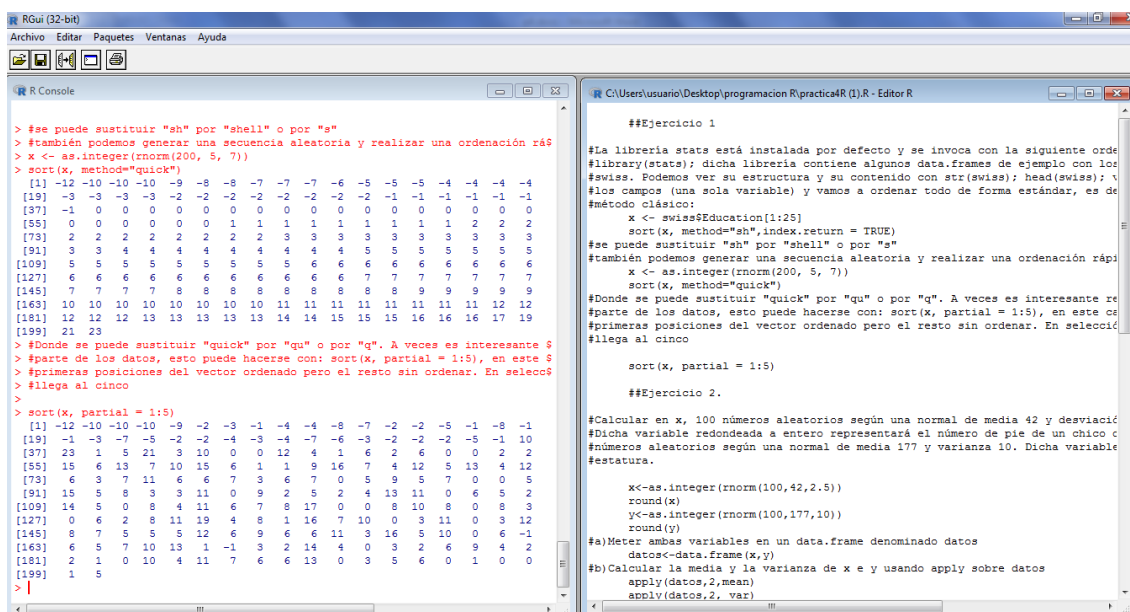


Programación con R. Práctica 4

Ejercicio 1

La librería stats está instalada por defecto y se invoca con la siguiente orden:

require(stats) o se instala con library(stats); dicha librería contiene algunos data.frames de ejemplo con los que poder trabajar, uno de ellos es swiss. Podemos ver su estructura y su contenido con str(swiss); head(swiss); vamos a coger los datos de uno de los campos (una sola variable) y vamos a ordenar todo de forma estándar, es decir, de menor a mayor por el método clásico: `x <- swiss$Education[1:25]` `sort(x, method="sh", index.return = TRUE)` se puede sustituir "sh" por "shell" o por "s" también podemos generar una secuencia aleatoria y realizar una ordenación rápida de los datos `x <- as.integer(rnorm(200, 5, 7))` `sort(x, method="quick")` 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



```
> #se puede sustituir "sh" por "shell" o por "s"
> #también podemos generar una secuencia aleatoria y realizar una ordenación rápida
> x <- as.integer(rnorm(200, 5, 7))
> sort(x, method="quick")
 [1] -12 -10 -10 -10 -9 -8 -8 -7 -7 -7 -6 -5 -5 -5 -4 -4 -4 -4
[19] -3 -3 -3 -3 -2 -2 -2 -2 -2 -2 -2 -2 -1 -1 -1 -1 -1 -1
[37] -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[55] 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2
[73] 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3
[91] 3 3 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5
[109] 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6
[127] 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7
[145] 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8
[163] 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 12
[181] 12 12 12 13 13 13 13 13 14 14 15 15 15 16 16 17 19
[199] 21 23
> #Donde se puede sustituir "quick" por "qu" o por "q". A veces es interesante re
> #parte de los datos, esto puede hacerse con: sort(x, partial = 1:5), en este c
> #primeras posiciones del vector ordenado pero el resto sin ordenar. En selecc
> #llega al cinco
>
> sort(x, partial = 1:5)
 [1] -12 -10 -10 -10 -9 -2 -3 -1 -4 -4 -8 -7 -2 -2 -5 -1 -8 -1
[19] -1 -3 -7 -5 -2 -2 -4 -3 -4 -7 -6 -3 -2 -2 -2 -5 -1 10
[37] 23 1 5 21 3 10 0 0 12 4 1 6 2 6 0 0 2 2
[55] 15 6 13 7 10 15 6 1 1 9 16 7 4 12 5 13 4 12
[73] 6 3 7 11 6 6 7 3 6 7 0 5 9 5 7 0 0 5
[91] 15 5 8 3 3 11 0 9 2 5 2 4 13 11 0 6 5 2
[109] 14 5 0 8 4 11 6 7 8 17 0 0 8 10 8 0 8 3
[127] 0 6 2 8 11 19 4 8 1 16 7 10 0 3 11 0 3 12
[145] 8 7 5 5 5 12 6 9 6 6 11 3 16 5 10 0 6 -1
[163] 6 5 7 10 13 1 -1 3 2 14 4 0 3 2 6 9 4 2
[181] 2 1 0 10 4 11 7 6 6 13 0 3 5 6 0 1 0 0
[199] 1 5
> |
```

```
##Ejercicio 1

#La librería stats está instalada por defecto y se invoca con la siguiente orde
#library(stats); dicha librería contiene algunos data.frames de ejemplo con los
#swiss. Podemos ver su estructura y su contenido con str(swiss); head(swiss); v
#los campos (una sola variable) y vamos a ordenar todo de forma estándar, es de
#método clásico:
x <- swiss$Education[1:25]
sort(x, method="sh", index.return = TRUE)
#se puede sustituir "sh" por "shell" o por "s"
#también podemos generar una secuencia aleatoria y realizar una ordenación rápida
x <- as.integer(rnorm(200, 5, 7))
sort(x, method="quick")
#Donde se puede sustituir "quick" por "qu" o por "q". A veces es interesante re
#parte de los datos, esto puede hacerse con: sort(x, partial = 1:5), en este ca
#primeras posiciones del vector ordenado pero el resto sin ordenar. En selecció
#llega al cinco

sort(x, partial = 1:5)

##Ejercicio 2.

#Calcular en x, 100 números aleatorios según una normal de media 42 y desviación
#Dicha variable redondeada a entero representará el número de pie de un chico c
#números aleatorios según una normal de media 177 y varianza 10. Dicha variable
#estatura.

x<-as.integer(rnorm(100,42,2.5))
round(x)
y<-as.integer(rnorm(100,177,10))
round(y)

#a)Meter ambas variables en un data.frame denominado datos
datos<-data.frame(x,y)
#b)Calcular la media y la varianza de x e y usando apply sobre datos
apply(datos,2,mean)
apply(datos,2,var)
```

Ejercicio 2.

