

Análisis de encuestas de hogares con R

Módulo 0: Introducción a R y dplyr

CEPAL - Unidad de Estadísticas Sociales

Tabla de contenidos I

Conceptos básicos en encuestas de hogares

Manejando una base de datos con R

La muestra

Conceptos básicos en encuestas de hogares

Motivación

Desde que se popularizaron las encuestas de hogares en 1940, se ha hecho evidente algunas tendencias que están ligadas a los avances tecnológicos en las agencias estadísticas y en la sociedad y se han acelerado con la introducción del computador.

Gambino & Silva (2009)

Encuestas de Hogares y los ODS

Las encuestas de hogares son uno de los instrumentos más importantes para hacer seguimiento a los indicadores de los ODS en el marco de la agenda 2030.

Universo de estudio

- ▶ El término encuesta se encuentra directamente relacionado con una población finita compuesta de individuos a los cuales es necesario entrevistar.
- ▶ Este conjunto de unidades de interés recibe el nombre de *población objetivo* o *universo* y sobre ellas se obtiene la información de interés para el estudio.
- ▶ Por ejemplo, *la Encuesta Nacional de Empleo y Desempleo* de Ecuador define su población objetivo como todas las personas mayores de 10 años residentes en viviendas particulares en Ecuador.

Unidades de análisis

- ▶ Corresponden a los diferentes niveles de desagregación establecidos para consolidar el diseño probabilístico y sobre los que se presentan los resultados de interés.
- ▶ En México, la *Encuesta Nacional de Ingresos y Gastos de los Hogares* define como unidades de análisis el ámbito al que pertenece la vivienda, urbano alto, complemento urbano y rural.
- ▶ La *Gran Encuesta Integrada de Hogares* de Colombia tiene cobertura nacional y sus unidades de análisis están definidas por 13 grandes ciudades junto con sus áreas metropolitanas.

Unidades de muestreo

- ▶ El diseño de una encuesta de hogares en América Latina plantea la necesidad de seleccionar en varias etapas ciertas *unidades de muestreo* que sirven como medio para seleccionar finalmente a los hogares que participarán de la muestra.
- ▶ La *Pesquisa Nacional por Amostra de Domicilios* en Brasil se realiza por medio de una muestra de viviendas en tres etapas.

Unidades de muestreo en PNAD

1. Las unidades primarias de muestreo (UPM) son los municipios,
2. Las unidades secundarias de muestreo (USM) son los sectores censales, que conforman una malla territorial conformada en el último Censo Demográfico.
3. Las últimas unidades en ser seleccionadas son las viviendas.

Marcos de muestreo

- ▶ Para realizar el proceso de selección sistemática de los hogares es necesario contar con un marco de muestreo que sirva de *link* entre los hogares y las unidades de muestreo y que permita tener acceso a la población de interés.
- ▶ El marco de muestreo debe permitir identificar y ubicar a todos los hogares que conforman la población objetivo.
- ▶ Los marcos de muestreo más utilizados en este tipo de encuestas son de áreas geográficas que vinculan directamente a los hogares o personas.

Ejemplo de Costa Rica

- ▶ La *Encuesta Nacional de Hogares* de utiliza un marco muestral construido a partir de los censos nacionales de población y vivienda de 2011.
- ▶ Corresponde a un marco de áreas en donde sus unidades son superficies geográficas asociadas con las viviendas.
- ▶ Permite la definición de UPM con 150 viviendas en las zonas urbanas y 100 viviendas en las zonas rurales.
- ▶ El marco está conformado por 10461 UPM (64.5% urbanas y 35.5% rurales).

Objetivos de la PNAD

- ▶ La *Pesquisa Nacional por Amostra de Domicílios Contínua* es implementada cada trimestre por el *Instituto Brasileiro de Geografia e Estatística*.
- ▶ Su objetivo es producir información básica para el estudio de la evolución económica de Brasil y la publicación continua de indicadores demográficos.
- ▶ Los constructos de ingreso, gastos y empleo son evaluados de forma continua.
- ▶ Además evalúa temas de vivienda, migración de los individuos del hogar, trabajo infantil, fecundidad, salud y seguridad alimentaria, uso de las tecnologías de información, transferencias de renta, uso del tiempo, entre otros.

Manejando una base de datos con R

R como herramienta de análisis

Es posible utilizar **R** como herramienta de análisis de una base de datos que contenga información de una encuesta de hogares.

Creación de proyectos en R

Para inicial un procesamiento en R, por experiencia y por una cultura de buenas practicas de programación se recomienda crear un proyecto en el cual tengamos disponibles toda nuestra información. A continuación se muestra el paso a paso para crear un proyecto dentro de RStudio

- ▶ **Paso 1:** Abrir RStudio.
- ▶ **Paso 2:** ir a file -> New Project

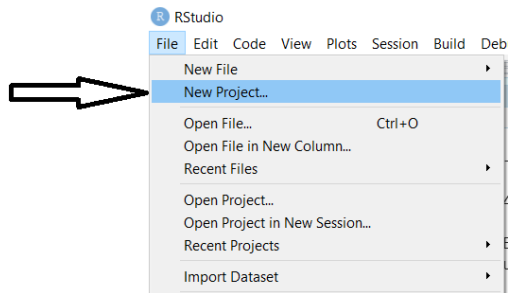


Figura 1: *Crear el proyecto*

Paso 3: Tipos de proyecto.

