

# Funciones y estructuras de control

## Software Estadístico

Dra. Eva Romero Ramos

Dpto. Estadística e Investigación Operativa

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

# Funciones definidas por el usuario

Las funciones definidas por los usuarios tienen la siguiente estructura:

```
Nombre < – function (argumento1, argumento2, ...) {  
    expresiones  
    return  
}
```

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE  
CALL OR WHATSAPP. 689 45 44 70

# Ejemplo de función definida por el usuario

La siguiente función permite calcular el coeficiente de variación de una variable:

```
CV <- function (x) {  
  media <- mean(x)  
  DT <- sd(x)  
  CVar <- DT/media  
  return CVar  
}
```

Esta función se puede simplificar a:

```
CV <- function (x) {
```

**Cartagena99**

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

# Funciones definidas por el usuario

- Si una función solo tiene una línea de código se pueden suprimir las llaves.

**Ejemplo.-** `CV < - function(x) sd(x)/mean(x)`

- Se pueden definir valores por defecto para los parámetros de la función.

**Ejemplo.-** `CV < -function(x = c(1,2,3)) sd(x)/mean(x)`

- Podemos crear las funciones dentro del Script con el que estemos trabajando o en uno independiente.
- Si creamos las funciones en un Script independiente podemos llamarlas usando el comando `source`.

Ejemplo `source("file.R")`

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

# Estructuras de control

- Las estructuras de control nos permiten organizar y gestionar la forma en que se ejecuta el código.
- Estas estructuras permiten establecer condiciones en el código o agrupar expresiones que queramos repetir un número determinado de veces.
- Las estructuras de control más conocidas son: if, else, for, while, repeat, break y next.

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

- Los comandos **if** y **else** se utilizan para establecer condiciones.
- Para establecer condiciones con **if** se utiliza la siguiente estructura:

```
if Condición {  
    expresiones  
}
```

- Si la condición se cumple, su valor será TRUE y se ejecutarán las expresiones dentro de las llaves. Si no se cumple no se ejecutarán.

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

- El comando **else** permite indicar las expresiones que se deben ejecutar si no se cumple la condición del **if**.
- La estructura es la siguiente:

```
if Condición {  
    expresiones if  
} else {  
    expresiones else  
}
```

- Si la condición se cumple, su valor será TRUE y se ejecutarán las expresiones if. Si no se cumple, su valor será FALSE y se ejecutarán las expresiones else.

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE  
CALL OR WHATSAPP. 689 45 44 70

## Ejemplo.- if - else

```
positivo_o_negativo <- function (numero) {  
  if (numero > 0) {  
    mensaje <- "El número es positivo."  
  } else if (numero < 0) {  
    mensaje <- "El número es negativo."  
  } else {  
    mensaje <- "El número es cero."  
  }  
  return(mensaje)  
}
```

Podemos probar la función con diferentes números:

```
positivo_o_negativo(3)
```

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

- Si queremos aplicar una condición a todos los elementos de un vector, podemos usar la estructura vectorizada **ifelse**.
- La estructura condicional **ifelse** se escribe así:  
`ifelse (condición, valor_si_verdadero, valor_si_falso)`
- **Ejemplo.-**  
`vector <- c(2, 5, 8, 4, 11, -12, -1, 6, -8, -2, 7, 10)`  
`ifelse(vector >= 0, 'Positivo', 'Negativo')`

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE  
CALL OR WHATSAPP. 689 45 44 70

# For

- La estructura **for** es una estructura de bucle que permite ejecutar un conjunto de expresiones un número determinado de veces.

- La estructura es la siguiente:

```
for (contador i Secuencia) {  
    expresiones  
}
```

- El contador es una variable a la que podemos llamar como queramos. Los nombres más habituales son: 'i','j','t',....
- La secuencia habitualmente es una secuencia de valores de 1 a n, contando de uno en uno donde n marca el número de veces

Cartagena99

CLASES PARTICULARES Y GRUPALES  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

## Ejemplo.- For

El siguiente código muestra en pantalla los primeros 10 números enteros:

```
for (i in 1:10) {  
    numero_impar <- 2 * i - 1  
    print(numero_impar)  
}
```

Otra forma:

```
secuencia <- seq(from = 1, to = 19, by = 2)  
for (i in secuencia){  
    print(i)
```

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

# While

- La estructura **while** también es una estructura de bucle.
- En este caso las expresiones dentro de la estructura se ejecutarán mientras se cumpla una condición.
- La estructura while es la siguiente:

```
while (condición) {  
    expresiones  
}
```

- A diferencia del bucle **for**, con el bucle **while** no necesitamos saber cuantas veces se ejecutará la condición para definir el bucle.

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

## Ejemplo.- While

Supongamos que queremos simular el crecimiento de una población de conejos, en la que cada pareja tiene 2 conejos cada 6 meses. Asumiremos que transcurridos 6 meses los conejos ya forman una nueva pareja.

```
poblacion_actual < - 4
objetivo < - 100
meses_transcurridos < - 0
while (poblacion_actual < objetivo) {
    poblacion_actual < - 2 * poblacion_actual
    meses_transcurridos < - meses_transcurridos + 6
}
```

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

# Comandos break and next

- Los comandos **break** and **next** se usan para modificar la ejecución de un bucle.
- El comando **break** se utiliza para interrumpir un bucle durante su ejecución, por ejemplo cuando se cumpla una condición.
- El comando **next** permite saltar a la siguiente iteración de un bucle, sin ejecutar las expresiones de la actual iteración.
- Ambos comandos se pueden ejecutar en los bucles: for, while o repeat.

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE  
CALL OR WHATSAPP. 689 45 44 70

## Ejemplo.- break

Incorporemos la opción **break** al ejemplo del crecimiento de la población de conejos para observar el resultado:

```
poblacion_actual < - 4
objetivo < - 100
meses_transcurridos < - 0
while (poblacion_actual < objetivo) {
    if (poblacion_actual == 32) {break}
    poblacion_actual < - 2 * poblacion_actual
    meses_transcurridos < - meses_transcurridos + 6
}
cat('Población alcanzada después de' meses_transcurridos
```

Cartagena99

CLASAS PARTICULARES TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

# Repeat

- La estructura **repeat** puede entenderse como el bucle en su form más libre.
- Se trata de un bucle que para terminar hace uso del comando **Break**.
- La estructura repeat es la siguiente:

```
repeat {  
    expresiones  
    if Condición {  
        expresiones  
    }  
}
```

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE  
CALL OR WHATSAPP. 689 45 44 70

## Ejemplo.- repeat

En este ejemplo vamos a generar números aleatorios hasta que salga un valor por encima de 0,8.

Para generar números aleatorios usaremos la función **runif** que genera valores aleatorios, todos con la misma probabilidad entre 0 y 1.

El bucle repeat es el siguiente:

```
repeat {  
  numero_aleatorio <- runif(1)  
  print(numero_aleatorio)  
  if (numero_aleatorio >= 0.8) {
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

# Funciones apply

- Las funciones **apply** son una familia de funciones que se utilizan para aplicar otras funciones a todos los elementos de un estructura de datos, que puede ser un vector, una matriz, un array, un data frame o una lista.
- Estas funciones permiten optimizar el código evitando los bucles que tradicionalmente recorren las estructuras para ejecutar una misma función sobre todos los datos.
- De este modo, se consigue **reducir el número de líneas** y se **incrementa la velocidad de ejecución**.
- Las funciones más utilizadas son **apply** y **lapply**.

Cartagena99

CLASES PARTICULARES, TUTORÍAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE  
CALL OR WHATSAPP. 689 45 44 70

# Función apply

- La función apply tiene 3 argumentos:
  - **X**: Un array o matriz.
  - **Margin**: Toma valores 1 o 2, 1 si queremos aplicar la función a las filas de la estructura y 2 si queremos aplicarla sobre las columnas.
  - **Fun**: La función que queremos aplicar a los datos de la estructura.
- **Ejemplo.-** Aplicaremos la función **nchar** para contar todos los caracteres del dat frame 'df\_personas'  
`apply(df_personas, 2, nchar)`

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SC  
CALL OR WHATSAPP. 689 45 44 70

# Función lapply

- La función **lapply** aplica cualquier función a los elementos de una lista.
- Los argumentos son:
  - **X**: Una lista u objeto que se pueda coercionar a una lista.
  - **Fun**: La función que queremos aplicar a los datos de la lista.
- **Ejemplo.-** Aplicaremos la función **nchar** para contar todos los caracteres de la lista:

```
lista <- list(nombre = 'Juan', edad = 21, calificaciones =  
c(9.5, 8.7, 7.8), aprobado = TRUE)  
lapply(lista, nchar)
```

Cartagena99

CLASES PARTICULARES TUTORÍAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE  
CALL OR WHATSAPP. 689 45 44 70