Introducción a la Programación en R

Mihai M. Craiu

7/9/2020

Contents

Condicionales

Las **estructuras selectivas** o **condicionales** son aquellas que nos permten hacer una seleccion entre dos o varias rutas de ejecución posible. La selección se llevará a cabo según el valor de una condicioón. en lo sucesivo entenderemos por expresion a cualquier agrupación de comandos entre llaves, bien separadas por punto y coma, o bien por salto de línea. La primera estructura condcional que vamos a conocer es **if**, y no es más que un **si condicional:**

if (condición1) expresión1
Si la condicion de **condición1** se verifica (TRUE), entonces la sentencia a ejecutar recogida en *expresión1* se lleva a cabo. Si el resultado de la evaluación es falso, la sentencia a ejecutar simplemente se ignora.

Ejemplo: Número positivo

```
x <- 20
if (x>0) {paste(x,"es positivo")}
```

```
## [1] "20 es positivo"
```

Ejemplo: Número positivo, negativo o cero

```
y <- -15
if (y>0) {paste(y, "es positivo")}
if (y==0) {paste(y,"es cero")}
if (y<0) {paste(y, "es negativo")}
```

```
## [1] "-15 es negativo"
```

Ejemplo: Número par o impar (supone un uso interactivo de R)

```
n <- readline(prompt = "Introduzca un entero: ")
n <- as.numeric(n)
if (n%%2 == 0) {paste(n, "es par")}
if (n%%2 == 1) {paste(n, "es impar")}
```

En los dos últimos ejemplos hemos visto que a veces necesitamos que una sentencia se ejecute cuando se cumple la condición, y que otra sentencia se ejecute cuando dicha condición sea falsa. Si nos encontramos en este caso, la forma de resolverlo únicamente a través de sentencias **if** es engorrosa y poco práctica, por lo que existe otra forma más sencilla para llevar a cabo dicha tarea: la estructura **if-else:**:

if (condición1) expresión1 else expresión2
Si **condición1** es evaluada com verdadera, entonces se ejecuta la **expresión1**, en cambio, cuando la c ondición se evalúa como falsa, se ejecuta la **expresión2**

Las estructuras if-else se pueden **anidar** y el lenguaje R dispone además de una version vectorizada de if, *ifelse()*, sobre la que volveremos posteriormente.

Ejemplo: Número par o impar (supone un uso interactivo de R)

```
n <- readline(prompt= "Introduzca un entero: ") n <- as.numeric(n) if (n%%2 == 0) {paste(n,
"es par")} else {n, "es impar"}
```

```
y <- -15
if (y>0){
    paste(y, "es positivo")
}else if (y==0){
    paste(y, "es cero")
}else {paste(y, "es negativo")}
```

```
## [1] "-15 es negativo"
```

Ejemplo: Tipo de Bono

```
Sector_Emisor <- "Industria_Textil"
Vencimiento <- 5

if ( (Sector_Emisor=="Gobierno") | (Sector_Emisor=="GOBIERNO")){
    if(Vencimiento<=1){
        Clasif_Bono <- "GOB_CORTO"
    }else if (Vencimiento<=3){
        Clasif_Bono <- "GOB_MEDIO"
    }else{Clasif_Bono <- "GOB_LARGO"}
}else if ((Sector_Emisor=="Financiero") | (Sector_Emisor=="FINANCIERO")){
        if(Vencimiento<=1){
            Clasif_Bono <- "FIN_CORTO"
        }else if(Vencimiento>=3){
            Clasif_Bono <- "FIN_MEDIO"
        }else{Clasif_Bono <- "FIN_LARGO"}
}else {
        if(Vencimiento<=1){
            Clasif_Bono <- "NOFIN_CORTO"
        }else if (Vencimiento<=3){
```

```
            Clasif_Bono <- "NOFIN_MEDIO"
        }else{Clasif_Bono <- "NOFIN_LARGO"}
}
Clasif_Bono
```

```
## [1] "NOFIN_LARGO"
```

