R para el monitoreo de la política de desarrollo social

Ana Escoto Mónica Lara

09/30/2022

Table of contents

Introducción al curso

Objetivo general

El objetivo del curso es que las personas adscritas a la CGMEFFI desarrollen habilidades en el uso del software especializado "R" para fortalecer el análisis y potenciar el alcance de la información derivada del monitoreo de la política de desarrollo social.

Temas

1. Manejo y procesamiento de datos

1.1 Tipos y estructuras de datos

1.2 Operaciones básicas

1.3 Manejo de datos

1.4 Ciclos, secuencias y condicionales

1.5 Funciones

2. Visualización de datos

2.1 Generación de gráficas con ggplot

2.2 Edición de gráficas con ggplot

2.3 Visualización espacial

2.4 Creación de tableros

3. Análisis de texto

3.1 Estructura y carga de datos

3.2 Análisis de palabras

3.3 Relación de texto

Metodología

La metodología del curso consistirá en lo siguiente:

1. La exposición de la facilitadora. Durante la primera parte de la sesión, se expondrán los comandos necesarios para llevar a cabo cada tema. Se dará una introducción sobre la temática y se buscará dar ejemplos concretos para facilitar el aprendizaje. Se espera que el personal exponga sus dudas o comentarios a lo largo de la explicación.

2. Realización de ejercicios prácticos. Al final de cada sesión, corresponderá a las personas asistentes del curso realizar individualmente o en parejas un ejercicio relacionado con lo visto en la primera parte de la clase.

3. Consulta autónoma de material. Tanto la exposición como los ejercicios serán acompañado de material de consulta realizado ad hoc para el curso y el contenido, de tal manera que el estudiantado pueda volver a los códigos y las explicaciones posteriormente.

Facilitadoras

Ana Ruth Escoto Castillo

Doctora en Estudios de Población. Centro de Estudios Demográficos y Urbanos, El Colegio de México.

Semblanza Profesora de tiempo completo en la Facultad de Ciencias Políticas y Sociales. Investigadora nivel I en elSistema Nacional de Investigadores. Maestra en Población y Desarrollo por la Facultad Latinoamericana de Ciencias Sociales (FLACSO) – Sede México. Posee experiencia en recolección de información estadística, diseño y control de procesos de recolección y su procesamiento. Ha aplicado diversos métodos y herramientas multivariadas, homologación de información y comparabilidad de fuentes en sus investigaciones, así como usa de diversos softwares estadísticos, y ha impartido clases de estadítica aplicada a nivel de licenciatura y posgrado. Es co-coordinadora del Capítulo de CDMX de la iniciativa RLadies.

Mónica Lara Escalante

Doctora en Ciencia Política. Centro de Investigación y Docencia Económicas (CIDE) México.

Semblanza Gerente de Información y Políticas Públicas en Sertech MX, asistente de docencia en FLACSO México y profesora de asignatura de Estadística en la UNAM. Maestra en Gobierno y Asuntos Públicos por la Facultad Latinoamericana de Ciencias Sociales (FLACSO) – Sede México. También se ha desempeñado como Analista de Datos Senior en ThinkData MX; como profesora de asignatura y adjunta de diversos cursos de métodos cuantitativos para

el análisis de políticas públicas en la Universidad Autónoma de San Luis Potosí, FLACSO México, Centro de Investigación y Docencia Económicas (CIDE) y Universidad Nacional Autónoma de México (UNAM). Sus líneas de investigación son los estudios legislativos, análisis de políticas públicas a nivel local, instituciones y partidos políticos en América Latina.

Instalación de R y Rstudio

Introducción a R

https://youtu.be/YkN5urybh2A Video en YouTube

Instalación en OS

https://youtu.be/icWV8jzYOtA Viedo en YouTube

Instalación en PC

https://youtu.be/TNSQikMfgJI Video en YouTube

Ojo

Pronto RStudio se volverá **"posit"**

Chapter 1

Primer acercamiento al uso del programa

1.1 Introducción

En RStudio podemos tener varias ventanas que nos permiten tener más control de nuestro "ambiente", el historial, los "scripts" o códigos que escribimos y por supuesto, tenemos nuestra consola, que también tiene el símbolo ">" con R. Podemos pedir operaciones básicas

```
  2+5
```

```
[1] 7
```

```
  5*3
```

```
[1] 15
```

```
  #Para escribir comentarios y que no los lea como operaciones ponemos el símbolo de gato
  # Lo podemos hacer para un comentario en una línea o la par de una instrucción
  1:5              # Secuencia 1-5
```

```
[1] 1 2 3 4 5
```

```
  seq(1, 10, 0.5)   # Secuencia con incrementos diferentes a 1
```

```
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0
[16]  8.5  9.0  9.5 10.0
```

```
  c('a','b','c')  # Vector con caracteres
```

```
[1] "a" "b" "c"
```

```
  1:7             # Entero
```

```
[1] 1 2 3 4 5 6 7
```

```
  40<80           # Valor logico
```

```
[1] TRUE
```

```
  2+2 == 5        # Valor logico
```

```
[1] FALSE
```

```
  T == TRUE       # T expresion corta de verdadero
```

```
[1] TRUE
```

R es un lenguaje de programación por objetos. Por lo cual vamos a tener objetos a los que se les asigna su contenido. Si usamos una flechita "<-" o "->" le estamos asignando algo al objeto que apunta la felcha.

```
  x <- 24         # Asignacion de valor 24 a la variable x para su uso posterior (OBJETO)
  x/2             # Uso posterior de variable u objeto x
```

```
[1] 12
```

```
  x               # Imprime en pantalla el valor de la variable u objeto
```

```
[1] 24
```

```
  x <- TRUE       # Asigna el valor logico TRUE a la variable x OJO: x toma el ultimo valor
  x
```

```
[1] TRUE
```

1.2 Vectores

Los vectores son uno de los objetos más usados en R.

```r
y <- c(2,4,6)     # Vector numerico
y <- c('Primaria', 'Secundaria') # Vector caracteres
```

Dado que poseen elementos, podemos también observar y hacer operaciones con sus elementos, usando "[ ]" para acceder a ellos

```r
y[2]              # Acceder al segundo valor del vector y
```

```
[1] "Secundaria"
```

```r
y[3] <- 'Preparatoria y más' # Asigna valor a la tercera componente del vector
sex <-1:2         # Asigna a la variable sex los valores 1 y 2
names(sex) <- c("Femenino", "Masculino") # Asigna nombres al vector de elementos sexo
sex[2]            # Segundo elemento del vector sex
```

```
Masculino
        2
```

1.3 Matrices

Las matrices son muy importantes, porque nos permiten hacer operaciones y casi todas nuestras bases de datos tendran un aspecto de matriz.

```r
m <- matrix (nrow=2, ncol=3, 1:6, byrow = TRUE) # Matrices Ejemplo 1
m
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

```r
m <- matrix (nrow=2, ncol=3, 1:6, byrow = FALSE) # Matrices Ejemplo 1
m
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```r
dim(m)
```

```
[1] 2 3
```

```r
attributes(m)
```

```
$dim
[1] 2 3
```

```
n <- 1:6     # Matrices Ejemplo 2
dim(n) <- c(2,3)
n
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
xx <-10:12   # Matrices Ejemplo 3
yy<-14:16
cbind(xx,yy) # Une vectores por Columnas
```

```
     xx yy
[1,] 10 14
[2,] 11 15
[3,] 12 16
```

```
rbind(xx,yy) # Une vectores por Renglones
```

```
   [,1] [,2] [,3]
xx   10   11   12
yy   14   15   16
```

```
mi_matrix<-cbind(xx,yy) # este resultado lo puedo asignar a un objeto
```

1.4 Funciones

Algunas funciones básicas son las siguientes. Vamos a ir viendo más funciones, pero para entender cómo funcionan, haremos unos ejemplos y cómo pedir ayuda sobre ellas.

```
sum (10,20,30)    # Función suma
```

```
[1] 60
```

```
rep('R', times=3) # Repite la letra R el numero de veces que se indica
```

```
[1] "R" "R" "R"
```

```
sqrt(9)           # Raiz cuadrada de 9
```

```
[1] 3
```

1.5 Ayuda

Pedir ayuda es indispensable para aprender a escribir nuestros códigos. A prueba y error, es el mejor sistema para aprender. Podemos usar la función help, example y ?

```
help(sum)         # Ayuda sobre función sum
example(sum)      # Ejemplo de función sum
```

```
sum> ## Pass a vector to sum, and it will add the elements together.
sum> sum(1:5)
[1] 15

sum> ## Pass several numbers to sum, and it also adds the elements.
sum> sum(1, 2, 3, 4, 5)
[1] 15

sum> ## In fact, you can pass vectors into several arguments, and everything gets added.
sum> sum(1:2, 3:5)
[1] 15

sum> ## If there are missing values, the sum is unknown, i.e., also missing, ....
sum> sum(1:5, NA)
[1] NA

sum> ## ... unless  we exclude missing values explicitly:
sum> sum(1:5, NA, na.rm = TRUE)
[1] 15
```

1.6 Mi ambiente

Todos los objetos que hemos declarado hasta ahora son parte de nuestro "ambiente" (environment). Para saber qué está en nuestro ambiente usamos el comando

```
ls()
```

```
[1] "m"         "mi_matrix" "n"         "sex"       "x"         "xx"
[7] "y"         "yy"
```

```
  gc()           # Garbage collection, reporta memoria en uso
```

```
         used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
Ncells  598877 32.0    1303138 69.6         NA  1303138 69.6
Vcells 1112485  8.5    8388608 64.0      16384  1839370 14.1
```

Para borrar todos nuestros objetos, usamos el siguiente comando, que equivale a usar la escobita de la venta de environment

```
  rm(list=ls())  # Borrar objetos actuales
```

1.7 Directorio de trabajo

Es muy útil saber dónde estamos trabajando y donde queremos trabajar. Por eso podemos utilizar los siguientes comandos para saberlo

Ojo, checa, si estás desdes una PC, cómo cambian las " " por"/" o por "\"

```
  getwd()           # Directorio actual
```

```
[1] "/Users/anaescoto/Dropbox/2022/Curso_r_cnvl/coneval"
```

```
  #setwd("C:/Users/anaes/Dropbox/2021/CursoR-posgrado")# Cambio de directorio

  list.files()      # Lista de archivos en ese directorio
```

```
 [1] "Icon\r"              "LICENSE"             "Mi_Exportación.xlsx"
 [4] "P1.html"             "P1.qmd"              "P1.rmarkdown"
 [7] "P2.qmd"              "P3.qmd"              "P4.R"
[10] "P4.qmd"              "P5.qmd"              "README.md"
[13] "_quarto.yml"         "coneval.Rproj"       "datos"
[16] "docs"                "index.html"          "index.qmd"
[19] "instala.html"        "instala.qmd"         "intro1.png"
[22] "site_libs"
```

Checar que esto también se puede hacer desde el menú:

1.8 Proyectos

Pero... a veces preferimos trabajar en proyectos, sobre todo porque nos da más control.

Hay gente que lo dice mejor que yo, como Hadley Wickham: https://es.r4ds.hadley.nz/flujo-de-trabajo-proyectos.html

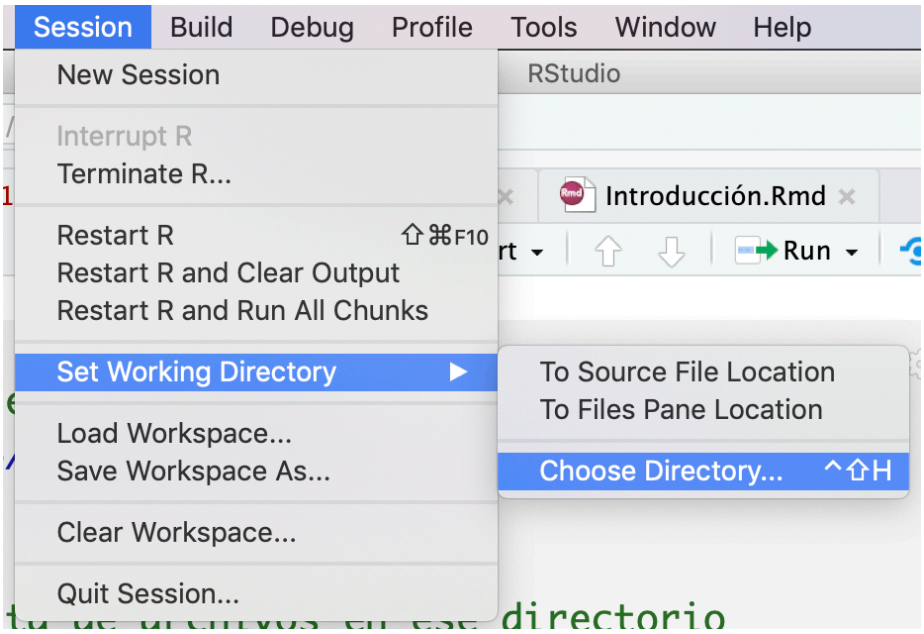


Figure 1.1: i0

1.9 Instalación de paquetes

Los paquetes son útiles para realizar funciones especiales. La especialización de paquetes es más rápida en R que en otros programas por ser un software libre.

Vamos a instalar el paquete "foreign", como su nombre lo indica, nos permite leer elementos "extranjeros" en R. Es sumamente útil porque nos permite leer casi todos los formatos, sin necesidad de usar paquetes especializados como StatTransfer.

Para instalar las paqueterías usamos el siguiente comando "install.packages()" Checa que adentro del paréntesis va el nombre de la librería, con comillas.

Con la opción "dependencies = TRUE" R nos instalará no sólo la librería o paquete que estamos pidiendo, sino todo aquellos paquetes que necesite la librería en cuestión. Muchas veces los diseños de los paquetes implican el uso de algún otro anterior. Por lo que poner esta sentencia nos puede ahorrar errores cuando estemos usando el paquete. Piensa que esto es similar a cuando enciendes tu computadora y tu sistema operativo te pide que mantengas las actualizaciones.

Vamos a instalar dos librerías que nos permiten importar formatos.

