Preprocesamiento de datos

Minería de datos

José Tomás Palma Méndez

Dept. of Information and Communication Engineering. University of Murcia
Contacting author: jtpalma@um.es

1. Introducción

En esta sesión vamos a ver los aspectos básicos y herramientas que nos va a permitir desarrollar téccnias de preprocesamiento de datos en R.

Paquetes necesarios: `foreign`, `car`, `Hmisc`, `VIM` y `mice`.

2. Valores especiales

Al igual que en la mayoría de los lenguajes de programación en R también podemos encontrar valores especiales que indian alguna excepción respecto a los valores normales. Estos valores son: NA, NULL, ±InF y NaN.

- **NA** (Not available). Esté valor se utiliza para indicar que el dato en cuestión no está disponible. Todas las operaciones básicas en R pueden manipular valores NA sin provocar errores y la mayoría devolverán valores NA si uno de los argumentos es NA. Para detectar valores NA se puede utilizar la función is.na().
- **NULL**. Se puede considerar como el conjunto vacío. NULL no tiene asociada ninguna clases y tiene longitud 0 y, por lo tanto, no ocupa espacio en el vector. Para detectar valores NULL se puede utilizar la función is.null().
- **Inf**. Este valor se utiliza para representar el *infinito* y solo esta asociado a variables de tipo numérico. Técnicamente, Inf es un valor numérico válido que es el resultado de operaciones como una división por cero. Por lo tanto, operaciones y comparaciones entre Inf y otros valores numéricos son perfectamente válidas. Las funciones is.infinite() y is.finite() nos pueden ayudar a detectar este tipo de valores.
- **NaN**. Se utiliza para representar valores desconocidos como resultado de alguna operación (0/0, Inf-Inf y Inf/Inf), pero de los que se tiene la seguridad de que no son números. La operaciones que implican un NaN dan como resultado un NaN. La función is.NaN() se puede usar para detectar NaN.

Ejercicio 1. Teniendo en cuenta las anteriores definiciones, intenta descubrir el resultado de las siguientes operaciones y porqué se produce dicho resultado.

Operación	Resultado	Explicación
NA + 1		
sum(c(NA,1,2))		
median(c(NA,1,2,3), na.rm = TRUE)		
length(c(NA,2,3,4))		
3 == NA		
NA == NA		
NA & FALSE		
TRUE | NA		
length(c(2,3,NULL,4))		
sum(c(2,3,NULL,4))		
x <- NULL		
NaN == NULL		
NULL == NULL		
c <- 2		
pi/0		
2 * Inf		
Inf - 10e+10		
Inf + Inf		
exp(-Inf)		
3 <-Inf		
exp(NaN)		
NaN + 1		
Inf - 10e+10		
Inf + Inf		
3 <-Inf		

Consejo 1. Ademas de las funciones is. que hemos visto, R nos ofrece otras muchas funciones para identificar el tipo de una variable: is.numeric(), is.character(), is.complex(), is.data.frame(), etc. Consulta la ayuda de R y examina todas las funciones de este tipo.

3. Lectura de datos

R es capaz de leer y procesar datos en multitud de formatos. Aquí nos vamos a centrar en datos en forma de tabla y en formato texto (la mayoría de las aplicaciones pueden exportar a este formato). A groso modo podemos decir que un dataset es técnicamente correcto cuando:

- está almacenado en un data.frame con los nombres adecuados para las columnas (atributos), y
- cada columna es del tipo de dato acorde con el domino de la variable que representa. Por ejemplo, los datos categóricos deben estar definidos como factor y los números como numeric o intenger.

Consejo 2. En tareas de clasificación, hay que asegurarse de que la variable que indica la clase esté definida como factor. Esto se debe a que hay casos en que las clases se etiquetan con números (0, 1, ..). Al importar esta información, R entiende que la variable es numérica, y cuando intentemos crear modelos de clasificación, puede que R genere modelos de regresión.

Ejercicio 2. Abre con algún editor el fichero hepatitis.csv que podrás encontrar en la carpeta **recursos** del aula virtual.

2.a) ¿Cuáles crees que deben ser los tipos asociados a cada columna?
2.b) ¿Existen valores desconocidos? ¿Cómo están representados?
2.c) ¿Qué información crees que falta?

