Clase práctica 1

Inferencia Estadística

1er semestre 2023

Por qué R

Porque:

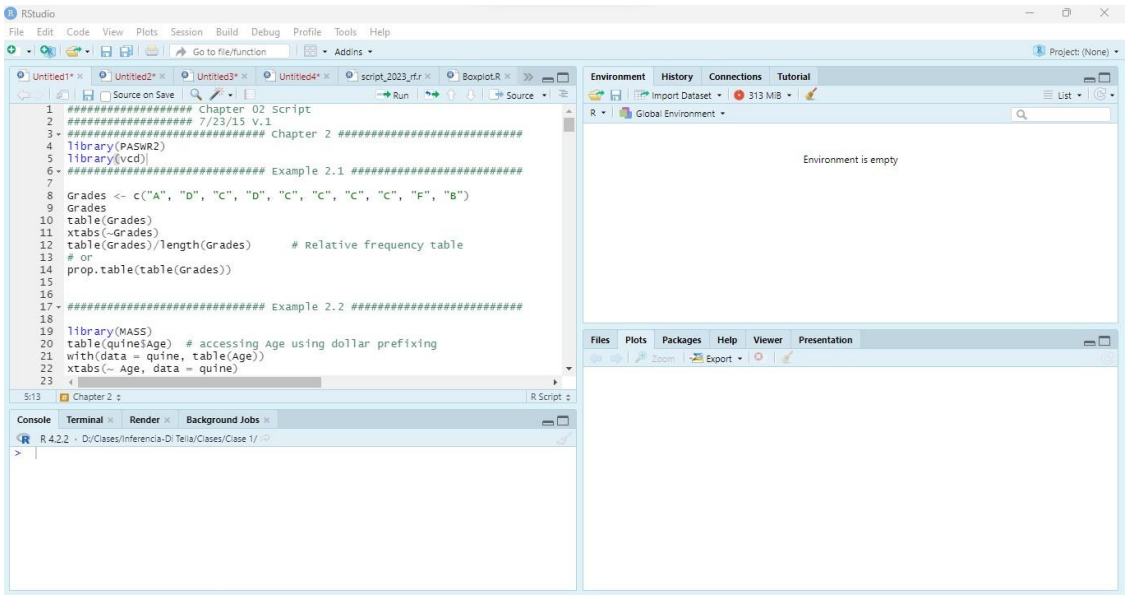
1. R es el software de mayor interés y uso dentro de la comunidad de Estadística y ramas afines.
2. Tiene una amplia variedad de técnicas estadísticas y gráficas ya implementadas y está muy actualizado.
3. Tiene una gran comunidad de usuarios que contribuyen con código, preguntas, respuestas y actividades.
4. Es gratuito y de código abierto.

Tutoriales

- Probability, Statistics, and Data : A Fresh Approach Using R
- Videos de Jemina:
  - Instalación
  - Operaciones básicas en R
  - Funciones y ciclos
  - Gráficos

Dónde usar R | RStudio

- Pueden usar cualquier IDE, pero acá vamos a usar RStudio.
- Se descarga de https://www.rstudio.com/.
- Se divide en 4 partes:
  - Consola
  - Script
  - Entorno
  - Salida



\

Datos en R

En R todo es un objeto: los datos y las estructuras de datos lo son. Todos los objetos tienen un nombre para poder identificarlos.

Algunos tipos de datos

1) Los tipos de datos más comunes que vamos a utilizar:

Nombre	Tipo	Ejemplo
integer	entero	1
numeric	numérico	1,5
character	cadena de texto	“Galicia”
lógico	lógico	TRUE/FALSE
NA	perdido	NA
null	vacío	NULL

- Se accede al tipo de dato con el commando str o el comando class.
- Ejemplo

```
# Defino un vector de caracteres
spice_girls <- c('Sporty','Baby','Posh','Scary','Ginger')

# Veo qué el tipo de datos es caracter
str(spice_girls)

##  chr [1:5] "Sporty" "Baby" "Posh" "Scary" "Ginger"

# lo redefino como un vector de factores (lo veremos con detalle más adelante)
spice_girl_class <- as.factor(spice_girls)
```

```
# Veo qué el tipo de datos ahora es factor
str(spice_girl_class)

##  Factor w/ 5 levels "Baby","Ginger",..: 5 1 3 4 2
```

2) Algunas estructuras de datos:

Las colecciones o conjunto de datos en R se organizan por su dimensión (1º, 2º, o varias dimensiones) y si son homogéneas (todos los objetos deben ser del mismo tipo) o heterogéneas (el contenido puede ser de diferentes tipos). A continuación mostramos los cinco tipos de datos más usados en el análisis de datos:

Dimensión	Homogénea	Heterogénea
1	Vector	Lista
2	Matriz	Data frame
n	Array	

Las más utilizadas en este curso son:

- Vector
- Hoja de datos o *data frame*

Vectores

Es una colección de uno o más datos del mismo tipo. Hay varias formas de definir un vector. Las más sencillas son:

- Usando la función *c()*, que significa *'concatenar'* o *'combinar'*:

```
x <-c(1,3,5)
x

## [1] 1 3 5

y <- c("arbol","casa")
y

## [1] "arbol" "casa"
```

- O usando secuencias numéricas:

```
x <- 1:5
x

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

Esta opción nos permite definir una secuencia del 1 al 5. Si queremos que la separación entre los valores sea distinta de 1, usamos la función *seq()* :

```
y <- seq(-3,3,0.5)
y
```

```
##  [1] -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
```

Si queremos un vector, por ejemplo, 10 números 5, usamos la función *rep()*:

```
z <- rep(x = 5, times = 10)
z
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5
```

R hace distinción entre mayúsculas y minúsculas: *x* y *X* son objetos distintos.

Para llegar a elementos concretos de un vector, debemos tener en cuenta que los vectores en R comienzan en *1*:

```
NV <- c(-4, 0, 2, 4, 6)

#Quiero el elemento que esta en la posición 3 (el número 2)

NV[3]
```

```
## [1] 2
```

Si quiero los elementos que están en la posición 2, 3 y 4:

```
NV[c(2:4)]
```

```
## [1] 0 2 4
```

- Operaciones con vectores

```
#rm(x,y,z)

x <- c(7,2)
y <- 1:7
z <- 3:9
```

Producto entre un vector y un escalar:

```
x * 2
```

```
## [1] 14  4
```

Suma de vectores de igual longitud

```
y
```

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

```
z
```

```
## [1] 3 4 5 6 7 8 9
```

```
y + z
```

```
## [1]  4  6  8 10 12 14 16
```

¿Qué pasa si sumo vectores de distinta longitud?

```
2 + y

## [1] 3 4 5 6 7 8 9

w <- 1:6


x + w

## [1]  8  4 10  6 12  8

x + y

## Warning in x + y: longitud de objeto mayor no es múltiplo de la
longitud de uno
## menor

## [1]  8  4 10  6 12  8 14
```

- Operadores lógicos:

<,>,<=,>=

== : exactamente igual

!= : distinto

Antes de empezar con los dataframe, veamos el directorio de trabajo

Preparación del entrono

- El directorio de trabajo es el sitio donde localizaremos todos los datos, los resultados, los gráficos, etc. Para situarnos en un directorio concreto tenemos dos formas:

1. Usando Session -> Set Working Directory -> Choose Directory …
2. Usando la función *setwd*:

```
# Elección del directorio de trabajo
setwd("/Path/al/directorio/de/trabajo")
```

3. Los paquetes (*packages*), nos permiten extender R: agregan funciones, conjuntos de datos y visualizaciones. Primero hay que instalarlos:

Tools > Install Packages…

o

```
# Paquete del libro Introduction to Statistical
#Learning
install.packages('ISLR2')
```

Para usarlos, hay que cargarlos al entorno con la función library():

```
# Importo la biblioteca datasets con el comando library
library(datasets)
```

Data frame

El dataframe es una estructura de datos en forma de tabla, usada para guardar información donde cada fila representa un individuo u observación y cada columna representa una variable. Las variables no están obligadas a ser del mismo tipo, a diferencia de los vectores y matrices.

Creando un data frame

La función data.frame() permite crear data frame usando vectores de igual longitud. Si queremos que los vectores de caracter no los convierta a *factor* (tipo de estructura que veremos más adelante), debemos usar stringsAsFactor=FALSE. Por defecto, las variables llevan los nombres de los vectores:

```
nv <- c(1, 3, 6, 8)                    # vector numerico
cv <- c("a", "d", "f", "p")            # vector caracter
lv <- c(TRUE, FALSE, FALSE, TRUE)      # vecror logico
DF1 <- data.frame(nv, cv, lv)
DF1
```

