Creación de marcos de datos

Data frames

Autor: Luis Javier Sánchez Martínez
luisja02@ucm.es

Departamento de Biodiversidad, Ecología y Evolución
Unidad de Antropología Física.
Universidad Complutense de Madrid (UCM)



Introducción 2
Crear un data.frame 2
Acceder a los elementos de un data.frame 3
¿Qué es una lista?.....3
Cómo matriz bidimensional……..5
Cómo lista…….5
Las funciones attach() y detach() 6
Seleccionar casos, variables y transformar variables 7
La función subset()…….....7
La función transform()……...7
Añadir una nueva variable 8
Las funciones with() y within()……..8
Eliminar una variable (columna) 10
Ordenar un data frame 12
Combinar data frames 16
La función merge(): Combinar dos data frame……..……...16
Combinar más de dos data frame. La función Reduce() 21
Eliminar filas duplicadas en un data frame 22
Funciones sobre data.frames 23
La función sapply()…….....23
La función tapply()……23
La función by()…….....24
¿Practicamos un poco? 26
Solucionario 28

Introducción

Un data frame es un objeto de R que permite almacenar datos bidimensionales. Tienen en común con las matrices que son objetos rectangulares donde todas las columnas deben tener la misma longitud, pero a diferencias de ellas cada columna puede ser de tipo numérico, carácter o lógico. De igual forma, pueden verse como una lista donde todos sus elementos deben tener la misma longitud, es decir al igual que las matrices están limitadas a que todas las columnas tengan la misma cantidad de datos.

Se puede ver como un archivo de datos rectangular donde las filas son los casos y las columnas son las variables que se miden en cada caso. Un data frame es equivalente a un fichero de datos de SPSS o de SAS. Normalmente la primera columna suele contener los nombres de los individuos o filas y la primera fila los nombres de las variables.

Las columnas (variables) no numéricas, en un data frame, son consideradas como factores, con tantos niveles como valores distintos haya (en los datos, salvo que se indique otra cantidad), aunque algunas veces conviene revisar que estas variables estén en el formato adecuado.

Estos materiales incluyen código de R como ejemplos, los cuales siempre aparecen con este color, os animamos a copiarlos y ejecutarlos en vuestra consola para poder jugar con ello y comprobar como cambian los resultados.

Crear un data frame

Un data frame puede crearse de diferentes formas, entre ellas se destacan:

- Por medio de la función data.frame().
- Leyendo un archivo de datos.

son las dos formas más comunes de obtener un data frame.

Ejemplo 1:

```
# Se crea el vector edad por medio de la función c().
EDAD<-c(12,15,17,10,45)

# Se crea el vector CASO por medio de la función seq()
CASO<- seq(1,5)

# Se crea el factor SEXO por medio de la función rep().
SEXO<-as.factor(rep(x=c("M","H"), times=c(2,3)))

# Creamos el data frame por medio de la función data.frame().
```

```
df<-data.frame(edad=EDAD, caso=CASO, sexo=SEXO)
df
  edad caso sexo
1   12    1    M
2   15    2    M
3   17    3    H
4   10    4    H
5   45    5    H
```

Ejemplo 2:

```
resultados <- data.frame(
        talla = c(185,172,165,177,168),
        peso  = c( 82, 70, 76, 52, 72),
        QI    = c(110,120, 92,120,142),
        sexo  = c("H","M","M", "M","H"),
        row.names= c("Pablo", "Marta","Lola","Genoveva", "Luis"))
resultados
        talla peso  QI sexo
Pablo     185   82 110    H
Marta     172   70 120    M
Lola      165   76  92    M
Genoveva  177   52 120    M
Luis      168   72 142    H
```

Acceder a los elementos de un data.frame

Los data frame comparten características con las matrices y con las listas. Para extraer elementos de las tablas de datos se pueden referenciar sus elementos como si fuese una matriz y como una lista.

¿Qué es una lista?

Una lista en R es una estructura ordenada de objetos de R, estos objetos pueden ser vectores, matrices, otras listas (incluso data frame).

Técnicamente una lista es un vector pero a diferencia del vector cada elemento de la lista puede ser cualquier tipo de objeto de R. Esto permite que cada objeto sea de un tipo (mod y class) cualquiera y tenga cualquier longitud.

