Datos: Lectura y escritura

José Enrique Martín García
Universidad Politécnica de Gimialcón

Bases de Datos

- En R básico están incluidas varias bases de datos que vienen en el paquéte "datasets“. Para ver la lista completa: library(help='datasets').
- En otros paquetes también hay bases de datos, pero no están disponibles de forma automática, si no que generalmente hay que usar la función "data()"
- Data() lista los bancos de datos disponibles en el package datasets, mientras que data(package = .packages(all.available = TRUE)) lista todos los disponibles en las librerías que hayamos cargado anteriormente.
- La función data(nom.banco.datos) carga el banco de datos para que podamos utilizarlo.

Bases de Datos

- La instrucción data permite cargar archivos de las librerías disponibles.

```
> data() # muestra todos los archivos
> data(iris) y  > data(iris, package = "base") # equivalente
> ?iris
```

- Cuando nos interese saber si nuestras tablas quedaron bien importadas o queremos hacer una visualización rápida de como son las variables y sus valorescutilizamos dos funciones que nos dan a conocer las 6 primeras o últimas filas de nuestra tabla. head y tail respectivamente.

```
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
  Species
1  setosa
2  setosa
3  setosa
4  setosa
5  setosa
6  setosa
```

Bases de Datos

- La función genérica 'str' es sumamente práctica para obtener información básica y resumida de cualquier objeto, incluyendo tablas de datos. Por ejemplo:

```
> str(iris)
'data.frame': 150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

- Lo primero que indica es la clase del objeto, en este caso "data.frame".
- Además muestra las dimensiones del mismo (150 observaciones/filas, 5 variables/columnas).
- Luego muestra las variables que componen al set de datos, usando el símbolo de '$' para indicar el nombre de cada una.
- En cada caso muestra la clase del objeto en cuestión ('num' por 'numeric', 'int' por 'integer', etc...).
- Por último aparecen los primeros valores, seguidos por puntos suspensivos.

Bases de Datos

- La función genérica “Sumary()” es muy flexible, abarcando métodos para la mayoría de las clases de objetos, Literalmente 'summary' es un sumario o resumen de un objeto.
- Con la tabla 'iris' podemos encontrar dos tipos de salida, una para variables numérica y otra categórica (factor).

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

- Cuando se trata con variables numéricas, como se puede ver en este ejemplo, se calculan algunos estadísticos de diagnóstico, como el promedio o los cuartiles.
- Para el caso del factor 'iris$Species' simplemente muestra los niveles y la cantidad de veces que ocurren en la tabla.

Importar datos formato .txt

- Para importar datos cuyo formato sea .txt separados por tabulaciones se puede usar tanto la función read.delim o read.table-
- El argumento sep que indica la forma en como se ha separado las columnas en el archivo, las alternativas son sep=',' para aquellos que son separados por coma, sep=';' para los que son separados por punto y coma (;) y sep='\t' para los que han sido separados con tabulaciones, este es el caso de nuestros datos del ejemplo.
- La función read.table también permite importar datos txt desde una web lo cual es muy útil si no queremos descargar los datos y almencenarlos en el disco duro, aunque siempre es recomendable descargarlos. La instrucción es muy simple, sólo hasta escribir read.table('dirección web') y listo.

```
> web <- "http://people.cst.cmich.edu/lee1c/spss/V16_materi
als/DataSets_v16/Diseaseoutbreak.txt"
> datosWeb1 <- read.table(web)  # o puede escribirse la dir
ección directamente dentro
> head(datosWeb1)  # vemos como queda
  V1 V2 V3 V4 V5 V6
1  1 33  1  1  0  1
2  2 35  1  1  0  1
3  3  6  1  1  0  0
4  4 60  1  1  0  1
5  5 18  3  1  1  0
6  6 26  3  1  0  0
```

Lectura de datos formato ASCII

- La función read.table permite leer datos desde ficheros en formato ASCII. Devuelve como resultado un data.frame, por tanto, se supone que cada línea contiene los datos para un individuo.
- Para leer un fichero simple, con los datos separados por espacios en blanco, tabuladores o saltos de línea, se utiliza la instrucción read.table en la forma:

```
> fichero.df <- read.table("c:/dir/fichero.txt",
+ header = TRUE, sep = "",
+ comment.char = "")
```

- Se pueden saltar líneas (skip) o leer un número fijo de líneas (nrows).
- Existen más argumentos de entrada que se pueden utilizar con la función read.table:

```
read.table(file, header = FALSE, sep = "", quote = "\"’", dec=".",
row.names, col.names, as.is = FALSE, na.strings = "NA",
colClasses = NA, nrows = -1, skip = 0, check.names=TRUE, fill =
!blank.lines.skip, strip.white= FALSE, blank.lines.skip   = TRUE,comment.char =
   "#")
```

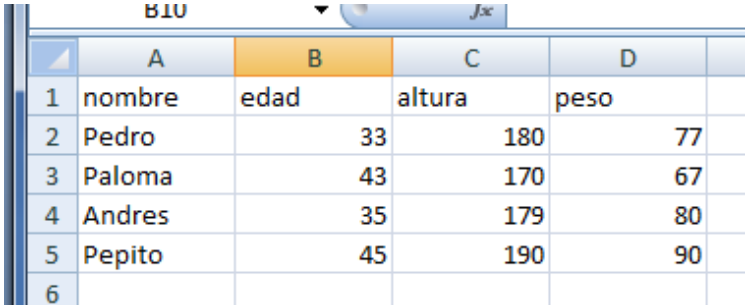
Lectura de datos

Argumentos de entrada

`file`	el nombre del archivo (entre "" o como una variable de tipo caracter), posiblemente con su dirección si se encuentra en un directorio diferente al de trabajo (el símbolo \no es permitido y debe reemplazarse con /, inclusive en Windows), o una dirección remota al archivo tipo URL (http://...)
`header`	una variable lógica (`FALSE` (falso) o `TRUE` (verdadero)) indicando si el archivo contiene el nombre de las variables en la primera fila o línea
`sep`	el separador de campo usado en el archivo; por ejemplo `sep="\t"` si es una tabulación
`quote`	los caracteres usados para citar las variables en modo caracter
`dec`	el caracter usado para representar el punto decimal
`row.names`	un vector con los nombres de las líneas de tipo caracter o numérico (por defecto: `1, 2, 3, ...`)
`col.names`	un vector con los nombres de las variables (por defecto: `V1, V2, V3, ...`)
`as.is`	controla la conversión de variables tipo caracter a factores (si es `FALSE`) o las mantiene como caracteres (`TRUE`); `as.is` puede ser un vector lógico o numérico que especifique las variables que se deben mantener como caracteres
`na.strings`	el valor con el que se codifican datos ausentes (convertido a `NA`)
`colClasses`	un vector de caracteres que proporciona clases para las columnas
`nrows`	el número máximo de líneas a leer (se ignoran valores negativos)
`skip`	el número de líneas ignoradas antes de leer los datos
`check.names`	si es `TRUE`, chequea que el nombre de las variables sea válido para R
`fill`	si es `TRUE` y todas las filas no tienen el mismo número de variables, agrega "blancos"
`strip.white`	(condicional a `sep`) si es `TRUE`, borra espacios extra antes y despues de variables tipo caracter
`blank.lines.skip`	si es `TRUE`, ignora líneas en "blanco"
`comment.char`	un caracter que define comentarios en el archivo de datos; líneas que comienzen con este caracter son ignoradas en la lectura (para desactivar este argumento utilize `comment.char =""`)

Importar datos .txt creado con EXCEL

- El fichero EXCEL ejemplo1.xls tiene el siguiente aspecto:



	A	B	C	D
1	nombre	edad	altura	peso
2	Pedro	33	180	77
3	Paloma	43	170	67
4	Andres	35	179	80
5	Pepito	45	190	90
6				

- Guardamos el fichero EXCEL como un fichero ASCII delimitado por tabulaciones y lo leemos con R

