Guía básica de R

1. Vectores, tablas y primeros comandos

Óscar García Hernández

23 de octubre de 2018

Contents

¿Qué es esto?

Este documento tiene la intención de ser un a *GUÍA BÁSICA* introductoria al lenguaje de programación en R.

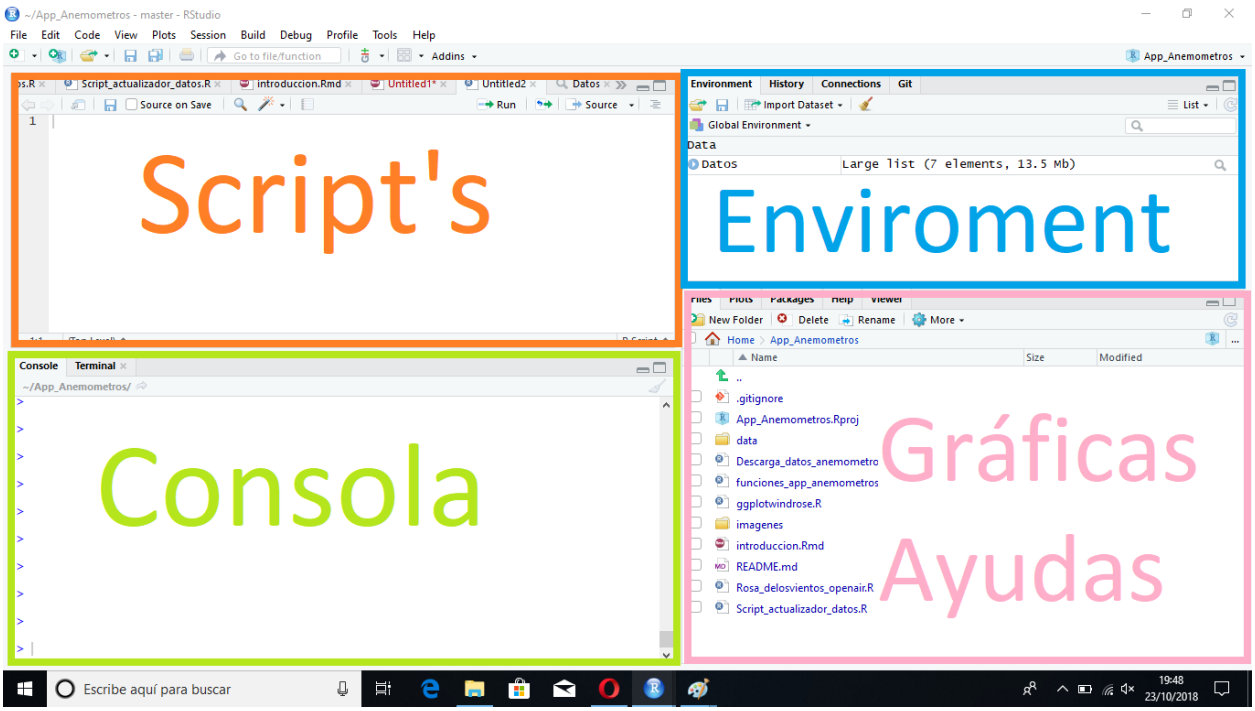
Para seguir está guía lo suyo es tener Rstudio abierto e ir pegando el código que aparece en los recuadrítos grises para ir tomando manejo con el software.

Mi intención es que estos documentos sean los suficientemente breve como para poder interiorizar la información bien y sin agobios.**¡Mucho ánimo!**