Ejemplo:

```
 L <- list (matrix(1:6,ncol=2),     # Matriz numérica
c(T,F),              # vector booleano
c("Buenos", "Días"))                # Vector de caracteres

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

 [[2]]
 [1]  TRUE FALSE

 [[3]]
 [1] "Buenos" "Días"
```

Los miembros de una lista están indexados y puede accederse a ellos. En el caso de las listas la indexación se realiza por medio de dobles corchetes [[]].
De esta forma el primer elemento es una matriz:

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

 class(L[[1]])
 [1] "matrix"
```

Una vez que las posiciones de una lista tiene nombre se puede hacer referencia a los miembros de la lista por medio de los nombres o de los dobles corchetes. El operador $ es equivalente al doble corchete [[ ]] cuando existen los nombres:

```
 L <- list (matrix(1:6,ncol=2), c(T,F),c("Buenos", "Días"))
 names(L) <-c("datos","respuestas","saludo")
 L
```

```
$datos
  [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

$respuestas
[1]  TRUE FALSE

$saludo
[1] "Buenos" "Días"

L$respuestas
[1]  TRUE FALSE
```

- **Como matriz bidimensional**

Los elementos se pueden extraer utilizando las referencias de una matriz bidimensional.

```
resultados[2,3]
 [1] 120
resultados[4,]
                talla peso  QI sexo
 Genoveva   177   52 120    M

resultados[4, c(2,4)]
                peso sexo
 Genoveva   52    M
```

- **Como lista**

Para trabajar con las variables (columnas) de un data.frame se puede utilizar la notación estándar de las listas,

```
nombre_data_frame$nombre_columna
nombre_data_frame[[ ]]
```

Ejemplo:

```
df <- data.frame(id=c(1,2,3,4,5),
        nombre=c("Pilar","Nuria","José","Luis","Javier"),
```

```
            edad=c(30,40,22,53,27),
            peso=c(75,60,55,78,80))
df[3]
 edad
1   30
2   40
3   22
4   53
5   27

 df$edad
[1] 30 40 22 53 27
```

Las funciones attach() y detach()

La acción de vincular los nombres de las columnas con sus nombres como vectores se realiza por medio de la función attach(). En este ejemplo, para mostrar los nombres del data frame se ejecuta la sentencia attach(df).

Ejemplo:

```
attach(df)
names(df)
[1] "id"     "nombre" "edad"   "peso"
(edad - mean(edad))/sd(edad)
[1] -0.3577118  0.4552695 -1.0080968  1.5121451 -0.6016061
detach(df)
```

IMPORTANTE

Una vez vinculado un data frame por medio de la función attach() las modificaciones hechas a las columnas por medio de las referencia de sus nombres no afectan al data frame original. A efectos prácticos, al ejecutar attach(df), R realiza una copia temporal del data frame df y es en esta copia donde se trabaja.

Por ejemplo, si se ejecuta:

```
attach(df)
edad <- 2*edad
edad
[1]  60  80  44 106  54
detach(df)
```

```
df$edad
[1] 30 40 22 53 27
```

Se puede apreciar que el data frame df no ha sido modificado. Para deshacer la vinculación se ejecuta la función detach().

Seleccionar casos, variables y transformar variables

Hay dos formas de seleccionar casos y variables en un data frame. La primera es utilizar las técnicas usuales de matrices, la selección por índices. La segunda es utilizar las funciones: subset() y transform(). Aunque el uso de las funciones resulta más cómodo, la selección por medio de índices es más poderosa.

- **La función subset()**

Permite seleccionar casos (filas).

Su sintaxis es:

```
subset (data.frame, condición lógica, select=vector con nombres de variables)
```

Ejemplo:

```
df.2 <- subset(df,edad>30,select=c(edad,peso))
df.2
  edad peso
2   40   60
4   53   78
```

- **La función transform()**

Para transformar unas determinadas columnas, o crear nuevas variables, se puede utilizar la función transform().

Su sintaxis es:

```
transform (data.frame, variable 1 = f(variables), variable2= f(variables)...)
```

Ejemplo:

```
df.3 <- transform(df,edad=edad*2)
df.3
  id nombre edad peso
1  1  Pilar   60   75
```

```
2  2  Nuria   80   60
3  3   José   44   55
4  4   Luis  106   78
5  5 Javier   54   80
```

Para realizar la misma operación sin la función transform() bastaría con escribir:

```
df$edad <-  df$edad*2
```

Añadir una nueva variable.

La primera manera consiste en usar la función transform()

Ejemplo:

```
df.3 <- transform(df, edad2 = edad+23)
df.3
  id nombre edad peso edad2
1  1  Pilar   30   75    53
2  2  Nuria   40   60    63
3  3   José   22   55    45
4  4   Luis   53   78    76
5  5 Javier   27   80    50
```

Una segunda manera sería sin emplear la función transform()

Ejemplo:

```
edad.n <- df$edad + 23 # Construimos la variable
df$edad.nueva <- edad.n          #Añadimos la variable por medio de la asignación
df
  id nombre edad peso edad.nueva
1  1  Pilar   30   75         53
2  2  Nuria   40   60         63
3  3   José   22   55         45
4  4   Luis   53   78         76
5  5 Javier   27   80         50
```

- **Las funciones with() y within()**

Otras opciones para transformar variables son utilizar las funciones with() y within() que permiten realizar transformaciones múltiples en un data frame.

Ejemplos:

```
N <- 3
df <- data.frame(x1=rnorm(N),x2=runif(N))
df$y <- with(df,{x1+x2})
df
         x1        x2         y
 1  0.7614611 0.49108331  1.2525444
 2 -1.1097564 0.01768844 -1.0920680
 3 -1.1552285 0.27559108 -0.8796374


df <- within(df,{
            x1.sq <- x1^2
             x2.sq <- x2^2
            y <- x1.sq+x2.sq
            x1 <- x2 <- NULL
           })
df
         y       x2.sq    x1.sq
 1 0.8209858 0.2411628164 0.579823
 2 1.2318721 0.0003128808 1.231559
 3 1.4105033 0.0759504421 1.334553
```

La diferencia fundamental entre ambas funciones with() y within() es que la primera de ellas no crea una copia de los datos, mientras que la segunda sí lo hace.

Ejemplo:

```
data <- data.frame(a=1:5,b=2:6)
data
  a b
1 1 2
2 2 3
3 3 4
4 4 5
5 5 6
with(data, {c <- a + b} )
  a b
1 1 2
```

```
2 2 3
3 3 4
4 4 5
5 5 6
data$c<-with(data, a + b)
data
  a b  c
1 1 2  3
2 2 3  5
3 3 4  7
4 4 5  9
5 5 6 11

data <- data.frame(a=1:5,b=2:6)
data
  a b
1 1 2
2 2 3
3 3 4
4 4 5
5 5 6
within(data, {c <- a + b} )
  a b  c
1 1 2  3
2 2 3  5
3 3 4  7
4 4 5  9
5 5 6 11
```

Eliminar una variable (columna)

Una variable de un data frame se puede eliminar directamente referenciándola por el número de columna que se desea eliminar con signo negativo: df [, -n]

Ejemplo:

```
misdatos <- data.frame(x=1:10, y=rnorm(10))
misdatos
    x         y
```

```
 1   1 -0.4789838
 2   2  0.3961842
 3   3 -0.1979678
 4   4  0.1526418
 5   5 -1.8517416
 6   6  0.6143180
 7   7 -2.0995205
 8   8  1.5929754
 9   9 -1.6747250
 10 10 -0.7459261
```

```
misdatos[, -1]
 [1]  0.93842912  0.01127989  1.23385977 -1.00448420  1.06895713 -0.04259880
 [7] -1.82802718 -0.98789726  0.90532043  0.53228974
```

```
misdatos <- data.frame(y = c(misdatos[, -1]))
misdatos
```

```
            y
1   0.70206657
2   0.42575240
3   0.70186797
4  -0.01788541
5  -0.62441418
6   0.58564147
7  -0.81397484
8   0.57030558
9   2.06918011
10  0.46480954
```

Por medio de NULL no se pierde la estructura de data frame:

```
misdatos <- data.frame(x=1:10, y=rnorm(10))
misdatos[,1] <- NULL
misdatos
```

```
            y
```

1 0.95619653

2 -0.68629251

3 -0.92055556

4 0.24672101

5 -2.35939339

6 -1.75098978

7 -0.09332705

8 -1.00329515

9 0.46188410

10 0.88133306

Ordenar un data frame

Para ordenar un data frame se dispone de diferentes opciones. La principal es ordenar los índices de las filas en función de una o más columnas por medio de la función order().

La función order() devuelve un vector de índices permutados, de tal forma que sus argumentos quedan ordenados de forma ascendente. Si hay más de un vector, en caso de empate en el primer vector se utilizan los valores del segundo. Si persiste el empate se miraría el tercero, y así sucesivamente.

Su sintaxis:

```
order(..., na.last = TRUE, decreasing = FALSE)
sort.list(x, partial = NULL, na.last = TRUE, decreasing = FALSE,
        method = c("shell", "quick", "radix"))
```

donde:

… , Una secuencia de vectores numéricos, complejos, de caracter o lógicos todos de igual longitud o un objeto de clase R.

x Un vector.

partial , Vector de índices para una ordenación parcial. (Los valores nulos no se utilizan)

decreasing , (T/F). Determina si el orden es creciente o decreciente.

na.last , Controla el tratamiento de los NAs. Si es TRUE, los valores missing se colocan al final de los datos, si es FALSE, se colocan al principio de los datos, si NA, se eliminan de los da-tos.

method , El método que se empleara. Los emparejamientos parciales están permitidos.

Ejemplos:

```
c1 <- c("A","B","C","D")                # Vector carácter

sort(c1,decreasing = TRUE)
[1] "D" "C" "B" "A"
order(c1)
[1] 1 2 3 4

c2 <- c("A","D","C","B")
sort(c2,decreasing = TRUE)
[1] "D" "C" "B" "A"
order(c2)
[1] 1 4 3 2
```

La función order() da como resultado los índices de ordenación. Mientras que la función sort() da como resultado el objeto ordenado.

```
x <- c(2,1:9)              # Vector numérico
x
[1] 2 1 2 3 4 5 6 7 8 9
(indice <- order(x))
[1]  2  1  3  4  5  6  7  8  9 10
x[indice]
[1] 1 2 2 3 4 5 6 7 8 9
x_sort <- sort(x)
x_sort
[1] 1 2 2 3 4 5 6 7 8 9
```

```
# Ordenar un data.frame
        df1 <- data.frame(
                pais = c("M","G","J","L","Z","B"),
                pob = c(234, 456, 267,987,123,369),
                area = c(1,2,3,2,3,1))

        df2 <- df1[order(df1$pais),]
        df2
          pais pob area
        6    B 369    1
        2    G 456    2
        3    J 267    3
        4    L 987    2
        1    M 234    1
        5    Z 123    3

        df3 <- df1[order(df1$pob),]
        df3
          pais pob area
        5    Z 123    3
        1    M 234    1
        3    J 267    3
        6    B 369    1
        2    G 456    2
        4    L 987    2

        df4 <- df1[order(df1$area),]
        df4
          pais pob area
        1    M 234    1
        6    B 369    1
```

```
2   G 456   2
4   L 987   2
3   J 267   3
5   Z 123   3
```

Si quisieramos ordenar un data frame por varias columnas en R: Por ejemplo, con los datos siguientes se quiere ordenar en primer lugar por la columna z (descendente) y en segundo lugar por la columna b (ascendente):

```
dd <- data.frame(b = factor(c("Hi", "Med", "Hi", "Low","Low","Low"),levels =
      c("Low", "Med", "Hi"), ordered = TRUE),
         x = c("A", "D", "A", "C","H","A"),
         y = c(8, 3, 9, 9,12,5),
         z = c(2, 1, 1, 2, 2, 1))
dd
   b x  y z
1  Hi A  8 2
2 Med D  3 1
3  Hi A  9 1
4 Low C  9 2
5 Low H 12 2
6 Low A  5 1

dd[with(dd, order(b,-z)), ]
   b x  y z
4 Low C  9 2
5 Low H 12 2
6 Low A  5 1
2 Med D  3 1
1  Hi A  8 2
3  Hi A  9 1
```

Combinar data frames

Dentro de los trabajos con data frame una de las tareas más comunes es la fusión de dos data frame. Las formas de fusionarlos son:

- Agregar filas: Dados los data frame, agregar filas consiste en "pegar" las filas de un conjunto de datos debajo del otro. Es lo que se conoce como unión vertical y se realiza con la función rbind(X,Y).
- Agregar columnas: Dados los data frame, agregar columnas consiste en "pegar" las columnas de un conjunto de datos al lado del otro. Es lo que se conoce como unión horizontal y se realiza con la función cbind(X,Y).
- Join: Dados los data frame, al primer data frame se le agrega el contenido del segundo. Generalmente, los data frame presentan una o varias variables que funcionan como campos claves o "keys". Suponiendo que cada data frame contiene al menos una key común a ambos.

- **La función merge(): Combinar dos data frame**

La función merge() combina dos data frame columna a columna o por los nombres de filas.
Sintaxis:

```
merge(x, y, by = intersect(names(x), names(y)),
by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
 sort = TRUE, suffixes = c(".x",".y"), incomparables = NULL, ...)
```

donde:

x, y, data frames, u objetos que se van a combinar en un objeto
by, by.x, by.y , especificación de las columnas comunes.
all , logica; all = L es una abreviatura de all.x = L y all.y = L.
all.x , logica; si TRUE, se añade una fila extra a la salida, una fila por cada fila en x que no tiene correspondencia con una fila en y. Estas filas tendrán NAs en aquellas columnas que sean rellenado con campos de y. El valor por defecto es FALSE, en este caso únicamente las filas con datos en los dos data frame se incluirán en la salida.
all.y , logica; analoga a all.x -
sort , logica. El resultado se ordenará en función de las columnas especificadas en by .
suffixes . caracter(2) especifica los sufijos que se utilizaran para crear nombres cuando los nombres especificados en by names() no sean únicos.

incomparables , Valores que no pueden ser emparejados. Ver match.

Ejemplo: Data frames con una columna en común y con el mismo nombre.

```
x <- data.frame(id=c(1,2,3,4,5),
   nombre=c("José","Pilar","Nuria","Pepa","Antonio"),
   edad=c(30,40,22,53,27),
   peso=c(75,60,55,78,80))
x
  id  nombre edad peso
1  1    José   30   75
2  2   Pilar   40   60
3  3   Nuria   22   55
4  4    Pepa   53   78
5  5 Antonio   27   80

y <- data.frame(id=c(1,2,3,4,5),
            carnet=c("si","si","si","no","no"),
            estad.civil=c("soltero","casado","soltero","divorciado","soltero"),
            hijos=c("no","si","no","si","no"),
            n.hijos=c(NA,2,NA,1,NA))

y
  id carnet estad.civil hijos n.hijos
1  1     si     soltero    no      NA
2  2     si      casado    si       2
3  3     si     soltero    no      NA
4  4     no  divorciado    si       1
5  5     no     soltero    no      NA
```

Los dos data frame tienen una columna en común, en el sentido que existe una columna que tienen los mismos elementos en los dos objetos y además tienen el mismo nombre. Para unirlas utilizamos el siguiente código:

```
merge(x,y,by="id")
  id  nombre edad peso carnet estad.civil hijos n.hijos
1  1    José   30   75     si     soltero    no      NA
2  2   Pilar   40   60     si      casado    si       2
```

```
3  3   Nuria   22   55     si     soltero    no      NA
4  4    Pepa   53   78     no  divorciado    si       1
5  5 Antonio   27   80     no     soltero    no      NA
```

Ejemplos: Dos data frame con variable clave común y con nombre distinto

```
authors <- data.frame(
   surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
   nationality = c("US", "Australia", "US", "UK", "Australia"),
   deceased = c("yes", rep("no", 4)))

authors
   surname nationality deceased
1    Tukey          US      yes
2 Venables   Australia       no
3  Tierney          US       no
4   Ripley          UK       no
5   McNeil   Australia       no

books <- data.frame(
   name = I(c("Tukey", "Venables", "Tierney",
         "Ripley", "Ripley", "McNeil", "R Core")),
   title = c("Exploratory Data Analysis",
          "Modern Applied Statistics ...",
          "LISP-STAT",
          "Spatial Statistics", "Stochastic Simulation",
          "Interactive Data Analysis",
          "An Introduction to R"),
          other.author = c(NA, "Ripley", NA, NA, NA, NA,
               "Venables & Smith"))

books
      name                    title     other.author
1    Tukey     Exploratory Data Analysis          <NA>
2 Venables Modern Applied Statistics ...        Ripley
3  Tierney                LISP-STAT           <NA>
4   Ripley          Spatial Statistics          <NA>
5   Ripley       Stochastic Simulation          <NA>
6   McNeil     Interactive Data Analysis          <NA>
7   R Core        An Introduction to R Venables & Smith
```

Lo data frame ‘authors’ y ‘books’ tiene en común la información del apellido del autor de la obra.

Sin embargo, en ‘authors’ esta variable se llama ‘surname’ y en ‘book’ ‘name’.

Para unir estos dos data frame utilizaremos como campo clave estas variables.

- by.x = ??? se especifica el nombre de la variable en el primer data frame (authors)
- by.y = ??? se especifica el nombre de la variable en el segundo data frame (books)

