

# Clase 2b - Preprocesamiento de la Encuesta Permanente de Hogares

José Rodríguez de la Fuente

## Tabla de contenidos

<b>Antes de comenzar</b>	<b>1</b>
<b>Organizando datos y variables: selecciones, filtros y orden</b>	<b>1</b>
Selección de variables . . . . .	2
Filtrado de casos . . . . .	3
Ordenando los datos . . . . .	3
Facilitando el trabajo: el operador <i>pipe</i> . . . . .	4

## Antes de comenzar

En esta clase utilizaremos las siguientes librerías:

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

También vamos a necesitar la última base disponible de la EPH.

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

## Organizando datos y variables: selecciones, filtros y orden

Luego de importar los datos con los que trabajaremos y de haberlos inspeccionado, lo siguiente consiste en organizarlos para posteriormente poder transformarlos, analizarlos e interpretarlos. En este apartado, veremos cómo seleccionar, filtrar y ordenar los datos, basándonos en el

ecosistema de funciones que nos provee el paquete `dplyr`<sup>1</sup> de `tidyverse`. Fundamentalmente, revisaremos las funciones `select()`, `filter()` y `arrange()`.

Como lo indica su [página web](#), `dplyr` es una *gramática* de manipulación de datos basado en un conjunto de verbos que permiten realizar las tareas más comunes de manipulación de datos. Estos verbos son intuitivos y fáciles de recordar, lo que facilita su uso.

## Selección de variables

A menudo trabajamos con bases de datos provenientes de registros o encuestas que contienen una gran cantidad de variables. En estos casos, es común que no todas las variables sean de interés para el análisis que queremos realizar. Otras veces, como en nuestro caso, disponemos de bases de datos, como en las encuestas de hogares, que cuentan con una mayor cantidad de filas que de columnas. Así mismo, seleccionar algunas de las variables con las que vamos a trabajar es una buena opción para reducir el tamaño de la base de datos y facilitar su manipulación y guardado.

Supongamos que quisieramos generar un nuevo objeto que contenga solo las variables `CODUSU`, `NRO_HOGAR`, `COMPONENTE`, `REGION`, `CH06` y `NIVEL_ED`. Para ello, utilizamos la función `select()` y le pasamos como argumento el objeto de la base de datos y las variables que queremos seleccionar, generando un nuevo objeto que se llame `eph_ind_225_sel`.

```
eph_ind_225_sel <- select(eph_ind_225, CODUSU, NRO_HOGAR, COMPONENTE, REGION,  
  CH06, NIVEL_ED)
```

Fijense en el ambiente de trabajo, que el nuevo objeto tiene las mismas observaciones que el objeto original, pero solo tiene las variables que seleccionamos. La función `select()` acepta también incluir rangos de variables, mediante el operador `:`. Por ejemplo, si quisiéramos seleccionar las variables `CODUSU` a `NIVEL_ED`, podríamos hacerlo de la siguiente manera. Sobre-escribimos el objeto `eph_ind_225_sel` para que contenga las variables seleccionadas.

```
eph_ind_225_sel <- select(eph_ind_225, CODUSU:NIVEL_ED)
```

La función, como todas las del paquete `dplyr`, es muy flexible y permite realizar selecciones de variables de forma muy sencilla. Recomendamos inspeccionar la misma mediante el comando `?select`.

---

<sup>1</sup> *Machete* del paquete `dplyr`

## Filtrado de casos

Si con `select` seleccionamos variables, con `filter` seleccionamos casos. Es decir, filtramos las observaciones que cumplen con ciertas condiciones. Para esto es necesario conocer los operadores lógicos y relacionales que vimos en la clase anterior.

Por ejemplo, si sobre el objeto `eph_ind_225` quisiéramos seleccionar solo a los casos de la región pampeana:

```
eph_ind_225_sel <- filter(eph_ind_225, REGION == 43)
```

El código 43 representa a la región *Pampeana* y que cuenta en la base de datos con 14125 casos.

Otro ejemplo que podemos aplicar sobre la base es quedarnos con los casos que tienen más de 18 años utilizando la variable `CH06`

```
eph_ind_225_sel <- filter(eph_ind_225, CH06 > 18)
```

También podríamos combinar ambos filtros, seleccionando casos que residan en la región pampeana y que tengan más de 18 años, usando el operador lógico `&`.

```
eph_ind_225_sel <- filter(eph_ind_225, REGION == 43 & CH06 > 18)
```

Las posibilidades son infinitas y dependiendo del filtro que querramos aplicar las expresiones lógico-relacionales pueden ser más o menos complejas.

### Cuidado

No confundir el operador lógico `==` con el operador de asignación `=`. El primero se utiliza para comparar valores, mientras que el segundo se utiliza para asignar valores a una variable. En SPSS se utiliza el operador `=` para ambos casos.