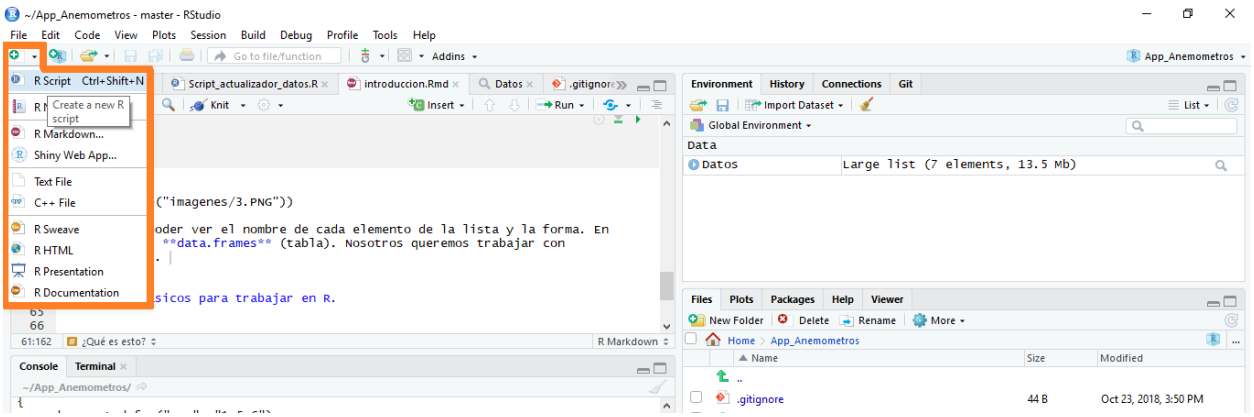
Interfaz Rstudio

Dentro de la interfaz de Rstudio hay varias secciones. Principalmente podemos definir 4.

- La parte destinada a Script's. Donde construiremos nuestro código.
- Una consola, donde ejecutamos comandos.
- Enviroment: donde se guardan las variable que definimos.
- Ventana multiusos donde se presentan las gráficas que elaboramos, aparecen las ayudas, los archivos de la carpeta de trabajo etc.



Si no te aparece la ventan de script's es porque aun no has creado uno.



Una vez que ya nos hemos creado el script podemos empezar a ver cositas básicas para programar.

Vectores

Es importante saber crear y tratar con vectores. En R los vectores se crean usando el siguiente comando.

```
#Se usa c(), y entre los parentesis ponemos los elementos del vector

c(1,2,3,4,5,6,7,8,9,10)
```

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

Si quiero guardar mi vector le pondré un nombre a la variable y se guardará en el enviroment.

```
vector_1 <- c(1,2,3,4,5,6,7,8,9,10)
```

Nótese que en R el símbolo "=" se escribe como "<-"

Tambien podemos crear vectores que contengan palabras o palabras y numeros.

```
#Las palabras se definen entre comillas
vector_2 <- c(1,"mono",3,4,"platano",6,7,8,"pato",10)
```

Si quiero acceder a un valor específico de mi vector usare corchetes "[]"

```
#Accedo al elemento 2 del vector
vector_2[2]
```

```
## [1] "mono"
```

Incluso puedo acceder a varios valores a la vez.

```
#Accedo al valor 2 y al valor 6 del vector.
vector_2[c(2,6)]
```

```
## [1] "mono" "6"
```

Nótese que para buscar el valor 2 y 6 del vector, debo introducir un vector. Si lo hago sin definir un vector da error, porque interpreta que estamos buscando el valor de la fila 2 y la columna 6, y nuestro vector solamente tienen una dimensión.

```
vector_2[2,6]
```

```
## Error in vector_2[2, 6]: número incorreto de dimensiones
```

Es importante interiorizar cuando se usan paréntesis y cuando se usan corchetes. Por norma general podemos decir que los corchetes nos sirven para buscar información dentro de vectores, matrices, tablas, etc.

A la hora de crear vectores se pueden usar otros atajos. Por ejemplo:

```
1:10
```

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

Nos crean un vector con un intervalo de 1.

Si queremos variar el intervalo del vector podemos usar el comando *seq*.

```
#seq(numero_inicial, numero_final, intervalo)
a<-seq(1,10,0.03)
```

```
##   [1]  1.00  1.03  1.06  1.09  1.12  1.15  1.18  1.21  1.24  1.27  1.30
##  [12]  1.33  1.36  1.39  1.42  1.45  1.48  1.51  1.54  1.57  1.60  1.63
##  [23]  1.66  1.69  1.72  1.75  1.78  1.81  1.84  1.87  1.90  1.93  1.96
##  [34]  1.99  2.02  2.05  2.08  2.11  2.14  2.17  2.20  2.23  2.26  2.29
##  [45]  2.32  2.35  2.38  2.41  2.44  2.47  2.50  2.53  2.56  2.59  2.62
##  [56]  2.65  2.68  2.71  2.74  2.77  2.80  2.83  2.86  2.89  2.92  2.95
##  [67]  2.98  3.01  3.04  3.07  3.10  3.13  3.16  3.19  3.22  3.25  3.28
##  [78]  3.31  3.34  3.37  3.40  3.43  3.46  3.49  3.52  3.55  3.58  3.61
##  [89]  3.64  3.67  3.70  3.73  3.76  3.79  3.82  3.85  3.88  3.91  3.94
## [100]  3.97  4.00  4.03  4.06  4.09  4.12  4.15  4.18  4.21  4.24  4.27
## [111]  4.30  4.33  4.36  4.39  4.42  4.45  4.48  4.51  4.54  4.57  4.60
```

```
## [122]  4.63  4.66  4.69  4.72  4.75  4.78  4.81  4.84  4.87  4.90  4.93
## [133]  4.96  4.99  5.02  5.05  5.08  5.11  5.14  5.17  5.20  5.23  5.26
## [144]  5.29  5.32  5.35  5.38  5.41  5.44  5.47  5.50  5.53  5.56  5.59
## [155]  5.62  5.65  5.68  5.71  5.74  5.77  5.80  5.83  5.86  5.89  5.92
## [166]  5.95  5.98  6.01  6.04  6.07  6.10  6.13  6.16  6.19  6.22  6.25
## [177]  6.28  6.31  6.34  6.37  6.40  6.43  6.46  6.49  6.52  6.55  6.58
## [188]  6.61  6.64  6.67  6.70  6.73  6.76  6.79  6.82  6.85  6.88  6.91
## [199]  6.94  6.97  7.00  7.03  7.06  7.09  7.12  7.15  7.18  7.21  7.24
## [210]  7.27  7.30  7.33  7.36  7.39  7.42  7.45  7.48  7.51  7.54  7.57
## [221]  7.60  7.63  7.66  7.69  7.72  7.75  7.78  7.81  7.84  7.87  7.90
## [232]  7.93  7.96  7.99  8.02  8.05  8.08  8.11  8.14  8.17  8.20  8.23
## [243]  8.26  8.29  8.32  8.35  8.38  8.41  8.44  8.47  8.50  8.53  8.56
## [254]  8.59  8.62  8.65  8.68  8.71  8.74  8.77  8.80  8.83  8.86  8.89
## [265]  8.92  8.95  8.98  9.01  9.04  9.07  9.10  9.13  9.16  9.19  9.22
## [276]  9.25  9.28  9.31  9.34  9.37  9.40  9.43  9.46  9.49  9.52  9.55
## [287]  9.58  9.61  9.64  9.67  9.70  9.73  9.76  9.79  9.82  9.85  9.88
## [298]  9.91  9.94  9.97 10.00
```

Incluso podemos utilizar **seq** para definir el tamaño del vector entre dos valores.

```
#seq(numero_inicial, numero_final, length.out= tamaño del vector)

b<- seq(1,10,length.out = 20)
```

```
##  [1]  1.000000  1.473684  1.947368  2.421053  2.894737  3.368421  3.842105
##  [8]  4.315789  4.789474  5.263158  5.736842  6.210526  6.684211  7.157895
## [15]  7.631579  8.105263  8.578947  9.052632  9.526316 10.000000
```

Nótese que en el script de R podemos añadir comentarios siempre y cuando pongamos delante "#". Esto es muy importe sobretodo cuando estamos aprendiendo, nos va a servir para "narrar" lo que hacemos en cada momento del código, escribir la funcionalidad de comandos nuevos que aprendemos, etc.

Comandos que nos dan información de las variables

Comando "class".

Este comando nos permite ver de que tipo es una variable.

```
class(vector_1)
```

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

```
class(vector_2)
```

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

Observamos que el *vector_1* es de tipo "numerico" y el *vector_2* es de tipo "character".

Comando "length"

Este comando nos da el tamaño de un elemento.

```
length(a)
```

```
## [1] 301
```

```
length(b)
```

```
## [1] 20
```

Tablas o matrices

A la hora de analizar datos es muy importante saber manejar tablas, ya que normalmente los datos suelen venir organizados de esta manera.

Comando "cbind" y "rbind"

Usaremos "vector_1" y "vector_2" definidos en el apartado de *vectores*.

El comando cbind nos une los elementos por columnas.

```
tabla_1 <- cbind(vector_1,vector_2)
```

```
##       vector_1 vector_2
##  [1,] "1"      "1"
##  [2,] "2"      "mono"
##  [3,] "3"      "3"
##  [4,] "4"      "4"
##  [5,] "5"      "platano"
##  [6,] "6"      "6"
##  [7,] "7"      "7"
##  [8,] "8"      "8"
##  [9,] "9"      "pato"
## [10,] "10"     "10"
```

Por defecto a la hora de realizar el comando cbind nos devuelve un elemento tipo "matriz".

```
class(tabla_1)
```

```
## [1] "matrix"
```

Comando "dim"

Este comando nos da información del tamaño de matrices y tablas

```
dim(tabla_1)
```

```
## [1] 10  2
```

Nos dice que "tabla_1" tiene 10 filas y 2 columnas.

Conversion de elementos

Suele ser interesante trabajar con tablas tipo *data.frame* en vez de con matrices.

```
tabla_1 <- as.data.frame(tabla_1)

class(tabla_1)
```

```
## [1] "data.frame"
```

De esta manera convertimos una matriz en un **data.frame**.

Suele ser normal tener que cambiar una variable, vector, matriz, tabla ... de "clase" o "forma". Por ello existen muchos comandos que empiezan por el prefijo: **as**.*formato_deseado*.

```
as.matrix()
as.array()
as.numeric()
as.vector()
as.factor()
as.integer()
as.Date()
as.character()
as.data.frame()
# Y muchos más, estos son los que más utilizaremos
```

Nombrar columnas

Suele ser habitual nombrar o renombrar las columnas de las tablas. Para ello usaremos **colnames**. Lógicamente los nombres de las columnas son palabras, definidas entre comillas y se usa un vector tan largo como número de columnas tengamos.

```
colnames(tabla_1) <- c("Columna 1", "Columna2")
```

```
##    Columna 1 Columna2
## 1          1        1
## 2          2     mono
## 3          3        3
## 4          4        4
## 5          5  platano
## 6          6        6
## 7          7        7
## 8          8        8
## 9          9     pato
## 10        10       10
```

Es menos habitual pero tambien se pueden nombrar las filas de las tablas usando **row.names()**. Para nosotros no tiene sentido, porque siempre trabajaremos con series temporales, es decir, las filas simplemente representan diferentes tiempos y por lo tanto no tienen nombre. A continuación un ejemplo de una tabla que recoje una serie temporal.

	s.since_1.1.1970	date_string_hour	Mean	Max	Dir_ch	Dir_deg
1	1540303537	23/10/2018 16:05:37	1.1	2.6	Northeast	45
2	1540303115	23/10/2018 15:58:35	2.1	4.8	Northeast	45
3	1540302694	23/10/2018 15:51:34	1.6	7.2	Northeast	45
4	1540302271	23/10/2018 15:44:31	0.9	6.5	North	0
5	1540301850	23/10/2018 15:37:30	1.1	4.8	Northeast	45
6	1540301428	23/10/2018 15:30:28	2.5	5.7	Northeast	45
7	1540301006	23/10/2018 15:23:26	1.7	5.1	Northeast	45
8	1540300584	23/10/2018 15:16:24	1	3.1	Northeast	45
9	1540300162	23/10/2018 15:09:22	2.6	5.5	Northeast	45
10	1540299740	23/10/2018 15:02:20	0.2	1.4	East-southeast	110
11	1540299319	23/10/2018 14:55:19	0.7	2.6	East-southeast	110
12	1540298896	23/10/2018 14:48:16	1	3.8	Northwest	315
13	1540298474	23/10/2018 14:41:14	0.2	1.3	Northwest	315
14	1540298053	23/10/2018 14:34:13	0.1	1.8	Southeast	135
15	1540297631	23/10/2018 14:27:11	0.8	3.3	Southeast	135
16	1540297209	23/10/2018 14:20:09	0.8	2.2	Northeast	45
17	1540296787	23/10/2018 14:13:07	0.8	3.2	Northeast	45
18	1540296366	23/10/2018 14:06:06	1.5	6.1	Northeast	45

Para accceder a la información de la tabla lo haremos de la siguiente manera.

```
#tabla[fila,columna]

tabla_1[1,1]
```

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

Si dejo la información de fila o columna vacía, se entiende que queremos todas las filas o todas las columnas.

```
#Queremos todas las filas de la columna 1

tabla_1[,1]
```

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

```
#Queremos todas las columnas de la fila 1
tabla_1[1,]
```

```
##   Columna 1 Columna2
## 1         1        1
```

Una de las ventajas de nombrar las columnas es la siguiente. Normalmente tendremos una tabla con muchas variables (columnas) y queremos acceder a una variable concreta. Si sabemos en qué columna está mi variable podemos acceder a esta información como ya hemos visto poniendo el número de columna. Pero, si hemos nombrado la columna debidamente podemos acceder a la informacion de dicha columna utilizando el símbolo "$". **Esto es muy útil**. *HAY QUE NOMBRAR LAS COLUMNAS.*

```
tabla_1$`Columna 1`
```

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

Moraleja

Este documento es el primero de unos cuantos que iré subiendo a nuestra carpeta compartida de github. Creo que es importante saber manejar y crear vectores y tablas para seguir avanzando en programación. Ya me comentaréis si preferís más nivel o vamos así, *poliki poliki*. Una vez ya estemos comodos trabajando con vectores y tablas pasaremos al siguiente nivel:

- Operaciones con vectores y tablas
- Operaciones lógicas
- Bucles, funciones if, while y for.

Pero eso más adelante.Un abrazzzzito.