

Fundamentals R

AUTHOR

Juan David Dueñas Garavito

Introducción

En este documento se abordarán los fundamentos del lenguaje de programación estadística R. R es uno de los lenguajes más populares para el análisis de datos y la ciencia de datos y es utilizado por muchos profesionales y organizaciones en todo el mundo. El objetivo de este documento es proporcionar una guía práctica para aquellos que quieren aprender a programar en R, así como para aquellos que ya tienen experiencia en R pero quieren mejorar sus habilidades y ampliar sus conocimientos.

Este documento está diseñado para ser utilizado como una referencia práctica para aquellos que trabajan con R en su trabajo o proyectos personales. Al final del documento, se espera que los lectores tengan las habilidades necesarias para trabajar con datos en R y aplicar técnicas de análisis de datos y visualización en sus proyectos de ciencia de datos.

Acerca de R

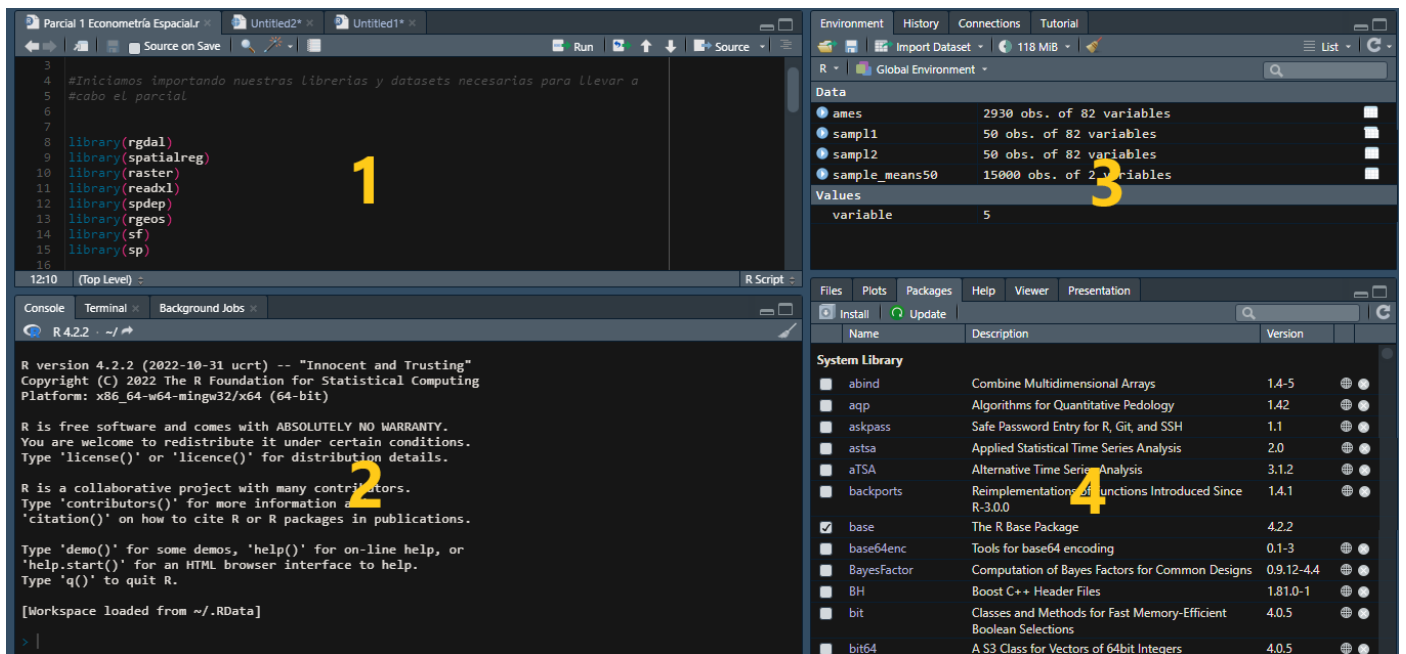
R es un lenguaje de programación y entorno de desarrollo estadístico gratuito y de código abierto, cualquier persona puede descargarlo utilizarlo y modificarlo esto permite que tenga una gran comunidad y documentación bien detallada de cada una de sus funcionalidades y paquetes, R es orientado a objetos lo que significa que los datos se organizan en objetos que se pueden manipular y analizar, R cuenta con una interfaz gráfica de usuario llamada RStudio que facilita el trabajo con el lenguaje y permite la creación y visualización de datos de manera sencilla.

