Clase 02 - Condicionales, ciclos y gráficos

Módulo Programación

Instituto del Cálculo

Algoritmo

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

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.

Instrucción

Una **instrucción** es una operación que:

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

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**.

Programas y variables

Un **programa** es una implementación de un algoritmo en un lenguaje de programación.

Una **variable** es un nombre que denota una posición en la **memoria** de la computadora en la que se almacena un valor, que puede ser almacenado, leído, modificado.

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.

Asignación

VARIABLE <- EXPRESIÓN

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

Ejemplos

- `a <- 2`
- `b <- c(3, 6, 9)`
- `b[a] <- 7`
- `b[b[1]] <- 25 * b[a] + a * 5`
- `b[b[1]%/%520 + 1] <- 666`

Tarea: ¿cuántos valen a y b luego de ejecutar todas estas instrucciones?

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 de estados

Dadas dos variables, `perro` y `gato` escribir un programa que intercambie los valores de ambas variables.

perro = ?, gato = ?

```
perro <- 70
```

perro = 70, gato = ?

```
gato <- 15
```

perro = 70, gato = 15

```
perro <- perro + gato
```

perro = 70+15=85, gato = 15

```
gato <- perro - gato
```

perro = 85, gato = 85-15=70

```
perro <- perro - gato
```

perro = 85-70=15, gato = 70

perro = 15, gato = 70

Condicional

```
if (CONDICIÓN) {
      PROG
}
```

CONDICIÓN es una expresión que arroja un resultado *verdadero* o *falso*; y **PROG** es un programa.
PROG se ejecuta si y sólo si **CONDICIÓN** resulta un valor verdadero.

Ejemplo:

```
if (5 > 8) { print("El 5 es mayor que el 8")}

if (5 < 8) { print("El 5 es menor que el 8")}
```

```
## [1] "El 5 es menor que el 8"
```

Condicional - Ejemplo

```
perro <- 15

if (perro %% 4 == 3) {
  perro <- 72
}

perro
```

```
## [1] 72
```

Condicional

```
if (CONDICIÓN) {
        PROG1
} else {
        PROG2
}
```

CONDICIÓN es una expresión que arroja un resultado *verdadero* o *falso*;
PROG1 y **PROG2** son programas.
PROG1 se ejecuta sii **CONDICIÓN** resulta un valor **verdadero**.
PROG2 se ejecuta sii **CONDICIÓN** resulta un valor **falso**.

Condicional - Ejemplo

```
perro <- 15

if (perro * 2 != 30) {
  print("Condición verdadera")
  perro <- 72
} else {
  print("Condición falsa")
  perro <- 8
}
```

```
## [1] "Condición falsa"
```

```
perro
```

```
## [1] 8
```

Condicional

Podemos tener varios condicionales anidados para tener más casos.

```
if (CONDICIÓN1) {
        PROG1
} else if (CONDICIÓN2) {
        PROG2
} else if (CONDICIÓN3) {
        PROG3
} else {
        PROG4
}
```

Condicional - Ejemplo

```
numero <- 0
rojos <- c(1,3,5,7,9,12,14,16,18,19,21,23,25,27,30,32,34,36)
negros <- c(2,4,6,8,10,11,13,15,17,20,22,24,26,28,29,31,33,35)

salio_rojo <- any(rojos == numero)
salio_negro <- any(negros == numero)

if (salio_rojo) {
  print("¡Ganador el rojo!")
} else if (salio_negro) {
  print("¡Ganador el negro!")
} else {
  print("Perdieron! Salio verde!")
}
```

```
## [1] "Perdieron! Salio verde!"
```

Ciclos

```
while (CONDICIÓN) {
      PROG
}
```

CONDICIÓN es una expresión que arroja un resultado *verdadero* o *falso*; y **PROG** un programa.
PROG se ejecuta repetidas veces **mientras** **CONDICIÓN** arroje un valor **verdadero**.

