Bases de datos en R

Manuel (mramon@jccm.es), 26 a 29 de abril 2021

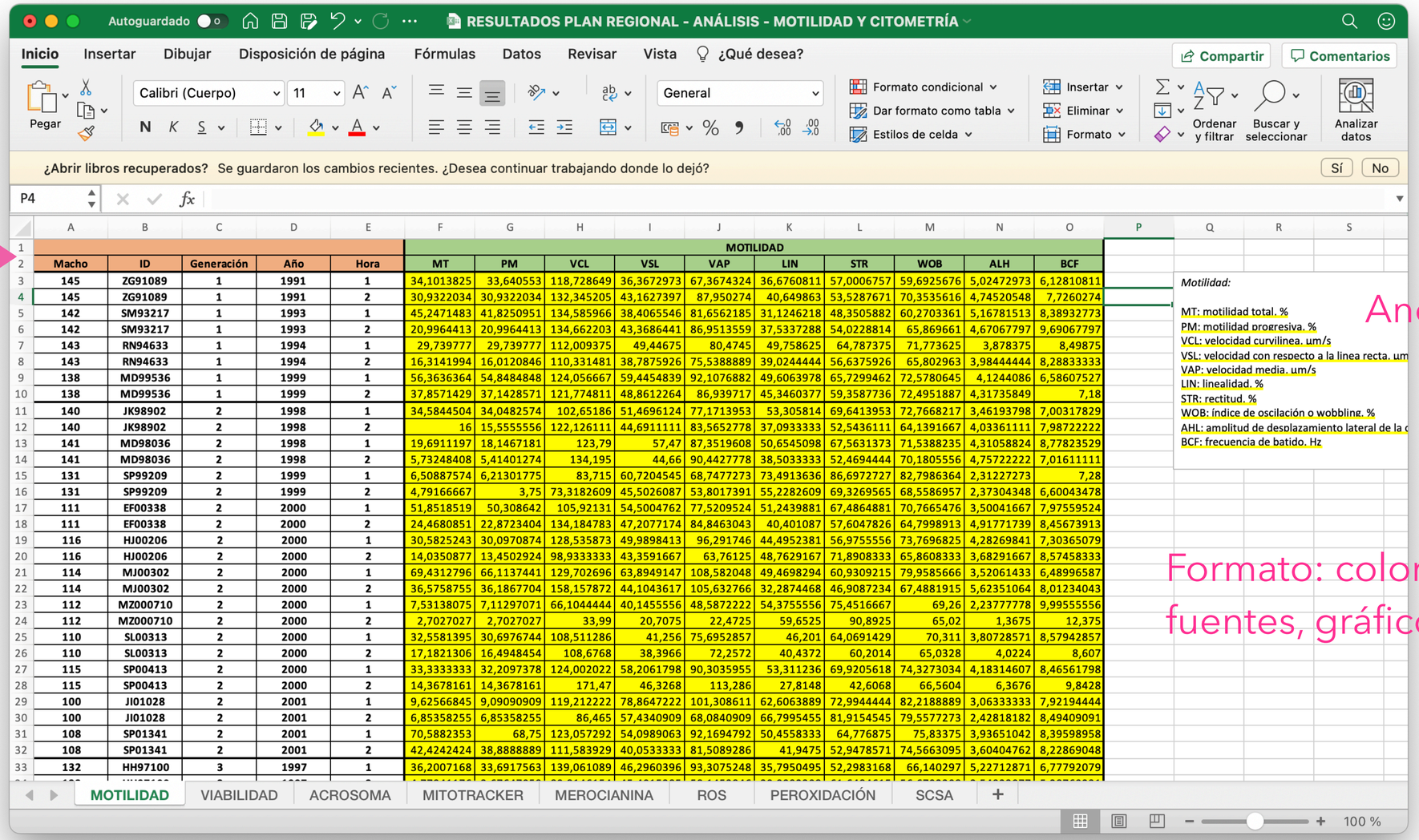
GUIÓN

- Bases de datos
- Operaciones básicas con bases de datos

¿Qué es una base de datos?