R permite hacer diferentes tipos de análisis, Estadística Descriptiva, Estadística Bayesiana, Modelos Estado Espacio, entre otros es decir en R podemos hacer macroeconomía, microeconomía, econometría financiera, evaluación de impacto, econometría bayesiana, econometría espacial entre otros análisis estadísticos.

Interfaz de RStudio

RStudio proporciona una interfaz gráfica de usuario (GUI) que facilita el trabajo con R, lo que lo hace más accesible para los usuarios que no están familiarizados con la programación en línea de comandos.

Al ingresar a RStudio nos vamos a encontrar con cuatro principales paneles:



- 1. Editor :** En el editor será donde colocaremos el código y los comandos de ejecución.
- 2. Consola:** Allí veremos los resultados o los outputs del código.
- 3. Espacio de Trabajo:** Aquí podemos ver el historial de comandos que hemos usado, las diferentes variables, datasets, objetos y funciones que hemos declarado entre otras cosas.
- 4. Archivos / graficas / paquetes:** Allí podremos visualizar los plots que generemos en el código, las librerías que tengamos instaladas, los archivos de nuestro PC entre otros.

Nos vamos a centrar en el trabajo que se hace en el primer panel.

Comentarios

Es una buena práctica comentar nuestro código, esto significa ir colocando texto no ejecutable que explique que está haciendo nuestro código, esto facilitará más adelante nuestra vida al corregir el código y a otras personas que quieran entender lo que estamos haciendo, para comentar el código usamos el `#`, con esto el texto que se encuentre después del numeral no será ejecutado.

```
#Esto es un comentario
```

Paquetes y limpieza de entorno

R siempre guarda las diferentes variables, datasets entre otros objetos, es por esto que es una buena práctica al principio de cada código limpiar nuestro entorno, esto lo hacemos con el siguiente comando.

```
remove(list = ls())
```

Dado que R es un lenguaje Open Source, los usuarios pueden crear paquetes para facilitar o realizar nuevas operaciones que el R base no puede, sin embargo para hacer uso de estos paquetes debemos instalarlos y

luego activarlos, la instalación solo se hace una vez, la activación si debe hacerse en cada código que deseemos volver a abrir RStudio, es una buena practica activar todos los paquetes en las primeras líneas de código.

```
options(repos = c(CRAN = "http://cran.rstudio.com"))  
install.packages("tinytex")
```

Installing package into 'C:/Users/PC/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)

package 'tinytex' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\PC\AppData\Local\Temp\RtmpIlvTd7\downloaded_packages

```
install.packages("TeX")
```

Installing package into 'C:/Users/PC/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)

Warning: package 'TeX' is not available for this version of R

A version of this package for your version of R might be available elsewhere,
see the ideas at
<https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages>

```
install.packages("dplyr")
```

Installing package into 'C:/Users/PC/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)

package 'dplyr' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\PC\AppData\Local\Temp\RtmpIlvTd7\downloaded_packages

```
install.packages("ggplot2")
```

Installing package into 'C:/Users/PC/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\PC\AppData\Local\Temp\RtmpIlvTd7\downloaded_packages

```
library(dplyr)
```

Warning: package 'dplyr' was built under R version 4.2.3

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

filter, lag

The following objects are masked from 'package:base':

intersect, setdiff, setequal, union

```
library(ggplot2)
```

Warning: package 'ggplot2' was built under R version 4.2.3

```
library(tinytex)
```

Warning: package 'tinytex' was built under R version 4.2.3

Operadores Aritmeticos

R incluye varios operadores aritmeticos que se utilizan para realizar operaciones matematicas. Los operadores aritmeticos incluyen:

- `+` : Suma dos números
- `-` : Resta dos números
- `*` : Multiplica dos números
- `/` : Divide dos números
- `**` : Eleva un número a una potencia

