Práctica 3. Introducción y manipulación de datos en R (2)

Jesús Martín Fernández

Contenidos

1. Data frame

Un **data frame** en R es una estructura de datos bidimensional que, a primera vista, se parece a una matriz, pero es más versátil. Mientras que una matriz solo puede contener datos del mismo tipo en todas sus celdas, un data frame permite que diferentes columnas tengan distintos tipos de datos, como numéricos, caracteres o lógicos. Esto hace que los data frames sean ideales para análisis estadísticos y manipulación de datos donde se necesitan múltiples tipos de variables, en contraste con las matrices, que son más adecuadas para operaciones matemáticas homogéneas.

1.1. Creación de un dataframe.

Los dataframe pueden crearse de diferentes formas.

La primera de ella es a partir de un número de vectores de igual longitud

```r
n<-10
v1 <- sample(c("Varón", "Mujer"), size = n, replace = TRUE)
v2 <- sample(c("Sí", "No"), size = n, replace = TRUE)
v3 <- sample(102:162, size = n, replace = TRUE)
v4 <- sample(60:96, size = n, replace = TRUE)
```

Ahora construimos el dataframe

```r
df <- data.frame(v1, v2, v3, v4)
df
```

```
      v1 v2  v3 v4
1  Mujer Sí 159 91
2  Varón Sí 120 67
3  Mujer No 119 83
4  Varón Sí 127 80
5  Mujer Sí 148 63
6  Mujer Sí 126 81
7  Varón Sí 140 94
8  Mujer No 142 93
9  Varón No 116 81
10 Varón Sí 162 74
```

Se pueden etiquetar las columnas

```r
df <- data.frame(Sexo = v1, Fumador = v2, TAS = v3, TAD = v4)
df
```

```
    Sexo Fumador TAS TAD
1  Mujer      Sí 159  91
2  Varón      Sí 120  67
3  Mujer      No 119  83
4  Varón      Sí 127  80
5  Mujer      Sí 148  63
6  Mujer      Sí 126  81
7  Varón      Sí 140  94
8  Mujer      No 142  93
9  Varón      No 116  81
10 Varón      Sí 162  74
```

La misma acción podría hacerse con la función `colnames`

```
colnames(df) <- c("Sexo", "Fumador", "TAS", "TAD")
df
```

```
    Sexo Fumador TAS TAD
1  Mujer      Sí 159  91
2  Varón      Sí 120  67
3  Mujer      No 119  83
4  Varón      Sí 127  80
5  Mujer      Sí 148  63
6  Mujer      Sí 126  81
7  Varón      Sí 140  94
8  Mujer      No 142  93
9  Varón      No 116  81
10 Varón      Sí 162  74
```

A este dataframe se le puede añadir una columna

```
DM<- c("No", "Sí","No","No","Si","No","No","Si","No","No")
df1<- data.frame(df,DM)
df1
```

```
    Sexo Fumador TAS TAD DM
1  Mujer      Sí 159  91 No
2  Varón      Sí 120  67 Sí
3  Mujer      No 119  83 No
4  Varón      Sí 127  80 No
5  Mujer      Sí 148  63 Si
6  Mujer      Sí 126  81 No
7  Varón      Sí 140  94 No
8  Mujer      No 142  93 Si
9  Varón      No 116  81 No
10 Varón      Sí 162  74 No
```

Se conseguiría el mismo resultado con la función `cbind`

```
df1<-cbind (df, DM)
df1
```

```
    Sexo Fumador TAS TAD DM
1  Mujer      Sí 159  91 No
2  Varón      Sí 120  67 Sí
3  Mujer      No 119  83 No
4  Varón      Sí 127  80 No
5  Mujer      Sí 148  63 Si
6  Mujer      Sí 126  81 No
7  Varón      Sí 140  94 No
8  Mujer      No 142  93 Si
9  Varón      No 116  81 No
10 Varón      Sí 162  74 No
```

También se podría añadir una fila (debe tener las mismas variables) con la función `rbind`

```
n.file<- c("Varón", "No", 136, 78, "No")
df2<- rbind (df1,n.file)
df2
```

```
    Sexo Fumador TAS TAD DM
1  Mujer      Sí 159  91 No
2  Varón      Sí 120  67 Sí
3  Mujer      No 119  83 No
4  Varón      Sí 127  80 No
5  Mujer      Sí 148  63 Si
6  Mujer      Sí 126  81 No
7  Varón      Sí 140  94 No
8  Mujer      No 142  93 Si
9  Varón      No 116  81 No
10 Varón      Sí 162  74 No
11 Varón      No 136  78 No
```