Para leer datos desde R tenemos las siguientes funciones

read.delim() read.delim2()
read.csv() read.csv2()
read.table() read.fwf()

Todas estas funciones devuelven un objeto de tipo data.frame. Si los nombres de las columnas se encuentran en el fichero pueden ser asignados de forma automática como nombres de los atributos.

Todas estas funciones llaman a la función read.table fijando algunos parámetros para permitir una fácil importación de datos. Concretamente

read.csv	para valores separados por comas con el punto para separar decimales.
read.csv2	para valores separados por puntos y comas con la coma para separar decimales.
read.delim	para valores separados por tabuladores con el punto para separar decimales.
read.delim2	para valores separados por tabuladores con la coma para separar decimales.
read.fwf	para columnas con un tamaño predeterminado.

Todas estas funciones aceptan entre otros los siguientes parámetros:

Parámetro	Descripción
header	¿La primera línea contiene los nombres de las columnas?
col.names	Array de strings con los nombres de las columnas.
na.string	Array para indicar cómo se representan los valores ausentes.
colClasess	Array que indica el tipo de cada columna.
stringsAsFactors	Si es TRUE indica que las columnas con strings se convierten a factors
sep	Separador de columnas (solo para read.fwf)

Todos estos parámetros se pueden configurar desde la interfaz que ofrece RStudio (ver Figura 1).

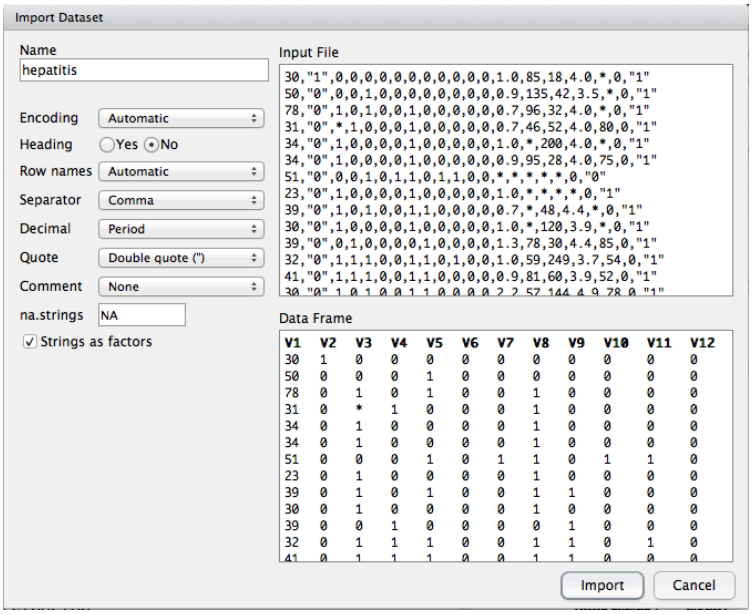

Figura 1. Interfaz para importación de datos en RStudio.

Consejo 3. Cuando importes unos datos con algunas de estas funciones, examínalos primero con funciones como head(), str() y summary().

Consejo 4. Puedes utiliar el paquete foreign para obtener más funciones de importación de datos desde otros plataformas como Weka, SPSS y SysStat.

Ejercicio 3. Importa el fichero hepatitis.csv desde RStudio, sin modificar los parámetros.

3.a) Ejecuta el comando str(hepatitis) y copia el resultado ¿Qué anomalías encuentras? Enuméralas.

3.b) La función complete.cases nos indica el número de filas completas que hay en el data.frame. Ejecútala sobre el conjunto de datos importados ¿Es el resultado esperado?

3.c) ¿Cómo podríamos con una sólo línea de código sustituir todos los valores que indican un dato no presente por un NA?

3.d) Prueba a volver a cargar los datos desde RStudio utilizando correctamente le parámetro na.strings. Una vez importados los datos ejecuta el comando str y compara los resultados con el caso anterior ¿Ves alguna anomalía?

3.e) ¿Cómo podríamos calcular en una sólo línea de código el número de filas que tiene algún valor desconocido?

