

Clase 01 - Introducción

Módulo Programación

Instituto del Cálculo

Algoritmo

Un **algoritmo** es una secuencia de **instrucciones**.

Ejemplo:

1. Moje el cabello,
2. Coloque champú,
3. Masajee suavemente y deje actuar por 2 min.,
4. Enjuague, y
5. Repita el procedimiento (desde 1.).

Algoritmo

Otro ejemplo:

Ingredientes: 15 huevos, 600 gramos de harina, 600 gramos de azúcar

Pasos:

1. Mientras no estén espumosos, batir los huevos junto con el azúcar,
2. agregar la harina en forma envolvente sin batir,
3. batir suavemente,
4. colocar en el horno a 180 grados,
5. si le clavo un cuchillo y sale húmedo, entonces ir a 4.
6. retirar del horno,
7. mientras no esté frío, esperar
8. desmoldar y servir,
9. fin.

Instrucción

Una **instrucción** es una operación que:

- **transforma los datos**, o bien
- **modifica el flujo de ejecución**.

Ejemplo:

1. **Mientras no estén espumosos**, **batir los huevos junto con el azúcar**,
2. **agregar la harina en forma envolvente sin batir**,
3. **batir suavemente**,
4. **colocar en el horno a 180 grados**,
5. **si le clavo un cuchillo y sale húmedo, entonces ir a 4.**
6. **retirar del horno**,
7. **mientras no esté frío, esperar**
8. **desmoldar y servir**,
9. **fin**.

Programa

Vamos a llamar programa a una implementación de un algoritmo en un lenguaje de programación.

```
# rm(list=ls())
require("Emcdf")
require("MASS")

# -----
empirica<-function(columna, punto)
{
  if (!is.null(dim(columna))){
    salida<-emcdf(columna,punto)
  } else {
    salida<-mean(columna<=punto)
  }
  salida
}

score_para_u <- function(u, datos, sobre.datos = TRUE, repeticiones=1000) {
  if (sobre.datos) {
    repeticiones <- dim(datos)[1]
  }
  diferencias<-rep(NA, repeticiones)
  d <- dim(datos)[2]
```

Datos

Los programas manipulan **valores** de diferentes **tipos**.

Ejemplos:

1. 153 es un valor de tipo **numérico** (*numeric*)
2. "Hola" es un valor de tipo **caracter** (*character*)
3. FALSE es un valor de tipo *bool* o **lógico** (*logical*)

Valores lógicos (o de verdad, o bool)

Denotan el resultado de una evaluación lógica: los valores “verdadero” (TRUE) y “falso” (FALSE).

Se puede operar con valores lógicos:

Operador	Operación
!	Negación
&	Conjunción
	Disyunción

Valores lógicos (o de verdad, o bool)

Ejemplo, negación de un valor lógico:

p	!p
TRUE	FALSE
FALSE	TRUE

Valores lógicos (o de verdad, o bool)

Ejemplo, conjunción de valores lógicos (Y lógico):

p	q	p & q
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

Valores lógicos (o de verdad, o bool)

Ejemplo, conjunción de valores lógicos (O lógico):

p	q	p q
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Valores numéricos

Operaciones con números

Operador	Operación	Ejemplo	Resultado
+	Adición	3+2	5
-	Substracción	8-6	2
*	Multiplicación	15*5	75
/	División	9/2	4.5
^	Potencia	2^3	8
%%	División entera	9%%2	4
%%	Resto de la división entera	9%%2	1

Valores numéricos

Comparaciones entre números

Operador	Operación	Ejemplo	Resultado
==	Igualdad	3==2	FALSE
!=	Distinto	3!=2	TRUE
<	Comparación por menor	4<5	TRUE
<=	Comparación por menor o igual	4<=5	TRUE
>	Comparación por mayor	4>5	FALSE
>=	Comparación por mayor o igual	5>=5	FALSE

Vectores

También se pueden concatenar distintos valores de un mismo tipo para tener un vector con la función `c(..., ...)`:

1. `c(31, 12, 53)` es un vector de 3 valores numéricos
2. `c("hola", "que", "tal", "8")` es un vector de 4 valores *character*
3. `c(TRUE, FALSE)` es un vector de 2 valores lógicos
4. `c(31, c(12, 53))` es un vector de 3 valores numéricos

Vectores operadores

Los vectores de valores se les pueden aplicar operadores de ese tipo.

