

Data frames

Sitio: [Plataforma de Formación On line del Instituto Andaluz de Administración Pública](#)
Curso: (I22F-PT05) Entorno de Programación R
Libro: Data frames

Imprimido por: ALFONSO LUIS MONTEJO RAEZ
Día: lunes, 4 de abril de 2022, 10:27

Tabla de contenidos

1. Que es un data frame

2. Creando data frame

3. Trabajando con data frames

3.1. Visualización

3.2. Otras opciones de visualización

3.3. Acceso

3.4. Filtrado

4. Modificando un data frame

4.1. Añadir variables calculadas

4.2. Añadir nueva variable

4.3. Eliminar variables

4.4. Modificar contenido

4.5. Modificar nombres

5. Operaciones con data frames

5.1. División

5.2. aggregate()

5.3. sapply()

6. Unión de data frames

6.1. Inner join

6.2. Left join

6.3. Right join

6.4. Full join

6.5. Otros parámetros de merge()

1. Que es un data frame

Un **data frame** es una tabla de doble entrada formada por columnas y filas.

Sus principales características son:

- Cada columna es una variable.
- Cada fila es una observación de las variables, normalmente un mismo caso o individuo.
- Cada columna puede ser de un tipo distinto (carácter, numérico, lógico, etc.)
- Este formato se corresponde con los ficheros de datos estadísticos con los que trabajamos normalmente.

2. Creando data frame

Los data frames se crean usando la orden **data.frame()** y concatenando vectores (separados por comas), donde cada uno de los vectores se corresponde con una variable. Hay que tener en cuenta que todos los vectores tienen que tener el mismo número de elementos.

data.frame(vectores)

Ejemplo

Vamos a crear un data frame con los datos de Población, N° de municipios y Extensión en km² para cada una de las provincias de Andalucía.

```
> Provincia<-c('AL','CA','CO','GR','HU','JA','MA','SE')
> Poblacion<-c(706672,1239435,788219,912938,518930,643484,1630615,1939527)
> N_Municipios<-c(103,44,75,172,79,97,103,105)
> Extension<-c(8775.1,7436.4,13771.6,12647.7,10128.5,13489.4,7309.0,14036.5)
```

En primer lugar tenemos que crear los vectores con cada una de las variables (Provincia, Población, N_Municipios y Extensión).

```
> Datos<-data.frame(Provincia,Poblacion,N_Municipios,Extension)
```

Por último, usamos la orden **data.frame** con los vectores separados por comas. Como siempre tenemos que asignar la orden a un objeto, en este caso, llamado Datos para guardar los datos creados.

Si observamos los datos creados vemos la tabla con la información que hemos almacenado.

```
> Datos
  Provincia Población N_Municipios Extension
1      AL      706672           103    8775.1
2      CA     1239435            44    7436.4
3      CO      788219            75   13771.6
4      GR      912938           172   12647.7
5      HU      518930            79   10128.5
6      JA      643484            97   13489.4
7      MA     1630615           103    7309.0
8      SE     1939527           105   14036.5
```

3. Trabajando con data frames

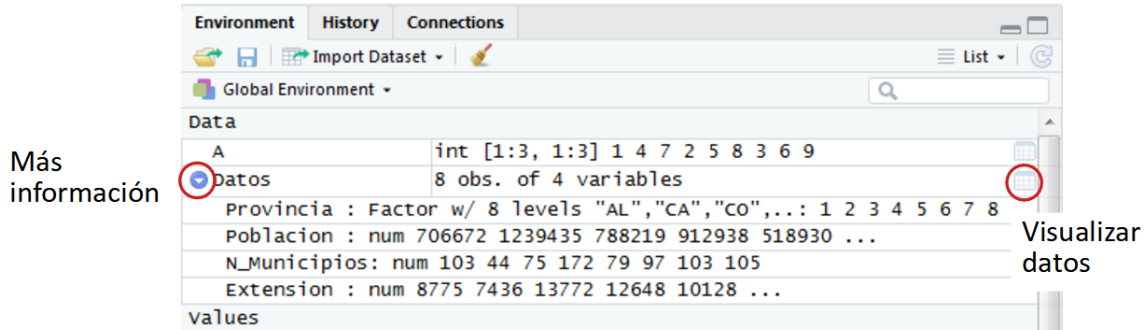
Ya sabemos como crear data frames ahora vamos a ver como podemos trabajar con ellos. Lo principal que vamos a ver va a ser:

- Visualizar un data frame.
- Acceder al contenido de un data frame.
- Filtrar el contenido de un data frame.

3.1. Visualización

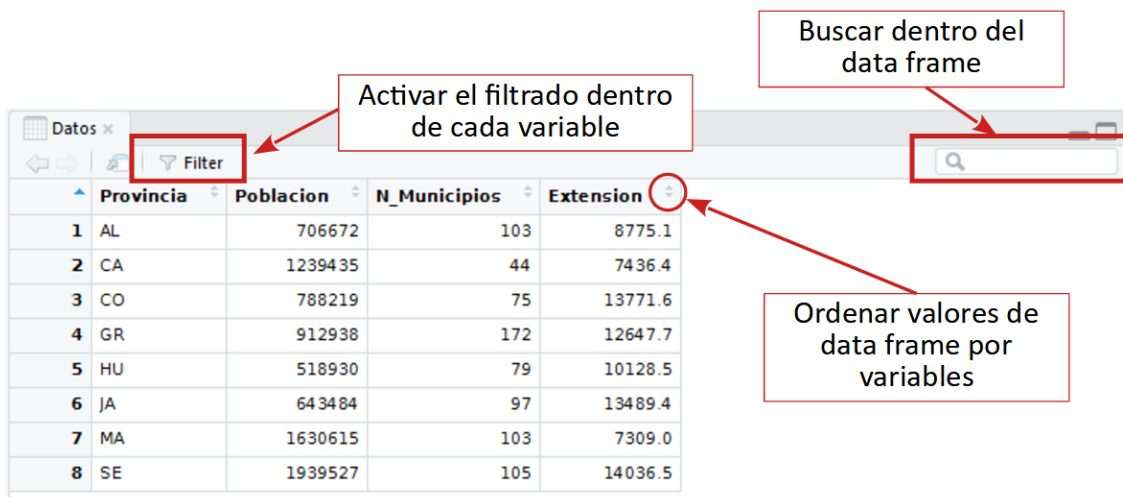
Una de las ventajas de trabajar con RStudio es la posibilidad de visualizar los objetos creados de una manera interactiva.