```
> setwd("C:/R/proyectos/UPG")
> personas<-read.table(file="ejemplo1.txt",header=T)
> personas
  nombre edad altura peso
1  Pedro   33    180   77
2 Paloma   43    170   67
3 Andres   35    179   80
4 Pepito   45    190   90
>
```

Importar datos txt desde una web

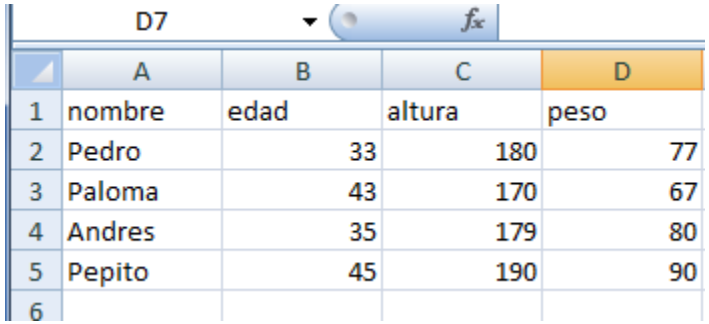
- La función read.table también permite importar datos txt desde una web lo cual es muy útil si no queremos descargar los datos y almencenarlos en el disco duro, aunque siempre es recomendable descargarlos,

```
> web <- "http://people.cst.cmich.edu/lee1c/spss/V16_material
s/DataSets_v16/Diseaseoutbreak.txt"
> datosweb1 <- read.table(web)
> head(datosweb1)
  V1 V2 V3 V4 V5 V6
1  1 33  1  1  0  1
2  2 35  1  1  0  1
3  3  6  1  1  0  0
4  4 60  1  1  0  1
5  5 18  3  1  1  0
6  6 26  3  1  0  0
>
```

- Se observará que tales datos no tienen encabezado (las columnas no tienen nombres) con lo cual se ha omitido el argumento header, al ser omitido este toma el valor FALSE que es su valor por defecto. Como las columnas no tienen nombres, entonces R ha creado nombres: V1, V2, ..., V6,

Importar datos con copy/paste

- Para usar el copiar y pegar en R basta con seleccinar los datos a ser importados (preferiblemente de Excel o de un formato .txt) y copiar (Ctrl + C)
- Ctrl + V en R no funciona, su equivalente es read.delim('clipboard').
- A manera de ejemplo se importarán los datos de las tres primeras filas de este fichero de Excel



D7

	A	B	C	D
1	nombre	edad	altura	peso
2	Pedro	33	180	77
3	Paloma	43	170	67
4	Andres	35	179	80
5	Pepito	45	190	90
6				

- Sólo basta con seleccionar todos los datos hacer Ctrl + C y luego ejecutar lo siguiente:

```
> datosCTR_c <- read.delim('clipboard')
> datosCTR_c
  nombre edad altura peso
1  Pedro   33    180   77
2 Paloma   43    170   67
```

Otras formas de leer y guardar datos

- La función save(objetos, list = character(0), file = "nomfich.ext", ascii = FALSE) nos permite guardar los objetos que queramos en el nomfich.ext.
- load() nos permite leer datos de un fichero binario que contiene los objetos de R previamente guardados con save().
- Como vimos en la primera sesión, cuando cerramos R existe la posibilidad de guardar todos los objetos de la sesión de trabajo en archivos .Rdata.
- En realidad, al grabar toda la imagen R está aplicando la función save.image(), un caso particular de la función save(list=ls(),file=".Rdata").
- Al cargar los datos desde el menú Archivo>cargar área de trabajo o hacer doble click sobre el chero .RData en el fondo estamos utilizando la función load().

Importar datos de otros programas

- Existen otras opciones para conectar R con diferentes bases de datos (Access, Excel, MySQL, dBase, etc.).
- La idea es utilizar el estándar de acceso a bases de datos utilizado por la mayoría de bases de datos existentes, lo que se conoce como Open DataBase Connectivity (ODBC).
- A través de ODBC, un sistema puede conectarse a cualquier base de datos (que esté local en nuestro ordenador o incluso que esté remota) que soporte ODBC sin necesidad de conocer sus características específicas.
- En el caso de R, existe una librería llamada RODBC que permite conectarse mediante un ODBC y desarrollar consultas contra bases de datos.
- La librería foreign contiene funciones para importar y exportar datos en el formato de otros programas.
- Utilizando help(package="foreign") podemos encontrar una lista de funciones y formatos soportados de otros programas estadísticos (SPSS, SAS, STATA, Minitab, SPlus, EpiInfo, Systat) y numéricos (Octave).

Importar datos de Excel

- Lo mejor es exportar los datos desde Excel a un archivo de texto separado por tabuladores.
- Cuidado con las últimas columnas y missing data (Excel elimina los “trailing tabs”). Cuidado también con líneas extra al final del fichero.
- Salvamos como texto (sólo salvamos una de las hojas). Importamos en R con read.table.
- La librería RODBC permite conectarse mediante un ODBC a un fichero Excel.

