Tema 3. Transmisión de datos entre R y otros formatos

Álvaro Briz Redón

Índice

En esta unidad van a mostrarse diferentes maneras de intercambiar información entre R y los siguientes formatos: .txt, .csv y .xlsx. En el caso de los ficheros .txt y .csv se indicarán brevemente las funciones requeridas. En el caso de los ficheros Excel (.xlsx) se proporcionará un mayor número de opciones, facilitando la creación de un fichero Excel relativamente complejo desde R.

3.1. Ficheros de texto (.txt). Lectura y escritura

Un archivo de texto (.txt) estructurado en filas y columnas puede cargarse en R con la función **read.delim**. El código básico sería el siguiente:

```
fichero_texto=read.delim(file="FicheroTexto.txt",header=TRUE,sep=" ")
```

Se ha indicado también que se considere la primera fila del fichero como el encabezado (nombre de las variables), para lo que se incluye el parámetro **header = TRUE**. Este parámetro también forma parte de las funciones para la lectura de ficheros .csv y .xlsx que se indican en apartados sucesivos.

Además, se ha indicado **sep = “”** (valor por defecto para esta función), que representa el carácter (espacio en blanco) que separa las columnas del fichero texto.

En cuanto a ficheros de texto no estructurados en filas y columnas, la función **scan** es posiblemente la principal (dispone de múltiples parámetros, que no van a tratarse). Por ejemplo, indicando que se cargue todo como un tipo **character**:

```
fichero_texto=scan(file="FicheroTexto.txt",what="character")
```

La creación de ficheros .txt se puede realizar principalmente con **write** (texto no estructurado) y **write.table** (texto estructurado en filas y columnas). Para ambas funciones, el parámetro **x** debe indicar el objeto de R que desea guardarse como texto y **file** el nombre del fichero resultante. Por ejemplo:

```
texto="Este texto va a guardarse en un fichero .txt"
write(x=texto,file="ejemplo_texto.txt")
write.table(x=iris,file="iris_table.txt")
```

Se ha creado una variable de tipo cadena para ejemplificar el uso de **write**. Si se trata de utilizar **write** para guardar el conjunto de datos **iris**, el programa devuelve un error, ya que **write** no está diseñada para guardar datos de este tipo. En cambio, **write.table** sí es adecuada para guardar un conjunto de datos.

3.2. Ficheros separados por tabuladores (.csv). Lectura y escritura

Los archivos separados por tabuladores (.csv) son de tipo estructurado (filas y columnas), constando de un caracter separador de columnas. La lectura y escritura de estos ficheros en R se realiza mediante las funciones **read.csv** y **write.csv**. Por ejemplo, puede guardarse como .csv el conjunto de datos **iris** separando las columnas por el símbolo ";". Luego se puede cargar indicando el separador para garantizar la correcta lectura de datos.

```
write.csv(x=iris,file="iris_table.csv")
iris_csv=read.csv(file="iris_table.csv")
```

3.3. Ficheros de Microsoft Excel (.xlsx)

La librería **openxlsx** permite la lectura y escritura de documentos Excel desde la consola de R. Otros paquetes de R que incluyen funciones parecidas a las de **openxlsx** son **xlsx** y **XLConnect**. Para esta unidad se ha escogido **openxlsx** por no depender de librerías adicionales que en ocasiones pueden dificultar su instalación (relacionadas con el lenguaje Java).

En primer lugar es necesario instalar esta librería.

```
install.packages("openxlsx")
library(openxlsx)
```

3.3.1. Lectura y escritura

La función **read.xlsx** sirve para leer cualquier fichero Excel disponible en nuestro equipo. Para su correcto uso, es necesario especificar el nombre completo del fichero Excel, incluyendo su extensión, y la hoja (**sheetIndex**) del fichero que se quiere importar. Por ejemplo:

```
fichero_excel=read.xlsx(file="FicheroExcel.xlsx",sheetIndex=1,header=TRUE
)
```

Por su parte, la función **write.xlsx** es la que permite pasar un objeto R a un fichero de Excel. Por ejemplo, el conjunto **iris** puede guardarse como fichero de Excel, indicándose el nombre del conjunto de datos en el parámetro **x** y el nombre del fichero donde se quiere almacenar el resultado en **file**:

```
write.xlsx=write.xlsx(x=iris,file="iris.xlsx")
```

3.3.2. Creación completa de un fichero Excel desde R

En este apartado se van a dar ejemplos de edición de un fichero Excel desde R tomando como referencia la documentación que se ofrece del paquete **openxlsx** en https://rdrr.io/cran/openxlsx/man/conditionalFormatting.html.