En el Ejercicio 3 hemos aprendido a identificar de forma correcta los valores ausentes al importar datos en R. El siguiente paso consiste en nombrar de forma correcta las columnas. Este paso nos lo podemos saltar en el caso que el conjunto de datos incluya dichos nombres en la primera línea y colocando el parámetro header a TRUE al llamar a la función read.csv. En nuestro caso, el nombre de cada una de las columnas aparece como información adicional y deberían nombrarse como: EDAD, SEXO, ESTEROIDES, ANTIVIRALES, FATIGA, MALAISE, ANOREXIA, HIGgrande, HIGfirme, BAZOpalpa, ARANIASvac., ASCITIS, VARICES, BILIRRUBINA, FOSFATOalc, SGOT, ALBUMINA, PROTIME, HISTIOLOGIA y PRONOSTICO. Para que las columnas de nuestro conjunto de datos adopten estos nombres debemos hacer:

```
> atributos <- c("EDAD", "SEXO", "ESTEROIDES", "ANTIVIRALES",
+                "FATIGA", "MALAISE", "ANOREXIA", "HIGgrande,
+                "HIGfirme", "BAZOpalpa", "ARANIASvac", "ASCITIS",
+                "VARICES", "BILIRRUBINA", "FOSFATOalc", "SGOT",
+                "ALBUMINA", "PROTIME", "HISTIOLOGIA", "PRONOSTICO")
> colnames(hepatitis) <- atributos
```

4. Conversion de tipos

Una vez interpretados de forma correcta los valores ausentes y haberle dado nombre a las columnas. El siguiente paso en el procesamiento de datos consiste en definir los tipos de datos adecuados para cada columna. Esto lo podemos hacer mendiante las funciones de tipo *as.* de R. Por ejemplo, para convertir una variable cualquiera, var, a variable real, sólo deberíamos ejecutar el siguiente comando:

```
> var <- as.numeric(var)
```

En nuestro caso, el atributo ANTIVIRALES es una variable booleana y la conversión la podríamos hacer con la siguiente instrucción:

```
> hepatitis$ANTIVIRALES <- as.logical(hepatitis$ANTIVIRALES)
```

Puedes comprobar con la función str() el efecto que ha producido la instrucción anterior.

Ejercicio 4. Tenemos que asignar a las columnas BILIRRUBINA y PROTIME los tipos real (numeric) y entero (integer).

4.a) Si realizamos dicha conversión directamente con las funciones as.numeric() y as.intenger() ¿Se obtienen los resultados deseados? Para realizar dicha comprobación utiliza el fichero original.

4.b) En caso de que no se obtengan los resultados deseados ¿A qué crees que es debido?. ¿Cuál hubiera sido la forma correcta de hacerlo?

4.c) Realiza las operaciones necesarias para asignar los tipos adecuados según la siguiente información:

Tipo	Atributos
integer	EDAD, SGOT y FOSFATO-ALC
numeric	ALBUMINA
logical	ESTEROIDES, FATIGA-VARICES e HISTIOLOGIA

Comprueba que el resultado final es el esperado.

Llegados a este punto, nos queda por tratar de forma correcta los atributos originales. En este caso obersevese que el atributo PRONOSTICO está definido como entero y, por lo tanto, si se intenta aplicar técnicas de clasificación, o bien no se podrán realizar o se construirá un modelo de regresión al estar definido dicho atributo como entero. En este caso bastaría con utilizar la función as.factor() sobre dicha columna.

Sin embargo, en estos casos sería más interesante que los elementos de la enumeración fuesen más informativos. En nuestro caso, sería deseable cambiar el 0 y el 1 por FALLECE Y VIVE, respectivamente. Esta recodificación de etiquetas se puede hacer utilizando la función recode() del paquete car, de la siguiente forma:

```
> library(car)
> hepatitis$PRONOSTICO <- as.factor(recode(hepatitis$PRONOSTICO,
+                                    "0 = 'FALLECE'; 1 = 'VIVE'"))
```

La función **recode()** también permite asignar una misma etiqueta a varios valores. Por ejemplo, la siguiente instrucción

recode(outcome, "c(1:3) = 'NORMAL ; 0 = 'ANORMAL"')

recodificaría los valores del 1 al 3 con la etiqueta NORMAL y el valor 0 con la de ANORMAL.

Ejercicio 5. Transforma el atributo SEXO en un factor y asigna al 1 la etiqueta MASCULINO y al 0 la etiqueta FEMENINO.