```
#install.packages("foreign", dependencies = TRUE)
#install.packages("haven", dependencies = TRUE)
```

Este proceso no hay que hacerlo siempre. Si no sólo la primera vez. Una vez instalado un paquete de librería, la llamamos con el comando "library"

```
library(foreign)
library(haven)
```

"foreing" nos permite leer archivos en formato de dBase, con extensión ".dbf". Si bien no es un formato muy común para los investigadores, sí para los que generan la información, puesto que dBase es uno de los principales programas de administración de bases de datos.

He puesto un ejemplo de una base de datos mexicana en dbf, en este formato.

```
ejemplo_dbf<-read.dbf("datos/ejemplo_dbf.DBF") #checa cómo nos vamos adentro de nuestro d
```

1.10 Paquete pacman

En general, cuando hacemos nuestro código querremos verificar que nuestras librerías estén instaladas. Si actualizamos nuestro R y Rstudio es probable (sobre todo en MAC) que hayamos perdido alguno.

Este es un ejemplo de un código. Y vamos a introducir un paquete muy útil llamado "pacman"

```
if (!require("pacman")) install.packages("pacman") # instala pacman si se requiere
```

```
Loading required package: pacman
```

```
pacman::p_load(tidyverse, readxl, writexl, haven, sjlabelled, foreign) #carga los paquete
```

Hay muchos formatos de almacenamiento de bases de datos. Vamos a aprender a importar información desde ellos.

Chapter 2

Manejo de datos: importación, selección y revisión

2.1 Previo

Vamos a llamar algunas librerías básicas, el tidyverse (que son muchas librerías) y sjlabelled que nos sirve para el manejo de etiquetas

```
if (!require("pacman")) install.packages("pacman") # instala pacman si se requiere
```

```
Loading required package: pacman
```

```
pacman::p_load(tidyverse, haven, sjlabelled, foreign, janitor) #carga los paquetes necesa
```

2.2 Importación de datos

2.2.1 Desde Excel

El paquete más compatible con RStudio es readxl. A veces, otros paquetes tienen más problemas de configuración entre R y el Java.

```
ejemploxl <- readxl::read_excel("datos/ejemplo_xlsx.xlsx", sheet = "para_importar")
```

```
New names:
* `` -> `...128`
* `` -> `...129`
```

```
* `` -> `...132`
* `PIB (Paridad de Poder Adquisitivo)` -> `PIB (Paridad de Poder
  Adquisitivo)...135`
* `PIB (Paridad de Poder Adquisitivo)` -> `PIB (Paridad de Poder
  Adquisitivo)...136`
* `PIB per cápita (Paridad de Poder Adquisitivo)` -> `PIB per cápita (Paridad
  de Poder Adquisitivo)...137`
* `PIB per cápita (Paridad de Poder Adquisitivo)` -> `PIB per cápita (Paridad
  de Poder Adquisitivo)...138`
* `PIB per cápita` -> `PIB per cápita...139`
* `PIB per cápita` -> `PIB per cápita...140`
* `PIB` -> `PIB...141`
* `PIB` -> `PIB...142`
```

Como el nombre de paquete lo indica, sólo lee. Para escribir en este formato, recomiendo el paquete "writexl". Lo instalamos anteriormente.

Si quisiéramos exportar un objeto a Excel

```
writexl::write_xlsx(ejemploxl, path = "Mi_Exportación.xlsx")
```

2.2.2 Desde STATA y SPSS

Si bien también se puede realizar desde el paquete foreign. Pero este no importa algunas características como las etiquetas y tampoco funciona con las versiones más nuevas de STATA. Vamos a instalar otro paquete, compatible con el mundo tidyverse.

Recuerda que no hay que instalarlo (viene adentro de tidyverse). Se instalasólo la primera vez. Una vez instalado un paquete, lo llamamos con el comando "library"

```
concentrado2020 <- haven::read_dta("datos/concentrado2020.dta")
```

!Importante, a R no le gustan los objetos con nombres que empiezan en números

El paquete haven sí exporta información.

```
haven::write_dta(concentrado2020, "datos/mi_exportación.dta", version = 12)
```

Con SSPS es muy parecido. Dentro de "haven" hay una función específica para ello.

```
#encevi_hogar<- haven::read_sav("datos/encevi_hogar.sav")
```

Para escribir

```
#haven::write_sav(concentrado2020 , "mi_exportacion.sav")
```

Checa que en todas las exportaciones en los nombres hay que incluir la extensión del programa. Si quieres guardar en un lugar diferente al directorio del trabajo, hay que escribir toda la ruta dentro de la computadora.

2.3 Revisión de nuestra base

Vamos a revisar la base, brevemente la base

```
class(concentrado2020) # tipo de objeto
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

```
names(concentrado2020) # lista las variables
```

```
  [1] "folioviv"   "foliohog"   "ubica_geo"  "tam_loc"    "est_socio"
  [6] "est_dis"    "upm"        "factor"     "clase_hog"  "sexo_jefe"
 [11] "edad_jefe"  "educa_jefe" "tot_integ"  "hombres"    "mujeres"
 [16] "mayores"    "menores"    "p12_64"     "p65mas"     "ocupados"
 [21] "percep_ing" "perc_ocupa" "ing_cor"    "ingtrab"    "trabajo"
 [26] "sueldos"    "horas_extr" "comisiones" "aguinaldo"  "indemtrab"
 [31] "otra_rem"   "remu_espec" "negocio"    "noagrop"    "industria"
 [36] "comercio"   "servicios"  "agrope"     "agricolas"  "pecuarios"
 [41] "reproducc"  "pesca"      "otros_trab" "rentas"     "utilidad"
 [46] "arrenda"    "transfer"   "jubilacion" "becas"      "donativos"
 [51] "remesas"    "bene_gob"   "transf_hog" "trans_inst" "estim_alqu"
 [56] "otros_ing"  "gasto_mon"  "alimentos"  "ali_dentro" "cereales"
 [61] "carnes"     "pescado"    "leche"      "huevo"      "aceites"
 [66] "tuberculo"  "verduras"   "frutas"     "azucar"     "cafe"
 [71] "especias"   "otros_alim" "bebidas"    "ali_fuera"  "tabaco"
 [76] "vesti_calz" "vestido"    "calzado"    "vivienda"   "alquiler"
 [81] "pred_cons"  "agua"       "energia"    "limpieza"   "cuidados"
 [86] "utensilios" "enseres"    "salud"      "atenc_ambu" "hospital"
 [91] "medicinas"  "transporte" "publico"    "foraneo"    "adqui_vehi"
 [96] "mantenim"   "refaccion"  "combus"     "comunica"   "educa_espa"
[101] "educacion"  "esparci"    "paq_turist" "personales" "cuida_pers"
[106] "acces_pers" "otros_gas"  "transf_gas" "percep_tot" "retiro_inv"
[111] "prestamos"  "otras_perc" "ero_nm_viv" "ero_nm_hog" "erogac_tot"
[116] "cuota_viv"  "mater_serv" "material"   "servicio"   "deposito"
[121] "prest_terc" "pago_tarje" "deudas"     "balance"    "otras_erog"
[126] "smg"
```

```
head(concentrado2020) # muestra las primeras 6 líneas
```

```
# A tibble: 6 x 126
  folioviv  folio~1 ubica~2 tam_loc est_s~3 est_dis upm   factor clase~4 sexo_~5
  <chr>     <chr>   <chr>   <chr>   <chr>   <chr>   <chr>  <dbl> <chr>   <chr>
1 01000136~ 1       01001   1       3       002     0000~    190 2       2
2 01000136~ 1       01001   1       3       002     0000~    190 2       1
3 01000178~ 1       01001   1       3       002     0000~    189 2       1
4 01000178~ 1       01001   1       3       002     0000~    189 2       1
5 01000178~ 1       01001   1       3       002     0000~    189 2       1
6 01000178~ 1       01001   1       3       002     0000~    189 2       1
# ... with 116 more variables: edad_jefe <dbl>, educa_jefe <chr>,
#   tot_integ <dbl>, hombres <dbl>, mujeres <dbl>, mayores <dbl>,
#   menores <dbl>, p12_64 <dbl>, p65mas <dbl>, ocupados <dbl>,
#   percep_ing <dbl>, perc_ocupa <dbl>, ing_cor <dbl>, ingtrab <dbl>,
#   trabajo <dbl>, sueldos <dbl>, horas_extr <dbl>, comisiones <dbl>,
#   aguinaldo <dbl>, indemtrab <dbl>, otra_rem <dbl>, remu_espec <dbl>,
#   negocio <dbl>, noagrop <dbl>, industria <dbl>, comercio <dbl>, ...
```

```
table(concentrado2020$clase_hog) # un tabulado simple
```

```
    1     2     3     4     5
10842 55339 21819   717   289
```

2.4 Revisión con dplyr

Operador de "pipe" o "tubería" %>% (Ctrl+Shift+M) Antes de continuar, presentemos el operador "pipe" %>%. dplyr importa este operador de otro paquete (magrittr). Este operador le permite canalizar la salida de una función a la entrada de otra función. En lugar de funciones de anidamiento (lectura desde adentro hacia afuera), la idea de la tubería es leer las funciones de izquierda a derecha.

```
concentrado2020 %>%
    dplyr::select(sexo_jefe, edad_jefe) %>%
    head
```

```
# A tibble: 6 x 2
  sexo_jefe edad_jefe
  <chr>         <dbl>
1 2                48
2 1                46
3 1                26
```

```
4 1                29
5 1                63
6 1                33
```

```
concentrado2020 %>%
    dplyr::select(sexo_jefe, edad_jefe) %>%
    glimpse
```

```
Rows: 89,006
Columns: 2
$ sexo_jefe <chr> "2", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "2", ~
$ edad_jefe <dbl> 48, 46, 26, 29, 63, 33, 60, 76, 74, 37, 76, 79, 37, 80, 46, ~
```

2.5 Etiquetas y cómo usarlas

Podemos ver que los objetos "data.frame"(*spoiler*, ya hablaremos de ellos)

```
class(concentrado2020$sexo_jefe)
```

```
[1] "character"
```

2.5.1 Ejemplo de etiquetado

Para que se vea mejor nuestro tabulado, sería bueno que nuestras variables tuvieran etiqueta. Para ello utilizaremos el paquete "sjlabelled"

```
etiqueta_sex<-c("Hombre", "Mujer")

concentrado2020<-concentrado2020 %>%
  mutate(sexo_jefe=as_numeric(sexo_jefe)) %>% # para quitar el "string"
  sjlabelled::set_labels(sexo_jefe, labels=etiqueta_sex)
```