Vamos a guardar df2, como archivo .csv, para luego seguir trabajando con él

```
write.csv(df2, "df2.csv", row.names = FALSE)
```

Un dataframe puede obtenerse a partir de otra. Vamos a obtener nuevas dataframes a partir de `df2´`. Imaginemos que queremos obtener `df3` eliminando las filas 1 a 5. Puede hacerse de varios modos

```
df3<- df2[-c(1,2,3,4,5), ]
df3
```

```
    Sexo Fumador TAS TAD DM
6  Mujer      Sí 126  81 No
7  Varón      Sí 140  94 No
8  Mujer      No 142  93 Si
9  Varón      No 116  81 No
10 Varón      Sí 162  74 No
11 Varón      No 136  78 No
```

```
df3<- df2[6:11, ]
df3
```

```
    Sexo Fumador TAS TAD DM
6  Mujer      Sí 126  81 No
7  Varón      Sí 140  94 No
8  Mujer      No 142  93 Si
9  Varón      No 116  81 No
10 Varón      Sí 162  74 No
11 Varón      No 136  78 No
```

También puede crearse un dataframe a partir de otro eliminando columnas. Probemos a obtener `df4` desde `df3` eliminando las columnas TAD y DM

```
df4<- df3[ , -c(4:5)]
df4
```

```
    Sexo Fumador TAS
6  Mujer      Sí 126
7  Varón      Sí 140
8  Mujer      No 142
9  Varón      No 116
10 Varón      Sí 162
11 Varón      No 136
```

1.2 Importación de un dataframe

En muchas ocasiones importamos a R bases de datos u hojas de cálculo creados con otras herramientas. Vamos a ver cómo se hacen estas operaciones.

- Importación de una base de datos CSV (Comma-Separated Values).

Se hace con la función `read.csv`

df <- read.csv("ruta/del/archivo.csv")

- Importación desde un archivo de texto (.txt)

Se usa la función read.table

df <- read.table("ruta/del/archivo.txt", header = TRUE, sep = "\")

Le estamos diciendo que lea la primera fila como encabezados y que use el separador con la instrucción `sep`

- Importación desde una hoja de cálculo excel

Primero debe instalarse la librería `readxl`

library(readxl) df <- read_excel("ruta/del/archivo.xlsx", sheet = 1)

La instrucción sheet permite elegir la hoja que leerá.

- Importación desde Stata (.dta) y SPSS (.sav)

Primero debe instalarse la librería `haven` (la librería `foreign` se usaba para versiones más antiguas de Stata y SPSS)

library(haven) df <- read_dta("ruta/del/archivo.dta")

library(haven) df <- read_sav("ruta/del/archivo.sav")

1.3 Operando con un dataframe

Primero borramos todas las variables creadas en la sesión

```
rm (list=ls())
```

Cargamos df_2, recordamos que es un archivo .csv que hemos guardado en nuestro mismo directorio de trabajo. Si no fuese así, fijaríamos nuestro directorio de trabajo primero.

```
df2 <- read.csv("df2.csv")
```

Vamos a hacer algunas operaciones básicas con el `df2`

Reordenar el dataframe en función del orden de una variable. Lo haremos con la bas `df2` y la función `order`

```
#df2<-df2[order(df2$Sexo, -df2$TAS), ]
#Ordena df2 primero por el sexo en orden creciente y luego por la TAS en orden decreciente
```

Da un error porque TAS se convirtió en `character` cuando usamos la función `rbind` . Así que debemos cambiar TAS a variable numérica `numeric`

```
class(df2$TAS)
```

```
[1] "integer"
```

```
df2$TAS <- as.numeric (df2$TAS)

df2_sorted <- df2[order(df2$Sexo, -df2$TAS), ]

df2_sorted
```

```
    Sexo Fumador TAS TAD DM
1  Mujer      Sí 159  91 No
5  Mujer      Sí 148  63 Si
8  Mujer      No 142  93 Si
6  Mujer      Sí 126  81 No
3  Mujer      No 119  83 No
10 Varón      Sí 162  74 No
7  Varón      Sí 140  94 No
11 Varón      No 136  78 No
4  Varón      Sí 127  80 No
2  Varón      Sí 120  67 Sí
9  Varón      No 116  81 No
```

- Seleccionar un subgrupo de casos o variables

Ya se mencionó como seleccionar casos y variables para construir un nuevo dataframe a partir de otro antiguo.

