Clase 3b - Visualización de resultados

José Rodríguez de la Fuente

Tabla de contenidos

Graficando con R

Una de las razones por las que muchas personas optan por utilizar lenguajes como `R` o `Python` para el análisis de datos es la posibilidad de generar gráficos de alta calidad de manera sencilla. Particularmente, `R` dispone de una variada cantidad de librerías para la generación de información gráfica que es digna de explorar. Cada una tiene su lógica y gramática propia.

Una **gramática de gráficos** es un conjunto de reglas y principios que definen cómo construir e interpretar gráficos. Estas gramáticas permiten generar visualizaciones de datos de manera sistemática y flexible. En `R`, la librería `ggplot2` [@wickham2010] es la más utilizada para estos fines. La misma forma parte del paquete `tidyverse` y es una implementación de la gramática de gráficos de Wilkinson [@wilkinson2005].

La idea de tener como fundamento una gramática es que iremos componiendo nuestros gráficos a partir de elementos independientes entre sí, sin tener que limitarnos a un número específico de gráficos predefinidos. `ggplot2` funciona en forma iterativa, es decir, iremos agregando capas de información a un gráfico básico, partiendo desde los datos crudos, hasta llegar a agregar geometrías y anotaciones. Veremos cómo esta forma de trabajo reducirá la distancia entre lo que pensamos y lo que finalmente queda en la visualización. De lo que se trata es de armar un gráfico sin hacer un solo *point and click*, ni depender de nuestro pulso de artista.

Figura 1: Concepto de gramática de gráficos de Wilkinson

Como se observa en esquema anterior, partiendo desde los datos con lo que contamos, tenemos seis capas posibles para superponer y editar, dando forma a nuestro gráfico:

1. **Datos**: los datos que queremos visualizar.
2. **Estética**: cómo se mapean los datos a los elementos visuales. Qué ira al eje x y al eje y, que elementos diferenciaremos por color, forma, tamaño, etc.

3. **Elementos geométricos**: qué tipo de geometrías utilizaremos. Pueden ser líneas, puntos, barras, polígonos, etc. O pueden ser combinaciones más complejas: *boxplot*, histogramas, mapas de calor, etc.

4. **Facetados**: cómo dividir los datos en subgráficos según alguna variable de interés.
5. **Estadísticas**: cómo resumir los datos antes de graficarlos. Por ejemplo, contar los casos, ajustarlos a un modelo lineal, etc.

6. **Coordenadas**: sistema de coordenadas utilizado.
7. **Temas**: aspecto visual del gráfico. Fuentes, tamaño de letra, títulos, subtítulos, etc. Hay temas predefinidos que ya vienen en la librería.

Con esta breve introducción, estamos en condiciones de construir nuestro primer gráfico en `R`.

Recursos a explorar

La web está llena de ejemplos e ideas que nos pueden inspirar para construir nuestros gráficos. Una buena página para explorar es R Graph Gallery. Allí encontraremos una gran cantidad de ejemplos de gráficos realizados con `R`, con el código fuente para replicarlos. Además, propone una buena clasificación de los gráficos según su tipo, lo que nos puede ayudar a encontrar rápidamente el tipo de gráfico que necesitamos.

Particularmente sobre `ggplot2`, sería recomendable que consulten el siguiente material:

- R para Ciencia de Datos: en el capítulo 3 del libro encontrarán una breve guía para trabajar con `ggplot2`.
- ggplot2: Elegant Graphics for Data Analysis: este es el libro oficial de `ggplot2`, escrito por Hadley Wickham, su creador. En el sitio encontrarán una versión online del libro, con ejemplos y explicaciones detalladas.
- Machete de ggplot2: una guía rápida con los principales comandos de `ggplot2`.

Otra fuente similar a la anterior es Data to Viz, donde también encontraremos una guía para elegir el tipo de gráfico adecuado según los datos que tengamos.

Por otro lado, existen dos librerías que no exploraremos en el curso pero que son de gran interés:

- `plotly`: permite generar gráficos interactivos. https://plotly.com/r/
- `gganimate`: permite generar gráficos animados. https://gganimate.com/

Librerías y bases que utilizaremos

Vamos a instalar estas librerías:

```
install.packages("scales")
```

Vamos a activar las siguientes librerías:

```
library(tidyverse)
library(scales)
library(eph)
```

Vamos a cargar las bases de datos que utilizaremos:

```r
eph_ind <- get_microdata(year = 2025, trimester = 2, type = "individual")

eph_hog <- get_microdata(year = 2022:2025, trimester = 2, type = "hogar")
```

Para la construcción de los gráficos vamos a transformar y crear algunas variables siguiendo los procedimientos aprendidos hasta ahora. En el caso de la base de hogares vamos a utilizar las variables de *estrategias del hogar*, específicamente en la sección de gastos. Vamos a otorgar valor **1** cuando se utilizó el activo y **0** cuando no se lo utilizó.

```r
eph_ind <- eph_ind %>%
  mutate(sexo = case_when(
    CH04 == 1 ~ "Varón",
    CH04 == 2 ~ "Mujer"))


eph_hog <- eph_hog %>%
  mutate(ahorros = case_when(V13 == 1 ~ 1,
                                    TRUE ~ 0),
         prestamo_familiar = case_when(V14 == 1 ~ 1,
                                       TRUE ~ 0),
         prestamo_bancario = case_when(V15 == 1 ~ 1,
                                      TRUE ~ 0),
         comprar_cuotas = case_when(V16 == 1 ~ 1,
                                    TRUE ~ 0),
         vender_bienes = case_when(V17 == 1 ~ 1,
                                   TRUE ~ 0))
```

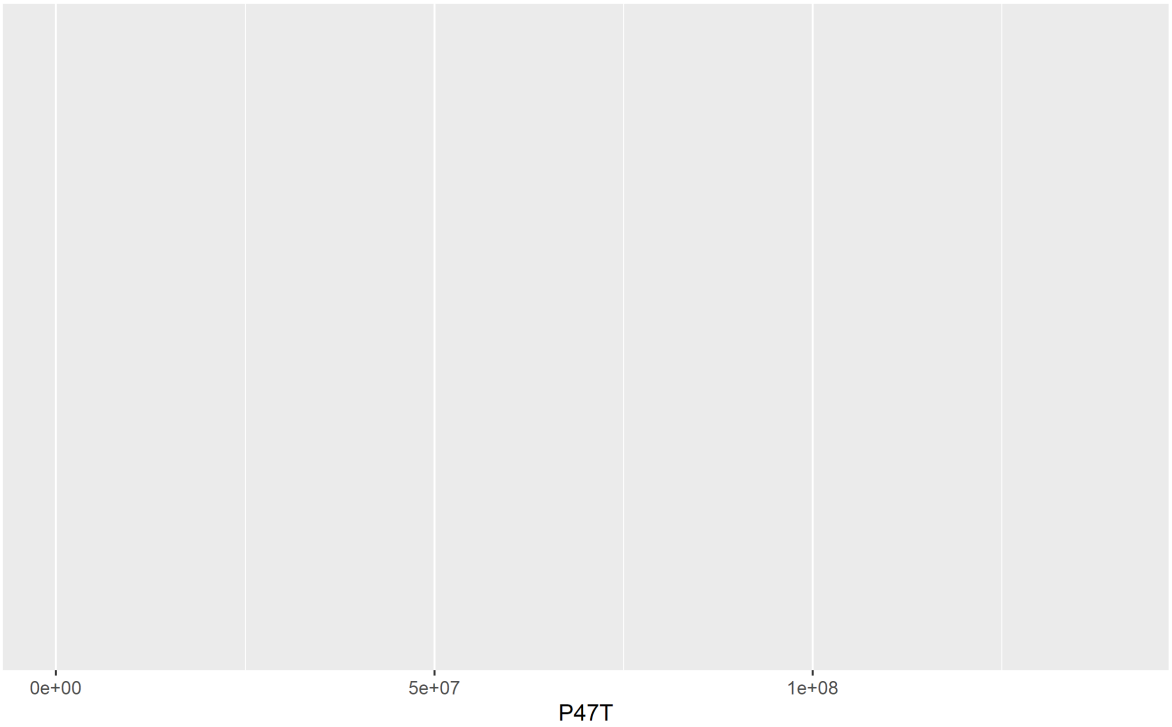
Construyendo el primer gráfico (paso a paso)

Vamos a comenzar construyendo un gráfico sencillo, pero lo haremos paso a paso, para que luego podamos ir agregando complejidad en su composición. Vamos a trabajar sobre la base de individuos. Tomaremos como variable de interés el ingreso total individual (`P47T`). El primer paso será llamar a la función `ggplot()` y pasarle como argumento la base de datos sobre la que queremos graficar.

```r
ggplot(data = eph_ind)
```

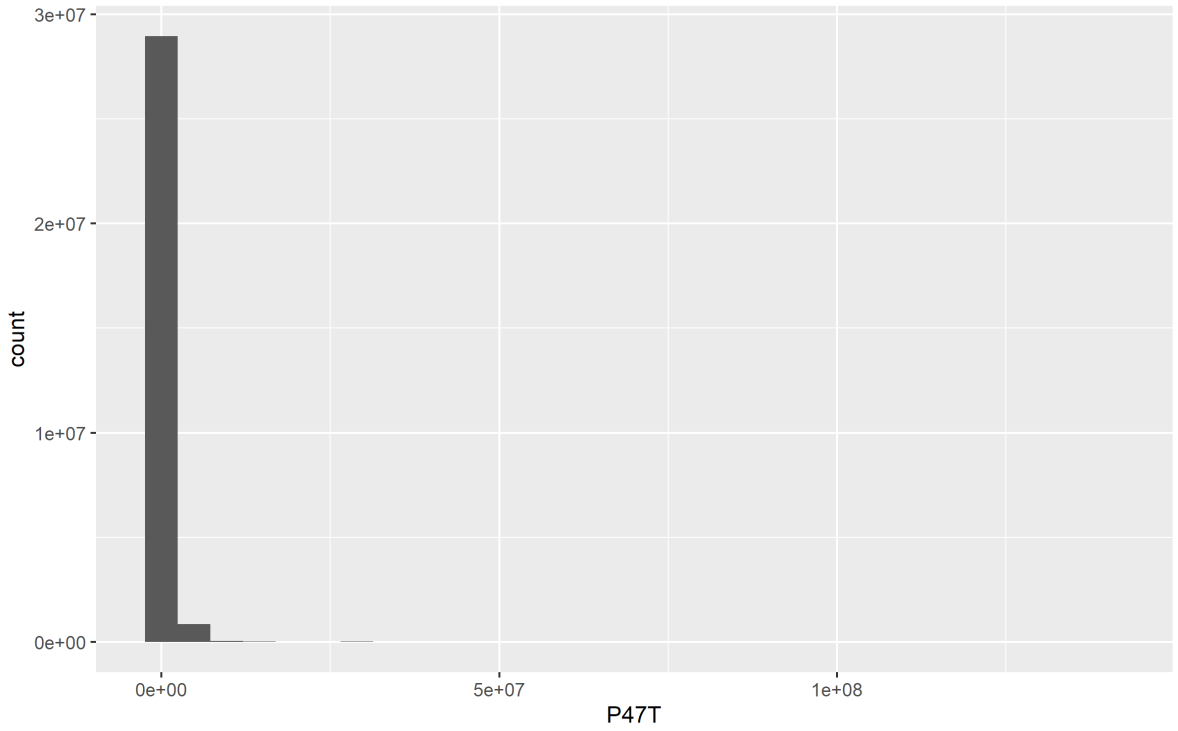
Esto nos devolverá como resultado, una hoja en blanco, ya que no le hemos indicado qué queremos graficar. Entonces, el siguiente paso será indicarle qué variable queremos graficar. Para ello, utilizaremos la función `aes()`, que nos permite *mapear* las variables de la base de datos a los elementos visuales del gráfico. En este caso, le pasaremos a `aes()` la variable `P47T` y el ponderador `PONDII`.

```
ggplot(data = eph_ind, aes(x = P47T, weights = PONDII))
```



Ahora el gráfico se va llenando. Ha aparecido la escala del eje x referida a los ingresos de las personas (en notación científica). Pero aún no hemos indicado qué queremos hacer con esa variable. Para ello, vamos a agregar una capa geométrica, en este caso, le pediremos que grafique un histograma.

```
ggplot(data = eph_ind, aes(x = P47T, weights = PONDII)) +
  geom_histogram()
```

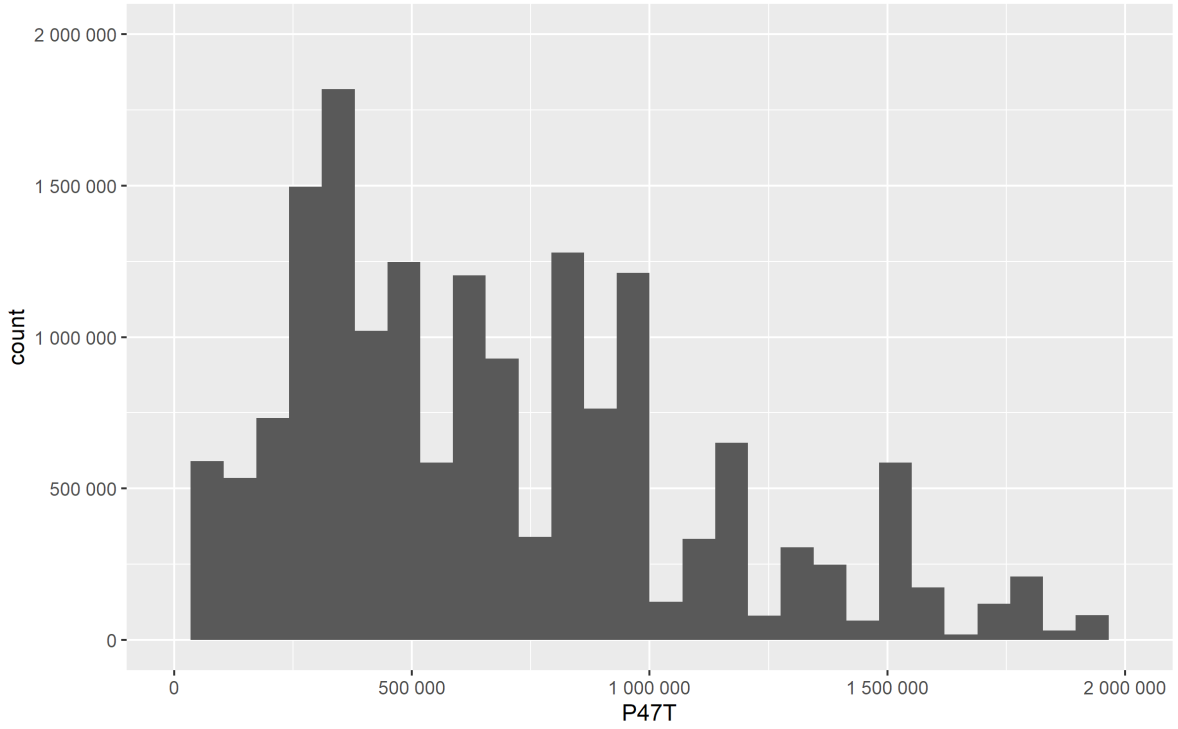


> **Atención**
>
> Fijense que de forma similar al `%>%` de `dplyr`, en `ggplot2` utilizamos el `+` para ir agregando capas al gráfico. Es común que nos confundamos entre ambos operadores, pero con la práctica, iremos adquiriendo la costumbre de utilizarlos correctamente.

Para que veamos mejor el gráfico vamos a seleccionar a los ingresos mayores de 0 y menores a 2.000.000 (percentil 95) y vamos a transformar la escala de notación científica a numérica en ambos ejes. Para ello, configuraremos los paramétros de escala: `scale_x_continuous` y `scale_y_continuous`. Dentro de estas funciones, podemos indicar los intervalos (*breaks*) y los límites (*limits*) que queremos para cada eje.

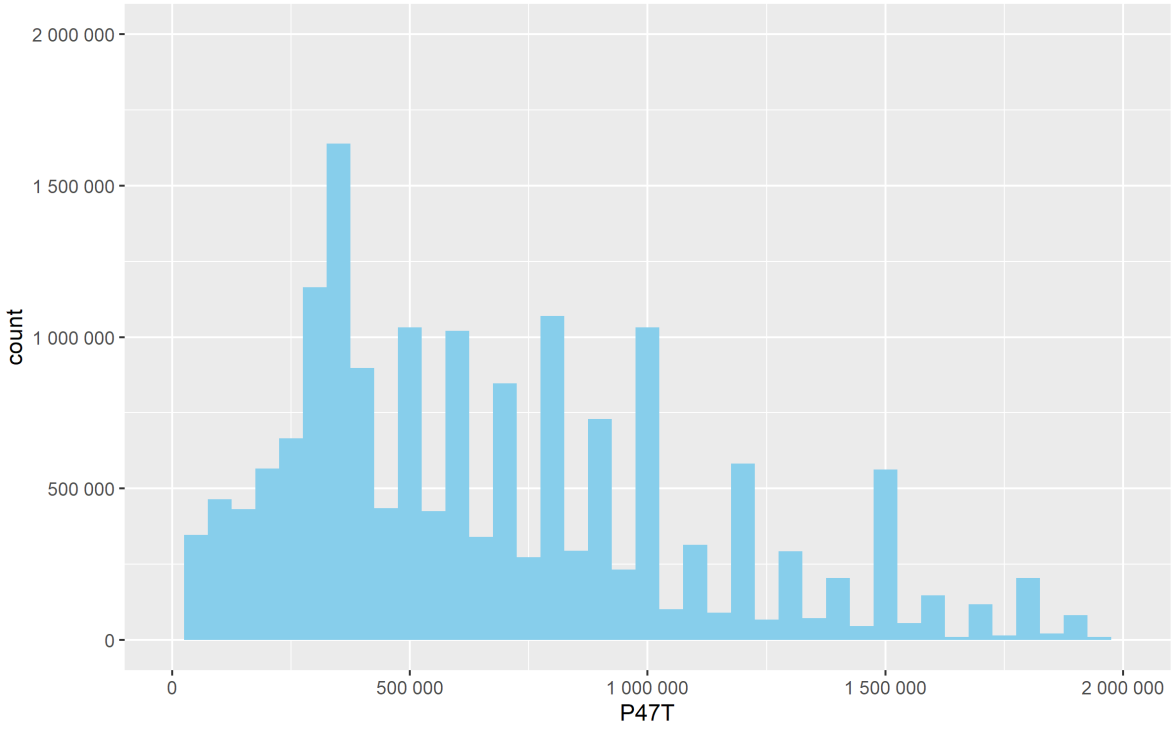
La opción `limits = c(0, 2000000)` indica que los límites del eje x e y serán de 0 a 2000000. También podemos modificar los puntos de corte, pero en este caso no hará falta.

```
ggplot(data = eph_ind, aes(x = P47T, weights = PONDII)) +
  geom_histogram() +
  scale_x_continuous(labels = label_number(), limits = c(0,2000000)) +
  scale_y_continuous(labels = label_number(), limits = c(0,2000000))
```



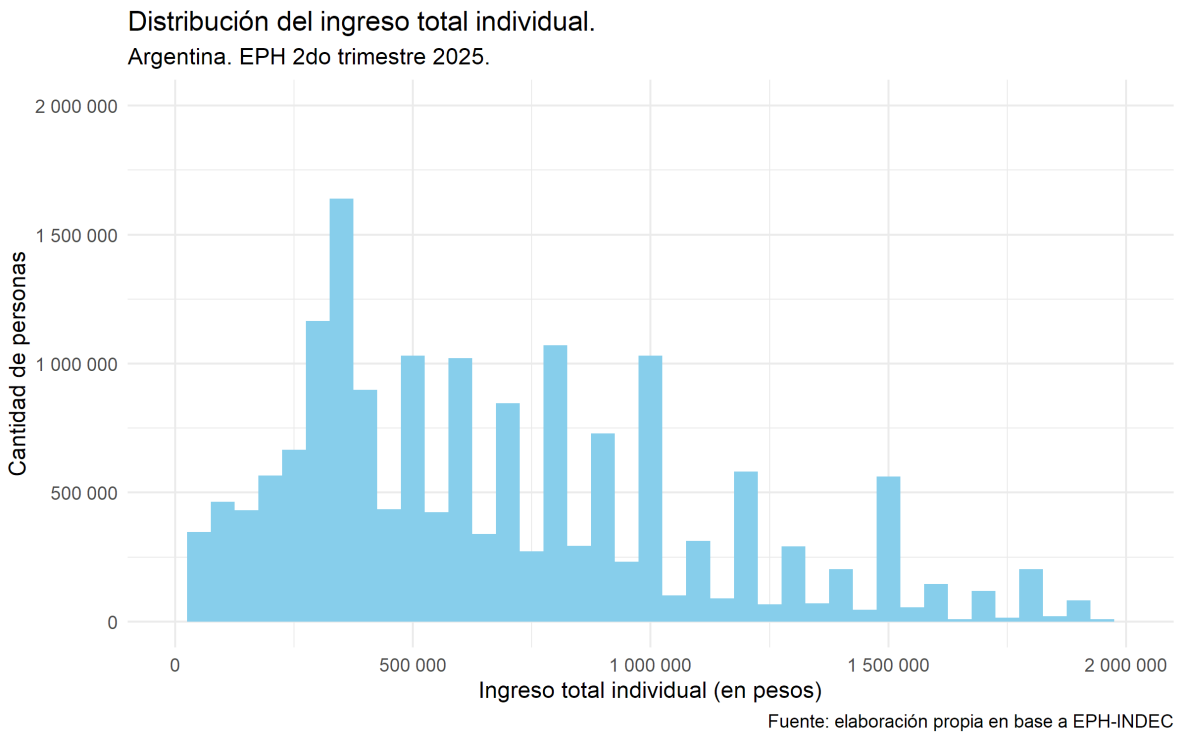
Dentro de la capa de geometría, podemos agregar argumentos que nos permitan personalizar el gráfico. Por ejemplo, cambiemos el color de las barras mediante la opción `fill` y el tamaño mediante la función `binwidth`. Pueden explorar las distintas formas de identificar las colores en este sitio.

```
ggplot(data = eph_ind, aes(x = P47T, weight = PONDII)) +
  geom_histogram(fill = "skyblue", binwidth = 50000) +
  scale_x_continuous(labels = label_number(), limits = c(0,2000000)) +
  scale_y_continuous(labels = label_number(), limits = c(0,2000000))
```



Ahora bien, para cerrar nuestro primer gráfico, vamos a agregarle un título y etiquetas a los ejes. Para ello, utilizaremos la función `labs()`. A su vez, podemos también modificar el tema visual del gráfico mediante la función `theme()`. Seleccionaremos el tema *minimal* mediante la capa `theme_minimal()`.

```
ggplot(data = eph_ind, aes(x = P47T, weight = PONDII)) +
  geom_histogram(fill = "skyblue", binwidth = 50000) +
  scale_x_continuous(labels = label_number(), limits = c(0,2000000)) +
  scale_y_continuous(labels = label_number(), limits = c(0,2000000)) +
  labs(title = "Distribución del ingreso total individual.",
       subtitle = "Argentina. EPH 2do trimestre 2025.",
       caption = "Fuente: elaboración propia en base a EPH-INDEC",
       x = "Ingreso total individual (en pesos)",
       y = "Cantidad de personas") +
  theme_minimal()
```



Si queremos exportar el gráfico a un archivo, para luego utilizarlo en algún documento o compartirlo, podemos hacerlo mediante la función `ggsave()`. Esta función nos permite guardar el gráfico en distintos formatos, como `pdf`, `png`, `jpeg`, `tiff`, entre otros.

