

# Análisis de encuestas de hogares con R

## Módulo 0: Introducción a R y dplyr

CEPAL - Unidad de Estadísticas Sociales

# Tabla de contenidos I

Manejando una base de encuestas de hogares con R

Descriptivos y reflexión

Resúmenes agrupados

Manejando una base de encuestas de hogares 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.

# Algunas librerías de interés

Para analizar la PNAD, 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")
```

# Cargando las librerías

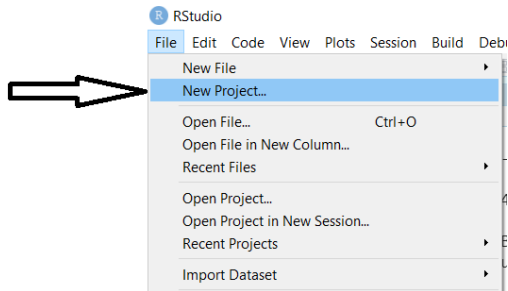
*Recuerde que es necesario haber instalado las librerías para poder utilizarlas. Una vez instaladas hay que informarle al software que vamos a utilizarlas con el comando `library`.*

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

# 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



### Paso 3: Tipos de proyecto.

En nuestro caso tomaremos *New Directory*

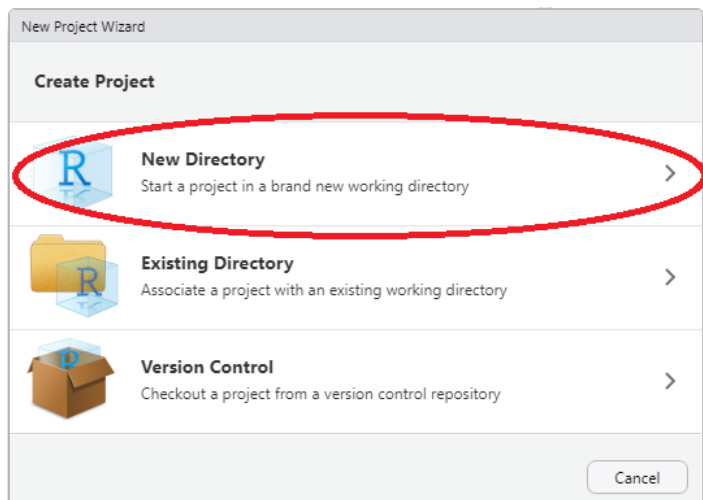


Figura 1: *Tipos de proyectos*

### 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.

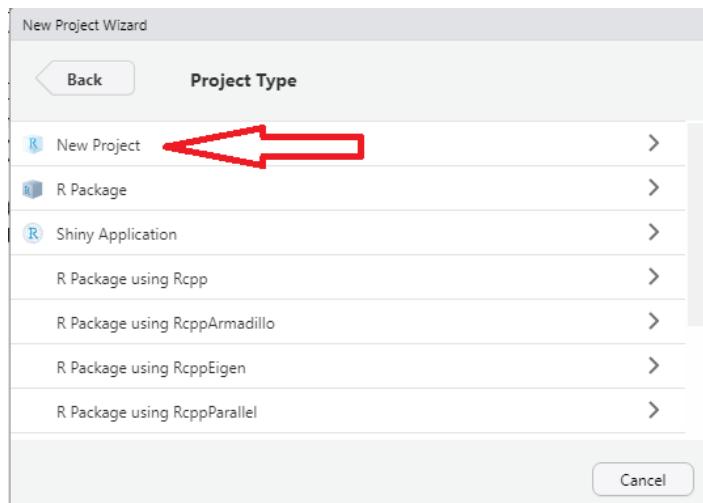


Figura 2: *Seleccionar el tipo de proyecto*

## Paso 5

Diligenciar el nombre del proyecto y la carpeta de destino.

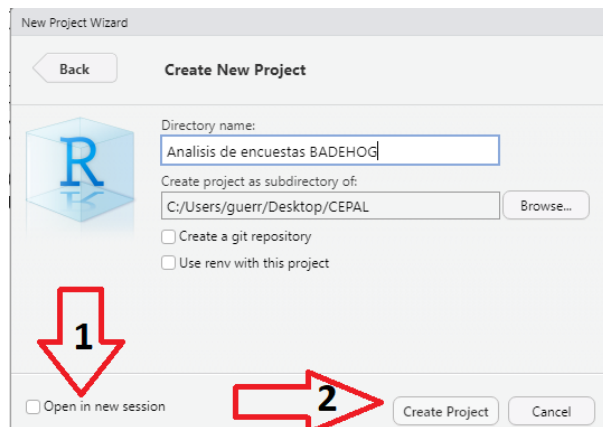


Figura 3: *Nombre de proyecto*

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

## Leyendo la base de datos

La función `read.dta13` permite leer la base de datos desde STATA 13 (es un proceso eficiente de menos de 3 segundos). Luego, convertimos esta base a formato `.RDS` que es un formato más eficiente y genérico de R.