Pero hay otra formas de sleeccionar partes de un dataframe. Volvemos a usar `df2`y la función `subset`

```
df2
```

```
    Sexo Fumador TAS TAD DM
1  Mujer      Sí 159  91 No
2  Varón      Sí 120  67 Sí
3  Mujer      No 119  83 No
4  Varón      Sí 127  80 No
5  Mujer      Sí 148  63 Si
6  Mujer      Sí 126  81 No
7  Varón      Sí 140  94 No
8  Mujer      No 142  93 Si
9  Varón      No 116  81 No
10 Varón      Sí 162  74 No
11 Varón      No 136  78 No
```

```
df2_h <- subset(df2,  Sexo == "Varón")
# Condición: selección de los "Hombres" en la variable "Sexo".
df2_h
```

```
    Sexo Fumador TAS TAD DM
2  Varón      Sí 120  67 Sí
4  Varón      Sí 127  80 No
7  Varón      Sí 140  94 No
9  Varón      No 116  81 No
10 Varón      Sí 162  74 No
11 Varón      No 136  78 No
```

La misma función sirve para seleccionar solo algunas variables de un dataset

```
df2_h2 <- subset(df2_h, select = c("TAS","TAD"))
df2_h2
```

```
   TAS TAD
2  120  67
4  127  80
7  140  94
9  116  81
10 162  74
11 136  78
```

En realidad esa función no difiere de la ya explicada

```r
df2_h2 <- df2_h[, c("TAS", "TAD")]
df2_h2
```

```
   TAS TAD
2  120  67
4  127  80
7  140  94
9  116  81
10 162  74
11 136  78
```

- Creación de una nueva variable

En el dataframe `df2` vamos a crear la variable `HTA` ("Si"/"No") cuando TAS>=140 o TAD>=90

```r
df2$HTA[df2$TAS >= 140 | df2$TAD >=90] <- "Sí"
df2$HTA[df2$TAS < 140 & df2$TAD <90] <- "No"
df2
```

```
    Sexo Fumador TAS TAD DM HTA
1  Mujer      Sí 159  91 No  Sí
2  Varón      Sí 120  67 Sí  No
3  Mujer      No 119  83 No  No
4  Varón      Sí 127  80 No  No
5  Mujer      Sí 148  63 Si  Sí
6  Mujer      Sí 126  81 No  No
7  Varón      Sí 140  94 No  Sí
8  Mujer      No 142  93 Si  Sí
9  Varón      No 116  81 No  No
10 Varón      Sí 162  74 No  Sí
11 Varón      No 136  78 No  No
```

- Recodificación de una variable

Se pueden atribuir nuevos valores a una variable por ejemplo para codificarla. A partir de `df2` añadimos la variable Edad , creando `df2_E`

```r
Edad<-c(38,48,52,60,70,49,52,72,68,61,59)
class (Edad)
```

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

```r
df2_E<-data.frame (df2, Edad)
str (df2_E)
```

```
'data.frame':   11 obs. of  7 variables:
 $ Sexo   : chr  "Mujer" "Varón" "Mujer" "Varón" ...
 $ Fumador: chr  "Sí" "Sí" "No" "Sí" ...
 $ TAS    : num  159 120 119 127 148 126 140 142 116 162 ...
 $ TAD    : int  91 67 83 80 63 81 94 93 81 74 ...
 $ DM     : chr  "No" "Sí" "No" "No" ...
 $ HTA    : chr  "Sí" "No" "No" "No" ...
 $ Edad   : num  38 48 52 60 70 49 52 72 68 61 ...
```

Vamos a categorizar la Edad en tres categorías usando los puntos de corte de 50 y 65 años

```r
class (df2_E$Edad)
```