En nuestro caso tomaremos *New Directory*

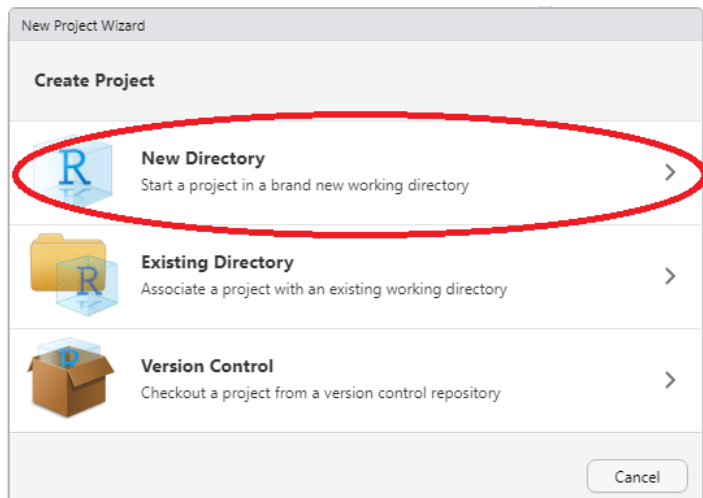


Figura 2: *Tipos de proyectos*

Paso 3: Definir el tipo de proyecto.

- ▶ *New Directory*: Aquí RStudio nos brinda una variedad de opciones dependiendo las características del procesamiento que desea realizar.
- ▶ *Existing Directory*: Si contamos con algunos código desarrollados previamente, esta sería la opción a elegir.
- ▶ *Version Control*: Si contamos con cuenta en *Git* y deseamos tener una copia de seguridad podemos emplear esta opción.

Paso 4:

Seleccionar el tipo de proyecto.

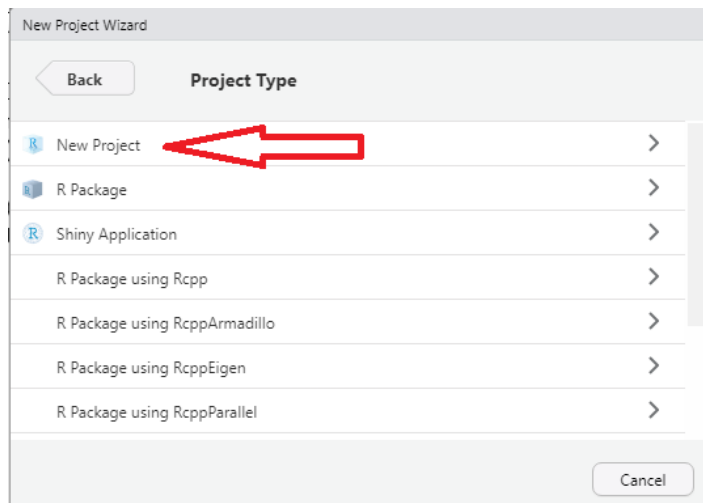


Figura 3: *Seleccionar el tipo de proyecto*

Paso 5

Diligenciar el nombre del proyecto y la carpeta de destino.

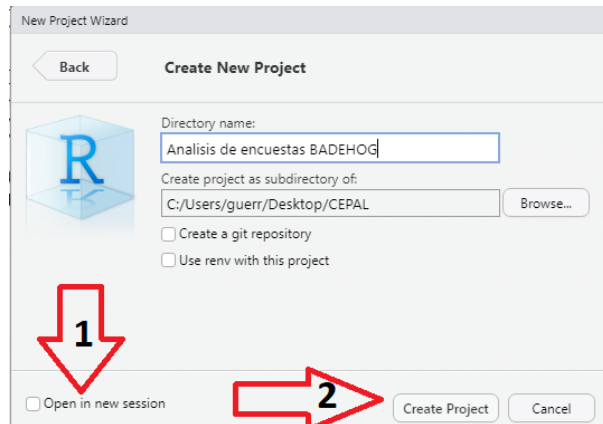


Figura 4: *Nombre de proyecto*

El realizar estos pasos permite que todas las rutinas creadas dentro del proyecto estén ancladas a la carpeta del proyecto.

Algunas librerías de interés

Para analizar una base de datos, en R utilizaremos las siguientes librerías:

- ▶ `dplyr`, para manejar eficientemente las bases de datos.
- ▶ `readstata13` para leer las bases de datos de STATA.
- ▶ `survey` para analizar los datos de las encuestas.
- ▶ `srvyr` para utilizar los *pipe operators* en las consultas.
- ▶ `ggplot2` para generar los gráficos.
- ▶ `TeachingSampling` para seleccionar muestras.
- ▶ `samplesize4surveys` para calcular los tamaños de muestra.