En primer lugar es necesario crear un objeto que representa un nuevo libro de Excel "en blanco" mediante la función **createWorkbook**. La clase de este nuevo objeto es **Workbook**, como puede comprobarse. A continuación, se genera una pestaña de este nuevo libro (mediante la función **addWorksheet**) para ser posteriormente editada.

```
libro <- createWorkbook()
class(libro)

## [1] "Workbook"
## attr(,"package")
## [1] "openxlsx"

addWorksheet(wb=libro, sheetName="createStyle pruebas")
```

Por su parte, la función **writeData** permite escribir datos en el objeto **Workbook**. Hay que indicar el nombre del objeto creado en R que quiere transferirse al **Workbook**, la pestaña del mismo sobre la que quiere transferirse y la fila (**startRow**) y columna (**startCol**) sobre las que la información del objeto va a situarse (existen más parámetros de **writeData** que pueden consultarse desde el *Help*, pero estos son los principales).

En lo que queda del apartado 3.3.2. van a tratarse dos funciones fundamentales de **openxlsx** para la edición de ficheros Excel desde R: **createStyle** y **conditionalFormatting**

3.3.2.1. Opciones de *createStyle*

En la siguiente tabla se indican la mayoría de los parámetros de **createStyle** junto con los posibles valores que pueden tomar, proporcionando prácticamente las mismas posibilidades de formato de celda/texto disponibles en Excel. Para conocer el listado completo de valores asociados a cada parámetro, consúltese el *Help*.

Parámetro	Significado (opciones)
fontName	Nombre de fuente

fontSize	Tamaño de fuente
fontColour	Color de fuente
numFmt	Formato de celda (**"general"**, **"number"**,…)
border	Tipo de borde (**"top"**, **"bottom"**,…)
borderColour	Color de borde
borderStyle	Estilo de borde (**"thin"**, **"medium"**,…)
bgFill	Color de celda
fgFill	Color de trama
halign	Alineación horizontal (**"left"**,…)
valign	Alineación vertical (**"top"**,…)
textDecoration	Estilo fuente (**"bold"**, **"italic"**,…)
wrapText	Ajustar texto al tamaño de celda (si **TRUE**)
textRotation	Rotación del texto en grados
indent	Indentar texto horizontalmente (si **TRUE**)

Por ejemplo, en la siguiente línea de código se define un estilo, **pruebaStyle**, con letra *Times New Roman*, tamaño 12, todos los bordes marcados y alineación horizontal centrada. Los colores de letra, fondo y borde de celda no se modifican, por lo que aparecen con sus valores por defecto.

```
pruebaStyle <- createStyle(fontName = "Times New Roman", fontSize = 12,
border=c("top", "bottom", "left", "right"),halign="center")
```

Utilizando **writeData** puede transferirse el conjunto de datos **iris** sobre la pestaña **"createStyle pruebas"** de **libro** (los datos son pegados en la celda A1). Posteriormente, se utiliza la función **addStyle** para formatear las celdas siguiendo el estilo definido por **pruebaStyle**. Los parámetros **rows** y **cols** deben abarcar el total de filas y columnas de los datos **iris** (se incrementa el valor de **rows** en 1 por el encabezado de los datos). El parámetro **gridExpand** debe toma valor **TRUE** para que el formato pueda aplicarse sobre todas las combinaciones celdas de la pestaña. En otro caso, el programa devuelve un error.

```
writeData(wb=libro, sheet="createStyle pruebas", x=iris, startCol=1,
startRow=1)
addStyle(wb=libro, sheet="createStyle pruebas", rows=1:(nrow(iris)+1),
cols=1:ncol(iris), style = pruebaStyle, gridExpand=TRUE)
```

Las instrucciones anteriores modificaban el objeto **libro** de tipo **Workbook**, pero es necesario un último paso que permite que dicho objeto sea representado como un fichero Excel en nuestro equipo. La función **saveWorkbook** es la que realiza esta acción.

```
saveWorkbook(wb=libro, file = "ExcelPruebaDesdeR.xlsx", overwrite = TRUE)
```