Accediendo a la pestaña **Environment**, donde aparecen todos los objetos que hemos creado en la sesión de trabajo, podemos observar el data frame que hemos creado.



Pulsando el botón azul (a la izquierda) con el triángulo blanco se pueden observar la **información de las variables** que componen el data frame.

Pulsando el icono de la tabla (a la derecha) se pueden **visualizar los datos** en una pantalla independiente donde podemos trabajar con la tabla de una manera más cómoda. Esta acción es equivalente a la orden **View()**.



Las acciones que podemos hacer sobre el data frame son

- Buscar dentro del contenido del data frame.
- Ordenar el data frame a partir de una variable.
- Filtrar el contenido.

Todas estas opciones solamente afectan a la visualización de los datos; en ningún momento suponen una modificación de los datos contenidos en el data frame.

Para activar el filtrado de datos hay que pulsar el botón **Filter**. En función del tipo de variable, la forma de filtrar es distinta: **Rango** para variables numéricas y **Texto** para variables de carácter o factores.

Datos x				
Filter				
	Provincia	Poblacion	N_Municipios	Extension
	All	[...]	All	All
1	AL	 500000 - 2000000	103	8775.1
2	CA		100	7436.4
3	CO		75	13771.6
4	GR		172	12647.7
5	HU		79	10128.5
6	JA		97	13489.4
7	MA		103	7309.0
8	SE		105	14036.5

Datos x	
Filter	
	Provincia
	[...]
1	AL
2	CA
3	CO
4	GR
5	HU
6	JA
7	MA
8	SE

3.2. Otras opciones de visualización

También se puede visualizar el contenido del data frame usando órdenes como **head()** y **tail()**.

- **head()**. Muestra los datos de las primeras filas del data frame.
- **tail()**. Muestra los datos de las últimas filas del data frame.

Los parámetros de las dos funciones son los mismos: nombre del data frame y el número de filas que se quieren mostrar.

Ejemplo

Para visualizar las tres primeras o las tres últimas filas del data frame Datos que hemos creado al principio.

```
> head(Datos,3)
  Provincia Poblacion N_Municipios Extension
1        AL    706672          103    8775.1
2        CA    1239435           44    7436.4
3        CO     788219           75   13771.6
```

```
> tail(Datos,3)
  Provincia Poblacion N_Municipios Extension
6        JA     643484           97   13489.4
7        MA    1630615          103    7309.0
8        SE    1939527          105   14036.5
```


3.3. Acceso

Para acceder al contenido de un data frame se puede hacer de dos formas:

- Haciendo referencia la **número de fila y/o columna [i,j]** igual que hemos hecho antes con las matrices.
- Nombrando las variables del data frame con \$.

nombre_data_frame\$nombre_variable

Cuando hacemos referencia a las variables usando \$ siempre debemos citar el nombre del data frame que queremos visualizar. Para evitar tener que estar constantemente nombrando el data frame se puede usar la función **attach()** y **detach()**.

Ejemplo 1

Vamos a ver algunas formas de acceso al data frame Datos.

```
> Datos$Poblacion
[1] 706672 1239435 788219 912938 518930 643484 1630615 1939527
> Datos[,c(1,3)]
  Provincia N_Municipios
1        AL           103
2        CA            44
3        CO            75
4        GR           172
5        HU            79
6        JA            97
7        MA           103
8        SE           105
> Datos[1:4,]
  Provincia Poblacion N_Municipios Extension
1        AL    706672           103    8775.1
2        CA   1239435            44    7436.4
3        CO    788219            75   13771.6
4        GR    912938           172   12647.7
```

Con la primera línea de código se accede a la variable denominada Población usando \$.

Con la segunda línea de código se accede a las columnas primera y tercera que hacen referencia a las variables Provincia y N_Municipios.

Con la tercera línea de código se accede a las filas de la primera a la cuarta que son los datos de las provincias de Almería, Cádiz, Córdoba y Granada.

Ejemplo 2

Vamos a ver un ejemplo del uso de la función **attach()** para acceder al contenido del data frame llamado Datos que tiene cuatro variables (Provincia, Población N_Municipios y Extensión).

```
> Poblacion
Error: object 'Poblacion' not found
> Datos$Poblacion
[1] 706672 1239435 788219 912938 518930 643484 1630615 1939527
> attach(Datos)
The following objects are masked _by_ .GlobalEnv:
    Extension, N_Municipios, Provincia
> Poblacion
[1] 706672 1239435 788219 912938 518930 643484 1630615 1939527
```

Si tratamos de acceder a la variable Población del data frame Datos usando simplemente su nombre R nos devuelve un error porque intenta buscar un objeto que se llame Población. Para acceder al contenido tenemos que usar **Datos\$Población**, es decir, tenemos que decir siempre en que data frame tiene que buscar la variable.

