Introducción al Análisis de Datos

Matías Alfonso

2019-08-07

Índice general

Parte I

Programación en R

Capítulo 1

Preliminares

R es un lenguaje de programación desarrollado inicialmente por Ross Ihaka y Robert Gentleman en el departamento de Estadística de la Universidad de Auckland en 1993. Está orientado específicamente con un enfoque al análisis estadístico.
R se desarrolla a partir de un lenguaje denominado S, desarrollado por John Chambers en 1976, disponible a partir del software comercial S-PLUS.
Es un lenguaje interactivo, permite la ejecución de instrucciónes en líneas de comando en una consola.

1.1 ¿Por qué R?

R puede ser ejecutado en múltiples plataformas y en la gran mayoría de los sistemas operativos. Puede ser ejecutado en tablets, teléfonos o computadoras. La utilización de scripts permite compartir fácilmente los análisis con los colegas, así como asegurar la reproductibilidad de los resultados. Todo lo que realizamos mediante una interfaz gráfica con el mouse no deja registros de nuestro trabajo e impide que podamos repasar nuestro trabajo para corregir errores.
La versatilidad y la potencia que otorga un lenguaje de programación es mucho mayor que la que podemos obtener con softwares estadísticos de interfaz gráfica.
La comunidad de usuarios y desarrolladores de R está en constante crecimiento en los últimos años. Hay una enorme cantidad de gente realizando nuevos desarrollos en R cada día, que están a la vanguardia de la ciencia computacional y estadística.

1.2 Software Libre

La mayor ventaja que tiene R con respecto a otros softwares de análisis estadístico es que es un software libre. ¿Qué quiere decir eso? Por un lado, que es gratuito. Por otro, que el código fuente con el que R fue desarrollado está abierto, se puede descargar y está diponible online. Actualmente el copyrigth de R lo posee la R Foundation. R forma parte del sistema GNU, desarrollado por la [Free Software Foundation](. De acuerdo a la Free Software Foundation, con el software libre se garantizan cuatro libertades fundamentales:

- La libertad de ejecutar el programa para cualquier propósito. (Libertad 0)
- La libertad de estudiar cómo el programa funciona y adaptarlo a tus propias necesidades. (Libertad 1)
- La libertad de redistribuir copias de manera que puedas ayudar a alguien. (Libertad 2)
- La libertad de mejorar el programa, y liberar tus mejoras al público, de manera que se beneficie toda la comunidad. (Libertad 3)

1.3 Sistema de Paquetes

El sistema de funcionalidades de R se encuentra agrupado en paquetes. La mayor parte de los paquetes se encuentran disponibles en Comprehensive R Archive Network (CRAN). Hay un conjunto de paquetes principales, de base, que incluye todos los paquetes que se instalan por defecto cuando instalamos R. Luego, tenemos un montón de paquetes con funcionalidades específicas que podemos instalar en función de nuestras necesidades.

Capítulo 2

Primeros pasos

2.1 Asignación de datos y evaluación.

R es un *lenguaje interpretado*. Esto quiere decir que le podemos ir pasando instrucciones y el programama las irá interpretando. Cuando ejecutamos el programa, nos encontramos con el prompt a la espera de intrucciones:

```
>
```

Una de las operaciones más sencillas que podemos realizar es la asignación de valores a las variables. El operador de asignación es `<-`.

```
x <- 1
print(x)
```

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

```
x
```

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

```
texto <- "hola mundo"
texto
```

```
## [1] "hola mundo"
```

Podemos imprimir el valor de una variable con la función `print()` o directamente escribiendo la variable.

Tenemos dos formas de interactuar con R:

- Tipear directamente los comandos en el prompt y ejecutarlos.
- Escribir un archivo de texto con todas las intrucciones y luego ejecutarlo. Este archivo se denomina script.

2.2 Working directory

Lo primero que debemos hacer cuando comenzamos a trabajar en R es configurar el directorio de trabajo. Una buena costumbre es crear un directorio nuevo de trabajo cuando comenzamos un proyecto nuevo. Luego configuramos esa carpeta como directorio de trabajo. Colocamos allí todos los archivos vinculados a ese proyecto. Para determinar en qué directorio estamos parados, podemos utilizar el comando `getwd()`. Para configurar el directorio de trabajo, utilizamos

```
setwd(#RUTA-A-DIRECTORIO)
```

2.3 Comentarios

Todo lo que escribamos luego de un `#` en una intrucción, no será evaluado.

```
x <- c(3, 4, 5)
## Esto no se ejecuta

x
```

```
## [1] 3 4 5
```

Ello nos permite comentar el código que escribimos, a manera de documentación.

2.4 Objetos básicos en R

Casi todo lo que encontremos en R, se denominan *objetos*. Hay 5 tipos de objetos básicos o atómicos:

- lógico
- numérico
- entero
- complejo
- caracter

Veamos algunos ejemplos:

```
## Logico
TRUE
```

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

```
FALSE
```

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

```
## Numérico
c(1.509, 2.859)
```

```
## [1] 1.509 2.859
```

```
## Enteros
1:10
```

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

```
## Caracter
"casa"
```

```
## [1] "casa"
```

Existen muchos más clases de objetos en R. Para averiguar de que tipo es un objeto, podemos utilizar la función `class()`

```
x <- 1:10
class(x)
```

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

```
class("TRUE")
```

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

Para preguntar por la clase de un objeto, podemos utilizar el comando `class()`

```
x <- 1:10
class(x)
```

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

```
y <- "casa"
class(y)
```

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

2.5 Factores

Los factores son básicamente objetos de clase entero, pero con etiquetas. Son datos categóricos y pueden estar ordenados o no.

```
f <- factor(c("si", "si", "no", "si"))
f
```

```
## [1] si si no si
## Levels: no si
```

```
f <- factor(c("bajo", "bajo", "medio", "alto"),
            levels = c("bajo", "medio", "alto"),
```

```
            ordered = TRUE)
f
```

```
## [1] bajo  bajo  medio alto
## Levels: bajo < medio < alto
```

2.6 Cómo buscar ayuda

R tiene un sistema de ayuda integrado. Si queremos saber para qué sirve un comando determinado o como pasarle los argumentos, podemos utilizar `?` o `help()`. Supongamos que queremos saber cómo se utiliza la función `sum()`

```
help(sum)

?vector
```

Capítulo 3

Estructuras de datos

3.0.1 Vectores

La forma más elemental de almacenar datos en R es en un vector. Un **vector** es una concatenación de objetos del mismo tipo. Podemos utilizar la función `c()` para crear vectores.

```
x <- c(1, 2, 3, 2.5)
x <- c(TRUE, FALSE)
x <- c(T, F)
x <- c("casa", "árbol", "patio")

## También podemos utilizar la función vector.
x <- vector(mode = "numeric", length = 10)
```

Si concatenamos elementos de diferente clase, R realizará una coerción automática de la clase de los objetos.

```
x <- c("casa", 2) ## character
x <- c(TRUE, 2) ## numeric

class(x)
```

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

3.1 Listas

Las listas también son una concatenación de elementos, pero pueden contener elementos de diferente clase. Para crear una lista, podemos utilizar `list()`

```
x <- list("peso", 2, "altura", 3, TRUE)
x
```

```
## [[1]]
## [1] "peso"
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] "altura"
##
## [[4]]
## [1] 3
##
## [[5]]
## [1] TRUE
```

3.2 Matrices

Las matrices son vectores, pero con un atributo de dimensión. La dimensión en sí es un vector de enteros de largo 2.

```
m <- matrix(1:9, nrow = 3)
m
```

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

```
dim(m)
```

```
## [1] 3 3
```

Al igual que los vectores, contienen objetos de la misma clase. Las matrices tienen algunas propiedades matemáticas interesantes, pues se pueden realizar operaciones especiales con ellas, por ejemplo, se pueden sumar o multiplicar.

3.3 Data frames

Los data frames son datos tabulados. Son tablas, donde cada columna puede ser de una clase diferente. Es un objeto particularmente útil para el análisis estadístico.

```
data.frame(Id = c(1, 2, 3),
           Nombre = c("Juan", "Carlos", "Ramona"),
           Altura = c(1.76, 1.80, 1.65))
```

```
##   Id Nombre Altura
## 1  1   Juan   1.76
## 2  2 Carlos   1.80
## 3  3 Ramona   1.65
```

3.4 Valores faltantes

Existen dos tipos de valores faltantes en R:

- NA
- NaN

Capítulo 4

Obteniendo datos

Existen una enorme cantidad de funciones para abrir archivos de diversos tipos.

4.1 Datos tabulares

Un formato estándar y abierto para guardar información en forma de tablas son archivos separados por comas (.csv) Para leer estos datos podemos utilizar:

- `read.tables()`
- `read.csv()`

```
## Leemos los datos desde un archivo y los guardamos en la variable base
base <- read.csv("data/titanic.csv")

## Imprimimos las primeras 5 filas de la base
head(base)[, 1:4]
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                 Name
## 1                             Braund, Mr. Owen Harris
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer)
## 3                              Heikkinen, Miss. Laina
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel)
## 5                            Allen, Mr. William Henry
```

```
## 6                                    Moran, Mr. James
```

También podemos leer datos directamente de la web

```
## Datos Abiertos
## Consulta de Medicamentos esenciales
salud <- read.csv("http://datos.salud.gob.ar/dataset/5fcacd04-58eb-4b43-89a0-55231c58f

head(salud)
```

```
##   provincia_id                  provincia_desc      año consultas_cantidad
## 1            2 Ciudad Autonoma de Buenos Aires año_2003              559785
## 2            6                     Buenos Aires año_2003             9544395
## 3           10                        Catamarca año_2003              372199
## 4           14                          Cordoba año_2003             3491961
## 5           18                        Corrientes año_2003              764887
## 6           22                             Chaco año_2003             1476825
```

Parte II

Introducción al análisis estadístico

Capítulo 5

Operaciones básicas

Comencemos calculando una proporción. Supongamos que realizamos 10 mediciones para la variables rendimiento escolar. La variable puede tomar tres valores: bajo, medio y alto.

```
rendimiento <- c(1, 3, 1, 2, 2, 1, 2, 2, 3, 2)

rendimiento <- factor(rendimiento,
                      labels = c("bajo", "medio", "alto"),
                      ordered = TRUE)

rendimiento
```

```
##  [1] bajo  alto  bajo  medio medio bajo  medio medio alto  medio
## Levels: bajo < medio < alto
```