Nota2 - Introducción

Santiago Casanova y Ernesto Barrios

Uso Básico

Ahora vamos a empezar a familililarizarnos con el ambiente de R y específicamente su sintaxis especial.

Declaración de variables

R tiene un operador especial para asignar valores que es diferente a otros lenguajes de programación. Además, las son flexibles y pueden pasar de contener un tipo de dato a otro sin problema. Por los mismo no es necesario especificar el tipo de dato como se hace en C o en Java. El operador de asignación es `<-`. También se puede asignar variables con `=` pero, por convención, `=` se reserva para operaciones dentro de funciones o paréntesis.

Vamos a asignar algunos valores a una serie de variables:

```
numero <- 3
numero2 <- 1234.56
entero <- 20L
texto1 <- "ejemplo"
texto2 <- 'tambien se puede con comillas simples'
booleano <- T
booleano = TRUE
booleano2 = F
booleano2 = FALSE

typeof(numero)
```

```
## [1] "double"
```

```
typeof(entero)
```

```
## [1] "integer"
```

Como describimos, no se necesita especificar que tipo de dato queremos en cada variable ya que esto puede cambiar mas adelante. Hablando de tipos de datos, vamos a ver cuáles son las opciones que maneja R.

Tipos de Datos

1. Numerico: No hay diferencia inicialmente entre integer y double o float.
   - Si se quiere especificar solamente integer se puede escribir con una `L` al final. Ejemplo: `2L`
2. Caractér: No hay diferencia entre string y caracter.
3. Booleano (lógico): Como vimos en el ejemplo anterior, se escribe `TRUE` o `FALSE` todo con mayúsculas, o bién, solo `T` o `F`.

Ausencia de datos

R tiene varias maneras de manejar la ausencia de datos dependiendo del tipo. Esto puede ser con un valor `NA` pero estos igualemnte pueden ser más especificos. `NA_real_`, `NA_integer_`, `NA_character_` y `NA_complex_`

describen puntualmente el tipo de dato que falta pero en ultima instancia todos son tratados como `NA` por R. Por ejemplo:

```
NA
```

```
## [1] NA
```

```
NA_real_
```

```
## [1] NA
```

tienen la misma salida.

Además del `NA`, R reconoce `NaN` como **Not a Number** y es especifico para cuando el resultado de una operación matemática resulta en algo imposible. Por ejemplo la divisón de 0 entre 0. Es diferente a `NA` porque no indica que falte un valor sino que el velor resulto en un no-numero.

```
print(0/0)
```

```
## [1] NaN
```

Por ultimo tenemos el valor `NULL` que indica la ausencia de todo dato. Puede ser usado para desasignar variables. Por ejemplo:

```
var <- 'Ejemplo'
print(var)
```

```
## [1] "Ejemplo"
```

```
var <- NULL
print(var)
```

```
## NULL
```

Operadores

La mayoría de los operadores son muy similares o iguales a otros lenguajes de programación por lo que no los analizaremos a fondo.

```
#suma
print(20 + 3)
```

```
## [1] 23
```

```
#resta
print(20 - 3)
```

```
## [1] 17
```

```
#multiplicación
print(20 * 3)
```

```
## [1] 60
```

```
#división
print(20 / 3)
```

```
## [1] 6.666667
```

```
#potencia (diferente a otros)
print(20 ^ 3)
```

```
## [1] 8000
```

```r
#módulo
print(20 %% 3)
```

```
## [1] 2
```

Operadores lógicos

Los operadores lógicos son cruciales en la programación. En R, los básicos son prácticamente iguales a otros lenguajes pero además tiene una serie de prubas lógicas especiales.

```r
print(20 > 3) # >=
```

```
## [1] TRUE
```

```r
print(20 < 3) # <=
```

```
## [1] FALSE
```

```r
20 == 3
```

```
## [1] FALSE
```

```r
20 != 3
```

```
## [1] TRUE
```

Aqui también podemos notar que no es necesario escribir `print()` para que la consola responda.

Otra forma de obtener valores lógicos es revisando el tipo de dato que tenemos en una variable. La forma mas "burda" sería comparando el resultado de `typeof()` con el nombre que buscamos:

```r
typeof('asd') == 'character'
```

```
## [1] TRUE
```

pero R nos presenta unas formas mas elegantes de revisar esto. La familila de funciones `is.*` ejecuta exactamente este mismo proceso en una sola función.

```r
palabra <- 'palabra'
numeros <- 123.2

is.character(palabra)
```

```
## [1] TRUE
```

```r
is.character(numeros)
```

```
## [1] FALSE
```

```r
is.numeric(palabra)
```

```
## [1] FALSE
```

```r
is.numeric(numeros)
```

```
## [1] TRUE
```

Uso de RStudio

Hasta ahora hemos usado solamente la consola para interactuar con R pero hay maneras mas amigables de escribir comandos y recibir respuestas.

Una interfaz o IDE como RStudio nos permite escribir archivos ejecutables en los que podemos incluir todo un programa y editarlo sin tener que correr cada linea todas las veces.

Aquí incluimos una guía para instalar RStudio en diferentes sistemas operativos.

Windows

1. Primero visitar https://www.rstudio.com/products/rstudio/download/#download.
2. Hacer click en **Download** en la sección que dice *RStudio Desktop*.
3. Hacer click en **Download RStuio for Windows** si aparece el botón.
4. Si no aparece el botón, buscar *Windows 10/11* (o el que corresponda) en la lista **All Installers** y guardar el archivo ejecutable.
5. Correr el archivo .exe y seguir las instrucciones de instalación.

Mac

1. Primero visitar https://www.rstudio.com/products/rstudio/download/#download.
2. Hacer click en **Download** en la sección que dice *RStudio Desktop*.
3. Hacer click en **Download RStuio for Mac** si aparece el botón.
4. Si no aparece el botón, buscar *macOS 10.15+* (o el que corresponda) en la lista **All Installers** y guardar el archivo ejecutable.
5. Correr el archivo .exe y seguir las instrucciones de instalación.