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

```r
df2_E$Edad[df2_E$Edad > 64] <- "65 y más años"
df2_E$Edad[df2_E$Edad > 50 & df2_E$Edad < 65] <- "Entre 50 y 64 años"
df2_E$Edad[df2_E$Edad <50] <- "Menor de 50 años"
# Se podría hacer también con la función `cut`
#df2_E$Edad <- cut(df2_E$Edad, breaks = c(-Inf, 50, 65, Inf), #
# Asignar etiquetas a los intervalos
#labels = c("Menor de 50 años", "Entre 50 y 64 años", "65 y más años"))
df2_E
```

```
    Sexo Fumador TAS TAD DM HTA               Edad
1  Mujer      Sí 159  91 No  Sí   Menor de 50 años
2  Varón      Sí 120  67 Sí  No   Menor de 50 años
3  Mujer      No 119  83 No  No Entre 50 y 64 años
4  Varón      Sí 127  80 No  No Entre 50 y 64 años
5  Mujer      Sí 148  63 Si  Sí      65 y más años
6  Mujer      Sí 126  81 No  No   Menor de 50 años
7  Varón      Sí 140  94 No  Sí Entre 50 y 64 años
8  Mujer      No 142  93 Si  Sí      65 y más años
9  Varón      No 116  81 No  No      65 y más años
10 Varón      Sí 162  74 No  Sí Entre 50 y 64 años
11 Varón      No 136  78 No  No Entre 50 y 64 años
```

Finalmente, vamos a cambiar el nombre a una variable de `df2_E` . Vamos a cambiar el nombre de la variable Tabaco por Fumador.

```
names(df2_E)[names(df2_E) == "Fumador"] <- "Tabaco"
df2_E
```

```
    Sexo Tabaco TAS TAD DM HTA                Edad
1  Mujer     Sí 159  91 No  Sí   Menor de 50 años
2  Varón     Sí 120  67 Sí  No   Menor de 50 años
3  Mujer     No 119  83 No  No Entre 50 y 64 años
4  Varón     Sí 127  80 No  No Entre 50 y 64 años
5  Mujer     Sí 148  63 Si  Sí      65 y más años
6  Mujer     Sí 126  81 No  No   Menor de 50 años
7  Varón     Sí 140  94 No  Sí Entre 50 y 64 años
8  Mujer     No 142  93 Si  Sí      65 y más años
9  Varón     No 116  81 No  No      65 y más años
10 Varón     Sí 162  74 No  Sí Entre 50 y 64 años
11 Varón     No 136  78 No  No Entre 50 y 64 años
```

En la siguiente tabla resumimos algunas de las funciones utilizadas en el dataframe y otras básicas para un dataframe genérico df

Función	Descripción	Ejemplo	Resultado
head	Muestra las primeras filas del dataframe	head(df)	Primeras 6 filas del dataframe df
tail	Muestra las últimas filas del dataframe	tail(df)	Últimas 6 filas del dataframe df
nrow	Devuelve el número de filas del dataframe	nrow(df)	Número total de filas en df
ncol	Devuelve el número de columnas del dataframe	ncol(df)	Número total de columnas en df
dim	Devuelve un vector con el número de filas y columnas	dim(df)	Dimensiones del dataframe (filas, columnas)
summary	Muestra un resumen estadístico de cada columna del dataframe	summary(df)	Resumen estadístico de las columnas de df
str	Muestra la estructura interna del dataframe	str(df)	Estructura y tipo de datos en df
colnames	Devuelve o establece los nombres de las columnas	colnames(df)	Nombres de las columnas de df

Función	Descripción	Ejemplo	Resultado
rownames	Devuelve o establece los nombres de las filas	rownames(df)	Nombres de las filas de df
subset	Extrae subconjuntos del dataframe basados en condiciones	subset(df, column > value)	Subconjunto de df donde column > value
df[column]	Selecciona una columna específica del dataframe	df$column	Vector con los datos de column
df[row, col]	Selecciona un elemento o subconjunto específico del dataframe	df[1, 2]	Elemento en la primera fila y segunda columna
df[, col]	Selecciona todas las filas para una columna específica	df[, 2]	Todos los datos de la segunda columna
df[row,]	Selecciona todas las columnas para una fila específica	df[1,]	Todos los datos de la primera fila
apply	Aplica una función a lo largo de las filas o columnas	apply(df, 2, mean)	Media de cada columna de df
rbind	Añade filas al dataframe	rbind(df, new_row)	Nuevo dataframe con new_row añadido
cbind	Añade columnas al dataframe	cbind(df, new_column)	Nuevo dataframe con new_column añadido
merge	Combina dos dataframes basados en una clave común	merge(df1, df2, by = “key”)	Dataframe combinado de df1 y df2
order	Ordena un dataframe según una o más columnas	df[order(df$column),]	Dataframe df ordenado por column
na.omit	Elimina filas con valores NA del dataframe	na.omit(df)	Dataframe df sin filas con valores NA

2. Array

Un array en R es una estructura de datos multi-dimensional que permite almacenar y manipular datos en más de dos dimensiones, extendiendo así las capacidades de los vectores y matrices.

