

Introducción al Análisis Exploratorio de Datos (EDA) en R

Módulo 1

2024-03-16

1 R y RStudio

2 Elementos básicos de R

3 Herramientas para la manipulación de datos

4 Bibliografía de consulta

R y RStudio

El entorno R

R es un entorno de programación para el análisis de datos y gráficos (R Core Team, 2000). Algunas características de R son las siguientes:

- Permite el almacenamiento y la manipulación de datos.
- Incluye una amplia colección integrada de herramientas para el análisis de datos.
- Dispone de un lenguaje de programación interpretado, simple y efectivo que incluye condicionales, ciclos, funciones recursivas, etc. (Muchas de las funciones suministradas en el Sistema están escritas en lenguaje R).
- La funcionalidad de R consiste en paquetes modulares.

¿Por qué R?

R tiene las siguientes ventajas:

- R es un software libre
- R es multiplataforma
- R tiene una **sofisticada capacidad para hacer gráficos**, incluyendo paquetes gráficos especializados.
- R tiene librerías que cubre un amplio rango de la metodologías de la estadística y las matemáticas (series temporales, optimización matemática, inferencia estadística, etc.)
- Existe una comunidad activa que ha promovido el incremento en su número de usuarios ([The R Project](#), [R Contributor Site](#), [RStudio Community](#) y [R Bloggers](#)).

R y RStudio

RStudio es un **IDE (Integrated Development Environment)** para el lenguaje de programación R.
Para instalar R: <https://cran.r-project.org/>
Para instalar RStudio:
<http://www.rstudio.com/download>
RStudio facilita el trabajo con R a través de una interfaz que es común a Windows, Mac OS y Linux.



Elementos básicos de R

Estructuras de datos en R

Nos concentraremos en las siguientes estructuras de datos:

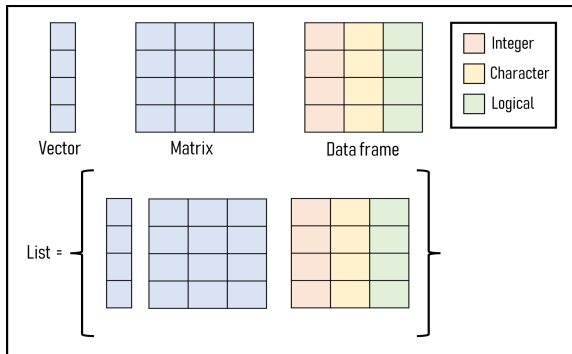


Figure 1: Estructuras rectangulares de datos en R

Clases de objetos

La función `class()` permite recuperar alguna de las siguientes clases

- Cadena de caracteres
- Numérico (*double*)
- Entera
- Lógico

1	"A"	"Female"
54	"Hello"	"Male"
-9	"Arg"	"Male"
2	"AAAA"	"Female"
Integer	Character	Factor

0.5	TRUE
2.86	FALSE
-0.09	FALSE
243.7	TRUE
Numeric	Logical

Figure 2: Caption for the picture.

Vectores

El **operador de asignación** `<-` es usado para asignar un nombre al objeto. Para definir vectores:

```
v1 <- c("a", "b", "c")  
v1
```

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

Para seleccionar el elemento i :

```
v1[3]
```

```
## [1] "c"
```

Recuerde que

- `length()` muestra la longitud
- `typeof()` muestra el tipo de vector
- `names()` nombra los elementos
- R usa **ejecución por elementos**

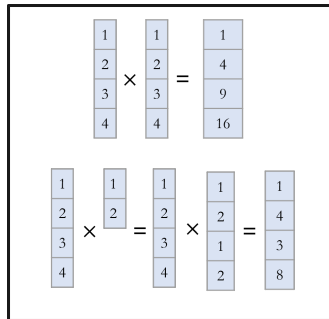


Figure 3: Caption for the picture.

Matrices

Una matriz se define usando `matrix()`. El parámetro `byrow` determina si las entradas son completadas por filas (T) o columnas (F).

```
m1 <- matrix(1:9, nrow = 3, ncol = 3, byrow = T)
m1
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

Las entradas son seleccionadas con `[i, j]`:

```
m1[2,3] # Fila 2 y columna 3
```

```
## [1] 6
```

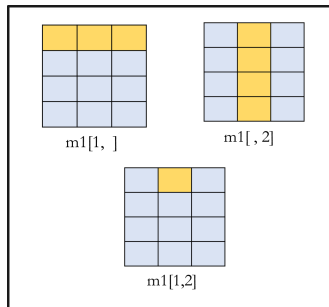


Figure 4: Caption for the picture.

Nota: operaciones con matrices

A diferencia de la ejecución por elemento $a*b$, el producto de matrices se obtiene de

