PRÁCTICA 2: Bucles y funciones

En esta práctica introduciremos el tópico de matrices, bucles y funciones, además de profundizar en el apartado de vectores, mediante ejercicios basados en operaciones y representaciones. Con este documento se busca explicar los conceptos pertinentes a las prácticas de bucles y funciones conceptualizando cualquiera de los comandos usados y proporcionando consejos.

Conceptos

Bucles

Introducción

Los bucles son una herramienta que nos permite repetir un proceso un número definido de veces hasta que se cumpla una condición (bucles condicionales) o se alcance un determinado valor de repeticiones.

R

Para la realización de estos ejercicios emplearemos la función for(i in x:y).
Donde *i* es una variable que tomará los valores enteros definidos en el intervalo [x,y].
Es decir, si x=1 e y=3, i tomará el valor 1 en la primera repetición, 2 en la segunda y 3 en la tercera, cuando se detiene finalmente el bucle. Los procesos que queremos que se repitan aparecerán delimitados entre dos llaves tal que:

```
for(i in 1:4){
        a=a+b
        }
Ej:
> a=2
> b=3
> for(i in 1:4){
+       a=a+b
+ }
> a
[1] 14
```

Matrices

Introducción

Otra forma de almacenar datos son las matrices, que nos permiten organizar la información en filas y columnas.

R

1. Para generar una matriz utilizamos la función matrix().
2. Los límites de la matriz se definirán mediante las nrow=x y ncol=y, siendo x el número de filas e y el número de columnas.
3. Además, si quisiéramos componerla con los elementos de un vector deberíamos incluirlo en el paréntesis

Tal que:

```
> v=c(1,2,3,4)
> A=matrix(v,nrow=1,ncol=4)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
```

Nota
Si las dimensiones del vector exceden la longitud de una columna o fila sus componentes se distribuirán de forma lógica en la matriz:

```
> B=matrix(v,nrow=2,ncol=2)
> B
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Para acceder a un elemento añadimos los componentes que permitan identificar el elemento tal que A[i,j], siendo i el número de fila y j el número de la columna:

```
> A[1,3]
[1] 3
```

También se puede componer una matriz a base de vectores gracias a las estructuras rbind() en el que los componentes de los vectores se dispondrán en una fila de izquierda a derecha, y cbind() en el que los componentes se dispondrán en una columna de arriba abajo. Estos vectores deberán tener las mismas dimensiones.

Ej:

```
> a=c("Alberto","Paco","Pablo")
> b=c(1,2,3)
> c=c(4,5,6)
> C=cbind(a,b,c)
> C
          a          b     c
[1,] "Alberto" "1" "4"
[2,] "Paco"    "2" "5"
[3,] "Pablo"   "3" "6"
```

Las nuevas operaciones que añade el concepto de matriz son:

- t(A) : transpuesta de la matriz A.
- det(A) : determinante de la matriz A.
- solve(A,b) : solución del sistema de ecuaciones Ax=b.
- solve(A) : inversa de la matriz A.
- svd(A) : descomposición en valores singulares.
- qr(A) : descomposición QR.

eigen(A) : valores y vectores propios.
diag(b) : matriz diagonal (b es un vector).
diag(A) : matriz diagonal (A es una matriz).

Nota

Al igual que con los vectores el producto de dos matrices mediante *, lleva a cabo una operación no definida que realiza el producto componente por componente de las matrices.

Para llevar a cabo el producto de matrices utilizamos %*%.

Ej:

```
> A=matrix(c(1,2,1,2),ncol=2,nrow=2)
> B=matrix(c(2,1,2,1),ncol=2,nrow=2)
> A*B
     [,1] [,2]
[1,]    2    2
[2,]    2    2
> A%*%B
     [,1] [,2]
[1,]    3    3
[2,]    6    6
```

Funciones

Introducción

Las funciones nos permiten almacenar un determinado proceso que se repetirá siempre que se llame a la función, evitando la monotonía de escribir varias veces un mismo procedimiento en el que únicamente cambian sus variables.

R

Acoge la forma de f=function(x), donde x será una o varias variables empleadas durante el proceso que lleve a cabo la función.

Nota: es muy recomendable definir las funciones al principio de un script para evitar así posibles errores al llamar a una función no definida.

Ej:

