

# Módulo 3 — Tidyverse II

## `pivot_longer()` y `pivot_wider()`

CEPAL - Unidad de Estadísticas Sociales

2025-11-05

# Introducción

En esta sección exploraremos cómo transformar la estructura de los datos entre formato **ancho (wide)** y **largo (long)** usando la librería `tidyverse`.

Estas transformaciones son esenciales para:

- ▶ Preparar bases para análisis estadístico o modelado.
- ▶ Ajustar la forma de los datos según los requerimientos de funciones o gráficos.
- ▶ Estandarizar estructuras provenientes de diferentes fuentes.

## Base de ejemplo

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

- datos\_persona: información individual.

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

## 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 = .50)

datos_hogar_t1 <- datos_hogar %>% sample_frac(size = .30) %>%
  mutate(tiempo = "t1" )

datos_hogar_t2 <- datos_hogar %>% sample_frac(size = .3) %>%
  mutate(tiempo = "t2" )

datos_hogar <- bind_rows(datos_hogar_t1, datos_hogar_t2)
```

## Contexto: formato largo y ancho

- ▶ **Formato ancho (wide)**: cada fila representa una unidad (hogar o persona) con múltiples variables como columnas.
- ▶ **Formato largo (long)**: cada fila representa una observación de variable–valor.
- ▶ La función principal utilizada será `pivot_longer()` y `pivot_wider()` del paquete `tidyverse`.

Estas transformaciones permiten manipular bases para análisis descriptivos o gráficos.

## pivot\_wider()

Convierte filas en columnas, expandiendo los valores de una variable.

```
pivot_wider(  
  data,  
  names_from,    # Variable cuyos valores se convierten en nombres de columnas  
  values_from,   # Variable cuyos valores se llenan en las nuevas columnas  
  values_fill = NULL,  # Valor por defecto si faltan datos  
  names_prefix = NULL  # Prefijo opcional para los nuevos nombres  
)
```

## Ejemplo

```
tabla_ancha <- datos_hogar %>%
  pivot_wider(names_from = tiempo,
              values_from = ingreso_hh)
tabla_ancha %>% head(5)
```

id_hogar	gasto_hh	pobreza	area	t1	t2
43307	5000.000	3	1	5000.000	5000.00
30063	5062.500	3	2	5062.500	5062.50
17291	11482.633	2	1	11482.633	11482.63
21875	7433.344	2	1	7433.344	NA
3965	19886.107	3	1	19886.107	NA

## Funciones similares:

- ▶ `spread()` (*deprecated*) — versión anterior de `pivot_wider()`.
- ▶ `dcast()` del paquete **reshape2** — más control para operaciones agregadas.
- ▶ `acast()` — igual que `dcast()`, pero devuelve un arreglo en lugar de un data frame.

## `pivot_longer()`

Convierte columnas en filas, condensando la estructura de los datos.

```
pivot_longer(  
  data,  
  cols,      # Columnas que se combinarán  
  names_to,  # Nombre de la variable que almacenará los nombres originales  
  values_to, # Nombre de la variable que almacenará los valores  
  names_prefix = NULL,  
  values_drop_na = FALSE  
)
```

## Ejemplo

```
tabla_larga <- tabla_ancha %>%
  pivot_longer(
    cols = starts_with("t"),
    names_to = "tiempo",
    values_to = "ingreso_hh"
  )
head(tabla_larga)
```

id_hogar	gasto_hh	pobreza	area	tiempo	ingreso_hh
43307	5000.00	3	1	t1	5000.00
43307	5000.00	3	1	t2	5000.00
30063	5062.50	3	2	t1	5062.50
30063	5062.50	3	2	t2	5062.50
17291	11482.63	2	1	t1	11482.63
17291	11482.63	2	1	t2	11482.63

## Funciones similares:

- ▶ `gather()` (*deprecated*) — versión anterior de `pivot_longer()`.
- ▶ `melt()` del paquete **reshape2** — útil para marcos de datos y matrices.
- ▶ `stack()` de base R — para estructuras más simples, sin necesidad de tidyverse.

## Ejemplo de formato ancho

Visualicemos el hogar con varias columnas de gasto y educación:

```
head(datos_persona)
```

id_hogar	id_pers	edad	sexo	etnia	anoest	ingreso	gasto
56557	1	62	Hombre	0	12	1895.363	1895.363
3646	2	16	Mujer	0	10	5586.273	5586.273
23913	2	27	Mujer	0	7	1013.027	1013.027
9857	1	53	Hombre	0	12	2151.003	2151.003
57119	4	33	Hombre	0	11	3358.248	3358.248
10121	1	90	Mujer	0	6	2036.462	2036.462

## Conversión a formato largo

Usamos pivot\_longer() para transformar columnas numéricas a formato largo.