Para evitar esto (tener que estar llamando constantemente al data frame) usamos la orden **attach** con lo que fijamos el data frame con el que vamos a trabajar. De esta forma, si ponemos solamente el nombre de una de las variables, el programa entiende que lo tiene que buscar dentro

del data frame Datos.

Cuando queramos dejar de trabajar con el data frame Datos debemos usar la orden detach() para dejar sin efecto la orden attach que hemos hecho al principio.

```
> detach(Datos)
> Poblacion
Error: object 'Poblacion' not found
```

3.4. Filtrado

Otra de las acciones que podemos hacer con un data frame es filtrar su contenido aplicando condiciones a una o varias de las variables del data frame.

Recordad que los operadores que se pueden usar en R para hacer filtros son los siguientes

Operador	=	≠	<	>	≤	≥	negación	conjunción	disjunción
Signo	==	!=	<	>	<=	>=	!	&	

Ejemplo 1

Vamos a ver como filtramos el data frame Datos aplicando condiciones.

```
> Datos[Datos$Poblacion>1000000,]  
  Provincia Poblacion N_Municipios Extension  
2        CA  1239435           44    7436.4  
7        MA  1630615           103    7309.0  
8        SE  1939527           105   14036.5
```

Aquí nos quedamos con todas las provincias que tienen más de un millón de habitantes.

```
> Datos[Datos$Poblacion<1000000&N_Municipios>100,]  
  Provincia Poblacion N_Municipios Extension  
1        AL   706672           103    8775.1  
4        GR   912938           172   12647.7
```

Aquí aplicamos un filtro sobre dos variables para quedarnos con las provincias de menos de un millón de habitantes pero que tienen más de 100 municipios uniendo las dos condiciones con el operador **&**.

4. Modificando un data frame

Podemos modificar el contenido de un data frame de varias maneras:

- Añadiendo una nueva variable, calculada a partir de variables existentes en el propio data frame.
- Añadiendo nuevas variables externas.
- Eliminando una variable.
- Modificando el contenido.

4.1. Añadir variables calculadas

La primera forma de modificar un data frame es añadir una nueva variable calculada a partir de otras variables que ya están en el propio data frame. Para realizar esto simplemente tenemos que nombrar la nueva variable dentro del data frame y asignarle el contenido.

nombre_data_frame\$nueva_variable<-operación que vayamos a realizar

Ejemplo

En el data frame Datos con el que estamos trabajando en este capítulo, vamos a crear una nueva variable llamada Densidad como Población entre extensión.

```
> Datos$Densidad<-Datos$Poblacion/Datos$Extension
> Datos
```

	Provincia	Poblacion	N_Municipios	Extension	Densidad
1	AL	706672	103	8775.1	80.53150
2	CA	1239435	44	7436.4	166.67137
3	CO	788219	75	13771.6	57.23511
4	GR	912938	172	12647.7	72.18214
5	HU	518930	79	10128.5	51.23463
6	JA	643484	97	13489.4	47.70294
7	MA	1630615	103	7309.0	223.09687
8	SE	1939527	105	14036.5	138.17739

Lo que hacemos es crear en el data frame una nueva variable llamada Densidad a la que le asignamos la división de la variable Población entre Extensión. Cuando visualizamos de nuevo el contenido nos aparece una variable más llamada Densidad con la densidad de población de cada provincia andaluza.

4.2. Añadir nueva variable

Otra acción frecuente con un data frame es añadirle una nueva variable usando la función **cbind()**.

Recordad que esta función concatenaba vectores por columnas y lo que queremos hacer nosotros es añadir una columna que sería una nueva variable. También tenemos que tener en cuenta que la variable que vayamos a añadir tiene que tener el mismo número de elementos (filas) que el data frame.

Ejemplo

Vamos a añadir al data frame Datos una nueva variable de tipo lógico (con valores True (verdadero) o False (falso)) en función de si la provincia tiene salida al mar o no la tiene.

```
> Mar<-c(T,T,F,T,T,F,T,F)
> cbind(Datos,Mar)->Datos
> Datos
```

	Provincia	Poblacion	N_Municipios	Extension	Densidad	Mar
1	AL	706672	103	8775.1	80.53150	TRUE
2	CA	1239435	44	7436.4	166.67137	TRUE
3	CO	788219	75	13771.6	57.23511	FALSE
4	GR	912938	172	12647.7	72.18214	TRUE
5	HU	518930	79	10128.5	51.23463	TRUE
6	JA	643484	97	13489.4	47.70294	FALSE
7	MA	1630615	103	7309.0	223.09687	TRUE
8	SE	1939527	105	14036.5	138.17739	FALSE

Lo que hacemos es crear un nuevo vector que llamamos Mar con los ocho valores necesarios y luego lo añadimos al data frame Datos usando la orden **cbind()**.

Tenemos que asignar el resultado de esa orden a otro a un objeto para que se almacene la modificación que le hemos hecho. En este caso lo volvemos a asignar al objeto Datos para que machaque el contenido anterior.

El resultado es el mismo data frame anterior pero con una nueva variable llamada Mar.

