Tipos de datos y operadores

En R los datos pueden ser de diferentes tipos. Cada tipo tiene características particulares que lo distinguen de los demás. Algunas operaciones sólo pueden realizarse con tipos de datos específicos.

En esta sección revisaremos los tipos de datos más comunes en R y sus propiedades, así como la coerción entre tipos de dato.

Tipos de Datos

Hay muchos tipos de datos en R. Los más utilizados son los que tenemos en la tabla a continuación.

	Ejemplo	Consola
Entero	1	integer
Numérico	1.3	numeric
Lógico	TRUE	logical
Factor	Uno	factor
Cadena de texto	"uno"	character
Perdido	NA	NA
Vacío	NULL	NULL

También existen otros tipos de datos, como los valores complejos, los datos raw (bytes), las fechas, los raster, entre otros. Pero ésos no los veremos en este curso.

Datos Numéricos

Son el tipo de dato más común en lenguaje R. Una variable, vector, o matriz, será guardada como tipo de dato numérico si sus valores son números o contienen valores decimales. Por ejemplo, tenemos aquí dos variables que son guardadas como numéricas por defecto.

In [1]:
```
dato_numerico1 <- c(1.48,1.73,1.81,1.57)
dato_numerico2 <- c(5,3,0)
```

Imprime la variable `dato_numerico1` en la siguiente celda.

In [11]:
```
dato_numerico1
```

1.48 1.73 1.81 1.57

Verifica el tipo de dato de la variable `dato_numerico1` utilizando la función `class()` en la siguiente celda.

In [12]:
```
class(dato_numerico1)
```

'numeric'

Las variables que contienen valores enteros también son reconocidos de forma automática como datos numéricos por el lenguaje R. Imprime la variable `dato_numerico2` en la siguiente celda.

In [13]:
```
dato_numerico2
```

5 3 0

Verifica el tipo de dato de la variable `dato_numerico2` utilizando la función `class()` en la siguiente celda.

In [14]:
```
class(dato_numerico2)
```

'numeric'

También es posible inspeccionar el tipo de dato de una variable utilizando la función `str()` . Utilízala para verificar el tipo de dato de cualquiera de las dos variables numéricas.

In [15]:
```
str(dato_numerico1)
```

```
 num [1:4] 1.48 1.73 1.81 1.57
```

Datos Enteros

El tipo de dato entero son un caso especial de los datos numéricos. Se recomienda usarlo únicamente si se está seguro que los datos dentro de la variable nunca contendrán decimales. Por ejemplo, veamos 12 tiradas de un dado típico de seis caras.

In [16]:
```
tiradas_dados <- c(1,4,3,6,4,2,1,5,4,2,3,2)
```

Verifica el tipo de dato de la variable `tiradas_dados` utilizando la función `class()` .

In [17]:
```
class(tiradas_dados)
```

'numeric'

Sabemos que el dado sólo nos arrojará como resultado 1, 2, 3, 4, 5 ó 6, por lo que podemos representarlo como una variable discreta y que jamás nos dará valores con decimales. Entonces podemos guardarla como dato entero usando el comando `as.integer()` . Guárdala bajo el mismo nombre utilizando la función `as.integer()` .

In [18]:
```
tiradas_dados <- as.integer(tiradas_dados)
```

Verifica nuevamente el tipo de dato de la variable `tiradas_dados` utilizando la función `class()` .

In [19]:
```
class(tiradas_dados)
```

'integer'

Datos Lógicos

Las variables lógicas son variables que sólo pueden tener dos valores: `TRUE` (también reconocida como `T`) ó `FALSE` (también reconocida como `F`). Veamos los siguientes ejemplos:

In [20]:
```
dato_logico1 <- TRUE
dato_logico2 <- T
dato_logico3 <- FALSE
dato_logico4 <- F
```

Imprime las cuatro variables en las celdas siguientes para verificar que fueron guardadas correctamente.