```
data1 <- read.dta13("Z:/BC/BRA_2015N.dta")
saveRDS(data1, "../data/BRA_2015N.rds")
data2 <- readRDS("../data/BRA_2015N.rds")
```

## Leyendo la base de datos

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

```
data2 <- readRDS("../data/BRA_2015N.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] 356904
```

```
ncol(data2)
```

```
[1] 109
```

```
dim(data2)
```

```
[1] 356904    109
```

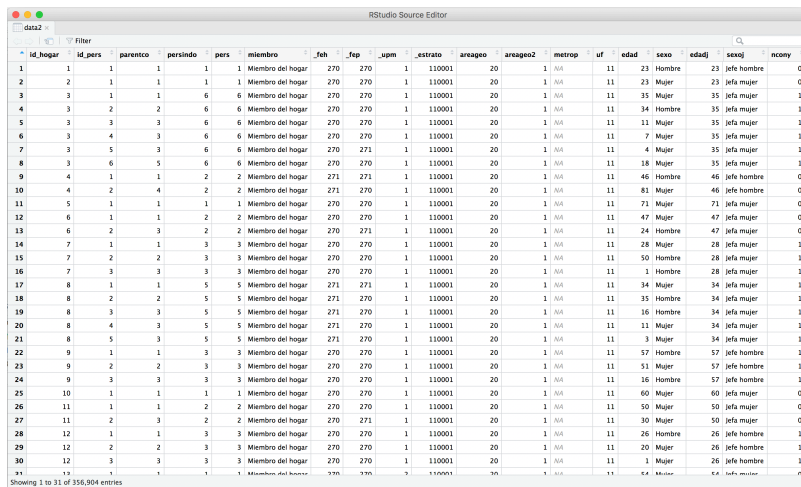
## Visor externo

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