4.3. Eliminar variables

En el caso de querer eliminar una o varias variables de un data frame simplemente tenemos que hacer referencia a la columna/as que queremos eliminar usando un valor negativo indicando su posición con los corchetes.

nombre_data_frame[-columna a eliminar]

Ejemplo

Del data frame Datos vamos a eliminar la variable Mar que acabamos de añadir en el paso anterior.

```
> Datos[,-6]
  Provincia Población N_Municipios Extension  Densidad
1        AL    706672         103    8775.1  80.53150
2        CA   1239435          44    7436.4 166.67137
3        CO    788219          75   13771.6  57.23511
4        GR    912938         172   12647.7  72.18214
5        HU    518930          79   10128.5  51.23463
6        JA    643484          97   13489.4  47.70294
7        MA   1630615         103    7309.0 223.09687
8        SE   1939527         105   14036.5 138.17739
```

En este caso, la variable Mar ocupaba la sexta fila por lo que usamos el negativo con el número de la columna que queremos eliminar. En este ejemplo no hemos asignado el resultado de la operación por lo que no eliminamos físicamente la columna del data frame.

Si quisiéramos eliminar más de una columna al mismo tiempo simplemente tenemos que concatenar las posiciones de las columnas que queremos eliminar.

```
> Datos[,-c(2,4)]
  Provincia N_Municipios  Densidad  Mar
1        AL          103  80.53150 TRUE
2        CA           44 166.67137 TRUE
3        CO           75  57.23511 FALSE
4        GR          172  72.18214 TRUE
5        HU           79  51.23463 TRUE
6        JA           97  47.70294 FALSE
7        MA          103 223.09687 TRUE
8        SE          105 138.17739 FALSE
```

En este caso queremos eliminar las columnas Población y Extension que ocupan la segunda y cuarta posición dentro del data frame Datos.

4.4. Modificar contenido

También podemos modificar el contenido de una de las variables o de un dato concreto dentro del data frame. Esta operación la podemos hacer de dos formas.

- Indicando la posición (fila y columna) del valor que queremos modificar
- Usando las funciones **edit()** o **fix()**.

Vamos a ver unos ejemplos para ilustrar cada uno de los métodos.

Ejemplo 1

Vamos a modificar un valor dentro del data frame Datos indicando su posición. En concreto, vamos asignarle a la provincia de Cádiz un número de municipios igual a 100.

Para ello tenemos que localizar el dato que queremos modificar y ver en que posición está. En nuestro caso la provincia de Cádiz es la segunda fila y la columna que tiene el número de municipios es la tercera, por lo que queremos modificar el dato situado en la segunda fila, tercera columna.

> Datos

	Provincia	Población	N_Municipios	Extension	Densidad	Mar
1	AL	706672	103	8775.1	80.53150	TRUE
2	CA	1239435	44	7436.4	166.67137	TRUE
3	CO	788219	75	13771.6	57.23511	FALSE
4	GR	912938	172	12647.7	72.18214	TRUE
5	HU	518930	79	10128.5	51.23463	TRUE
6	JA	643484	97	13489.4	47.70294	FALSE
7	MA	1630615	103	7309.0	223.09687	TRUE
8	SE	1939527	105	14036.5	138.17739	FALSE

> Datos[2,3]<-100

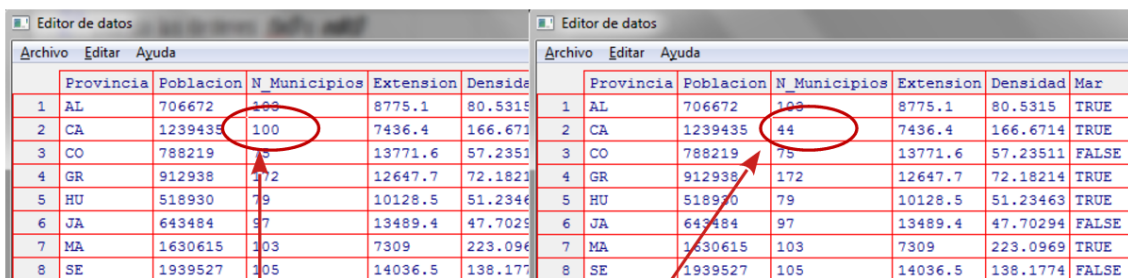
> Datos

	Provincia	Población	N_Municipios	Extension	Densidad	Mar
1	AL	706672	103	8775.1	80.53150	TRUE
2	CA	1239435	100	7436.4	166.67137	TRUE
3	CO	788219	75	13771.6	57.23511	FALSE
4	GR	912938	172	12647.7	72.18214	TRUE
5	HU	518930	79	10128.5	51.23463	TRUE
6	JA	643484	97	13489.4	47.70294	FALSE
7	MA	1630615	103	7309.0	223.09687	TRUE
8	SE	1939527	105	14036.5	138.17739	FALSE

Ejemplo 2

Vamos a volver a modificar el valor del número de municipios de la provincia de Cádiz para devolverle su verdadero valor que es 44.

Para ello podemos usar las órdenes **fix()** o **edit()**. Con ellas podemos modificar de forma interactiva el dato ya que se nos abre una hoja donde se representan los valores para que podamos modificar el que queramos. La diferencia está en que con **fix** el cambio se guarda automáticamente y con **edit** hay que guardarlo asignándolo a un objeto.



	Provincia	Población	N_Municipios	Extension	Densidad	Mar
1	AL	706672	103	8775.1	80.5315	TRUE
2	CA	1239435	44	7436.4	166.6714	TRUE
3	CO	788219	75	13771.6	57.23511	FALSE
4	GR	912938	172	12647.7	72.18214	TRUE
5	HU	518930	79	10128.5	51.23463	TRUE
6	JA	643484	97	13489.4	47.70294	FALSE
7	MA	1630615	103	7309	223.0969	TRUE
8	SE	1939527	105	14036.5	138.1774	FALSE

Se modifica el 100 por 44 y se cierra la ventana

> edit(Datos)->Datos

4.5. Modificar nombres

También podemos consultar o modificar los nombres de las variables con la orden **names()**.

