Clase 2c - Preprocesamiento de la Encuesta Permantente de Hogares

José Rodríguez de la Fuente

Table of contents

Transformando los datos: creación y modificación variables

Teniendo ya la base en formato ordenado, habiendo seleccionado aquellas variables que vayamos a utilizar y filtrando los casos que nos interesan, el siguiente paso en el trabajo cuantitativo suele ser la creación de nuevas variables y/o la modificación de las existentes.

Esta fase en el trabajo investigativo puede implicar, entre otras opciones:

- Generar nuevas variables a partir de las existentes. Por ejemplo, realizar una sumatoria o resta aritmética a partir de los datos disponibles en dos o más variables.
- Recodificar una variable existente. Por ejemplo, reduciendo el número de categorías de una variable categórica o numérica (grupos etarios, niveles de ingreso, etc.).
- Generar una nueva variable a partir del cumplimiento de determinadas condiciones combinadas. Generalmente este trabajo se realiza para construir variables complejas. Por ejemplo, crear una variable que indique el tipo de hogar en cada caso.

Creación de variables mediante cálculos y expresiones condicionales

Para realizar estas operaciones utilizaremos siempre la función `mutate()` del paquete `dplyr`. Básicamente esta función nos creará una nueva columna en la matriz de datos, que contendrá los valores que le indiquemos. Por ejemplo, utilizando esta vez la base de hogares de la última EPH disponible, vamos a calcular la cantidad de personas por ambiente que tiene cada hogar.

```
library(eph)

eph_hog_225 <- get_microdata(year = 2025, period = 2, type = "hogar")
```

Para construir dicho indicador, primero vamos a identificar aquellas variables que necesitamos, es decir, el número de miembros del hogar `IX_TOT` y la cantidad de ambientes que tiene el hogar `IV2`. Entonces, en primer lugar, vamos a crear una nueva variable llamada `personas_ambientes` que represente el cociente entre ambas variables.

> **i** Manual de códigos
>
> Recuerden siempre tener *a mano* el manual de códigos o el diseño de registro de la encuesta que están utilizando para saber el nombre de las variables sobre las que se realizarán las operaciones. En este caso usaremos directamente el diseño de registro.

```
library(tidyverse)

eph_hog_225 <- eph_hog_225 %>%
  mutate(personas_ambiente = IX_TOT / IV2)
```

Ya contamos con la variable `personas_ambiente` que nos indica la proporción de personas por ambiente. A continuación, podemos armar una nueva variable que identifique a aquellos hogares que, según el INDEC, presentan una situación de *hacinamiento crítico* (más de 3 personas por ambiente).

Para ello, introduciremos una nueva función que nos permite plantear condiciones: `case_when()`. Esta función es muy similar al `IF` en `SPSS` o la función `ifelse()` de R base. `case_when()` se pasa dentro del mutate, y se le indica la condición que se quiere evaluar y el valor que se le asignará a la nueva variable en caso de que se cumpla dicha condición, separado por el símbolo `~`. Cada argemento se separa por una coma.

```
eph_hog_225 <- eph_hog_225 %>%
  mutate(hacinamiento_critico = case_when(personas_ambiente > 3 ~ 1,
                                          personas_ambiente <= 3 ~ 0))
```

Este cálculo podría resumirse aún más, concatenando ambas operaciones dentro de la función `mutate()`.

```
eph_hog_225 <- eph_hog_225 %>%
  mutate(personas_ambiente = IX_TOT / IV2,
         hacinamiento_critico = case_when(personas_ambiente > 3 ~ 1,
                                          personas_ambiente <= 3 ~ 0))
```

Si queremos observar los resultados de la tarea realizada, podemos hacerlo a través de la función `table()`.

```
table(eph_hog_225$hacinamiento_critico)
```

```
    0     1
15938   231
```

```
table(eph_hog_225$hacinamiento_critico, useNA = "always") # Agrega los casos
 ↪  NAs
```

```
    0     1  <NA>
15938   231     0
```