Los arrays son útiles cuando se necesita trabajar con datos que tienen más de dos dimensiones,

como por ejemplo en análisis de series temporales, datos espaciales, o cualquier situación en la que se manejen múltiples capas de datos relacionados entre sí.

Vamos a ver un ejemplo de creación de un array en el que archivamos las temperaturas diarias en tres ciudades distintas, tomadas en dos momentos del día

```r
# Temperaturas diarias para 3 ciudades durante 7 días
# Cada ciudad tiene 2 temperaturas al día (mañana y tarde)
temperaturas <- c(
  # Ciudad 1: (mañana, tarde para 7 días)
  20, 25,  # Día 1
  22, 27,  # Día 2
  24, 29,  # Día 3
  26, 31,  # Día 4
  28, 33,  # Día 5
  30, 35,  # Día 6
  32, 37,  # Día 7

  # Ciudad 2: (mañana, tarde para 7 días)
  18, 23,  # Día 1
  19, 24,  # Día 2
  21, 26,  # Día 3
  22, 27,  # Día 4
  24, 29,  # Día 5
  25, 30,  # Día 6
  27, 32,  # Día 7

  # Ciudad 3: (mañana, tarde para 7 días)
  28, 33,  # Día 1
  29, 34,  # Día 2
  30, 35,  # Día 3
  32, 37,  # Día 4
  33, 38,  # Día 5
  35, 40,  # Día 6
  37, 41   # Día 7
)

# Crear el array
# Dimensiones: 7 días, 3 ciudades, 2 períodos del día (mañana y tarde)
temp_array <- array(temperaturas, dim = c(7, 3, 2))

# Asignar nombres a las dimensiones para mayor claridad
dimnames(temp_array) <- list(
```

```r
  Días = paste("Días", 1:7),         # Nombres para los días
  Ciudad = paste("Ciudad", 1:3),      # Nombres para las ciudades
  Periodo = c("Mañana", "Tarde") )

# Mostrar el array
print(temp_array)
```

```
, , Periodo = Mañana

        Ciudad
Días    Ciudad 1 Ciudad 2 Ciudad 3
  Días 1       20       31       18
  Días 2       25       28       23
  Días 3       22       33       19
  Días 4       27       30       24
  Días 5       24       35       21
  Días 6       29       32       26
  Días 7       26       37       22

, , Periodo = Tarde

        Ciudad
Días    Ciudad 1 Ciudad 2 Ciudad 3
  Días 1       27       28       37
  Días 2       24       33       33
  Días 3       29       29       38
  Días 4       25       34       35
  Días 5       30       30       40
  Días 6       27       35       37
  Días 7       32       32       41
```

3. Lista

Una lista es una estructura de datos muy flexible y versátil que permite almacenar elementos de diferentes tipos y tamaños. A diferencia de los vectores y matrices, que requieren que todos sus elementos sean del mismo tipo, las listas pueden contener elementos de tipos variados, incluyendo números, cadenas de texto, vectores, matrices, dataframes e incluso otras listas.

Las listas permiten realizar manipulaciones de datos complejas. Puedes acceder a cada componente de una lista de forma individual y realizar operaciones específicas sobre ellos.

Algunas funciones en R, especialmente en paquetes de análisis y modelado, devuelven resultados en forma de listas. Estas listas pueden contener múltiples componentes, como coeficientes del modelo, estadísticas, gráficos y otros resultados.

Veamos un ejemplo básico de creación de una lista y a mostrar el resultado

```r
lista <- list(
  Nombres = c("Carlos", "Ana"),
  # Vector con nombres de los sujetos

  Edades = c(25, 30),
  # Vector con edades de los sujetos
  Calificaciones = list(
    c(88, 92, 75),          # Primer vector de calificaciones
       c(90, 85, 80)        # Segundo vector de calificaciones
      )
)

# Mostrar la lista
print(lista)
```

```
$Nombres
[1] "Carlos" "Ana"

$Edades
[1] 25 30

$Calificaciones
$Calificaciones[[1]]
[1] 88 92 75

$Calificaciones[[2]]
[1] 90 85 80
```