Variables y tipos de objetos

Diremos que una variable es un “contenedor de información” en la que podemos “guardar” diferentes objetos, generalmente se escriben en minúscula, sin espacios ni tildes, comas, puntos, signos de interrogación o cualquier otro caracter a excepción del guión bajo, para declarar variable en R podemos usar `<-` como tambien `=`

Para notación de este trabajo usaremos unicamente el igual.

```
nombre_de_la_variable <- "Valor de la variable"  
nombre_de_la_variable = "Valor de la variable"
```

para visualizar nuestra variable en la consola (Panel #2) solo basta con llamarla de nuevo o utilizar la función print, o podemos usar un ; y volver a llamar la función en la misma línea

```
nombre_de_la_variable
```

```
[1] "Valor de la variable"
```

```
print(nombre_de_la_variable)
```

```
[1] "Valor de la variable"
```

```
nombre_de_la_variable = "Valor de la variable"; nombre_de_la_variable
```

```
[1] "Valor de la variable"
```

Los objetos que guardamos en las variables pueden ser de diferentes tipos y en función de esto la forma de guardarlos cambia.

```
objeto_numeric = 3  
objeto_logic = FALSE  
objeto_character = "Hola Mundo"  
objeto_complex = 1+0i
```

Para verificar de que tipo o clase es un objeto podemos usar la función class, puede darse el caso de que requiero convertir un objeto de un tipo a otro tipo, para esto debo usar alguna de las siguientes funciones según corresponda

```
x = 1  
class(x)
```

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

```
as.numeric(x)
```

```
[1] 1
```

```
as.logical(x)
```

```
[1] TRUE
```

```
as.character(x)
```

```
[1] "1"
```

Para hacer operaciones entre variables nos aseguramos que las variables sean de tipo numerico y operamos:

```
a = 5
b = 7

c = a + b
c
```

```
[1] 12
```

Vectores

Los vectores son entes matematicos que se usan para guardar datos de cierto tipo especifico, para crear vectores en R debemos usar el comando C, los vectores nos permiten hacer operaciones entre varios datos al mismo tiempo sin tener que recurrir a bucles

```
mi_primer_vector = c(25,5,3,6,3,6)
vector_dos = c("Hola", "mundo")
vector_tres = c(FALSE, TRUE, FALSE, TRUE)
```

Como los vectores solamente pueden guardar un tipo de dato si llegamos a introducir diferentes tipos de datos el vector automaticamente va a transformar estos datos para que el vector quede solamente en función de un tipo de dato.

```
y <- c(1.7, "a")  ## character
y <- c(TRUE, 2)   ## numeric
y <- c("a", TRUE) ## character
```

Podemos crear vectores con determinadas características de maneras mucho más eficientes, por ejemplo:

```
#Crear un vector secuencia 1 a 1.
Dias = 0:15; Dias
```

```
[1] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

```
#Crar un vector secuencia que avance cada j unidades: seq(a,b,j)
Medicion = seq(1,5, 0.5); Medicion
```

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
#Crear un vector donde Los objetos se repiten n veces: rep(objeto,n)
Numerico = rep(1,10); Numerico
```

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

```
#Crear un vector de repeticiones
```

```
Dummy = c(rep(0,5), rep(1,10), rep(0,5)); Dummy
```

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

Para concatenar vectores simplemente debemos:

```
vector_a = c(2,54,3)
```

```
vector_b = c(3,7,8)
```

```
vector_c = c(vector_a, vector_b); vector_c
```

```
[1] 2 54 3 3 7 8
```

Las operaciones con vectores funcionan de manera similar a las operaciones con variables, en últimas seguimos operando variables solo que en este caso las variables no contienen un solo valor sino contienen un vector de valores, los vectores entre si deben ser iguales o multiplos. En caso de ser iguales se hará la operación uno a uno, en cambio si son multiplos se repetirá la operación en los diferentes vectores, tal que:

```
#Operación uno a uno
```

```
vector_a = c(2,54,3)
```

```
vector_b = c(3,7,8)
```

```
vector_c = vector_a + vector_b; vector_c
```

```
[1] 5 61 11
```

```
#Operación entre multiplos
```

```
vector_a = c(2,54,3,4,8,5)
```

```
vector_b = c(3,7,8)
```

```
vector_c = vector_a + vector_b; vector_c
```

```
[1] 5 61 11 7 15 13
```

Matrices