Ejemplo: Múltiplos de 4

```
m <- 15
cociente <- floor(m/4)
if(m%%4 ==0){
    paste(m,"es de la forma 4k, con k=", cociente)
}else if(m%%4==1){
    paste(m,"es de la fomra 4k+1, con k=", cociente)
}else if(m%%4==2){
    paste(m,"es de la forma 4k+2, con k=", cociente)
}else {paste(m,"es de la forma 4k+3,con k=", cociente)}
```

```
## [1] "15 es de la forma 4k+3,con k= 3"
```

A menudo sucede, como en el ejemplo anterior, que tenemos difernetes opciones dependiendo del valor de una variable. Para evitar la concatenación de excesivos condicionales if, tenemos a nuestra disposición la estructura **switch**:

switch(varaible, expresión1, expresión2, . . . ,expresionN)
Si **varaible** es un entero positivo i, entonces se ejecuta la expresión i-ésima.

switch(varaible, string1=expresión1, . . . , stringN=expresiónN)
Si **variable** es del tipo cadena de caracteres, entonces se ejecuta la expresión cuyo identificador coincida con el contenido de la variable. A diferencia del caso numérico, en caso de no coincidencia de la variable con los indetificadores, se ejectura la expresión por defecto **default**.

Ejemplos:

```
switch(1,"Madrid","Barcelona","Valencia")
```

```
## [1] "Madrid"
```

```
switch(3,"Madrid","Barcelona","Valencia")
```

```
## [1] "Valencia"
```

```
switch(0,"Madrid","Barcelona","Valencia")
switch("tres",uno="1",dos="2",tres="3")
```

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

```r
switch("siete",uno="1",dos="2",tres="3","Otro")
```

```
## [1] "Otro"
```

Ejemplo: Tipo de seguro

```r
Identif_seguro <- 3
switch(Identif_seguro,
        {TIPO_SEGURO <- "Hogar"},
        {TIPO_SEGURO <- "Autos"},
        {TIPO_SEGURO <- "Vida"})
TIPO_SEGURO
```

```
## [1] "Vida"
```

Ejemplo: Múltiplos de 4... ¿es correcto?

```r
m <- 9
cociente <- floor(m/4)
resto <- m%%4
switch(resto,
        {paste(m,"es de la forma 4k+1, con k =", cociente)},
        {paste(m,"es de la forma 4k+2, con k =", cociente)},
        {paste(m,"es de la forma 4k+3, con k =", cociente)},
        paste(m,"es de la forma 4k, con k =", cociente) )
```

```
## [1] "9 es de la forma 4k+1, con k = 2"
```

Ejemplo: Múltiplos de 4

```r
m <- 12
cociente <- floor(m/4)
resto <- as.character(m%%4)
switch(resto,
        "1" = {paste(m,"es de la forma 4k+1, con k =", cociente)},
        "2" = {paste(m,"es de la forma 4k+2, con k =", cociente)},
        "3" = {paste(m,"es de la forma 4k+3, con k =", cociente)},
        paste(m,"es de la forma 4k, con k =", cociente) )
```

```
## [1] "12 es de la forma 4k, con k = 3"
```

Ejercicios

Ejercicio propuesto: Clasificación por edad

Escribe un código que permita clasificar la variable RIESGO_ASEGURADO en "Alto", "Bajo" o "Moderado", en función de la variable EDAD_ASEGURADO, según que ésta sea menor de 25 años o mayor que 70.

```r
EDAD_ASEGURADO <- 35

if (EDAD_ASEGURADO < 25){
    RIESGO_ASEGURADO <- "Alto"; RIESGO_ASEGURADO
} else if (EDAD_ASEGURADO > 70) {
    RIESGO_ASEGURADO <- "Medio"; RIESGO_ASEGURADO
} else {RIESGO_ASEGURADO <- "Bajo"; RIESGO_ASEGURADO}
```

```
## [1] "Bajo"
```