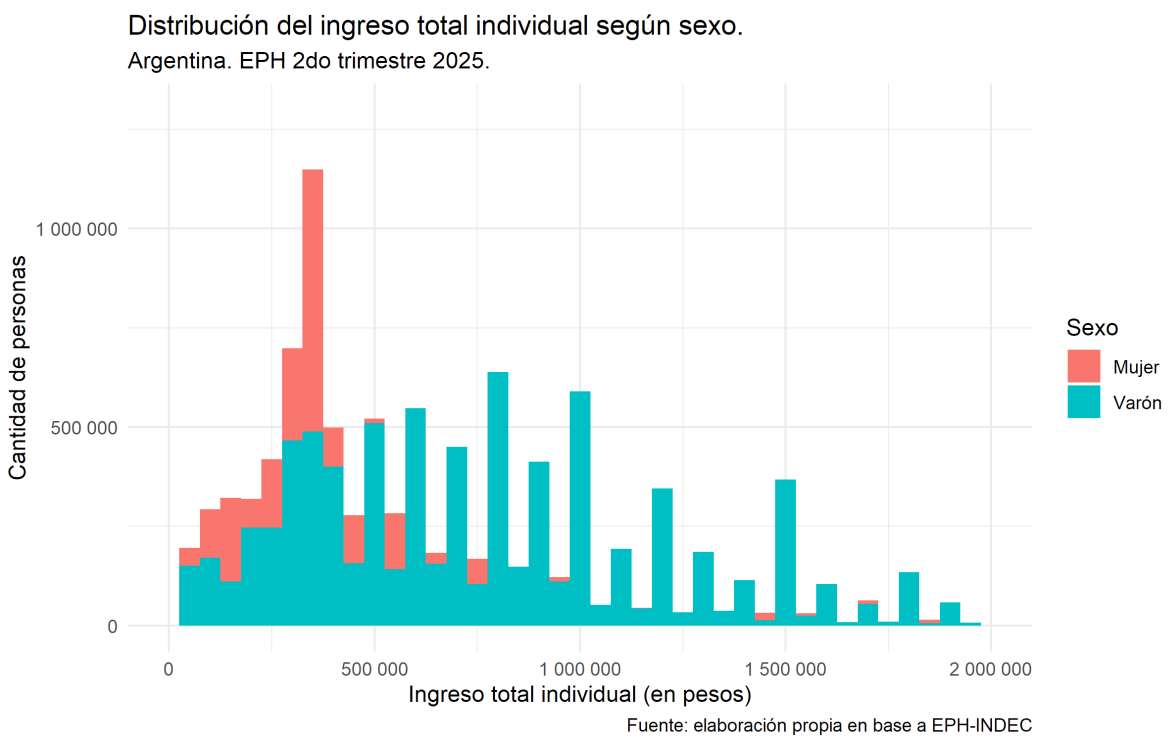
```
ggsave("resultados/grafico_ingresos.jpeg", width = 8, height = 5, dpi = 300)
```

Observando diferencias por grupos

Siguiendo con el gráfico que fuimos contruyendo, vamos a ver como hacer para comparar la distribución de ingresos según sexo. Para ello, vamos a utilizar la variable `sexo` que construímos anteriormente. El principal cambio que vamos a generar en nuestras capas es que en la opción *fill* de la capa `aes()` vamos a indicar que queremos diferenciar por rama.

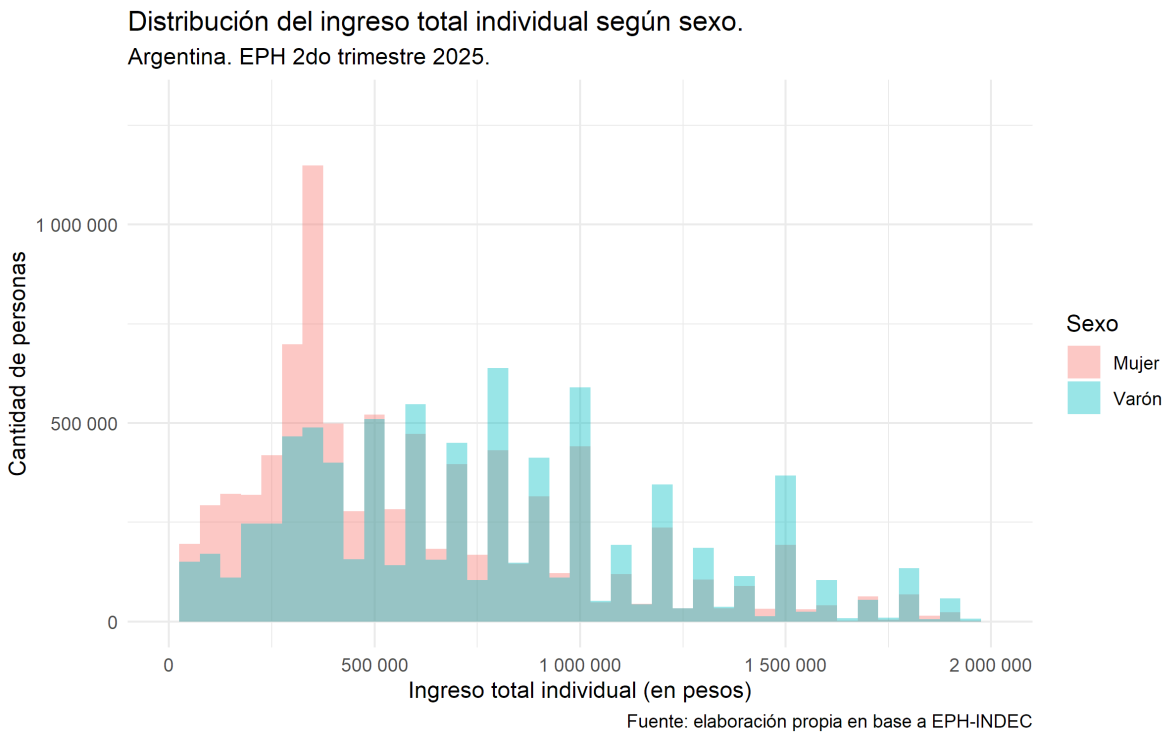
```
ggplot(data = eph_ind, aes(x = P47T, weight = PONDII, fill = sexo)) +
  geom_histogram(binwidth = 50000, position = "identity") +
  scale_x_continuous(labels = label_number(), limits = c(0,2000000)) +
  scale_y_continuous(labels = label_number(), limits = c(0,1300000)) +
  labs(title = "Distribución del ingreso total individual según sexo.",
       subtitle = "Argentina. EPH 2do trimestre 2025.",
       caption = "Fuente: elaboración propia en base a EPH-INDEC",
```

```
      x = "Ingreso total individual (en pesos)",
      y = "Cantidad de personas",
      fill = "Sexo") +
  theme_minimal()
```