- Una base de datos es un objeto o programa que nos permite almacenar información de forma estructurada
- La información puede ser de naturaleza muy diversa
- Habitualmente, se tratará de información medida en individuos/unidades: edad de una oveja y número de partos, crecimiento de una variedad de cereal y la composición del abono, sujetos sometidos a un tratamiento experimentan e indicadores de salud/enfermedad, etc…

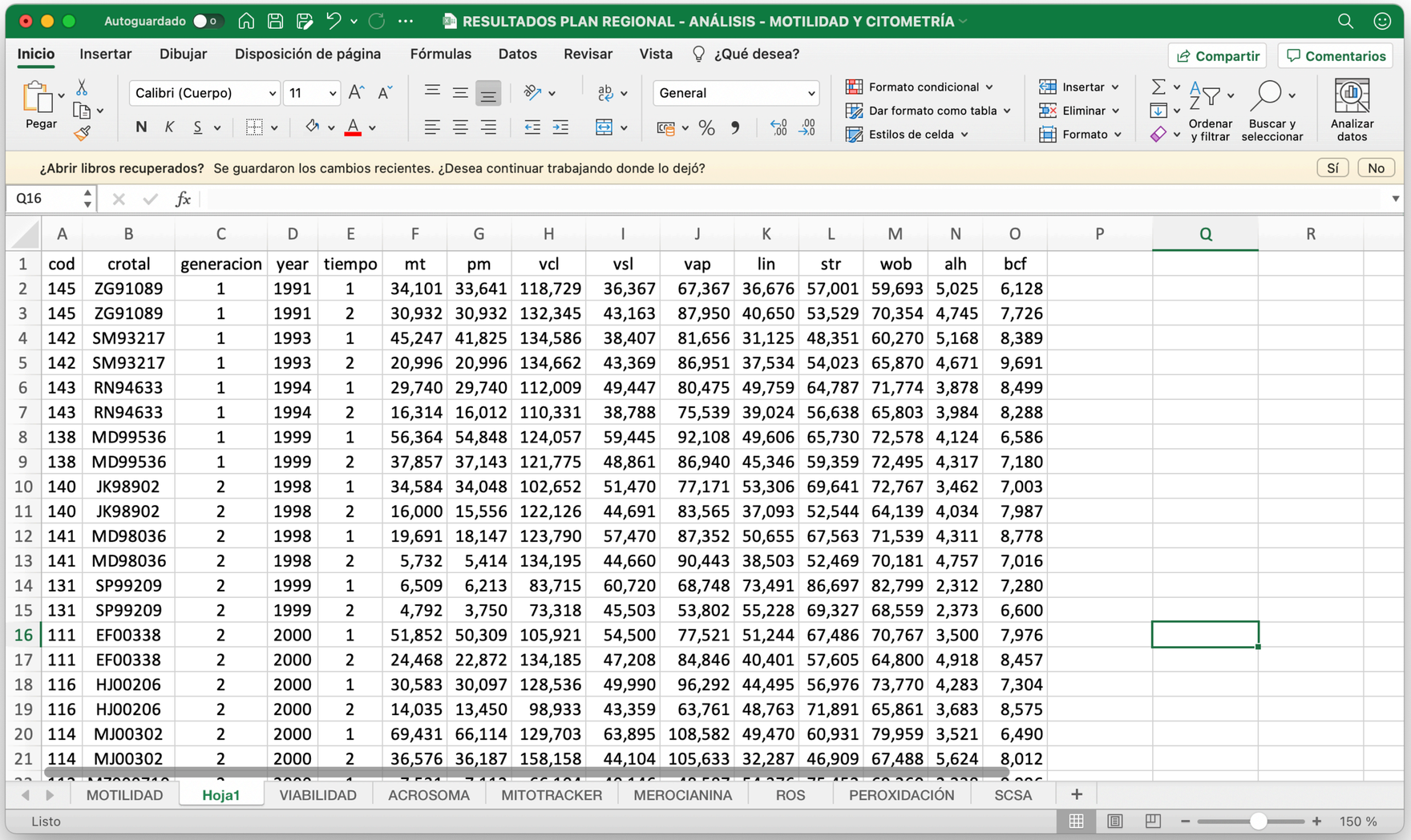
Encabezados complejos



Anotaciones

Formato: colores, fuentes, gráficos, etc

Al hablar de bases de datos solemos asociarlas a Microsoft Excel®



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	cod	crotal	generacion	year	tiempo	mt	pm	vcl	vsl	vap	lin	str	wob	alh	bcf			
2	145	ZG91089	1	1991	1	34,101	33,641	118,729	36,367	67,367	36,676	57,001	59,693	5,025	6,128			
3	145	ZG91089	1	1991	2	30,932	30,932	132,345	43,163	87,950	40,650	53,529	70,354	4,745	7,726			
4	142	SM93217	1	1993	1	45,247	41,825	134,586	38,407	81,656	31,125	48,351	60,270	5,168	8,389			
5	142	SM93217	1	1993	2	20,996	20,996	134,662	43,369	86,951	37,534	54,023	65,870	4,671	9,691			
6	143	RN94633	1	1994	1	29,740	29,740	112,009	49,447	80,475	49,759	64,787	71,774	3,878	8,499			
7	143	RN94633	1	1994	2	16,314	16,012	110,331	38,788	75,539	39,024	56,638	65,803	3,984	8,288			
8	138	MD99536	1	1999	1	56,364	54,848	124,057	59,445	92,108	49,606	65,730	72,578	4,124	6,586			
9	138	MD99536	1	1999	2	37,857	37,143	121,775	48,861	86,940	45,346	59,359	72,495	4,317	7,180			
