Ejercicios Clase 1

Marília Melo Favalesso

27/08/2021

• • Estructura de repetición • •

1. Elija las opciones correctas para if(){}else{}:

☐ Sirve para repetir uno o más comandos varias veces.
☒ Es una estructura de selección.
☐ Sirve para ejecutar algún comando solo si se satisface alguna condición.
☒ Podemos usar `else if()` para poner otras condiciones.

2. ¿Qué necesitas escribir en los cuadrados naranjas para que la función funcione?

```
# 1. niveles de glicose en pacientes:
glicose <- sample(70:130, 4)
glicose

for(          in glicose){
  if(         <= 99){print("normal")}
  else if(          >= 126){"diabetes"}
  else{print("prediabetes")}
}
```

☐ Un carácter que representa cada valor en "diabetes".

☐ Nada.

☒ Un carácter que representa cada valor en "glicose".

3. Tomando como entrada la altura y el sexo de una persona, calcule e imprima el peso ideal:

```
# Para femenino
Fem <- c(1.65, 1.72)

for(f in Fem){
  fideal <- ((62.1 * f) - 44.7)
  print(fideal)
}
```

```
## [1] 57.765
## [1] 62.112
```

```r
# Para masculino
Masc <- c(1.78, 1.81)

for(m in Masc){
  mideal <- ((72.7 * m) - 58)
  print(mideal)
}
```

```
## [1] 71.406
## [1] 73.587
```

```r
datos <- matrix(c(1.65, 1.72, 1.78, 1.81), ncol = 4)
colnames(datos) <- c(1, 1, 2, 2)

if(colnames(datos) == "1"){
  m <- (62.1 * datos) - 44.7
  print(m)
} else {
  print((72.7 * datos) - 58)
}
```

```
##           1      1      2      2
## [1,] 57.765 62.112 65.838 67.701
```

4. Elija las opciones correctas para `while()`:

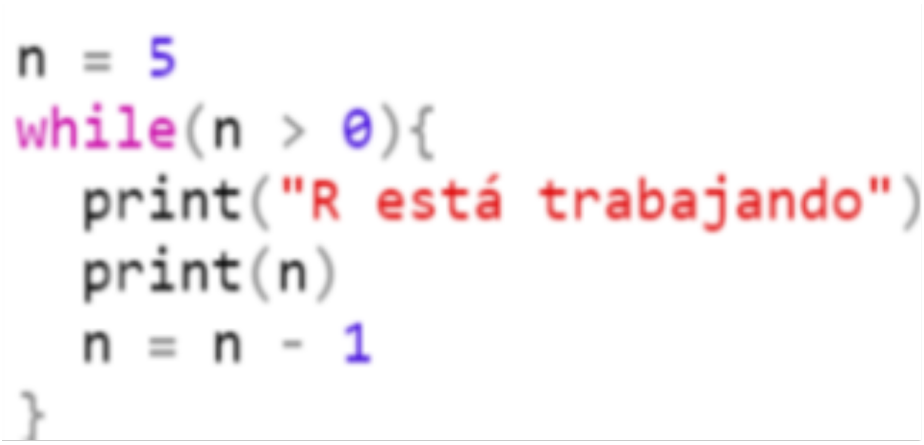
☐ Sirve para ejecutar algún comando solo si se satisface alguna condición.

☐ Es similar al loop `if(){}else{}`.

☒ Ejecuta algún código hasta que se cumpla una condición.

☒ Si bien la condición lógica es VERDADERA, el código no dejará de ejecutarse.

5. ¿Qué hace este loop while()?



☐ Siempre que “n” no sea 5, R no detendrá la ejecución de este ciclo.

☒ Siempre que “n” no sea 0, R no detendrá la ejecución de este ciclo.

☐ Resta 1 de “n” hasta obtener el valor de 5.

• • Funciones • •

6. ¿Qué son funciones en R?

☐ Son objetos que contienen valores.

☒ Son nombres que contienen un código R.

7. Dé uno o más ejemplos de funciones R:

   - `data.frame()`
   - `ls()`
   - `class()`
   - `rep()`
   - `sum()`

8. "Los argumentos de una función son los valores que encerramos entre paréntesis para que las funciones funcionen calculando un resultado a partir de eso".

☒ Totalmente de acuerdo

☐ Parcialmente de acuerdo

☐ No se

☐ Parcialmente en desacuerdo

☐ Muy en desacuerdo

9. "Para las funciones que toman más de un argumento, tenemos que separar los argumentos con puntos".

☐ Totalmente de acuerdo.

☐ Parcialmente de acuerdo.

☐ No se.

☐ Parcialmente en desacuerdo.

☒ Muy en desacuerdo.

10. Cuál es el resultado de la siguiente función: class(2, 3, 4, 5)?

☐ Numeric.

☒ Error in class(): 4 argumentos pasados a 'clase', que requiere 1.

☐ Integer.

11. ¿Cuál es la sintaxis para crear una función en R?

- [ ]

```
function <- f(a, b){
  j <- a + b
  j
}
```

- [x]

```
f <- function(a, b){
  j <- a + b
  j
}
```

- [ ]

```
f(a,b) <- function{
  j <- a + b
  j
}
```

• • Paquetes • •

12. ¿Qué es un paquete R?
    - Contiene una familia de funciones.
    - Un grupo de funciones preestabelecidades.
    - Conjunto de funciones que permite manipular y analisar datos, crear gráficos, etc.
13. Para instalar paquetes de CRAN, ¿qué comando usamos?

☐ packages.install("nombre_paquete")

☐ install.packages(nombre_paquete)

☒ install.packages("nombre_paquete")

14. Para instalar paquetes de GITHUB, ¿qué comando usamos?

☒ devtools::install_github("repo/nombre_paquete")

☐ devtools::install.github(repo/nombre_paquete)

15. ¿Cómo cargar el paquete para usarlo en R?

☐ library("nombre_paquete")

☒ library(nombre_paquete)

☐ library_R("nombre_paquete")

16. Es verdadero...

☒ `help("nombre_función")` es muy útil cuando necesitamos ayuda para comprender una función.

☒ Para citar los paquetes: `citation("nombre_paquete")`.

☐ R no proporciona una función para obtener la cita del programa.

• • home work! • •

1. Durante su doctorado, pasó 2 años en el campo, una vez a la semana. La duración de cada campo fue de ~ 4 horas. En total, ¿ fuerán cuántos días de campo? ¿Y cuantas horas? Utilice R para calcular los resultados y guardarlos en los respectivos objetos: días y horas. Considerando que cada mes tenga siempre 4 semanas

```
dias <- 4*24; dias
```

```
## [1] 96
```

```
horas <- dias*4; horas
```

```
## [1] 384
```

2. Cree una función que calcule el total de días de campo (=ejercicio 1) simplemente dando el número de días: total_dias(x). Utilice la función para calcular un total de días para 1 y 3 años de campo.

```r
total_dias <- function(anos){
  d <- anos*12*4
  d
}

total_dias(1)
```

```
## [1] 48
```

```r
total_dias(3)
```

```
## [1] 144
```

3. Cree un vector con tres especies de animales venenosos o tres vectores de enfermedades distintos y guárdelos en un objeto.

```r
sp_ven <- c('Micrurus corallinus', 'Apis mellifera', 'Lonomia obliqua'); sp_ven
```

```
## [1] "Micrurus corallinus" "Apis mellifera"      "Lonomia obliqua"
```

4. Cree una matriz de datos con valores enteros aleatorios entre 0 y 100. La matriz debe contener 3 filas y 3 columnas.

```r
maal <- matrix(sample(0:100, 9), ncol = 3); maal
```

```
##      [,1] [,2] [,3]
## [1,]   63   92    4
## [2,]   90   66    1
## [3,]   35   37    9
```

5. Cree un data.frame con los datos del ejercicio 2 y el ejercicio 3. Las especies animales deben estar en las filas.

```r
maal <- data.frame(sp_ven, maal); maal
```

```
##                sp_ven X1 X2 X3
## 1 Micrurus corallinus 63 92  4
## 2      Apis mellifera 90 66  1
## 3     Lonomia obliqua 35 37  9
```

6. Cambie el nombre de las columnas a: c(“animal/vector”, “2018”, “2019”, “2020”).

```r
colnames(maal) <- c("animal/vector", "2018", "2019", "2020"); maal
```

```
##         animal/vector 2018 2019 2020
## 1 Micrurus corallinus   63   92    4
## 2      Apis mellifera   90   66    1
## 3     Lonomia obliqua   35   37    9
```

7. Haga una loop ‘for’ para imprimir los números almacenados en el objeto a.

```r
a <- sample(0:200, 30)

for(i in a){
  print(i)
}
```

```
## [1] 161
## [1] 65
## [1] 188
## [1] 130
```

```
## [1] 121
## [1] 144
## [1] 87
## [1] 131
## [1] 166
## [1] 68
## [1] 64
## [1] 6
## [1] 69
## [1] 97
## [1] 74
## [1] 77
## [1] 107
## [1] 40
## [1] 36
## [1] 178
## [1] 2
## [1] 159
## [1] 179
## [1] 152
## [1] 101
## [1] 55
## [1] 125
## [1] 122
## [1] 28
## [1] 186
```

8. Haga un loop que imprime los valores del objeto "a" multiplicados por -2.

```
for(i in a){
  print(i*-2)
}
```

```
## [1] -322
## [1] -130
## [1] -376
## [1] -260
## [1] -242
## [1] -288
## [1] -174
## [1] -262
## [1] -332
## [1] -136
## [1] -128
## [1] -12
## [1] -138
## [1] -194
## [1] -148
## [1] -154
## [1] -214
## [1] -80
## [1] -72
## [1] -356
## [1] -4
## [1] -318
```

```
## [1] -358
## [1] -304
## [1] -202
## [1] -110
## [1] -250
## [1] -244
## [1] -56
## [1] -372
```

9. ¿Qué hacen “for”, “ifelse” y “while”?

- `for()` ejecuta un comando repetido para todos los elementos dento de un objeto (vector), `if(){}else{}` hacen agregan que ese comando se realice sólo en los elementos que tengan una condición determinada, pudiendo agregar más de una condición (con else), y `while()` aplica un comando a un grupo de datos repetidas veces terminando únicamente bajo una condición determinada por uno.
- `for()`: es una estructura de control, específicamente de repetición, que permite repetir uno o más comandos varias veces. Es decir, dado un conjunto de valores, cada elemento i contenido dentro del conjunto, R ejecutara el comando que le asignemos dentro de los corchetes, a este comando se lo llama code. `If()`: es una estructura de control, específicamente de selección, que nos permite ejecutar algún comando solo si este satisface una cierta condición. En este caso solo puede darse 2 condiciones en este tipo de estructura de control. `While()`: es otra estructura de repetición que permite ejecutar un código hasta que el mismo cumpla con una condición preestablecida. El número de condiciones que se le puede preestablecer es mayor a 2.

10. Instale el siguiente paquete: https://github.com/rstudio/rmarkdown.

```
install.packages("rmarkdown")
```