```
View(data2)
```



# La base de datos



RStudio Source Editor

data2

Filter

	id_hogar	id_pers	parentco	persindo	pers	miembro	_feh	_fep	_upm	_estrato	areageo	areageo2	metrop	uf	edad	sexo	edadj	sexoj	ncomy
1	1	1	1	1	1	Miembro del hogar	270	270	1	110001	20	1	NA	11	23	Hombre	23	jefe hombre	0
2	2	1	1	1	1	Miembro del hogar	270	270	1	110001	20	1	NA	11	23	Mujer	23	jefa mujer	0
3	3	1	1	1	6	Miembro del hogar	270	270	1	110001	20	1	NA	11	35	Mujer	35	jefa mujer	1
4	3	2	2	2	6	Miembro del hogar	270	270	1	110001	20	1	NA	11	34	Hombre	35	jefa mujer	1
5	3	3	3	3	6	Miembro del hogar	270	270	1	110001	20	1	NA	11	11	Mujer	35	jefa mujer	1
6	3	4	3	3	6	Miembro del hogar	270	270	1	110001	20	1	NA	11	7	Mujer	35	jefa mujer	1
7	3	5	3	3	6	Miembro del hogar	270	271	1	110001	20	1	NA	11	4	Mujer	35	jefa mujer	1
8	3	6	5	6	6	Miembro del hogar	270	270	1	110001	20	1	NA	11	18	Mujer	35	jefa mujer	1
9	4	1	1	2	2	Miembro del hogar	271	271	1	110001	20	1	NA	11	46	Hombre	46	jefe hombre	0
10	4	2	4	2	2	Miembro del hogar	271	270	1	110001	20	1	NA	11	81	Mujer	46	jefe hombre	0
11	5	1	1	1	1	Miembro del hogar	270	270	1	110001	20	1	NA	11	71	Mujer	71	jefa mujer	0
12	6	1	1	2	2	Miembro del hogar	270	270	1	110001	20	1	NA	11	47	Mujer	47	jefa mujer	0
13	6	2	3	2	2	Miembro del hogar	270	271	1	110001	20	1	NA	11	24	Hombre	47	jefa mujer	0
14	7	1	1	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	28	Mujer	28	jefa mujer	1
15	7	2	2	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	50	Hombre	28	jefa mujer	1
16	7	3	3	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	1	Hombre	28	jefa mujer	1
17	8	1	1	5	5	Miembro del hogar	271	271	1	110001	20	1	NA	11	34	Mujer	34	jefa mujer	1
18	8	2	2	5	5	Miembro del hogar	271	270	1	110001	20	1	NA	11	35	Hombre	34	jefa mujer	1
19	8	3	3	5	5	Miembro del hogar	271	270	1	110001	20	1	NA	11	16	Hombre	34	jefa mujer	1
20	8	4	3	5	5	Miembro del hogar	271	270	1	110001	20	1	NA	11	11	Mujer	34	jefa mujer	1
21	8	5	3	5	5	Miembro del hogar	271	270	1	110001	20	1	NA	11	3	Mujer	34	jefa mujer	1
22	9	1	1	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	57	Hombre	57	jefe hombre	1
23	9	2	2	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	51	Mujer	57	jefe hombre	1
24	9	3	3	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	16	Hombre	57	jefe hombre	1
25	10	1	1	1	1	Miembro del hogar	270	270	1	110001	20	1	NA	11	60	Mujer	60	jefa mujer	0
26	11	1	1	2	2	Miembro del hogar	270	270	1	110001	20	1	NA	11	50	Mujer	50	jefa mujer	0
27	11	2	3	2	2	Miembro del hogar	270	271	1	110001	20	1	NA	11	30	Mujer	50	jefa mujer	0
28	12	1	1	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	26	Hombre	26	jefe hombre	1
29	12	2	2	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	20	Mujer	26	jefe hombre	1
30	12	3	3	3	3	Miembro del hogar	270	270	1	110001	20	1	NA	11	1	Mujer	26	jefe hombre	1
31	13	1	1	1	1	Miembro del hogar	270	270	1	110001	20	1	NA	11	64	Mujer	64	jefa mujer	0

Showing 1 to 31 of 356,904 entries

Figura 4: *Visor de bases de datos de RStudio*

# Reconociendo las variables

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

