Data Frames

Oscar Gerardo Hernández Martínez

12/8/2019

Tipos de datos que existen

Datos estructurados

En este tipo de archivo los datos se introducen en campos específicos que contienen cadenas de texto o bases numéricas. Cada uno de estos campos suele tener un espacio máximo definido. Los datos dentro de este tipo de archivo tienen un acceso rígido y limitado.

Datos semi-estructurados

Son ficheros con cierta estructura pero no tienen por qué tener una organización en un modelo racional (tabla o grafo). Este formato es más flexible al momento de expresar los datos y su esctructura garantiza cierto orden al momento de trabajar con los datos. Los tipos de archivos más comunes de este tipo son *xml* y *json*.

Datos no estructurados

Esto cuando la fuente de texto proviene de un archivo de texto o un archivo binario, es decir, no tiene una estructura claramente definida. Los tipos de archivos de este tipo más comunes son de tipo *csv* y *txt*. Este tipo de archivos son muy flexibles, pero suelen ser problemáticos al momento de querer trabajar con ellos.

¿Qué es un Data Frame?

Un data frame es una tabla de doble entrada, formada por variables en las columnas y observaciones de estas variables en las filas, de manera que cada fila contiene los valores de las variables para un mismo caso o un mismo individuo.

- **data():** para abrir una ventana con la lista de los objetos de datos a los que tenemos acceso en la sesión actual de `R` (los que lleva la instalación básica de `R` y los que aportan los paquetes que tengamos cargados).
  - Si entramos **data(package=.packages(all.available = TRUE))** obtendremos la lista de todos los objetos de datos a los que tenemos acceso, incluyendo los de los paquetes que tengamos instalados, pero que no estén cargados en la sesión actual.

Obteniendo información de un Data Frame

- **head(d.f,n):** Para mostrar las n primeras filas del data frame. Por defecto se muestran las 6 primeras filas.
- **tail(d.f,n):** Para mostrar las n últimas filas del data frame. Por defecto semuestran las 6 últimas.
- **str(d.f):** Para conocer la estructura global de un data frame.
- **names(d.f):** Para producir un vector con los nombres de las columnas.
- **rownames(d.f):** Para producir un vector con los identificadores de las filas.
  - `R` entiende siempre que estos identificadores son palabras, aunque sean números, de ahí que los imprima entre comillas.
- **colnames(d.f):** Para producir un vector con los identificadores de las columnas.
- **dimnames(d.f):** Para producir una list formada por dos vectores (el de los identificadores de las filas y el de los nombres de las columnas).
- **nrow(d.f):** Para consultar el número de filas de un data frame.
- **ncol(d.f):** Para consultar el número de columnas de un data frame.
- **dim(d.f):** Para producir un vector con el número de filas y el de columnas.

- **d.f$nombre_variable:** Para obtener una columna concreta de un dataframe.
  - El resultado será un vector o un factor, según cómo esté definida la columna dentro del data frame.
  - Las variables de un data frame son internas, no están definidas en el entorno global de trabajo de R.
- **d.f[n,m]:** Para extraer "trozos" del data frame por filas y columnas (funciona exactamente igual que en matrices) donde n y m pueden definirse como:
  - intervalos
  - condiciones
  - números naturales
  - no poner nada
  - Si sólo queremos definir la subtabla quedándonos con algunas variables, basta aplicar el nombre del data frame al vector de variables.
  - Estas construcciones se pueden usar también para reordenar las filas o columnas.

Leyendo tablas de datos:

- **read.table():** Para definir una data frame a partir de una tabla de datos contenida en un fichero.
  - Este fichero puede estar guardado en nuestro ordenador o bien podemos conocer su URL. Sea cual sea el caso, se aplica la función al nombre del fichero o a la dirección entre comillas.

Algunas listas para practicar<—- Pica en el texto para acceder.

Parámetros de read.table()

- **header = TRUE:** Para indicar si la tabla que importamos tiene una primera fila con los nombres de las columnas. El valor por defecto es FALSE.
- **col.names = c(. . .):** Para especificar el nombre de las columnas. No olvidar que cada nombre debe ir entre comillas.
- **sep:** Para especificar las separaciones entre columnas en el fichero (si no es un espacio en blanco). Si es así, hay que introducir el parámetro pertinente entre comillas.
- **dec:** Para especificar el signo que separa la parte entera de la decimal (si no es un punto. Si es así, hay que introducir el parámetro pertinente entre comillas.
- **stringsAsFactors:** Para prohibir la transformación de las columnas de palabras en factores debemos usar stringsAsFactors=FALSE (ya que por defecto, R realiza dicha transformación).
- Para importar un fichero de una página web segura (cuyo URL empiece con https), no podemos ingresar directamente la dirección en **read.table()**; una solución es instalar y cargar el paquete RCurl y entonces usar la instrucción **read.table (textConnection(getURL("url")),. . .)**.

Leyendo diferentes tipos de fichero

- **read.csv():** Para importar ficheros en formato CSV.
- **read.xls()** o **read.xlsx():** Para importar hojas de cálculo tipo Excel u OpenOffice en formato XLS o XLSX, respectivamente. Se necesita el paquete xlsx.
- **read.mtb():** para importar tablas de datos Minitab. Se necesita el paquete foreign.
- **read.spss():** para importar tablas de datos SPSS. Se necesita el paquete foreign.

