

Fundamentos R -2

Las herramientas del científico de datos

Juan Manuel Moreno — jmmoreno@profesores.imf.com



ÍNDICE

1. Objetivos unidad 4
2. Arrays
3. Listas
4. Factores
5. Matrices
6. Dataframes

01

Objetivos unidad 4

1.– Objetivos Unidad 4

- Conocer qué es el lenguaje de programación R y habituarse al IDE de programación RStudio.
- Aprender sobre la sintaxis básica de R, desde la creación de scripts y notebooks en donde el alumno podrá realizar anotaciones en Markdown, realizar comentarios, conocer los principales operadores, hasta, implementar variables y conocer los diferentes tipos de variables que existen en R.
- Distinguir y saber utilizar las siguientes estructuras de datos en R: Vectores, arrays, factores, listas, matrices y dataframes.
- Saber controlar el flujo de un programa a través de sentencias condicionales y bucles.
- Desarrollar funciones propias, diferenciando entre parámetros de entrada y salida, así como utilizar funciones propias que integra R internamente.



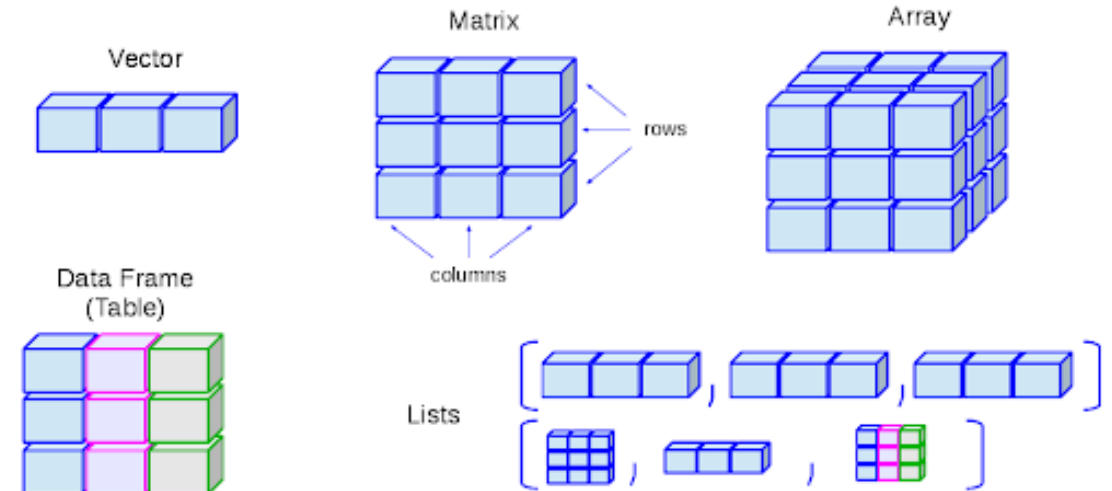
02

Arrays

2.- Arrays

2.1.- Arrays

- Todos sus elementos deben ser del mismo tipo.
- Son indexables
- Se emplea la función **array**.
- Para fijar la longitud de los elementos de las dimensiones de un array se emplea el parámetro **dim**.
- Un array se puede redimensionar, asignando un vector como nuevas dimensiones (en forma de filas x columnas).
- La estructura será la siguientes (**filas**, **columnas**, n **dimensión**)



2.2.– Funciones sobre Arrays

- Cuando seleccionemos una fila o, una columna o, un conjunto de valores de un array, lo que tendremos será un vector, por lo que podremos aplicar cualquier función vista sobre vectores.
- Algunas funciones específicas sobre arrays son:
 - **colSums**: Suma de las columnas de la dimensión n.
 - **rowSums**: Suma de las filas de la dimensión n.
 - **colMeans**: Media de las columnas de la dimensión n.
 - **rowMeans**: Media de las filas de la dimensión n.
 - **aperm**: Permutar la dimensiones de un array.
 - **class**: Devuelve la clase `array`



03

Listas

3.1.– Listas

- Son similares a los vectores.
- Colección de objetos (similar a una columna de un dataframe).
- Se pueden indexar por `[]` o mediante `[[]]`.
- Para acceder a un objeto se puede, acceder desde el operador `$nombre_objeto` o, `[nombre_objeto]`
`[[nombre_objeto]]`
- Para crear una lista se utiliza la función `list()`
- En función del tipo de dato que tengamos almacenado en la lista, podemos aplicar cualquier función de los vectores.



04

Factores


4.1.– Factores

- Tipo especial de vector utilizando exclusivamente para variables categóricas.
- Para crear un vector categórico se emplea **factor()** como parámetro recibe un vector.
- Para conocer el nombre de las categorías (factores), se emplea la función **levels**.
- Si queremos eliminar una categoría utilizamos la función **droplevels**.
- Cualquier vector se puede transformar en categórico con la función **as.factor**

```
var.cat <- factor(c("ENERO", "FEBRERO", "MARZO", "ABRIL"))  
var.cat
```

```
## [1] ENERO  FEBRERO MARZO  ABRIL  
## Levels: ABRIL ENERO FEBRERO MARZO
```

```
var.cat = droplevels(x = var.cat, "MARZO")  
var.cat  
## [1] ENERO  FEBRERO <NA>  ABRIL  
## Levels: ABRIL ENERO FEBRERO
```

05

Matrices

5.1.– Matrices

- Similar a una dimensión de una array en la que nos encontramos una matriz (es decir, filas y columnas)
- Para designar una nueva matriz, utilizamos la función **matrix**.
- Para designar el número de filas el parámetro **nrow** y, el de columnas, **ncol**.