```
personas_largo <- datos_persona %>%
  pivot_longer(cols = c(ingreso, gasto),
               names_to = "variable",
               values_to = "valor")
```

## Visualización del formato largo

Cada fila ahora representa un **par (variable, valor)**, lo que facilita cálculos o gráficos por tipo de variable.

```
head(personas_largo, 8)
```

id_hogar	id_pers	edad	sexo	etnia	anoest	variable	valor
56557	1	62	Hombre	0	12	ingreso	1895.363
56557	1	62	Hombre	0	12	gasto	1895.363
3646	2	16	Mujer	0	10	ingreso	5586.273
3646	2	16	Mujer	0	10	gasto	5586.273
23913	2	27	Mujer	0	7	ingreso	1013.027
23913	2	27	Mujer	0	7	gasto	1013.027
9857	1	53	Hombre	0	12	ingreso	2151.003
9857	1	53	Hombre	0	12	gasto	2151.003

## Aplicación práctica: promedio por tipo de variable

```
personas_largo %>%
  group_by(variable) %>%
  summarise(media = mean(valor, na.rm = TRUE))
```

variable	media
gasto	3328.860
ingreso	3435.624

## Transformación inversa: formato ancho

Si se desea volver al formato anterior, se utiliza `pivot_wider()`.

```
personas_ancho2 <- personas_largo %>%
  mutate(id_pers = paste0(id hogar, id_pers)) %>%
  group_by(id_pers, variable) %>%
  summarise(valor = mean(valor), .groups = "drop") %>%
  pivot_wider(names_from = variable, values_from = valor)
```

## Verificación del resultado

```
glimpse(personas_ancho2)
```

Rows: 15,051

Columns: 3

```
$ id_pers <chr> "100332", "100391", "100392", "100393", "100394", "100411"  
$ gasto    <dbl> 1567.647, 1712.622, 1712.622, 1741.095, 1712.622, 1813.960  
$ ingreso   <dbl> 1567.647, 1712.622, 1712.622, 1741.095, 1712.622, 1813.960
```

```
head(personas_ancho2)
```

	id_pers	gasto	ingreso
1	100332	1567.647	1567.647
2	100391	1712.622	1712.622
3	100392	1712.622	1712.622
4	100393	1741.095	1741.095
5	100394	1712.622	1712.622
6	100411	1813.960	1813.960

## Transformación inversa: formato ancho

Si se desea volver al formato anterior, se utiliza `pivot_wider()`.

```
personas_ancho3 <- personas_largo %>%
  mutate(id_pers = paste0(id_hogar, id_pers)) %>%
  pivot_wider(names_from = variable,
              values_from = valor,
              values_fn = mean)
```

## Verificación del resultado

```
head(personas_ancho3)
```

id_hogar	id_pers	edad	sexo	etnia	anoest	ingreso	gasto
56557	565571	62	Hombre	0	12	1895.363	1895.363
3646	36462	16	Mujer	0	10	5586.273	5586.273
23913	239132	27	Mujer	0	7	1013.027	1013.027
9857	98571	53	Hombre	0	12	2151.003	2151.003
57119	571194	33	Hombre	0	11	3358.248	3358.248
10121	101211	90	Mujer	0	6	2036.462	2036.462

## Ejemplo adicional: categorización por sexo

Podemos analizar la estructura de gasto e ingreso según el sexo:

```
personas_largo %>%
  group_by(sexo, variable) %>%
  summarise(media = mean(valor, na.rm = TRUE), .groups = "drop") %>%
  pivot_wider(names_from = variable, values_from = media)
```

sexo	gasto	ingreso
Hombre	3419.658	3546.490
Mujer	3243.481	3331.375

## Recomendaciones finales

- ▶ `pivot_longer()` facilita la estandarización para análisis gráficos.
- ▶ `pivot_wider()` es útil para crear matrices o reportes tabulares.
- ▶ Siempre verificar la cantidad de filas antes y después de cada transformación.
- ▶ Estas funciones reemplazan el antiguo `gather()` y `spread()` de `tidyR`.