En el caso de cambiar los nombres tenemos que tener cuidado con usar espacios o acentos o usar caracteres raros ya que podemos tener problemas. En el caso de usar este tipo de caracteres tenemos que nombrar la variable con tildes graves (`).

Ejemplo

Si consultamos los nombres de las variables del data frame Datos

```
> names(Datos)
[1] "Provincia"    "Poblacion"    "N_Municipios" "Extension"    "Densidad"
[6] "Mar"          "Zona"
```

Podemos modificar todos los nombres asignando un nuevo vector con los nuevos nombres o modificar uno solo indicando entre corchetes la posición del nombre que queramos cambiar.

```
> names(Datos)[4] <- 'Extension km2'
> names(Datos)
[1] "Provincia"    "Poblacion"    "N_Municipios" "Extension km2" "Densidad"
[6] "Mar"          "Zona"
```

Ojo: Al cambiar el nombre de la variable, le hemos introducido un espacio. Ahora la forma de llamar a esa variable cambia y hay que añadir **`nombre`** (nombre de la variable entre tildes graves).

```
> Datos$Extension Km2
Error: unexpected symbol in "Datos$Extension Km2"
> Datos$`Extension km2`
[1] 8775.1 7436.4 13771.6 12647.7 10128.5 13489.4 7309.0 14036.5
```

5. Operaciones con data frames

Vamos a ver ahora un conjunto de funciones que son útiles para realizar operaciones sobre un data frame. Las funciones son:

- **subset()**. Para hacer subconjuntos del data frame original
- **split()**. Trozar un data frame a partir de una variable.
- **aggregate()**. Resumir información de un data frame.
- **sapply()**. Aplicar funciones a un data frame.

5.1. División

Las funciones **subset()** y **split()** permiten hacer subconjuntos y divisiones de un data frame.

- **subset()**. Para hacer subconjuntos del data frame original.
- **split()**. Trozar un data frame a partir de una variable.

Vamos a ver, con ejemplos, como funcionan estas dos funciones.

Ejemplo 1

La función **subset()** realiza subconjuntos a partir de un data frame estableciendo condiciones a una o a varias variables del data frame.

Con el data frame Datos vamos a filtrar y quedarnos solo que las provincias que tiene salida al mar.

```
> subset(Datos, Mar==T, 1:4)
```

	Provincia	Poblacion	N_Municipios	Extension
1	AL	706672	103	8775.1
2	CA	1239435	44	7436.4
4	GR	912938	172	12647.7
5	HU	518930	79	10128.5
7	MA	1630615	103	7309.0

En la orden tenemos que usar como argumentos el nombre del data frame que queremos filtrar, la condición que queramos establecer y las variables que va a contener el data frame resultante.

Si no indicamos las variables que va a contener el data frame, el programa entiende que las queremos todas.

Si almacenamos el resultado de la orden subset, lo que obtenemos es un nuevo data frame con las características que le hemos indicado en la orden.

Ejemplo 2

La función **split()** trocea un data frame en función de alguna de las variables del propio data frame. Normalmente las variables usadas para trocear el data frame deben ser categóricas.

Siguiendo con el ejemplo anterior vamos a trocear el data frame Datos en función de la variable Mar. Con esto conseguimos dos data frame: uno con las provincias que tienen salida al mar y otro para las que carecen de ella.

```
> split(Datos, Datos$Mar)
```

\$`FALSE`

	Provincia	Poblacion	N_Municipios	Extension	Densidad	Mar
3	CO	788219	75	13771.6	57.23511	FALSE
6	JA	643484	97	13489.4	47.70294	FALSE
8	SE	1939527	105	14036.5	138.17739	FALSE

\$`TRUE`

	Provincia	Poblacion	N_Municipios	Extension	Densidad	Mar
1	AL	706672	103	8775.1	80.53150	TRUE
2	CA	1239435	44	7436.4	166.67137	TRUE
4	GR	912938	172	12647.7	72.18214	TRUE
5	HU	518930	79	10128.5	51.23463	TRUE
7	MA	1630615	103	7309.0	223.09687	TRUE

La orden split usa como parámetros el nombre del data frame que queremos trocear y la variable a partir de la cual se realiza la partición.

Si almacenamos el resultado de la orden split, lo que obtenemos es un objeto tipo lista (ese tipo de dato lo veremos más adelante) donde cada trozo del data frame se guarda dentro de la lista como objetos independientes.

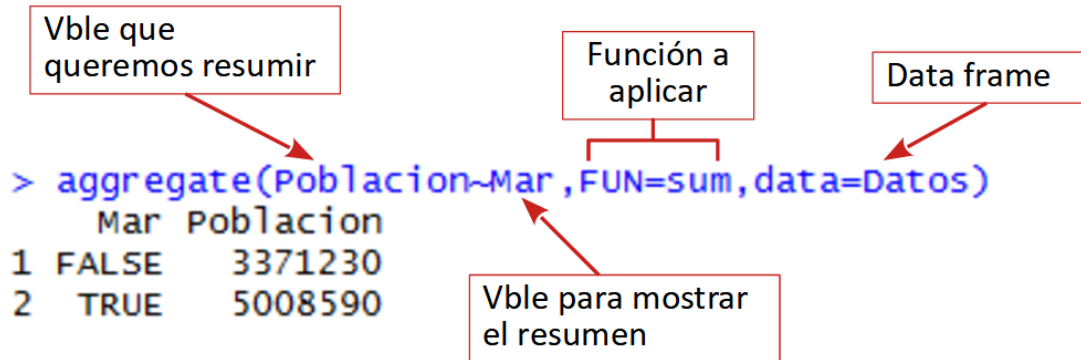
5.2. aggregate()

La función **aggregate()** nos sirve para resumir la información de una o más variables dentro de un data frame.

