Guía de trabajo programación con R

Diplomado en Análisis de datos con R para la acuicultura

true

29 abril 2021

Introducción **R** es un lenguaje de programación orientado a *objetos*. Las funciones y comandos para trabajar con objetos están contenidas en librerías o **packages**, algunas de las cuales ya están habilitadas al iniciar cualquier proyecto de **R** como las librerías **{base}**, **{graphics}**, **{stats}** o **{datasets}**.

Esta guía se elaboró usando **Rmarkdown** y está en formato html, lo que permitió integrar texto de bajo nivel o **markdown** y códigos desplegables con la solución a los ejercicios propuestos.

Cuando realices los ejercicios intenta **no mirar** la respuesta, hasta que hayas trabajado duro en generarla tu mismo. Trabajar en equipo es la mejor manera de aprender y conseguir resultados de calidad, si tienes dudas pregunta y si puedes ayudar a otro, hazlo.

Objetivos de aprendizaje Los objetivos de aprendizaje de esta guía son:

1). Iniciar un proyecto de análisis de datos con **R** usando el software **Rstudio.Cloud**.

2). Escribir un código de programación o *script* con el **software R**.

3). Familiarizarse con la creación y manipulación de objetos con **R**.

Objetos de uso frecuente en R

Objeto	Descripción
data.frame	Objeto que contiene datos. Habitualmente se crea al importar datos externos, aunque pueden ser creados dentro del mismo **R**. Consiste en una serie de variables (**vectors**) de igual longitud y que pueden ser del mismo tipo o no.
vectors	Colección de datos del mismo tipo (números, letras, etc.) que puede ser creado dentro de **R** o importado como una columna de un **data.frame**. Existen muchos tipos de vectores, entre ellos: **numeric** consiste de números reales; **integer** consiste de números enteros; **character** contiene letras, nombres, etc. (*Notar que cada elemento se encuentra entre comillas, por ej.* "A1"); **factor** sirve para representar variables categóricas, contiene información sobre los niveles del factor **(*levels*)** y sobre si estos niveles siguen un orden o no.
matrix	Arreglo que está conformado por filas y columnas, su estructura esta dada por la unión de vectores de un mismo tipo y largo. Cabe resaltar, que los elementos de la matriz deben ser de la misma naturaleza.

Comandos A continuación, se presentan algunos comandos que deberás usar para resolver los ejercicios de esta guía. Otros, los aprenderás mientras vas ejecutando algunas instrucciones y otros, los deberás investigar por tu propia cuenta.

Este comando permite crear objetos
<-
Por ejemplo, el siguiente código crea un objeto llamado datos con 4 elementos denominados 1, 2, 3 y 4.

datos **<- c**(1,2,3,4)

Crea un data.frame
data.frame()

Crea una matrix
matrix()

Explora objetos
class()
head()
tail()

Extrae variables de un objeto tipo dataframe
data.frame$variable
data.frame[,]

Función genérica usada para resumir datos de un objeto
summary()

Función genérica de la librería **base** para generar gráficos x-y
plot(x~y)

Ejercicios

Ejercicio 1. Conociendo un script

Abre el archivo script_example.R y familiarízate con la estructura de un *script* de **R**, con la forma en que está estructurado y de cómo ejecutar comandos. Observa la importancia de establecer con claridad los **metadatos** de un *script* y que el simbolo # se usa frecuentemente para hacer comentarios. Finalmente, toma un tiempo y discute con tus compañeros cual es la función de los comandos que se ejecutan en el ejemplo.

Ejercicio 2. Crea tu propio script

a). Ahora, crea tu propio *script* usando la barra de herramientas de **Rstudio**. Selecciona **File > New file > R script**. Guarda inmediatamente este script como **script_1_nombre_apellido**. Esto, es importante porque al finalizar la actividad deberás exportar el *script* y almacenarlo en tu carpeta drive **TAREAS_RSTUDIO.**

b). Crea los metadatos de tu *script* siguiendo el ejemplo anterior y haciendo referencia a la **tarea 1** y como primera tarea remueve los objetos de la sesión de trabajo usando **rm(list = ls())**.

Ejercicio 3. Trabajando con vectores

a). Investiga la librería **datasets** y el set de datos ***rivers*** usando la función **help().**

```r
help("datasets")
```

```
## starting httpd help server ... done
```

```r
help("rivers")
```

b). Ejecuta los comandos **class()** y **str()** para determinar algunas propiedades del objeto ***rivers***.

```r
class(rivers)
```

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

```r
str(rivers)
```

```
##  num [1:141] 735 320 325 392 524 ...
```

c). Comprueba que tipo de objeto es ***rivers*** usando las funciones lógicas **is.vector** e **is.data.frame.**

```r
is.data.frame(rivers)
```

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

```r
is.vector(rivers)
```

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

d). Crea un objeto llamado *datos* y almacena en él, el número de elementos del objeto ***rivers***. A continuación, explora la función del comando **cat** y responde la siguiente pregunta ¿Cuántos elementos o datos tiene el objeto ***rivers***?

```r
datos <- length(rivers)
cat("El objeto rivers tiene",datos,"datos")
```

```
## El objeto rivers tiene 141 datos
```

e). Ejecuta los comandos **head()** y **tail()** para inspeccionar los elementos contenidos al inicio y al final del objeto ***rivers***.

```r
head(rivers)
```

```
## [1] 735 320 325 392 524 450
```

```r
tail(rivers)
```

```
## [1]  500  720  270  430  671 1770
```

f). Usa la función **summary()** para obtener algunos estadísticos relevantes del objeto ***rivers***.

```
summary(rivers)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   135.0   310.0   425.0   591.2   680.0  3710.0
```

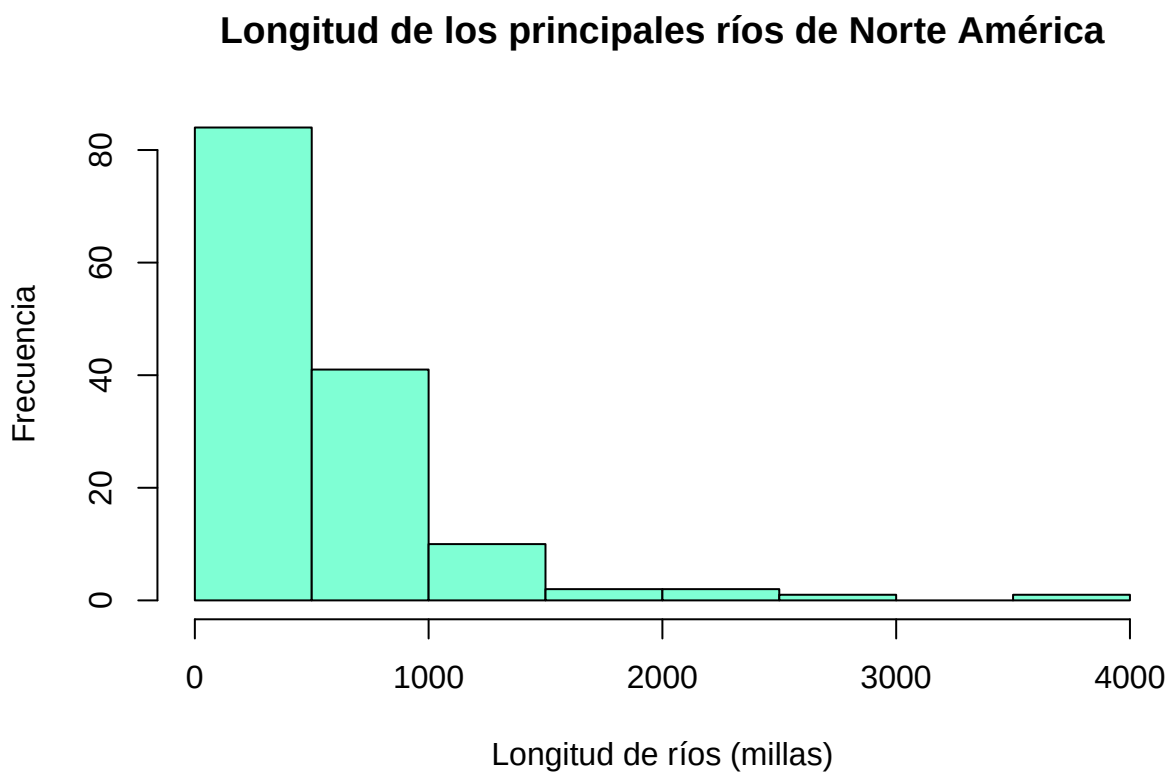
g). ¿Cuál es el rango de los datos contenidos en ***rivers***?. Usa el comando **cat()** para redactar tu respuesta.

```
rango <- max(rivers)-min(rivers)
cat("El rango de datos del objeto rivers es",rango, "millas")
```

```
## El rango de datos del objeto rivers es 3575 millas
```

h). Crea un histograma del objeto ***rivers***. Incluye los argumentos necesarios para agregar una leyenda a la figura, etiquetar los ejes y colorearla con el color de tu preferencia.

```
hist(rivers, main = "Longitud de los principales ríos de Norte América",
     xlab = "Longitud de ríos (millas)", ylab = "Frecuencia", col = "Aquamarine")
```



Ejercicio 4. Trabajando con matrices

La siguiente tabla contiene los coeficientes de parentesco entre 3 peces reproductores (fish10, fish20 y fish30) del núcleo de un programa de mejora genética de salmones. Un coeficiente de 0 significa que no son parientes, 0.25 que son medios hermanos, de 0.5 que son hermanos completos, mientras que el valor de 1 representa el parentesco de un individuo consigo mismo.

```r
kin_example <- matrix(c(1.1,0.5,0.25,0.5,1,0.25,0.25,0.25,1), nrow= 3,
                      dimnames = list(response = c("fish10", "fish20",
                                                   "fish30"),
                      treatment = c("fish10", "fish20","fish30")))

knitr::kable(kin_example, caption = "**Tabla 1.** Coeficientes de parentesco de 3 reproductores.")
```

Table 2: **Tabla 1.** Coeficientes de parentesco de 3 reproductores.

	fish10	fish20	fish30
fish10	1.10	0.50	0.25
fish20	0.50	1.00	0.25
fish30	0.25	0.25	1.00

a). Usando la función **matrix()** cree e imprime un objeto denominado **kin** con los datos de parentesco de la tabla anterior. No olvide usar el argumento *nrow=* para indicar el número de filas del objeto. No es necesario incluir los nombres de las filas y columnas del objeto.

```r
kin <- matrix(c(1,0.5,0.25,0.5,1,0.25,0.25,0.25,1), nrow= 3)
kin
```

```
##      [,1] [,2] [,3]
## [1,] 1.00 0.50 0.25
## [2,] 0.50 1.00 0.25
## [3,] 0.25 0.25 1.00
```

b). ¿Cuál es el parentesco del pez 10 consigo mismo?

```r
kin[1,1]
```

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

c). ¿Cuál es el parentesco entre el pez 10 y el 20?

```r
kin[1,2]
```

```
## [1] 0.5
```

```r
# kin[2,1] también es una respuesta válida
```

d). ¿Cuál es el parentesco del individuo 30 con el resto de los peces?

```r
kin[1:2,3]
```

```
## [1] 0.25 0.25
```

```
# kin[3,1:2] también es una respuesta válida
```

e). Investigue la función **lower.tri()**.

```
help("lower.tri")
```

f). Ejecute los siguientes comandos uno a uno para borrar la diagonal y la matriz inferior: a) **lower.tri(kin)** b) **kin[lower.tri(kin, diag = TRUE)] <- NA** c) **kin**.

```
lower.tri(kin)
```

```
##       [,1]  [,2]  [,3]
## [1,] FALSE FALSE FALSE
## [2,]  TRUE FALSE FALSE
## [3,]  TRUE  TRUE FALSE
```

```
kin[lower.tri(kin, diag = TRUE)] <- NA
kin
```

```
##      [,1] [,2] [,3]
## [1,]   NA  0.5 0.25
## [2,]   NA   NA 0.25
## [3,]   NA   NA   NA
```

g). ¿Qué representan las letras **NA**?

```
help(NA)
```

h). Ahora que ha borrado los registros repetidos y la diagonal, calcule el *promedio* del parentesco entre los peces con la función **mean()**. Note que debe especificar la existencia de valores perdidos **NA**.

```
mean(kin, na.rm=T)
```

```
## [1] 0.3333333
```