```
> setwd("C:/R/proyectos/UPG")
> library(RODBC) # Cargamos el paquete
> conexion<-odbcConnectExcel("ejemplo1.xls")
```

- En el caso de tener instalado R con 64 bits podemos tener el siguiente error

```
Error in odbcConnectExcel("ejemplo1.xls") :
  odbcConnectExcel is only usable with 32-bit Windows
>
```

Importar datos de Excel

- Para importar directamente desde excel o exportar a excel en su formato .xls se deben cargar paquetes adicionales por ejemplo el paquete XLConnect

```
> library(XLConnect)
> vignette('XLConnect') # nos muestra el manual muy practico
> Libro <- loadWorkbook('C:/R/proyectos/UPG/Ejemplo2.xls', create = TRUE)
> Hoja <- readWorksheet(Libro, sheet = 'ejemplo1')
> Hoja
  nombre edad altura peso
1  Pedro   33    180   77
2 Paloma   43    170   67
3 Andres   35    179   80
4 Pepito   45    190   90
```

- Otra manera de hacerlo, aparte de XLConnect es con la función read.xls del paquete gdata pero tenemos que tener instalado perl porque sino nos da un error.

```
> library(gdata)
> excel2 <- read.xls('C:/R/proyectos/UPG/Ejemplo2.xls', sheet=1) # Much
o más fácil que el anterior.
Error in findPerl(verbose = verbose) :
  perl executable not found. Use perl= argument to specify the correct
path.
Error in file.exists(tfn) : invalid 'file' argument
```

Importar datos de Excel

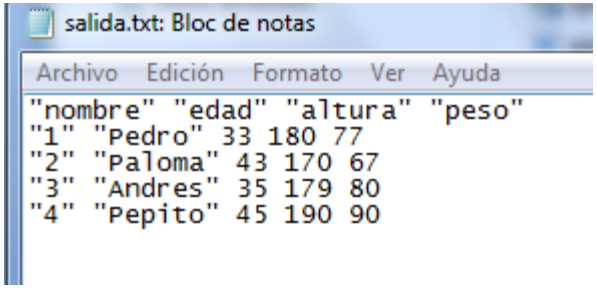
- Para importar directamente desde excel mas fácilmente se puede utilizar el paquete xlsx y utilizar la función read.xlsx()

```
> library(xlsx)
Loading required package: rJava
Loading required package: xlsxjars
> ficheroexcel <- "C:/R/proyectos/UPG/Ejemplo2.xls"
> mydataframe <- read.xlsx(ficheroexcel, 1)
> # La instrucción anterior de la primera hoja del fichero ficheroexcel
.xlsx
> mydataframe
  nombre edad altura peso
1  Pedro   33    180   77
2 Paloma   43    170   67
3 Andres   35    179   80
4 Pepito   45    190   90
```

Exportar datos: función write.table

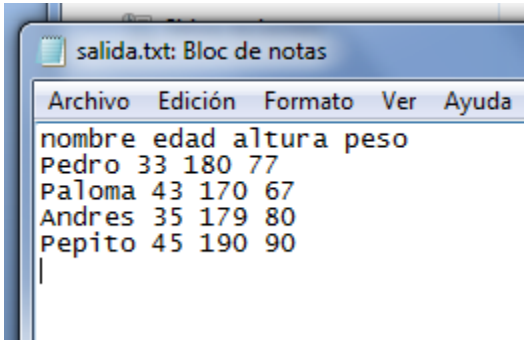
- La función write.table() exporta a un fichero de texto cualquier variable data.frame que haya en R-Console.
- Consideremos que existe la tabla mydataframe del punto anterior y queremos grabarla en un fichero que se llame “salida.txt”