Ciclos - Ejemplo

```
i <- 0
while (i < 4) {
  print(i)
  i <- i + 1
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
```

```
i
```

```
## [1] 4
```

Ciclos con programas más complejos

```
i <- 0
while (i < 4) {
  if (i %% 2 == 0) {
    print(paste(i, "es par"))
  } else {
    print(paste(i, "es impar"))
  }
  i <- i + 1
}
```

```
## [1] "0 es par"
## [1] "1 es impar"
## [1] "2 es par"
## [1] "3 es impar"
```

```
i
```

```
## [1] 4
```

Ciclos dentro de ciclos

```
fila <- 0
while (fila <= 5) {
  columna <- 1
  while (columna <= fila) {
    cat(columna)
    cat(" ")
    columna <- columna + 1
  }
  cat("\n")
  fila <- fila + 1
}
```

```
##
## 1
## 1 2
## 1 2 3
## 1 2 3 4
## 1 2 3 4 5
```

```
fila
```

```
## [1] 6
```

Ciclos con *for*

Se pueden realizar ciclos para recorrer vectores. Cada elemento del vector es asignado por turnos a una *variable*.

Ejemplos:

```
for (i in 0:9){
  print(i)
}

animales <- c('perro', 'gato', 'mamut')
for (i in animales){
  print(i)
}
```

Equivalencia de *for* a *while*

Los ciclos expresados con *for* pueden ser re-escritos en utilizando *while* (al revés no siempre se puede). Ejemplo:

```
animales <- c('perro', 'gato', 'mamut')
for (i in animales){
  print(i)
}
```

```
j <- 1
while (j <= length(animales)) {
  i <- animales[j]
  print(i)
  j <- j + 1
}
```

```
## [1] "perro"
## [1] "gato"
## [1] "mamut"
```

Programa

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

- Asignación
  - **VARIABLE <- EXPRESIÓN**
- Condicional
  - **if (CONDICIÓN) { PROG1 } else { PROG }**
- Ciclos
  - **while (CONDICIÓN) { PROG }**
  - **for (E en ELEMENTOS) { PROG }**

Combinando estructuras

Podemos anidar estructuras para lograr programas más complejos:

```
numeros <- 1:10

for (j in numeros) {
  if (j %% 2 == 0){
    print(j)
  }
}
```

```
## [1] 2
## [1] 4
## [1] 6
## [1] 8
## [1] 10
```

Combinando estructuras 2

Podemos anidar estructuras para lograr programas más complejos: Ejemplo:

```
animales <- c('perro', 'gato', 'mamut')

j <- 1
while (j <= length(animales)) {
  k <- 1
  while (k <= j) {
    algunos <- animales[j:k]
    print(algunos)
    k <- k + 1
  }
  j <- j + 1
}
```

```
## [1] "perro"
## [1] "gato"  "perro"
## [1] "gato"
## [1] "mamut" "gato"  "perro"
## [1] "mamut" "gato"
## [1] "mamut"
```

Sucesiones

Calculemos los primeros 10 términos de la sucesión $a_n = n^3 - n^2 + n$:

```
ies <- seq(1, 10, 1)
valores <- c()

for (i in ies){
  nuevo_valor <- i**3 - i**2 + i

  valores <- c(valores, nuevo_valor)
}
print(ies)
```