10	140	JK98902	2	1998	1	34,584	34,048	102,652	51,470	77,171	53,306	69,641	72,767	3,462	7,003			
11	140	JK98902	2	1998	2	16,000	15,556	122,126	44,691	83,565	37,093	52,544	64,139	4,034	7,987			
12	141	MD98036	2	1998	1	19,691	18,147	123,790	57,470	87,352	50,655	67,563	71,539	4,311	8,778			
13	141	MD98036	2	1998	2	5,732	5,414	134,195	44,660	90,443	38,503	52,469	70,181	4,757	7,016			
14	131	SP99209	2	1999	1	6,509	6,213	83,715	60,720	68,748	73,491	86,697	82,799	2,312	7,280			
15	131	SP99209	2	1999	2	4,792	3,750	73,318	45,503	53,802	55,228	69,327	68,559	2,373	6,600			
16	111	EF00338	2	2000	1	51,852	50,309	105,921	54,500	77,521	51,244	67,486	70,767	3,500	7,976			
17	111	EF00338	2	2000	2	24,468	22,872	134,185	47,208	84,846	40,401	57,605	64,800	4,918	8,457			
18	116	HJ00206	2	2000	1	30,583	30,097	128,536	49,990	96,292	44,495	56,976	73,770	4,283	7,304			
19	116	HJ00206	2	2000	2	14,035	13,450	98,933	43,359	63,761	48,763	71,891	65,861	3,683	8,575			
20	114	MJ00302	2	2000	1	69,431	66,114	129,703	63,895	108,582	49,470	60,931	79,959	3,521	6,490			
21	114	MJ00302	2	2000	2	36,576	36,187	158,158	44,104	105,633	32,287	46,909	67,488	5,624	8,012			

Es preferible que la base de datos esté estructurada así

Recomendaciones bases de datos

- Filas = individuos o muestras
- Columnas = variables
- Usar nombres cortos, entendibles, sin espacios ni signos de puntuación poco comunes
- Si usamos unidades diferentes a las estándar, podemos indicarlo: peso_gr, altura_cm, tiempo_min
- Si medimos una misma variable varias veces en un individuo