Etiquetemos también la variable "clase_hog". Podemos checar cómo está estructurada esta base acá https://www.inegi.org.mx/rnm/index.php/catalog/685/data-dictionary

```
concentrado2020<-concentrado2020 %>%
  mutate(clase_hog=as_numeric(clase_hog)) %>% # para quitar el "string"
  sjlabelled::set_labels(clase_hog, labels=c("unipersonal",
                                             "nuclear",
                                             "ampliado",
                                             "compuesto",
                                             "corresidente"))
```

```
table(concentrado2020$sexo_jefe)
```

```
    1     2
63230 25776
```

```
table(sjlabelled::as_label(concentrado2020$sexo_jefe))
```

```
Hombre  Mujer
 63230  25776
```

2.5.2 Ojeando

```
dplyr::glimpse(concentrado2020)
```

```
Rows: 89,006
Columns: 126
$ folioviv   <chr> "0100013605", "0100013606", "0100017801", "0100017802", "01~
$ foliohog   <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1",~
$ ubica_geo  <chr> "01001", "01001", "01001", "01001", "01001", "01001", "0100~
$ tam_loc    <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1",~
$ est_socio  <chr> "3", "3", "3", "3", "3", "3", "3", "3", "3", "3", "3", "3",~
$ est_dis    <chr> "002", "002", "002", "002", "002", "002", "002", "002", "00~
$ upm        <chr> "0000001", "0000001", "0000002", "0000002", "0000002", "000~
$ factor     <dbl> 190, 190, 189, 189, 189, 189, 189, 168, 168, 168, 168, 168,~
$ clase_hog  <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 1, 1, 1, 3, 3, 2, 3, 5, 2,~
$ sexo_jefe  <dbl> 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 1, 1, 2, 1, 1,~
$ edad_jefe  <dbl> 48, 46, 26, 29, 63, 33, 60, 76, 74, 37, 76, 79, 37, 80, 46,~
$ educa_jefe <chr> "09", "08", "10", "08", "10", "06", "03", "08", "03", "06",~
$ tot_integ  <dbl> 3, 4, 2, 2, 2, 4, 3, 2, 2, 6, 6, 1, 1, 1, 2, 3, 3, 2, 2, 5,~
$ hombres    <dbl> 1, 3, 1, 2, 1, 2, 2, 1, 1, 3, 4, 0, 1, 0, 0, 2, 1, 1, 2, 3,~
$ mujeres    <dbl> 2, 1, 1, 0, 1, 2, 1, 1, 1, 3, 2, 1, 0, 1, 2, 1, 2, 1, 0, 2,~
$ mayores    <dbl> 3, 3, 2, 1, 2, 2, 3, 2, 2, 3, 6, 1, 1, 1, 2, 3, 2, 2, 2, 5,~
$ menores    <dbl> 0, 1, 0, 1, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,~
$ p12_64     <dbl> 3, 3, 2, 1, 2, 2, 3, 0, 0, 3, 4, 0, 1, 0, 2, 1, 2, 1, 2, 5,~
$ p65mas     <dbl> 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 2, 1, 0, 1, 0, 2, 0, 1, 0, 0,~
$ ocupados   <dbl> 1, 1, 2, 1, 1, 1, 1, 0, 1, 3, 1, 1, 1, 0, 2, 0, 1, 1, 2, 2,~
$ percep_ing <dbl> 2, 2, 2, 1, 1, 1, 2, 1, 2, 2, 5, 1, 1, 1, 1, 2, 1, 2, 2, 3,~
$ perc_ocupa <dbl> 1, 1, 2, 1, 1, 1, 1, 0, 1, 2, 1, 1, 1, 0, 1, 0, 1, 1, 2, 2,~
$ ing_cor    <dbl> 16229.49, 31425.68, 33979.16, 71557.37, 90703.26, 30368.84,~
$ ingtrab    <dbl> 13278.68, 22254.09, 33979.16, 71557.37, 48113.11, 30368.84,~
$ trabajo    <dbl> 0.00, 22254.09, 24098.35, 71557.37, 48113.11, 30368.84, 148~
$ sueldos    <dbl> 0.00, 21639.34, 23606.55, 67868.85, 47213.11, 29508.19, 140~
```

```
$ horas_extr <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ comisiones <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ aguinaldo  <dbl> 0.00, 614.75, 491.80, 3688.52, 0.00, 860.65, 737.70, 0.00, ~
$ indemtrab  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ otra_rem   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ remu_espec <dbl> 0.00, 0.00, 0.00, 0.00, 900.00, 0.00, 0.00, 0.00, 0.00, 0.0~
$ negocio    <dbl> 1573.77, 0.00, 9880.81, 0.00, 0.00, 0.00, 0.00, 0.00, 8659.~
$ noagrop    <dbl> 1573.77, 0.00, 9880.81, 0.00, 0.00, 0.00, 0.00, 0.00, 8659.~
$ industria  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ comercio   <dbl> 1573.77, 0.00, 9880.81, 0.00, 0.00, 0.00, 0.00, 0.00, 8659.~
$ servicios  <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ agrope     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ agricolas  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ pecuarios  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ reproducc  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ pesca      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ otros_trab <dbl> 11704.91, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0~
$ rentas     <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 15497~
$ utilidad   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 154979, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ arrenda    <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ transfer   <dbl> 2459.01, 1671.59, 0.00, 0.00, 22131.14, 0.00, 25967.21, 130~
$ jubilacion <dbl> 0.00, 0.00, 0.00, 0.00, 22131.14, 0.00, 25967.21, 7336.95, ~
$ becas      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ donativos  <dbl> 885.24, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 29.34, 0.~
$ remesas    <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ bene_gob   <dbl> 1573.77, 1573.77, 0.00, 0.00, 0.00, 0.00, 0.00, 5086.95, 10~
$ transf_hog <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 606.51, 0.00, 0.0~
$ trans_inst <dbl> 0.00, 97.82, 0.00, 0.00, 0.00, 0.00, 0.00, 58.69, 0.00, 0.0~
$ estim_alqu <dbl> 0.00, 7500.00, 0.00, 0.00, 18000.00, 0.00, 12000.00, 11612.~
$ otros_ing  <dbl> 491.80, 0.00, 0.00, 0.00, 2459.01, 0.00, 0.00, 0.00, 0.00, ~
$ gasto_mon  <dbl> 24626.04, 20397.10, 44955.73, 82950.42, 30140.68, 39991.94,~
$ alimentos  <dbl> 14732.80, 9321.32, 15081.32, 26921.53, 11969.93, 7547.03, 1~
$ ali_dentro <dbl> 13549.96, 9321.32, 9295.63, 22164.39, 3355.69, 7547.03, 112~
$ cereales   <dbl> 3990.78, 1324.26, 1594.26, 2441.54, 0.00, 1529.96, 1259.98,~
$ carnes     <dbl> 989.99, 3882.84, 0.00, 4513.33, 3034.27, 4204.25, 2031.41, ~
$ pescado    <dbl> 0.00, 0.00, 0.00, 1025.87, 0.00, 0.00, 771.42, 0.00, 0.00, ~
$ leche      <dbl> 1613.54, 925.71, 0.00, 449.99, 321.42, 321.42, 2494.25, 707~
$ huevo      <dbl> 822.85, 745.70, 925.71, 0.00, 0.00, 244.28, 642.85, 0.00, 0~
$ aceites    <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 1067.13, 0.00, 0.00, 0.00, 0.~
$ tuberculo  <dbl> 347.14, 0.00, 0.00, 197.48, 0.00, 0.00, 0.00, 411.42, 0.00,~
$ verduras   <dbl> 655.70, 1157.10, 385.71, 2413.26, 0.00, 0.00, 1896.37, 3439~
$ frutas     <dbl> 0.00, 0.00, 0.00, 1367.85, 0.00, 0.00, 642.85, 1504.25, 0.0~
$ azucar     <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ cafe       <dbl> 925.71, 0.00, 0.00, 86.52, 0.00, 0.00, 0.00, 77.14, 0.00, 0~
$ especias   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ otros_alim <dbl> 3304.26, 1285.71, 3278.56, 9668.55, 0.00, 179.99, 1542.85, ~
```

```
$ bebidas    <dbl> 899.99, 0.00, 3111.39, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, ~
$ ali_fuera  <dbl> 1182.84, 0.00, 5785.69, 4757.14, 8614.24, 0.00, 0.00, 0.00,~
$ tabaco     <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ vesti_calz <dbl> 0.00, 0.00, 1006.60, 4509.73, 0.00, 371.73, 0.00, 215.21, 0~
$ vestido    <dbl> 0.00, 0.00, 1006.60, 4294.52, 0.00, 0.00, 0.00, 215.21, 0.0~
$ calzado    <dbl> 0.00, 0.00, 0.00, 215.21, 0.00, 371.73, 0.00, 0.00, 0.00, 0~
$ vivienda   <dbl> 2850.00, 2308.50, 11097.00, 13984.50, 3179.50, 12450.00, 34~
$ alquiler   <dbl> 0.00, 0.00, 9000.00, 12000.00, 0.00, 10500.00, 0.00, 0.00, ~
$ pred_cons  <dbl> 0.0, 0.0, 0.0, 0.0, 212.5, 0.0, 300.0, 100.0, 100.0, 150.0,~
$ agua       <dbl> 750.00, 990.00, 420.00, 756.00, 408.00, 1500.00, 600.00, 39~
$ energia    <dbl> 2100.00, 1318.50, 1677.00, 1228.50, 2559.00, 450.00, 2550.0~
$ limpieza   <dbl> 375.00, 924.00, 2530.16, 708.00, 920.80, 408.00, 845.73, 72~
$ cuidados   <dbl> 375.00, 924.00, 2403.00, 708.00, 429.00, 408.00, 699.00, 72~
$ utensilios <dbl> 0.00, 0.00, 39.13, 0.00, 0.00, 0.00, 146.73, 0.00, 0.00, 0.~
$ enseres    <dbl> 0.00, 0.00, 88.03, 0.00, 491.80, 0.00, 0.00, 0.00, 0.00, 0.~
$ salud      <dbl> 0.00, 782.60, 4509.77, 39.13, 2412.39, 229.87, 213.25, 309.~
$ atenc_ambu <dbl> 0.00, 782.60, 3913.04, 0.00, 0.00, 229.87, 0.00, 309.12, 0.~
$ hospital   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ medicinas  <dbl> 0.00, 0.00, 596.73, 39.13, 2412.39, 0.00, 213.25, 0.00, 426~
$ transporte <dbl> 5447.24, 4915.68, 7029.68, 7022.39, 7154.75, 16171.31, 4200~
$ publico    <dbl> 1812.82, 1465.68, 514.28, 899.99, 0.00, 1594.27, 0.00, 1092~
$ foraneo    <dbl> 634.42, 0.00, 1475.40, 1475.40, 0.00, 0.00, 0.00, 0.00, 0.0~
$ adqui_vehi <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 7377.04, 0.00, 0.00, 0.00, 0.~
$ mantenim   <dbl> 0.00, 1200.00, 3000.00, 0.00, 6014.75, 1950.00, 3000.00, 11~
$ refaccion  <dbl> 0.00, 0.00, 0.00, 0.00, 2114.75, 0.00, 0.00, 538.04, 0.00, ~
$ combus     <dbl> 0.00, 1200.00, 3000.00, 0.00, 3900.00, 1950.00, 3000.00, 58~
$ comunica   <dbl> 3000.00, 2250.00, 2040.00, 4647.00, 1140.00, 5250.00, 1200.~
$ educa_espa <dbl> 120.00, 0.00, 693.44, 26408.75, 1440.00, 1035.00, 0.00, 0.0~
$ educacion  <dbl> 120.00, 0.00, 0.00, 7650.00, 0.00, 1035.00, 0.00, 0.00, 0.0~
$ esparci    <dbl> 0.00, 0.00, 693.44, 13840.72, 1440.00, 0.00, 0.00, 0.00, 0.~
$ paq_turist <dbl> 0.00, 0.00, 0.00, 4918.03, 0.00, 0.00, 0.00, 0.00, 0.00, 0.~
$ personales <dbl> 1101.00, 2145.00, 2766.78, 2767.30, 112.50, 1779.00, 521.50~
$ cuida_pers <dbl> 1101.00, 2145.00, 2082.00, 2601.00, 0.00, 1029.00, 384.00, ~
$ acces_pers <dbl> 0.00, 0.00, 684.78, 166.30, 0.00, 0.00, 0.00, 0.00, 0.00, 1~
$ otros_gas  <dbl> 0.00, 0.00, 0.00, 0.00, 112.50, 750.00, 137.50, 125.00, 0.0~
$ transf_gas <dbl> 0.00, 0.00, 240.98, 589.09, 2950.81, 0.00, 0.00, 386.40, 0.~
$ percep_tot <dbl> 0.00, 2571.42, 6014.03, 1799.99, 4885.71, 5528.56, 0.00, 22~
$ retiro_inv <dbl> 0.00, 0.00, 3442.61, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.~
$ prestamos  <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ otras_perc <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 48.91, 0.00, 2445~
$ ero_nm_viv <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ ero_nm_hog <dbl> 0.00, 2571.42, 2571.42, 1799.99, 4885.71, 5528.56, 0.00, 22~
$ erogac_tot <dbl> 0.00, 2360.65, 1062.28, 885.24, 5901.63, 0.00, 0.00, 0.00, ~
$ cuota_viv  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ mater_serv <dbl> 0.00, 0.00, 78.68, 0.00, 0.00, 0.00, 0.00, 0.00, 29.34, 0.0~
$ material   <dbl> 0.00, 0.00, 78.68, 0.00, 0.00, 0.00, 0.00, 0.00, 29.34, 0.0~
```

```
$ servicio   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ deposito   <dbl> 0.00, 0.00, 983.60, 0.00, 5901.63, 0.00, 0.00, 0.00, 0.00, ~
$ prest_terc <dbl> 0.00, 0.00, 0.00, 885.24, 0.00, 0.00, 0.00, 0.00, 0.00, 0.0~
$ pago_tarje <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,~
$ deudas     <dbl> 0.00, 2360.65, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.~
$ balance    <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ otras_erog <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ smg        <dbl> 11089.8, 11089.8, 11089.8, 11089.8, 11089.8, 11089.8, 11089~
```

```
dplyr::glimpse(concentrado2020[,20:30]) # en corchete del lado derecho podemos ojear colu
```

```
Rows: 89,006
Columns: 11
$ ocupados   <dbl> 1, 1, 2, 1, 1, 1, 1, 0, 1, 3, 1, 1, 1, 0, 2, 0, 1, 1, 2, 2,~
$ percep_ing <dbl> 2, 2, 2, 1, 1, 1, 2, 1, 2, 2, 5, 1, 1, 1, 1, 2, 1, 2, 2, 3,~
$ perc_ocupa <dbl> 1, 1, 2, 1, 1, 1, 1, 0, 1, 2, 1, 1, 1, 0, 1, 0, 1, 1, 2, 2,~
$ ing_cor    <dbl> 16229.49, 31425.68, 33979.16, 71557.37, 90703.26, 30368.84,~
$ ingtrab    <dbl> 13278.68, 22254.09, 33979.16, 71557.37, 48113.11, 30368.84,~
$ trabajo    <dbl> 0.00, 22254.09, 24098.35, 71557.37, 48113.11, 30368.84, 148~
$ sueldos    <dbl> 0.00, 21639.34, 23606.55, 67868.85, 47213.11, 29508.19, 140~
$ horas_extr <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ comisiones <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,~
$ aguinaldo  <dbl> 0.00, 614.75, 491.80, 3688.52, 0.00, 860.65, 737.70, 0.00, ~
$ indemtrab  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

Podemos hacer un tipo "labelbook", usando una función que viene de la librería "sjlabelled", "get_labels". Funciona para toda la base o para columnas, o para variables.

