Básicos_de_R

Zyanya Tanahara

9/25/2020

Paquetes

Instalar paquetes

Para instalar paquetes se utiliza la función *install.packages*. No hay que olvidar el uso de dobles comillas en la sintaxis. Se puede instalar más de un paquete al mismo tiempo.

```
# install.packages(c("ggplot2", "swirl"))
# install.packages("swirl")
```

En R Studio también tienen la opción de usar la pestaña Packages para seleccionar directamente el paquete que quieren instalar.

Cargar paquetes

Usualmente tienen que cargar los paquetes para poder usarlos, esto se hace con la función library(). Notemos que en este caso la sintaxis es sin comillas

```
library(swirl)
```

```
##
## | Hi! Type swirl() when you are ready to begin.
```

```
library (ggplot2)
```

Paquetes útiles para instalar

```
# install.packages("swirl")
# install.packages("knitr")
# install.packages("rmarkdown")
```

Objetos

R tiene 5 tipos diferentes de objetos atómicos:

- Caracteres
- Numéricos (en $\mathbb{R}$)
- Enteros
- Complejos
- Lógicos (True/False)

Podemos conocer la clase de un objeto usando la función class()

Vectores

Contienen *objetos de la misma clase*. La excepción es una lista, que puede tener objetos de diferentes clases.

Para crear un objeto de tipo vector vacío se usa la función vector(). Para crear un vector dando sus elementos se usa c(x, y, z,...). La excepción a esto son las series, se asignan directamente.

```
a <- c(1,2,3)
class(a)
```

```
## [1] "numeric"
```

```
b <- 1:15
class (b)
```

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

```
c <- c(0.5, 0.6)
class (c)
```

```
## [1] "numeric"
```

```
d <- c(TRUE, FALSE)
class (d)
```

```
## [1] "logical"
```

```
e <- c(T, F)
class(e)
```

```
## [1] "logical"
```

```
f <- c("a", "b", "c")
class (f)
```

```
## [1] "character"
```

```
g <- c(1+0i, 2+4i)
class(g)
```

```
## [1] "complex"
```

```
h <- vector("numeric", length = 10)
h
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

Coerción de objetos Cuando se enlistan diferentes tipos de objetos, R lo convierte en un vector según su denominador común.

```
x <- c(1.7, "a")
y <- c(TRUE, 2)
z <- c("a", TRUE)
```

Ejercicio. Piensa los ejemplos anteriores para deducir su clase, luego compruébalo con la función class()

```
#Tu código va aquí
```

Listas

Es un tipo muy especial de vector que permite tener elementos diferentes.

```r
x <- list(1, "a", TRUE, 1 + 4i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1+4i
```

Matrices

Las matrices son vectores con el atributo de dimensión, el cual es un vector de dos entradas: (nrow,ncol). Un primer comando que podemos utilizar para crear matrices es la función matrix(nrow = x, ncol = y)

```r
m <- matrix(nrow = 10, ncol = 5)
m
```

```
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]   NA   NA   NA   NA   NA
##  [2,]   NA   NA   NA   NA   NA
##  [3,]   NA   NA   NA   NA   NA
##  [4,]   NA   NA   NA   NA   NA
##  [5,]   NA   NA   NA   NA   NA
##  [6,]   NA   NA   NA   NA   NA
##  [7,]   NA   NA   NA   NA   NA
##  [8,]   NA   NA   NA   NA   NA
##  [9,]   NA   NA   NA   NA   NA
## [10,]   NA   NA   NA   NA   NA
```

```r
 dim(m)
```

```
## [1] 10  5
```

```r
attributes(m)
```

```
## $dim
## [1] 10  5
```

Las matrices se construyen por columna en R, de manera predeterminada. Es decir, después de especificar el número de columnas y filas, la matriz se construye hacia abajo y al terminar la última fila de la primer columna, sigue con la segunda columna.

Ejercicio. Crea un vector de longitud 10. Escribe matrix(x, y, nrow= ?, ncol=?) con los valores adecuados.

```r
#Tu código va aquí
```

Ejercicio. Copia el código del vector que definiste anteriormente y haz con él una matriz de 4 columnas. Repite el ejercicio con una matriz de 3 columnas

```r
#Tu código va aquí
```

Una segunda forma de crear una matriz es añadiéndole el atributo de dimensión a un vector.

```
m <- 1:10
dim(m) <- c(2, 5)
```

Ejercicio. ¿Cuál es la diferencia entre los dos objetos del siguiente código?

```
x <- 1:10
y <- 1:10
dim(y) <- c(1,10)
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
y
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    2    3    4    5    6    7    8    9    10
```

Una tercera forma que tenemos para construir matrices es usando las funciones rbind() y cbind()

```
x <- 1:10
y <- 100:109
cbind(x,y)
```

```
##        x   y
##  [1,]  1 100
##  [2,]  2 101
##  [3,]  3 102
##  [4,]  4 103
##  [5,]  5 104
##  [6,]  6 105
##  [7,]  7 106
##  [8,]  8 107
##  [9,]  9 108
## [10,] 10 109
```

```
rbind(x,y)
```

```
##   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## x    1    2    3    4    5    6    7    8    9    10
## y  100  101  102  103  104  105  106  107  108   109
```

Factores

Se utilizan para representar variables categóricas. Por lo general no las usaremos tanto en este curso, pero es bueno que las conozcan.

```
x <- factor(c("hombre", "mujer", "hombre", "mujer", "hombre"))
x
```

```
## [1] hombre mujer  hombre mujer  hombre
## Levels: hombre mujer
```

```
table(x)
```

```
## x
## hombre  mujer
##      3      2
```

```
unclass(x)
```

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

```
## attr(,"levels")
## [1] "hombre" "mujer"
```

Data Frames

Son un tipo especial de lista y se suelen usar en conjunción con archivos del tipo de Excel (filas y columnas de la misma longitud). **A diferencia de las matrices, los data frames pueden estar formados por datos de tipo diferente**.

- Usualmente se crean usando las funciones read.csv() o read.table()
- Tienen un atributo especial llamado row.names()
- Con la función data.matrix() se puede convertir un data frame a una matriz

```
data("PlantGrowth")
# force(PlantGrowth) si no carga
class(PlantGrowth$weight)
```

```
## [1] "numeric"
```

```
class(PlantGrowth$group)
```

```
## [1] "factor"
```

```
x <- data.frame(peras = 1:4, manzanas = c(T, T, F, F))
```

Ejercicio. Si quieres repasar lo que vimos, haz los ejercicios 1, 3, 4 de swirl. Ejercicio. Haz los ejercicios 6, 7 y 8 de swirl.

Referencias

R Programming for Data Science de Roger D. Peng
Se puede conseguir gratis en el siguiente enlace
https://leanpub.com/rprogramming