1. `c(31, 12, 53)`
 - a. `c(31, 12, 53) + 5` nos da como resultado 36, 17, 58
 - b. `c(31, 12, 53) == 12` nos da como resultado FALSE, TRUE, FALSE
 - c. `c(31, 12, 53) %% 2` nos da como resultado 1, 0, 1
2. `c("hola", "que", "tal", "8")` es un vector de 4 valores *character*
 - a. `c("hola", "que", "tal", "8") == "que"` nos da como resultado FALSE, TRUE, FALSE, FALSE
 - b. `c("hola", "que", "tal", "8") > "p"` nos da como resultado FALSE, TRUE, TRUE, FALSE
3. `c(TRUE, FALSE)` es un vector de 2 valores lógicos
 - a. `!c(TRUE, FALSE)` nos da como resultado FALSE, TRUE
 - b. `c(TRUE, FALSE) | TRUE` nos da como resultado TRUE, TRUE

Vectores operaciones

Funciones interesantes de vectores:

- `length()` que da la longitud: Ej. `length(c(31, 12, 53))`
- `c(...)` para combinar elemento, pero también vectores: Ej. `c(c(2,3), c(4, 5))`
- `seq(...)` para crear vectores de secuencias. Ej. `seq(1, 5)`
- `:` similar al anterior. Ej. `1:5`

Vectores acceso

Podemos acceder a los elementos de un vector con `[]`:

- `c(31, 12, 53)[3]` nos da 53
- `c(31, 12, 53)[1]` nos da 31

Incluso, si dentro de los `[]` ponemos un vector con índices podemos obtener:

- `c(31, 12, 53)[c(1,3,2)]` nos da 31, 53, 12
- `c(31, 12, 53)[c(1,1,2,2,3,3)]` nos da 31, 31, 12, 12, 53, 53

También se puede usar como selector si dentro de `[]` tenemos valores lógicos (TRUE incluye, FALSE excluye)

- `c(31, 12, 53)[c(TRUE, FALSE, TRUE)]` nos da 31, 53
- `c(31, 12, 53)[c(31, 12, 53) > 17]` nos da 31, 53

Expresiones

Una **expresión** es una combinación de valores, variables y operadores. La **evaluación** de una expresión arroja como resultado un valor.

Ejemplos: ¿Qué valores resultan de evaluar estas expresiones?

- `1 + 1`: 2
- `(1>0) | (!(('a'<'b')))`: TRUE
- `nchar('hola') + 6`: 10
- `(5.6 > 2.0) & (nchar('hola') < 2)`: FALSE
- `length(c(3, 6, 1))`: 3
- `length(c("hola"))`: 1

Comentarios

Además de expresiones en lenguaje R, existe la posibilidad de tener texto para poder documentar, aclarar y ayudar a quién lea un programa en el futuro. R simplemente los ignora.

Se escribe desde el caracter `#`.

```
# estos es un comentario  
# el siguiente código calcula la suma de dos números  
3 + 2 # esto también es un comentario  
  
# este es el comentario del final
```

Idealmente se utilizan para expresar decisiones tomadas y realizar aclaraciones sobre un programa.

Variables

Una **variable** es un nombre que denota una posición en la **memoria** de la computadora en la que se almacena un valor.

En otras palabras, es como una caja en la que podemos colocar un valor y etiquetarla con un nombre declarativo.

Es posible guardar un valor en una variable, recuperarlo después, modificarlo y volver a guardarlo, etc. . .

Asignaciones

Nuestra primera instrucción

`VARIABLE <- EXPRESIÓN`

Almacena el valor de la `EXPRESIÓN` en una posición de la memoria de la computadora que identificaremos como `VARIABLE`

Ejemplos

- `x <- 10`
- `x <- x`
- `x <- y`
- `x <- x + y * 22 / 33`

Variable

Memoria, inicialmente vacía



`peso <- 6.5`

Programa

Un **programa** es una secuencia de **instrucciones**.

En particular,
¡una asignación es un programa!

Programa

Podemos combinar 2 programas, poniendo uno detrás de otro de manera *secuencial* y obtener un programa nuevo más largo que los 2 iniciales:

PROG1
PROG2

Se ejecuta primero **PROG1** y una vez que termina se ejecuta **PROG2**

Ejemplo

```
peso <- 1.5  
costo <- peso * 2  
costo <- costo + 3
```

Estado

Se denomina **estado** al valor de todas las variables de un programa en un punto de su ejecución, más la siguiente instrucción a ejecutar.

**Es una *foto* de la memoria
en un momento determinado.**

Estado

Ejemplo

Instrucciones del lenguaje de programación R

Estados

```
    peso = ?, costo = ?  
peso <- 1.5  
    peso = 1.5, costo = ?  
costo <- peso * 2  
    peso = 1.5, costo = 3  
costo <- costo + 3  
    peso = 1.5, costo = 6
```

Fin