Ejercicio propuesto: Clasificación por edad y sexo

Modifica el código anterior para que, en caso de que la variable SEXO ASEGURADO sea "M"(mujer, frente a "H", hombre), el correspondiente perfil de riesgo disminuya un nivel.

```r
EDAD_ASEGURADO <- 18
SEXO_ASEGURADO <- "M"

if (EDAD_ASEGURADO < 25){
    RIESGO_ASEGURADO <- "Alto"
    if (SEXO_ASEGURADO == "M") {RIESGO_ASEGURADO <- "Medio"}
    RIESGO_ASEGURADO
} else if (EDAD_ASEGURADO > 70) {
    RIESGO_ASEGURADO <- "Medio"
    if (SEXO_ASEGURADO == "M") {RIESGO_ASEGURADO <- "Bajo"}
    RIESGO_ASEGURADO
} else {RIESGO_ASEGURADO <- "Bajo"; RIESGO_ASEGURADO}
```

```
## [1] "Medio"
```

Ejercicio propuesto: Opción call

Escribe un código que permita obtener el benefcio/pérdida para un inversor que compra por 1e el díaa 3 de septiembre de 2018 una opción de compra (call) sobre unas acciones de Telefónica con vencimiento el 28 de septiembre de 2018, a un precio de ejercicio de 7.19e.

```r
Strike <- 7.19
Prima <- 1
PyG <- 0

Cotiz_Telef_28_09_2018 <- 8.19

if(Cotiz_Telef_28_09_2018 <= Strike){
        PyG <- -Prima; PyG
} else {PyG <- Cotiz_Telef_28_09_2018 - (Strike + Prima); PyG}
```

```
## [1] -1.776357e-15
```

```r
#Observación: Para R, debido a la aritmética de punto flotante que emplea para representar
#números reales, sucede que
sqrt(2)^2 == 2              #Arroja FALSE
```

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

```r
#En tal caso es preferible hacer:
all.equal(sqrt(2)^2, 2)     #Devuelve TRUE
```

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

```r
#En consecuencia,
all.equal(0, PyG)
```

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

Bucles

Los **bucles** o **ciclos** son estructuras que nos permiten repetir la misma instrucción un número determinado de veces, o hasta que se verifique cierta condición. Como veremos posteriormente, el carácter vectorial de R permite prescindir a menudo de este tipo de estructuras (que además pueden llegar a relantizar mucho la ejecución), si bien hay situaciones en que es imprescindible hacer uso de las mismas, además de ser uno de los pilares báscios de la programación. El primer bucle que veremos es el bucle **for**:

***for(variable1 in vector1) expresión1
Se evalúa la **expresión1** para cada uno de los valroes que **variable1** toma en el **vector1**.

Ejemplos:

```r
for(i in 1:10) {print(paste("Caso",toString(i)))}
```

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

```r
for(j in c("Madrid","Valencia","Barcelona")) {print(j)}
```

```
## [1] "Madrid"
## [1] "Valencia"
## [1] "Barcelona"
```

```r
suma <-0; for(i in 1:100) {suma <- suma + i}; suma
```

```
## [1] 5050
```

```r
temp <-2; for(k in c(1,3,5,7,9)) {temp<- temp + k}; temp
```

```
## [1] 27
```

Ejemplo: Factorial de un número

```r
N <- 5
factorial_N <- 1
for(i in 1:N){factorial_N<- factorial_N*i}
factorial_N
```

```
## [1] 120
```

Ejemplo: Lista de pares e impares

```r
for(i in 1:14){
    if(i%%2==0){print(paste(toString(i),"es par"))}
    else{print(paste(toString(i),"es impar"))}
}
```

```
## [1] "1 es impar"
## [1] "2 es par"
## [1] "3 es impar"
## [1] "4 es par"
## [1] "5 es impar"
## [1] "6 es par"
## [1] "7 es impar"
## [1] "8 es par"
## [1] "9 es impar"
## [1] "10 es par"
## [1] "11 es impar"
## [1] "12 es par"
## [1] "13 es impar"
## [1] "14 es par"
```

Como hemos visto, el bucle *for* es muy útil para repetir algo cuando conocemos el número de repeticiones previamente. Sin emabrgo, en algunas ocasiones no es posible conocer dicho número de antemano, sino que necesitamoes un ciclo que pueda ser controaldo por una condición, y la evaluación de esta condición dependerá del estado del programa en un momento dado. El bucle **while** nos permite hacer esto: