manuel.carmona@cunef.edu

Programación con R. Práctica 5

EJERCICIO 1

```
require(stats)
str(swiss)
```

```
## 'data.frame':    47 obs. of  6 variables:
##  $ Fertility       : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4
82.9 ...
##  $ Agriculture     : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3
45.2 ...
##  $ Examination     : int  15 6 5 12 17 9 16 14 12 16 ...
##  $ Education       : int  12 9 5 7 15 7 7 8 7 13 ...
##  $ Catholic        : num  9.96 84.84 93.4 33.77 5.16 ...
##  $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21
24.4 ...
```

```
head(swiss)
```

```
##              Fertility Agriculture Examination Education Catholic
## Courtelary        80.2        17.0          15        12     9.96
## Delemont          83.1        45.1           6         9    84.84
## Franches-Mnt      92.5        39.7           5         5    93.40
## Moutier           85.8        36.5          12         7    33.77
## Neuveville        76.9        43.5          17        15     5.16
## Porrentruy        76.1        35.3           9         7    90.57
##              Infant.Mortality
## Courtelary               22.2
## Delemont                 22.2
## Franches-Mnt             20.2
## Moutier                  20.3
## Neuveville               20.6
## Porrentruy               26.6
```

```
x <- swiss$Education[1:25]
x
```

```
##  [1] 12  9  5  7 15  7  7  8  7 13  6 12  7 12  5  2  8 28 20  9 10  3
12
## [24]  6  1
```

```
sort(x, method="sh",index.return = TRUE)
```

```
## $x
##  [1]  1  2  3  5  5  6  6  7  7  7  7  7  8  8  9  9 10 12 12 12 12 13
15
## [24] 20 28
##
## $ix
##  [1] 25 16 22  3 15 11 24  4  6  7  9 13  8 17  2 20 21  1 12 14 23 10
5
## [24] 19 18
```

```
#metodo sh ordena letras y numeros. Index return ($ix en la consola) me
da la posición de cada valor
x <- as.integer(rnorm(200, 5, 7))
sort(x, method="quick")                          #metodo quick ordena
numeros
```

```
##   [1] -13 -11 -10  -9  -9  -8  -7  -7  -6  -6  -6  -6  -5  -5  -5  -4
-4
##  [18]  -4  -4  -3  -3  -3  -3  -2  -2  -2  -2  -1  -1  -1  -1  -1  -1
0
##  [35]   0   0   0   0   0   0   0   0   0   0   0   1   1   1   1   1
1
##  [52]   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2
2
##  [69]   2   3   3   3   3   3   3   3   3   3   4   4   4   4   4   4
4
##  [86]   4   4   4   4   4   4   4   4   5   5   5   5   5   5   5   5
5
## [103]   5   5   5   5   5   6   6   6   6   6   6   6   6   6   6   6
7
## [120]   7   7   7   7   7   7   7   7   7   7   7   7   7   7   8   8
8
## [137]   8   8   8   8   8   8   8   8   8   8   9   9   9   9   9   9
9
## [154]   9   9   9   9  10  10  10  10  10  10  11  11  11  11  11  11
11
## [171]  11  11  11  11  12  12  12  13  13  13  14  14  14  14  15  15
15
## [188]  15  16  16  16  16  16  16  17  17  19  19  19  20
```

Donde se puede sustituir "quick" por "qu" o por "q". A veces es interesante realizar la ordenación solo de una parte de los datos, esto puede hacerse con: sort(x, partial = 1:5), en este caso, devuelve el vector con las cinco primeras posiciones del vector ordenado pero el resto sin ordenar. En selección directa, corta el bucle cuando llega al cinco

Como podemos observar, los valores aparecen ordenados de menor a mayor.

```
sort(x, partial = 1:5)
```

```
##   [1] -13 -11 -10  -9  -9  -8  -7  -5  -5  -4  -4  -5  -6  -6  -7  -6
-4
##  [18]  -6   3   1   2  -2   0  -2   2   1   3   2   2  -1   1   2   0
-3
##  [35]   1  -3   2   3   3   1   1   2  -1   1   0   3   0   2   1   1
-1
##  [52]   2  -3   0   0  -2   0  -2   3   0   0  -1   1   0   1   1   0
-4
##  [69]   1  -1   1   3   0  -3   1  -1   5   9  10   6  13   6   9   7
9
##  [86]   9   6   7   6   5  14   7   6   8   5   4   8   9  11  12  12
5
## [103]   7   6   7   7   7   4  11   8  11   7   4   3   7   4  11  11
9
## [120]  13   5  10   5   4   4   4   7  11   8   8  13  14   5   5  11
11
## [137]  11   8   8   4   4   4   6  10   9   6  15  14  10  14  10   4
8
## [154]   8  12   9   7   6   7  10   8  11   9   4   7   5   3   9   5
8
## [171]   7   5   8   6   5   6  11   9   4   5   4   4   7   5   8  16
15
## [188]  19  16  16  16  20  17  19  17  15  16  19  15  16
```