```
names(data2)
```

[1] "id_hogar"	"id_pers"	"cotiza_ee"
[4] "parentco"	"persindo"	"pers"
[7] "miembro"	"_feh"	"_fep"
[10] "_upm"	"_estrato"	"areageo"
[13] "areageo2"	"metrop"	"uf"
[16] "edad"	"sexo"	"edadj"
[19] "sexoj"	"ncony"	"anoest"
[22] "conduct"	"conduct3"	"ocupr_p"
[25] "ocupr_s"	"ocupr_pj"	"ocupr68_ee"
[28] "rama_p"	"rama_s"	"rama_ee"
[31] "ramar_p"	"ramar_s"	"ramar_pj"
[34] "ramar_ee"	"categ_p"	"categ_s"
[37] "categ_pj"	"categ5_p"	"categ5_s"
[40] "horas_ee"	"area_ee"	"conductr_ee"
[43] "ocup_ee"	"sector_ee"	"paren_ee"
[46] "ecivil_ee"	"tenen_ee"	"etnia_ee"
[49] "asiste_ee"	"lee_ee"	"nbi_agua_ee"
[52] "nbi_saneamiento_ee"	"nbi_elect_ee"	"nbi_combus_ee"
[55] "nbi_interhog_ee"	"nbi_compuhog_ee"	"nbi_pared_ee"
[58] "nbi_techo_ee"	"tamest_ee"	"nhijos_ee"
[61] "tipol_ee"	"afilia_ee"	"ncuartos_ee"
[64] "ndormitorios_ee"	"niveduc_ee"	"bajaprod_ee"
[67] "nbi_piso_ee"	"sys_po"	"gan_po"
[70] "yemp_po"	"yjub_po"	"sys_ho"
[73] "gan_ho"	"yemp_ho"	"yjub_ho"
[76] "yaim_ho"	"sys_pe"	"gan_pe"
[79] "yoemp_pe"	"yemp_pe"	"yjub_pe"
[82] "yotr_pe"	"ycap_pe"	"yotn_pe"
[85] "yto_pe"	"sys_he"	"gan_he"
[88] "yoemp_he"	"yemp_he"	"yjub_he"

# 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':  356904 obs. of  109 variables:
 $ id_hogar      : int  1 2 3 3 3 3 3 3 4 4 ...
 $ id_pers       : int  1 1 1 2 3 4 5 6 1 2 ...
 $ cotiza_ee     : Factor w/ 2 levels "No cotiza/aporta a la seguridad social",...: 1 2 2 1 1 1 1 2 1 1 ...
 $ parentco      : int  1 1 1 2 3 3 3 5 1 4 ...
 $ persindo      : int  1 1 6 6 6 6 6 6 2 2 ...
 $ pers          : int  1 1 6 6 6 6 6 6 2 2 ...
 $ miembro       : Factor w/ 2 levels "No es miembro del hogar",...: 2 2 2 2 2 2 2 2 2 2 ...
 $ _feh          : int  270 270 270 270 270 270 270 270 271 271 ...
 $ _fep          : int  270 270 270 270 270 270 271 270 271 270 ...
 $ _upm          : int  1 1 1 1 1 1 1 1 1 1 ...
 $ _estrato      : int  110001 110001 110001 110001 110001 110001 110001 110001 110001 110001 ...
 $ areageo       : int  20 20 20 20 20 20 20 20 20 20 ...
 $ areageo2      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ metrop        : int  NA NA NA NA NA NA NA NA NA NA ...
 $ uf            : int  11 11 11 11 11 11 11 11 11 11 ...
 $ edad          : int  23 23 35 34 11 7 4 18 46 81 ...
 $ sexo          : Factor w/ 2 levels "Hombre","Mujer": 1 2 2 1 2 2 2 2 1 2 ...
 $ edadj         : int  23 23 35 35 35 35 35 35 46 46 ...
 $ sexoj         : Factor w/ 2 levels "Jefe hombre",...: 1 2 2 2 2 2 2 2 1 1 ...
 $ ncony         : int  0 0 1 1 1 1 1 1 0 0 ...
 $ anoest        : int  12 12 15 15 4 2 0 12 6 3 ...
 $ conduct       : int  1 1 1 1 3 3 -1 1 1 3 ...
 $ conduct3      : int  1 1 1 1 3 3 -1 1 1 3 ...
 $ ocupr_p       : int  5 4 2 1 -1 -1 -1 4 8 -1 ...
 $ ocupr_s       : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
 $ ocupr_pj      : int  5 4 2 2 2 2 2 2 8 8 ...
 $ ocupr68_ee    : Factor w/ 10 levels "No aplica","Trabajadores que no pueden ser clasificados",...: 7 5 3 4 1 1 1 5 10 1 ...
```

## Añadiendo el nombre a los estados

En algunas ocasiones es necesario re-codificar los niveles de los factores. El siguiente código permite generar los nombres de los estados en Brasil.

```
data2$estados <- factor(data2$uf,  
  levels = c(11:17, 21:29, 31:33, 35, 41:43, 50:53),  
  labels = c("Rondonia", "Acre", "Amazonas", "Roraima", "Para",  
    "Amapa", "Tocantins", "Maranhao", "Piaui", "Ceara",  
    "RioGrandeNorte", "Paraiba", "Pernambuco", "Alagoas",  
    "Sergipe", "Bahia", "MinasGerais", "EspirituSanto",  
    "RioJaneiro", "SaoPaulo", "Parana", "SantaCatarina",  
    "RioGrandeSur", "MatoGrossoSur", "MatoGrosso", "Goias",  
    "DistritoFederal"))
```