Instalando las librerías

Antes de poder utilizar las diferentes funciones que cada librería trae, es necesario descargarlas de Internet. El comando `install.packages` permite realizar esta tarea. Note que algunas librerías pueden depender de otras, así que para poder utilizarlas es necesario instalar también las dependencias.

```
install.packages("dplyr")
```

```
install.packages("readstata13")
```

```
install.packages("ggplot2")
```

```
install.packages("TeachingSampling")
```

```
install.packages("samplesize4surveys")
```

```
install.packages("survey")
```

```
install.packages("srvyr")
```

Cargando las librerías

Recuerde que es necesario haber instalado las librerías para poder utilizarlas. Una vez instaladas hay que informarle al software que vamos a utilizarlas con el comando `library`.

```
rm(list = ls())  
  
library(dplyr)  
library(readstata13)  
library(survey)  
library(srvyr)  
library(ggplot2)  
library(TeachingSampling)  
library(samplesize4surveys)
```

Leyendo la base de datos

Para mostrar el uso de algunas funciones básicas en el análisis de datos emplearemos la base de datos BigCity disponible en la librería TeachingSampling

```
data("BigCity", package = "TeachingSampling")  
saveRDS(BigCity, "Imágenes/01_ManejoBase/BigCity.rds")
```

Leyendo la base de datos

Para cargar la base de datos en R es necesario utilizar la función `readRDS`.

```
data2 <- readRDS("Imagenes/01_ManejoBase/BigCity.rds")
```


Registros y variables

La función `nrow` identifica el número de registros (unidades efectivamente medidas) en la base de datos y la función `ncol` muestra el número de variables en la base de datos.

```
nrow(data2)
```

```
[1] 150266
```

```
ncol(data2)
```

```
[1] 12
```

```
dim(data2)
```

```
[1] 150266      12
```

Visor externo

La función `View` abre un visor externo y permite navegar por los registros de la base de datos

```
View(data2)
```

La base de datos



The image shows a screenshot of the RStudio data viewer window. The window title is 'data2'. Below the title bar, there are icons for refreshing, saving, and filtering. The main area displays a table with 12 columns and 20 rows. The columns are: HHID, PersonID, Stratum, PSU, Zone, Sex, Age, MaritalST, Income, Expenditure, Employment, and Poverty. The rows contain numerical and categorical data, including IDs, strata, PSUs, zones, genders, ages, marital status, income, expenditure, employment status, and poverty status.

	HHID	PersonID	Stratum	PSU	Zone	Sex	Age	MaritalST	Income	Expenditure	Employment	Poverty
1	idHH00001	idPer01	idStrt001	PSU0001	Rural	Male	38	Married	555.00	488.33	Employed	NotPoor
2	idHH00001	idPer02	idStrt001	PSU0001	Rural	Female	40	Married	555.00	488.33	Employed	NotPoor
3	idHH00001	idPer03	idStrt001	PSU0001	Rural	Female	20	Single	555.00	488.33	Inactive	NotPoor
4	idHH00001	idPer04	idStrt001	PSU0001	Rural	Male	19	Single	555.00	488.33	Employed	NotPoor
5	idHH00001	idPer05	idStrt001	PSU0001	Rural	Male	18	Single	555.00	488.33	Inactive	NotPoor
6	idHH00002	idPer01	idStrt001	PSU0001	Rural	Male	35	Married	298.34	216.70	Employed	Relative
7	idHH00002	idPer02	idStrt001	PSU0001	Rural	Female	29	Married	298.34	216.70	Employed	Relative
8	idHH00002	idPer03	idStrt001	PSU0001	Rural	Female	14	Single	298.34	216.70	NA	Relative
9	idHH00002	idPer04	idStrt001	PSU0001	Rural	Female	13	Single	298.34	216.70	NA	Relative
10	idHH00002	idPer05	idStrt001	PSU0001	Rural	Male	6	NA	298.34	216.70	NA	Relative
11	idHH00003	idPer01	idStrt001	PSU0001	Rural	Male	49	Married	1070.00	610.37	Employed	NotPoor
12	idHH00003	idPer02	idStrt001	PSU0001	Rural	Female	52	Married	1070.00	610.37	Employed	NotPoor
13	idHH00003	idPer03	idStrt001	PSU0001	Rural	Female	28	Single	1070.00	610.37	Employed	NotPoor
14	idHH00003	idPer04	idStrt001	PSU0001	Rural	Male	24	Single	1070.00	610.37	Employed	NotPoor
15	idHH00004	idPer01	idStrt001	PSU0001	Rural	Male	44	Married	550.00	462.83	Employed	NotPoor
16	idHH00004	idPer02	idStrt001	PSU0001	Rural	Female	44	Married	550.00	462.83	Inactive	NotPoor
17	idHH00004	idPer03	idStrt001	PSU0001	Rural	Male	18	Single	550.00	462.83	Employed	NotPoor
18	idHH00004	idPer04	idStrt001	PSU0001	Rural	Female	13	Single	550.00	462.83	NA	NotPoor
19	idHH00005	idPer01	idStrt001	PSU0001	Rural	Male	46	Partner	623.75	613.70	Employed	NotPoor
20	idHH00005	idPer02	idStrt001	PSU0001	Rural	Female	45	Partner	623.75	613.70	Inactive	NotPoor