Al aplicar esta función necesitamos una variable de tipo numérica (de la que queremos obtener la información) y una variable de tipo categórica.

Ejemplo

Con el data frame Datos vamos a calcular la población que reside en provincias con salida al mar.



Para la orden `aggregate()` tenemos que indicar, la función que queremos usar como resumen (en este caso la suma), de que data frame queremos calcular las cosas (Datos) y las variables implicadas.

Para escribir las variables tenemos que poner la variable que queremos sumar seguida de la virgulilla (~) y la variable categórica de la que queremos la información. Para obtener en el teclado la virgulilla tenemos que pulsar las teclas `Alt+126`.

De esta manera todas las variables que se pongan a la izquierda de la virgulilla tienen que ser numéricas (porque es a las que se les va aplicar la función) y las variables que se pongan a la derecha de la virgulilla tienen que ser categóricas.

Así, si queremos usar más de una variable numérica, debemos usar la concatenación de columnas **cbind()**.

```
> aggregate(cbind(Poblacion, N_Municipios)~Mar, FUN=sum, data=Datos)
  Mar Poblacion N_Municipios
1 FALSE   3371230         277
2  TRUE   5008590         501
```

En este caso queremos obtener la información de la Población y el N° de municipios.

Si lo que queremos es usar más de una variable categórica, tenemos que unir las usando un más (+).

Para ilustrar esto vamos a crear una nueva variable categórica llamada Zona en el data frame Datos que va a clasificar las provincias andaluzas en función de su ubicación geográfica (Este, Oeste y Centro).

```
> Datos$Zona<-c('E', 'O', 'C', 'E', 'O', 'E', 'C', 'O')
```

```
> aggregate(Poblacion~(Mar+Zona), FUN=sum, data=Datos)
  Mar Zona Poblacion
1 FALSE   C    788219
2  TRUE   C   1630615
3 FALSE   E    643484
4  TRUE   E   1619610
5 FALSE   O   1939527
6  TRUE   O   1758365
```

En este caso queremos obtener la información de la Población de las provincias en función de la Zona geográfica y su salida al mar.

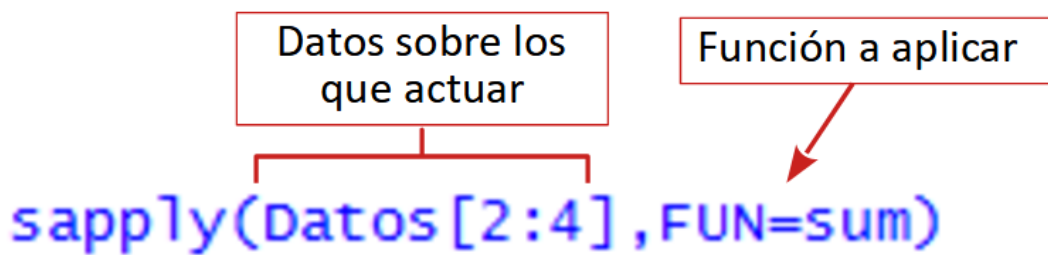
5.3. sapply()

La orden **sapply()** sirve para aplicar una función a las variables de un data frame. En este caso podemos aplicar cualquier función a todas las variables de un data frame.

La única limitación que tenemos que tener en cuenta es el tipo de variable con la que queremos trabajar ya que no podemos aplicar una función numérica a variables que son de texto, es decir, no podemos aplicar una función de suma a una variable categorica.

Ejemplo

Vamos a sumar todas las variables numéricas del data frame Datos para obtener el total de Andalucía para las variables Población, N° de municipios y Extensión.



```
> sapply(Datos[2:4], FUN=sum)
Poblacion N_Municipios Extension
8379820.0      778.0      87594.2
```

6. Unión de data frames

Hay situaciones en las que tenemos dos conjuntos de datos (data frames) que tienen información sobre los mismos individuos y necesitamos unir dicha información.

Nosotros hemos visto la función **cbind()** que concatena columnas de un data frame con otro pero, para asegurarnos que la unión se hace correctamente, necesitamos que las filas coincidan en ambos data frame.

Para evitar este problema y, siempre que tengamos uno o varios campos en común entre los dos data frames, se puede usar la función **merge()**.

merge(df1,df2,campo de unión)

Los tipos de unión que podemos hacer con la orden merge son:

- **Inner join.** La intersección entre los dos data frames.
- **Left join.** Los elementos que están en el primer data frame pero no en el segundo.
- **Right join.** Los elementos que están en el segundo data frame pero no en el primero.
- **Full join.** Todos los elementos independientemente del data frame al que pertenezcan.

Ejemplo

Vamos a unir dos data frames que contienen información por provincias de las emigraciones e inmigraciones.

El objetivo sería obtener un solo data frame donde esté la información de los dos data frames anteriores pero de manera coherente, es decir, los datos de cada provincia bien pegados.

Provincia	Cod_prov	Inmigraciones
Almería	4	20541
Granada	18	20867
Jaén	23	6508
Sevilla	41	26671

Provincia	Cod_prov	Emigraciones
Almería	4	19784
Cádiz	11	16827
Córdoba	14	9306
Granada	18	20149

Si estos dos data frames los unimos con la función **cbind()** simplemente los pega tal cual están.

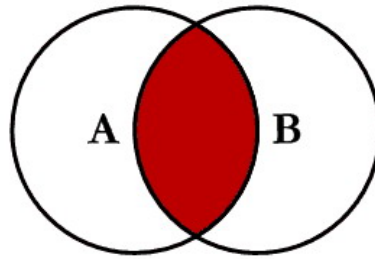
```
> cbind(inmigraciones,emigraciones)
  Provincia Cod_prov Inmigraciones Provincia Cod_prov Emigraciones
1  Almería      4      20541      Almería      4      19784
2  Granada     18      20867      Cádiz      11      16827
3   Jaén      23       6508     Córdoba     14       9306
4  Sevilla     41      26671     Granada     18      20149
```

En este caso no podemos usar el **cbind()** porque ha pegado mal los datos sin comprobar las provincias y mezclando información entre provincias distintas.

En esta situación lo correcto sería usar la orden **merge()** que busca los elementos comunes entre los dos data frames para realizar la unión.

6.1. Inner join

La unión **Inner join** entre dos data frames sería la intersección entre ellos, es decir, la parte común que aparecen en los dos data frames.



Ejemplo

Siguiendo con el ejemplo anterior, lo correcto sería usar la orden **merge()** que busca los elementos comunes entre los dos data frames para realizar la unión.

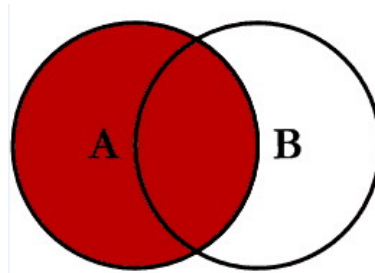
```
> merge(inmigraciones,emigraciones)
  Provincia Cod_prov Inmigraciones Emigraciones
1   Almería      4         20541         19784
2   Granada     18         20867         20149
```

En este caso toma como comunes las variables Provincia y Cod_prov porque se llaman igual en los dos data frames (por eso no hay que indicar en la orden que variables son las que identifican los casos).

Como resultado nos devuelve solamente las únicas provincias que están en los dos data frames que son Almería y Granada.

6.2. Left join

La unión **Left join** entre dos data frames serían los elementos que están los dos data frames pero con la información del primer data frame pero no del segundo.



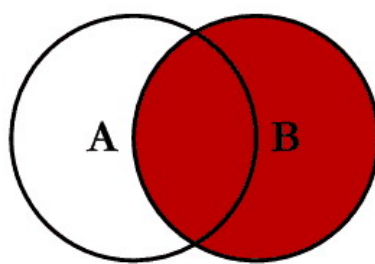
Ejemplo

Para realizar este tipo de unión debemos incluir el parámetro `all.x=T`, es decir, que incluya todos los elementos que están en el primer data frame aunque no tengan información en el segundo. La información que falta la asigna como nula (que en R es el valor NA).

```
> merge(inmigraciones,emigraciones,all.x =T)
  Provincia Cod_prov Inmigraciones Emigraciones
1  Almería      4         20541         19784
2  Granada     18         20867         20149
3   Jaén      23          6508            NA
4  Sevilla     41         26671            NA
```

6.3. Right join

La unión **Right join** entre dos data frames serían los elementos que están los dos data frames pero con la información del segundo data frame pero no del primero.



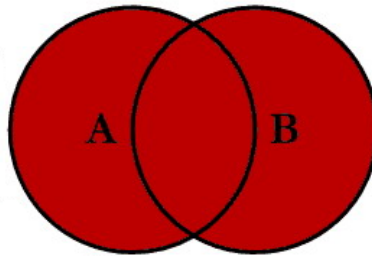
Ejemplo

Para realizar este tipo de unión debemos incluir el parámetro `all.y=T`, es decir, que incluya todos los elementos que están en el segundo data frame aunque no tengan información en el primero. La información que falta la asigna como nula (que en R es el valor NA).

```
> merge(inmigraciones,emigraciones,all.y=T)
  Provincia Cod_prov Inmigraciones Emigraciones
1   Almería      4         20541         19784
2    Cádiz      11           NA         16827
3  Córdoba      14           NA          9306
4   Granada      18         20867         20149
```

6.4. Full join

La unión **Full join** entre dos data frames serían los elementos que están los dos data frames independientemente de la información que contenga en cada uno de los data frames.



Ejemplo

Para realizar este tipo de unión debemos incluir el parámetro `all=T`, es decir, que incluya todos los elementos que están en el segundo data frame aunque no tengan información en el primero y viceversa. La información que falta la asigna como nula (que en R es el valor NA).

```
> merge(inmigraciones,emigraciones,all=T)
  Provincia Cod_prov Inmigraciones Emigraciones
1  Almería      4      20541      19784
2   Cádiz     11         NA      16827
3 Córdoba     14         NA       9306
4  Granada     18      20867      20149
5   Jaén      23       6508         NA
6 Sevilla     41      26671         NA
```