```
#print(get_labels(concentrado2020)) #todas
print(get_labels(concentrado2020[, 20:30])) #de las segundas 10 variables
```

```
$ocupados
NULL

$percep_ing
NULL

$perc_ocupa
NULL

$ing_cor
NULL

$ingtrab
NULL
```

```
$trabajo
NULL

$sueldos
NULL

$horas_extr
NULL

$comisiones
NULL

$aguinaldo
NULL

$indemtrab
NULL
```

No tienen :(

En singular nos da las etiquetas de las variables, no de los valores:

```
  #print(get_label(concentrado2020)) #todas
  print(get_label(concentrado2020[, 1:10])) #de las primeras 10 variables
```

```
                     folioviv                      foliohog
"Identificador de la vivienda"     "Identificador del hogar"
                     ubica_geo                      tam_loc
       "Ubicación geográfica"         "Tamaño de localidad"
                     est_socio                      est_dis
     "Estrato socioeconómico"  "Estrato de diseño muestral"
                           upm                       factor
 "Unidad primaria de muestreo"         "Factor de expansión"
                     clase_hog                    sexo_jefe
             "Clase de hogar"     "Sexo del jefe del hogar"
```

```
  print(get_label(concentrado2020$clase_hog)) #
```

```
[1] "Clase de hogar"
```

2.5.3 Selección de casos y de variables

Poco a poco vamos comprendiendo más la lógica de R. Hay varias "formas" de programar. Por lo que no te asustes si varios códigos llegan al mismo resultado

Para revisar el contenido de un data frame podemos usar, como lo hicimos

anteriormente, el formato basededatos$var o usar corchete, checa como estas cuatro formas tan el mismo resultado.

```
x<-concentrado2020$ing_cor
x<-concentrado2020[["ing_cor"]]  # ¡Ojo con las comillas!
x<-concentrado2020[,23]
x<-concentrado2020[,"ing_cor"]
```

Ahora, con el formato de dplyr podemos llegar a lo mismo

```
x<-concentrado2020 %>%
  select(ing_cor)
```

2.6 "Subsetting"

Selección "inversa" O sea no "botar algo", es con el negativo. No funciona con todos los formatos

```
x<-concentrado2020 %>%
  select(-ing_cor)

rm(x) #rm sólo bota objetos
```

Pero con los otros formatos podemos "asignar" valores adentro de un data.frame, y uno de eso valores puede ser "la nada""

```
concentrado2020$aproba2<-concentrado2020$ing_cor
concentrado2020$aproba2<-NULL
```

De aquí viene esa cuesta en el aprendizaje; tenemos que comprender en qué forma programó el que hizo la librería e incluso a veces cómo aprendió quién te está enseñando o el foro que estás leyendo.

Rara vez utilizamos una base de datos completa, y rara vez queremos hacer operaciones completas con ellas.

Vamos a pedir cosas más específicas y podemos seleccionar observaciones o filas. Como nuestra base de datos es muy grande, guardaremos el filtro o selección en un objeto.

```
subset1<-concentrado2020[concentrado2020$ing_cor>5000,]
```

También podemos seleccionar columnas

```
subset2<- concentrado2020[, c("sexo_jefe", "edad_jefe", "ing_cor")]
```

podemos combinar los dos tipos de selección

```
subset3<- concentrado2020[(concentrado2020$ing_cor>5000 & concentrado2020$sexo_jefe==1 ),
```

Con dplyr, podemos usar “filter” y “select”

```
subset4<-concentrado2020 %>%
  dplyr::filter(ing_cor>5000 & sexo_jefe==1) %>%
    dplyr::select(sexo_jefe, edad_jefe, ing_cor)
```

Chapter 3

Análisis descriptivo básico

3.1 Leer desde archivos de texto y desde una url

Desde el portal https://datos.gob.mx/ tenemos acceso a directo a varias fuentes de información, al ser datos abiertos, los archivos de texto son muy comunes.

Leeremos parte de esa información, específicamente la de CONAPO https://datos.gob.mx/busca/dataset/proyecciones-de-la-poblacion-de-mexico-y-de-las-entidades-federativas-2016-2050

En estas bases hay acentos y otros carecteres especiales del español, por lo que agregaremos una opción de "encoding", de lo contrario da error.

```
mig_inter_quin_proyecciones <- read.csv("http://www.conapo.gob.mx/work/models/CONAPO/Datc
#View(mig_inter_quin_proyecciones)
names(mig_inter_quin_proyecciones)
```

```
[1] "RENGLON"     "AÑO"         "ENTIDAD"     "CVE_GEO"     "EDAD"
[6] "SEXO"        "EMIGRANTES"  "INMIGRANTES"
```

3.2 Análisis descriptivo básico

Vamos a llamar algunas librerías básicas, el tidyverse (que son muchas librerías) y sjlabelled que nos sirve para el manejo de etiquetas

```
if (!require("pacman")) install.packages("pacman") # instala pacman si se requiere
```

```
Loading required package: pacman
```

```
pacman::p_load(tidyverse, haven, sjlabelled, foreign, janitor) #carga los paquetes necesa
```

E importamos la base

```
concentrado2020 <- haven::read_dta("datos/concentrado2020.dta")
```

3.3 Variables nominales

La variable nominal "sexo_jefe", se captura con "1" para hombres y con un "2" para mujeres en la base de datos. Podemos establecer una operación de igual y además sumar los casos que cumplan con esta condición:

```
concentrado2020 %>%
  dplyr::count(sexo_jefe==2) # cuentan los casos que cumplen con la condición "sexo_jefe=
```

```
# A tibble: 2 x 2
  `sexo_jefe == 2`     n
  <lgl>            <int>
1 FALSE            63230
2 TRUE             25776
```

Esto es a lo que nos referimos con contar frecuencias. Podemos contar casos que cumplan con una operación de igualdad.

```
concentrado2020 %>%
  with(
    table(sexo_jefe)
    )
```

```
sexo_jefe
    1     2
63230 25776
```

3.3.1 Recordemos nuestro etiquetado

```
etiqueta_sex<-c("Hombre", "Mujer")

concentrado2020<-concentrado2020 %>%
  mutate(sexo_jefe=as_numeric(sexo_jefe)) %>% # para quitar el "string"
  sjlabelled::set_labels(sexo_jefe, labels=etiqueta_sex)
```

```
concentrado2020<-concentrado2020 %>%
  mutate(clase_hog=as_numeric(clase_hog)) %>% # para quitar el "string"
  sjlabelled::set_labels(clase_hog, labels=c("unipersonal",
                                             "nuclear",
                                             "ampliado",
                                             "compuesto",
                                             "corresidente"))
```

Con "tabyl()"" de "janitor""

```
concentrado2020 %>%
  dplyr::mutate(sexo_jefe=as_label(sexo_jefe))  %>%
  janitor::tabyl(sexo_jefe)
```

```
 sexo_jefe     n   percent
    Hombre 63230 0.7104015
     Mujer 25776 0.2895985
```

Para ver que esto es una distribución de frecuencias sería muy útil ver la proporción total, ello se realiza agregando un elemento más en nuestro código con una "tubería":

```
concentrado2020 %>%
  dplyr::mutate(sexo_jefe=as_label(sexo_jefe))  %>%
  janitor::tabyl(sexo_jefe) %>%
  janitor::adorn_totals()
```

```
 sexo_jefe     n   percent
    Hombre 63230 0.7104015
     Mujer 25776 0.2895985
     Total 89006 1.0000000
```

Hoy revisamos algunos tipos de variables

```
class(concentrado2020$sexo_jefe) # variable sin etiqueta
```

```
[1] "numeric"
```

```
class(as_label(concentrado2020$sexo_jefe)) # variable con etiqueta
```

```
[1] "factor"
```

```
class(as_label(concentrado2020$educa_jefe)) # variable ordinal
```

```
[1] "character"
```

```
class(concentrado2020$ing_cor) # variable de intervalo/razón
```

```
[1] "numeric"
```

En general, tendremos variables de factor que podrían ser consideradas como cualitativas y numéricas. Aunque en realidad, R tiene muchas formas de almacenamiento. Como mostramos con el comando "glimpse()" en la práctica anterior, podemos revisar una variable en específico:

```
dplyr::glimpse(concentrado2020$sexo_jefe)
```

```
num [1:89006] 2 1 1 1 1 1 1 1 1 1 ...
- attr(*, "labels")= Named num [1:2] 1 2
 ..- attr(*, "names")= chr [1:2] "Hombre" "Mujer"
- attr(*, "label")= chr "Sexo del jefe del hogar"
```

```
concentrado2020 %>% mutate(sexo_jefe=as_label(sexo_jefe)) %>% # cambia los valores de la
                tabyl(sexo_jefe) %>% # para hacer la tabla
                adorn_totals() %>% # añade totales
                adorn_pct_formatting()  # nos da porcentaje en lugar de proporción
```