Creamos una matriz.

```
my.mat <- matrix(1:30, ncol = 6, nrow = 5, byrow = F)
```


Probar byrow TRUE

```
my.mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    6   11   16   21   26
## [2,]    2    7   12   17   22   27
## [3,]    3    8   13   18   23   28
## [4,]    4    9   14   19   24   29
## [5,]    5   10   15   20   25   30
```

5.2.– Funciones sobre Matrices

- **ncol**: Devuelve el número de columnas
- **nrow**: Devuelve el número de filas
- **diag**: Devuelve la diagonal de la matriz
- **crossprod**: Realiza el producto entre dos matrices.
- **t**: Devuelve la traspuesta de una matriz.
- **svd**: Valores singulares de una matriz.
- **eigen**: Realiza el cálculo de autovalores y autovectores.
- **as.matrix**: Transforma un vector como matriz



06

Dataframes

6.– Dataframes

6.1.– Dataframes

- Similar a una hoja de Excel o, una base de datos SQL.
- Todos sus elementos tienen un índice común.
- Indexable por filas y columnas.
- Cada columna de un dataframe, se denomina también variable.
- Aceptan diferentes tipos de datos.
- Acceso mediante indexación `[]` o con el operador `$`

The diagram illustrates a DataFrame structure. At the top, the word "Columns" is written in blue, with four arrows pointing down to the column headers: "Name", "Team", "Number", "Position", and "Age". To the left of the table, the word "Rows" is written in orange, with four arrows pointing to the row indices 0, 1, 2, and 3. A purple box labeled "Data" is positioned at the bottom right, with a bracket indicating the data cells for the first row (index 0). The table itself has a light green background and contains the following data:

| | <i>Name</i> | <i>Team</i> | <i>Number</i> | <i>Position</i> | <i>Age</i> |
|---|-----------------|----------------|---------------|-----------------|------------|
| 0 | Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 |
| 1 | John Holland | Boston Celtics | 30.0 | SG | 27.0 |
| 2 | Jonas Jerebko | Boston Celtics | 8.0 | PF | 29.0 |
| 3 | Jordan Mickey | Boston Celtics | NaN | PF | 21.0 |
| 4 | Terry Rozier | Boston Celtics | 12.0 | PG | 22.0 |
| 5 | Jared Sullinger | Boston Celtics | 7.0 | C | NaN |
| 6 | Evan Turner | Boston Celtics | 11.0 | SG | 27.0 |

IMF

6.2.– Leyendo un csv como dataframe

- Además de crear dataframes de cero, trabajaremos principalmente con csv.
- Para leer un dataframe como csv se utiliza la función **read.csv** comúnmente con los siguientes parámetros:
 - **sep**: Tipo de separador de los datos (tabulador, coma, punto y coma, otros...)
 - **header**: Si el dataframe tiene cabecera o no.
 - **na.strings**: Indicar si hay un string que comúnmente se emplee como valor para designar valores nulos.

```
df <- read.csv("FB.csv")
```


6.3.– Funciones sobre dataframes

- **class**: Para saber la clase (tipo) del dataframe.
- **attach / detach**: Permite integrar todas las columnas del dataframe como variables sin necesidad de llamar al dataframe.
- **summary**: Muestra el resumen estadístico
- **str**: Devuelve el tipo de variables involucradas en el dataframe.
- **attributes**: Devuelve el nombre de las filas y columnas.
- **dim**: Dimensiones del dataframe en filas x columnas.
- **colnames**: Nombre de las columnas de un dataframe.
- **rownames**: nombre de las filas de un dataframe.
- **nrow**: N° filas.
- **ncol**: N° columnas.
- **head**: Primeras posiciones del dataframe.

6.4.– Funciones sobre dataframes

- **head**: Primeras posiciones del dataframe.
- **tail**: Últimas posiciones del dataframe.
- **rbind**: Añadir una nueva fila (También se puede por índice)
- **cbin**: Añadir una nueva columna (También se puede por índice)
- **<- NULL**: eliminar una variable
- **subset**: Obtener un subconjunto del dataframe
- **order**: Ordenar dataframe por una columna.
- **names**: Nombres de las columnas.
- **as.data.frame**: Convertir otra estructura de datos como dataframe.
- También es posible utilizar funciones de **arrays** y de **vectores**

6.5.– Gestión de nulos

- Comenzar con un resumen estadístico (**summary**).
- A través de la función **is.na()**
- Detectar las filas en las que hay valores nulos con **complete.cases**.
- **<- NA**: Se asigna como valor nulo.
- Para **borrar** nulos se puede realizar:
 - `nombre_df[complete.cases(nombre_df),]`
- Para **reemplazar** un valor nulo por otro valor:
 - `nombre_df $nombre_columna[which(is.na(nombre_df $ nombre_columna))] <- valor de reemplazo`

Seguimiento práctico del contenido

A partir de aquí, vamos arrays, matrices, factores, listas y dataframes con.

4_4_Estructuras_datos_2.RMD

IMF

Smart Education