ANÁLISIS DE DATOS

Ricardo Aler Mur

EJERCICIOS Y SOLUCIONES A EJERCICIOS DE R

Ejercicio 1: VECTORIZACIÓN. Hacer un script que genere dos vectores:

- x=seq(1,10^6)
- y=x*5.2

Y los multiplique componente a componente de dos maneras:

- Con un bucle
- Solución vectorizada: x*y
- Medir tiempos y comparar

Solución:

```
## Vectores iniciales
x=seq(1,10^6)
y=x*5.2

t0 = proc.time()
z = rep(NA,length(x)) ## Inicialización de solucion

for(ind in 1:length(x)){
   z[ind] = x[ind] * y[ind]
 }
print(paste("Tiempo de la primera solución:", (proc.time()-t0)[1]))

t0 = proc.time()
```

```
z = x*y

print(paste("Tiempo de la segunda solución:", (proc.time()-t0)[1]))
```

Ejercicio 2: VECTORIZACIÓN E ÍNDICES NEGATIVOS Sea un vector x = seq(1,10^5).

Queremos calcular otro vector y de tal manera que:

y[i] = x[i]-x[i+1] para todo i de 1 a length(x)-1

Hacedlo de dos maneras distintas (con bucle y vectorizado) y pensad si se os ocurre alguna variante mas. Medid tiempos.

Regla: evitar usar bucles en la medida de lo posible

Solución:

```
x = seq(1,10^5)

end = length(x)


t0 = proc.time()

for(i in 1:(end-1)){

  y[i] = x[i]-x[i+1]

}

print(paste("Tiempo de la primera solución:", (proc.time()-t0)[1]))


t0 = proc.time()

y = x[1:(end-1)]-x[2:end]

print(paste("Tiempo de la segunda solución:", (proc.time()-t0)[1]))


t0 = proc.time()

y = x[-end]-x[-1]

print(paste("Tiempo de la tercera solución:", (proc.time()-t0)[1]))
```

Ejercicio 3: ACCESO/SUBSETTING: Crear un vector de 10 elementos así: x=sample(1:100, 10)

Hacer

1. Poner a cero los valores pares.
2. Poner a cero las **posiciones** pares. Hacedlo de **tres** maneras distintas
3. Escribir una función avg_gt con dos argumentos: x y gt (x es un vector y gt es un real). La función computa la media de los valores de x mas grandes que gt.

Solución:

```
## Ejercicio 3.1

x[x %% 2 == 0] = 0

## Ejercicio 3.2

print("PRIMERA SOLUCIÓN")

x=sample(1:100, 10)

print(x)

print(seq(2,10,by=2))

x[seq(2,10,by=2)] = 0

print(x)


print("SEGUNDA SOLUCIÓN")

x=sample(1:100, 10)

print(x)

print(1:10)

print(1:10 %% 2)

print(1:10 %% 2 == 0)

x[1:10 %% 2 == 0] = 0

print(x)


print("TERCERA SOLUCIÓN")

x=sample(1:100, 10)
```

```
print(x)

print(rep(c(FALSE,TRUE), 5))

x[rep(c(FALSE,TRUE), 5)] = 0

print(x)


## Ejercicio 3.3

avg_gt = function(x,gt) {mean(x[x>gt])}

print(avg_gt(sample(1:100,10),5))
```

Ejercicio 4: SUBSETTING EN MATRICES

1. Crear una matriz de 10x10 y poner a cero aquellas coordenadas (i,j) donde i es par y j es impar
2. Crear una matriz de 10x10 y poner a cero el rectángulo 3 a 5 (en la coordenada x) y de 5 a 8 (en la coordenada y)

Solución:

Nota, runif(n,a,b) crea n valores de una distribucion uniforme entre a y b (a=0 y b=1 por omision)

```
matriz = matrix(runif(10*10),10,10)

print(matriz)

matriz[seq(2,10,by=2), seq(1,9,by=2)]=0

print(matriz)


matriz = matrix(runif(10*10),10,10)

print(matriz)

matriz[3:5, 5:8]=0

print(matriz)
```

Ejercicio 5: FUNCIONES DE ALTO NIVEL.

1. Crear una lista con 5 vectores numéricos. Escribir una función que ordene cada vector (nota: usar la función sort()).
2. Escribir una función que compute el valor mínimo de cada columna de una matriz de cualquier tamaño (cualquier número de columnas). Probadla con una matriz aleatoria de 10x5

Solución:

Primera parte

lista = lapply(1:5, function(x) runif(5)) ## Crea una lista con cinco vectores aleatorios cada una. Se podria haber hecho tambien así:

lista = list(runif(5), runif(5), runif(5), runif(5), runif(5))

print(lista)

lapply(lista, sort) ## Ordenación creciente

lapply(lista, sort, decreasing=TRUE) ## Ordenacion decreciente

Segunda parte

matriz = matrix(round(runif(10*5,1,10)), 10, 5)

print(matriz)

minimos = function(m) {apply(m,2,min)}

minimos(matriz)