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

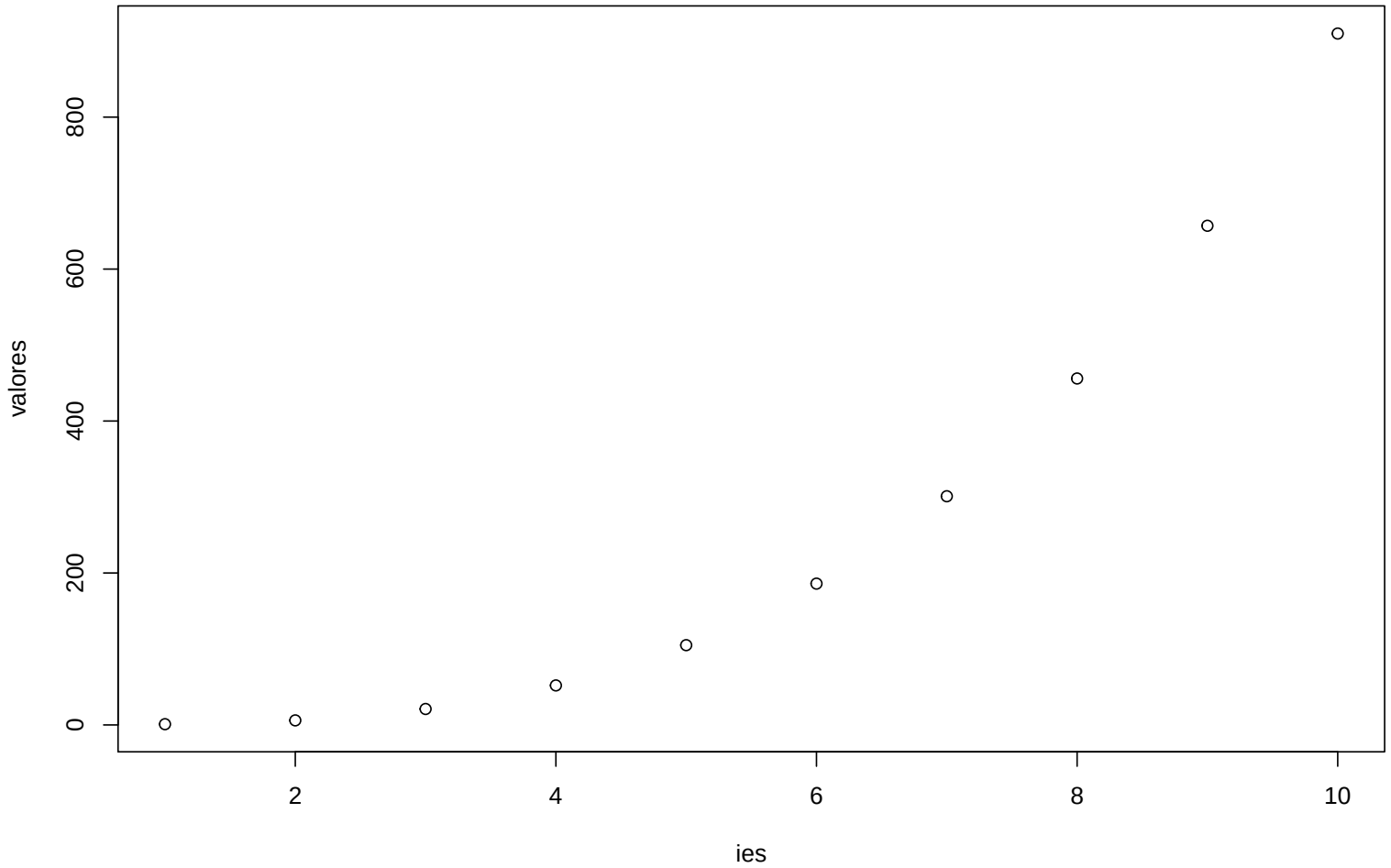
```
print(valores)
```

```
##  [1]   1   6  21  52 105 186 301 456 657 910
```

Sucesiones (continuación)

A su vez podemos graficar estos valores:

```
plot(ies, valores)
```



Plot

La función `plot` nos permite realizar gráficos tomando los valores de x e y. Además pueden agregarse más parámetros para poder darle mayores funcionalidades:

```
plot(x, y)
plot(n, valores)
plot(n, valores, type="b")
```