Figura 5: *Visor de bases de datos de RStudio*

Reconociendo las variables

La función `names` identifica las variables de la base de datos.

```
names(data2)
```

```
[1] "HHID"      "PersonID"  "Stratum"   "PSU"       "Zone"
[6] "Sex"       "Age"       "MaritalST" "Income"    "Expenditure"
[11] "Employment" "Poverty"
```

Reconociendo las variables

La función `str` muestra de manera compacta la estructura de un objeto y sus componentes, en este caso la base de datos.

```
str(data2)
```

```
'data.frame':  150266 obs. of  12 variables:
 $ HHID      : chr  "idHH00001" "idHH00001" "idHH00001" "idHH00001" ...
 $ PersonID  : chr  "idPer01" "idPer02" "idPer03" "idPer04" ...
 $ Stratum   : chr  "idStrt001" "idStrt001" "idStrt001" "idStrt001" ...
 $ PSU       : chr  "PSU0001" "PSU0001" "PSU0001" "PSU0001" ...
 $ Zone      : chr  "Rural" "Rural" "Rural" "Rural" ...
 $ Sex       : chr  "Male" "Female" "Female" "Male" ...
 $ Age       : int   38 40 20 19 18 35 29 14 13 6 ...
 $ MaritalST : Factor w/ 6 levels "Partner","Married",...: 2 2 5 5 5 2 2 5
 $ Income    : num   555 555 555 555 555 ...
 $ Expenditure: num   488 488 488 488 488 ...
 $ Employment: Factor w/ 3 levels "Unemployed","Inactive",...: 3 3 2 3 2 3
 $ Poverty   : Factor w/ 3 levels "NotPoor","Extreme",...: 1 1 1 1 1 3 3 3
```

Definir las regiones

La base de datos no cuenta con regiones geográficas definidas las vamos a construir agrupando algunos estratos, para ello usamos el siguiente código:

```
Region <- as.numeric(  
  gsub(pattern = "\\D",  
        replacement = "", x = data2$Stratum))  
data2$Region <-  
  cut(Region, breaks = 5,  
      labels = c("Norte", "Sur", "Centro", "Occidente", "Oriente"))
```

Añadiendo el códigos a las regiones

En algunas ocasiones es necesario re-codificar los niveles de los factores. El siguiente código permite generar los nombres de las regiones.

```
data2$IDRegion <- factor(data2$Region,  
  levels = c("Norte","Sur","Centro","Occidente","Oriente"),  
  labels = c("01", "02","03","04","05"))
```

Añadiendo el nombre corto a las regiones

Para efectos de visualización en tablas y gráficos a veces conviene codificar los nombres de las variables.

```
data2$Nom_corto <- factor(data2$Region,  
  levels = c("Norte","Sur","Centro","Occidente","Oriente"),  
  labels = c("N", "S","C","O","E"))
```


El operador pipe

R es un lenguaje de programación creado por estadísticos para estadísticos. Una de las contribuciones recientes es el desarrollo de los `pipelines` que permiten de una forma intuitiva generar consultas y objetos desde una base de datos.

El operador más importante es `%>%` que le indica a R que el objeto que está a su izquierda debe ser un argumento del código a su derecha.

Número de registros

El operador `%>%` indica que el objeto a su izquierda (la base de datos BigCity) debe ser un argumento para la función que está a su derecha (el número de filas).

```
data2 %>% count()
```

n
150266

Verbos que debemos aprender

- ▶ **filter**: mantiene un criterio de filtro sobre alguna variable o mezcla de variables.
- ▶ **select**: selecciona columnas por nombre.
- ▶ **arrange**: re-ordena las filas de la base de datos.
- ▶ **mutate**: añade nuevas variables a la base de datos.
- ▶ **summarise**: reduce variables a valores y los presenta en una tabla.
- ▶ **group_by**: ejecuta funciones y agrupa el resultado por las variables de interés.

Utilizando pipes

El número de hogares en la base de datos

```
data2[,1:5] %>% slice(1:8)
```

HHID	PersonID	Stratum	PSU	Zone
idHH00001	idPer01	idStrt001	PSU0001	Rural
idHH00001	idPer02	idStrt001	PSU0001	Rural
idHH00001	idPer03	idStrt001	PSU0001	Rural
idHH00001	idPer04	idStrt001	PSU0001	Rural
idHH00001	idPer05	idStrt001	PSU0001	Rural
idHH00002	idPer01	idStrt001	PSU0001	Rural
idHH00002	idPer02	idStrt001	PSU0001	Rural
idHH00002	idPer03	idStrt001	PSU0001	Rural

El número de registros (personas) en la base de datos

```
data2 %>% count()
```

n
150266

filter

- ▶ Las encuestas de hogares muchas veces recopilan información a nivel de viviendas, hogares y personas.
- ▶ Las bases de datos de datos que están disponibles en BADEHOG están a nivel de persona.
- ▶ Las encuestas se pueden analizar para un región de interés

filter para region

El siguiente código filtra la base de datos por la condición de región

```
dataregion1 <- data2 %>% filter(IDRegion == "01")  
  
dataregion2 <- data2 %>% filter(Region == "Norte")  
  
# View(dataregion1)  
  
# View(dataregion2)
```

filter para Zona

El siguiente código filtra la base de datos por la ubicación de la persona en el Zona rural y urbana.

```
dataurbano <- data2 %>%  
  filter(Zone == "Urban")  
  
datarural <- data2 %>%  
  filter(Zone == "Rural")  
  
# View(dataurbano)  
# View(datarural)
```

filter para ingresos

El siguiente código filtra la base de datos por personas de ingresos mensuales bajos y altos.

```
dataingreso1 <- data2 %>%  
  filter(Income %in% c(50, 100))  
  
dataingreso2 <- data2 %>%  
  filter(Income %in% c(1000, 2000))  
  
# View(dataingreso1)  
# View(dataingreso2)
```


select para reducción de columnas

El siguiente código reduce la base de datos original utilizando la función `select`.

```
datared <- data2 %>% select(`HHID`, `PSU`,  
                           `Region`, `Stratum`)  
  
datablue <- data2 %>% select(PersonID, Age,  
                             Sex, Income)  
  
# View(datared)  
# View(datablue)
```

select para reducción de columnas

El siguiente código reduce la base de datos original utilizando la función `select`.

```
datagrey <- data2 %>% select(-MaritalST, -IDRegion)
datagrey %>% View()
```

arrange para ordenar la base

El siguiente código ordena la base de datos original utilizando la función `arrange`.

```
datadog <- datablue %>% arrange(Income)
datadog %>% head()
```

PersonID	Age	Sex	Income
idPer01	82	Female	10.00
idPer01	82	Female	10.00
idPer01	68	Male	11.92
idPer01	68	Male	11.92
idPer01	58	Male	12.08
idPer02	50	Female	12.08

arrange sobre más variables

Es posible utilizar la función `arrange` para hacer ordenamientos más complicados.

```
data %>% arrange(Sex, Age) %>% head()
```

PersonID	Age	Sex	Income
idPer04	0	Female	151.67
idPer04	0	Female	151.67
idPer06	0	Female	527.84
idPer06	0	Female	527.84
idPer04	0	Female	355.00
idPer04	0	Female	355.00

arrange sobre más variables

Es posible utilizar la función `arrange` junto con la opción `desc()` para que el ordenamiento sea descendente.

```
data %>% arrange(desc(Age)) %>% head()
```

PersonID	Age	Sex	Income
idPer02	98	Female	135.00
idPer02	98	Female	135.00
idPer01	98	Female	136.89
idPer01	98	Female	136.89
idPer01	98	Male	244.92
idPer01	98	Male	244.92

mutate para crear nuevas variables

Esta función crea nuevas variables en la base de datos que pueden ser guardadas como un objeto diferente en R.

```
datablue2 <- datablue %>%  
  mutate(Income2 = 2 * Income)  
datablue2 %>% head()
```

PersonID	Age	Sex	Income	Income2
idPer01	38	Male	555.00	1110.00
idPer02	40	Female	555.00	1110.00
idPer03	20	Female	555.00	1110.00
idPer04	19	Male	555.00	1110.00
idPer05	18	Male	555.00	1110.00
idPer01	35	Male	298.34	596.68

mutate sistemático

La función `mutate` reconoce sistemáticamente las variables que van siendo creadas de manera ordenada.

```
datacat <- datablue %>%  
  mutate(Income2 = 2 * Income,  
         Income4 = 2 * Income2)  
datacat %>% head()
```

PersonID	Age	Sex	Income	Income2	Income4
idPer01	38	Male	555.00	1110.00	2220.00
idPer02	40	Female	555.00	1110.00	2220.00
idPer03	20	Female	555.00	1110.00	2220.00
idPer04	19	Male	555.00	1110.00	2220.00
idPer05	18	Male	555.00	1110.00	2220.00
idPer01	35	Male	298.34	596.68	1193.36

Definir categorías para una variable

A veces, resulta fundamental categorizar variables como la edad para agilizar el análisis. En este escenario, optaremos por emplear la función `case_when`, la cual posibilita la evaluación de distintas condiciones para una variable particular.

```
data2 <- data2 %>% mutate(  
  CatAge = case_when(  
    Age <= 5 ~ "0-5",  
    Age <= 15 ~ "6-15",  
    Age <= 30 ~ "16-30",  
    Age <= 45 ~ "31-45",  
    Age <= 60 ~ "46-60",  
    TRUE ~ "Más de 60"  
  ),  
  CatAge = factor(  
    CatAge,  
    levels = c("0-5", "6-15", "16-30", "31-45", "46-60", "Más de 60"),  
    ordered = TRUE  
  )  
)
```


