Control de Flujo

Luciano Selzer

28 June, 2018

¿Qué es el control de flujo?

Durante nuestros análisis vamos a querer que la tomar decisiones automáticamente bajo ciertas reglas dispuestas de antemano:

- Si una condición se cumple realiza una acción.
- Si una condición no se cumple realizar otra acción.
- O tal vez, repetir muchas veces la misma acción un número determinado de veces

---

```
# if
if (condicion es verdadera) {
  realizar acción
}

# if ... else
if (condicion es verdadera) {
  realizar acción
} else {  # esto es, la condición de if es falsa
  realizar acción alternativa
}
```

Por ejemplo, queremos que R imprima un mensaje si la variable `x` tiene un valor determinado:

```
# Muestra un número de una distribución Poisson
# con media (lambda) 8
x <- rpois(1, lambda = 8)
if (x >= 10) {
  print("x es más grande o igual que 10")
}
x
```

```
[1] 8
```

Puede que no obtengan el mismo número que su compañero porque son número aleatorios o mejor dicho, pseudo-aleatorios

---

Seteando una semilla todos van a generar los mismos números pseudo-aleatorisos

```
x <- rpois(1, lambda=8)

if (x >= 10) {
  print("x es más grande o igual que 10")
} else if (x > 5) {
  print("x es mayor que 5")
} else {
```

```
  print("x es menor que 5")
}
```

```
[1] "x es mayor que 5"
```

Tip: Números pseudo-aleatorios

La función `rpois()` genera números aleatorios que siguen una distribución de Poisson con media lambda, en el caso anterior 8. La función `set.seed()` garantiza que todas las máquinas van a generar exactamente el mismo número pseudo-aleatorio (más sobre números pseudo-aleatorios). Entonces si ponemos `set.seed(10)`, deberíamos obtener que `x` vale 8.

Ejercicio 1

Usa una declaración `if()` para imprimir un mensaje que diga si hay algún registro del año 2002 en el dataset `gapminder`. Haz lo mismo para 2012

¿Alguien obtuvo un mensaje de advertencia?

```
Warning in if (gapminder$year == 2012) {: the condition has length > 1 and
only the first element will be used
```

Si tu condición evalua un vector con más de un elemento, `if()` va a funcionar igual, pero solo evaluará la condición del primer elemento. Necesitas asegurarte que la condición es de longitud 1.

Tip: any() y all()

La función `any()` es como un *o lógico* que opera sobre un vector. Con que haya al menos un valor `TRUE` la función devuelve `TRUE` y `FALSE` en otro caso. La función `all()` es un *y lógico* que opera sobre un vector y devuelve `TRUE` solo si todos los valores son `TRUE`

Repitiendo operaciones

Para itererar sobre un conjunto de valores, cuando el orden cada operación es importante usamos un loop `for()`.

Es el método más flexible para realizar iteraciones y por eso el más complicado de usar.

Evita usar `for()` a menos que el orden da la iteración sea importante: por ejemplo el cálculo de la iteración depende de la anterior.

La estructura básica de un bucle `for()` es:

```
for(iterador in conjunto de valores){
  hacer algo
}
```

Por ejemplo:

```
for(i in 1:10){
  print(i)
}
```

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

La parte de `1:10` crea un vector al vuelo; es posible iterrar sobre cualquier otro vector.

Podemos usar un bucle `for()` anidado dentro otro bucle `for()` para iterar sobre dos cosas a la vez.

```
for(i in 1:5){
  for(j in c('a', 'b', 'c', 'd', 'e')){
    print(paste(i,j))
  }
}
```

```
[1] "1 a"
[1] "1 b"
[1] "1 c"
[1] "1 d"
[1] "1 e"
[1] "2 a"
[1] "2 b"
[1] "2 c"
[1] "2 d"
[1] "2 e"
[1] "3 a"
[1] "3 b"
[1] "3 c"
[1] "3 d"
[1] "3 e"
[1] "4 a"
[1] "4 b"
[1] "4 c"
[1] "4 d"
[1] "4 e"
[1] "5 a"
[1] "5 b"
[1] "5 c"
[1] "5 d"
[1] "5 e"
```

