Funciones para crear vectores de tipo *crudo* o para verificar si un vector es de tipo *crudo*

1 Descripción

La función `raw()` crea un vector de tipo "`raw`" o crudo, que almacena *bytes* sin codificar representados por valores hexadecimales. Para más información sobre el almacenamiento de datos crudos véase la sección *Detalles*, y para una introducción al formato hexadecimal véase la sección *Formato binario y hexadecimal*, ambas en esta página de ayuda.

La función `as.raw()` constriñe hacia el tipo crudo cualquier vector. Por su parte, la función `is.raw()` verifica si un objeto es de tipo crudo.

Índice

2 Forma de uso o sintaxis

```
raw(length = 0)
as.raw(x)
is.raw(x)
```

3 Argumentos

Tabla 1. Argumentos de las funciones de creación de vectores crudos

Argumento	Valor esperado	Descripción
`extensión`		
`length`	Un valor entero.	Extensión deseada del vector, es decir, el número de elementos que almacenará. Cada elemento representará a un *byte*.
`objeto`		
`x`	Un objeto.	Un objeto para ser constreñido como vector de tipo crudo.

4 Detalles

El tipo de almacenamiento "raw" consiste en un formato específico de R para contener *bytes* binarios, es decir, *bytes* crudos o sin codificar, dentro de vectores del mismo tipo. En este formato, cada *byte* almacenado es representado como un par de números en notación hexadecimal. Los elementos de un vector crudo consistirán, entonces, en pares de dígitos hexadecimales. De esta manera, un dígito individual representará medio octeto, y cada par de dígitos representará un octeto completo. Para una breve explicación sobre la relación entre datos crudos y hexadecimales véase primero la sección *Formato binario y hexadecimal* en esta página de ayuda. Los vectores crudos son un tipo de vector atómico o elemental de R y también son un tipo de objeto básico del lenguaje.

La extracción o reemplazo de elementos individuales de un vector crudo será posible mediante indización simple. Sin embargo, la operación de remplazo o asignación de valores solo podrá realizarse con valores de entrada que sean de tipo crudo o que hayan sido constreñidos a este tipo. Por ejemplo, con elementos que provengan de otros vectores de tipo crudo. Para los vectores crudos, los operadores relacionales funcionarán con el mismo método que los operadores lógicos, que consiste en interpretar la operación *bit* a *bit*. Para mayor información, véase la página de ayuda del tema *Comparaciones*, que detalla cómo los operadores relacionales interpretarán el orden numérico de la representación de *bytes*.

Los elementos de un vector crudo serán, entonces, pares de dígitos hexadecimales que representan, a su vez, *bytes* individuales. El despliegue en pantalla de los vectores crudos mostrará a cada *byte* separado uno del otro, en caso de que el vector contenga más de un elemento. Utilice la función `rawToChar()` si desea ver la representación de caracteres de cada *byte*. En caso de que los códigos hexadecimales correspondan a caracteres no imprimibles se desplegarán también las secuencias de escape.

La coerción al tipo crudo tratará a los valores de entrada como si representaran números naturales (en notación decimal) menores a 256, por lo que el valor de entrada se convertirá primero en un número entero. Si los valores de entrada están fuera del rango inclusivo entre 0 y 255, o son valores no disponibles (`NA`), su valor se establecerá con el valor cero (`00`), conocido como el *byte* nulo.

Las funciones `as.raw()` e `is.raw()` son funciones primitivas, por lo que su código fuente y especificación formal es ligeramente diferente al resto de funciones de R.

5 Valor devuelto

La función `raw()` devolverá un vector crudo con el número de elementos especificado en el argumento de extensión `length`. Al momento de su creación, cada elemento del vector será igual a cero (`00`). Luego, los vectores crudos podrán ser usados para almacenar secuencias de *bytes* con un número de dígitos fijo. Esto es, *bytes* cuyo número de dígitos tendrá siempre dos valores hexadecimales. Para más información, véase la sección *Formato binario y hexadecimal*.

La función `as.raw()` intentará constreñir al objeto del argumento `x` para convertirlo al tipo crudo y devolverlo como un vector de este tipo. Los valores originales que estén fuera del rango válido de conversión (de 0 a 255, véase la sección *Detalles*) serán transformados a cero (`00`) en el orden de los elementos. En cualquier otro caso se considerará que el constreñimiento fue exitoso.

La función `is.raw()` devolverá el valor lógico verdadero (`TRUE`) si y solo si la prueba lógica `typeof(x) == "raw"` es cierta, lo que significa que este es uno de los tipos de almacenamiento básico en R y que no habrá otras clases de objetos definidas como tipo crudo diferente a los vectores de tipo crudo.