5. Imputación

Como ya se ha visto en teoría el proceso de imputación consiste en la estimación de posibles valores para aquello campos con valores ausentes. Existen muchos métodos de imputación y ninguno destaca sobre los demás. El método elegido dependerá de

muchos factores como el tipo de datos (no todos los métodos de imputación admiten valores booleanos o factores), el tipo de probelma, rango de la variable, etc.

En R existen multitud de paquetes que ofrecen funciones para implementar distintas técnicas de imputación, aquí vamos a analizar dos de ellos los paquetes Hmisc y mice.

El paquete Hmisc nos ofrece la función impute() que tiene los siguientes parámetros:

Parámetro	Descripción
x	Array con valores NA
fun	La función de imputación {random, median, mean}

Por defecto se utiliza para imputar la mediana, aunque se puede suministrar un vector que contenga los valores a utlizar para la imputación. En el caso de vectores de tipo factor se utiliza el valor más frecuente. Una ventaja de esta función es que marca los valores imputados, de tal forma que con la fucnión is.impute() podemos saber si el valor ha sido imputado o no. Para imputar los valores ausentes en el atributo ALBUMINA hay que realizar las siguientes opreaciones:

```
> library(Hmisc)
> hepatitis.imp1 <- hepatitis
> hepatitis.imp1$ALBUMINA <- impute(hepatitis.imp1$ALBUMINA,fun=mean))
> hepatitis.imp1$ALBUMINA
```

Observa que los valores imputados van marcados con un asterisco. El paquete VIM nos ofrece, aparte de algunos métodos de imputación, funciones para visualizar y analizar la incidencia de los valores ausentes en un conjunto de datos. Para poder realizar una imputación utilizando el método KNN podemos utilizar la función kNN(). Este método intentan inferir un nuevo valor utilizando la información de los k elementos más cercanos (por defcto $k = 5$). Por defecto, para variables númericas se utiliza la mediana de los vecinos más próximos, mientras que para variables categórticas se utiliza el valor más frecuente. Para una imputación con esté método sólo debemos hacer:

```
> library(VIM)
> hepatitis.knnImp <- kNN(hepatitis)
```

La librería misc nos ofrece la función mice(), que nos permite imputar los valores ausentes de todo el conjunto de datos al mismo tiempo, considerando, además de los valores de la propia columna, tiene en cuenta los valores del resto de columnas. Esta función, nos ofrece multitud de métodos los cuales dependen del tipo de variables en el conjunto de datos (para ver la lista de métodos disponibles se puede ejecutar methods(mice). Como ejemplo, para aplicar el método Predictive mean matching ejecutaríamos las siguientes instrucciones:

```
> library(mice)
> tempImp.mice <- mice(hepatitis, m=5, method = "pmm", seed = 500)
> summary(tempImp.mice)
```

Si quisiéramos ver los valores que se han utilizado para la imputación en la columna ALBUMINA deberíamos ejecutar la siguiente instrucción:

```
>tempImp.mice$imp$ALBUMINA
```

Esto nos muestra que valor ha sido imputado en cada fila, para cada una de los 5 conjuntos de datos generados (parámetro m=5). Para hacer efectiva la imputación en el conjutno de datos original hay que utilizar la función complete():

```
> hepatitis.imp2 <- complete(tempImp.mice, 1)
```

Ejercicio 6. Partiendo del conjunto de datos obtenido una vez recodificado el atributo de clase (resultado del Ejercicio5):

- 6.a) ¿Cómo imprimirías por pantalla el porcentaje de (NA) de cada columna? Una vez realizada esta operación, genera algunas gráficas con las funciones del paquete VIM que permitan visualizar la distribución de los valores ausentes.
- 6.b) Utiliza la función impute() para imputar los valores ausentes en los atributos SGOT y FOSFATOalc. Utiliza los métodos median y mean y analiza los resultados ¿Qué cambios aprecias en el conjunto de datos?
- 6.c) Compara los resultados con los que se obtendrían con la función mice() y el método pmm y la imputación con la función kNN.
- 6.d) Genera cuatro ficheros con los resultados obtenidos con los siguientes nombres hepatitis.medianImpute.csv, hepatitis.meanImpute.csv, hepatitis.pmmImpute.csv y hepatitis.kNNImpute.csv.