## Añadiendo el nombre a los estados

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

```
data2$deptos <- factor(data2$uf,  
  levels = c(11:17, 21:29, 31:33, 35, 41:43, 50:53),  
  labels = c("RO", "AC", "AM", "RR", "PA",  
    "AP", "TO", "MA", "PI", "CE", "RN", "PB",  
    "PE", "AL", "SE", "BA", "MG", "ES", "RJ", "SP",  
    "PR", "SC", "RS", "MS", "MT", "GO", "DF"))
```

## 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 encuestas en Brasil

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

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

<u>n</u>
<u>356904</u>

# 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)
```

id_hogar	id_pers	cotiza_ee	parentco	persindo
1	1	No cotiza/aporta a la seguridad social	1	1
2	1	Cotiza/Aporta a la seguridad social	1	1
3	1	Cotiza/Aporta a la seguridad social	1	6
3	2	No cotiza/aporta a la seguridad social	2	6
3	3	No cotiza/aporta a la seguridad social	3	6
3	4	No cotiza/aporta a la seguridad social	3	6
3	5	No cotiza/aporta a la seguridad social	3	6
3	6	Cotiza/Aporta a la seguridad social	5	6

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

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

n
356904

## 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.
- ▶ Se puede filtrar por hogar muy fácilmente porque sólo hay un jefe de hogar por hogar.

## **filter** para hogar

El siguiente código filtra la base de datos por la condición de parentesco.

```
datahogar1 <- data2 %>% filter(parentco == 1)
datahogar2 <- data2 %>% filter(paren_ee == "Jefe")

# View(datahogar1)
# View(datahogar2)
```

## filter para área

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

```
dataurbano <- data2 %>%  
  filter(area_ee == "Area urbana")  
datarural <- data2 %>%  
  filter(area_ee == "Area 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(ingcorte %in% c(50, 100))  
dataingreso2 <- data2 %>%  
  filter(ingcorte %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(`id_hogar`, `_upm`,  
                           `_feh`, `_estrato`)  
datablue <- data2 %>% select(id_pers, edad,  
                           sexo, ingcorte)  
  
# 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(-id_hogar, -id_pers)
datagrey %>% View()
```

## arrange para ordenar la base

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

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

id_pers	edad	sexo	ingcorte
1	38	Mujer	0
2	12	Mujer	0
1	26	Hombre	0
2	29	Mujer	0
1	50	Hombre	0
1	53	Mujer	0



## arrange sobre más variables

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

```
datatable %>% arrange(sexo, edad) %>% head()
```

id_pers	edad	sexo	ingcorte
6	0	Hombre	660.4400
6	0	Hombre	162.5000
3	0	Hombre	381.6667
5	0	Hombre	320.0000
6	0	Hombre	375.0000
4	0	Hombre	1425.0000

## arrange sobre más variables

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

```
datatable %>% arrange(desc(edad)) %>% head()
```

id_pers	edad	sexo	ingcorte
2	115	Mujer	103.0000
4	110	Mujer	1156.5300
2	107	Hombre	415.5904
1	107	Mujer	1754.4600
3	105	Mujer	380.7904
2	105	Mujer	898.3200

## 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(ingreso2 = 2 * ingcorte)  
datablue2 %>% head()
```

id_pers	edad	sexo	ingcorte	ingreso2
1	23	Hombre	800.0	1600.0
1	23	Mujer	1150.0	2300.0
1	35	Mujer	904.4	1808.8
2	34	Hombre	904.4	1808.8
3	11	Mujer	904.4	1808.8
4	7	Mujer	904.4	1808.8

## mutate sistemático

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

```
datacat <- datablue %>%  
  mutate(ingreso2 = 2 * ingcorte,  
         ingreso4 = 2 * ingreso2)  
datacat %>% head()
```

id_pers	edad	sexo	ingcorte	ingreso2	ingreso4
1	23	Hombre	800.0	1600.0	3200.0
1	23	Mujer	1150.0	2300.0	4600.0
1	35	Mujer	904.4	1808.8	3617.6
2	34	Hombre	904.4	1808.8	3617.6
3	11	Mujer	904.4	1808.8	3617.6
4	7	Mujer	904.4	1808.8	3617.6