Número de registros por región

El siguiente código permite generar el número de registros en cada una de las regiones de BigCity. El comando `group_by` agrupa los datos por región, el comando `summarise` hace los cálculos requeridos y el comando `arrange` ordena los resultados

```
data2 %>%  
  group_by(Region) %>%  
  summarise(n = n()) %>% arrange(desc(n))
```

Número de registros por región

El resultado de la anterior consulta es el siguiente:

Region	n
Oriente	39160
Occidente	33868
Centro	25944
Sur	25898
Norte	25396

Número de registros por sexo

El siguiente código permite generar el Número de registros discriminado por el sexo.

```
data2 %>%  
  group_by(Sex) %>%  
  summarise(n = n()) %>% arrange(desc(n))
```

Sex	n
Female	79190
Male	71076

Número de registros por área geográfica

El siguiente código reporta el Número de registros en el área urbana y rural.

```
data2 %>%  
  group_by(Zone) %>%  
  summarise(n = n()) %>% arrange(desc(n))
```

Zone	n
Urban	78164
Rural	72102

Número de registros estado de ocupación

El siguiente código reporta el Número de registros clasificado Ocupado, desocupado e inactivo

```
data2 %>%  
  group_by(Employment) %>%  
  summarise(n = n()) %>% arrange(desc(n))
```

Employment	n
Employed	62188
Inactive	44104
NA	39344
Unemployed	4630

La muestra

Bibliografía y referencias

- ▶ Kish, L. (1965) *Survey Sampling*. John Wiley and Sons.
- ▶ Cochran, W. G. (1977) *Sampling Techniques*. John Wiley and Sons.
- ▶ Särndal, et. al. (2003) *Model-assisted Survey Sampling*. Springer.
- ▶ Gutiérrez, H. A. (2016) *Estrategias de muestreo: diseño de encuestas y estimación de parámetros*. Ediciones de la U.
- ▶ Gutiérrez, H. A. (2017) TeachingSampling. *R package*.

Muestreo en dos etapas estratificado

- ▶ La teoría discutida en las secciones anteriores es aplicable cuando las unidades primarias de muestreo son seleccionadas dentro de un estrato.
- ▶ No hay nuevos principios de estimación o diseño involucrado en el desarrollo de esta estrategia de muestreo.

Muestreo en dos etapas estratificado

- ▶ Se supone que el muestreo en cada estrato respeta el principio de la independencia.
- ▶ Las estimaciones del total, así como el cálculo y estimación de la varianza son simplemente resultado de añadir o sumar para cada estrato la respectiva cantidad.

Muestreo en dos etapas estratificado

- ▶ Dentro de cada estrato U_h $h = 1, \dots, H$ existen N_{Ih} unidades primarias de muestreo, de las cuales se selecciona una muestra s_{Ih} de n_{Ih} unidades mediante un diseño de muestreo aleatorio simple.
- ▶ Suponga, además que el sub-muestreo dentro de cada unidad primaria seleccionada es también aleatorio simple.
- ▶ Para cada unidad primaria de muestreo seleccionada $i \in s_{Ih}$ de tamaño N_i se selecciona una muestra s_i de elementos de tamaño n_i .

Muestreo en dos etapas estratificado

Para utilizar los principios de estimación del último conglomerado en este diseño particular se definen las siguientes cantidades:

1. $d_{I_i} = \frac{N_{Ih}}{n_{Ih}}$, que es el factor de expansión de la i -ésima UPM en el estrato h .
2. $d_{k|i} = \frac{N_i}{n_i}$, que es el factor de expansión del k -ésimo hogar para la i -ésima UPM.
3. $d_k = d_{I_i} \times d_{k|i} = \frac{N_{Ih}}{n_{Ih}} \times \frac{N_i}{n_i}$, que es el factor de expansión final del k -ésimo elemento para toda la población U .

Práctica en R

```
data('BigCity')

FrameI <- BigCity %>% group_by(PSU) %>%
  summarise(Stratum = unique(Stratum),
            Persons = n(),
            Income = sum(Income),
            Expenditure = sum(Expenditure))

attach(FrameI)
```

Práctica en R

```
head(FrameI, 10)
```

PSU	Stratum	Persons	Income	Expenditure
PSU0001	idStrt001	118	70911.72	44231.78
PSU0002	idStrt001	136	68886.60	38381.90
PSU0003	idStrt001	96	37213.10	19494.78
PSU0004	idStrt001	88	36926.46	24030.74
PSU0005	idStrt001	110	57493.88	31142.36
PSU0006	idStrt001	116	75272.06	43473.28
PSU0007	idStrt001	68	33027.84	21832.66
PSU0008	idStrt001	136	64293.02	47660.02
PSU0009	idStrt001	122	33156.14	23292.16
PSU0010	idStrt002	70	65253.78	37114.76