```
(m1 <- merge(authors, books, by.x = "surname", by.y = "name"))
```

```
   surname nationality deceased                    title other.author
1   McNeil   Australia       no   Interactive Data Analysis        <NA>
2   Ripley          UK       no      Spatial Statistics        <NA>
3   Ripley          UK       no   Stochastic Simulation        <NA>
4  Tierney          US       no               LISP-STAT        <NA>
5    Tukey          US      yes Exploratory Data Analysis        <NA>
6 Venables   Australia       no Modern Applied Statistics ...       Ripley
```

En este caso, primero aparecen las columnas (variables) del data frame 'authors' seguidos de las columnas del data frame 'books'. El campo común, declarado en by.x= y en by.y= es el que indexa el nuevo data frame y mantiene el nombre del primer data frame. En este caso el nombre que aparece en el data frame 'authors' que es name.

Ejemplo: Dos data frame con variable clave común y variables repetidas
Volviendo al ejemplo anterior supongamos que en el data frame y aparece la columna peso pero con valores distintos.

```
x <- data.frame(id=c(1,2,3,4,5),
   nombre=c("José","Pilar","Nuria","Pepa","Antonio"),
   edad=c(30,40,22,53,27),
   peso=c(75,60,55,78,80))

x
  id  nombre edad peso
1  1    José   30   75
2  2   Pilar   40   60
3  3   Nuria   22   55
4  4    Pepa   53   78
5  5 Antonio   27   80

y <- data.frame(id=c(1,2,3,4,5),
            carnet=c("si","si","si","no","no"),
            estad.civil=c("soltero","casado","soltero","divorciado","soltero"),
            hijos=c("no","si","no","si","no"),
```

```
                n.hijos=c(NA,2,NA,1,NA),
                peso=c(25,6,5,7,8))

y
  id carnet estad.civil hijos n.hijos peso
1  1     si     soltero    no      NA   25
2  2     si      casado    si       2    6
3  3     si     soltero    no      NA    5
4  4     no  divorciado    si       1    7
5  5     no     soltero    no      NA    8
```

Al aparece una misma variable en los dos data.frame, al unirles crea dos columnas una peso.x y la otra peso.y indicando de esta forma de donde procede cada una de ellas.

```
merge(x,y, by="id")

  id  nombre edad peso.x carnet estad.civil hijos n.hijos peso.y
1  1    José   30     75     si     soltero    no      NA     25
2  2   Pilar   40     60     si      casado    si       2      6
3  3   Nuria   22     55     si     soltero    no      NA      5
4  4    Pepa   53     78     no  divorciado    si       1      7
5  5 Antonio   27     80     no     soltero    no      NA      8
```

Ejemplo: Dos data frame sin una variable clave común como campo clave

Supongamos ahora que eliminamos del data frame la segunda fila. Es decir, los dos data frame tienen una columna que se llama id pero que no coincide en el contenido.

```
y <- y[-2,]
y
  id carnet estad.civil hijos n.hijos peso
1  1     si     soltero    no      NA   25
3  3     si     soltero    no      NA    5
4  4     no  divorciado    si       1    7
5  5     no     soltero    no      NA    8

merge(x,y,by="id")

  id  nombre edad peso.x carnet estad.civil hijos n.hijos peso.y
```

```
1  1    José   30    75    si    soltero    no     NA     25
2  3   Nuria   22    55    si    soltero    no     NA      5
3  4    Pepa   53    78    no  divorciado   si      1      7
4  5 Antonio   27    80    no    soltero    no     NA      8
```

En la tabla de datos final no aparece el individuo 2 (de nombre “Pilar”). Por defecto, la función merge() solo utiliza las filas que están en los dos data frame. Este comportamiento se puede dirigir por medio de los argumentos: all, all.x y all.y

- all=TRUE se utilizan todas las filas tanto del data frame x como del y.
- all.x=TRUE se utilizan todas las filas del data frame x.
- all.y=TRUE se utilizan todas las filas del data frame y.