In [21]:
```
dato_logico1
```

TRUE

In [22]:
```
dato_logico2
```

TRUE

In [23]:
```
dato_logico3
```

FALSE

In [24]:
```
dato_logico4
```

FALSE

Es posible "convertir" los datos lógicos en datos numéricos, donde un `FALSE` o `F` será convertido a 0 y un `TRUE` o `T` será convertido a 1 utilizando la función `as.numeric()`. Prueba cambiando la variable `dato_logico2` a variable numérica con la función `as.numeric()`.

In [25]:
```
as.numeric(dato_logico2)
```

1

También es posible "convertir" los datos lógicos en datos enteros utilizando la función `as.integer()`. Prueba cambiando la variable `dato_logico3` a variable entera con la función `as.integer()`.

In [26]:
```
as.integer(dato_logico3)
```

0

De forma similar, es posible cambiar los datos numéricos (y enteros) a lógicos utilizando la función `as.logical()`. Todos los datos que valen 0 serán convertidos a `FALSE` y los que son distintos a 0 serán convertidos a `TRUE`. Prueba convirtiendo la variable `dato_numerico2` a variable de tipo lógico.

In [27]:
```
as.logical(3)
as.logical(0)
as.logical(-1.1)
```

TRUE

FALSE

TRUE

Datos Factor

Las variables de tipo factor pudieran parecer como variables tipo caracter en el sentido de que también contienen texto. Sin embargo, este tipo de datos son usados cuando representan una variable categórica. Por ejemplo, los estados civiles (en México) sólo pueden tener seis valores distintos: "Soltero", "Casado", "Divorciado", "Separación en proceso judicial", "Viudo" o "Concubinato", por lo que podemos representarlo como una variable tipo factor. Por otro lado, los nombres de personas tienen demasiadas posibilidades y es por ésto que nos conviene representarlas como una variable de tipo caracter. Para crear una variable de tipo factor utilizamos la función `factor()` .

In [28]:
```
dato_factor <- factor(c("Soltero","Casado","Soltero","Soltero","Divorciado","Viudo","Concubina
                        "Casado","Separación en proceso judicial","Concubinato","Soltero","Cas
```

Imprime la variable `dato_factor` en la siguiente celda.

In [29]:
```
dato_factor
```

Soltero Casado Soltero Soltero Divorciado Viudo Concubinato Casado Soltero Casado Separación en proceso judicial Concubinato Soltero Casado

▶ **Levels**:

Para conocer los diferentes niveles de una variable de tipo factor, utilizamos la función `levels()` . Úsala para conocer los niveles de la variable `dato_factor` .

In [30]:
```
levels(dato_factor)
```

'Casado' 'Concubinato' 'Divorciado' 'Separación en proceso judicial' 'Soltero' 'Viudo'

Es posible "convertir" los datos de tipo factor en datos numéricos o enteros utilizando las funciones `as.numeric()` o `as.integer()` . Prueba convirtiendo la variable `dato_factor` en dato entero utilizando la función `as.integer()` .

In [31]:
```
as.integer(dato_factor)
```

5 1 5 5 3 6 2 1 5 1 4 2 5 1

¿Qué representan los números enteros en la celda anterior?

Datos Caracter

El tipo de dato caracter se utiliza para guardar texto, también conocidos como cadenas (o *strings*) en R. Hay dos formas de guardar datos de tipo caracter en R: usando comillas dobles `""` o comillas simples `''` . Veamos el siguiente ejemplo.

In [5]:
```
dato_caracter1 <- "¡Hola!"
dato_caracter2 <- 'Bienvenido al Curso Introductorio de R.'
```

Ahora imprime en cada una de las celdas las dos variables anteriores.

In [32]:
```
dato_caracter1
```

'¡Hola!'

In [33]:
```
dato_caracter2
```

'Bienvenido al Curso Introductorio de R.'

Verifica el tipo de dato de ambas variables utilizando la función `class()` .