Con partial ordenamos solo una parte de los datos, en este caso devolviendo el vector con las cinco primeras posiciones del vector ordenado pero el resto sin ordenar. En selección directa, corta el bucle cuando llega al cinco

Ejercicio 2

calculamos las variables:

```
x<-as.integer(rnorm(100,42,(2.5**(1/2))))
x
```

```
##   [1] 40 42 42 41 43 41 38 42 41 43 40 43 42 43 42 42 43 40 41 41 42
44 41
##  [24] 39 40 43 42 41 41 43 46 41 42 43 42 41 42 42 40 43 45 43 40 41
41 42
##  [47] 43 40 39 38 43 42 38 40 41 39 40 41 42 44 44 41 41 41 40 42 40
44 41
##  [70] 39 42 43 42 39 41 41 42 42 43 44 39 43 40 43 41 40 40 41 42 42
42 41
##  [93] 42 39 40 40 43 39 39 42
```

```
y<-as.integer(rnorm(100,177,(10**(1/2))))
y
```

```
##   [1] 177 179 180 171 178 176 169 172 179 174 177 178 178 176 172 178
181
##  [18] 174 178 182 171 177 175 179 175 177 178 181 172 178 181 171 181
175
##  [35] 175 180 174 176 179 179 179 182 179 173 173 174 175 179 178 178
178
##  [52] 179 176 179 180 178 177 176 176 173 181 176 177 176 179 173 171
169
##  [69] 173 176 176 179 170 176 175 178 176 177 174 176 175 174 179 175
175
##  [86] 177 176 178 179 174 177 179 173 175 175 177 175 174 186 179
```

Metemos las variables en el dataframe datos

```
datos<-data.frame(x,y)
datos
```

```
##      x   y
## 1   40 177
## 2   42 179
## 3   42 180
## 4   41 171
## 5   43 178
## 6   41 176
## 7   38 169
## 8   42 172
## 9   41 179
## 10  43 174
## 11  40 177
## 12  43 178
## 13  42 178
## 14  43 176
## 15  42 172
## 16  42 178
## 17  43 181
## 18  40 174
## 19  41 178
## 20  41 182
## 21  42 171
## 22  44 177
## 23  41 175
## 24  39 179
## 25  40 175
## 26  43 177
## 27  42 178
## 28  41 181
## 29  41 172
## 30  43 178
## 31  46 181
## 32  41 171
```

```
## 33  42 181
## 34  43 175
## 35  42 175
## 36  41 180
## 37  42 174
## 38  42 176
## 39  40 179
## 40  43 179
## 41  45 179
## 42  43 182
## 43  40 179
## 44  41 173
## 45  41 173
## 46  42 174
## 47  43 175
## 48  40 179
## 49  39 178
## 50  38 178
## 51  43 178
## 52  42 179
## 53  38 176
## 54  40 179
## 55  41 180
## 56  39 178
## 57  40 177
## 58  41 176
## 59  42 176
## 60  44 173
## 61  44 181
## 62  41 176
## 63  41 177
## 64  41 176
## 65  40 179
## 66  42 173
## 67  40 171
## 68  44 169
## 69  41 173
## 70  39 176
## 71  42 176
## 72  43 179
## 73  42 170
## 74  39 176
## 75  41 175
## 76  41 178
## 77  42 176
## 78  42 177
## 79  43 174
## 80  44 176
## 81  39 175
```

```
## 82  43 174
## 83  40 179
## 84  43 175
## 85  41 175
## 86  40 177
## 87  40 176
## 88  41 178
## 89  42 179
## 90  42 174
## 91  42 177
## 92  41 179
## 93  42 173
## 94  39 175
## 95  40 175
## 96  40 177
## 97  43 175
## 98  39 174
## 99  39 186
## 100 42 179
```

Calculamos la media y la varianza

```
media<-lapply(datos,mean)
varianza<-lapply(datos,var)
media
```

```
## $x
## [1] 41.4
##
## $y
## [1] 176.45
```

```
varianza
```

```
## $x
## [1] 2.444444
##
## $y
## [1] 9.199495
```

Calculamos covarianza y correlacion

```
covar<-cov(datos$x,datos$y)
corr<-cor(x,y,method = "pearson")
covar
```