6 Formato binario y hexadecimal

En computación, los valores crudos se refieren a datos almacenados en formato binario, es decir, sin codificar o, figurativamente, "sin formato". El formato binario es el estado de almacenamiento fundamental de la información digital y está basado en la notación del sistema de numeración binario. Un número binario está compuesto por dígitos que sólo pueden adoptar dos valores alternativos: el cero (0) o el uno (1). Aunque un número en notación binaria puede tener un número indefinido de dígitos, en el formato de almacenamiento de datos binario el número máximo de dígitos para cada número suele fijarse en ocho. A este rango de valores de ocho dígitos binarios se le denomina, por convención, *byte* u octeto.

Por consiguiente, un byte de ocho dígitos podrá adoptar solamente 256 valores diferentes, resultado del número de combinaciones únicas de ceros y unos que hay en ocho posiciones (2^8 = 256). Leído en notación decimal, el rango de valores de un byte va de 0 a 255. Debido a que la representación en notación binaria de los bytes es algo difícil de interpretar para cualquier persona, los datos crudos suelen desplegarse en pantalla con la notación hexadecimal, la cual requiere de sólo dos dígitos para representar el valor de un byte completo.

Un número hexadecimal está compuesto por dígitos que pueden adoptar dieciséis valores diferentes: los dígitos de cero (0) a nueve (9) seguidos de las letras del alfabeto latino de la letra a (a = 10) a la letra efe (f = 15). Dado que cada dígito hexadecimal puede adoptar dieciséis valores diferentes, bastarán dos dígitos hexadecimales para representar todas las combinaciones de valores posibles (16^2 = 256) de un byte de ocho dígitos. Por ello, gracias a que permite representar código binario en forma —ligeramente— más legible para las personas, la notación hexadecimal es utilizada ampliamente en programación para representar datos binarios o crudos.

Para realizar la conversión del valor de un byte en notación hexadecimal a su valor en notación decimal basta con multiplicar el valor en decimal del primer dígito hexadecimal (de izquierda a derecha) por dieciséis, y luego sumar al resultado el valor en decimal del segundo dígito hexadecimal. Por ejemplo, el valor del *byte* hexadecimal `ef` es equivalente al resultado de (14 x 16) + 15 = 239. De esta manera, un byte binario de extensión fijada en ocho dígitos, tendrá un valor equivalente de tres dígitos en notación decimal y dos dígitos en notación hexadecimal.

Cabe mencionar que las letras de los dígitos hexadecimales pueden denotarse en mayúsculas o minúsculas indistintamente. No obstante, el tipo crudo de `R` utilizará solo las minúsculas para representar a los dígitos hexadecimales en pantalla.

7 También véase

Las funciones `charToRaw()`, `rawShift()` y las otras funciones de conversión entre tipos de almacenamiento y el tipo crudo.

El operador lógico `&` para operaciones *bit* a *bit* sobre vectores crudos.

`writeBin()` y `readBin()` para escribir y leer vectores crudos como conexiones.

8 Ejemplos

```
xx <- raw(2)
xx[1] <- as.raw(40) # Adviértase: no simplemente 40.
xx[2] <- charToRaw("A")
xx ## 28 41 - un vector crudo imprime valores hexadecimales
```

```
dput(xx) ## igual que as.raw(c(0x28, 0x41))
as.integer(xx) ## 40 65
x <- "Una cadena de prueba"
(y <- charToRaw(x))
is.vector(y) # TRUE
rawToChar(y)
is.raw(x)
is.raw(y)
stopifnot( charToRaw( "\xa3") == as.raw(0xa3) )

esASCII <- function(txt) all(charToRaw(txt) <= as.raw(127))
esASCII(x) # verdadero
esASCII("\xa325.63") # falso (en el estándar Latin-1, ésta es una cantidad en libras esterlinas)
```

9 Código fuente

9.1 raw()

```
function (length = 0L)
   .Internal(vector("raw", length))
```

9.2 as.raw()

```
function (x) .Primitive("as.raw")
```

9.3 is.raw()

```
function (x) .Primitive("is.raw")
```

10 Sobre la traducción

La traducción al español de esta página de ayuda fue actualizada el 09 de mayo de 2021 y está basada en la documentación original de R en inglés para la versión 4.0.3. La revisión técnica de esta página de ayuda todavía no ha sido realizada. Si deseas participar realizando la revisión técnica o sugiriendo mejoras gramaticales/ortográficas/estilísticas al texto, por favor dirígete a la página del proyecto en: https://github.com/sicabi/documentacionR, para saber un poco más sobre cómo contribuir. Toda contribución será atribuida a la persona que la realice.