In [34]:
```
class(dato_caracter1)
```

'character'

In [35]:
```
class(dato_caracter2)
```

'character'

Es posible forzar cualquier tipo de dato a caracter utilizando la función `as.character()` . Intenta cambiar las variables `dato_numerico1` , `tiradas_dados` , `dato_logico3` y `dato_factor` con la función `as.character()` .

In [36]:
```
as.character(dato_numerico1)
```

'1.48' '1.73' '1.81' '1.57'

In [37]:
```
as.character(tiradas_dados)
```

'1' '4' '3' '6' '4' '2' '1' '5' '4' '2' '3' '2'

In [38]:
```
as.character(dato_logico3)
```

'FALSE'

In [39]:
```
as.character(dato_factor)
```

'Soltero' 'Casado' 'Soltero' 'Soltero' 'Divorciado' 'Viudo' 'Concubinato' 'Casado' 'Soltero' 'Casado' 'Separación en proceso judicial' 'Concubinato' 'Soltero' 'Casado'

Coerción implícita y explícita

La **coerción implícita** de datos se realiza de los tipos de datos más restrictivos a los tipos de datos más flexibles. Esta coerción ocurre siempre de la siguiente manera:

$$\text{lógico} \rightarrow \text{numérico} \rightarrow \text{caracter}$$

En las secciones anteriores hemos hecho ya coerción implícita debido a que jamás especificamos el tipo de dato que queríamos, salvo para los datos de tipo factor y de tipo entero.

En las secciones anteriores también hicimos **coerción explícita**, aunque sin llamarle de ese modo. Para hacer ésto basta con utilizar alguna de las siguientes funciones: `as.logical()`, `as.integer()`, `as.numeric()`, `as.character()`, `as.factor()`, `as.null()`. Estas funciones convierten el dato que pongamos como parámetro en un tipo de dato que indica la función.

Para verificar si una variable está en algún tipo de dato en particular utilizamos las funciones `is.logical()`, `is.integer()`, `is.numeric()`, `is.character()`, `is.factor()`, `is.null()` y `is.na()`. Estas expresiones nos ayudarán a hacer validación de datos al momento de crear nuestras propias funciones, como veremos en el tema **Funciones**.

Sin embargo, si lo que queremos es revisar qué tipo de dato es nuestra variable, utilizamos la función `class()` como hemos visto anteriormente. La función `str()`, aunque nos ayuda también a conocer el tipo de dato que estamos usando, tiene otros usos que veremos cuando veamos estructuras de datos heterogéneas en los temas de **Data Frames** y **Listas**.

Operadores

Ya que conocemos los distintos tipos de datos, podemos realizar operaciones entre ellas utilizando distintos tipos de operadores:

- Operadores Aritméticos
- Operadores Relacionales
- Operadores Lógicos
- Operadores de Asignación

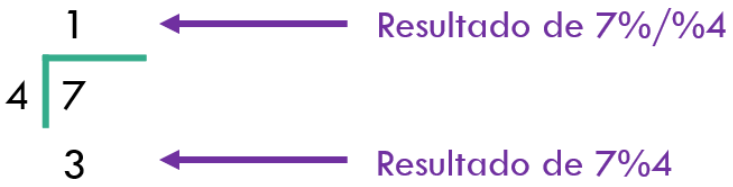
Operadores Aritméticos

Estos operadores nos ayudarán a realizar operaciones entre variables numéricas o enteras. Los operadores aritméticos son los siguientes:

Símbolo	Operación
+	Adición
-	Substracción
*	Multiplicación
/	División
^ **	Exponenciación
%%	Módulo
%/%	División entera
%*%	Producto matricial

Retomaremos el producto matricial en la **Matrices y Arreglos**. Sin embargo, nos interesa hacer un repaso de las operaciones módulo y división entera, pues no son tan comunes en la vida cotidiana.

Operación Módulo y División Entera



¿Qué obtendremos al evaluar `-3 %/% -2` ?

