

Salvando datos

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

Imprimido por: ALFONSO LUIS MONTEJO RAEZ
Día: lunes, 11 de abril de 2022, 10:55

Tabla de contenidos

1. Introducción
2. Formato CSV
3. Formato Excel
4. Otros formatos

1. Introducción

Hay ocasiones en las que tenemos que guardar un conjunto de datos con el que hemos trabajado en R pero **usando otros formatos de fichero**.

Hasta ahora hemos aprendido a salvar los datos en formato R, pero ese formato no podemos abrirlo con otros programas. Por ejemplo, si queremos trabajar con unos datos y llevarlos a Excel no podemos hacerlo directamente.

Para evitar este problema vamos a ver como se salvan datos de R pero en otros formatos como csv o Excel.

2. Formato CSV

Para salvar en csv un conjunto de datos que hemos trabajado en R podemos usar las órdenes **write.csv()** y **write.csv2()**.

El uso básico de estas órdenes sería:

write.csv(objeto de R a salvar, nombre del fichero csv)

Los parámetros que podemos definir en el uso de estas funciones son:

- **sep**. Carácter que separa los campos. Normalmente puede ser una coma o punto y coma.
- **dec**. Carácter que se usa para separar los decimales.
- **fileEncoding**. Para seleccionar la codificación del fichero (UTF-8, ANSI, WINDOWS-1252, etc.)
- **row.names**. Para añadir en el csv el número de fila que tiene en el data frame de R. Podemos elegir T o TRUE (si queremos que estén) o FALSE si queremos que no estén).
- **file**. Nombre que va a tener el csv que creamos. El archivo se guarda en el directorio activo que tenemos seleccionado.

La diferencia entre **write.csv** y **write.csv2** son los parámetros que están definidos por defecto.

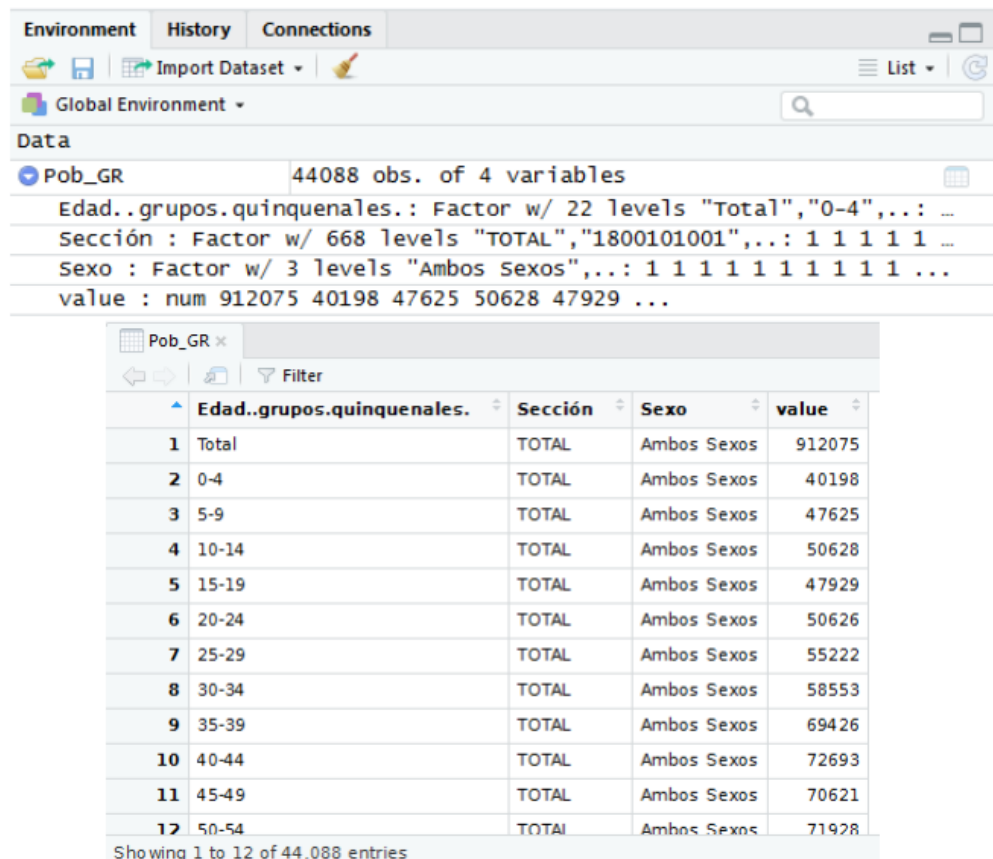
En el caso de **write.csv()** el separador de campos es la coma y el separador de decimales es el punto.