En vez de imprimir los resultados, los podemos escribir a un objeto.

```r
vector_salida <- c()
for (i in 1:5){
  for (j in c('a', 'b', 'c', 'd', 'e')){
    salida_temporal <- paste(i, j)
    vector_salida <- c(vector_salida, salida_temporal)
  }
}
```

```r
vector_salida
```

```
 [1] "1 a" "1 b" "1 c" "1 d" "1 e" "2 a" "2 b" "2 c" "2 d" "2 e" "3 a"
[12] "3 b" "3 c" "3 d" "3 e" "4 a" "4 b" "4 c" "4 d" "4 e" "5 a" "5 b"
[23] "5 c" "5 d" "5 e"
```

Esta forma puede ser útil, pero 'hacer crecer los objetos' (construirlos de manera incremental) es ineficiente computacionalmente, por eso hay que evitarlo cuando iteras muchos valores.

Tip: no hagas crecer tus resultados

Es muy lento hacer crecer los objectos durante un loop porque R tiene que copiar el objeto viejo, sumarle los nuevos datos y volver a escribirlo.
En vez de eso es mejor determinar el tamaño que tendrá la salida, y crear un objeto vacío de esa longitud. Y luego guardar los resultados en el lugar correcto.

Una mejor manera es definir un objeto vacío para que contenga los resultados. En este ejemplo, se ve más complicado, pero es más eficiente.

```r
matrix_salida <- matrix(nrow = 5, ncol = 5)
vector_j <- c('a', 'b', 'c', 'd', 'e')
for (i in 1:5) {
  for (j in 1:5) {
    valor_j_temp <- vector_j[j]
    salida_temp <- paste(i, valor_j_temp)
    matrix_salida[i, j] <- salida_temp
  }
}
vector_salida2 <- as.vector(matrix_salida)
```

```r
vector_salida2
```

```
 [1] "1 a" "2 a" "3 a" "4 a" "5 a" "1 b" "2 b" "3 b" "4 b" "5 b" "1 c"
[12] "2 c" "3 c" "4 c" "5 c" "1 d" "2 d" "3 d" "4 d" "5 d" "1 e" "2 e"
[23] "3 e" "4 e" "5 e"
```

Tip: Bucles while

A veces vas a encontrar que necesitas repetir una operación hasta que se cumpla cierta condición. Podes hacer esto con un bucle `while()`.

```
while (esta condición sea verdad){
  haz algo
}
```

Tip: Bucles while

Como ejemplo, acá hay un bucle `while()` que genera números aleatorios de distribución uniforme (la función `runif()`) entre 0 y 1 hasta que obtiene uno menor a 0.1.

```
z <- 1
while (z > 0.1){
  z <- runif(1)
  print(z)
}
```

Los bucles `while()` pueden no ser apropiados siempre. Hay que ser muy cuidadoso de que no termines un bucle infinto porque tu condición nunca será falsa.

Ejercicio 2

Compara los objetos vector_salida y el vector_salida2 ¿Son iguales? Si no lo son ¿Por qué? ¿Como podés cambiar el último bloque de código para que la vector_salida2 sea igual a vector_salida?

Ejercicio 3

Escribe un script que itere sobre los datos de `gapminder` por continente e imprima si la expectativa de vida promedio es menor o mayor a 50 años.

Ejercicio 4

Modifica el script del ejercicio anterior para que también itere sobre cada pais. Esta vez debe imprimir si la expectativa de vida es menor que 50, entre 50 y 70 o mayor que 70.

Ejercicio 5 - Avanzado

Escribe un script que itere sobre cada país del set de datos `gapminder` y pruebe si el nombre del país empieza con 'B' y grafique la expectativa de vida versus el tiempo como una gráfico de lineas si la media de la expectativa de vida es menor a 50 años.