# Número de encuestas por estado

El siguiente código permite generar el número de encuestas efectivas en cada uno de los estados de Brasil. El comando `group_by` agrupa los datos por estados, el comando `summarise` hace los cálculos requeridos y el comando `arrange` ordena los resultados

```
data2 %>%  
  group_by(estados) %>%  
  summarise(n = n()) %>% arrange(desc(n)) %>% slice(1:10)
```

# Número de encuestas por estado

El resultado de la anterior consulta es el siguiente:

estados	n
SaoPaulo	40008
MinasGerais	32933
RioGrandeSur	26259
Bahia	26155
RioJaneiro	25858
Para	22489
Pernambuco	21309
Parana	18707
Ceara	17819
Goias	14666

## Número de encuestas por sexo

El siguiente código permite generar el número efectivo de encuestas discriminado por el sexo del respondiente.

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

sexo	n
Mujer	183681
Hombre	173223

# Número de encuestas por área geográfica

El siguiente código reporta el número efectivo de encuestas en el área urbana y rural.

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

area_ee	n
Area urbana	304564
Area rural	52340



## Número de encuestas por parentesco

El siguiente código reporta el número efectivo de encuestas clasificado por jefe de hogar, hijos, conyugues, etc.

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

paren_ee	n
Hijos	126206
Jefe	117939
Cónyuge	73725
Otros parientes	36508
Otros no parientes	2342
Servicio doméstico	184

Descriptivos y reflexión

# Funciones estadísticas básicas

- ▶ Media: `mean()`
- ▶ Mediana: `median()`
- ▶ Varianza: `var()`
- ▶ Desviación estándar: `sd()`
- ▶ Percentiles: `quantile()`
- ▶ Algunas medidas descriptivas: `summary()`
- ▶ Covarianza: `cov( , )`
- ▶ Correlación: `cor( , )`

# Reflexionemos

- ▶ ¿Qué es una encuesta?
- ▶ ¿Qué es una muestra?
- ▶ ¿Qué es una muestra representativa?
- ▶ ¿Está bien sacar conclusiones sobre una muestra?
- ▶ ¿Podemos tomar la muestra y hacer inferencia directamente desde la muestra?

# Reflexionemos

- ▶ Si calculamos el promedio de los ingresos en una encuesta, ¿qué significa esa cifra?
- ▶ Si calculamos el total de los ingresos en una encuesta, ¿qué significa esa cifra?
- ▶ ¿Qué necesitamos para que la inferencia sea precisa y exacta?
- ▶ ¿Qué es el principio de representatividad?
- ▶ ¿Qué es el factor de expansión?

## Para reflexionar...

- ▶ Una encuesta de hogares requiere análisis de todas las variables que se quisieron medir, este proceso debe ser llevado a cabo por separado para asegurar la calidad y consistencia de los datos recolectados.
- ▶ Sin embargo, *no* vamos a adentrarnos en el análisis de las variables en la muestra, porque los datos muestrales no son de interés para el investigador.
- ▶ A nosotros nos interesa lo que suceda a nivel poblacional y este análisis se debe abordar desde la teoría del muestreo.

# ¡PELIGRO!

*Los siguientes resultados no tienen interpretación poblacional y se realizan con el único propósito de ilustrar el manejo de las bases de datos de las encuestas.*

## Medias y totales

La función `summarise` permite conocer el total de los ingresos en la base de datos y la media de los ingresos sobre los respondientes.

```
data2 %>% summarise(total.ing = sum(ingcorte),  
                    media.ing = mean(ingcorte))
```

total.ing	media.ing
422286293	1183.193

NO TIENE INTERPRETACIÓN POBLACIONAL



## Medianas y percentiles

La función `summarise` permite conocer algunas medidas de localización de los ingresos en la base de datos.

```
data2 %>% summarise(mediana = median(ingcorte),  
                    decil1 = quantile(ingcorte, 0.1),  
                    decil9 = quantile(ingcorte, 0.9),  
                    rangodecil = decil9 - decil1)
```

mediana	decil1	decil9	rangodecil
732.8571	244.8872	2308.5	2063.613

NO TIENE INTERPRETACIÓN POBLACIONAL

# Varianza y desviación estándar

La función `summarise` permite conocer el comportamiento variacional de los ingresos sobre los respondientes.

```
data2 %>% summarise(varianza = var(ingcorte),  
                    desv = sd(ingcorte))
```