Por último, debido a que es una variable categórica, podemos convertirla en factor con la función `factor()` como lo vimos anteriormente. Vamos a indicar que el valor 0 se denomine "Sin hacinamiento crítico" y el valor 1 "Con hacinamiento crítico".

```
eph_hog_225 <- eph_hog_225 %>%
  mutate(hacinamiento_critico = factor(hacinamiento_critico,
                                       levels = c(0, 1),
                                       labels = c("Sin hacinamiento crítico",
                                        ↪  "Con hacinamiento crítico")))

table(eph_hog_225$hacinamiento_critico)
```

```
Sin hacinamiento crítico Con hacinamiento crítico
                   15938                      231
```

Recodificación de variables

El combo `mutate()` y `case_when()` también nos permite recodificar variables, es decir, cambiar los valores de una variable por otros o reducir su número de categorías. Es una práctica frecuente en el análisis de datos, ya que permite simplificar, posteriormente, la interpretación de los resultados.

Por ejemplo, siguiendo con la base de hogares, supongamos que queremos recodificar la variable de *tenencia de la vivienda* (II7). Esta variable tiene un total de 8 categorías:

- propietario de la vivienda y el terreno

- propietario de la vivienda solamente
- inquilino/arrendatario de la vivienda
- ocupante por pago de impuestos/expensas
- ocupante en relación de dependencia
- ocupante gratuito (con permiso)
- ocupante de hecho (sin permiso)
- está en sucesión

Frecuentemente, en los análisis esta variable se agrupa en tres categorías:

- Propietarios (categorías 1 y 2)
- Inquilinos (categoría 3)
- Otra situación (categorías 4 a 8)

Vamos a ver cómo podemos realizar esta recategorización con la función `case_when()`.

```
eph_hog_225 <- eph_hog_225 %>%
  mutate(tenencia_recod = case_when(II7 == 1 | II7 == 2 ~ "Propietario",
                                    II7 == 3 ~ "Inquilino",
                                    II7 %in% c(4:8) ~ "Otra situación"))
```

Una forma de indicar que queremos agrupar valores que se encuentran en un rango es través del operador de pertenencia `%in%` que evalúa si los elementos de un vector están presentes en otro vector. En este caso lo utilizamos para identificar todas las categorías que se englobaban en *otra situación*.

> **Atención**
>
> El operador `%in%` también puede ser utilizado para identificar valores en un vector alfanumérico o de cadena. En ese caso debemos ingresar los valores a evaluar dentro de la función c(). Por ejemplo, `provincia %in% c("Buenos Aires", "Chaco", "Formosa")`.

Variables agregadas

Frecuentemente necesitamos resumir la información de muchas observaciones en un único valor. Por ejemplo, si queremos calcular el promedio de edad o de ingresos, o cualquier otra medida de tendencia central, lo que estamos haciendo es resumir la información de una variable en un único valor.

Otras veces, nuestra variable es de tipo categórica y queremos conocer la frecuencia de cada categoría en el total de la población. Es común, a partir de los datos censales o de encuestas de hogares, calcular tasas de actividad, de empleo, de analfabetismo, o razones masculinidad o femineidad, entre otras. En estos casos, previo al conteo de casos, necesitaremos agrupar a la población en las categorías de interés.

Para ello, en este apartado, revisaremos el uso de las funciones `summarise()` y `group_by()` del paquete `dplyr` para resumir y agregar datos.

Para esta clase, necesitaremos tener instaladas las siguientes librerías:

```
install.packages("summarytools")
```

Para este apartado utilizaremos la base de individuos de la EPH. Vamos a descargarla.

```
eph_ind_225 <- get_microdata(year = 2025, period = 2, type = "individual")
```

Uso de ponderadores para las estimaciones

Generalmente las encuestas de hogares o de individuos como la *EPH* o el *Latinobarómetro*, al ser relevamientos realizados con muestreos probabilísticos, utilizan ponderadores para corregir el sesgo de la muestra. Es decir, permiten ajustar en los casos en que determinadas poblaciones se encuentran sub-representadas o sobre-representadas en la muestra. Los ponderadores son variables de tipo numéricas que *multiplican* el valor de cada caso para alcanzar una estimación más cercana a la realidad sobre la que se quiere investigar.