```
merge(x,y, all=TRUE)
```

```
  id peso  nombre edad carnet estad.civil hijos n.hijos
1  1   25   <NA>   NA     si     soltero    no      NA
2  1   75   José   30   <NA>       <NA>  <NA>      NA
3  2   60  Pilar   40   <NA>       <NA>  <NA>      NA
4  3    5   <NA>   NA     si     soltero    no      NA
5  3   55  Nuria   22   <NA>       <NA>  <NA>      NA
6  4    7   <NA>   NA     no   divorciado   si       1
7  4   78   Pepa   53   <NA>       <NA>  <NA>      NA
8  5    8   <NA>   NA     no     soltero    no      NA
9  5   80 Antonio   27   <NA>       <NA>  <NA>      NA
```

Combinar más de dos data frame. La función Reduce()

Una opción para unir más de dos data frame sería usar reiteradamente la función merge(), sin embargo, la función reduce() automatiza este proceso:

Ejemplo:

```
A <- data.frame(id = c("A", "B", "C", "D"), edad = c(24, 25, 17, 19), altura = c(180,190,
175, 165))
B <- data.frame(gender = c("H", "H", "M", "M"), id = c("A", "B", "C", "D"))
C <- data.frame(id = c("A", "B", "C", "D"),
                mates = c(6.5, 8.9, 7.4, 9.2),
                ciencias = c(7.2,8.4, 6.5, 8.7))
```

```
Reduce(merge,list(A,B,C))
```

La función Reduce() permite aplicar una función binaria, en este caso merge(), sucesivamente a los elementos de una lista.

Eliminar filas duplicadas en un data frame

Ejemplo:

```
a <- c(rep("A", 3), rep("B", 3), rep("C",2))
b <- c(1,1,2,4,1,1,2,2)
df <-data.frame(a,b)

df
  a b
1 A 1
2 A 1
3 A 2
4 B 4
5 B 1
6 B 1
7 C 2
8 C 2

duplicated(df)
 [1] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE
df[duplicated(df), ]
  a b
2 A 1
6 B 1
8 C 2

df[!duplicated(df), ]
  a b
1 A 1
3 A 2
4 B 4
5 B 1
7 C 2
```

Una manera más directa mediante la función unique() sería:

```
unique(df)
  a b
1 A 1
3 A 2
4 B 4
5 B 1
7 C 2
```

Funciones sobre data.frames

- **La función sapply()**

La función sapply() funciona de manera similar a la función apply() ya vista en matrices, permite aplicar una función a más de una columna de un data.frame

Su sintaxis es:

```
sapply(X, FUN)
```

donde:

X , será un objeto o data.frame

FUN , función que se desea aplicar a cada elemento de x

Ejemplo:

Si a partir del data.frame de R cars (que contiene datos de velocidad y la distancia de frenado de 50 coches) queremos calcular el valor mínimo de cada una de esas variables, lo podemos hacer mediante la función sapply

```
dt <- cars
smn_cars <- sapply(dt, min)
smn_cars
speed  dist
   4     2
```

- **La función tapply()**

La función tapply aplica una función a un vector en las categorías que define un factor, es decir, aplica una misma función a un grupo de datos organizados en los distintos niveles de un Factor.

Su sintaxis es:

tapply(X, INDEX, FUN)

donde:

X , será un objeto o data.frame donde queramos aplicar la función

INDEX , será una lista de uno o más factores

FUN , función que se desea aplicar

Ejemplo:

En este caso queremos calcular la media de la longitud de los pétalos de las tres especies de flores que tenemos en el dataset Iris de R. Por lo tanto, la variable de longitud de pétalo será nuestro argumento X y la variable de especies será el factor de nuestro argumento INDEX:

```
tapply(iris$Petal.Length, iris$Species, mean)
    setosa versicolor  virginica
     1.462      4.260      5.552
```

- **La función by()**

La función by() permite aplicar una misma función a distintos subconjuntos de datos de un data.frame.

La sintaxis de la función by() es:

by(data, INDICES, FUN, …, simplify = TRUE)

donde:

data objeto de R, data frame o matrix

INDICES factor o lista de factores de igual longitud que data

FUN función que se aplica a cada uno de los subconjuntos de data

Ejemplo:

A partir del data.frame de R iris, calcula la media de la longitud del sépalo en función de la variable Species

```
by(data = iris[, 1], INDICES = iris$Species, summary)
iris$Species: setosa
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   4.800   5.000   5.006   5.200   5.800
```

