Módulo 3 — Tidyverse II

Joins (Left y Right)

CEPAL - Unidad de Estadísticas Sociales

2025-11-05

Introducción

En el análisis de datos, es frecuente combinar información proveniente de distintas fuentes.

El paquete **dplyr** del *tidyverse* ofrece la familia de funciones `join` para unir dos tablas basadas en una o más llaves comunes.

En esta sección exploraremos **left_join()** y **right_join()**, los métodos más utilizados para combinar bases en R.

Conceptos básicos

Una *unión* (join) se realiza entre dos data frames:

- **x**: tabla principal (izquierda)
- **y**: tabla secundaria (derecha)

Cada observación se combina según una **clave** o conjunto de claves compartidas (por ejemplo: `id_hogar` o `id_pers`).

Tipos de joins en dplyr

Función	Descripción breve
`left_join()`	Conserva todas las filas de la izquierda (x).
`right_join()`	Conserva todas las filas de la derecha (y).
`inner_join()`	Mantiene solo las coincidencias.
`full_join()`	Incluye todas las filas de ambas tablas.

Lectura de base de ejemplos

```
datos <- readRDS("data/base_personas_gasto.rds")
library(dplyr)
glimpse(datos)
```

```
Rows: 19,427
Columns: 17
$ id_hogar   <dbl> 262, 262, 265, 265, 265, 277, 277, 277, 277, 288, 288, 1
$ id_pers    <dbl> 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 5, 1
$ upm        <dbl> 1100100006, 1100100006, 1100100006, 1100100006, 11001000
$ estrato    <dbl> 11001, 11001, 11001, 11001, 11001, 11001, 11001, 11001,
$ area       <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1",
$ fep        <dbl> 19, 19, 16, 16, 16, 16, 16, 16, 16, 19, 19, 19, 19, 30,
$ pobreza    <chr> "3", "3", "3", "3", "3", "3", "3", "3", "3", "3", "3",
$ ingreso_hh <dbl> 10783.053, 10783.053, 21259.723, 21259.723, 21259.723, 9
$ gasto_hh   <dbl> 10783.053, 10783.053, 21259.723, 21259.723, 21259.723, 9
$ parentesco <chr> "1", "2", "1", "2", "3", "1", "3", "3", "3", "1", "2",
$ edad       <dbl> 51, 46, 26, 24, 7, 42, 20, 17, 13, 60, 32, 13, 5, 39, 3
$ sexo       <fct> Hombre, Mujer, Mujer, Hombre, Hombre, Mujer, Mujer, Homb
$ etnia      <fct> 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

Variables principales:

- `id_hogar`, `id_pers`: identificadores.

```
n_distinct(datos$id_hogar)
```

```
[1] 6219
```

- `ingreso_hh`, `gasto_hh`: variables de ingreso y gasto por hogar.

```
summary(datos$ingreso_hh)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0	5168.893	8175.31	12116.45	13373.84	1015275

```
summary(datos$gasto_hh)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0	5168.893	8175.31	11758.38	13363.07	294127.5

- `edad`, `sexo`, `etnia`: características individuales.

División de la base

Dividiremos la base en dos subconjuntos:

- `datos_hogar`: información a nivel de hogar.

```
set.seed(1245)
datos_hogar <- datos %>%
  select(id_hogar, ingreso_hh, gasto_hh, pobreza, area) %>%
  distinct() %>%
  sample_frac(size = .5)
```

- `datos_persona`: información individual.

```
set.seed(1235)
datos_persona <- datos %>%
  select(id_hogar, id_pers, edad, sexo, etnia, anoest) %>%
    sample_frac(size = .8)
```

Exploración de los subconjuntos

```
head(datos_hogar, 4)
```

id_hogar	ingreso_hh	gasto_hh	pobreza	area
25204	4229.1600	4229.1600	3	1
24097	703.0533	703.0533	1	1
46141	1183.0533	1183.0533	2	1
35209	4411.6959	4411.6959	2	1

```
head(datos_persona, 4)
```

id_hogar	id_pers	edad	sexo	etnia	anoest
56557	1	62	Hombre	0	12
3646	2	16	Mujer	0	10
23913	2	27	Mujer	0	7
9857	1	53	Hombre	0	12

Ejemplo de Left Join

Objetivo: añadir la información del hogar a cada persona, manteniendo todas las personas.

```
n_distinct(datos_hogar$id_hogar)
```

```
[1] 3322
```

```
datos_left <- datos_persona %>%
  left_join(datos_hogar, by = "id_hogar")
n_distinct(datos_left$id_hogar)
```

```
[1] 5948
```

Interpretación del Left Join

```
datos_left  %>% head(6)
```

id_hogar	id_pers	edad	sexo	etnia	anoest	ingreso_hh	gasto_hh	pobreza	area
56557	1	62	Hombre	0	12	5686.090	5686.090	3	1
3646	2	16	Mujer	0	10	NA	NA	NA	NA
23913	2	27	Mujer	0	7	2026.053	2026.053	2	1
9857	1	53	Hombre	0	12	8547.063	8547.063	3	1
57119	4	33	Hombre	0	11	26865.985	26865.985	3	1
10121	1	90	Mujer	0	6	4072.923	4072.923	3	1

- Todas las filas de `datos_persona` se conservan.
- Las variables de `datos_hogar` se añaden según coincidencia en `id_hogar`.
- Si algún hogar no existe en `datos_hogar`, se asignan valores `NA`.

Ejemplo de Right Join

Objetivo: mantener todos los hogares, incluso aquellos sin personas registradas.

```
n_distinct(datos_hogar$id_hogar)
```

```
[1] 3322
```

```
datos_right <- datos_persona %>%
  right_join(datos_hogar, by = "id_hogar")
n_distinct(datos_right$id_hogar)
```

```
[1] 3322
```

Resultados del right Join

```
datos_left  %>% head(6)
```

id_hogar	id_pers	edad	sexo	etnia	anoest	ingreso_hh	gasto_hh	pobreza	area
56557	1	62	Hombre	0	12	5686.090	5686.090	3	1
3646	2	16	Mujer	0	10	NA	NA	NA	NA
23913	2	27	Mujer	0	7	2026.053	2026.053	2	1
9857	1	53	Hombre	0	12	8547.063	8547.063	3	1
57119	4	33	Hombre	0	11	26865.985	26865.985	3	1
10121	1	90	Mujer	0	6	4072.923	4072.923	3	1

Comparación entre left y right join

Aspecto	Left Join	Right Join
Tabla principal	Personas	Hogares
Filas conservadas	Todas las personas	Todos los hogares
NA posibles	En variables del hogar	En variables de persona

Conclusiones de la sección

- `left_join()` es ideal cuando se desea conservar el universo principal (por ejemplo, personas).
- `right_join()` resulta útil al priorizar la información del hogar o marco maestro.
- Ambos son **simétricos** si se invierte el orden de los data frames.

Próxima sección

En la siguiente sección abordaremos las funciones:

- `full_join()`
- `inner_join()`
- `semi_join()`
- `anti_join()`

Analizaremos cómo filtran y combinan registros según coincidencias exactas entre bases.