```
sexo_jefe     n percent
   Hombre 63230   71.0%
    Mujer 25776   29.0%
    Total 89006  100.0%
```

La tubería o "pipe" %>% nos permite ir agregando elementos de manera sencilla nuestros comandos. En este caso decimos que dentro del objeto haga el cambio, luego la tabla, que le ponga porcentajes y finalmente que nos dé los totales. El total del 100% no aparece, por un elemento propio del programa.

3.4 Variables ordinales

Son variables que dan cuenta de cualidades o condiciones a través de categorías que guardan un orden entre sí.

Vamos a darle una "ojeada" a esta variable

```
glimpse(concentrado2020$educa_jefe)
```

```
chr [1:89006] "09" "08" "10" "08" "10" "06" "03" "08" "03" "06" "03" "03" ...
- attr(*, "label")= chr "Educación formal del jefe del hogar"
- attr(*, "format.stata")= chr "%2s"
```

Etiquetemos también nuestra variable ordinal

```
concentrado2020 <-concentrado2020 %>%
  mutate(educa_jefe=as.numeric(educa_jefe)) %>%
  set_labels(educa_jefe,
             labels=c("Sin instrucción",
                      "Preescolar",
                      "Primaria incompleta",
                      "Primaria completa",
                      "Secundaria incompleta",
                      "Secundaria completa",
                      "Preparatoria incompleta",
                      "Preparatoria completa",
                      "Profesional incompleta",
                      "Profesional completa",
                      "Posgrado"))
```

Hoy hacemos la tabla, con las etiquetas y vemos que se ve más bonita:

```
concentrado2020 %>%
  mutate(educa_jefe=as_label(educa_jefe)) %>%
  tabyl(educa_jefe)
```

```
           educa_jefe     n     percent
       Sin instrucción  6160 0.069208817
            Preescolar    20 0.000224704
    Primaria incompleta 14577 0.163775476
      Primaria completa 15136 0.170055951
  Secundaria incompleta  2974 0.033413478
    Secundaria completa 23865 0.268127991
Preparatoria incompleta  3029 0.034031414
  Preparatoria completa 10550 0.118531335
 Profesional incompleta  2535 0.028481226
   Profesional completa  8474 0.095207065
             Posgrado  1686 0.018942543
```

Para que no nos salgan las categorías sin datos podemos poner una opción dentro del comando "tabyl()"

```
concentrado2020 %>%
  mutate(educa_jefe=as_label(educa_jefe)) %>%
  tabyl(educa_jefe, show_missing_levels=F ) %>% # esta opción elimina los valores con 0
  adorn_totals()
```

```
           educa_jefe     n     percent
       Sin instrucción  6160 0.069208817
            Preescolar    20 0.000224704
```

```
     Primaria incompleta 14577 0.163775476
       Primaria completa 15136 0.170055951
   Secundaria incompleta  2974 0.033413478
     Secundaria completa 23865 0.268127991
 Preparatoria incompleta  3029 0.034031414
   Preparatoria completa 10550 0.118531335
  Profesional incompleta  2535 0.028481226
    Profesional completa  8474 0.095207065
                Posgrado  1686 0.018942543
                   Total 89006 1.000000000
```

3.5 Bivariado cualitativo

3.5.1 Cálculo de frecuencias

Las tablas de doble entrada tiene su nombre porque en las columnas entran los valores de una variable categórica, y en las filas de una segunda. Basicamente es como hacer un conteo de todas las combinaciones posibles entre los valores de una variable con la otra.

Por ejemplo, si quisiéramos combinar las dos variables que ya estudiamos lo podemos hacer, con una tabla de doble entrada:

```
concentrado2020 %>%
  mutate(clase_hog=as_label(clase_hog)) %>%
  mutate(sexo_jefe=as_label(sexo_jefe)) %>% # para que las lea como factor
                tabyl(clase_hog, sexo_jefe, show_missing_levels=F ) %>% # incluimos aquí
                adorn_totals()
```

```
    clase_hog Hombre Mujer
  unipersonal   6010  4832
      nuclear  43151 12188
     ampliado  13410  8409
    compuesto    477   240
 corresidente    182   107
        Total  63230 25776
```

Observamos que en cada celda confluyen los casos que comparten las mismas características:

```
concentrado2020 %>%
    count(clase_hog==1 & sexo_jefe==1) # nos da la segunda celda de la izquierda
```

```
# A tibble: 2 x 2
  `clase_hog == 1 & sexo_jefe == 1`     n
  <lgl>                             <int>
```

```
1 FALSE                                82996
2 TRUE                                  6010
```

3.5.2 Totales y porcentajes

De esta manera se colocan todos los datos. Si observa al poner la función "adorn_totals()" lo agregó como una nueva fila de totales, pero también podemos pedirle que agregue una columna de totales.

```
concentrado2020 %>%
  mutate(clase_hog=as_label(clase_hog)) %>%
  mutate(sexo_jefe=as_label(sexo_jefe)) %>% # para que las lea como factor
  tabyl(clase_hog, sexo_jefe, show_missing_levels=F ) %>% # incluimos aquí dos variables
  adorn_totals("col")
```

```
   clase_hog Hombre Mujer Total
 unipersonal   6010  4832 10842
     nuclear  43151 12188 55339
    ampliado  13410  8409 21819
   compuesto    477   240   717
corresidente    182   107   289
```

O bien agregar los dos, introduciendo en el argumento "c("col", "row")" un vector de caracteres de las dos opciones requeridas:

```
concentrado2020 %>%
  mutate(clase_hog=as_label(clase_hog)) %>%
  mutate(sexo_jefe=as_label(sexo_jefe)) %>% # para que las lea como factor
  tabyl(clase_hog, sexo_jefe, show_missing_levels=F ) %>% # incluimos aquí dos variable
  adorn_totals(c("col", "row"))
```

```
   clase_hog Hombre Mujer Total
 unipersonal   6010  4832 10842
     nuclear  43151 12188 55339
    ampliado  13410  8409 21819
   compuesto    477   240   717
corresidente    182   107   289
       Total  63230 25776 89006
```

Del mismo modo, podemos calcular los porcentajes. Pero los podemos calcular de tres formas. Uno es que lo calculemos para los totales calculados para las filas, para las columnas o para el gran total poblacional.

Para columnas tenemos el siguiente código y los siguientes resultados:

```
concentrado2020 %>%
  mutate(clase_hog=as_label(clase_hog)) %>%
  mutate(sexo_jefe=as_label(sexo_jefe)) %>% # para que las lea como factor
  tabyl(clase_hog, sexo_jefe, show_missing_levels=F ) %>% # incluimos aquí dos variable
  adorn_totals(c("col", "row")) %>%
  adorn_percentages("col") %>% # Divide los valores entre el total de la columna
  adorn_pct_formatting() # lo vuelve porcentaje
```

```
   clase_hog Hombre  Mujer  Total
 unipersonal   9.5%  18.7%  12.2%
     nuclear  68.2%  47.3%  62.2%
    ampliado  21.2%  32.6%  24.5%
   compuesto   0.8%   0.9%   0.8%
corresidente   0.3%   0.4%   0.3%
       Total 100.0% 100.0% 100.0%
```

Cuando se hagan cuadros de distribuciones (que todas sus partes suman 100), los porcentajes pueden ser una gran ayuda para la interpretación, sobre todos cuando se comparar poblaciones de categorías de diferente tamaño. Por lo general, queremos que los cuadros nos den información de donde están los totales y su 100%, de esta manera el lector se puede guiar de porcentaje con respecto a qué está leyendo. En este caso, vemos que el 100% es común en la última fila.

Veamos la diferencia de cómo podemos leer la misma celda, pero hoy, hemos calculado los porcentajes a nivel de fila:

```
concentrado2020 %>%
  mutate(clase_hog=as_label(clase_hog)) %>%
  mutate(sexo_jefe=as_label(sexo_jefe)) %>% # para que las lea como factor
  tabyl(clase_hog, sexo_jefe, show_missing_levels=F ) %>% # incluimos aquí dos variable
  adorn_totals(c("col", "row")) %>%
  adorn_percentages("row") %>% # Divide los valores entre el total de la fila
  adorn_pct_formatting() # lo vuelve porcentaje
```

```
   clase_hog Hombre Mujer  Total
 unipersonal  55.4% 44.6% 100.0%
     nuclear  78.0% 22.0% 100.0%
    ampliado  61.5% 38.5% 100.0%
   compuesto  66.5% 33.5% 100.0%
corresidente  63.0% 37.0% 100.0%
       Total  71.0% 29.0% 100.0%
```

Finalmente, podemos calcular los porcentajes con referencia a la población total en análisis. Es decir la celda en la esquina inferior derecha de nuestra tabla original.

```r
concentrado2020 %>%
  mutate(clase_hog=as_label(clase_hog)) %>%
  mutate(sexo_jefe=as_label(sexo_jefe)) %>% # para que las lea como factor
  tabyl(clase_hog, sexo_jefe, show_missing_levels=F ) %>% # incluimos aquí dos variable
  adorn_totals(c("col", "row")) %>%
  adorn_percentages("all") %>% # Divide los valores entre el total de la población
  adorn_pct_formatting() # lo vuelve porcentaje
```

```
   clase_hog Hombre Mujer  Total
 unipersonal   6.8%  5.4%  12.2%
     nuclear  48.5% 13.7%  62.2%
    ampliado  15.1%  9.4%  24.5%
   compuesto   0.5%  0.3%   0.8%
corresidente   0.2%  0.1%   0.3%
       Total  71.0% 29.0% 100.0%
```

3.6 Descriptivos para variables cuantitativas

Vamos a empezar a revisar los gráficos para variables cuantitativas.

3.6.1 Medidas numéricas básicas

5 números

```r
summary(concentrado2020$ing_cor) ## ingresos
```

```
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
      0    21392    35172    47838    57640 10702107
```

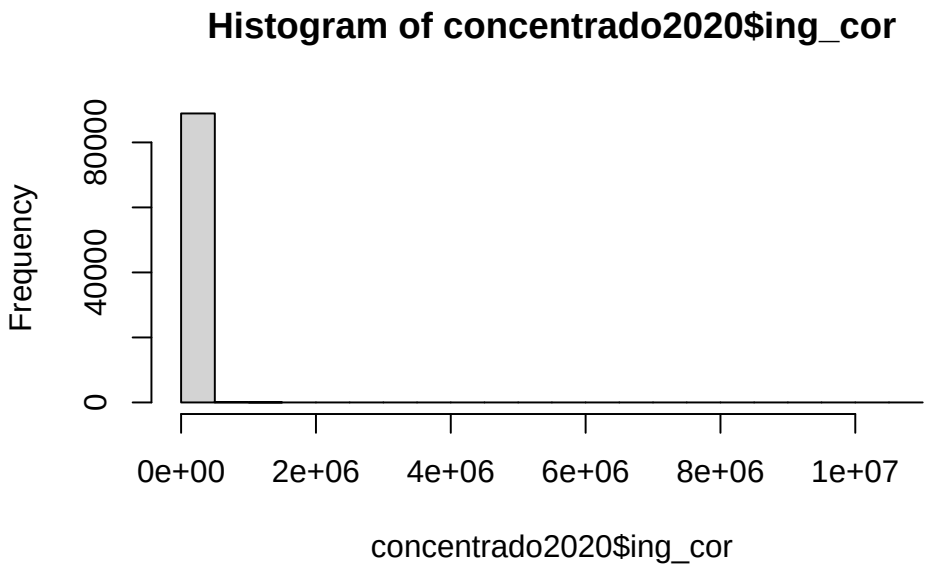
Con pipes se pueden crear "indicadores" de nuestras variables es un tibble

```r
concentrado2020 %>%
  summarise(nombre_indicador=mean(ing_cor, na.rm=T))
```