En el caso de la **EPH** contamos con cuatro variables de ponderación:
- *PONDERA*: es el utilizado para la mayoría de las variables de la encuesta.
- *PONDII*: para el tratamiento del ingreso total individual.
- *PONDIIO*: para el ingreso de la ocupación principal.
- *PONDIH*: para el ingreso total del hogar.

Estadísticas resumen

El análisis exploratorio de datos (**EDA** en inglés) funciona como una primera aproximación a las variables por separado. A la vez, nos sirve para observar si se cumplen determinados supuestos estadísticos.

Podemos clasificar cuatro tipos de estadísticas resumen: las de tendencia central, las de dispersión, las de forma y las de posición. Las primeras nos permiten resumir la información de una variable en un único valor, las segundas nos permiten conocer la variabilidad de los datos, las terceras nos permiten conocer la forma de la distribución de los datos y las últimas nos permiten identificar en que posición se ubican determinados valores de las variables.

Medida de resumen		Escala de medición		
		Cualitativa		Cuantitativa
		Nominal	Ordinal	
Posición central	Moda	✓	✓	✓
	Mediana		✓	✓
	Media			✓
Posición	Valores extremos			✓
	Percentil		✓	✓
Dispersión	Rango			✓
	Rango intercuartil			✓
	Varianza			✓
	Desviación típica			✓
	Coeficiente de variación			✓
Forma	Asimetría			✓
	Curtosis			✓

Figure 1: Fuente: Fachelli y López Roldán (2015)

Calculando medidas resumen en `R`

`R base` cuenta con una función que nos permite calcular estadísticos resumen de una variable de forma rápida y simple, la función `summary()`. Por ejemplo, si quisiéramos estimar el promedio de edad de la población carcelaria, y otros estadísticos descriptivos, podríamos hacerlo de la siguiente forma:

```
summary(eph_ind_225$CH06)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -1.00   18.00   35.00   36.97   54.00  103.00
```

De este modo, observamos que la edad promedio es 36.97, la mediana es 35 y la edad máxima es 103.

El único problema con la función `summary()` es que no nos permite calcular estadísticos ponderados. Para trabajar con datos muestrales existe una librería llamada `summarytools` que nos permite agregar el argumento `weights` para calcular las estadísticas ponderadas. Por ejemplo, si quisiéramos calcular el promedio de edad de la población encuestada utilizando el ponderador, podríamos hacerlo de la siguiente forma utilizando la función `descr()`:

```
library(summarytools)

descr(eph_ind_225$CH06, weights = eph_ind_225$PONDERA)
```

```
Weighted Descriptive Statistics
eph_ind_225$CH06
Weights: PONDERA
N: 29908869

                  eph_ind_225$CH06
--------------- ------------------
           Mean              35.30
        Std.Dev              22.13
            Min              -1.00
         Median              33.00
            Max             103.00
            MAD              25.20
             CV               0.63
        N.Valid        29908869.00
              N        29908869.00
      Pct.Valid             100.00
```

En `dplyr`, la función `summarise()` nos permite realizar el mismo cálculo de forma más ordenada y clara.

```
eph_ind_225 %>%
  summarise(promedio_edad = weighted.mean(CH06, PONDERA)) #ponderado
```

```
# A tibble: 1 x 1
  promedio_edad
          <dbl>
1          35.3
```

```
eph_ind_225 %>%
  summarise(promedio_edad = mean(CH06)) #sin ponderado
```

```
# A tibble: 1 x 1
  promedio_edad
          <dbl>
1          37.0
```

`summarise` acepta toda una serie de estadísticos básicos que pueden utilizar:

- `mean()`: promedio
- `weighted.mean()`: promedio ponderado
- `median()`: mediana
- `min()`: mínimo
- `max()`: máximo
- `sd()`: desviación estándar
- `var()`: varianza
- `n()`: número de observaciones
- `sum()`: suma
- `first()`: primer valor
- `last()`: último valor
- `n_distinct()`: número de valores distintos

```
eph_ind_225 %>%
  summarise(promedio_edad = mean(CH06),
            mediana_edad = median(CH06),
            min_edad = min(CH06),
            max_edad = max(CH06),
            sd_edad = sd(CH06))
```

```
# A tibble: 1 x 5
  promedio_edad mediana_edad min_edad max_edad sd_edad
          <dbl>        <dbl>    <int>    <int>   <dbl>
1          37.0           35       -1      103    22.2
```

De este modo, la función `summarise` nos permite calcular múltiples estadísticos de forma rápida y sencilla y nos devuelve un *data frame* con un valor para cada estadístico calculado.

Para revisar

La mayoría de los estadísticos descriptivos y de dispersión que nos brinda `Rbase` no nos permite utilizar ponderadores. Por lo tanto, si queremos calcular estadísticos ponderados puntuales debemos utilizar otras librerías. Les recomendamos inspeccionar las librerías `survey` y `DescTools`.

Agregación de datos

Otra forma de realizar resúmenes o cálculos sobre las variables es a partir de la información agrupada. Frecuentemente necesitamos que nuestras estadísticas sean calculadas por grupos de edad, sexo, nivel educativo, provincia de residencia, país, etc. Para ello, utilizamos la función `group_by()`, que como su nombre lo indica, nos agrupa la información en base a una variable (o más). En términos de la matriz de datos, lo que hace es cambiar la unidad de análisis: de personas a grupos de edad, por ejemplo.

En los estudios de estratificación social es común calcular variables a nivel del hogar a partir de información individual. Por ejemplo, el ingresto total familiar (ITF), si bien es una variable que ya viene por defecto en la EPH, es calculada luego de que se realiza la encuesta, a partir de la sumatoria de todos los ingresos individuales ($P47T$). Vamos a probar calculándola nosotros y guardándola en un objeto que se llame eph_itf.

```
eph_ind_225 <- eph_ind_225 %>%
  group_by(CODUSU, NRO_HOGAR) %>%  #Agrupamos por hogar
  mutate(ITF_calculado = sum(P47T, na.rm = TRUE)) #Sumamos los ingresos
   ↪  individuales al interior de cada hogar y los asignamos a una nueva
   ↪  variable
```

Fíjense que utilizando los *pipes* es sencillo encadenar cada una de las funciones. Primero agrupamos la información por vivienda y hogar y luego calculamos la sumatoria de ingresos.

Ahora sí, podemos comparar la variable que construimos con aquella que construye el **INDEC**. Vamos a quedarnos solo con los 20 primeros casos de ambas variables. ¿Ven alguna diferencia?

```
eph_ind_225 %>%
  select(CODUSU, NRO_HOGAR, ITF_calculado, ITF) %>%
  head(n = 20)
```

```
# A tibble: 20 x 4
# Groups:   CODUSU, NRO_HOGAR [8]
```

	CODUSU	NRO_HOGAR	ITF_calculado	ITF
	<chr>	<int>	<int>	<int>
1	TQRMNOQWSHMKMPCDEIIAD00878792	1	1400000	1400000
2	TQRMNOQWSHMKMPCDEIIAD00878792	1	1400000	1400000
3	TQRMNOQUVHMLMQCDEIIAD00878965	1	-9	0
4	TQRMNOQUVHMLMQCDEIIAD00878965	1	-9	0
5	TQRMNOQUVHMLMQCDEIIAD00878965	1	-9	0
6	TQRMNOQSWHKMLMCDEHMHF00866730	2	980000	980000
7	TQRMNOQPPHLMKTCDEHMHF00887017	1	532256	532256
8	TQRMNOQPPHLMKTCDEHMHF00887017	1	532256	532256
9	TQRMNOQPPHLMKTCDEHMHF00887017	1	532256	532256
10	TQRMNOQYPHLOKUCDEHMHF00887018	1	830000	830000
11	TQRMNOQYPHLOKUCDEHMHF00887018	1	830000	830000
12	TQRMNOQYPHLOKUCDEHMHF00887018	1	830000	830000
13	TQRMNORSXHMOLMCDEHMHF00877906	1	439991	0
14	TQRMNORSXHMOLMCDEHMHF00877906	1	439991	0
15	TQRMNOQWPHLOLNCDEHMHF00887019	1	1620000	1620000
16	TQRMNOQWPHLOLNCDEHMHF00887019	1	1620000	1620000
17	TQRMNOQWPHLOLNCDEHMHF00887019	1	1620000	1620000
18	TQRMNOPQUHMOLTCDEFIAH00882520	1	1090000	1090000
19	TQRMNOPQUHMOLTCDEFIAH00882520	1	1090000	1090000
20	TQRMNOPQUHMOLTCDEFIAH00882520	1	1090000	1090000