```
------------------------------------------------------------
iris$Species: versicolor
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.900   5.600   5.900   5.936   6.300   7.000
------------------------------------------------------------
iris$Species: virginica
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.900   6.225   6.500   6.588   6.900   7.900
```

¿Practicamos un poco?

1.- Trabajaremos con la tabla de datos “iris”.

a. Calcula la media de la longitud del pétalo y del sépalo.
b. Calcula la mediana de la anchura de los pétalos.
c. Calcula la varianza de la anchura de los sépalos y la desviación típica de la longitud de los pétalos.
d. Muestra en pantalla la primera y la tercera columna de los 5 primeros individuos.
e. Muestra en pantalla los 10 primeros individuos pero sin la variable especies.
f. Muestra un caso de cada 4 a partir del primero.
g. Muestra los individuos que tienen una longitud e pétalo superior a 4.9
h. Obten todos los individuos de la especie setosa con una longitud de sépalo superio a 4.5 y con una anchura se pétalo menor de 0.3.

2.- Sigue las instrucciones:

a. Crea los vectores tallaM, tallaM, pesoH y pesoH conteniendo los datos de n2 = 257 mujeres y n2 = 312 hombres. Quédate sólo con la parte entera de los valores que obtengas.

   Nota: Para conseguir los datos usa la función rnorm(), simulando las siguientes distribuciones:

   - La altura de las mujeres se distribuye según una normal de media 165 cm y desviación típica de 6 cm.
   - La altura de los hombres se distribuye según una normal de media 175 cm y desviación típica de 7 cm.
   - EL peso de las mujeres se distribuye según una normal de media 60 kg y desviación típica de 2 kg.
   - EL peso de los hombres se distribuye según una normal de media 75 kg y desviación típica de 4 kg.

b. Crea un data.frame “datos” con las variables talla, peso y sexo (sexo debe ser un factor). Nombra a las columnas de forma apropiada.
c. A partir del data frame “datos” calcula la altura menor de hombres y mujeres. Realiza el mismo cálculo para la variable “peso”.
d. ¿Hay individuos de más de 1,70 m y menos de 75 kilos? ¿Cuántos? ¿Cuántos son hombres y cuántos mujeres?

3. En primer lugar cargue el data.frame disponible en R “mtcars”, se pide:

a. Selecciona las 10 primeras filas de la variable mpg.
b. Seleccione las filas 4 y 8 de la variable cyl.

c. Seleccione el elemento 3 de la columna 4.

d. Seleccione las columnas 1 y 4 de dos formas distintas. ¿Los objetos son los mismos?

e. Seleccione los casos que verifican mpg > 20 y wt > 4.

f. Seleccione los casos con valores en todas las variables (donde tenga sentido) por encima del correspondiente tercer cuartil (Q3). (?quantile)

Solucionario

1

a

```
mean(iris$Petal.Length)
mean(iris$Sepal.Length)
```

b

```
median(iris$Petal.Width)
```

c

```
var(iris$Sepal.Width)
sd(iris$Petal.Length)
```

d

```
iris[1:5, c(1, 3)]
```

e

```
iris[1:10, -5]
```

f

```
iris[seq(1, nrow(iris), by = 4), ]
```

g

```
iris[iris$Petal.Length > 4.9, ]
```

h

```
iris[
  iris$Species == "setosa" &
    iris$Sepal.Length > 4.5 &
    iris$Petal.Width < 0.3,]
```

2

a

```
set.seed(123)
nM <- 257
nH <- 312
tallaM <- trunc(rnorm(nM, mean = 165, sd = 6))
tallaH <- trunc(rnorm(nH, mean = 175, sd = 7))
pesoM <- trunc(rnorm(nM, mean = 60, sd = 2))
pesoH <- trunc(rnorm(nH, mean = 75, sd = 4))
```

b

```
datos <- data.frame(
  talla = c(tallaM, tallaH),
  peso = c(pesoM, pesoH),
  sexo = factor(c(rep("Mujer", nM), rep("Hombre", nH))))
```

c

```
aggregate(talla ~ sexo, data = datos, FUN = min)
```

```
aggregate(peso ~ sexo, data = datos, FUN = min)
d
seleccionados <- datos[
  datos$talla > 170 & datos$peso < 75,]
any(datos$talla > 170 & datos$peso < 75)
nrow(seleccionados)
table(seleccionados$sexo)

3
a
mtcars[1:10, "mpg"]
b
mtcars[c(4, 8), "cyl"]
c
mtcars[3, 4]
d
objeto1 <- mtcars[, c(1, 4)]
objeto2 <- mtcars[, c("mpg", "hp")]
identical(objeto1, objeto2)
e
mtcars[
  mtcars$mpg > 20 & mtcars$wt > 4,]
f
Q3 <- sapply(
  mtcars,
  quantile,
  probs = 0.75,
  na.rm = TRUE)

condiciones_Q3 <- sweep(
  mtcars,
  MARGIN = 2,
  STATS = Q3,
  FUN = ">")

mtcars[rowSums(condiciones_Q3) == ncol(mtcars), ]
```