```
# A tibble: 1 x 1
  nombre_indicador
             <dbl>
1           47838.
```

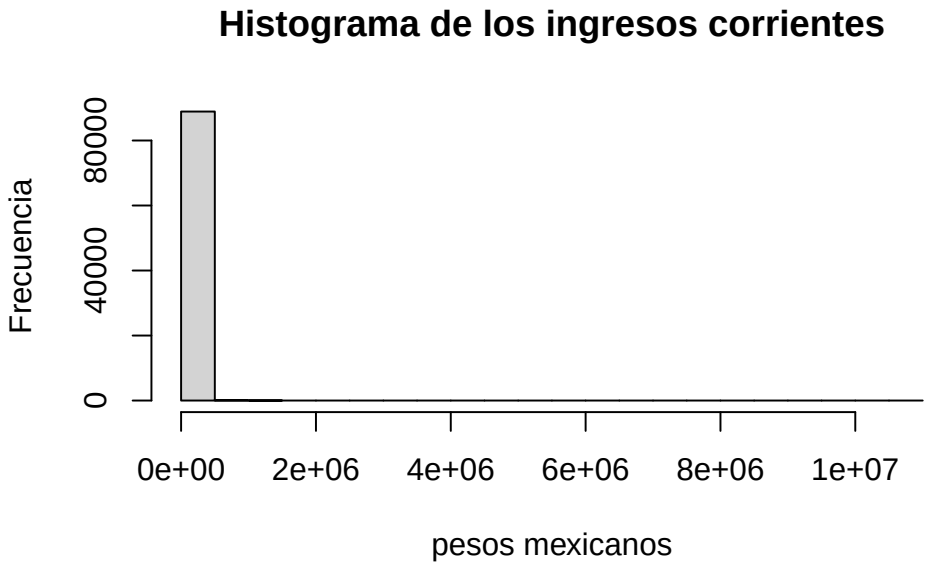
3.6.2 Histograma básico

```r
hist(concentrado2020$ing_cor)
```



Le podemos modificar el título del eje de las x y de las y

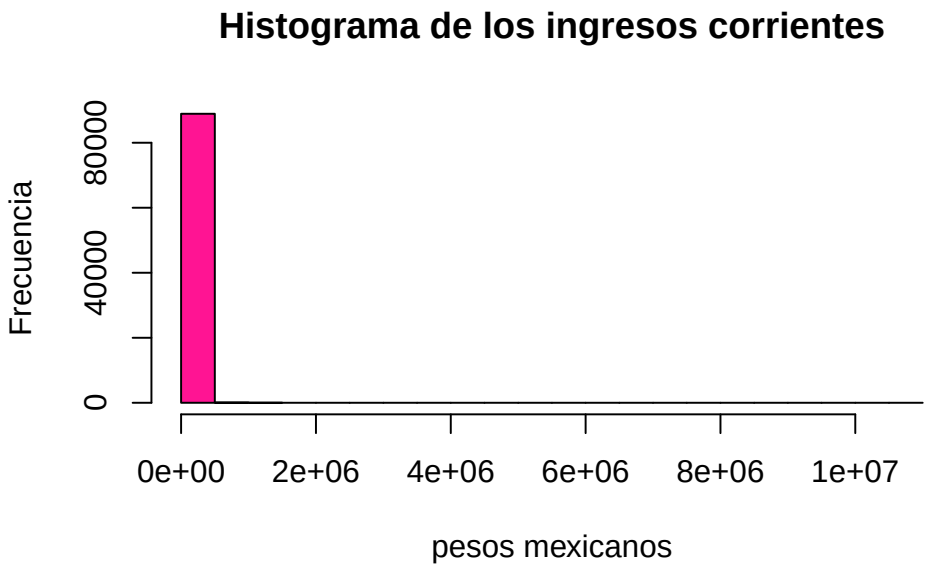
```
hist(concentrado2020$ing_cor,
     main="Histograma de los ingresos corrientes",
     xlab="pesos mexicanos", ylab="Frecuencia")
```



¡A ponerle colorcitos! Aquí hay una lista http://www.stat.columbia.edu/~tzh

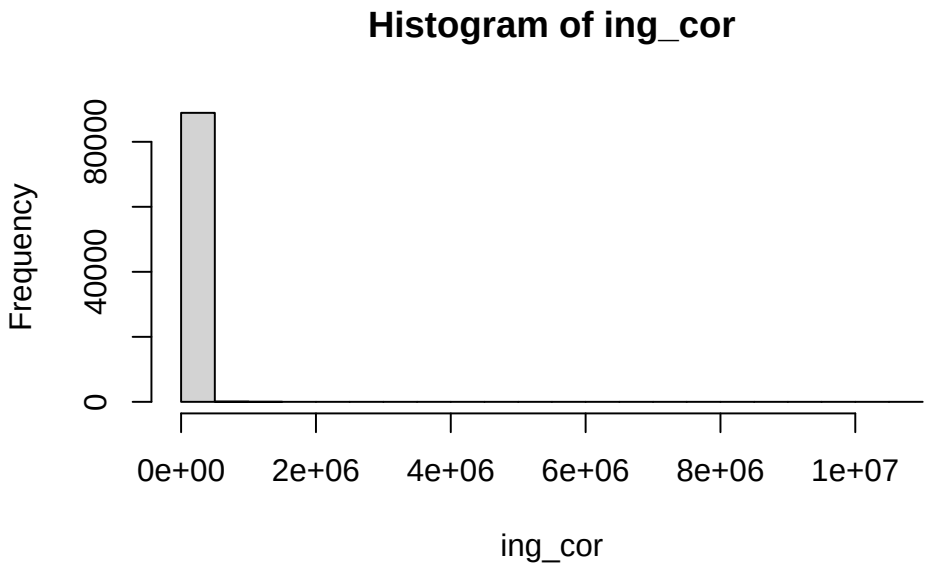
eng/files/Rcolor.pdf

```
hist(concentrado2020$ing_cor,
     main="Histograma de los ingresos corrientes",
     xlab="pesos mexicanos", ylab="Frecuencia",
     col="deeppink1")
```



Con pipes:

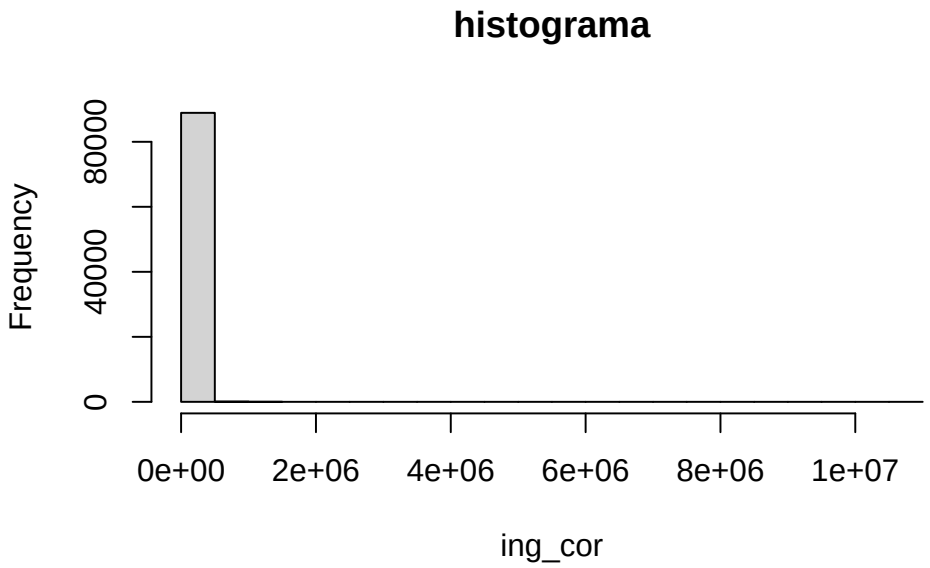
```
concentrado2020 %>%
    with(hist(ing_cor)) # con with, para que entienda
```



Cuando usamos pipes, se debe de recordar que no es necesario escribir el nombre del data.frame en el filtro porque es lo primero que colocamos en nuestro "pipe".

Checa que cualquier aditamiento debe ir en el pipe donde está el comando de hist(). Ten cuidado con los paréntesis.

```
concentrado2020 %>%
  filter(!is.na(ing_cor)) %>% # la ventaja de esta forma es que podemos hacer más operaci
    with(hist(ing_cor, main= "histograma"))
```


histograma
Frequency
0
40000
80000
0e+00
2e+06
4e+06
6e+06
8e+06
1e+07
ing_cor

Chapter 4

Factores de expansión y algunas otras medidas

4.1 Paquetes

```
if (!require("pacman")) install.packages("pacman")#instala pacman si se requiere
```

```
Loading required package: pacman
```

```
pacman::p_load(tidyverse,
               readxl,
               writexl,
               haven,
               sjlabelled,
               janitor,
               magrittr,
               GGally,
               wesanderson,
               gt,
               srvyr,
               dineq
)
```

4.2 Cargando los datos

Desde STATA y haremos unos cambios...

```
concentrado2020 <- read_dta("datos/concentrado2020.dta") %>%
  mutate(across(c(sexo_jefe, clase_hog, educa_jefe), as.numeric)) %>% # ojo aquí
  set_labels(sexo_jefe, labels=c("Hombre", "Mujer")) %>%
  set_labels(clase_hog, labels=c("unipersonal","nuclear", "ampliado",
                                 "compuesto","corresidente"))   %>%
   set_labels(educa_jefe,
              labels=c("Sin instrucción",
                       "Preescolar",
                       "Primaria incompleta",
                       "Primaria completa",
                       "Secundaria incompleta",
                       "Secundaria completa",
                       "Preparatoria incompleta",
                       "Preparatoria completa",
                       "Profesional incompleta",
                       "Profesional completa",
                       "Posgrado"))
```

4.3 La función tally

El comando "tabyl()" del paquete "janitor" es muy útil pero no es compatible con los factores del expansión. En realidad, tabyl() nos ahorra un poco el hecho de tener que agrupar nuestra base en categorías y luego hacer un conteo para cada una de ellas. "tally()" es un comando que nos hace ese conteo y "group_by" nos agrupa las observaciones de nuestra base de datos para hacer cualquier operación.

```
concentrado2020 %>%
 group_by(as_label(sexo_jefe)) %>%
  tally(factor) %>% #nombre del factor
  adorn_totals()  # Agrega total
```

```
as_label(sexo_jefe)        n
             Hombre 25072652
              Mujer 10677007
              Total 35749659
```

Podemos usar funciones de tabyl

```
concentrado2020 %>%
 group_by(as_label(sexo_jefe)) %>%
  tally(factor) %>% #nombre del factor
  adorn_totals() %>% # Agrega total
```

```
    adorn_percentages("all")  %>%
    adorn_pct_formatting()
```

```
 as_label(sexo_jefe)      n
              Hombre  70.1%
               Mujer  29.9%
               Total 100.0%
```

4.4 Otras formas

La función "count()" también permite dar pesos

```
  concentrado2020 %>%
    count(sexo_jefe, clase_hog,  wt = factor)
```

```
# A tibble: 10 x 3
   sexo_jefe clase_hog        n
       <dbl>     <dbl>    <dbl>
 1         1         1  2288234
 2         1         2 17103678
 3         1         3  5408464
 4         1         4   179580
 5         1         5    92696
 6         2         1  1944813
 7         2         2  4989763
 8         2         3  3591323
 9         2         4    98773
10         2         5    52335
```

Es compatible con etiquetas

```
  concentrado2020 %>%
    count(as_label(sexo_jefe), as_label(clase_hog),  wt = factor)
```

```
# A tibble: 10 x 3
   `as_label(sexo_jefe)` `as_label(clase_hog)`        n
   <fct>                 <fct>                    <dbl>
 1 Hombre                unipersonal            2288234
 2 Hombre                nuclear               17103678
 3 Hombre                ampliado               5408464
 4 Hombre                compuesto               179580
 5 Hombre                corresidente             92696
 6 Mujer                 unipersonal            1944813
 7 Mujer                 nuclear                4989763
 8 Mujer                 ampliado               3591323
```

```
 9 Mujer                 compuesto               98773
10 Mujer                 corresidente            52335
```

Podemos mover un poquito con pivot_wider para que se vea más a lo que acostumbramos a una tabla de frecuencias

```
concentrado2020 %>%
  mutate_at(vars(sexo_jefe, clase_hog), as_label) %>%
  count(sexo_jefe, clase_hog,  wt = factor) %>%
  tidyr::pivot_wider(names_from = sexo_jefe,
              values_from = n)
```

```
# A tibble: 5 x 3
  clase_hog      Hombre   Mujer
  <fct>           <dbl>   <dbl>
1 unipersonal   2288234 1944813
2 nuclear      17103678 4989763
3 ampliado      5408464 3591323
4 compuesto      179580   98773
5 corresidente    92696   52335
```

```
concentrado2020 %>%
  mutate_at(vars(sexo_jefe, clase_hog), as_label) %>% # otra forma de mutate y as_label
  count(sexo_jefe, clase_hog,  wt = factor) %>%
  pivot_wider(names_from = sexo_jefe,
              values_from = n) %>%
  adorn_totals() %>% # Agrega total
  adorn_percentages("col")  %>%
  adorn_pct_formatting()
```

