Clases de objetos y funciones base

intro-R

Febrero 2018

Secciones

¿Script o consola?

Abrir RStudio

Secciones

R como calculadora

Podemos usar R como una calculadora muy avanzada:

- 1 + 1
- 24 / 7.2
- 1i * 1i

Aparte de los operadores típicos, hay algunos otros que nos pueden resultar muy útiles:

- 24 %% 6
- 23 %% 6
- 23 %/% 6

También podemos usarlo para comparar cantidades:

- 2 >3
- 12341 <= 1e7

Secciones

Asignación de objetos

Programar va de guardar información en la memoria (RAM) del ordenador.
Un trozo de memoria se llama un **objeto**.
Podemos guardar objetos *asignándoles un nombre*. La asignación se hace con el operador <-

- a <- 2
- b <- "hola caracola"

Nombres de objetos

Varias prácticas para nombrar objetos:

- snake_case. Ej: my_funcion, number_of_trees
- camelCase. Ej: myFunction, numberOfTrees

En este curso vamos a utilizar **snake_case**

Nombres de objetos

Varias prácticas para nombrar objetos:

- snake_case. Ej: my_funcion, number_of_trees
- camelCase. Ej: myFunction, numberOfTrees

En este curso vamos a utilizar **snake_case**

Buenos nombres

Un buen nombre no debe ser demasiado corto ni demasiado largo. Ante la duda, mejor pasarse de longitud.

Nombres de objetos

Varias prácticas para nombrar objetos:

- snake_case. Ej: my_funcion, number_of_trees
- camelCase. Ej: myFunction, numberOfTrees

En este curso vamos a utilizar **snake_case**

Buenos nombres

Un buen nombre no debe ser demasiado corto ni demasiado largo. Ante la duda, mejor pasarse de longitud.

Malos nombres

Según Andy Lester, los dos peores nombres para un objeto son **data** y **data2**

Secciones

Vectores I

Los objetos que guardas en la memoria pueden ser de distinta clase. La clase más importante en R es el vector atómico.
Un vector es un conjunto de uno o más elementos de un mismo tipo. Hay tres tipos básicos: numeric, character (o string) y boolean.

- a <- 2
- b <- "dos palabras"
- c <- TRUE

Vectores I

Los objetos que guardas en la memoria pueden ser de distinta clase. La clase más importante en R es el vector atómico.
Un vector es un conjunto de uno o más elementos de un mismo tipo. Hay tres tipos básicos: numeric, character (o string) y boolean.

- a <- 2
- b <- "dos palabras"
- c <- TRUE

Los vectores se crean con la función **c** (*concatenate)*

- numeric_vector <- c(1,2,3,4,5)
- character_vector <- c("Lo Malo", "Aitana War")
- boolean_vector <- c(T, T, T, F, F, T, F)

Vectores I

Los objetos que guardas en la memoria pueden ser de distinta clase. La clase más importante en R es el vector atómico.
Un vector es un conjunto de uno o más elementos de un mismo tipo. Hay tres tipos básicos: numeric, character (o string) y boolean.

- a <- 2
- b <- "dos palabras"
- c <- TRUE

Los vectores se crean con la función **c** (*concatenate)*

- numeric_vector <- c(1,2,3,4,5)
- character_vector <- c("Lo Malo", "Aitana War")
- boolean_vector <- c(T, T, T, F, F, T, F)

Existen otras formas de crear vectores:

- 1:10
- seq(from = 6, to = 124, by = 2.2)

Vectores I

Los objetos que guardas en la memoria pueden ser de distinta clase. La clase más importante en R es el vector atómico.
Un vector es un conjunto de uno o más elementos de un mismo tipo. Hay tres tipos básicos: numeric, character (o string) y boolean.

- a <- 2
- b <- "dos palabras"
- c <- TRUE

Los vectores se crean con la función **c** (*concatenate)*

- numeric_vector <- c(1,2,3,4,5)
- character_vector <- c("Lo Malo", "Aitana War")
- boolean_vector <- c(T, T, T, F, F, T, F)

Existen otras formas de crear vectores:

- 1:10
- seq(from = 6, to = 124, by = 2.2)

Vectores II: subscripting

Para acceder a los elementos de un vector usamos el número de orden del elemento entre corchetes (empezando por 1):

- plantas <- c("Fagus sylvatica", "Quercus suber", "Prunus spinosa", "Sorbus aucuparia")
- plantas[1]

Vectores II: subscripting

Para acceder a los elementos de un vector usamos el número de orden del elemento entre corchetes (empezando por 1):

- plantas <- c("Fagus sylvatica", "Quercus suber", "Prunus spinosa", "Sorbus aucuparia")
- plantas[1]