Otro ejemplo frecuente es utilizar alguna variable individual para estratificar a todo el hogar. Por ejemplo, una variable que identifique el máximo nivel educativo que se obtuvo entre todos los miembros. Primero recodificaremos la variable NIVEL_ED para que los casos "sin instrucción" y "NS/NC" tengan valor 0. La línea `TRUE ~ NIVEL_ED` señala que el resto de las categorías se mantienen igual que la original.

```
eph_ind_225 <- eph_ind_225 %>%
  mutate(nivel_educ = case_when(NIVEL_ED >= 7 ~ 0,
                                TRUE ~ NIVEL_ED))
```

Ahora sí para hallar el máximo nivel educativo por hogar, utilizaremos la función `max()`.

```
eph_ind_225 <- eph_ind_225 %>%
  group_by(CODUSU, NRO_HOGAR) %>%
  mutate(max_nivel_educ = max(nivel_educ))
```

Finalmente, podemos observar los resultados obtenidos.

```r
eph_ind_225 %>%
  select(CODUSU, NRO_HOGAR, nivel_educ, max_nivel_educ) %>%
  head(n = 20)
```

```
# A tibble: 20 x 4
# Groups:   CODUSU, NRO_HOGAR [8]
   CODUSU                         NRO_HOGAR nivel_educ max_nivel_educ
   <chr>                              <int>      <dbl>           <dbl>
 1 TQRMNOQWSHMKMPCDEIIAD00878792          1          2               4
 2 TQRMNOQWSHMKMPCDEIIAD00878792          1          4               4
 3 TQRMNOQUVHMLMQCDEIIAD00878965          1          4               6
 4 TQRMNOQUVHMLMQCDEIIAD00878965          1          4               6
 5 TQRMNOQUVHMLMQCDEIIAD00878965          1          6               6
 6 TQRMNOQSWHKMLMCDEHMHF00866730          2          5               5
 7 TQRMNOQPPHLMKTCDEHMHF00887017          1          4               4
 8 TQRMNOQPPHLMKTCDEHMHF00887017          1          0               4
 9 TQRMNOQPPHLMKTCDEHMHF00887017          1          0               4
10 TQRMNOQYPHLOKUCDEHMHF00887018          1          2               4
11 TQRMNOQYPHLOKUCDEHMHF00887018          1          4               4
12 TQRMNOQYPHLOKUCDEHMHF00887018          1          3               4
13 TQRMNORSXHMOLMCDEHMHF00877906          1          2               4
14 TQRMNORSXHMOLMCDEHMHF00877906          1          4               4
15 TQRMNOQWPHLOLNCDEHMHF00887019          1          2               4
16 TQRMNOQWPHLOLNCDEHMHF00887019          1          4               4
17 TQRMNOQWPHLOLNCDEHMHF00887019          1          1               4
18 TQRMNOPQUHMOLTCDEFIAH00882520          1          5               5
19 TQRMNOPQUHMOLTCDEFIAH00882520          1          3               5
20 TQRMNOPQUHMOLTCDEFIAH00882520          1          3               5
```