```
> g=function(x){
+ b=x^2*cos(x^2)
+ a=10*x
+ return(b)
+ }
> c=8
> g(x=c)
[1] 25.07886
```

La función return(x), nos permitirá que la función nos devuelva el valor de una variable concreta calculada durante el proceso, en caso de hacerse uso de varias variables locales.

*Anexo
Variables locales: aquellas empleadas dentro de una función
Variables globales: se utilizan fuera de la función, en el script “general”
Es importante establecer una barrera entre ambos tipos de variables para evitar llamar a una variable local fuera de la función y así evitar posibles errores.

Ejercicios

Ejercicios con bucles

Ejercicio de vectores

1) Obtener la suma de los dos vectores mediante bucles y comprobar empleando v+u.

```
> suma=0
> v=c(12,-3,5,18.7)
> u=c(12,0.25,77,exp(2))
> sumaA=u+v
> for (i in 1:4){
+ suma[i]=u[i]+v[i]
+ }
> suma
[1] 24.00000 -2.75000 82.00000 26.08906
> sumaA
[1] 24.00000 -2.75000 82.00000 26.08906
```

Ya que i va a ir aumentando en 1 cada vez que haya una repetición, podemos aprovechar esto para que la suma del componente del vector u se sume con la que tiene la misma posición en el vector v. Y esta suma se registre en la componente de un vector suma, de los vectores.
Es un recurso muy útil que utilizaremos a lo largo de los ejercicios

2) Obtener la suma de las componentes del vector v y almacenarlos en SumaC.

```
> sumaC=0
> v=c(12,-3,5,18.7)
> for(i in 1:4){
+ sumaC=sumaC+v[i]
+ }
> sumaC
[1] 32.7
```

Todos los valores de las componentes del vector se irán sumando y registrándose en la variable sumaC

3) Realizar el producto escalar de ambos vectores mediante bucles y DESPUÉS comprobar empleando %*%.

```
> producto=c(0)
> v=c(12,-3,5,18.7)
> u=c(12,0.25,77,exp(2))
> R=u%*%v
> escalar=0
```

```
> for (i in 1:4){
+ producto[i]=u[i]*v[i]
+ escalar=escalar+producto[i]
+ }
> escalar
[1] 666.4253
> R
      [,1]
[1,] 666.4253
```

En el primer bucle ocurre:
producto[1]=12*12
escalar=0+12*12
En el segundo:
producto[2]=-3*0
escalar=144+0

4) Multiplicar ambos vectores componente a componente mediante bucles

```
> producto=c(0)
> v=c(12,-3,5,18.7)
> u=c(12,0.25,77,exp(2))
> for (i in 1:4){
+ producto[i]=u[i]*v[i]
+ }
> producto
[1] 144.0000  -0.7500 385.0000 138.1753
```

5) Realizar la operación: zj=vj+2uj , j = 1,…,length(v) mediante bucles

La función length() nos permite saber la longitud de un objeto. Resulta muy útil para saber el número de componentes de un vector

```
> z=c(0)
> v=c(12,-3,5,18.7)
> u=c(12,0.25,77,exp(2))
> for(j in 1:length(v)){
+ z[j]=u[j]+2*v[j]
+ }
> z
[1] 36.00000 -5.75000 87.00000 44.78906
```

6) Construir una tabla, data.frame que contenga:

'Suma'	*Resultado del apartado 2)*
'Producto escalar'	*Resultado del apartado 3)*

```
> sumaC=0
> producto=c(0)
> v=c(12,-3,5,18.7)
> u=c(12,0.25,77,exp(2))
> escalar=0
> for (i in 1:4){
+ sumaC=sumaC+v[i]
+ producto[i]=u[i]*v[i]
+ escalar=escalar+producto[i]
+ }
> nombres=c("suma","escalar")
```

```
> resultados=c(sumaC,escalar)
> tabla=data.frame(nombres,resultados)
> tabla
  nombres resultados
1    suma    32.7000
2 escalar   666.4253
```

Ejercicio de bucles anidados

1. Construir una matriz A1 de manera que los vectores v, w sean sus filas.