```
>write.table(mydataframe,file="C:/R/proyectos/UPG/salida.txt")
```



Exportar datos: función write.table

- Incorporando más argumentos de entrada a la función write.table() podemos quitarle las comillas, respetar la cabecera y dejarlo en un formato como este:




```
>write.table(mydataframe,file="C:/R/proyectos/UPG/salida.txt",
+quote=FALSE,row.names=FALSE)
```

- El argumento 'quote' se refiere a las comillas dobles que se escriben en el fichero y el argumento 'row.name' hace referencia a la numeración de las filas de la tabla.
- Para una descripción detallada de los posibles argumentos de entrada para la función write.table utilícese la ayuda help(write.table)

Exportar datos a Excel con portapapeles

- La transferencia de datos desde un data.frame de R a una hoja de Excel, se completa con los dos pasos siguientes:
- Copiamos el contenido del data.frame de R al portapapeles, usando para ello la función

```
>write.table(mydataframe, "clipboard", sep="\t", row.names=F)
```

- Desde Excel obtenemos los datos sencillamente colocándonos en la celdilla de destino y pegando desde el portapapeles
- La ventaja de usar el portapapeles para intercambiar información entre Excel y R es que no es necesario usar archivos intermedios ni tampoco la instalación de paquetes adicionales en R.
- La principal desventaja estriba en que no es un medio cómodo cuando el volumen de los datos es considerable

Exportar datos a Excel

Con el paquete xlsx

- Para exportar directamente desde excel mas fácilmente se puede utilizar el paquete xlsx y utilizar la función read.xlsx()
- Este paquete también es capaz de trabajar con archivos Excel en los dos formatos habituales, .xls y .xlsx, ofreciendo funciones que permiten tanto escribir como leer datos de sus hojas. Las dos funciones fundamentales son write.xlsx() y read.xlsx().
- También están disponibles las funciones write.xlsx2() y read.xlsx2(), funcionalmente equivalentes a las anteriores pero que ofrecen un mejor rendimiento cuando se trabaja con hojas de cálculo muy grandes.

Obtener datos a partir de una URL

- Si los datos con los necesitamos trabajar están alojados en un sitio web, como puede ser un repositorio de **GitHub**, no es necesario que descarguemos el archivo a una ubicación local para a continuación leerlo. Gracias a la función **getURL()** del paquete **RCurl** podemos leer directamente desde la URL

```
> library('RCurl')
Loading required package: bitops
> url <- getURL(
+ 'https://raw.githubusercontent.com/fcharte/ExploraVisualizaconR/
+ master/data/results.csv', ssl.verifypeer = FALSE)
> results2 <- read.csv(textConnection(url))
> str(results2)
'data.frame': 0 obs. of  1 variable:
 $ X400..Invalid.request: logi
```

- Descarga la información especificada por el parámetro URL. Se aceptan muchos otros parámetros cuya finalidad es controlar la solicitud HTTP y la gestión de la respuesta.

Formatos .csv y .dif

El formato CSV (Comma_Separated Values)

- Utiliza comas para separar los valores y se estructura como una serie de filas (registros) con columnas de longitud variable. Los valores no numéricos pueden ir o no entrecomillados y en los numéricos el separador decimal es el punto. En algunos países europeos se usa la coma como separador decimal y el punto y coma para separar unos valores de otros. Es importante destacar que Excel utiliza el punto y coma como separador incluso cuando su configuración establece que el separador decimal es el punto.

El formato DIF

- Fue diseñado a principios de la década de los 80 del siglo pasado para facilitar el intercambio de datos entre hojas de cálculo, usándose por entonces en Visicalc y Lotus 1, 2, 3 y habiendo llegado hasta Excel y FreeOffice Calc. Se trata de un formato más descriptivo y, en consecuencia, menos compacto que CSV.