In [ ]:

Operadores Relacionales

Estos operadores nos permiten realizar una comparación entre dos datos, independientemente del tipo de dato que estemos comparando, y nos dará como resultado un valor de tipo lógico. Los operadores relacionales son los siguientes:

Símbolo	Operación
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Exactamente igual que
!=	Distinto que

¿Recuerdas las variables lógicas que creamos anteriormente? Recordemos las variables `dato_logico2` y `dato_logico3` .

In [6]:
```
dato_logico2
```

TRUE

In [7]:
```
dato_logico3
```

FALSE

Compara `dato_logico2` con `dato_logico3` utilizando operadores relacionales.

In [45]:
```
dato_logico2 <= dato_logico3
```

FALSE

Operadores Lógicos

Estos operadores nos permiten comparar uno o dos valores booleanos (o lógicos) y nos devolverá un valor de tipo lógico. Los operadores lógicos son los siguientes:

Símbolo	Operación
x$\|$y	Disyunción

Símbolo	Operación
x&y	Conjunción
!x	Negación
isTRUE(x)	Afirmación

Estos operadores, junto con los operadores relacionales, son particularmente útiles al momento de utilizar **Condicionales** y **Ciclos**. Sin embargo, en R son particularmente útiles para el manejo de vectores y matrices, como se verá cuando veamos **Vectores** y **Matrices**.

Tablas de Verdad

x	y	x$\|$y	x&y	!x	isTRUE(x)
TRUE	TRUE	TRUE	TRUE	FALSE	TRUE
TRUE	FALSE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	TRUE	FALSE

Recordemos las siguientes dos variables:

In [8]:
```
dato_logico1
```

TRUE

In [9]:
```
dato_logico3
```

FALSE

¿Qué valor nos arrojará la consola si escribimos `dato_logico1 & dato_logico1` ?

In [46]:
```
dato_logico1 & dato_logico1
```

TRUE

¿Qué valor nos arrojará la consola si escribimos `isTRUE(dato_logico1)` ?

In [48]:
```
isTRUE(dato_logico3)
```

FALSE

¿Qué valor nos arrojará la consola si escribimos `dato_logico1 | dato_logico3` ?

In [49]:
```
dato_logico1 | dato_logico3
```

TRUE

¿Qué valor nos arrojará la consola si escribimos `dato_logico1 & dato_logico3` ?

In [50]:
```
dato_logico1 & dato_logico3
```

FALSE

¿Qué valor nos arrojará la consola si escribimos `!dato_logico3` ?

In [51]:
```
!dato_logico3
```

TRUE

Operadores de Asignación

Como su nombre lo indica, este operador nos ayuda a asignar los valores o el resultado de una expresión (combinando los operadores anteriores) a una variable. Los operadores de asignación son dos: `<-` y `=` . Aunque sirven para lo mismo, el más usado por los usuarios del lenguaje R es el operador `<-` .

Jerarquía de Operadores

Al escribir una expresión en lenguaje R se sigue la siguiente jerarquía de operadores.

Orden	Operación
1	()
2	^
3	* /
4	+ -
5	< <= > >= == !=
6	!
7	&
8	$\|$
9	<-

Para expresiones muy largas se recomienda el uso de paréntesis `()` o bien realizar las operaciones en comandos por separado.

Ejemplos

¿Cuánto vale `9/3*(1+2)` ? Piénsalo un momento y compara tu respuesta con el resultado que te arroje la consola.

In [53]:
```
9/(3*(1+2))
```

1

¿Es `3*(3^3)` diferente a `(3*3)^3` ? Escríbelo utilizando operadores aritméticos y relacionales.

In [54]:
```
3*(3^3) != (3*3)^3
```

TRUE

In [55]:
```
3*(3^3)
```

81

In [56]:
```
(3*3)^3
```

729

In [57]:
```
3*(3^3) == (3*3)^3
```

FALSE

In [58]:
```
TRUE && TRUE
```

TRUE

In [ ]:
```

```