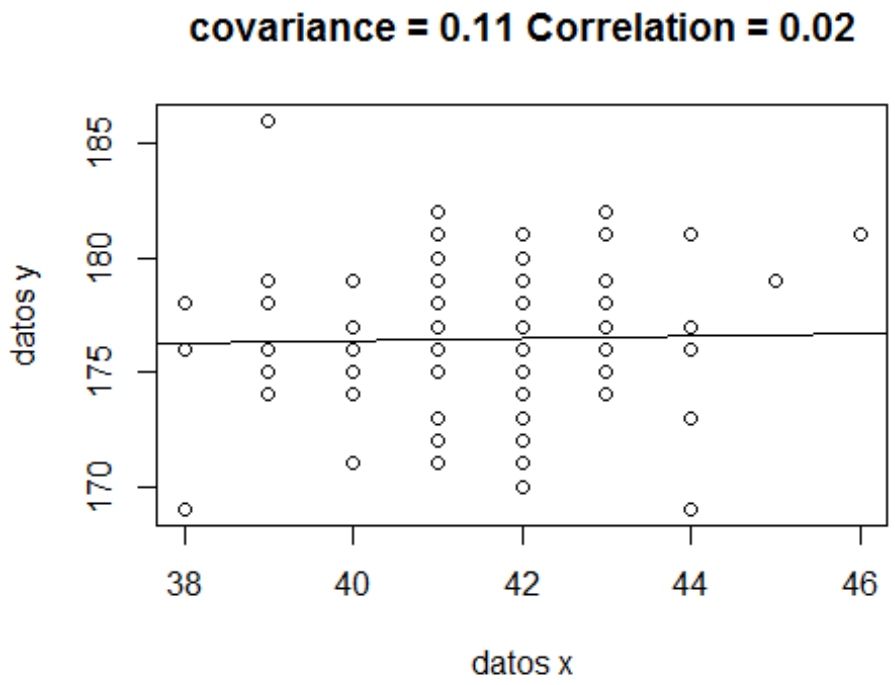
```
## [1] 0.1111111
```

```
corr
```

```
## [1] 0.02343071
```

Determinamos la regresion lineal entre variables y representamos los resultados

```
msg<-sprintf("covariance = %.2g Correlation = %.2f", covar, corr)
plot(datos$x,datos$y,main=msg,
     xlab="datos x",
     ylab="datos y")
lmPR4<-lm(y~x)
abline(lmPR4)
```



EJERCICIO 3

Calcular una matriz de k1 filas por k2 columnas con numeros aleatorios segun dist. de poisson de parametro lambda

```
k1<-3
k2<-5
lambda<-5
x<-rpois(k1*k2,lambda)
x

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

forma facil

```
mat<-matrix(x,nrow=k1,ncol=k2)
mat
```

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

```
mat2<-mat
```

forma con bucles

hay qye crear un objeto aleatorio porque si la pones vacia el bucle cuando va al elemento 1,1 por ejemplo, se encontraría que la posición no existe y me daría mal resultado.

Como vemos, el siguiente bucle completa la matriz incluyendo los elementos 1 a 1 por filas (K1):

```
for(i in 1:k1){
        for(j in 1:k2){
                mat2[i,j]<-x[k2*(i-1)+j]
        }
}
mat2
```

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

```
x
```

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

```
k2
```

```
## [1] 5
```

matrix hace la matriz y mete los numeros por columnas. Con el bucle rellena por filas. si quiero que lo coloque como en matrix seria:

completando por columnas:

Como vemos, en este bucle los elementos del bucle anterior correspondientes a K2 y K1 cambian su orden de entrada al bucle. Como definimos al principio, K2 corresponde a columnas y por eso, en este caso, empezamos completando por K2.

```
mat3<-mat
for(j in 1:k2){
        for(i in 1:k1){
                mat3[i,j]<-x[k1*(j-1)+i]    #si queremos que nos salga
```

```
como en matrix, habria que cambiar los bucles
        }
}
mat3
```

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

```
x
```

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

Calcular la matriz traspuesta:

1ºcreo una matriz de 0s con dimensiones k2 y k1

```
mattrasp<-matrix(0,k2,k1)
mattrasp
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
## [3,]    0    0    0
## [4,]    0    0    0
## [5,]    0    0    0
```

2º

```
for(i in 1:k2){
        for(j in 1:k1){
                mattrasp[i,j]<-mat2[j,i]}
}
```

Comprobamos los resultados utilizando la función t(x) de R y mirando mattrasp

```
trasp.mat2<-t(mat2)
mattrasp
```

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