

## Módulo 2 — Tidyverse I

`select()` y `filter()`

CEPAL - Unidad de Estadísticas Sociales

Invalid Date

3. mutate() y transmute()

# Objetivo

Permiten **crear o transformar columnas** dentro de un `data.frame` o `tibble`.

# mutate()

- ▶ Agrega **nuevas variables** o **modifica existentes**.
- ▶ Mantiene todas las columnas originales.
- ▶ Puede usar variables recién creadas dentro del mismo mutate().

```
library(dplyr)

mtcars %>%
  mutate(
    consumo_km = mpg * 0.425,          # de millas/galón a km/litro
    potencia_relativa = hp / max(hp)  # proporción de potencia
  )
```

*Permite encadenar transformaciones sin perder las variables originales.*

## transmute()

- ▶ Similar a mutate(), pero **solo conserva las variables creadas**.
- ▶ Útil para construir indicadores o reducir la salida.

```
mtcars %>%  
  transmute(  
    consumo_km = mpg * 0.425,  
    potencia_relativa = hp / max(hp)  
  )
```

*Devuelve un tibble únicamente con las nuevas variables.*

## Uso combinado con funciones auxiliares

- ▶ `if_else()` → creación condicional
- ▶ `case_when()` → múltiples condiciones
- ▶ `across()` → operaciones sobre grupos de columnas

Ejemplo:

```
mtcars %>%  
  mutate(  
    tipo = if_else(cyl == 4, "Eficiente", "Potente"),  
    potencia_log = log(hp)  
  )
```

## Ejemplo práctico

**Objetivo:** Crear un indicador estandarizado de eficiencia.

```
mtcars %>%  
  mutate(  
    consumo_km = mpg * 0.425,  
    eficiencia = consumo_km / mean(consumo_km)  
  ) %>%  
  select(model = rownames(mtcars), eficiencia)
```

Resultado: tibble con índice de eficiencia relativo al promedio del conjunto.

## Comparación resumida

Función	Conserva columnas originales	Crea nuevas variables	Ejemplo típico
<code>mutate()</code>	Sí	Sí	Añadir una columna calculada
<code>transmute()</code>	No	Sí	Crear indicadores derivados



## Recomendaciones de uso

- ▶ Usa `mutate()` para **procesos exploratorios o pasos intermedios**.
- ▶ Usa `transmute()` para **resultados finales o salidas resumidas**.
- ▶ Combina con `across()` para aplicar transformaciones en bloque.

```
mtcars %>%  
  mutate(across(starts_with("d"), scale))
```

# Conclusión

Ambas funciones son **pilares en el flujo de trabajo del tidyverse**. Facilitan la transformación progresiva y reproducible de datos, alineando la sintaxis con la lógica de la manipulación funcional en R.

“ “