R para el análisis estadístico de datos

Ana Escoto

6/10/24

Table of contents

Sobre el curso **3**
1. Introducción a R y Rstudio (4 horas) . . . 3
2. Importación de información y primera revisión de fuentes demográficas (4 horas) 3
3. Revisión de elementos estadísticos básicos desde `{tidyverse}` (8 horas) . . . 3
4. Estimaciones por intervalo y diseño complejo muestral (4 horas) . . . 4

Facilitadora **5**
Ana Ruth Escoto Castillo . . . 5

Instalación de R y Rstudio **6**
Introducción a R . . . 6
Instalación en OS . . . 6
Instalación en PC . . . 7
Ojo . . . 7

1 Primer acercamiento al uso del programa **8**
1.1 Introducción . . . 8
1.2 Vectores . . . 9
1.3 Matrices . . . 10
1.4 data.frames o conjuntos de datos . . . 12
1.5 Valores y perdidos . . . 13
1.6 Funciones . . . 14
1.7 Listas . . . 15
1.8 Ayuda . . . 16
1.9 Mi ambiente . . . 17
1.10 Directorio de trabajo . . . 18
1.11 Proyectos . . . 18
1.12 Instalación de paquetes . . . 20
1.13 Paquete `{pacman}` . . . 20
1.14 Estilos . . . 21
1.15 Ejercicio 1 . . . 21

Videos y extras **22**
Sesión 1 . . . 22

Sobre el curso

1. Introducción a R y Rstudio (4 horas)

Objetivo: que el estudiantado sea familiarice con la interfase de trabajo y la programación por objetos, del mismo modo que sean capaces de realizar tareas básicas tales como crear un script, un proyecto, objetos, ambientes e instalar paqueterías.

2. Importación de información y primera revisión de fuentes demográficas (4 horas)

a. Importación de información a R en diferentes formatos
b. Revisión de encuestas y ejemplos de importación de datos en formatos diferentes
c. Revisión preliminar y limpieza de información

Objetivo: que el estudiantado sea capaz de importar información desde diferentes formatos (.txt, .csv, .xlsx, .dta, .dbf) a R, así como de exportar sus resultados en estos formatos. Del mismo modo que sean capaces de revisar de manera preliminar los objetos de tipo `data.frame`*: funciones* `dplyr::glimpse()`*,* `skimr::skim()`*, manejo de etiquetas y hacer subconjuntos de información*

3. Revisión de elementos estadísticos básicos desde `{tidyverse}` (8 horas)

a. Tabulados con `janitor::tabyl()` y uso de factores de expansión con `pollster::topline()`, `pollster::crosstab`. Lectura e interpretación de tablas de doble entrada
b. Estadística descriptiva básica (medidas de tendencia central, dispersión y de posición) con el paquete `{dplyr}`
c. Gráficos univariados y bivariados usando `{ggplot2}` y extensiones de `{ggplot2}`
d. Fusionado de información

Objetivo: que el estudiantado sea capaz de realizar análisis estadísticos básicos utilizando encuestas mexicanas

4. Estimaciones por intervalo y diseño complejo muestral (4 horas)

a. Estimaciones para medias

b. Estimaciones para proporciones

c. Estimaciones para totales

d. Estimaciones lineales de coeficientes

Objetivo: que el estudiantado sea capaz de realizar intervalos de confianza, calculo de errores estándar con diseño muestral complejo y sepa evaluar la confiabilidad de sus estimaciones

Facilitadora

Ana Ruth Escoto Castillo

Profesora de tiempo completo en la Facultad de Ciencias Políticas y Sociales, UNAM. Doctora en Estudios de Población por El Colegio de México, cuenta con el nivel I en el Sistema Nacional de Investigadores. Coorganizadora del capítulo de la CDMX de la iniciativa global Rladies. Le interesa el bienestar de la población, en el presente, analizando los procesos de desigualdad y exclusión en los mercados laborales latinoamericanos; y, en el futuro, a través del estudio de la sustentabilidad. Su experiencia en el ámbito académico se ha concentrado en el estudio de este bienestar, específicamente en el análisis sociodemográfico de las condiciones laborales y la vinculación del comercio exterior con el mercado de trabajo, en la relación del cambio climático y la distribución de ingresos, el consumo energético de los hogares y sus implicaciones ambientales. 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.

