Módulo 3 — Tidyverse II

Joins (full, inner, semi y anti)

CEPAL - Unidad de Estadísticas Sociales

2025-11-05

Introducción

La manipulación de datos relacionales es fundamental en el análisis estadístico aplicado. Con frecuencia se dispone de múltiples fuentes de información que comparten una o más llaves identificadoras (por ejemplo, `id_hogar` y `id_pers`), lo que requiere estrategias eficientes de unión.

El paquete **dplyr**, dentro del ecosistema **tidyverse**, ofrece una familia de funciones `*_join()` que permiten combinar marcos de datos de manera controlada, transparente y eficiente.

Base de ejemplo

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

División de la base original

- `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,ingreso) %>%
    sample_frac(size = .8)
```

inner_join()

Devuelve únicamente las observaciones **que existen en ambas bases**.

```
nrow(datos_persona)
```

```
[1] 15542
```

```
base_inner <- datos_hogar %>%
  inner_join(datos_persona, by = "id_hogar")
nrow(base_inner)
```

```
[1] 9399
```

- Filtra hogares que tienen al menos una persona registrada.
- Elimina registros sin correspondencia (casas deshabitadas o personas sin hogar asignado).
- Es útil cuando se requiere consistencia estricta entre módulos.

full_join()

Combina **todas las observaciones** de ambas tablas.

Si una llave no tiene correspondencia, las variables faltantes se completan con `NA`.

```
nrow(datos_persona)
```

```
[1] 15542
```

```
base_full <- datos_hogar %>%
  full_join(datos_persona, by = "id_hogar")
nrow(base_full)
```

```
[1] 16499
```

- Equivale a una unión externa completa en SQL.
- Es la más informativa para explorar **inconsistencias y duplicaciones**.

Diagnóstico con `full_join()`

```r
base_full %>%
  mutate(origen = case_when(
    !is.na(ingreso_hh) & !is.na(ingreso) ~ "Ambas",
    !is.na(ingreso_hh) &  is.na(ingreso) ~ "Solo hogares",
     is.na(ingreso_hh) & !is.na(ingreso) ~ "Solo personas"
  )) %>%
  count(origen)
```

origen	n
Ambas	9399
Solo hogares	141
Solo personas	6959

Permite identificar la *cantidad y tipo de registros no emparejados*. Esto es clave en depuración de microdatos.

semi_join(): Filtro positivo

Retiene solo los registros de la tabla izquierda (datos_hogar) que tienen correspondencia en la derecha (datos_persona).

```
nrow(datos_persona)
```

```
[1] 15542
```

```
hogares_con_personas <- datos_hogar %>%
  semi_join(datos_persona, by = "id_hogar")
nrow(hogares_con_personas)
```

```
[1] 3405
```

- Mantiene las variables originales de hogares.
- No añade columnas de personas.
- Ideal para filtrar subconjuntos consistentes.

anti_join(): Filtro negativo

Extrae los registros de hogares que **no tienen correspondencia** en personas.

```
nrow(datos_persona)
```

```
[1] 15542
```

```
hogares_sin_personas <- datos_hogar %>%
  anti_join(datos_persona, by = "id_hogar")
nrow(hogares_sin_personas)
```

```
[1] 141
```

Sirve para detectar:

- Hogares vacíos o sin módulo individual.
- Errores de codificación en id_hogar.
- Registros perdidos en bases complementarias.

Validación cruzada con `semi` y `anti`

```
nrow(hogares_con_personas) + nrow(hogares_sin_personas)
```

```
[1] 3546
```

```
nrow(datos_hogar)
```

```
[1] 3546
```

La suma debe coincidir con el total original, garantizando que los filtros son complementarios.

Ejemplo de control de calidad

```
hogares_sin_personas %>%
  select(id_hogar, area,  pobreza, ingreso_hh, gasto_hh) %>%
  head()
```

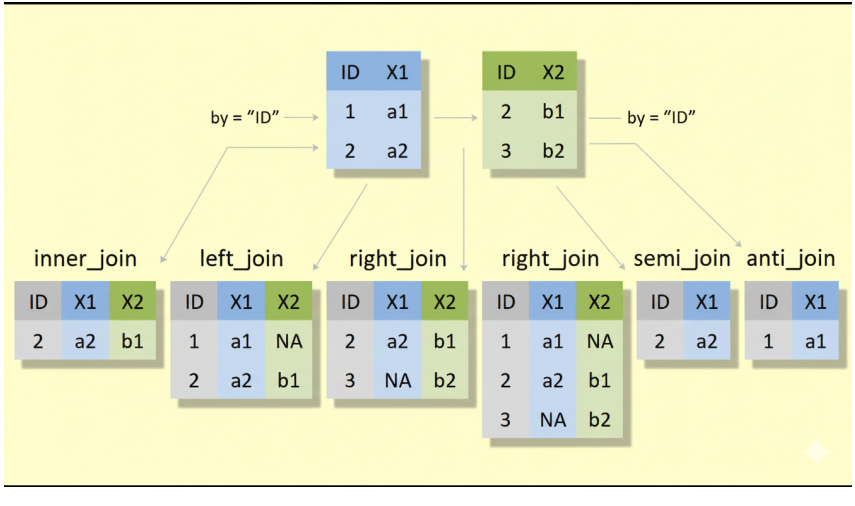
id_hogar	area	pobreza	ingreso_hh	gasto_hh
49324	1	2	2962.273	2962.273
47067	1	3	2455.370	2455.370
32260	1	3	1776.473	1776.473
16989	2	3	1695.780	1695.780
55829	1	3	2208.293	2208.293
33707	2	3	2853.133	2853.133

Estos registros deberían revisarse antes de estimar totales ponderados o tasas de cobertura.

Resumen de la familia `join`

Función	Tipo de unión	Foco
`left_join()`	Unión externa izquierda	Base principal
`right_join()`	Unión externa derecha	Población secundaria
`inner_join()`	Intersección	Consistencia
`full_join()`	Unión total	Exploración
`semi_join()`	Filtro positivo	Subconjunto existente
`anti_join()`	Filtro negativo	Subconjunto ausente

Resumen de la familia `join`



Buenas prácticas

- Verificar que la llave (`by`) sea única en la base principal.
- Evitar duplicados que generen multiplicación de filas.
- Usar `distinct()` antes de unir si hay riesgo de registros repetidos.

Aplicación práctica en encuestas

En estudios de hogares, los `join` se utilizan para:

- Vincular módulos de vivienda, hogar y persona.
- Asociar variables de ponderación o estratificación (`fep`, `upm`).
- Enlazar bases de gasto o ingreso individual al hogar de referencia.

Conclusión

- El dominio de la familia `join` del tidyverse es esencial para garantizar integridad y coherencia en los procesos de integración de datos.
- Comprender qué registros se conservan o excluyen según el tipo de unión es clave para evitar sesgos y duplicaciones en los análisis.