Tarea: explorar la ayuda de R sobre el comando plot:

```
?plot
```

Funciones

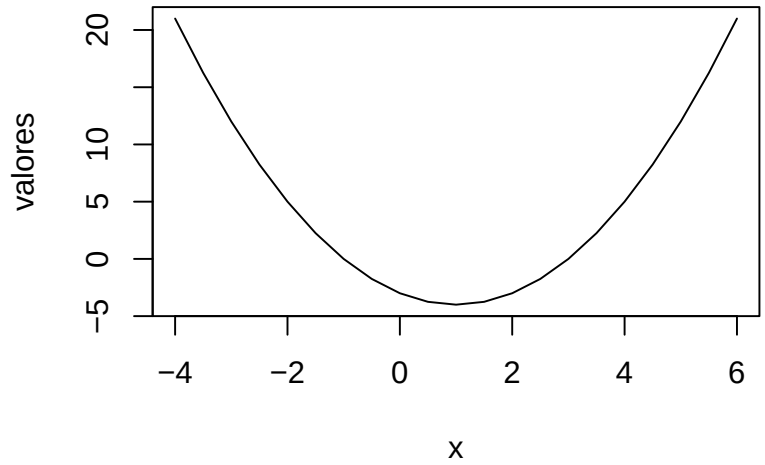
Podemos calcular el valor de una función (en el sentido matemático) en muchos puntos cercanos para poder aproximar su valor:

```
x <- seq(-4, 6, 0.5)
valores <- c()

for (i in x){
  nuevo_valor <- i**2 - 2 * i - 3

  valores <- c(valores, nuevo_valor)
}

plot(x, valores, type = "l") # explorar p, b en lugar de l
```

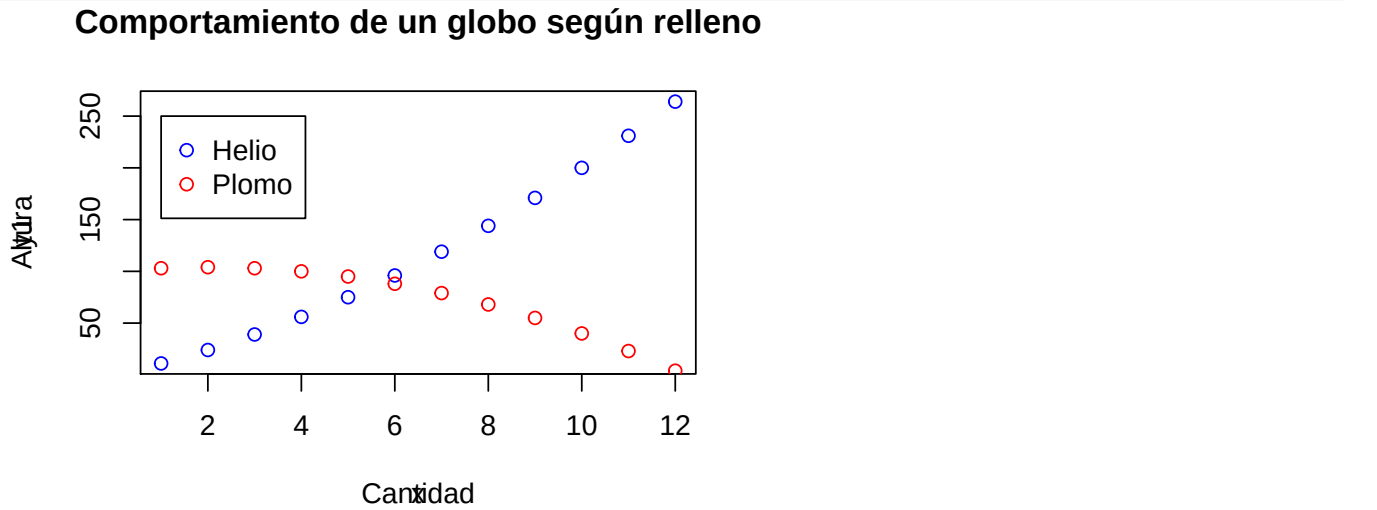


Graficando varias cosas

Podemos agregar a un gráficos más puntos (con la función `points(...)` o líneas con `lines(...)`) e incluso ponerle título y leyenda.

```
x <- 1:12
y1 <- x**2 + 10 * x
y2 <- -x**2 + 4*x + 100

plot(x,y1,col="blue") # crea un gráfico nuevo con x e y1
points(x,y2, col="red") # agrega los puntos x, y2 en rojo
title("Comportamiento de un globo según relleno", xlab = "Cantidad", ylab="Altura")
legend(1,250,legend=c("Helio","Plomo"), col=c("blue","red"), pch = 1)
```



Fin