Recomendaciones bases de datos

- Si medimos una misma variable varias veces en un individuo, es aconsejable no colocarla en columnas. Mejor usamos una sola columna para esa variable y añadimos otra columna que indique el tiempo

	A	B	C	D	E
1	id	peso_1m	peso_2m	peso_3m	
2	s001	5	12	21	
3	s002	4	10	17	
4	s003	5	10	20	
5					
6					

NO ✗

	A	B	C
1	id	mes	peso
2	s001	1	5
3	s001	2	12
4	s001	3	21
5	s002	1	4
6	s002	2	10
7	s002	3	17
8	s003	1	5

SÍ ✓

Recomendaciones bases de datos

- Es preferible no usar formatos bajo licencia o poco comunes. Por ejemplo, todo el mundo tiene acceso a un editor de texto (TXT, CSV), pero podría no tener acceso a Excel o Access.
- Algunos formatos tienen limitación en el número de filas/columnas que se pueden almacenar. Los archivos de texto plano no.
- Prestar atención a los signos de puntuación usados para indicar decimales. En Excel español, los decimales se separan por comas; en otros programas (R, por ejemplo) se usa un punto.
- Carácter usado para separar campos/columnas: espacio, tabulador, punto y coma, ancho fijo, …

Recomendaciones bases de datos

Entonces, ¿cómo almacenamos los datos? Mis sugerencias

- Almacenar los datos en archivos de texto CSV con los campos separados por puntos y comas (;)
- Incluir en la primera fila el nombre de las variables
- Los nombres no deben contener espacios
- Usar puntos (.) para separar decimales
- No usar ningún signo para separar miles, millones,…
- No usar símbolos tipo #,*,!,? que pueden dar problemas de lectura

Bases de datos en R

- R permite trabajar con bases de datos
- La base de datos más común en R se denomina `data.frame`
- Estas bases de datos pueden importarse a R (desde el disco dura o una online), crearse a mano en R, o bien existe la posibilidad de trabajar con bases de datos precargadas en R.

Bases de datos en R

- Para crear una base de datos en R usamos la función `data.frame()`

```
R> db0 <- data.frame(nombre = c("Mario","Camila","Jose","Ana") ,
                     edad = c(35,31,44,39),
                     genero=c("male","female","male","female"))

R> db0
  nombre edad genero
1  Mario   35   male
2 Camila   31 female
3   Jose   44   male
4    Ana   39 female
```

Bases de datos en R

- Al igual que vimos para vectores y matrices, podemos acceder a los elementos de una base de datos usando índices entre corchetes:

```
R> dim(db0) # nos devuelve el número de filas y columnas de
nuestra base
[1] 4 3
R> db0[1, ] # primera fila
  nombre edad genero
1  Mario   35   male
R> db0[, 1] # primera columna
[1] "Mario"  "Camila" "Jose"   "Ana"
```

Bases de datos en R

- Sin embargo, es más común acceder a los nombres de las variables usando el símbolo del dólar (`$`) :

```
R> db0$nombre
[1] "Mario"  "Camila" "Jose"   "Ana"
R> db0$edad
[1] 35 31 44 39
R> db0$genero
[1] "male"   "female" "male"   "female"
```

Bases de datos en R

- Además, podemos usar como índice entre corchetes el nombre de las variables:

```
R> db0[, "nombre"]
[1] "Mario"  "Camila" "Jose"   "Ana"
R> db0[, c("nombre","edad")]
  nombre edad
1  Mario   35
2 Camila   31
3   Jose   44
4    Ana   39
```

Información general de la base de datos

- La función `names()` nos dice que variables tenemos
- La función `str()` nos da información general sobre nuestra base de datos. Es una de las funciones más interesantes de R.

```
R> str(db0)
'data.frame':       4 obs. of  3 variables:
 $ nombre: chr  "Mario" "Camila" "Jose" "Ana"
 $ edad  : num  35 31 44 39
 $ genero: chr  "male" "female" "male" "female"
```

Información general de la base de datos

- La función `summary()` nos presenta un pequeño resumen de cada variable en nuestra base de datos. El resumen varía en función del tipo de variable (carácter, numérica, etc)

```
R> summary(db0)
    nombre                edad          genero
 Length:4           Min.   :31.0   Length:4
 Class :character   1st Qu.:34.0   Class :character
 Mode  :character   Median :37.0   Mode  :character
                    Mean   :37.2
                    3rd Qu.:40.2
                    Max.   :44.0
```

Información general de la base de datos

- La función `table()` nos presenta los diferentes niveles de una variable cualitativa y cuántas observaciones para cada nivel

```
R> unique(db0$genero) # nos dice los distintos niveles
[1] "male"   "female"
R> table(db0$genero)  # nos da niveles y número de obs por nivel
female   male
     2      2
R> prop.table(table(db0$provincia)) # nos da porcentajes
Albacete   Cuenca   Toledo
    0.25     0.50     0.25
```

Tu turno!

Abre el script "`bases_datos.R`" y ejecuta los comandos, prestando atención a las salidas.

Crear y modificar variables de una base de datos

- Para crear una variable nueva en una base de datos, simplemente hay que darle un nombre y definir que valores tendrá
- La variable debe tener el mismo número de observaciones que nuestra base de datos

```
R> db0$provincia <- c("Toledo","Cuenca","Cuenca","Albacete")
R> db0
  nombre edad genero provincia
1  Mario   35   male    Toledo
2 Camila   31 female    Cuenca
3   Jose   44   male    Cuenca
4    Ana   39 female  Albacete
R> db0$provincia <- c("Toledo","Cuenca","Cuenca","Albacete", "Ciudad Real")
Error in `$<-.data.frame`(`*tmp*`, provincia, value = c("Toledo", "Cuenca",  :
  replacement has 5 rows, data has 4
```

Crear y modificar variables de una base de datos

- De igual forma, podemos modificar una variable

```
R> db0
  nombre edad genero provincia altura
1  Mario   35   male    Toledo   1.83
2 Camila   31 female    Cuenca   1.74
3   Jose   44   male    Cuenca   1.77
4    Ana   39 female  Albacete   1.69
R> db0$altura <- db0$altura*100 # altura en cm
R> db0$altura
[1] 183 174 177 169
```

Crear y modificar variables de una base de datos

- Para eliminar una variable, le asignamos un valor nulo (NULL en R)

```
R> db0$genero <- NULL
R> db0
  nombre edad provincia altura
1  Mario   35    Toledo    183
2 Camila   31    Cuenca    174
3   Jose   44    Cuenca    177
4    Ana   39  Albacete    169
```

Ordenar una base de datos

- Existen diversas funciones para ordenar bases de datos: `sort()`,`order()` o `arrange()` del paquete `dplyr`

```
db0$nombre

sort(db0$nombre)

sort(db0$nombre, decreasing = TRUE) # ordena de mayor a mayor, o de Z a A

R> order(db0$nombre) # que diferencia hay con la función sort?

[1] 4 2 3 1
```

Ordenar una base de datos

- Para ordenar una base de datos por una columna:

```
R> db0[order(db0$nombre), ]
   nombre edad provincia altura
4     Ana   39  Albacete    169
2  Camila   31    Cuenca    174
3    Jose   44    Cuenca    177
1   Mario   35    Toledo    183
R> db0[order(db0$edad, decreasing = TRUE), ]
```

Fijaros que aunque ordenamos por una columna, el argumento va delante de la coma, lo que correspondería a las filas. Es así, porque ordenamos las filas en base al valor de una columna, en este caso el nombre

Ordenar una base de datos

```
R> library(dplyr)
R> arrange(db0, provincia, -edad) # ordenamos por 2 campos
  nombre edad provincia altura
1    Ana   39  Albacete    169
2   Jose   44    Cuenca    177
3 Camila   31    Cuenca    174
4  Mario   35    Toledo    183
```

Ordenar una base de datos

- IMPORTANTE: aseguraros de ordenar la base de datos y no una columna (es como cuando en Excel os pregunta si queréis expandir a toda la base de datos)

```
R> db0$nombre <- sort(db0$nombre)
R> db0
  nombre edad provincia altura
1    Ana   35    Toledo    183
2 Camila   31    Cuenca    174
3   Jose   44    Cuenca    177
4  Mario   39  Albacete    169
```

HEMOS ALTERADO LA CORRESPONDENCIA DE DATOS!! Nuestra base de datos ya no es correcta

Combinar datos

- Existe 2 funciones muy útiles que nos permiten concatenar datos
- La función `cbind()` concatena por columnas, es decir, pega unas columnas a continuación de otras. Es requisito que las dos bases de datos tengan el mismo número de observaciones (filas)
- La función `rbind()` concatena por filas, es decir, pega una base de datos debajo de otra. Es requisito que las 2 bases de datos tengan las mismas variables

Combinar datos

```
R> db1 <- data.frame(peso = c(75,62,84,56),
+                     hijos = c(0,0,3,1))
R> db0 <- cbind(db0, db1) # añado dos columnas
R> db0
  nombre edad provincia altura peso hijos
1  Mario   35    Toledo    183   75     0
2 Camila   31    Cuenca    174   62     0
3   Jose   44    Cuenca    177   84     3
4    Ana   39  Albacete    169   56     1
```

Combinar datos

```
R> db2 <- data.frame(nombre="Dario", edad=51, provincia="Ciudad Real",
+                     altura=179, peso=77, hijos=2)
R> db0 <- rbind(db0, db2)
R> db0
  nombre edad   provincia altura peso hijos
1  Mario   35      Toledo    183   75     0
2 Camila   31      Cuenca    174   62     0
3   Jose   44      Cuenca    177   84     3
4    Ana   39    Albacete    169   56     1
5  Dario   51 Ciudad Real    179   77     2
```

Combinar datos

- Además, existe una tercera función que nos permite combinar bases de datos a partir de la información de un campo común: `merge()`

```
R> db3 <- data.frame(nombre = db0$nombre,
+                    coche = c("no","si","si","no","si"))
R> db0 <- merge(db0, db3, by = "nombre")
R> db0
```

	nombre	edad	provincia	altura	peso	hijos	coche
1	Ana	39	Albacete	169	56	1	no
2	Camila	31	Cuenca	174	62	0	si
3	Dario	51	Ciudad Real	179	77	2	si
4	Jose	44	Cuenca	177	84	3	si
5	Mario	35	Toledo	183	75	0	no

bd0 bd3

Subconjuntos de datos

- Para seleccionar datos de una base de datos en función de diversos criterios, usaremos la función `subset()`. Se trata de otra función de gran utilidad para trabajar con datos

```
R> subset(db0, hijos>0) # seleccionamos todos los que tengan hijos
  nombre edad   provincia altura peso hijos coche
1    Ana   39    Albacete    169   56     1    no
3  Dario   51 Ciudad Real    179   77     2    si
4   Jose   44      Cuenca    177   84     3    si
```

Subconjuntos de datos

```
R> subset(db0, provincia == "Cuenca")
  nombre edad provincia altura peso hijos coche
2 Camila   31    Cuenca    174   62     0    si
4   Jose   44    Cuenca    177   84     3    si

R> subset(db0, hijos!=0 & coche=="si")
  nombre edad   provincia altura peso hijos coche
3  Dario   51 Ciudad Real    179   77     2    si
4   Jose   44      Cuenca    177   84     3    si
```

Subconjuntos de datos

- Además de seleccionar datos en base a unas condiciones, podemos seleccionar variables

```
R> subset(db0, subset = altura>175, select =
c("nombre","altura","peso"))
  nombre altura peso
3  Dario    179   77
4   Jose    177   84
5  Mario    183   75
```

Tu turno!

Abre el script "`bases_datos.R`" y ejecuta los comandos, prestando atención a las salidas. A continuación haz los ejercicios de "`bases_datos_ex.R`"