6.5. Otros parámetros de merge()

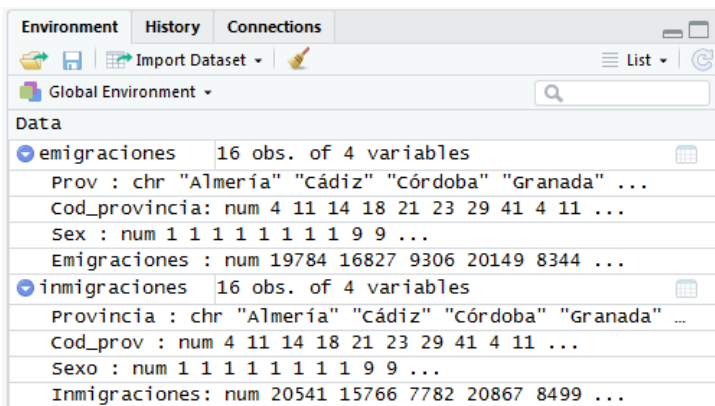
En el uso que le hemos dado a la orden **merge()** en los ejemplos solo hemos empleado los parámetros **all**, **all.x** y **all.y** para usar los distintos modos de unir dos data frames. Pero la orden merge tiene más opciones de las que señalamos las más interesantes:

- **by**. Para indicar los campos por los que se van a cruzar los data frames.
- **by.x**. Para indicar que campos son los que deben cruzar en el primer data frame.
- **by.y**. Para indicar que campos son los que deben cruzar en el segundo data frame.
- **sort**. Si queremos ordenar el resultado por alguna variable.

Estos parámetros debemos emplearlos cuando los nombres de las variables que unen los dos data frames son distintos.

Ejemplo

Ahora tenemos dos data frames con la misma información (emigraciones e inmigraciones) pero los nombres de las variables no coinciden.



Variable	emigraciones	inmigraciones
Prov	chr "Almería" "Cádiz" "Córdoba" "Granada" ...	chr "Almería" "Cádiz" "Córdoba" "Granada" ...
Cod_provincia	num 4 11 14 18 21 23 29 41 4 11 ...	num 4 11 14 18 21 23 29 41 4 11 ...
Sex	num 1 1 1 1 1 1 1 1 9 9 ...	num 1 1 1 1 1 1 1 1 9 9 ...
Emigraciones	num 19784 16827 9306 20149 8344 ...	
Inmigraciones		num 20541 15766 7782 20867 8499 ...

En este caso las variables de Provincia, Código de provincia y Sexo, que son las comunes en ambos data frame y por tanto son las que debemos usar para unirlos no se llaman igual.

Así que si tratamos de usar la función merge sin indicar ningún parámetro, al no encontrar nada en común, lo une todo con todo realizando un producto cartesiano entre los dos data frames.

```
> merge(inmigraciones,emigraciones)
  Provincia Cod_prov Sexo Inmigraciones Prov Cod_provincia Sex Emigraciones
1   Almería      4     1      20541 Almería      4     1      19784
2    Cádiz      11     1      15766 Almería      4     1      19784
3  Córdoba      14     1       7782 Almería      4     1      19784
4   Granada      18     1      20867 Almería      4     1      19784
5    Huelva      21     1       8499 Almería      4     1      19784
6     Jaén      23     1       6508 Almería      4     1      19784
7    Málaga      29     1      40181 Almería      4     1      19784
8    Sevilla      41     1      26671 Almería      4     1      19784
9   Almería       4     9      16012 Almería      4     1      19784
10    Cádiz      11     9      15483 Almería      4     1      19784
11  Córdoba      14     9       8076 Almería      4     1      19784
12  Granada      18     9      20552 Almería      4     1      19784
13    Huelva      21     9       9063 Almería      4     1      19784
14     Jaén      23     9       6482 Almería      4     1      19784
15    Málaga      29     9      40740 Almería      4     1      19784
16    Sevilla      41     9      27980 Almería      4     1      19784
17  Almería       4     1      20541 Cádiz      11     1      16827
18    Cádiz      11     1      15766 Cádiz      11     1      16827
19  Córdoba      14     1       7782 Cádiz      11     1      16827
20  Granada      18     1      20867 Cádiz      11     1      16827
21    Huelva      21     1       8499 Cádiz      11     1      16827
22     Jaén      23     1       6508 Cádiz      11     1      16827
23    Málaga      29     1      40181 Cádiz      11     1      16827
24    Sevilla      41     1      26671 Cádiz      11     1      16827
25   Almería       4     9      16012 Cádiz      11     1      16827
26    Cádiz      11     9      15483 Cádiz      11     1      16827
27  Córdoba      14     9       8076 Cádiz      11     1      16827
28  Granada      18     9      20552 Cádiz      11     1      16827
```

En esta situación debemos usar los parámetros **by.x** para indicar los nombres de las variables comunes del primer data frame y **by.y** para los nombres del segundo data frame. De esta forma conseguimos la unión correcta entre los dos data frames.

```
> merge(inmigraciones,emigraciones,by.x=c('Cod_prov','Sexo'),by.y=c('Cod_provincia','sex'))
```

	Cod_prov	Sexo	Provincia	Inmigraciones	Prov	Emigraciones
1	11	1	Cádiz	15766	Cádiz	16827
2	11	9	Cádiz	15483	Cádiz	15588
3	14	1	Córdoba	7782	Córdoba	9306
4	14	9	Córdoba	8076	Córdoba	9053
5	18	1	Granada	20867	Granada	20149
6	18	9	Granada	20552	Granada	19783
7	21	1	Huelva	8499	Huelva	8344
8	21	9	Huelva	9063	Huelva	7958
9	23	1	Jaén	6508	Jaén	8587
10	23	9	Jaén	6482	Jaén	8321
11	29	1	Málaga	40181	Málaga	39087
12	29	9	Málaga	40740	Málaga	38152
13	4	1	Almería	20541	Almería	19784
14	4	9	Almería	16012	Almería	15956
15	41	1	Sevilla	26671	Sevilla	26740
16	41	9	Sevilla	27980	Sevilla	26909