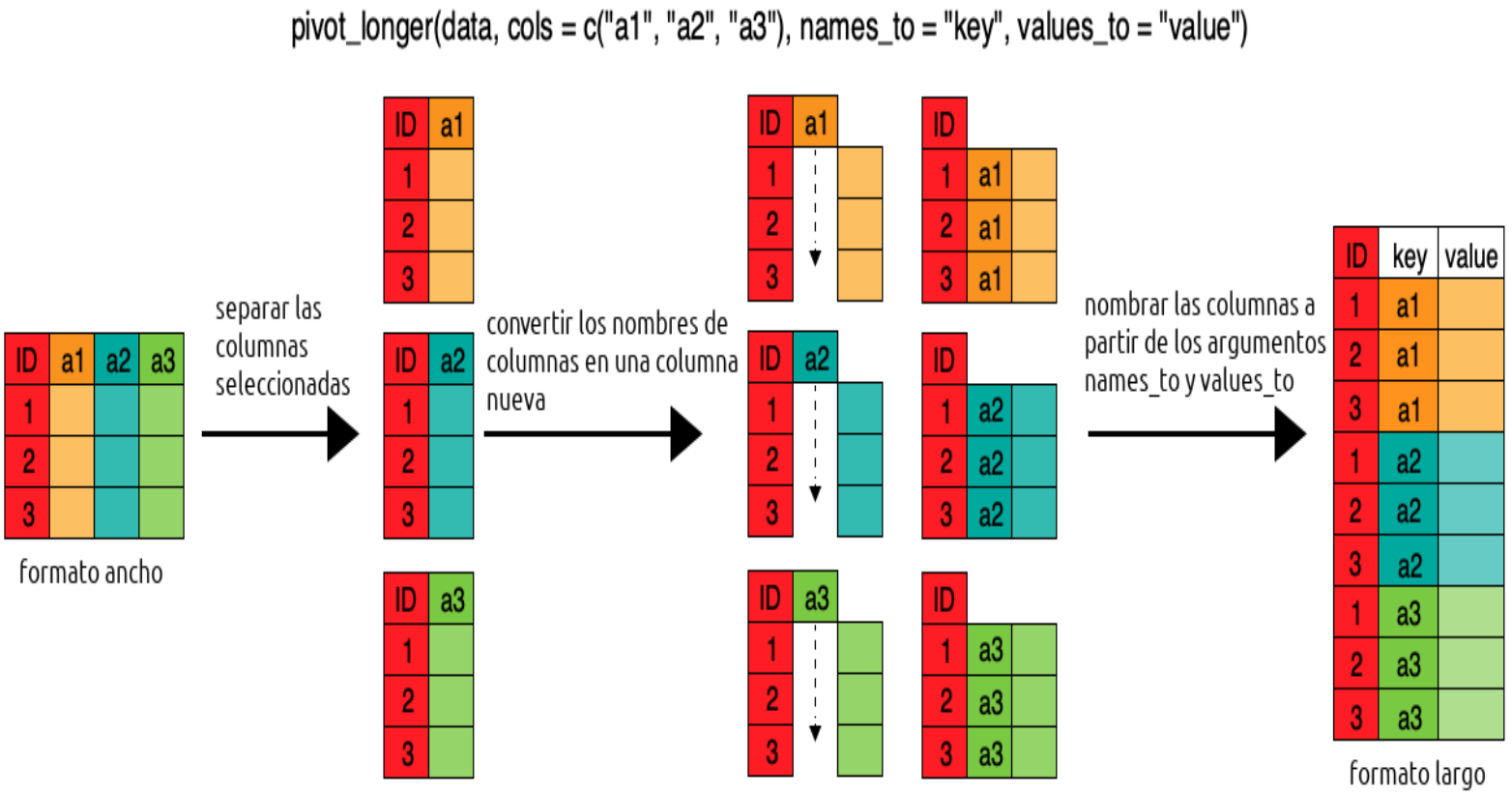
```
pedidos_asilo_tidy <- pivot_longer(pedidos_asilo,
                                   cols = c(2, 3, 4),
                                   names_to = "Anio",
                                   values_to = "cantidad")
```

- data corresponde a la tabla a modificar
- cols corresponde a las **columnas a ser pivoteadas**
- names_to corresponde al **nombre** que agrupará a las columnas de interés
- values_to corresponde al **nombre** de la variable que agrupará a los valores

Pais	Anio	cantidad
ARG	2011	7000
ARG	2012	6900
ARG	2013	7000
PER	2011	5800
PER	2012	6000
PER	2013	6200

`pivot_longer`: idea del funcionamiento

desde el **formato ancho** al **formato largo**



Transformando a tidy: `pivot_wider`

Por otro lado, tenemos

ciudad	tamanio_particula	cantidad
Cusco	p grande	23
Cusco	p pequenia	14
London	p grande	22
London	p pequenia	16
Beijing	p grande	121
Beijing	p pequenia	121

Transformando a tidy: `pivot_wider`

Que tampoco esta ordenada, pero en el sentido contrario (es larga en este caso !). Me gustaría que estuviera así:

ciudad	p grande	p pequenia
Cusco	23	14
London	22	16
Beijing	121	121

Debo identificar a las dos columnas que esconden más de una variable. En este caso, tamanio_particula contiene los nombres de las futuras variables (names_from), y cantidad esconde el nombre de la segunda variable

```
contaminacion_tidy <- pivot_wider(contaminacion,
                                  names_from = tamanio_particula,
                                  values_from = cantidad)
```

me devuelve el resultado deseado

Tu turno 1: `pivot_wider`

Covertir paises_largo al formato ordenado (variables como columnas)

```
paises_largo <- read_csv("data/paises_largo.csv")
knitr::kable(paises_largo[1:6,])
```

pais	continente	anio	variable	valor
Afganistán	Asia	1952	esperanza_de_vida	28.801
Afganistán	Asia	1957	esperanza_de_vida	30.332
Afganistán	Asia	1962	esperanza_de_vida	31.997
Afganistán	Asia	1967	esperanza_de_vida	34.020
Afganistán	Asia	1972	esperanza_de_vida	36.088
Afganistán	Asia	1977	esperanza_de_vida	38.438

Combinando tablas de datos!

Hasta ahora todo lo que usamos de `dplyr` involucra trabajar y modificar con una sola tabla a la vez

En ese caso, tenemos que unir estas tablas. a partir de una o más variables en común o keys.

En Excel:

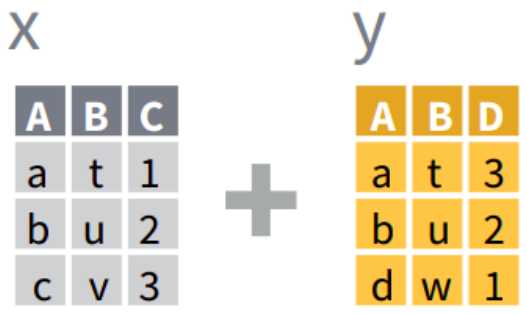
“VLOOKUP” o “BUSCARV”

En R:

familia de funciones *_join().

Hay una función cada tipo de unión que queramos hacer.

familia `join`

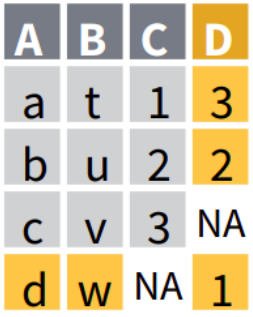


- `full_join()`: devuelve todas las filas y todas las columnas de ambas tablas `x` e `y`. Cuando no coinciden los elementos, devuelve `NA` (dato faltante). Esto significa que no se pierden filas de ninguna de las dos tablas aún cuando no hay coincidencia.

- `left_join()`: devuelve todas las filas de `x` y todas las columnas de `x` e `y`. Las filas en `x` que no tengan coincidencia con `y` tendrán `NA` en las nuevas columnas. Si hay múltiples coincidencias entre `x` e `y`, devuelve todas las coincidencias posibles.

- `inner_join()`: devuelve todas las filas de `x` donde hay coincidencias con `y` y todas las columnas de `x` e `y`. Si hay múltiples coincidencias entre `x` e `y`, entonces devuelve todas las coincidencias. Eliminará las filas (observaciones) que no coincidan en ambas tablas.

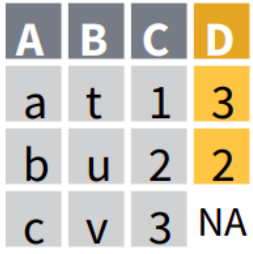
full_join



A	B	C	D
a	t	1	3
b	u	2	2
c	v	3	NA
d	w	NA	1

full_join(x, y, by = “A”**)**

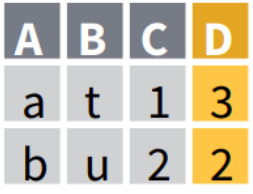
Une las filas que coinciden en x e y. En las filas donde no coincide agrega NA.



A	B	C	D
a	t	1	3
b	u	2	2
c	v	3	NA

left_join(x, y, by = “A”**)**

Une las filas que coinciden en x e y. Retiene todas las filas de x pero no de y.



A	B	C	D
a	t	1	3
b	u	2	2

inner_join(x, y, by = “A”**)**

Une las filas que coinciden en x e y. Retiene solo las filas donde hay coincidencia

Imagen del curso "R para Clima" - Paola Corrales y Elio Campitelli
https://eliocamp.github.io/r-clima/index.html

Tu turno 2: para practicar después

Unir la tabla de países obtenida en Tu turno 1 con la tabla de decisiones de asilo (decisiones_asilo_peru). Utiliza como llave a las columnas con el nombre del país y el año.

Licencia y material usado

Licencia: Creative Commons Attribution-ShareAlike 4.0 International License.

Este material está inspirado y utiliza explicaciones de:

- R para Clima de Paola Corrales y Elio Campitelli
- Master the Tidyverse de Garrett Grolemund