Sin embargo, si usamos **write.csv2()**, el separador de campos es el punto y coma y el separador de decimales es la coma.

Nosotros, con la forma en la que definimos los datos en España, debemos usar la orden **write.csv2()** ya que nos ahorraremos definir los parámetros que ya vienen por defecto seleccionados.

Ejemplo

Vamos a trabajar la información del **fichero 1801_2018.px**, que ya hemos usado con anterioridad, y que contiene información de población de la provincia de Granada.



The screenshot shows the R Studio interface. The 'Environment' pane displays a data object 'Pob_GR' with 44088 observations and 4 variables. Below this, a preview of the data is shown in a table format. The table has four columns: 'Edad..grupos.quinquenales.', 'Sección', 'Sexo', and 'value'. The first 12 rows are visible, showing population data for different age groups and sections.

	Edad..grupos.quinquenales.	Sección	Sexo	value
1	Total	TOTAL	Ambos Sexos	912075
2	0-4	TOTAL	Ambos Sexos	40198
3	5-9	TOTAL	Ambos Sexos	47625
4	10-14	TOTAL	Ambos Sexos	50628
5	15-19	TOTAL	Ambos Sexos	47929
6	20-24	TOTAL	Ambos Sexos	50626
7	25-29	TOTAL	Ambos Sexos	55222
8	30-34	TOTAL	Ambos Sexos	58553
9	35-39	TOTAL	Ambos Sexos	69426
10	40-44	TOTAL	Ambos Sexos	72693
11	45-49	TOTAL	Ambos Sexos	70621
12	50-54	TOTAL	Ambos Sexos	71928

Showing 1 to 12 of 44,088 entries

Para cargar los datos usamos el siguiente código.

```
# Cargamos el paquete
library(pxR)

# Cargamos el fichero
read.px('1801_2018.px')->Pob_GR
as.data.frame(Pob_GR)->Pob_GR
```

Estos datos vamos a filtrarlos para quedarnos con los datos provinciales (solo con los que tienen el campo Sección igual a TOTAL) y sin distinguir sexo (el campo Sexo igual a Ambos Sexos), y solo las variables Edad y Población.

```
# Renombramos variables
names(Pob_GR)[1]<- 'Edad'
names(Pob_GR)[4]<- 'Población'

Pob_GR[Pob_GR$Sección=='TOTAL' &
  Pob_GR$Sexo=='Ambos Sexos',c(1,4)]->Pob_GR
```

Ya tenemos los datos como queremos y vamos a salvar el objeto Pob_GR, que es un data frame, como un archivo csv al que vamos a llamar Pob_Prov_GR.csv. El separador de campos va a ser la coma.

```
# Salvamos en csv (valores separados por ,)
write.csv(Pob_GR,row.names=F,file='Pob_Prov_GR.csv')
```

Si usamos como separador el punto y coma.

```
# Salvamos en csv (valores separados por ;)
write.csv2(Pob_GR,row.names=F,file='Pob_Prov_GR.csv')
```

Recordad que el archivo csv se guarda en el directorio activo. El resultado quedaría así:

Fichero
generado
con write.csv

	Pob_Prov_GR.csv
1	"Edad","Población"
2	"Total",912075
3	"0-4",40198
4	"5-9",47625
5	"10-14",50628
6	"15-19",47929
7	"20-24",50626
8	"25-29",55222
9	"30-34",58553
10	"35-39",69426
11	"40-44",72693
12	"45-49",70621
13	"50-54",71928
14	"55-59",63642
15	"60-64",51523
16	"65-69",43442
17	"70-74",38305
18	"75-79",28501
19	"80-84",26567
20	"85-89",16947
21	"90-94",6154
22	"95-99",1304
23	"100 y más",241

Fichero
generado con
write.csv2

	Pob_Prov_GR.csv
1	"Edad";"Población"
2	"Total";912075
3	"0-4";40198
4	"5-9";47625
5	"10-14";50628
6	"15-19";47929
7	"20-24";50626
8	"25-29";55222
9	"30-34";58553
10	"35-39";69426
11	"40-44";72693
12	"45-49";70621
13	"50-54";71928
14	"55-59";63642
15	"60-64";51523
16	"65-69";43442
17	"70-74";38305
18	"75-79";28501
19	"80-84";26567
20	"85-89";16947
21	"90-94";6154
22	"95-99";1304
23	"100 y más";241

3. Formato Excel

También podemos guardar un data frame de R en formato Excel, pero necesitamos la ayuda de un paquete externo llamado **writexl**.

Al ser un paquete que no está en la base de R, hay que instalarlo y cargarlo antes de su uso.

La orden básica para salvar en Excel sería

write.xlsx(data frame de R a salvar, nombre del fichero Excel)

Los parámetros que podemos definir en el uso de esta función son:

- **path**. Donde indicamos el nombre del fichero que queremos crear.
- **col_names**. Que incluye el nombre de las variables en la primera fila del Excel.

Ejemplo

Con el data frame que teníamos antes abierto (Pob_GR), vamos a salvarlo en formato Excel.

```
# Salvar en formato Excel
library(writexl)
write.xlsx(Pob_GR,col_names = T,path='Pob_Prov_GR.xlsx')
```

Y el resultado sería:

	A	B	C
1	Edad	Población	
2	Total	912075	
3	0-4	40198	
4	5-9	47625	
5	10-14	50628	
6	15-19	47929	
7	20-24	50626	
8	25-29	55222	
9	30-34	58553	
10	35-39	69426	
11	40-44	72693	
12	45-49	70621	
13	50-54	71928	
14	55-59	63642	
15	60-64	51523	
16	65-69	43442	
17	70-74	38305	
18	75-79	28501	
19	80-84	26567	
20	85-89	16947	
21	90-94	6154	
22	95-99	1304	
23	100 y más	241	
24			

4. Otros formatos

Con la ayuda del **paquete foreign** podemos guardar un data frame de R en otros formatos como SPSS, SAS, etc.

La orden básica para guardar sería:

write.foreign(data frame de R a salvar, nombre del fichero a crear, paquete)

Los parámetros básicos que podemos definir son:

- **package**. Sistema en el que queremos guardar el archivo (SPSS, Stata, SAS, etc.)
- **datafile**. Nombre del fichero a crear con los datos en SPSS.
- **codefile**. Nombre del fichero con la codificación de las variables usadas en SPSS para crear el archivo sav.

Ejemplo

Con el data frame que teníamos antes abierto (Pob_GR), vamos a salvarlo en formato SPSS.

```
# Salvamos en sav (Formato de SPSS)
foreign::write.foreign(Pob_GR, datafile='Pob_Prov_GR.sav',
                      codefile='Pob_Prov_gr',
                      package='SPSS')
```

En este ejemplo hemos usado la orden write del paquete foreign sin tener que cargarlo, empleando ::

Con esta orden se crean dos ficheros. Por un lado los datos en formato SPSS (extensión .sav) y por otro un archivo con la codificación y definición de las variables en formato SPSS.

Archivo de código generado.

```
Pob_Prov_gr
1 SET DECIMAL=DOT.
2
3 DATA LIST FILE= "Pob_Prov_GR.sav"
4 ENCODING="Locale"
5 / Edad (F8.0) Población
6 .
7
8 VARIABLE LABELS
9 Edad "Edad"
10 Población "Población"
11 .
12
13 VALUE LABELS
14 /
15 Edad
16 1 "Total"
17 2 "0-4"
18 3 "5-9"
19 4 "10-14"
20 5 "15-19"
21 6 "20-24"
22 7 "25-29"
23 8 "30-34"
24 9 "35-39"
25 10 "40-44"
26 11 "45-49"
27 12 "50-54"
28 13 "55-59"
29 14 "60-64"
30 15 "65-69"
31 16 "70-74"
32 17 "75-79"
33 18 "80-84"
34 19 "85-89"
35 20 "90-94"
36 21 "95-99"
37 22 "100 y más"
38 .
39 VARIABLE LEVEL Población
40 (scale).
41
42 EXECUTE.
```