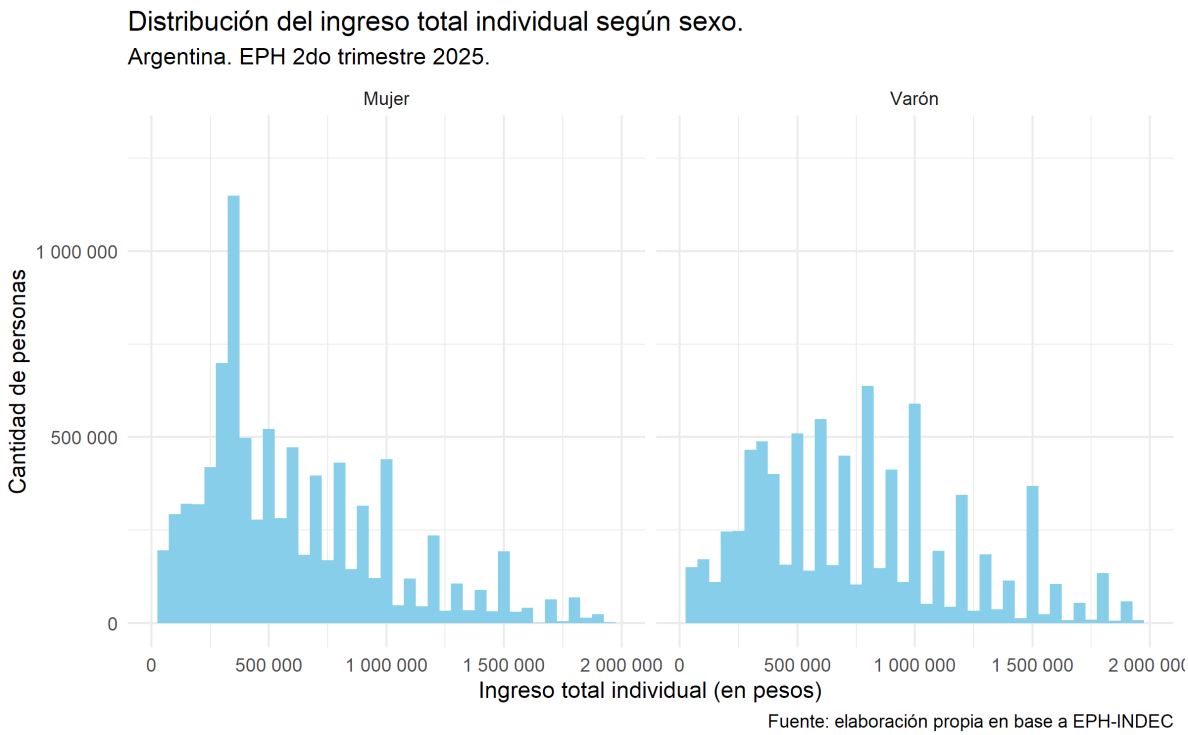
Podríamos mejorar la gráfica retocando la transparencia de las barras mediante la opción *alpha* en `geom_histogram`.

```
ggplot(data = eph_ind, aes(x = P47T, weight = PONDII, fill = sexo)) +
  geom_histogram(binwidth = 50000, position = "identity", alpha = 0.4) +
  scale_x_continuous(labels = label_number(), limits = c(0,2000000)) +
  scale_y_continuous(labels = label_number(), limits = c(0,1300000)) +
  labs(title = "Distribución del ingreso total individual según sexo.",
       subtitle = "Argentina. EPH 2do trimestre 2025.",
       caption = "Fuente: elaboración propia en base a EPH-INDEC",
       x = "Ingreso total individual (en pesos)",
       y = "Cantidad de personas",
       fill = "Sexo") +
  theme_minimal()
```



Otra forma de evaluar las diferenciaciones por grupos es mediante *facetados*. Vamos a ver cómo hacer para que el gráfico se divida en dos paneles, uno para cada sexo. Para ello, vamos a utilizar la función `facet_wrap()` y le pasaremos como argumento la variable `sexo`.

```
ggplot(data = eph_ind, aes(x = P47T, weight = PONDII)) +
  geom_histogram(binwidth = 50000, fill = "skyblue") +
  scale_x_continuous(labels = label_number(), limits = c(0,2000000)) +
  scale_y_continuous(labels = label_number(), limits = c(0,1300000)) +
  labs(title = "Distribución del ingreso total individual según sexo.",
       subtitle = "Argentina. EPH 2do trimestre 2025.",
       caption = "Fuente: elaboración propia en base a EPH-INDEC",
       x = "Ingreso total individual (en pesos)",
       y = "Cantidad de personas") +
  theme_minimal() +
  facet_wrap(~sexo)
```



Gráficos de barras

Uno de los gráficos más utilizados en ciencias sociales para observar la distribución de las variables categóricas son los gráficos de barras. Por ejemplo, vamos a representar la proporción de hogares que realizaron distintos tipos de gastos especiales como estrategia de reproducción. Utilizaremos solo los casos correspondientes al 2do trimestre de 2025.

Como vamos a querer que los valores del eje y se presenten en porcentajes, primero vamos a calcular la proporción de personas en cada categoría. Para ello, utilizaremos la función `summarise` de `dplyr`. Asignaremos la nueva tabla agregada al objeto **gastos**. Como queremos obtener los valores ponderados vamos a dividir al **ponderador** cuando el gasto sea igual a 1 sobre la suma total. Esto nos dara una proporción como resultado.

```
gastos <- eph_hog %>%
  filter(TRIMESTRE == 2, ANO4 == 2025) %>%
  summarise(ahorros = sum(PONDERA[ahorros == 1], na.rm = T) / sum(PONDERA,
    ↪ na.rm = T),
            prestamo_familiar = sum(PONDERA[prestamo_familiar == 1], na.rm =
              ↪ T) / sum(PONDERA, na.rm = T),
            prestamo_bancario = sum(PONDERA[prestamo_bancario == 1], na.rm =
              ↪ T) / sum(PONDERA, na.rm = T),
```

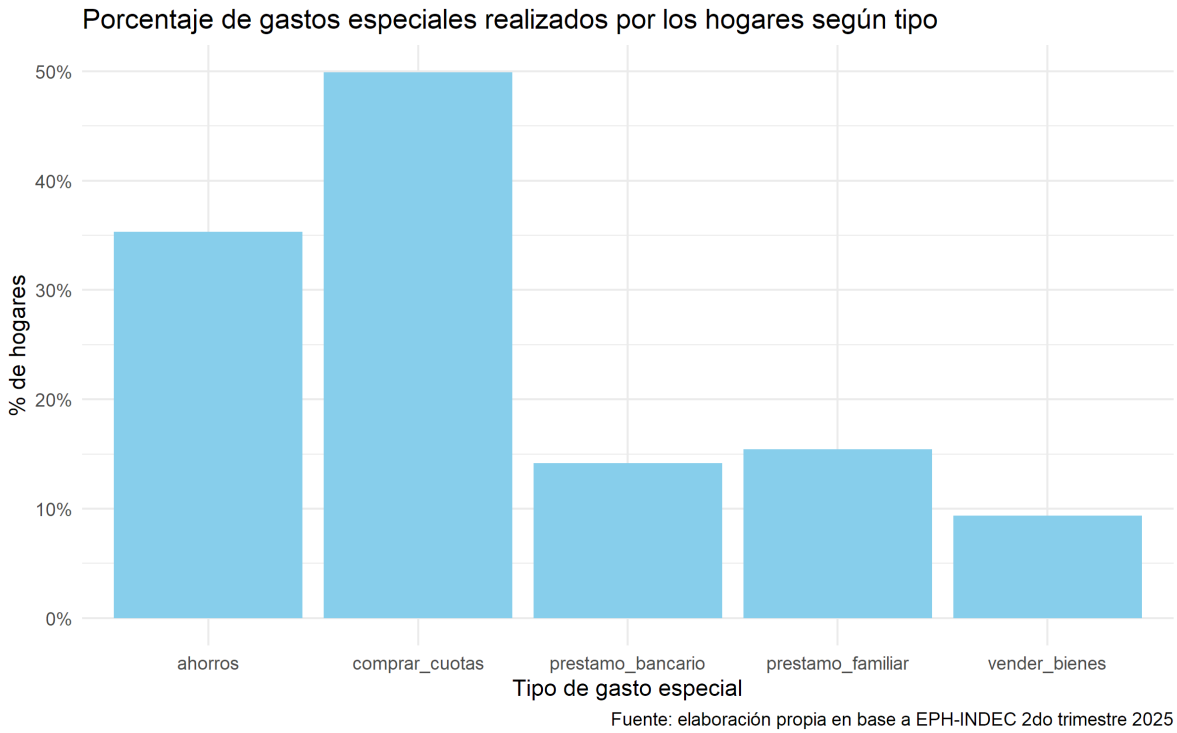
```
            comprar_cuotas = sum(PONDERA[comprar_cuotas == 1], na.rm = T) /
             ↪ sum(PONDERA, na.rm = T),
            vender_bienes = sum(PONDERA[vender_bienes == 1], na.rm = T) /
             ↪ sum(PONDERA, na.rm = T))
```

Sin embargo, para poder graficar con `ggplot2`, necesitamos que la tabla esté en formato *largo*. Por ello, vamos a utilizar la función `pivot_longer` de `tidyr`. En esta función, le pasaremos como argumento las columnas que queremos transformar a formato largo, el nombre de la nueva columna que contendrá los nombres de las categorías y el nombre de la nueva columna que contendrá los valores de proporción.

```
gastos <- gastos %>%
  pivot_longer(cols = everything(),
               names_to = "tipo_gasto",
               values_to = "prop")
```

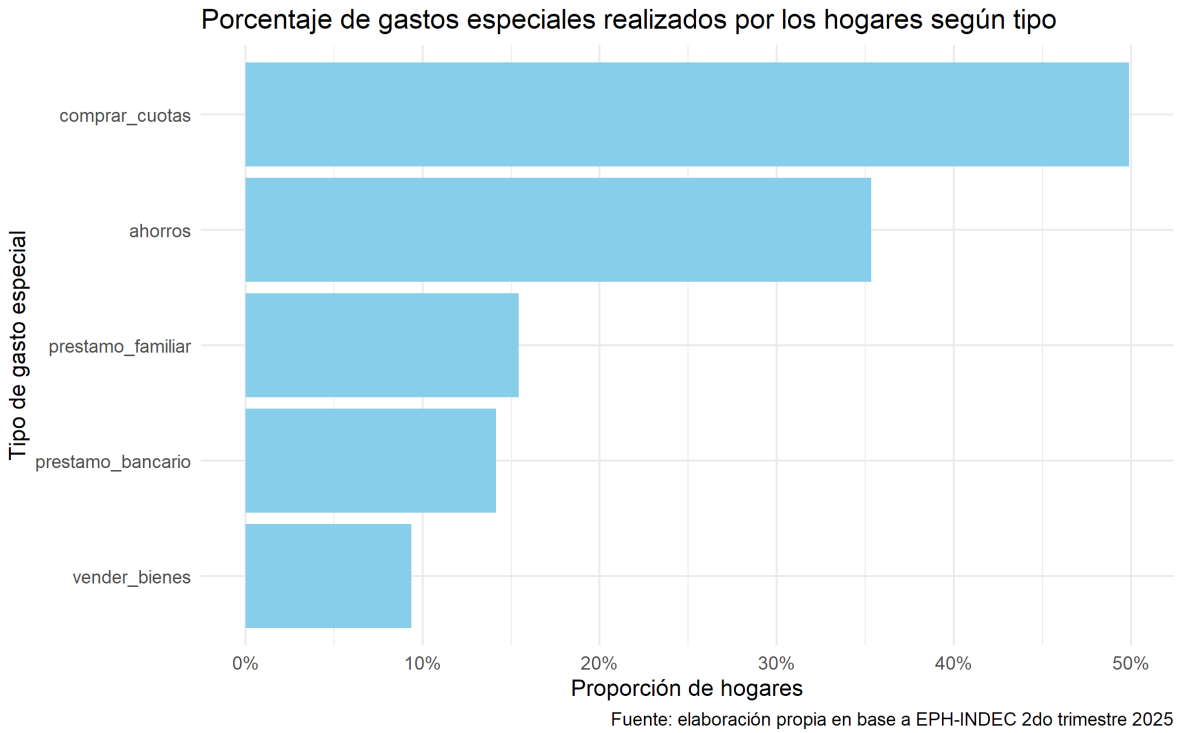
Ahora estamos en condiciones de probar la función `geom_bar` para construir un gráfico de barras. Es importante declarar `stat = "identity"` para que las barras se muestren en función de la proporción de personas en cada categoría.

```
ggplot(data = gastos, aes(x = tipo_gasto, y = prop)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Porcentaje de gastos especiales realizados por los hogares
   ↪ según tipo",
       caption = "Fuente: elaboración propia en base a EPH-INDEC 2do
        ↪ trimestre 2025",
       x = "Tipo de gasto especial",
       y = "% de hogares") +
  theme_minimal() +
  scale_y_continuous(labels = percent_format())
```



