PROBABILIDAD Y ESTADÍSTICA

Curso de R

Gustavo A. Colmenares
gcolmenares@yachaytech.edu.ec
gcolmena@gmail.com

3- Fundamentos del Lenguaje

UNIVERSIDAD YACHAY TECH

SCHOOL OF MATHEMATICAL AND COMPUTATIONAL SCIENCES

Asignar un valor a una variable

El **operador asignar** (<-) en **R**, es probablemente el operador más importante de todos, pues nos permite asignar datos a variables. Por ejemplo, asignamos valores a las variables estatura y peso:

```
estatura <- 1.73
peso <- 83
```

Observaciones:

1. Aunque podemos usar el signo igual (=) para una asignación, es preferible utilizar <-, primero, porque es característico de R, segundo porque es fácil de reconocer visualmente y tercero, porque es el signo igual se usa para los atributos de los objetos y funciones (se verá más adelante).
2. Observe los espacios antes y después del operador <-. Esto es intencional y es aconsejable como "buenas prácticas" a la hora de escribir. Cuando tenemos muchas líneas de código es más agradable a la vista y facilita la lectura.

Llamamos a sus valores asignados, simplemente escribiendo el nombre de la variable:

```
> estatura
[1] 1.73
> peso
[1] 83
```

Asignar un valor a una variable

Usamos los valores asignados para realizar operaciones, por ejemplo:

```
peso / estatura ^ 2
```

```
[1] 27.7323
```

Cambiamos el valor de una variable a uno nuevo y realizamos operaciones:

```
peso <- 76
peso
```

```
[1] 76
```

```
peso / estatura ^ 2
```

```
[1] 25.39343
```

```
estatura <- 1.56
peso <- 48
peso / estatura ^ 2
```

```
[1] 19.72387
```

Asignamos el resultado de una operación a una variable nueva:

```
resultado <- peso / estatura ^ 2
resultado
```

```
[1] 19.72387
```

Asignar un valor a una variable

R discrimina entre letras mayúsculas y minúsculas para los nombres de variables y objetos (se verán en breve), es decir que **area** y **AREA** se refieren a variables diferentes en memoria.

Si está usando palabras completas para dar nombre a las variables, puede usar combinaciones de minúsculas y mayúsculas, también puede usar el signo _ y el punto (.)

```
b <- 1.75
a <- 2.5
arearectangulo <- b * a
AreaRectangulo <- b * a
area_rectangulo <- b * a
area.rectangulo <- b * a

arearectangulo
AreaRectangulo
area_rectangulo
area.rectangulo
```

Tipos de datos

En **R** los datos pueden ser de diferentes tipos. Cada tipo tiene características particulares que lo distinguen de los demás. Entre otras cosas algunas operaciones sólo pueden realizarse con tipos de datos específicos. Los tipos de datos de uso más común son los siguientes:

Tipo	Ejemplo	Nombre en inglés
Entero	1	integer
Numérico	1.3	numeric
Cadena de texto	“uno”	character
Factor	uno	factor
Lógico	TRUE	logical
Perdido	`NA`	NA
Vacio	`NULL`	null

Además de estos tipos, en R también contamos con datos **complejos** (con una parte real y una imaginaria), **raw** (bytes), **fechas** y **raster** (fotografías aéreas digitales, imágenes de satélite, imágenes digitales o incluso mapas escaneados) entre otros. Estos tipos tiene aplicaciones muy específicas, por ejemplo, los datos de tipo fecha son ampliamente usados en economía, para análisis de series de tiempo.

Tipos de datos

Entero (Integer): Representan números enteros sin una parte decimal o fraccionaria, que pueden ser usados en operaciones matemáticas.

Numérico (Numeric): Representan números. La diferencia de estos con los datos enteros es que tiene una parte decimal o fraccionaria. Un dato numérico en **R** es equivalente a *double* o *float* (flotantes) en otros lenguajes de programación.

Lógico (Logic): Sólo admite dos valores específicos: verdadero (TRUE) y falso (FALSE). Representan si una condición o estado se cumple, es verdadero, o no, es falso. Este tipo de dato es, generalmente, el resultado de operaciones relacionales y lógicas, son esenciales para trabajar con álgebra Booleana.

Cadena de texto (Character): representa texto y es fácil reconocerlo porque un dato siempre esta rodeado de comillas, simples o dobles. De manera convencional, nos referimos a este tipo de datos como cadenas de texto, es decir, secuencias de caracteres. Este es el tipo de datos más flexible de **R**, pues una cadena de texto puede contener letras, números, espacios, signos de puntuación y símbolos especiales.

NA: Indica valor perdido. Abreviatura de "Not available“. Es usado para representar explícitamente datos perdidos, omitidos o que por alguna razón son faltantes.

Tipos de datos

Inf: Indica infinito.

NaN: Indica no numérico. Abreviatura de "Not a number"

NULL: Indica valor nulo, es decir, sin contenido. Generalmente se usa en parámetros de función, lo que indica que al parámetro no se le asigna ningún valor. También se usa a menudo para inicializar una variable, lo que significa que la variable no tiene contenido, por lo que su longitud es cero.

Factor: Es un tipo de datos propio de **R**. Puede ser descrito como un dato numérico representado por una etiqueta.

Supongamos que tenemos un conjunto de datos que representan el sexo de personas encuestadas por teléfono, pero estos se encuentran capturados con los números 1 y 2. El número 1 corresponde a femenino y el 2 a masculino. Si se convierten estos datos a factor, se mostrará los 1 como femenino y los 2 como masculino. **R** trata a los factores de manera diferente a un dato numérico para así facilitar cálculos estadísticos y para reducir el espacio de almacenamiento necesario.

Por último, cada una de las etiquetas o valores que puedes asumir un factor se conoce como nivel. En nuestro ejemplo con femenino y masculino, tendríamos dos niveles.

Tipos de datos

Coerción: En **R** los datos pueden ser coercionados, es decir, forzados, para transformarlos de un tipo a otro. Cuando pedimos a **R** ejecutar una operación, intentará coercionar de manera implícita, sin avisarnos, los datos de su tipo original al tipo correcto que permita realizarla. Habrá ocasiones en las que **R** tenga éxito y la operación ocurra sin problemas, y otras en las que falle y obtengamos un error.

Lo anterior ocurre porque no todos los tipos de datos pueden ser transformados a los demás, para ello se sigue una regla general.

La coerción de tipos se realiza de los tipos de datos más restrictivos a los más flexibles, por tanto ocurren en el siguiente orden:

logical -> integer -> numeric -> character

No pueden ocurrir en orden inverso. Podemos coercionar un dato de tipo entero a uno numérico, pero nunca uno de cadena de texto a numérico.

Tipos de datos

También podemos hacer coerciones explícitas usando la familia de funciones **as()**.

Función	Tipo al que hace coerción
`as.integer()`	Entero
`as.numeric()`	Numerico
`as.character()`	Cadena de texto
`as.factor()`	Factor
`as.logical()`	Lógico
`as.null()`	`NULL`

Todas estas funciones aceptan como argumento datos o vectores (veremos qué es un vector en breve). Cuando estas funciones tienen éxito en la coerción, nos devuelven datos del tipo pedido. Si fallan, obtenemos NA como resultado.

Tipos de datos

Ejemplo de coerción válida:

```
as.character(5)
[1] "5"
```

Pero, si intentamos convertir la palabra “cinco” a un dato numérico, obtendremos un Warning y NA

```
as.numeric("cinco")

Warning message:
NAs introduced by coercion
```

Si coercionamos un dato de tipo lógico a numérico, TRUE siempre devolverá 1 y FALSE dará como resultado 0.

```
as.numeric(TRUE)
[1] 1
as.numeric(FALSE)
[1] 0
```

Por último, la función as.null() siempre devuelve NULL, sin importar el tipo de dato que demos como argumento.

```
as.null(FALSE)
NULL
```

Verificación con la familia de funciones is()

También podemos verificar si un dato es de un tipo específico con la familia de funciones **is()**.

Función	Tipo que verifican
`is.integer()`	Entero
`is.numeric()`	Numerico
`is.character()`	Cadena de texto
`is.factor()`	Factor
`is.logical()`	Lógico
`is.na()`	`NA`
`is.null()`	`NULL`

Estas funciones toman como argumento un dato, si este es del tipo que estamos verificando, nos devolverán TRUE y en caso contrario devolverán FALSE.

```
is.numeric(5)
[1] TRUE
is.character(5)
[1] FALSE
```

Operadores aritméticos

Este tipo de operador es usado para operaciones aritméticas

Operador	Operación	Ejemplo	Resultado
`+`	Suma	`5 + 3`	8
`-`	Resta	`5 - 3`	2
`*`	Multiplicación	`5 * 3`	18
`/`	División	`5 /3`	1.666667
`^`	Potencia	`5 ^ 3`	125
`%%`	División entera	`5 %% 3`	2

Es posible realizar operaciones aritméticas con datos de tipo **entero** y **numérico**.

Operadores relacionales

Los operadores lógicos son usados para hacer comparaciones y siempre devuelven como resultado TRUE o FALSE

Operador	Comparación	Ejemplo	Resultado
`<`	Menor que	`5 < 3`	`FALSE`
`<=`	Menor o igual que	`5 <= 3`	`FALSE`
`>`	Mayor que	`5 > 3`	`TRUE`
`>=`	Mayor o igual que	`5 >= 3`	`TRUE`
`==`	Exactamente igual que	`5 == 3`	`FALSE`
`!=`	No es igual que	`5 != 3`	`TRUE`

Operadores relacionales

Es posible comparar cualquier tipo de dato sin que resulte en un error. Sin embargo, al usar los operadores $>, >=, <$ y $<=$ con cadenas de texto, estos tienen un comportamiento especial. Por ejemplo:

```
"casa" > "barco"
[1] TRUE
```

Este resultado se debe a que se ha hecho una comparación por orden alfabético.

Operadores lógicos

Los operadores lógicos son usados para operaciones de álgebra booleana, es decir, para describir relaciones lógicas, expresadas como verdadero (TRUE) o falso (FALSO).

Operador	Comparación	Ejemplo	Resultado
`x | y`	x Ó y es verdadero	`TRUE | FALSE`	`TRUE`
`x & y`	x Y y son verdaderos	`TRUE & FALSE`	`FALSE`
`!x`	x no es verdadero (negación)	`!TRUE`	`FALSE`
`isTRUE(x)`	x es verdadero (afirmación)	`isTRUE(TRUE)`	`TRUE`

Los operadores `|` y `&` siguen estas reglas:

- `|` devuelve `TRUE` si alguno de los datos es `TRUE`
- `&` solo devuelve `TRUE` si ambos datos es `TRUE`
- `|` solo devuelve `FALSE` si ambos datos son `FALSE`
- `&` devuelve `FALSE` si alguno de los datos es `FALSE`

Orden de operaciones

Cuanto tenemos varias operaciones ocurriendo al mismo tiempo, en realidad, algunas de ellas son realizadas antes que otras y el resultado de ellas dependerá de este orden.

Orden	Operadores
1	^
2	* /
3	+ -
4	< > <= >= == !=
5	!
6	&
7	|
8	<-

Si deseamos que una operación ocurra antes que otra, rompiendo este orden de evaluación, usamos paréntesis o paréntesis anidados.

Miscelánea

La función **ls()** lista los objetos en memoria: sólo se muestran los nombres de los mismos.

Si se quiere listar solo aquellos objetos que contengan un carácter en particular, se puede usar la opción **pattern** (que se puede abreviar como pat):

Para borrar objetos en memoria, utilizamos la función **rm()**.

Para guardar el espacio o entorno de trabajo en **R** se puede utilizar la función **save.image()**. Los datos se guardarán en un archivo de tipo **Rdata**

Para guardar algunos objetos de tu workspace se usa la función **save()**

UNIVERSIDAD YACHAY TECH

SCHOOL OF MATHEMATICAL AND COMPUTATIONAL SCIENCES

FIN

Gustavo A. Colmenares gcolmenares@yachaytech.edu.ec / gcolmena@gmail.com