Solución: Ejercicio 2

Importar datos y funciones de `dplyr` y `tidyr`

Unidad de Estadística Servicio de Gestión Estratégica
Departamento de Desarrollo Sostenible e Inteligente

18 Noviembre, 2023

Índice

1. Introducción

En este ejercicio aplicaremos algunas de las cosas aprendidas en el Módulo 3. Trabajaremos con los datos de la **ECH**[1] que estan dentro de la carpeta **Bases**. La idea es que importen los datos, realicen algunos procesamientos utilizando las funciones de la librería **dplyr** y **tidyr**. También vamos a guardar algunos objetos generados en el ejercicio en formato **RData**

1.1. Librerías y conflictos

Antes que nada, ponemos a disposición las librerías o paquetes que vamos a utilizar.

```
library(tidyverse)
library(conflicted)
```

1.2. Arreglamos conflictos

Recordemos que puede pasar que exístan funciones que tengan el mismo nombre en distintas librerías, esto genera que si las usamos, `R` no sabe interpretar qué función queremos implementar. Una forma de solucionar esto es usando la librería **conflicted** indicando a `R` que librería preferimos que use para las funciones repetidas. En nuestro caso, le estamos indicando que si usamos la función **filter** o **select** que sean las que están en la libería **dplyr**.

```
conflict_prefer(name = 'filter',winner = 'dplyr')
conflict_prefer(name = 'select',winner = 'dplyr')
```

2. Importar datos

Importar los datos de la ECH **baseHog.csv** que están dentro de la carpeta **Bases**. Cargar la base en el objeto **dat**. Se puede usar cualquier función para importar datos vista en clase.

En este caso usamos la función **read.table**, pero podría ser cualquier otra de las vistas en clase. Al importar los datos ya le indicamos el separador de decimales (`,`), el separador de campos (`;`), que las columnas tienen nombre (`header`) y le definimos un vector de strings que tienen que interpretarse como datos faltantes (`na.strings`).

```
dat <- read.table('../../Bases/ECH/baseHog.csv',sep = ';',
                  header = T, dec = ',',na.strings = c('NA','','---'))
```

2.1. Chequeo de que los datos en el objeto dat están bien:

- Que tenga la dimensión adecuada
- Que las variables tengan la clase que corresponde

Para ver esto, miramos la dimension de los datos con la función **dim()** y la estructura con la función **str()**. El siguiente chunk lo configuramos para que no se evalúe y no se impriman los resultados en el informe (el parámetro es **eval = F**)

```
### Por ejemplo usaar estas dos, pero no es necesario imprimirlo en el informe ni al
## resultado ni al código
dim(dat)
str(dat)
```

Si encuentran algo en los datos que pueda solucionarse cuando los datos se importan, cambien la forma de importar los datos (como estan los decimales, hay caracteres raros que se pueden definir como `NA`).

[1] Más información sobre la ECH en https://cfevirtual.montevideo.gub.uy/moodle/mod/folder/view.php?id=73598

3. Crear nuevas variables

Al objeto **dat** que contiene los datos le vamos a agregar las siguientes variables:

- **ypc_svl**: Ingreso del hogar sin valor locativo per cápita. (ht11 - ht13)/ht19
- **ind_VmedAdec**: Varible que indique si la vivienda es **Inadecuada** o **Medianamente Adecuada**
- **ht13_rec**: Recodificar la variable valor locativo en 3 categorías (**Menos de 10000**, **Entre 10000 y 20000** y **Más de 20000**).
- **ht11_centr**: Restarle el promedio a la variable **ht11**.
- **ht11_centrMun**: Hacer lo mismo que en la variable anterior pero dentro de cada municipio, es decir, restarle el promedio a la variable ht11.

```
dat = dat %>%
  ### Creamos las variables que se piden usando mutate()
  mutate(ypc_svl = (ht11 - ht13)/ht19,
         ind_VmedAdec = case_when(VivAdec %in% c(2,3) ~ 1,
                                  TRUE ~ 0),
         ht13_rec = cut(ht13,breaks = c(-Inf,9999.9,20000,Inf),
                        labels = c('Menos de 10000','Entre 10000 y 20000','Más de 20000')),
         ht11_centr = ht11 - mean(ht11,na.rm = T)) %>%
  ### Agrupo para crear la variable que usa info de cada municipio
  group_by(municipio) %>%
  ### Creo la variable usando mutate
  mutate(ht11_centrMun = ht11 - mean(ht11,na.rm = T)) %>%
  ### Desarmamoos la agrupación
           ungroup()
```

Comentarios:

- En esta parte del código solo usamos la fuunción **mutate()** para crear nuevas variables. Estas variables serán usadas más adelante para hacer otros procesamientos.
- Es interesante ver como se crean las variables **ht11_centr** y **ht11_centrMun**, si vemos lo que va del lado derecho es exactamente lo mismo, la diferencia es que la primera se calcula con los promedios y desvíos de toda la columna y la segunda con el promedio y desvío de cada municipio.