  Para más tipos de read, se puede ver en la consola de R con la instrucción help.search("read")

Exportando datos a ficheros

- **write.table(df, file = ""):** Para exportar un data frame a un fichero
  - *file = "":* Es donde indicaremos el nombre que queremos darle al fichero.
  - Podemos usar el parámetro *sep* para indicar el símbolo de separación de columnas. Siempre entre comillas.
  - También podemos utilizar el parámetro *dec* para indicar la separación entre la parte entera y decimal de los datos.

Construyendo Data Frames

- **data.frame(vector_1,. . . ,vector_n):** Para construir un data frame a partir de vectores introducidos en el orden en el que queremos disponer las columnas de la tabla.
  - R considera del mismo tipo de datos todas las entradas de una columna de un data frame.
  - Las variables tomarán los nombres de los vectores. Estos nombres se pueden especificar en el argumento de **data.frame** entrando una construcción de la forma **nombre_variable = vector**.
  - **rownames:** para especificar los identificadores de las filas. También en esta función podemos hacer uso del parámetro **stringsAsFactors** para evitar la transformación de las columnas de tipo palabra en factores.
- **fix(d.f):** Para crear / editar un data frame con el editor de datos.
- **names(d.f):** Para cambiar los nombres de las variables.
- **rownames(d.f):** Para modificar los identificadores de las filas. Han de ser todos diferentes.
- **dimnames(d.f)=list(vec_nom_fil, vec_nom_col):** Para modificar el nombre de las filas y de las columnas simultáneamente.
- **d.f[núm_fila,] = c(. . .):** Para añadir una fila a un data frame.
  - Las filas que añadimos de esta manera son vectores, y por tanto sus entradas han de ser todas del mismo tipo.
  - Si no añadimos las filas inmediatamente siguientes a la última fila del data frame, los valores entre su última fila y las que añadimos quedarán no definidos y aparecerán como NA.
  - Para evitar el problema anterior, vale más usar la función **rbind()** para concatenar el data frame con la nueva fila.
- **d.f$new_var:** Para añadir una nueva variable al data frame.
  - Podemos concatenar columnas con un data frame existente mediante la función **cbind()**. De este modo se puede añadir la columna directamente sin necesidad de convertirla antes a data frame.
  - Esta nueva variable ha de tener la misma longitud que el resto de columnas del data frame original. Si no, se añadirán valores NA a las variables del data frame original o a la nueva variable hasta completar la misma longitud.

  El comando **View()** nos mostrará el Data Frame en la pantalla de la consola de R.

Cambiando los tipos de datos (Casting)

- **as.character:** Para transformar todos los datos de un objeto en palabras.
- **as.integer:** Para transformar todos los datos de un objeto a números enteros.
- **as.numeric:** Para transformar todos los datos de un objeto a números reales.

Sub-Data Frames

- **droplevels(d.f):** Para borrar los niveles sobrantes de todos los factores, ya que las columnas que son factores heredan en los sub-data frames todos los niveles del factor original, aunque no aparezcan en el trozo que hemos extraído.
- **select(d.f, parámetros):** Para especificar qué queremos extraer de un data frame.
- **starts_with("x"):** Extrae del data frame las variables cuyo nombre empieza con la palabra "x".
- **ends_with("x"):** Extrae del data frame las variables cuyo nombre termina con la palabra "x".
- **contains("x"):** Extrae del data frame las variables cuyo nombre contiene la palabra "x". Se necesita el paquete dplyr o mejor aún tidyverse.
- **subset(d.f,condición,select = columnas):** Para extraer del data frame las filas que cumplen la condición y las columnas especificadas.
  - Si queremos todas las filas, no hay que especificar ninguna condición.

  - Si queremos todas las columnas, no hace especificar el parámetro select.
  - Las variables en la condición se especifican con su nombre, sin añadir antes el nombre del data frame.

Aplicando funciones a Data Frames

- **sapply(d.f, función):** Para aplicar una función a todas las columnas de un data frame en un solo paso.
  - **na.rm=TRUE:** Para evitar que el valor que devuelva la función para las columnas que contengan algún NA sea NA.
- **aggregate(variables~factors,data=d.f,FUN=función):** Para aplicar una función a variables de un data frame clasificadas por los niveles de un, o más de un, factor.
  - Si queremos aplicar la función a más de una variable, tenemos que agruparlas con un **cbind**.
  - Si queremos separar las variables mediante más de un factor, tenemos que agruparlos con signos +

Variables Globales

- **attach(d.f):** Para hacer que R entienda sus variables como globales y que las podamos usar por su nombre, sin necesidad de añadir delante el nombre del data frame y el símbolo $.
  - Si ya hubiera existido una variable definida con el mismo nombre que una variable del data frame al que aplicamos attach, hubiéramos obtenido un mensaje de error al ejecutar esta función y no se hubiera reescrito la variable global original.
- **detach(d.f):** Para devolver la situación original, eliminando del entorno global las variables del data frame.