```
   clase_hog Hombre  Mujer
 unipersonal   9.1%  18.2%
     nuclear  68.2%  46.7%
    ampliado  21.6%  33.6%
   compuesto   0.7%   0.9%
corresidente   0.4%   0.5%
       Total 100.0% 100.0%
```

4.5 Diseño complejo

Hay muchos diseños muestrales, asumiremos el diseño simple, pero hay que revisar la documentación de la base

```
# Muestreo aleatorio
ags_srvy <- concentrado2020 %>%
  as_survey_design(weights = factor)
```

Si revisamos las encuestas tiene un diseño complejo, hay estratos y unidades primarias de muestreo

```
# Muestreo estratificado
ags_srvy <- concentrado2020 %>%
  as_survey_design(
    upm = upm,
    strata = est_dis,
    weights = factor,
    nest = TRUE)
```

Como vemos esto es un archivo bien grande, por lo que mejor vamos a seleccionar un par de variables:

```
# simple random sample
ags_srvy <- concentrado2020 %>%
  select(upm, est_dis, factor, clase_hog,
         sexo_jefe, edad_jefe, educa_jefe,ing_cor, factor) %>%
  as_survey_design(
    upm=upm,
    strata = est_dis,
    weights = factor,
    nest = TRUE)
```

Para una media ponderada

```
ags_srvy %>%
  filter(ing_cor>0) %>% # sólo con ingresos
  summarise(
    media_ponderada = survey_mean(ing_cor, na.rm=T))
```

```
# A tibble: 1 x 2
  media_ponderada media_ponderada_se
            <dbl>               <dbl>
1          50315.                341.
```

Si queremos los intervalos de confianza (*spoiler*):

```
ags_srvy %>%
  summarize(
```

```
      media_ponderada = survey_mean(ing_cor,
                                    vartype = "ci") )
```

```
# A tibble: 1 x 3
  media_ponderada media_ponderada_low media_ponderada_upp
            <dbl>               <dbl>               <dbl>
1          50309.              49640.              50979.
```

```
  ags_srvy %>%
    summarize(
      mediana_ponderada = survey_median(ing_cor,
                                    vartype = "ci") )
```

```
# A tibble: 1 x 3
  mediana_ponderada mediana_ponderada_low mediana_ponderada_upp
              <dbl>                 <dbl>                 <dbl>
1            36624.                36365.                36882.
```

```
  ags_srvy %>%
    mutate(sexo_jefe=as_label(sexo_jefe)) %>%
    group_by(sexo_jefe) %>% #variables cuali
    summarize(proportion = survey_mean(), # proporción
              total = survey_total() ) # totales
```

```
# A tibble: 2 x 5
  sexo_jefe proportion proportion_se    total total_se
  <fct>          <dbl>         <dbl>    <dbl>    <dbl>
1 Hombre         0.701       0.00217 25072652   80320.
2 Mujer          0.299       0.00217 10677007   77840.
```

4.6 Creación de quintiles y otros grupos

Uno de los elementos más comunes es crear grupos. Por ejemplo, la función cut, nos ayuda a crear variables con ciertos cortes. Por ejemplo, para recodificar por grupos etarios

```
  concentrado2020 %<>%
    mutate(grupo=cut(edad_jefe,
                     breaks=c(0, 25, 50, 75, 100)))


  concentrado2020 %>%
    tabyl(grupo)
```

```
   grupo     n      percent valid_percent
  (0,25]  3327 0.0373795025    0.03738328
 (25,50] 42558 0.4781475406    0.47819589
 (50,75] 36085 0.4054221064    0.40546311
(75,100]  7027 0.0789497337    0.07895772
    <NA>     9 0.0001011168            NA
```

Algunas opciones se pueden modificar dentro de la función cut

```
concentrado2020 %<>%
  mutate(grupo=cut(edad_jefe,
                   breaks=c(0, 25, 50, 75, 100),
                   include.lowest=T,
                   right= F))


concentrado2020 %>%
  tabyl(grupo)
```

```
   grupo     n      percent valid_percent
  [0,25)  2502 0.0281104645    0.02811331
 [25,50) 41068 0.4614070962    0.46145376
 [50,75) 37488 0.4211850886    0.42122768
[75,100]  7939 0.0891962340    0.08920525
    <NA>     9 0.0001011168            NA
```

Esto nos puede ayudar para hacer variables de rangos de cualquier tipo.

Otro tipo de variables muy importante son los quintiles y demás.

```
concentrado2020 %<>%
  mutate(quintil0=ntile(ing_cor, n=5))

concentrado2020 %>%
  tabyl(quintil0)
```

```
quintil0     n   percent
       1 17802 0.2000090
       2 17801 0.1999978
       3 17801 0.1999978
       4 17801 0.1999978
       5 17801 0.1999978
```

Pero quizás nos interesa más los quintiles que toman en cuenta el factor de expansión

```
concentrado2020 %<>%
  mutate(quintil1=dineq::ntiles.wtd(ing_cor, n=5, weights=factor))

concentrado2020 %>%
  tabyl(quintil1)
```

```
quintil1     n   percent
       1 19133 0.2149630
       2 18253 0.2050761
       3 17803 0.2000202
       4 17609 0.1978406
       5 16208 0.1821001
```

```
concentrado2020 %>%
  count(quintil1, wt=factor) %>%
  mutate(p=n/sum(n)*100) %>%
  adorn_totals()
```

```
quintil1        n         p
       1  7150004  20.00020
       2  7150151  20.00061
       3  7149344  19.99836
       4  7150470  20.00151
       5  7149690  19.99932
   Total 35749659 100.00000
```

Podemos también ver la diferencia en los máximos y minimos de ambas variables

```
concentrado2020 %>%
  group_by(quintil0) %>%
  summarise(min=min(ing_cor),
            max=max(ing_cor))
```

```
# A tibble: 5 x 3
  quintil0    min       max
     <int>  <dbl>     <dbl>
1        1     0     18934.
2        2 18935.    29188.
3        3 29188.    42257.
4        4 42257.    65267.
5        5 65268. 10702107.
```

Veamos con la ponderación:

```
  concentrado2020 %>%
    group_by(quintil1) %>%
    summarise(min=min(ing_cor),
              max=max(ing_cor))
```

```
# A tibble: 5 x 3
  quintil1    min       max
     <dbl>  <dbl>     <dbl>
1        1     0     19666.
2        2 19668.    30326.
3        3 30326.    44017.
4        4 44017.    68533.
5        5 68534. 10702107.
```

La flexibilidad de dplyr nos permite además hacer quintiles fácilmente adentro de grupos. Por ejemplo si quisiéramos hacer quintiles estatales… Claro para eso debemos tener la variable.

La variable "ubica_geo", nos da esa información pero junta

```
  concentrado2020 %>%
    select(ubica_geo) %>%
    head
```

```
# A tibble: 6 x 1
  ubica_geo
  <chr>
1 01001
2 01001
3 01001
4 01001
5 01001
6 01001
```

Vamos a crear dos variables, uan que nos diga la entidad y la otra el municipio

```
  concentrado2020 %<>%
    mutate(ent=stringr::str_sub(ubica_geo, start = 1, end = 2)) %>%
    mutate(mun=stringr::str_sub(ubica_geo, start = 3, end = 5))

  concentrado2020 %>% tabyl(ent)
```

```
 ent    n    percent
  01 2669 0.02998674
  02 4142 0.04653619
  03 2717 0.03052603
```

```
 04 2174 0.02442532
 05 3922 0.04406445
 06 3282 0.03687392
 07 2123 0.02385232
 08 4572 0.05136732
 09 2570 0.02887446
 10 2746 0.03085185
 11 3083 0.03463811
 12 2490 0.02797564
 13 2213 0.02486349
 14 2779 0.03122261
 15 3568 0.04008719
 16 2047 0.02299845
 17 2564 0.02880705
 18 2103 0.02362762
 19 3502 0.03934566
 20 2596 0.02916657
 21 2141 0.02405456
 22 3769 0.04234546
 23 2196 0.02467249
 24 2521 0.02832393
 25 3429 0.03852549
 26 2420 0.02718918
 27 2088 0.02345909
 28 2311 0.02596454
 29 2159 0.02425679
 30 2717 0.03052603
 31 2889 0.03245849
 32 2504 0.02813293
```

```
concentrado2020 %>% tabyl(mun)
```

```
mun    n      percent
001 4929 5.537829e-02
002 4164 4.678336e-02
003 3196 3.590769e-02
004 3636 4.085118e-02
005 3578 4.019954e-02
006 3230 3.628969e-02
007 3069 3.448082e-02
008 2271 2.551513e-02
009 1779 1.998742e-02
010 2050 2.303216e-02
011 1819 2.043682e-02
012 1738 1.952677e-02
```

```
013 1317 1.479676e-02
014 2189 2.459385e-02
015  855 9.606094e-03
016 1022 1.148237e-02
017 2582 2.900928e-02
018 1436 1.613374e-02
019 1277 1.434735e-02
020 1733 1.947060e-02
021  963 1.081950e-02
022  601 6.752354e-03
023  355 3.988495e-03
024  569 6.392827e-03
025  716 8.044402e-03
026  606 6.808530e-03
027  925 1.039256e-02
028 1017 1.142620e-02
029  882 9.909444e-03
030 1851 2.079635e-02
031  853 9.583624e-03
032  889 9.988091e-03
033 1339 1.504393e-02
034  407 4.572725e-03
035 1637 1.839202e-02
036  467 5.246837e-03
037 1695 1.904366e-02
038  789 8.864571e-03
039 1434 1.611127e-02
040  309 3.471676e-03
041  756 8.493809e-03
042  402 4.516549e-03
043  445 4.999663e-03
044  337 3.786262e-03
045  269 3.022268e-03
046  524 5.887244e-03
047  259 2.909916e-03
048  634 7.123115e-03
049  234 2.629036e-03
050 1125 1.263960e-02
051  427 4.797429e-03
052  340 3.819967e-03
053  593 6.662472e-03
054  266 2.988563e-03
055  557 6.258005e-03
056  437 4.909781e-03
057  320 3.595263e-03
058  289 3.246972e-03
```

```
059  240 2.696447e-03
060  143 1.606633e-03
061  206 2.314451e-03
062  252 2.831270e-03
063  232 2.606566e-03
064  167 1.876278e-03
065  203 2.280745e-03
066  146 1.640339e-03
067  381 4.280610e-03
068   91 1.022403e-03
069  246 2.763859e-03
070   83 9.325214e-04
071  157 1.763926e-03
072   60 6.741119e-04
073  251 2.820035e-03
074  147 1.651574e-03
075   38 4.269375e-04
076  306 3.437970e-03
077  159 1.786396e-03
078  127 1.426870e-03
079  277 3.112150e-03
080   21 2.359392e-04
081   70 7.864638e-04
082  157 1.763926e-03
083  157 1.763926e-03
084   89 9.999326e-04
085  181 2.033571e-03
086   92 1.033638e-03
087  257 2.887446e-03
088   77 8.651102e-04
089  237 2.662742e-03
090   41 4.606431e-04
091  119 1.336989e-03
092   78 8.763454e-04
093   81 9.100510e-04
094   63 7.078175e-04
095   23 2.584095e-04
096  176 1.977395e-03
097  356 3.999730e-03
098  263 2.954857e-03
099  140 1.572928e-03
100   94 1.056109e-03
101  537 6.033301e-03
102  359 4.033436e-03
104  222 2.494214e-03
105  222 2.494214e-03
```

106	308	3.460441e-03
107	125	1.404400e-03
108	364	4.089612e-03
109	114	1.280813e-03
110	64	7.190526e-04
111	95	1.067344e-03
112	92	1.033638e-03
113	83	9.325214e-04
114	639	7.179291e-03
115	123	1.381929e-03
116	16	1.797632e-04
117	23	2.584095e-04
118	65	7.302878e-04
119	34	3.819967e-04
120	381	4.280610e-03
121	120	1.348224e-03
122	49	5.505247e-04
123	76	8.538750e-04
124	110	1.235872e-03
125	29	3.258207e-04
127	21	2.359392e-04
128	59	6.628767e-04
129	24	2.696447e-04
130	34	3.819967e-04
131	70	7.864638e-04
132	80	8.988158e-04
133	32	3.595263e-04
134	20	2.247040e-04
135	8	8.988158e-05
136	24	2.696447e-04
138	22	2.471743e-04
140	36	4.044671e-04
141	109	1.224637e-03
142	20	2.247040e-04
143	25	2.808799e-04
144	64	7.190526e-04
145	42	4.718783e-04
149	18	2.022336e-04
153	24	2.696447e-04
154	42	4.718783e-04
156	96	1.078579e-03
157	7	7.864638e-05
160	66	7.415230e-04
163	44	4.943487e-04
164	35	3.932319e-04
167	23	2.584095e-04

169	24	2.696447e-04
170	63	7.078175e-04
171	33	3.707615e-04
173	58	6.516415e-04
174	43	4.831135e-04
175	72	8.089342e-04
176	24	2.696447e-04
177	22	2.471743e-04
179	23	2.584095e-04
181	17	1.909984e-04
183	38	4.269375e-04
184	159	1.786396e-03
186	20	2.247040e-04
187	20	2.247040e-04
188	19	2.134688e-04
189	73	8.201694e-04
191	16	1.797632e-04
193	161	1.808867e-03
194	22	2.471743e-04
197	21	2.359392e-04
199	21	2.359392e-04
200	24	2.696447e-04
201	83	9.325214e-04
202	43	4.831135e-04
204	21	2.359392e-04
205	23	2.584095e-04
206	42	4.718783e-04
208	43	4.831135e-04
212	14	1.572928e-04
227	5	5.617599e-05
234	20	2.247040e-04
261	23	2.584095e-04
266	20	2.247040e-04
271	16	1.797632e-04
277	39	4.381727e-04
278	47	5.280543e-04
293	16	1.797632e-04
295	22	2.471743e-04
302	22	2.471743e-04
309	20	2.247040e-04
315	19	2.134688e-04
318	42	4.718783e-04
324	143	1.606633e-03
334	66	7.415230e-04
348	24	2.696447e-04
349	21	2.359392e-04