```
##   nv cv    lv
## 1  1  a  TRUE
## 2  3  d FALSE
## 3  6  f FALSE
## 4  8  p  TRUE
```

```
str(DF1)
```

```
## 'data.frame':    4 obs. of  3 variables:
##  $ nv: num  1 3 6 8
##  $ cv: chr  "a" "d" "f" "p"
##  $ lv: logi  TRUE FALSE FALSE TRUE
```

```
# data frame con nombres en las filas
DF2 <- data.frame(nv, cv, lv, row.names = c("Joe", "Bob", "Jill", "Sam"),
                  stringsAsFactors = FALSE)

DF2
```

```
##      nv cv    lv
## Joe   1  a  TRUE
## Bob   3  d FALSE
## Jill  6  f FALSE
## Sam   8  p  TRUE
```

```
str(DF2)
```

```
## 'data.frame':    4 obs. of  3 variables:
##  $ nv: num  1 3 6 8
```

```
##  $ cv: chr  "a" "d" "f" "p"
##  $ lv: logi  TRUE FALSE FALSE TRUE
```

```
rm("nv", "cv", "lv")  # remueve las variables del entorno de trabajo actual
```

```
#asignar nombres a las filas del data frame
row.names(DF1) <- c("Joe", "Bob", "Jill", "Sam")
DF1
```

```
##      nv cv    lv
## Joe   1  a  TRUE
## Bob   3  d FALSE
## Jill  6  f FALSE
## Sam   8  p  TRUE
```

Para acceder a un elemento concreto de un data frame, se puede utilizar $ o [[]]:

```
DF2
```

```
##      nv cv    lv
## Joe   1  a  TRUE
## Bob   3  d FALSE
## Jill  6  f FALSE
## Sam   8  p  TRUE
```

```
# formas de extraer la variable "nv"
DF2$nv
```

```
## [1] 1 3 6 8
```

```
DF2[[1]]
```

```
## [1] 1 3 6 8
```

```
DF2[["nv"]]
```

```
## [1] 1 3 6 8
```

```
DF2[, "nv"]   # todas las filas, columna nv
```

```
## [1] 1 3 6 8
```

```
DF2[, 1]      # todas las filas, columna 1
```

```
## [1] 1 3 6 8
```

```
DF2[c(1, 3)]  # extraer la 1ra y la  3er columna
```

```
##      nv    lv
## Joe   1  TRUE
## Bob   3 FALSE
```

```
## Jill  6 FALSE
## Sam   8  TRUE
```

```
DF2[c("nv", "lv")]  # extraer columnas llamadas nv y lv
```

```
##      nv    lv
## Joe   1  TRUE
## Bob   3 FALSE
## Jill  6 FALSE
## Sam   8  TRUE
```

Accediendo a un data frame desde un paquete

Para acceder a un data frame desde un paquete, primero debemos cargar dicho paquete con la función library() (library(PackageName)).

```
# Cargo paquete y datos
library(MASS)

data(Animals)
```

Descripción de la base de datos: cuales son las variables y de qué tipo son:

```
str(Animals)
```

```
## 'data.frame':    28 obs. of  2 variables:
##  $ body : num  1.35 465 36.33 27.66 1.04 ...
##  $ brain: num  8.1 423 119.5 115 5.5 ...
```

Visualizo las 6 primeras filas

```
head(Animals)
```

```
##                      body brain
## Mountain beaver      1.35   8.1
## Cow                465.00 423.0
## Grey wolf           36.33 119.5
## Goat                27.66 115.0
## Guinea pig           1.04   5.5
## Dipliodocus      11700.00  50.0
```

Leyendo datos en R

R tiene la capacidad de leer datos de archivos externos almacenados en varios formatos.Para leer datos en formatos que no sean ASCII, se puede utilizar el paquete foreign.

Cuando se trabaja con datos almacenados en un formato que R no puede leer, casi siempre resultará más fácil primero guardar los datos originales como un archivo de texto y luego leer el archivo externo usando read.table( ) o read.csv( )

Datos *txt*

```
datos <- read.table('archivo.txt')
```

Datos *csv*

```
datos <- read.csv('archivo.txt', sep='', header = TRUE)
```

También es posible utilizar una URL para leer datos. Es necesario conocer la dirección directa al archivo de texto y su extensión.

```
datos <- read.table("http://www.algunapagina.com/datos/nombre-
archivo.txt")
```

O si tiene extension .r

```
source("http://www.algunapagina.com/datos/nombre-archivo.R")
```