3.1. Resumen de datos

La idea de este apartado es que utilicen las funciones **group_by()** y la función de resumen **summarise()**

En esta parte vamos a tomar como insumo el objeto **dat** ya con las variables nuevas en el punto anterior y vamos a construir una tabla de resumen **por municipio** llamada **res01** con la siguiente información:

- **TotCasos**: Total de casos. usar la función **n()** para contar casos.
- **Nombrarla Igual**: Promedios de las variables **ht11**, **ht11_centr** y **ht11_centrMun**.
- **mujer_porc**: Porcentaje de Mujeres jefas de hogar.
- **VInMed_porc**: Porcentaje de hogares en viviendas inadecuadas y medianamente adecuadas.
- **vl20000_porc**: Porcentaje de hogares con valor lOcativo **Mayor a 20000**.
- **auto_porc**: Total de hogares que tienen auto
- Ordenar los datos por la variable **auto_porc**

```
res01 = dat %>%
  group_by(municipio) %>%
  summarise(
    TotCasos = n(),
    ht11 = round(mean(ht11,na.rm = T)),
```

```
    ht11_centr = round(mean(ht11_centr,na.rm = T)),
    ht11_centrMun = round(mean(ht11_centrMun,na.rm = T)),
    mujer_porc = round(100*mean(e26 == 2,na.rm = T),1),
    VInMed_porc = round(100*mean(ind_VmedAdec,na.rm = T),1),
    vl20000_porc = round(100*mean(ht13 > 20000,na.rm = T),1),
    auto_porc =  round(100*mean(Auto,na.rm = T),1)) %>%
  ungroup() %>%
  arrange(auto_porc)
```

Comentarios:

- Notemos que para calcular los porcentajes también usamos la función **mean()** (promedio). Esto lo hacemos en los casos donde la variable es *dummy* (vale 0 o 1), al hacer el promedio nos da la proporción que se cumple la condición y al multiplicar por 100 lo pasamos a porcentaje. Esto lo hacemos en el caso de las variables **ind_VmedAdec** y **Auto**.
- De forma similar, también calculamos porcentajes definiendo una condición lógica, en este caso se hace un promedio de un vector lógico que interpreta los `TRUE` como `1` y los `FALSE` como `0`.

3.2. Uso de lista y filtrado

Crear una lista llamada **mi_lista** y guardar:

- En la primer componente los resultados **res01**
- En la segunda componente guardar los resultados de **res01** solo para los municipios (A,C,D)
- En la tercer componente guardar los resultados de **res01** solo para los municipios (A,C,D) y las columnas **mujer_porc** y **ht11**.

```
# Creo una lista vacía
mi_lista = list()
# Completar componente 1
mi_lista[[1]] = res01
# Completar componente 2
mi_lista[[2]] = res01 %>%
  filter(municipio %in% c('A','C','D'))
# Completar componente 3
mi_lista[[3]] =  res01 %>%
  filter(municipio %in% c('A','C','D')) %>%
  select(c(mujer_porc,ht11))
```

4. Uso de joins y funciones de `tidyr`

- Utilizar la función **left_join** para juntar datos de hogares de la ECH con datos de *personas* (**basePer.csv**).
- Elegir solamente las columnas "correlativ", "desocupado", "ocupados", "subempleo", "pobpcoac" y "TotEduc" de personas y considerar los mismos `NA`s que para la base de hogares.
- Juntar datos de personas ECH:
  - Levantar los datos personas ECH
  - Elegir solamente las columnas "correlativ", "desocupado", "ocupados", "subempleo", "pobpcoac" y "TotEduc" de personas
- Mezclar datos personas y hogares considerando los mismos NAs que para hogares.

```
#primero cambio nombre dat a dath
dath <- dat
rm(list=ls(pattern="^dat$")) #rm remueve objetos seleccionados
```

```
#levanto hogares, left join
datp <- read.table('../../Bases/ECH/basePer.csv',sep = ';',
                   header = T, dec = ',',na.strings = c('NA','','---')) %>%
  select(1,9:13) #puede ser tambien select(c(correlativ,desocupado:TotEduc))

dat <- dath %>% left_join(datp,by="correlativ")
```

La variable **pobpcoac** de la base de hogares cuantifica la *condición de actividad económica* de una persona. Presentar en formato de tabla la distribución de personas inactivas que realizan quehaceres en el hogar (pobpcoac==6), en cuyas filas figuren los municipios, en las columnas figure el sexo y en cada una de las celdas de la tabla se muestre el promedio de edad según sexo, para cada municipio de Montevideo.

```
dat %>% select(correlativ,municipio,e26,e27,pobpcoac) %>% filter(pobpcoac==6) %>%
  group_by(municipio,e26) %>%
  summarise(EdPromInact6=round(mean(e27,na.rm=TRUE))) %>% #promedio por edad
  pivot_wider(names_from=e26,values_from=EdPromInact6) %>% #ensancho tabla
  rename("Msc"="1","Fem"="2") %>% #renombro columnas con nombres de categorias
  ungroup
```

```
## # A tibble: 8 x 3
##   municipio   Msc   Fem
##   <chr>     <dbl> <dbl>
## 1 A            56    47
## 2 B            61    60
## 3 C            63    55
## 4 CH           67    67
## 5 D            59    54
## 6 E            61    55
## 7 F            54    53
## 8 G            63    55
```

5. Creamos carpeta y guardamos en RData

5.1. Crear un directorio

Vamos a crear desde `R` una carpeta llamada **Resultados** para guardar los objetos que generamos, para esto usamos el siguiente código

```
if(!dir.exists('Resultados')) { # Pregunto si existe el directorio
  # SI NO EXISTE, LO CREO
  dir.create('Resultados')
}
```

5.2. Guardamos en RData

En esta parte vamos a guardar los objetos **mi_lista** y los datos **dat** en la carpeta que creamos **Resultados** con el nombre **misRes.RData**

```
save(mi_lista,dat,file = 'Resultados/misRes.RData')
```