Práctica 1. Importación de Datos

Tratamiento de Datos, Grado en Ciencia de Datos - UV

XXXX

Contents

Introducción

El objetivo de esta práctica es realizar la importación de ficheros con diferentes formatos, organizar los datos en un **data frame**, y almacenarlo en un fichero tipo **RData**. Los ficheros a importar son los siguientes:

1. `DFP1_11122015 124700.csv ----------> Deposito.RData`
2. `"Transactions from a Bakery" (Kaggle) ----------> BreadBasket.RData`
3. `(SPSS) PsychBike.sav ----------> PsychBike.RData`
4. `FileCodificado.json ----------> FileCodificado.RData`
5. `ERCA.xls ----------> ERCA.RData`
6. `subjectInfo.xlsx ----------> Pacientes.RData`

Crea un proyecto

Para cada práctica crea un nuevo proyecto con:

- Una carpeta `./data` que contenga los datos.
- Opcionalmente, otra carpeta `./figure` para las figuras.
- Opcionalmente, otras para organizar la información, `./program`, etc.

Rutas locales

Usa siempre rutas referidas a la carpeta, en la que se encuentra el fichero fuente, y siempre rutas *relativas* a dicha ubicación. Ejemplos:

- `ruta1 <- 'data/tabla1.txt'`: Fichero en la carpeta datos del directorio donde está el código.
- `ruta2 <- '../data/tabla1.txt'`: Fichero en la carpeta datos que cuelga de un nivel superior.
- `ruta3 <- '../../data/tabla1.txt'`: Fichero en la carpeta datos que cuelga de dos niveles superiores.
- `ruta4 <-'C:/MisDatos/Ej.txt'`: **NO USAR NUNCA RUTAS ABSOLUTAS COMO ÉSTA**.

Ejemplo: fichero *DFP1_11122015 124700.csv*

Importación del fichero

1. Utiliza *Import Dataset* de RStudio para previsualizar el resultado final. Compara las opciones *base* y *readr*.
2. Importa el fichero con *base* y comprueba el resultado. Pon el parámetro `stringsAsFactors = FALSE`.
3. Convertir fecha y hora con `as.POSIXlt` y `as.hms` (`hms` library).
4. Almacenar en un nuevo fichero con `save`.

```
fname <- 'data/DFP1_11122015 124700.csv'
```

Inspección de los datos

Inspecciona los datos con `str`, `head`, `tail` y `summary`.

```
# str, head, tail, summary
```

Fichero *Transactions from a Bakery* (Kaggle)

Importación del fichero

Una vez hemos descargado el fichero podríamos leerlo con una opción de la librería `base` y transformar los datos a los tipos correctos.

Otra opción es usar la librería `readr` para especificar directamente el tipo de datos.

1. Utiliza *Import Dataset* de RStudio, opción *readr*, para previsualizar el fichero y especificar los tipos de datos.
2. Copia el código resultante y comprueba la lectura de datos.
3. Guarda el data frame con `save`.

```
library(readr)
filename <- "data/BreadBasket_DMS.csv"
```

Inspección de los datos

Inspecciona los datos con `str`, `head`, `tail` y `summary`.

```
# dim, str, ...
```

Fichero SPSS *PsychBike.sav*

Importación del fichero

Podemos leer ficheros de IBM SPSS mediante la librería `haven`.

1. Leer el fichero con `read_sav`.
2. Guardar el data frame con `save`.

```
library(haven)
filename <- "data/PsychBike.sav"
```

Inspección de los datos

Inspecciona los datos con `str`, `head`, `tail` y `summary`.

```
# dim, str, etc.
```

Fichero JSON *FileCodificado.json*

Importación del fichero

Los ficheros JSON (*JavaScript Object Notation*) son ficheros de texto que contienen datos (originalmente presentaciones de objetos JavaScript). Es un formato muy popular que se utiliza frecuentemente como alternativa a los ficheros XML, puesto que son más fáciles de leer e interpretar.

En R podemos importar ficheros JSON con la librería `jsonlite`. Las funciones más empleadas son:

- `read_json` y `write_json`: leen/escriben objetos R directamente de/en ficheros JSON.
- `toJSON`: obtiene una representación JSON de un objeto R.
- `fromJSON`: inversa de la anterior, convierte una representación JSON en un objeto R.

Importar el fichero `FileCodificado.json`

1. Abrir el fichero `FileCodificado.json` con un editor de texto (el de RStudio vale) para ver qué contiene. Comprobaremos que se puede leer e interepretar con facilidad.
2. Importar el fichero a R con `read_json`. Observar la importancia del parámetro `simplifyVector`.
3. Como alternativa, leer el fichero en modo texto con `read_file` de la librería **readr** y después convertir el objeto character a data frame con `fromJSON`.
4. Almacenar los resultados en `FileCodificado.RData`.

```
library(jsonlite)
filename <- 'data/FileCodificado.json'
```

Inspección de los datos

Inspecciona los datos con `str`, `head`, `tail` y `summary`.

```
# str, head, tail, summary
```

Extra: exportar datos en formato JSON

Es importante saber cómo exportar datos en este formato.

1. Almacenar el dataset `iris` en disco con `write_json`. Abrir el fichero con RStudio y comprobar el resultado.
2. Como alternativa, obtener la representación JSON de los datos con `toJSON` y luego almacenarlos con `write_file`. Observar la diferencia de usar o no el parámetro `pretty` en `toJSON`.

```
# ...
```

Fichero *ERCA.xlsx*

Importación del fichero

- Abrir el excel para ver qué pinta tiene.
- Como el fichero es muy irregular la función de importación `read_excel` no es capaz de detectar automáticamente los tipos de datos y lo lee todo como caracteres (strings).
- Fechas: las importa como un string que interpretado como 'numeric' es el número de días transcurridos desde 01-01-1900.
- Si se indica en `col_type` que son `date` los lee bien.
- Formatos numéricos: se puede indicar el tipo en `col_type`, o convertirlos después con `as.numeric`.
- Guardar datos en formato .RData.

```
library(readxl)
fname <- "data/ERCA.xlsx"

# Usaremos la función read_excel de readxl
df <- read_excel(fname, sheet=2) # El formato de las fechas no es correcto

# Columnas date: FECHA*
# Columnas numeric: COL*, TG, HIDROF, ANTI*, DIURET, IECAS, PESO, TALLA, HB,
#                   IST, FERRITINA, UREA, FOSFORO, CALCIO, CICALCET, QUELANTES,
#                   PTH, BICARBO, ALOPURIN, URICO, PROTEINU*, ALBUMI*, sodio,
#                   potasio, cloro, HB GLIC, GLUCOSA, PCR
# Es más fácil ver cuales NO son 'numeric'.
# Definir los tipos de datos en 'ct' y releer con 'read_excel'.
# df <- read_excel(fname, sheet=2, col_types=ct)
```

```
# save()
```

Inspección de los datos

- Utiliza las funciones `head`, `tail`, `str` y `summary` para inspeccionar el fichero.
- Limpia el data frame descartando las filas con `NA` en la primera columna (`IDENTIFICA`).

```
# Conviene limpiar primero un poco => eliminar is.na, etc.
# head, tail, str, summary
```

Fichero *subjectInfo.xlsx*

Importación del fichero

```
filename <- "data/subjectInfo.xlsx"
```

1. Leer cada hoja (hasta 4) en data frames separados (`Hoja1`, `Hoja2`, ...) usando `read_excel` y el parámetro `sheet`.
2. Leer todas las hojas automáticamente usando:

- Usando `excel_sheets`: obtén los nombres de las hojas del fichero excel y guardalos en la variable `sheets`.
- Usando un bucle `for` sobre los nombres devueltos por `excel_sheets`. Guardamos sobre una lista creada con `df <- list()` y añadiendo nuevos elementos con `df[[n]]`, donde `n` es una variable numérica entera o carácter (ojo al doble corchete).

3. En lugar de un bucle for usa `lapply`
   - Recuerda que `lapply` aplica una función a todos los elementos de un vector o lista y devuelve **una lista** con los resultados obtenidos.
   - Hay que definir una función que llame a `read_excel` y que tome como parámetro la hoja a leer.
   - La lista de devuelta por `lapply` no tiene nombres. Podemos nombrar esta lista con `names(x) <- sheets`, donde `x` es la lista obtenida con `lapply`, y `sheets` los nombres de las hojas obtenidas con excel_sheets‘.
4. Como siempre al final guardamos el resultado con `save`.

```
# save()
```

Inspección de los datos

Inspecciona los datos con `str` (jugar con `max.level`), `head`, `tail` y `summary`.

```
# str, head, tail, summary
```