En realidad, dentro de los corchetes puedes poner cualquier vector:

- plantas[2:3]
- plantas[c(1,3)]

Vectores II: subscripting

Para acceder a los elementos de un vector usamos el número de orden del elemento entre corchetes (empezando por 1):

- plantas <- c("Fagus sylvatica", "Quercus suber", "Prunus spinosa", "Sorbus aucuparia")
- plantas[1]

En realidad, dentro de los corchetes puedes poner cualquier vector:

- plantas[2:3]
- plantas[c(1,3)]

Incluso puedes usar un vector lógico **de la misma longitud que el vector original**

- plantas[c(F,T,T,F)]

Vectores II: subscripting

Para acceder a los elementos de un vector usamos el número de orden del elemento entre corchetes (empezando por 1):

- plantas <- c("Fagus sylvatica", "Quercus suber", "Prunus spinosa", "Sorbus aucuparia")
- plantas[1]

En realidad, dentro de los corchetes puedes poner cualquier vector:

- plantas[2:3]
- plantas[c(1,3)]

Incluso puedes usar un vector lógico **de la misma longitud que el vector original**

- plantas[c(F,T,T,F)]

O una condición:

- numeros <- 1:10
- numeros[numeros >5]

Vectores II: subscripting

Para acceder a los elementos de un vector usamos el número de orden del elemento entre corchetes (empezando por 1):

- plantas <- c("Fagus sylvatica", "Quercus suber", "Prunus spinosa", "Sorbus aucuparia")
- plantas[1]

En realidad, dentro de los corchetes puedes poner cualquier vector:

- plantas[2:3]
- plantas[c(1,3)]

Incluso puedes usar un vector lógico **de la misma longitud que el vector original**

- plantas[c(F,T,T,F)]

O una condición:

- numeros <- 1:10
- numeros[numeros >5]

Listas

A diferencia del vector atómico, una lista puede contener elementos de distinto tipo. Se crean con la función list:

- list(45, "jamón", T, 1i, c(1,2,3))

Listas

A diferencia del vector atómico, una lista puede contener elementos de distinto tipo. Se crean con la función list:

- list(45, "jamón", T, 1i, c(1,2,3))

Incluso pueden contener otras listas:

- list(45, "jamón", list("esta es otra lista", 27.31))

Listas

A diferencia del vector atómico, una lista puede contener elementos de distinto tipo. Se crean con la función list:

- list(45, "jamón", T, 1i, c(1,2,3))

Incluso pueden contener otras listas:

- list(45, "jamón", list("esta es otra lista", 27.31))

Los elementos de una lista pueden tener nombre propio ("key"):

- plantas <- list("especie" = c("Fagus sylvatica", "Quercus suber"), "familia" = "Fagaceae")

Listas

A diferencia del vector atómico, una lista puede contener elementos de distinto tipo. Se crean con la función list:

- list(45, "jamón", T, 1i, c(1,2,3))

Incluso pueden contener otras listas:

- list(45, "jamón", list("esta es otra lista", 27.31))

Los elementos de una lista pueden tener nombre propio ("key"):

- plantas <- list("especie" = c("Fagus sylvatica", "Quercus suber"), "familia" = "Fagaceae")

Se puede acceder a los elementos de una lista de varias maneras:

- plantas[[1]]
- plantas$especie
- plantas$familia

Matrices

Una matriz es una particularización de un vector en dos dimensiones. Se crean utilizando la función **matrix**:

- mi_matriz <- matrix(1:9, ncol = 3, byrow = T)

Para acceder a los elementos de una matriz se utilizan la(s) fila(s) y la(s) columna(s) entre corchetes:

- mi_matriz[3,2]
- mi_matriz[1,]
- mi_matriz[2:3, 2:3]

Data Frames

Un **data frame** es a una matriz lo que una lista a un vector. Pueden contener elementos de distinto tipo entre columnas y estas columnas pueden ser llamadas por nombre:

- df <- data.frame("letras" = c("a", "b", "c", "d"), "numeros" = 4:7)

Hay muchas maneras de acceder a los elementos de un data frame:

- df[3,]
- df$letras

Secciones

Funciones base

Para crear, modificar o eliminar objetos en R utilizamos funciones. R tiene muchas funciones ya instaladas; hemos utilizado algunas en los ejemplos anteriores.
Una función puede tomar varios argumentos. Por ejemplo, la función sort, que sirve para ordenar los elementos de un vector, toma dos:

- sort(c(1, 4, 2), decreasing = T)

Algunos argumentos son opcionales; otros son obligatorios:

- sort(c(1,4,2)
- sort(decreasing = T)

A veces no sabemos qué hace una función o qué argumentos usa. Para eso tenemos la ayuda:

- ?sort
- ??sort