```
m1 %*% t(m1) # m1 por su traspuesta
```

```
##      [,1] [,2] [,3]  
## [1,]  14  32  50  
## [2,]  32  77 122  
## [3,]  50 122 194
```

La **Figura X** muestra otras operaciones útiles usando matrices

Operación	Descripción
$\dim(m1)$	Dimensión de $m1$ ($n \times m$)
$m1 + m2$	Suma
$m1 - m2$	Resta
$t(m1)$	Traspuesta de $M1$
$2*m1$	Multiplicación por escalar
$m1 \%*\% m2$	Multiplicación de matrices
$\det(m1)$	Determinante de $m1$
$\text{solve}(m1)$	Inversa de $m1$
$\text{diag}(m1)$	Diagonal de $m1$

Figure 5: Caption for the picture.

Data frames

Los data frames son estructuras rectangulares de datos que pueden contener objetos de diferente tipo (cadena, numéricos, lógicos, etc.). Creamos un data frame con tres columnas (*id*, *sexo* y *edad*):

```
id = 1:4
sexo = factor(c("male", "male", "female", "female"))
edad = c(15, 26, 43, 56)
df = data.frame(id, sexo, edad)
df
```

```
##   id  sexo edad
## 1  1  male  15
## 2  2  male  26
## 3  3 female  43
## 4  4 female  56
```

Al igual que una matriz, se seleccionan sus entradas usando `[i,j]`. También:

```
df$edad
```

```
## [1] 15 26 43 56
```

```
df$edad[1]
```

```
## [1] 15
```

```
df[c("id", "edad")]
```

```
##   id edad
```

Listas

Las listas son estructuras heterogéneas de datos. Aunque son estructuras unidimensionales, las listas permiten almacenar objetos de distinta clase (vectores, matrices, data frames, otras listas). Así:

```
lista <- list(vector1 = v1,  
             matriz = m1, dataframe = df)  
lista
```

```
## $vector1  
## [1] "a" "b" "c"  
##  
## $matriz  
##      [,1] [,2] [,3]  
## [1,]    1    2    3  
## [2,]    4    5    6  
## [3,]    7    8    9  
##  
## $dataframe  
##   id  sexo edad  
## 1  1  male   15  
## 2  2  male   26  
## 3  3 female  43  
## 4  4 female  56
```

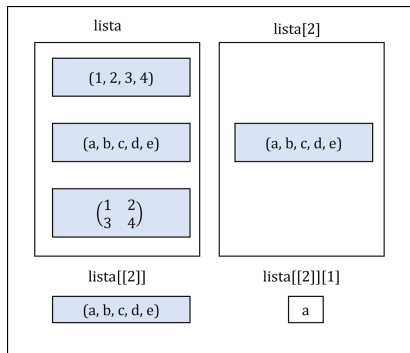


Figure 6: Caption for the picture.

La **Figura X** muestra el uso de los índices en el caso de las listas.

Condicionales

Usando condicionales, una operación es ejecutada si la condición se cumple (TRUE); de otro modo (ELSE), establece la operación que es ejecutada si la condición no se cumple (FALSE). La estructura es esta:

```
if (condition) {  
  # Ejecutado cuando la condición es verdadera  
} else {  
  # Ejecutado cuando la condición es falsa  
}
```

Operator	Description
$x > y$	Greater than
$x \geq y$	Greater than or equal to
$x == y$	Exactly equal to
$!x$	Not
$x != y$	Not equal to
$x y$	OR
$x \& y$	AND
$x \%in\% y$	In the set

Figure 7: Caption for the picture.

Funciones

La estructura general de una función es la siguiente

```
function(arg_1 = x1, arg_2 = x2, ..., arg_n = xn)
```

Usando `function()` se pueden crear funciones a partir de la siguiente estructura:

```
my_function = function(args){  
  #statement  
  #statement  
  return(y)  
}
```

Usando `??` y `help()` se pueden conocer los detalles sobre los argumentos y la salida de las funciones en una librería.

Ejercicio

- 1 Crear una matriz partir de tres vectores (una cadena y dos numéricos). Convertir la matriz en data frame y seleccionar el elemento (2, 3)
- 2 Crear una función que, si la entrada es un vector numérico, la salida sea la media, la mediana, el percentil 25 y 75. (Para la salida use una lista).

Herramientas para la manipulación de datos

Paquetes en R

Los paquetes en R son colecciones de funciones, datos y documentación cuyo objetivo es extender las capacidades básicas de R. CRAN (The Comprehensive R Archive Network) es una red de servidores que almacenan versiones de R, así como librerías en R que cumplen las políticas del repositorio CRAN (CRAN, 2022).

Para instalar paquetes del repositorio **CRAN**:

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

Después de instalar el paquete, se debe cargar la librería:

```
library(dplyr)
```

Para encontrar la documentación del paquete:

```
help(dplyr)
```

Tidyverse

Tidyverse es un conjunto de librerías en R diseñadas para el análisis de datos (importar, transforma, visualizar y modelar con datos) (Wickham, 2019).

Nos concentraremos en las siguientes librerías:

- dplyr
- ggplot2
- forcats*



Figure 8: Caption for the picture.

Importar datos

Definir el directorio de trabajo:

```
setwd("path")
```

Nos concentraremos en funciones para importar los siguientes formatos de datos

Formato	Formato específico	Función	Paquete
Texto o tabulares	CSV	read_csv()	readr
	Otros formatos de texto	read_delim()	readr
Formatos de otros programas	Excel	read_excel()	readxl
	SPSS	read_sav()	haven
	STATA	read_dta()	haven
	SAS	read_sas()	haven
Formatos propios de R	.rda	load()	base
	.rds	readRDS()	base

Pipe (%>%)

Las tuberías de comando u operador *pipe* (%>%) es una herramienta utilizada para el encadenamiento de funciones. El operador nos permite escribir una secuencia de operaciones

```
# Forma estándar:  
dataset_2 <- filter(dataset, attend > 15 & attend != 20)  
  
# Forma encadenada:  
dataset_2 <- dataset %>% filter(attend > 15 & attend != 20)
```

Recuerde el siguiente atajo:



Dplyr

El paquete **dplyr** proporciona una sintaxis para la manipulación de datos. (El operador `%>%` pertenece a la sintaxis de dplyr). Nos concentraremos en las siguientes funciones:

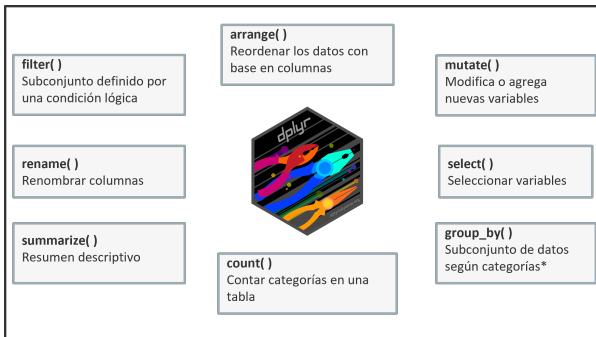


Figure 9: Caption for the picture.

Resumen por grupo

Usando las funciones `summarize()` y `group_by()`, obtenemos un resumen descriptivo de la base de datos diferenciado según una o más variables de control. Por ejemplo:

```
# Resumen general
table_1 <- new_dataset %>% filter(Int_attend == "Group 4")
%>% summarize(MeanAttend = mean(attend), SdAttend = sd(attend))
```

```
# Resumen diferenciado
table_2 <- new_dataset %>% group_by(Int_attend) %>%
  summarize(MeanAttend = mean(attend), SdAttend = sd(attend))
```

La **Figura X** muestra el funcionamiento de `summarize()` y `group_by()`.

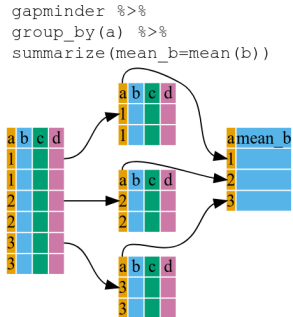


Figure 10: Caption for the picture.

ggplot2

El paquete **ggplot2** proporciona un sistema coherente para visualizar datos y crear gráficos. La versatilidad de **ggplot2** radica en el uso de la Gramática de Gráficos (*Grammar of Graphics*).

```
ggplot(dataset, aes()) + geometría + faceta + opciones
```

donde:

- 1 *dataset* es un data frame
- 2 Las características del mapa **aes()** describe los ejes (x, y), el color exterior (**color** o **colour**), el color interior (**fill**), la forma de los puntos (**shape**), el tipo de línea (**linetype**) y el tamaño (**size**)
- 3 Los objetos geométricos (**geometría**) determinan el tipo de gráfico:
 - Puntos (*geom_point*)
 - Líneas (*geom_lines*)
 - Histogramas (*geom_histogram*)
 - Boxplot (*geom_boxplot*)
- 4 La **faceta** permite dividir un gráfico en múltiples gráficos de acuerdo con grupos

Ejercicios

- 1 Crear un gráfico que el ingreso promedio de acuerdo con la edad en años para las personas ocupados. Presente la gráfica diferenciada por ciudades.
- 2 Filtra la base de datos de personas ocupadas para el grupo de mujeres con ingresos de \$2.000.000 – \$5.000.000. Muestre un gráfico sobre el nivel de educación en esta población.

Recursos alternativos

- La librería `swirl` proporciona un tutorial sobre elementos básicos en R

```
install.packages("swirl")  
library (swirl)  
swirl()
```

- Data wrangling with dplyr and tidyr (Cheat Sheet): [Recurso 1.2](#)
- Visualización de datos usando ggplot2 (Guía Rápida): [Recurso 1.3](#)
- Factors with forcats (Cheat Sheet): [Recurso 1.4](#)

Bibliografía de consulta

Bibliografía de consulta

- Wickham, H. (2016) GGplot2. Elegant Graphics for Data Analysis. Springer
- Golemund, G. (2014). Hands-On Programming with R. O'Reilly Media: Sebastopol, CA.
- Schutt, R. & O'Neil, C. (2014). Doing Data Science. O'Reilly Media: Sebastopol, CA.
- Wickham & Golemund, G. (2016). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media: Sebastopol, CA.