3.3.2.1. Opciones de *conditionalFormatting*

En primer lugar, se crean tres estilos de celda que corresponderán en los sucesivos ejemplos de **conditionalFormatting** a celdas de valor negativo, positivo o duplicado. En este caso, solo se edita el color del texto (**fontColour**) y el color de fondo (**bgFill**):

```
negStyle <- createStyle(fontColour = "#9C0006", bgFill = "#FF6F61")
posStyle <- createStyle(fontColour = "#006100", bgFill = "#C6EFCE")
dupStyle <- createStyle(fontColour = "#0070C0", bgFill = "#A3D8FF")
```

En este caso, los colores se han indicado siguiendo la base hexadecimal. La web https://www.color-hex.com/ es muy útil para escoger un color, obtener su código hexadecimal y poder luego utilizarlo en R (siempre con # encabezando el códido de 6 símbolos). En caso de no elegirse un estilo personalizado, **conditionalFormatting** utiliza el establecido por defecto en Excel.

La función **conditionalFormatting** de **openxlsx** dispone de los siguientes parámetros:

Parámetro	Significado (opciones)
wb	Nombre del objeto **Workbook**
sheet	Nombre de una pestaña de **Workbook**
cols	Columnas a las que aplicar el formato condicional
rows	Filas a las que aplicar el formato condicional
rule	Regla que determina el formato condicional
style	Estilo para las celdas que verifican la regla (**rule**)
type	Tipo de regla: **“expression”**, **“databar”**, **“duplicates”** o**“contains”**

Una vez descritos los parámetros de **conditionalFormatting**, en lo que queda del apartado se dan varios ejemplos de uso, que serán aplicados a cada una de las siguientes pestañas.

```
libro <- createWorkbook()
addWorksheet(wb=libro, sheetName="cellIs")
addWorksheet(wb=libro, sheetName="Moving Row")
addWorksheet(wb=libro, sheetName="Moving Col")
addWorksheet(wb=libro, sheetName="Dependent on")
addWorksheet(wb=libro, sheetName="Duplicates")
addWorksheet(wb=libro, sheetName="containsText")
addWorksheet(wb=libro, sheetName="databar")
addWorksheet(wb=libro, sheetName="between")
addWorksheet(wb=libro, sheetName="logical operators")
```

En muchas de ellas, se incluirá inicialmente dos columnas de datos sobre las que poder aplicar el formato condicional: los números del -5 al 5 y las 11 primeras letras del abecedario (almacenadas en la constante de R **LETTERS**). Concretamente, este será el código que dará comienzo a estos ejemplos (otros incluirán datos similares):

```
writeData(wb=libro, "cellIs", -5:5, startCol=1)
writeData(wb=libro, "cellIs", LETTERS[1:11], startCol=2)
```

Introducir datos y colorear celdas en función de su valor

En este primer ejemplo, se aplica el estilo **posStyle** a las celdas de valor positivo y **negStyle** a las de valor negativo. Las celdas con valor 0 se quedan con su formato original.

```
writeData(wb=libro, "cellIs", -5:5, startCol=1)
writeData(wb=libro, "cellIs", LETTERS[1:11], startCol=2)
conditionalFormatting(wb=libro, "cellIs", cols=1, rows=1:11, rule="<0",
style = negStyle)
conditionalFormatting(wb=libro, "cellIs", cols=1, rows=1:11, rule=">0",
style = posStyle)
```

Introducir datos y colorear celdas en función del valor de una celda concreta

En el ejemplo anterior el formato condicional de cada celda se define a partir de su propio valor. Otra situación habitual es definir un formato condicional a partir del valor de otra celda. En este caso es necesario incluir en **rule** la referencia de la celda a partir de la cual se define el valor del resto. Por ejemplo, si se utiliza una única celda fijada de referencia (A1).

```
writeData(wb=libro, "Dependent on", -5:5, startCol=1)
writeData(wb=libro, "Dependent on", LETTERS[1:11], startCol=2)
conditionalFormatting(wb=libro, "Dependent on", cols=1:2, rows=1:11,
rule="$A$1<0", style = negStyle)
conditionalFormatting(wb=libro, "Dependent on", cols=1:2, rows=1:11,
rule="$A$1>0", style = posStyle)
```