```
350   11 1.235872e-04
364   23 2.584095e-04
365   15 1.685280e-04
372   18 2.022336e-04
385   70 7.864638e-04
390   20 2.247040e-04
394   20 2.247040e-04
397   16 1.797632e-04
399   11 1.235872e-04
401   21 2.359392e-04
403    5 5.617599e-05
406   33 3.707615e-04
413   42 4.718783e-04
418   48 5.392895e-04
439   41 4.606431e-04
441   24 2.696447e-04
447   22 2.471743e-04
460   21 2.359392e-04
466   20 2.247040e-04
467   25 2.808799e-04
469   37 4.157023e-04
482   22 2.471743e-04
483   22 2.471743e-04
491   21 2.359392e-04
504   22 2.471743e-04
515   21 2.359392e-04
539   19 2.134688e-04
546   21 2.359392e-04
551   43 4.831135e-04
553    9 1.011168e-04
559   43 4.831135e-04
570   46 5.168191e-04
```

Hoy sí podemos hacer nuestras variables dentro de cada entidad federativa

```
concentrado2020 %<>%
  group_by(ent) %>%
  mutate(quintil2=dineq::ntiles.wtd(ing_cor, n=5, weights=factor)) %>%
  ungroup()
```

¿Discreparán muchos los hogares en sus distribuciones a nivel nacional y por entidad?

```
concentrado2020 %>%
  tabyl(quintil1,quintil2) %>%
```

```
  adorn_totals(c("row", "col"))
```

```
quintil1     1     2     3     4     5 Total
       1 15878  3088   167     0     0 19133
       2  4413 10071  3503   266     0 18253
       3     0  5583  8917  3221    82 17803
       4     0     0  5089 10301  2219 17609
       5     0     0     0  2969 13239 16208
   Total 20291 18742 17676 16757 15540 89006
```

Y si queremos este tabulado más bonito

```
concentrado2020 %>%
  tabyl(quintil1,quintil2) %>%
  adorn_totals(c("row", "col")) %>%
  gt()
```

quintil1	1	2	3	4	5	Total
1	15878	3088	167	0	0	19133
2	4413	10071	3503	266	0	18253
3	0	5583	8917	3221	82	17803
4	0	0	5089	10301	2219	17609
5	0	0	0	2969	13239	16208
Total	20291	18742	17676	16757	15540	89006

concentrado2020 %>% tabyl(quintil1,quintil2) %>% adorn_totals(c("row", "col")) %>% gt() %>% tab_header(title = md("Distribución de los hogares en **México**"), subtitle = md("Según quintiles y *quintiles*")) %>% tab_footnote(footnote = paste(get_label(concentrado2020$ing_cor)))

4.7 Recodificación de variables

Por ejemplo, si quisiéramos hacer una variable que separara a los hogares de acuerdo al grupo etario del jefe

4.7.1 if_else()

```
concentrado2020 %<>%
  mutate(joven=dplyr::if_else(edad_jefe<30, 1, 0))

concentrado2020 %>% tabyl(edad_jefe,joven)
```

edad_jefe	0	1
14	0	1
15	0	2
16	0	19
17	0	28
18	0	93
19	0	153
20	0	243
21	0	312
22	0	423
23	0	588
24	0	640
25	0	825
26	0	914
27	0	1028
28	0	1163
29	0	1193
30	1461	0
31	1221	0
32	1495	0
33	1426	0
34	1493	0
35	1627	0
36	1654	0
37	1619	0
38	1899	0
39	1769	0
40	2146	0
41	1590	0
42	2232	0
43	2029	0
44	1827	0
45	2163	0
46	1975	0
47	2138	0
48	2153	0
49	2028	0
50	2315	0
51	1672	0
52	2044	0
53	1859	0
54	1942	0
55	1900	0
56	1900	0
57	1644	0
58	1704	0

59	1589	0
60	1930	0
61	1283	0
62	1563	0
63	1545	0
64	1344	0
65	1514	0
66	1187	0
67	1216	0
68	1331	0
69	1003	0
70	1255	0
71	837	0
72	1027	0
73	965	0
74	919	0
75	912	0
76	754	0
77	655	0
78	764	0
79	532	0
80	695	0
81	404	0
82	463	0
83	406	0
84	430	0
85	402	0
86	317	0
87	250	0
88	215	0
89	144	0
90	171	0
91	72	0
92	85	0
93	76	0
94	54	0
95	45	0
96	33	0
97	29	0
98	19	0
99	7	0
100	5	0
101	2	0
102	2	0
103	1	0
104	2	0

```
      105     1     0
      107     1     0
```

4.7.2 case_when()

Esto nos ayuda para recodificación múltiple

```
concentrado2020 %<>%
  mutate(grupo_edad2=dplyr::case_when(edad_jefe<30 ~ 1,
                                      edad_jefe>29 & edad_jefe<45 ~ 2,
                                      edad_jefe>44 & edad_jefe<65 ~ 3,
                                      edad_jefe>64 ~ 4))

#TRUE~ 4

concentrado2020 %>% tabyl(edad_jefe,grupo_edad2)
```

```
edad_jefe    1    2    3    4
       14    1    0    0    0
       15    2    0    0    0
       16   19    0    0    0
       17   28    0    0    0
       18   93    0    0    0
       19  153    0    0    0
       20  243    0    0    0
       21  312    0    0    0
       22  423    0    0    0
       23  588    0    0    0
       24  640    0    0    0
       25  825    0    0    0
       26  914    0    0    0
       27 1028    0    0    0
       28 1163    0    0    0
       29 1193    0    0    0
       30    0 1461    0    0
       31    0 1221    0    0
       32    0 1495    0    0
       33    0 1426    0    0
       34    0 1493    0    0
       35    0 1627    0    0
       36    0 1654    0    0
       37    0 1619    0    0
       38    0 1899    0    0
       39    0 1769    0    0
       40    0 2146    0    0
       41    0 1590    0    0
```

42	0	2232	0	0
43	0	2029	0	0
44	0	1827	0	0
45	0	0	2163	0
46	0	0	1975	0
47	0	0	2138	0
48	0	0	2153	0
49	0	0	2028	0
50	0	0	2315	0
51	0	0	1672	0
52	0	0	2044	0
53	0	0	1859	0
54	0	0	1942	0
55	0	0	1900	0
56	0	0	1900	0
57	0	0	1644	0
58	0	0	1704	0
59	0	0	1589	0
60	0	0	1930	0
61	0	0	1283	0
62	0	0	1563	0
63	0	0	1545	0
64	0	0	1344	0
65	0	0	0	1514
66	0	0	0	1187
67	0	0	0	1216
68	0	0	0	1331
69	0	0	0	1003
70	0	0	0	1255
71	0	0	0	837
72	0	0	0	1027
73	0	0	0	965
74	0	0	0	919
75	0	0	0	912
76	0	0	0	754
77	0	0	0	655
78	0	0	0	764
79	0	0	0	532
80	0	0	0	695
81	0	0	0	404
82	0	0	0	463
83	0	0	0	406
84	0	0	0	430
85	0	0	0	402
86	0	0	0	317
87	0	0	0	250

```
 88    0    0    0  215
 89    0    0    0  144
 90    0    0    0  171
 91    0    0    0   72
 92    0    0    0   85
 93    0    0    0   76
 94    0    0    0   54
 95    0    0    0   45
 96    0    0    0   33
 97    0    0    0   29
 98    0    0    0   19
 99    0    0    0    7
100    0    0    0    5
101    0    0    0    2
102    0    0    0    2
103    0    0    0    1
104    0    0    0    2
105    0    0    0    1
107    0    0    0    1
```

4.7.3 rename()

Para cambiar los nombres de las variables podemos cambiarlos nombres

```
concentrado2020 %<>%
  dplyr::rename(nuevo_nombre=grupo_edad2)
```

Esto en base sería similar a

```
names(concentrado2020)[134]<-"grupo_edad2"
names(concentrado2020)
```

```
 [1] "folioviv"    "foliohog"    "ubica_geo"   "tam_loc"     "est_socio"
 [6] "est_dis"     "upm"         "factor"      "clase_hog"   "sexo_jefe"
[11] "edad_jefe"   "educa_jefe"  "tot_integ"   "hombres"     "mujeres"
[16] "mayores"     "menores"     "p12_64"      "p65mas"      "ocupados"
[21] "percep_ing"  "perc_ocupa"  "ing_cor"     "ingtrab"     "trabajo"
[26] "sueldos"     "horas_extr"  "comisiones"  "aguinaldo"   "indemtrab"
[31] "otra_rem"    "remu_espec"  "negocio"     "noagrop"     "industria"
[36] "comercio"    "servicios"   "agrope"      "agricolas"   "pecuarios"
[41] "reproducc"   "pesca"       "otros_trab"  "rentas"      "utilidad"
[46] "arrenda"     "transfer"    "jubilacion"  "becas"       "donativos"
[51] "remesas"     "bene_gob"    "transf_hog"  "trans_inst"  "estim_alqu"
[56] "otros_ing"   "gasto_mon"   "alimentos"   "ali_dentro"  "cereales"
[61] "carnes"      "pescado"     "leche"       "huevo"       "aceites"
```

```
 [66] "tuberculo"   "verduras"    "frutas"      "azucar"      "cafe"
 [71] "especias"    "otros_alim"  "bebidas"     "ali_fuera"   "tabaco"
 [76] "vesti_calz"  "vestido"     "calzado"     "vivienda"    "alquiler"
 [81] "pred_cons"   "agua"        "energia"     "limpieza"    "cuidados"
 [86] "utensilios"  "enseres"     "salud"       "atenc_ambu"  "hospital"
 [91] "medicinas"   "transporte"  "publico"     "foraneo"     "adqui_vehi"
 [96] "mantenim"    "refaccion"   "combus"      "comunica"    "educa_espa"
[101] "educacion"   "esparci"     "paq_turist"  "personales"  "cuida_pers"
[106] "acces_pers"  "otros_gas"   "transf_gas"  "percep_tot"  "retiro_inv"
[111] "prestamos"   "otras_perc"  "ero_nm_viv"  "ero_nm_hog"  "erogac_tot"
[116] "cuota_viv"   "mater_serv"  "material"    "servicio"    "deposito"
[121] "prest_terc"  "pago_tarje"  "deudas"      "balance"     "otras_erog"
[126] "smg"         "grupo"       "quintil0"    "quintil1"    "ent"
[131] "mun"         "quintil2"    "joven"       "grupo_edad2"
```

4.8 Práctica

- Genere una variable de deciles de ingresos dentro de cada tamaño de localidad tam_loc
- Etiquete los valores de los deciles con números romanos
- Encuentre el coeficiente de variación para las estimaciones dentro de esa variable, sexo del jefe y tamaño de localidad