varianza	desv
3407496	1845.94

NO TIENE INTERPRETACIÓN POBLACIONAL

# Rangos

La función `summarise` permite conocer el comportamiento variacional de los ingresos sobre los respondientes.

```
data2 %>% summarise(mini = min(ingcorte),  
                    maxi = max(ingcorte),  
                    rango = maxi - mini,  
                    rangoiq = IQR(ingcorte))
```

mini	maxi	rango	rangoiq
0	171000	171000	869.8312

NO TIENE INTERPRETACIÓN POBLACIONAL

## Resúmenes agrupados

## Media de los ingresos por área

```
data2 %>% group_by(area_ee) %>%  
  summarise(n = n(),  
            media = mean(ingcorte))
```

area_ee	n	media
Area urbana	304564	1277.5786
Area rural	52340	633.9673

NO TIENE INTERPRETACIÓN POBLACIONAL

## Media de los ingresos por sexo

```
data2 %>% group_by(sexo) %>%  
  summarise(n = n(),  
            media = mean(ingcorte))
```

sexo	n	media
Hombre	173223	1192.463
Mujer	183681	1174.450

NO TIENE INTERPRETACIÓN POBLACIONAL

## Media de los ingresos por sexo

```
data2 %>% group_by(sexoj) %>%  
  summarise(n = n(),  
            media = mean(ingcorte))
```

sexoj	n	media
Jefe hombre	218476	1240.028
Jefa mujer	138428	1093.492

NO TIENE INTERPRETACIÓN POBLACIONAL

## Descriptivos de los ingresos por sexo en hogares

```
data2 %>% filter(paren_ee == "Jefe") %>%  
  group_by(sexoj) %>%  
  summarise(n = n(),  
            media = mean(ingcorte),  
            desv = sd(ingcorte),  
            rangoiq = IQR(ingcorte))
```

sexoj	n	media	desv	rangoiq
Jefe hombre	70154	1456.456	2324.760	1025.7312
Jefa mujer	47785	1333.633	2076.323	942.6096

NO TIENE INTERPRETACIÓN POBLACIONAL



## Descriptivos de los ingresos por condición de ocupación

```
data2 %>% group_by(condact) %>%  
  summarise(n = n(),  
            media = mean(ingcorte),  
            desv = sd(ingcorte),  
            rangoiq = IQR(ingcorte))
```

condact	n	media	desv	rangoiq
-1	22937	764.2203	1135.5858	524.3429
1	165325	1458.3929	2190.5175	1027.7429
2	17896	694.6610	948.8768	496.8096
3	150746	1003.1239	1526.9308	705.5163

NO TIENE INTERPRETACIÓN POBLACIONAL

## Descriptivos de los ingresos por condición de ocupación en hogares

```
data2 %>% filter(paren_ee == "Jefe") %>%  
  group_by(conduct) %>%  
  summarise(n = n(),  
            media = mean(ingcorte),  
            desv = sd(ingcorte),  
            rangoiq = IQR(ingcorte))
```

conduct	n	media	desv	rangoiq
1	77852	1525.8277	2459.4976	1095.9096
2	4469	535.1117	778.3737	440.9096
3	35618	1255.6499	1730.2059	879.6096

NO TIENE INTERPRETACIÓN POBLACIONAL

# Descriptivos de los ingresos en hogares por pobreza

```
data2 %>% filter(paren_ee == "Jefe") %>%  
  group_by(pobreza) %>%  
  summarise(n = n(),  
            media = mean(ingcorte),  
            desv = sd(ingcorte),  
            rangoiq = IQR(ingcorte))
```

pobreza	n	media	desv	rangoiq
Pobreza extrema	3918	79.87131	52.72501	88.8872
Pobreza no extrema	13688	269.02412	62.47018	107.1667
Fuera de la pobreza	100333	1613.71191	2355.32143	1054.8800

NO TIENE INTERPRETACIÓN POBLACIONAL

¡Gracias!

*Email:* [andres.gutierrez@cepal.org](mailto:andres.gutierrez@cepal.org)