Introducir datos y colorear celdas en función del valor variable de una fila o columna

Por otra parte, puede ser conveniente que la celda de referencia para definir el formato varíe en la fila o la columna. Para ello, es necesario usar el símbolo $ (como en el propio Excel) para hacer depender el formato de la fila ($A1) o de la columna (A$1). En primer lugar, dependiente de la fila:

```
writeData(wb=libro, "Moving Row", -5:5)
writeData(wb=libro, "Moving Row", LETTERS[1:11], startCol=2)
conditionalFormatting(wb=libro, "Moving Row", cols=1:2, rows=1:11,
rule="$A1<0", style = negStyle)
conditionalFormatting(wb=libro, "Moving Row", cols=1:2, rows=1:11,
rule="$A1>0", style = posStyle)
```

En segundo lugar, dependiente de la columna:

```
writeData(wb=libro, "Moving Col", -5:5)
writeData(wb=libro, "Moving Col", LETTERS[1:11], startCol=2)
conditionalFormatting(wb=libro, "Moving Col", cols=1:2, rows=1:11,
rule="A$1<0", style = negStyle)
```

```
conditionalFormatting(wb=libro, "Moving Col", cols=1:2, rows=1:11,
rule="A$1>0", style = posStyle)
```

Identificar valores duplicados

```
writeData(wb=libro, "Duplicates", sample(LETTERS[1:15], size = 10,
replace = TRUE))
conditionalFormatting(wb=libro, "Duplicates", cols = 1, rows = 1:10, type
= "duplicates", style = dupStyle)
```

Identificar si la celda contiene un cierto carácter (por ejemplo, la letra A)

Se define previamente una función (**fn**) que genera cadenas de letras aleatorias separadas por guiones.

```
fn <- function(x) paste(sample(LETTERS, 10), collapse = "-")
writeData(wb=libro, "containsText", sapply(1:10, fn))
conditionalFormatting(wb=libro, "containsText", cols = 1, rows = 1:10,
type = "contains", rule = "A")
```

Gráfico de barras coloreadas

```
writeData(wb=libro, "databar", -5:5)
conditionalFormatting(wb=libro, "databar", cols = 1, rows = 1:11, type =
"databar")
```

Identificar valores entre dos extremos

```
writeData(wb=libro, "between", -5:5)
conditionalFormatting(wb=libro, "between", cols = 1, rows = 1:11, type =
"between", rule = c(-2,2))
```

Identificar valores cumpliendo una condición lógica compleja

```
writeData(wb=libro, "logical operators", 1:10)
conditionalFormatting(wb=libro, "logical operators", cols = 1, rows =
1:10, rule = "OR($A1=1,$A1=3,$A1=5,$A1=7)")
```

Guardar el objeto generado como fichero Excel

Las instrucciones anteriores modificaban el objeto **libro** de tipo **Workbook**, pero es necesario un último paso que permite que dicho objeto sea representado como un fichero Excel en nuestro equipo. La función **saveWorkbook** es la que realiza esta acción.

```
saveWorkbook(wb=libro, file = "FormatoCondicionalDesdeR.xlsx", overwrite
= TRUE)
```

3.3.3. Rellenar un Excel a modo de plantilla desde R

Otra opción que puede resultar más conveniente en ocasiones es la de preparar un Excel (desde el propio programa de Office, manualmente) para luego introducir datos en él desde R. En este caso debería bastar con usar la función **writeData** adecuadamente. Si se

asume que el Excel de plantilla se denomina "PlantillaExcel.xlsx", el proceso de carga se haría mediante **loadWorkbook**:

```
libro_plantilla=loadWorkbook(file="PlantillaExcel.xlsx")
```

En este caso debe usarse **loadWorkbook** para cargar en R un libro previamente creado desde Excel, dando lugar a un objeto de clase **Workbook** (**libro_plantilla**). En el apartado anterior, en cambio, se creaba desde R un objeto **Workbook** vacío (con **createWorkbook**), al cual se le incorporaban pestañas y formatos para ser finalmente guardado como fichero Excel común (con **saveWorkbook**).