Ejercicio 5. Trabajando con Dataframe

a). Investiga y explora el set de datos ***BOD*** de la librería **datasets** con la función **summary()**.

```
help(BOD)
summary(BOD)
```

```
##       Time           demand
##  Min.   :1.000   Min.   : 8.30
##  1st Qu.:2.250   1st Qu.:11.62
##  Median :3.500   Median :15.80
##  Mean   :3.667   Mean   :14.83
##  3rd Qu.:4.750   3rd Qu.:18.25
##  Max.   :7.000   Max.   :19.80
```

b). ¿Qué propiedades y estructura tiene el objeto ***BOD***?

```r
str(BOD)
```

```
## 'data.frame':    6 obs. of  2 variables:
##  $ Time  : num  1 2 3 4 5 7
##  $ demand: num  8.3 10.3 19 16 15.6 19.8
##  - attr(*, "reference")= chr "A1.4, p. 270"
```

```r
BOD[3,1]
```

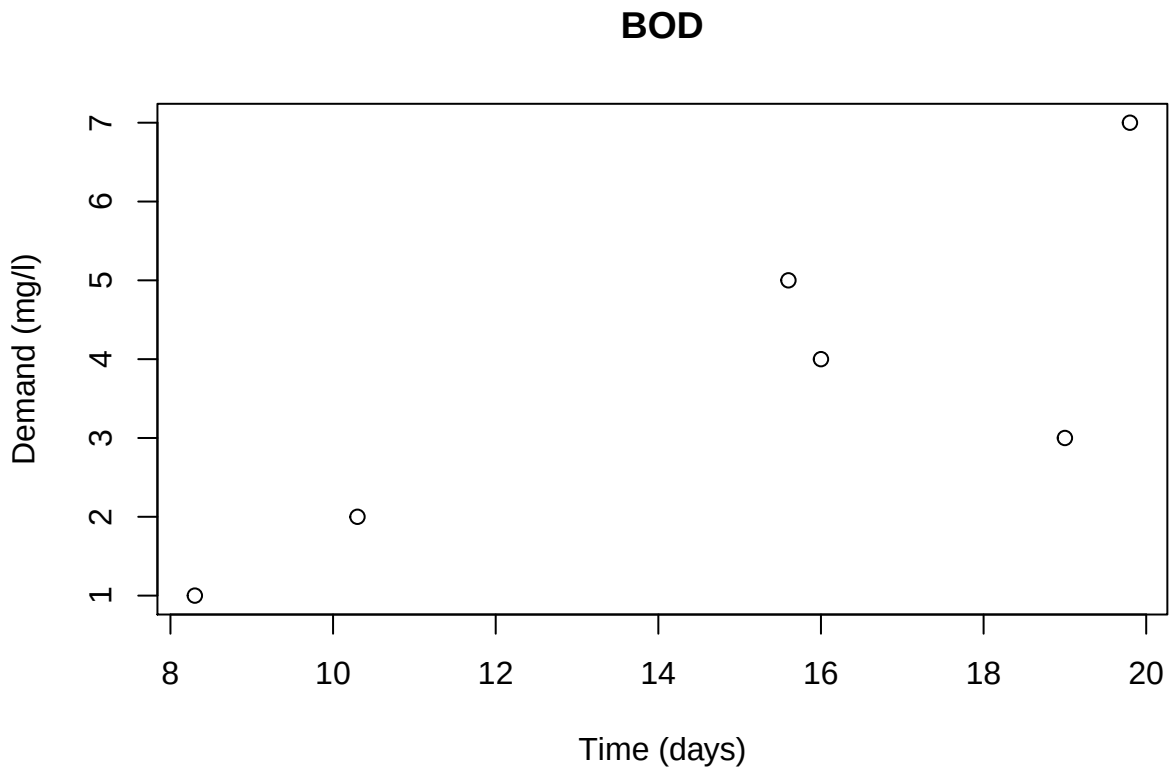
```
## [1] 3
```

c). Gráfique la relación entre ambas variables usando la función **plot()**.

Tips 1: Para graficar la relación entre dos variables con **plot()** debe escribir la relación entre **x** e **y** como **plot(x~y)** y no como **plot(x,y)**.

Tips 2: Dado que el data.frame tiene 2 variables debe indicar claramente que variable usará en cada eje **data.frame$variable**.

```r
plot(BOD$Time ~ BOD$demand, main = "BOD", xlab = "Time (days)", ylab = "Demand (mg/l)")
```



d). Calcule la correlación entre las variables Time y Demand usando la función **corr**.

```r
cor(BOD$Time,BOD$demand)
```

```
## [1] 0.8030693
```