Como las etiquetas de las columnas se pueden solapar entre sí, lo mejor es invertir el eje x y el eje y mediante la función `coord_flip`. Además podemos ordenar las barras en dirección descendente con la función `reorder` en la parte de la estética.

```
ggplot(data = gastos, aes(x = reorder(tipo_gasto, prop), y = prop)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Porcentaje de gastos especiales realizados por los hogares
   ↪  según tipo",
       caption = "Fuente: elaboración propia en base a EPH-INDEC 2do
        ↪  trimestre 2025",
       x = "Tipo de gasto especial",
       y = "Proporción de hogares") +
  theme_minimal() +
  scale_y_continuous(labels = percent_format()) +
  coord_flip()
```

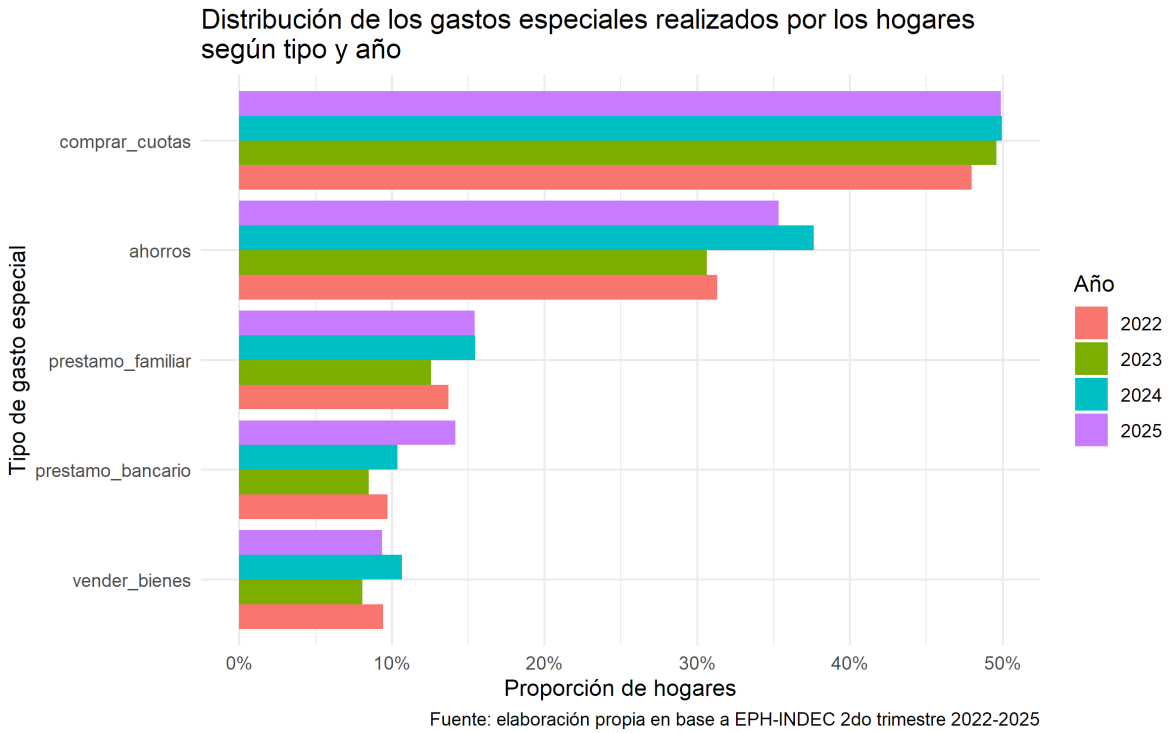


Ahora bien, existen otros formatos de gráficos de barras que nos permiten representar un cruce de variables, tal cómo lo hicimos en la clase anterior al elaborar tablas de contingencia. Por ejemplo, podríamos preguntarnos cómo evolucionaron los distintos tipos de gastos por año. Para ello, primero vamos a calcular los porcentajes por año. Luego, vamos a pasar en la capa `aes()` en la opción `fill`, la variable `ANO4`. Al mismo tiempo, en la capa `geom_bar` indicaremos la opción `position = "dodge"`.

```
gastos <- eph_hog %>%
  group_by(ANO4) %>%
  summarise(ahorros = sum(PONDERA[ahorros == 1], na.rm = T) / sum(PONDERA,
    ↪  na.rm = T),
            prestamo_familiar = sum(PONDERA[prestamo_familiar == 1], na.rm =
              ↪  T) / sum(PONDERA, na.rm = T),
            prestamo_bancario = sum(PONDERA[prestamo_bancario == 1], na.rm =
              ↪  T) / sum(PONDERA, na.rm = T),
            comprar_cuotas = sum(PONDERA[comprar_cuotas == 1], na.rm = T) /
              ↪  sum(PONDERA, na.rm = T),
            vender_bienes = sum(PONDERA[vender_bienes == 1], na.rm = T) /
              ↪  sum(PONDERA, na.rm = T)) %>%
  pivot_longer(cols = -ANO4,
               names_to = "tipo_gasto",
```

```
              values_to = "prop")

gastos %>%
  ggplot(aes(x = reorder(tipo_gasto, prop), y = prop, fill =
    ↪  as.factor(ANO4))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Distribución de los gastos especiales realizados por los
    ↪  hogares \nsegún tipo y año",
       caption = "Fuente: elaboración propia en base a EPH-INDEC 2do
         ↪  trimestre 2022-2025",
       x = "Tipo de gasto especial",
       y = "Proporción de hogares",
       fill = "Año") +
  theme_minimal() +
  scale_y_continuous(labels = percent_format()) +
  coord_flip()
```



Por otro lado, si queremos que las etiquetas de los distintos géneros se muestren en forma apropiada, podemos indicarlo a traves de la capa `scale_x_discrete`, utilizando la opción *labels*.

```r
gastos %>%
  ggplot(aes(x = reorder(tipo_gasto, prop), y = prop, fill =
    ↪  as.factor(ANO4))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Distribución de los gastos especiales realizados por los
    ↪  hogares \nsegún tipo y año",
       caption = "Fuente: elaboración propia en base a EPH-INDEC 2do
         ↪  trimestre 2022-2025",
       x = "Tipo de gasto especial",
       y = "Proporción de hogares",
       fill = "Año") +
  theme_minimal() +
  scale_y_continuous(labels = percent_format()) +
  scale_x_discrete(labels = c("ahorros" = "Ahorros",
                              "prestamo_familiar" = "Préstamo familiar",
                              "prestamo_bancario" = "Préstamo bancario",
                              "comprar_cuotas" = "Comprar en cuotas",
                              "vender_bienes" = "Vender bienes")) +
  coord_flip()
```