Instalación de R y Rstudio

Introducción a R

https://youtu.be/YkN5urybh2A

Instalación en OS

1. Necesito que instalen la versión más nueva de R: Download R-4.4.0 of MAC. *The R-project for statistical computing*. https://cran.r-project.org/bin/macosx/

Elije la versión de acuerdo a tu procesador, intel o ARM.

2. Instalar también las herramientas Quartz, xcode y fortran

- https://www.xquartz.org/
- https://developer.apple.com/xcode/resources/
- https://mac.r-project.org/tools/gfortran-12.2-universal.pkg

3. Después de eso instalar el Rstudio, que hoy se encuentra alojado en el sitio posit, que vaya acorde con MAC

https://posit.co/download/rstudio-desktop/

Algunas indicaciones en video, pero son algo viejitas y pueden cambiar las versiones de R.

https://youtu.be/icWV8jzYOtA

Algunas indicaciones en video, pero son algo viejitas y pueden cambiar las versiones de R.

Instalación en PC

1. Necesito que instalen la versión más nueva de R: Download R-4.4.0 for Windows. *The R-project for statistical computing.* https://cran.r-project.org/bin/windows/base/
2. Instalar también la herramienta RTools https://cran.r-project.org/bin/windows/Rtools/rtools44/rtools.html
3. Después de eso instalar el Rstudio, que hoy se encuentra alojado en el sitio posit, que vaya acorde con Windows https://posit.co/download/rstudio-desktop/

Algunas indicaciones en video, pero son algo viejitas y pueden cambiar las versiones de R.

https://youtu.be/TNSQikMfgJI

Ojo

Desde octubre de 2022, RStudio se volvió **"Posit"**

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"
```

```r
1:7             # Entero
```

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

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

```
[1] TRUE
```

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

```
[1] FALSE
```

```r
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.

```r
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
```

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

```
[1] 24
```

```r
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
```

```
dim(m)
```

```
[1] 2 3
```

```
attributes(m)
```

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

¿Qué hace “byrow”?

```
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 data.frames o conjuntos de datos

```
mi_dataframe<-as.data.frame(m)
```

El formato matricial sigue sirviendo:

```
mi_dataframe[2,]
```

```
  V1 V2 V3
2  2  4  6
```

```
mi_dataframe[,2]
```

```
[1] 3 4
```

Pero también podemos utilizar el símbolo de peso para cada variable:

```
mi_dataframe$V2
```

```
[1] 3 4
```

Puedo agregar variables columnas:

```
cbind(mi_dataframe, c("a", "b"), c(T, F))
```

```
  V1 V2 V3 c("a", "b") c(T, F)
1  1  3  5           a    TRUE
2  2  4  6           b   FALSE
```

Qué pasa con las matrices

```
cbind(m, c("a", "b"),  c(T, F))
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,] "1"  "3"  "5"  "a"  "TRUE"
[2,] "2"  "4"  "6"  "b"  "FALSE"
```

Checa cómo cambian los elementos. En una matriz todos los elementos deben ser del mismo tipo.

Podemos crear "a mano" dataframes:

```
data<-data.frame(
  "entero" = 1:4,
  "factor" = as.factor(c("a", "b", "c", "d")),
  "numero" = c(1/1, 1/2, 1/3, 1/4),
  "cadena" = as.character(c("a", "b", "c", "d"))
)
```

Los data.frames tienen una estructura

```
str(data)
```

```
'data.frame':   4 obs. of  4 variables:
 $ entero: int  1 2 3 4
 $ factor: Factor w/ 4 levels "a","b","c","d": 1 2 3 4
 $ numero: num  1 0.5 0.333 0.25
 $ cadena: chr  "a" "b" "c" "d"
```

1.5 Valores y perdidos

Además de caracteres, numéricos y lógicos hay también valores perdidos. Y de varios tipos

```
vector<-c(1:5, # numérico
          T, # lógico
          NA, # perdido
          "a", # caracter
          5/0, # no es un número
          sqrt(-1))
```

```
Warning in sqrt(-1): Se han producido NaNs
```

Si lo imprimimos vamos a ir viendo cómo se convierten ciertos valores a otros al quererlos incluir en un mismo conjunto:

```
vector
```

```
 [1] "1"    "2"    "3"    "4"    "5"    "TRUE" NA     "a"    "Inf"  "NaN"
```

Quitaremos el caracter

```
vector<-c(1:5, # numérico
          T, # lógico
          NA, # perdido
          5/0, # Infinito
          sqrt(-1))
```

```
Warning in sqrt(-1): Se han producido NaNs
```

```
vector
```

```
[1]   1   2   3   4   5   1  NA Inf NaN
```

¿Qué le pasó al valor lógico?

Hay unos operadores que nos señalan si los valores son perdidos o infinitos o "Not a number"

```
is.na(vector)
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
```

```
is.nan(vector)
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
is.infinite(vector)
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
```

1.6 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.

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

```
[1] 60
```

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

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

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

```
[1] 3
```

1.7 Listas

Las listas son conjuntos de objetos y pueden ser de varios tipos

```r
milista<- list(data, m, xx, "a")
```

```r
milista
```

```
[[1]]
  entero factor    numero cadena
1      1      a 1.0000000      a
2      2      b 0.5000000      b
3      3      c 0.3333333      c
4      4      d 0.2500000      d

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

[[3]]
[1] 10 11 12

[[4]]
[1] "a"
```

Ojo con los corchetes

```
milista[[1]]
```

```
  entero factor    numero cadena
1      1      a 1.0000000      a
2      2      b 0.5000000      b
3      3      c 0.3333333      c
4      4      d 0.2500000      d
```

Si queremos ponerle nombres a los elementos

```
milista<- list(datos = data,
               matriz = m,
               vector = xx,
               valor = "a")

milista
```

```
$datos
  entero factor    numero cadena
1      1      a 1.0000000      a
2      2      b 0.5000000      b
3      3      c 0.3333333      c
4      4      d 0.2500000      d

$matriz
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

$vector
[1] 10 11 12

$valor
[1] "a"
```

1.8 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 ?

```r
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
```

1.9 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

```r
ls()
```

```
 [1] "data"         "m"            "mi_dataframe" "mi_matrix"    "milista"
 [6] "n"            "sex"          "vector"       "x"            "xx"
[11] "y"            "yy"
```

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

```
         used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
Ncells  639595 34.2    1346222 71.9         NA  1346222 71.9
Vcells 1200727  9.2    8388608 64.0      16384  2147502 16.4
```

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
```

1.10 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/2024/Curso_R_inter/r_analisis_datos/r_analisis_datos"
```

```
#setwd("")# Cambio de directorio

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

```
 [1] "P1.html"               "P1.qmd"                "P1.rmarkdown"
 [4] "P2.qmd"                "README.md"             "_quarto.yml"
 [7] "datos"                 "docs"                  "index.html"
[10] "index.qmd"             "instala.html"          "instala.qmd"
[13] "intro1.png"            "r_analisis_datos.Rproj" "scripts"
[16] "site_libs"             "videos.qmd"
```

Checar que esto también se puede hacer desde el menú:

1.11 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

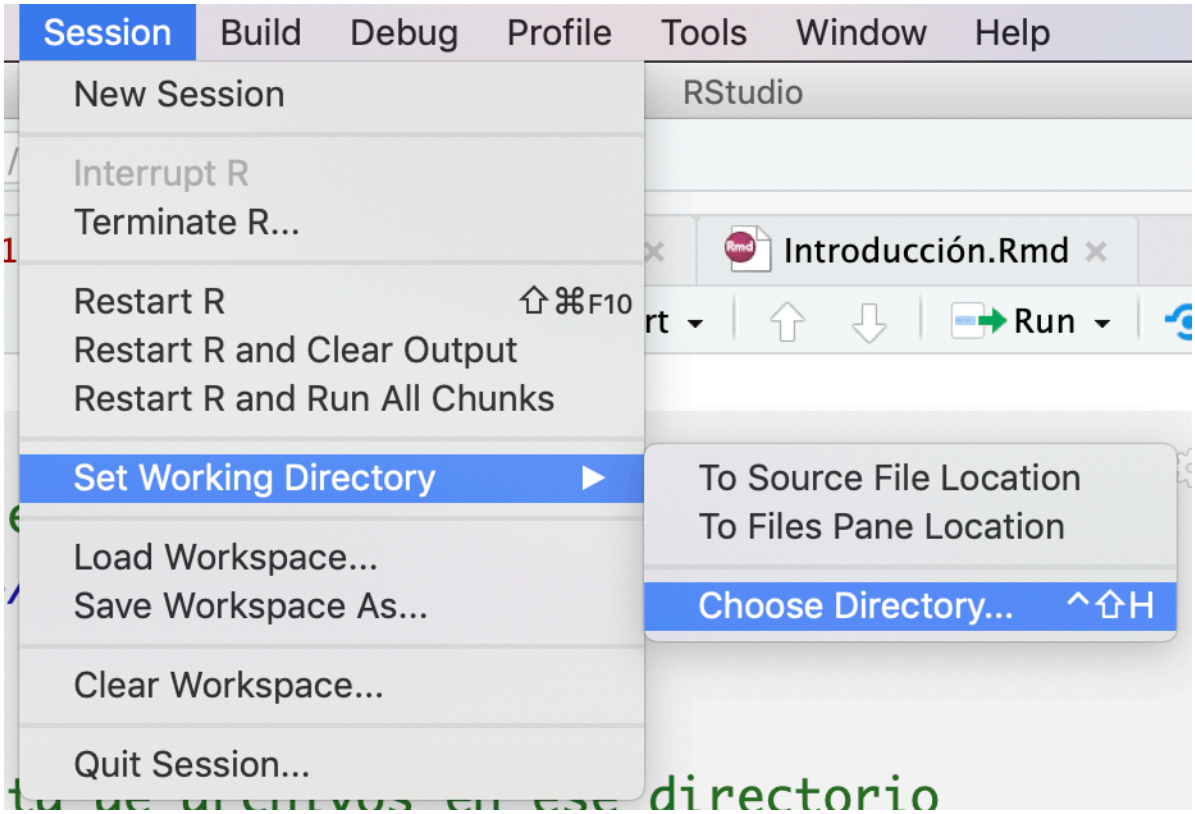
Session
Build
Debug
Profile
Tools
Window
Help
New Session
Interrupt R
Terminate R...
Restart R
Restart R and Clear Output
Restart R and Run All Chunks
Set Working Directory
Load Workspace...
Save Workspace As...
Clear Workspace...
Quit Session...
To Source File Location
To Files Pane Location
Choose Directory...
RStudio
Introducción.Rmd
Run

Figure 1.1: i0

1.12 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 software libre.

Vamos a instalar el paquete `{foreign}`, como su nombre lo indica, nos permite leer elementos "extranjeros" en R.

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

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(haven)
library(foreign)
```

`{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<-foreign::read.dbf("datos/ejemplo_dbf.DBF") #checa cómo nos vamos adentro de n
```

Los `::` sirven para tres cosas:

- cargar un comando de un paquete, sin haberlo cargado
- para identificar de qué paquete viene el comando.
- para especificar en caso que hayan dos comandos iguales en un paquete, usar el que tenemos de los paquetes.

1.13 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
```

```
Cargando paquete requerido: pacman
```

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

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

1.14 Estilos

Escribir código tiene su gramática. Por lo general en este curso seguiremos el estilo de Google

https://google.github.io/styleguide/Rguide.html

1.15 Ejercicio 1

Realice en un **nuevo script** lo siguiente:

1. Escriba un vector "x", con los elementos 2,3,7,9. Muestre el resultado
2. Escriba un vector "y", con los elementos 9, 7, 3, 2. Muestre el resultado
3. Escriba un vector "year" con los años que van desde 1990 a 1993. Muestre el resultado
4. Escriba un vector "name" con los nombres de 4 de sus compañeros de curso. Muestre el resultado
5. Cree una matrix "m" 2x4 que incluya los valores 101 a 108, que se ordene según fila
6. ¿Cuáles son las dimensiones de la matriz "m"?
7. Cree una matriz "m2" juntado los vectores "x" y "y", por sus filas ¿Cuáles son las dimensiones de la matriz "m2"?
8. Convierta esa matriz en un *data.frame*
9. Escriba una lista

Entregue su resultado en este formulario

Videos y extras

Sesión 1

https://youtu.be/N78ZLRTZeLg

Código