Práctica en R

```
sizes = FrameI %>% group_by(Stratum) %>%  
  summarise(NIh = n(),  
    nIh = 2,  
    dI = NIh/nIh)  
  
NIh <- sizes$NIh  
nIh <- sizes$nIh
```

Práctica en R

```
head(sizes, 10)
```

Stratum	Nlh	nlh	dl
idStrt001	9	2	4.5
idStrt002	11	2	5.5
idStrt003	7	2	3.5
idStrt004	13	2	6.5
idStrt005	11	2	5.5
idStrt006	5	2	2.5
idStrt007	14	2	7.0
idStrt008	7	2	3.5
idStrt009	8	2	4.0
idStrt010	8	2	4.0

Práctica en R

```
set.seed(1234)
samI <- S.STSI(Stratum, NIh, nIh)
UI <- levels(as.factor(FrameI$PSU))
sampleI <- UI[samI]

FrameII <- left_join(sizes,
                     BigCity[which(BigCity$PSU %in% sampleI), ])
attach(FrameII)
```


Práctica en R

```
head(FrameII, 10) %>% select(Stratum:Zone)
```

Stratum	Nlh	nlh	dl	HHID	PersonID	PSU	Zone
idStrt001	9	2	4.5	idHH00053	idPer01	PSU0005	Rural
idStrt001	9	2	4.5	idHH00053	idPer02	PSU0005	Rural
idStrt001	9	2	4.5	idHH00053	idPer03	PSU0005	Rural
idStrt001	9	2	4.5	idHH00053	idPer04	PSU0005	Rural
idStrt001	9	2	4.5	idHH00053	idPer05	PSU0005	Rural
idStrt001	9	2	4.5	idHH00053	idPer06	PSU0005	Rural
idStrt001	9	2	4.5	idHH00054	idPer01	PSU0005	Rural
idStrt001	9	2	4.5	idHH00054	idPer02	PSU0005	Rural
idStrt001	9	2	4.5	idHH00054	idPer03	PSU0005	Rural
idStrt001	9	2	4.5	idHH00054	idPer04	PSU0005	Rural

Práctica en R

```
HHdb <- FrameII %>%  
  group_by(PSU) %>%  
  summarise(Ni = length(unique(HHID)))  
  
Ni <- as.numeric(HHdb$Ni)  
ni <- ceiling(Ni * 0.1)  
sum(ni)
```

```
[1] 697
```

Práctica en R

```
sam = S.SI(Ni[1], ni[1])

clusterII = FrameII[which(FrameII$PSU == sampleI[1]),]

sam.HH <- data.frame(HHID = unique(clusterII$HHID)[sam])

clusterHH <- left_join(sam.HH, clusterII, by = "HHID")

clusterHH$dki <- Ni[1] / ni[1]

clusterHH$dk <- clusterHH$dI * clusterHH$dki

sam_data = clusterHH
```

Práctica en R

```
head(sam_data, 10) %>% select(Stratum:Zone)
```

Stratum	Nlh	nlh	dl	PersonID	PSU	Zone
idStrt001	9	2	4.5	idPer01	PSU0005	Rural
idStrt001	9	2	4.5	idPer02	PSU0005	Rural
idStrt001	9	2	4.5	idPer03	PSU0005	Rural
idStrt001	9	2	4.5	idPer04	PSU0005	Rural
idStrt001	9	2	4.5	idPer01	PSU0005	Rural
idStrt001	9	2	4.5	idPer02	PSU0005	Rural
idStrt001	9	2	4.5	idPer03	PSU0005	Rural
idStrt001	9	2	4.5	idPer04	PSU0005	Rural
idStrt001	9	2	4.5	idPer01	PSU0005	Rural
idStrt001	9	2	4.5	idPer02	PSU0005	Rural

Práctica en R

```
set.seed(1234)
for (i in 2:length(Ni)) {
  sam = S.SI(Ni[i], ni[i])
  clusterII = FrameII[which(FrameII$PSU == sampleI[i]), ]

  sam.HH <- data.frame(HHID = unique(clusterII$HHID)[sam])
  clusterHH <- left_join(sam.HH, clusterII, by = "HHID")

  clusterHH$dki <- Ni[i] / ni[i]
  clusterHH$dk <- clusterHH$dI * clusterHH$dki

  data1 = clusterHH
  sam_data = rbind(sam_data, data1)
}
encuesta <- sam_data
```

Práctica en R

```
dim(encuesta)
```

```
[1] 2675  17
```

```
sum(encuesta$dk)
```

```
[1] 157538
```

```
nrow(BigCity)
```

```
[1] 150266
```

```
attach(encuesta)
```

Práctica en R

Definir diseño muestral con la librería srvyr

```
library(srvyr)

diseno <- encuesta %>%
  as_survey_design(
    strata = Stratum,
    ids = PSU,
    weights = dk,
    nest = T
  )

sum(weights(diseno))
```

```
[1] 157538
```

Práctica en R

Calibrando los pesos muestrales, para ello empleamos la función `calibrate` de la librería `survey`

```
library(survey)
totales <- colSums(
  model.matrix(~ -1 + Zone:Sex, BigCity)) # Obtener totales Pob.
diseno_cal <- calibrate(
  diseno, ~-1 + Zone:Sex, totales, calfun = "linear")

sum(weights(diseno))
```

```
[1] 157538
```

```
sum(weights(diseno_cal))
```

```
[1] 150266
```

```
nrow(BigCity)
```

```
[1] 150266
```

```
encuesta$wk <- weights(diseno_cal)
```


Práctica en R

```
par(mfrow = c(1,2))  
hist(encuesta$dk) ; hist(encuesta$wk)
```

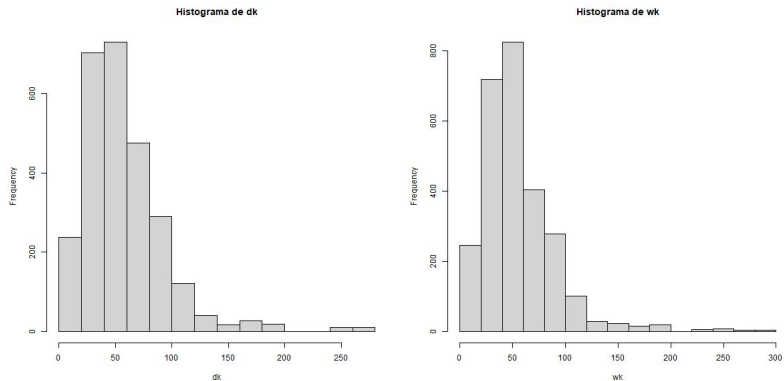
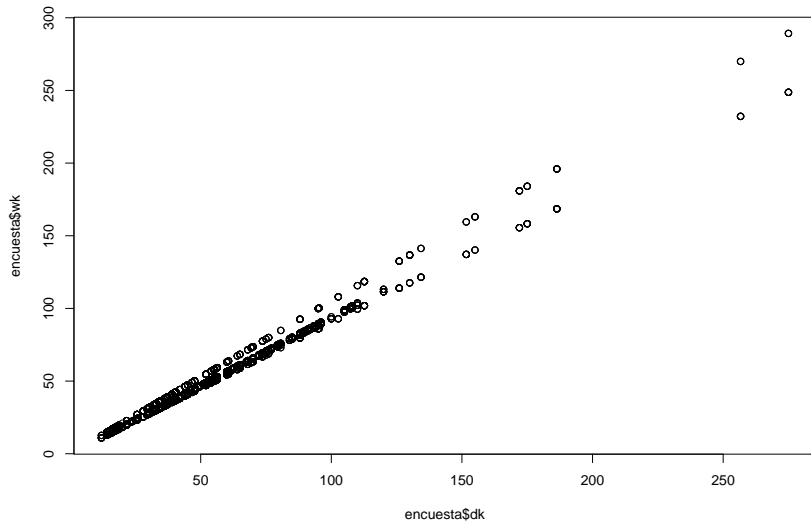


Figura 6: *Histograma de los pesos*

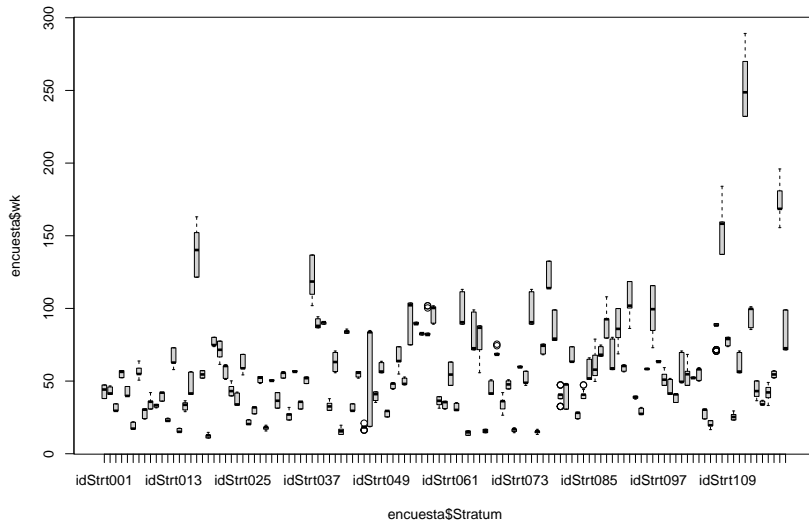
Práctica en R

```
plot(encuesta$dk, encuesta$wk)
```



Práctica en R

```
boxplot(encuesta$wk ~ encuesta$Stratum)
```



Práctica en R

```
saveRDS(object = encuesta, file = "../Data/encuesta.rds")
```

¡Gracias!

Email: andres.gutierrez@cepal.org