## Ordenando los datos

Por último, la función `arrange()` nos permite ordenar los datos de acuerdo a una o más variables. Por ejemplo, si quisiéramos ordenar los casos por edad. Por defecto, la función ordena de forma ascendente, pero podemos cambiarlo a descendente utilizando la función `desc()`.

```
eph_ind_225_sel <- arrange(eph_ind_225, CH06)

eph_ind_225_sel <- arrange(eph_ind_225, desc(CH06))
```

## Facilitando el trabajo: el operador *pipe*

Ahora bien, aprovechando que estamos aprendiendo las principales funciones de `dplyr`, vamos a aprender un nuevo modo de trabajo que nos permite una dinámica más sencilla y eficiente. Para esto haremos uso del operador *pipe* (`%>%`), del paquete `magrittr` (tidyverse), que nos permite encadenar funciones de forma más legible y eficiente.



Figura 1: <https://magrittr.tidyverse.org/logo.png>

Por ejemplo, podríamos realizar las siguientes operaciones de selección, filtrado y ordenamiento de la siguiente manera:

```
eph_ind_225_sel <- eph_ind_225 %>%
  select(CODUSU, NRO_HOGAR, COMPONENTE, REGION, CH06, NIVEL_ED) %>%
  filter(REGION == 43 & CH06 > 18) %>%
  arrange(desc(CH06))
```

Fijense que el uso del operador *pipe* no solo nos permite trabajar de un modo más eficiente, sino que también nos simplifica el trabajo de no tener que estar *llamando* al objeto en cada función que aplicamos. La base de datos que estamos transformando, es indicada en la primera línea, y luego cada función se aplica automáticamente sobre la misma.

### 💡 Atención

El operador `%>%` es una de las herramientas más poderosas de R e iremos incorporando su uso a lo largo del seminario. Es fácil de entender su lógica de funcionamiento. Permite que vayamos encadenando funciones, una por una, que apuntan a un único objeto que queda declarado en la primera línea. El atajo de teclado para invocarlo es a través de `Ctrl + Shift + M` en *Windows* y `Cmd + Shift + M` en *Mac*.

Actualmente RStudio proporciona un operador pipe nativo `|>` que funciona del mismo modo que `%>%`.

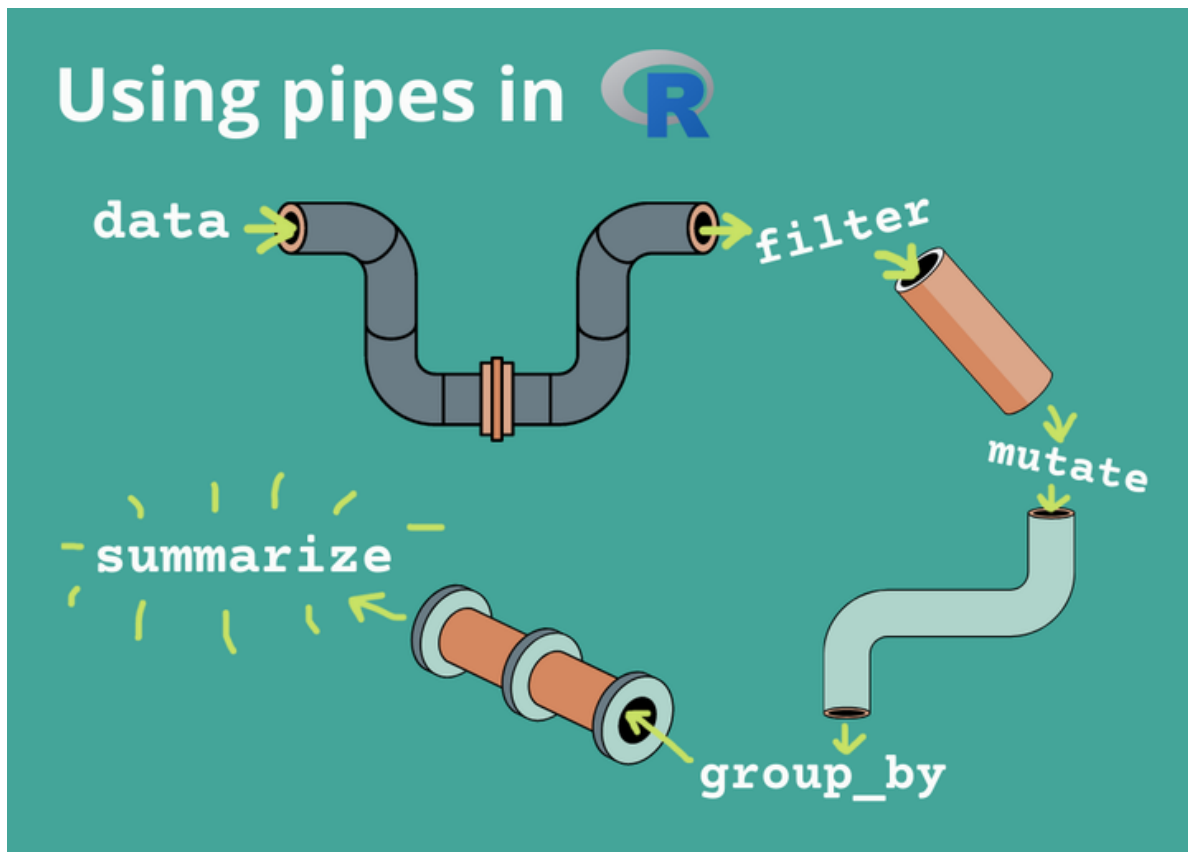


Figura 2: Fuente: <https://www.rforecology.com/post/how-to-use-pipes>