```
> A1=rbind(v,u)
> A1
  [,1]  [,2]    [,3]     [,4]
v   12 -3.00    5 18.700000
u   12  0.25   77  7.389056
```

Asigna los vectores a las filas

2. Construir una matriz A2 de manera que los vectores v, w sean sus columnas.

```
> A2=cbind(v,u)
> A2
      v        u
[1,] 12.0 12.000000
[2,] -3.0  0.250000
[3,]  5.0 77.000000
[4,] 18.7  7.389056
```

Asigna los vectores a las columnas

3. Multiplicar, empleando bucles, ambas matrices, obteniendo una matriz C

```
> v=c(12,-3,5,18.7)
> u=c(12,0.25,77,exp(2))
> suma=0
> A1=rbind(v,u)
> A2=cbind(v,u)
> C=matrix(c(0),nrow=nrow(A1),ncol=ncol(A2))
> for (i in 1:nrow(A1)){
+ for (j in 1:ncol(A2)){
+ for (k in 1:ncol(A1)){
+ C[i,j]=C[i,j]+A1[i,k]*A2[k,j]
+ }
+ }
+ }
> C
         [,1]             [,2]
[1,] 527.6900  666.4253
[2,] 666.4253 6127.6607
```

Estos bucles nos permitirán llevar a cabo el producto de matrices.

1º bucle:
i=1
j=1
k=1
C[1,1]=C[1,1]+A1[1,1]*A2[1,1]

2º bucle:
i=1
j=1
k=2
C[1,1]=C[1,1]+A1[1,2]*A2[2,1]

3º bucle
i=1
j=1
k=3
C[1,1]=C[1,1]+A1[1,3]*A2[3,1]

Queda definido el 1º componente de la matriz C

4. Verificar el resultado obtenido previamente empleando %*%

```
> v=c(12,-3,5,18.7)
> u=c(12,0.25,77,exp(2))
> suma=0
> A1=rbind(v,u)
> A2=cbind(v,u)
> A1%*%A2
         v         u
v 527.6900  666.4253
u 666.4253 6127.6607
```

5. Inventar una matriz 2x2 y llamarla D.

```
> D=matrix(ncol=2,nrow=2)
> D
     [,1] [,2]
[1,]   NA   NA
[2,]   NA   NA
```

6. Sumar, mediante bucles, las matrices C y D

```
> suma=matrix(0,ncol=2,nrow=2)
> A1=rbind(v,u)
> A2=cbind(v,u)
> D=matrix(0,ncol=2,nrow=2)
> for (i in 1:2){
+ for (j in 1:2){
+ suma[i,j]=C[i,j]+D[i,j]
+ }
+ }
> suma
         [,1]      [,2]
[1,] 527.6900  666.4253
[2,] 666.4253 6127.6607
```

7. Multiplicar las matrices C y D elemento a elemento.

```
> C*D
     [,1] [,2]
[1,]    0    0
[2,]    0    0
```

Ejercicio funciones

1. Define la función $f(x)=x^2*\cos(x^2)$ y obtén el valor f(pi/4)

```
> f=function(x){
+ x^2*cos(x^2)
+ }
```

2. Obtén un vector xx que tome 1001 valores en [0,10] (utilizando el comando seq con «salto» 0.01).

```
> xx=seq(1,10,0.01)
```

3. Representa gráficamente la función f, tomando como abscisas los valores xx. Etiqueta el eje de abscisas con 'Abscisa' y el eje de ordenadas con 'mis funciones f,g,h'. La gráfica irá en color verde.

```
> f=function(x){
+ x^2*cos(x^2)
+ }
> xx=seq(1,10,0.01)
> plot(xx,f(xx),col="green",xlab="abscisas",ylab="funciones f,g,h")
```

4. Define la función $g(x)=\sin(x^2)e^{-x2/10}$

```
> g=function(x){
+ z=sin(x^2)*e^(-x^2/10)
+ }
```

5. Utiliza la instrucción: par(new="true") para superponer curvas.

```
> f=function(x){
+ x^2*cos(x^2)
+ }
> g=function(x){
+ z=sin(x^2)*exp(1)^(-x^2/10)
+ }
> xx=seq(1,10,0.01)
> plot(xx,f(xx),col="green",xlab="abscisas",ylab="funciones f,g,h'")
> par(new="true")
> plot(xx,g(xx),col="blue")
```

La función par(new="true") nos permitirá añadir dentro de una misma representación varias funciones. Esta función se dispondrá entre las funciones plot() de ambas funciones

6. Dibuja la función g en los puntos xx en color azul y empleando: xlab=' ', ylab=' ',axes=FALSE (para eliminar ejes),pch=4 (para símbolos)

```
> f=function(x){
+ x^2*cos(x^2)
+ }
```

```
> g=function(x){
+ z=sin(x^2)*exp(1)^(-x^2/10)
+ }
> xx=seq(1,10,0.01)
> plot(xx,f(xx),col="green",xlab="abscisas",ylab="funciones f,g,h'")
> par(new="true")
> plot(xx,g(xx),col="blue",xlab="",ylab="",axes=FALSE,pch=4)
```

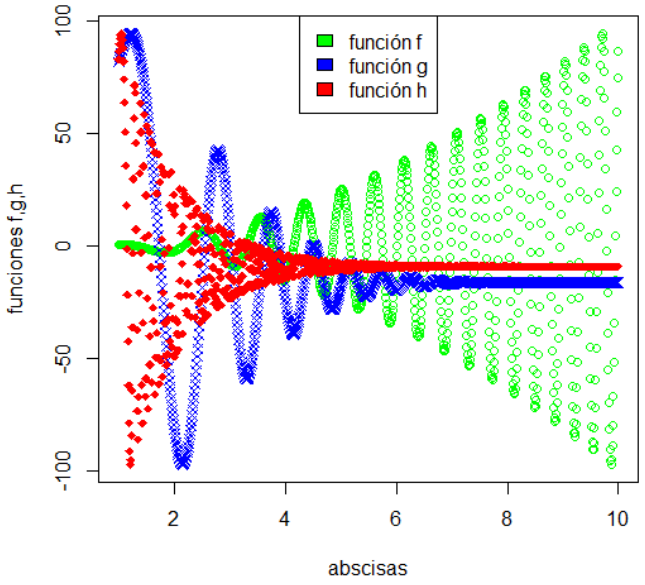
7. Idem a 5-6 con la función $h(x)=\sin(x^8)*e^{-x}$ en rojo y usando pch=18

```
> f=function(x){
+ x^2*cos(x^2)
+ }
> g=function(x){
+ sin(x^2)*exp(1)^(-x^2/10)
+ }
> h=function(x){
+ sin(x^8)*exp(1)^(-x)
+ }
> xx=seq(1,10,0.01)
> plot(xx,f(xx),col="green",xlab="abscisas",ylab="funciones f,g,h")
> par(new="true")2
> plot(xx,g(xx),col="blue",xlab="",ylab="",axes=FALSE,pch=4)
> par(new="true")
> plot(xx,h(xx),col="red",xlab="",ylab="",axes=FALSE,pch=18)
```

8. Añade a continuación la leyenda: legend(x = "top", c("función f", "función g", "función h"), fill = c("green", "blue", "red"))

```
> f=function(x){
+ x^2*cos(x^2)
+ }
> g=function(x){
+ sin(x^2)*exp(1)^(-x^2/10)
+ }
> h=function(x){
+ sin(x^8)*exp(1)^(-x)
+ }
> xx=seq(1,10,0.01)
> plot(xx,f(xx),col="green",xlab="abscisas",ylab="funciones f,g,h")
> par(new="true")
> plot(xx,g(xx),col="blue",xlab="",ylab="",axes=FALSE,pch=4)
> par(new="true")
> plot(xx,h(xx),col="red",xlab="",ylab="",axes=FALSE,pch=18)
> legend(x = "top", c("función f", "función g", "función h"), fill = c("green",
+ "blue", "red"))
```

Resultado:


función f
función g
función h
funciones f,g,h
abscisas
100
50
0
-50
-100
2
4
6
8
10