Calcular en x, 100 números aleatorios según una normal de media 42 y desviación típica de 2.5. Dicha variable redondeada a entero representará el número de pie de un chico de 18 años. Calcular en y, 100 números aleatorios según una normal de media 177 y varianza 10. Dicha variable redondeada representa la estatura. a. Meter ambas variables en un data.frame denominado datos b. Calcular la media y la varianza de x e y usando apply sobre 'datos' c. Determinar la covarianza y la correlación, representar mediante un gráfico d. Determinar la regresión lineal entre las variables y representar los resultados.

```

R Console
> #Calcular en x, 100 números aleatorios según una normal de media 42 y desviación
> #Dicha variable redondeada a entero representará el número de pie de un chico
> #números aleatorios según una normal de media 177 y varianza 10. Dicha variable
> #estatura.
>
> x<-as.integer(rnorm(100,42,2.5))
> round(x)
[1] 46 42 44 43 36 43 40 41 38 41 39 42 43 36 46 43 42 46 39 38 42 39 42 44 46
[26] 44 44 41 45 42 39 39 40 42 41 41 43 41 41 45 48 42 48 43 42 45 41 36 40 42
[51] 41 43 40 44 44 43 42 38 38 41 39 42 49 43 43 41 42 43 38 48 43 45 42 46 40
[76] 41 37 40 41 44 39 40 41 41 35 43 42 47 37 37 41 37 41 44 42 39 40 38 42 43
> y<-as.integer(rnorm(100,177,10))
> round(y)
[1] 190 177 179 171 165 171 173 174 183 175 172 200 177 173 199 176 196 180
[19] 162 173 172 184 178 174 161 173 168 178 179 186 186 174 168 195 205 177
[37] 177 171 172 183 182 192 166 179 172 179 190 175 182 182 175 176 177 173
[55] 188 158 159 163 177 174 180 169 180 177 171 173 189 183 174 165
[73] 183 178 182 202 163 181 172 175 186 177 159 185 167 178 170 169 184 183
[91] 201 175 182 172 165 174 170 193 191 178
> #a) Meter ambas variables en un data.frame denominado datos
> datos<-data.frame(X=y)
> #b) Calcular la media y la varianza de x e y usando apply sobre datos
> apply(datos,2,mean)
      x      y
41.66 177.60
> apply(datos,2,var)
      x      y
8.145859 91.919192
> #c) Determinar la covarianza y la correlación, representar mediante un gráfico
> cov<-cov(x,y)
> cor<-cor(x,y)
> grafico<-plot(x,y)
> #d) Determinar la regresión lineal entre las variables y representar los resultados
> lml<-lm(y~x)
> abline(lml)
>

R Editor
##Ejercicio 2.
#Calcular en x, 100 números aleatorios según una normal de media 42 y desviación
#Dicha variable redondeada a entero representará el número de pie de un chico
#números aleatorios según una normal de media 177 y varianza 10. Dicha variable
#estatura.

x<-as.integer(rnorm(100,42,2.5))
round(x)
y<-as.integer(rnorm(100,177,10))
round(y)

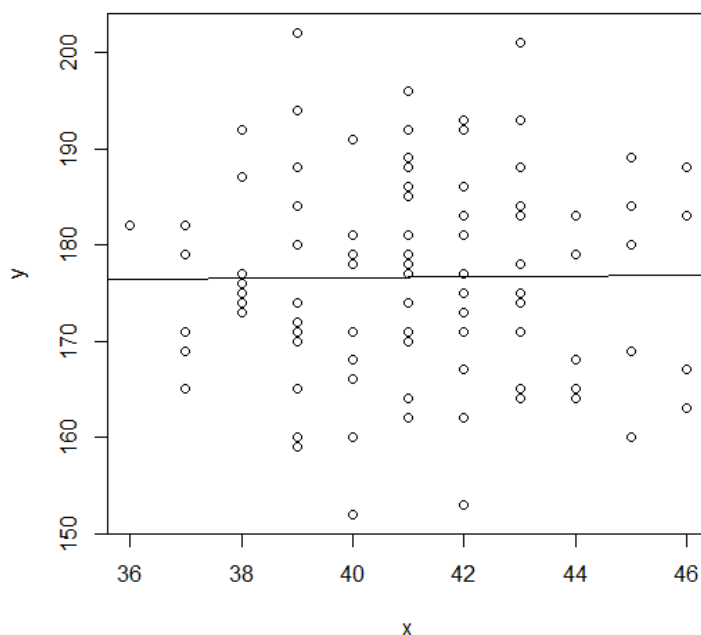
#a) Meter ambas variables en un data.frame denominado datos
datos<-data.frame(x,y)
#b) Calcular la media y la varianza de x e y usando apply sobre datos
apply(datos,2,mean)
apply(datos,2,var)

#c) Determinar la covarianza y la correlación, representar mediante un gráfico
cov<-cov(x,y)
cor<-cor(x,y)
grafico<-plot(x,y)
#d) Determinar la regresión lineal entre las variables y representar los resultados
lml<-lm(y~x)
abline(lml)

##Ejercicio 3
#Calcular una matriz de k1 filas por k2 columnas cuyas componentes sean números
#según una distribución de Poisson de parámetro Lambda. Calcular la traspuesta
#resultados usando la función t de R.

k1<-readline(prompt="introduce k1 ")
k1<-as.integer(k1)
k2<-readline(prompt="introduce k2 ")
k2<-as.integer(k2)
lambda<-readline(prompt="introduce lambda ")
lambda<-as.integer(lambda)

```



Ejercicio 3.

Calcular una matriz de k_1 filas por k_2 columnas cuyas componentes sean números aleatorios según una distribución de Poisson de parámetro λ . Calcular la traspuesta de dicha matriz y comprobar los resultados usando la función `t` de R. Observaciones. Los datos k_1 , k_2 y λ deben solicitarse por teclado al usuario del script generado. La implementación de todo el programa debe realizarse utilizando sentencias de iteración (`for`, `while`, `repeat`). La función `t` de R sirve para calcular la traspuesta de una matriz. El ejercicio debe comprobar que los resultados son los mismos que los que se obtendrían con esta función.

```
RGui (32-bit)
Archivo  Editar  Visualizar  Misc  Paquetes  Ventanas  Ayuda

R Console
> k1<-readline(prompt=("introduce  k1 "))
introduce  k1 3
> k1<-as.integer(k1)
> k2<-readline(prompt=("introduce  k2 "))
introduce  k2 5
> k2<-as.integer(k2)
> lambda<-readline(prompt=("introduce  lambda "))
introduce  lambda 3
> lambda<-as.integer(lambda)
>
> x<-rpois(k1*k2,lambda)
> #forma facil:matrix completa por columnas
> mat<-matrix(x, nrow=k1, ncol=k2);mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    3    2    4    0    1
[2,]    4    2    3    1    3
[3,]    3    6    5    0    6
> #forma de construir la matriz a partir de un vector:
> mat2<-mat;mat2
     [,1] [,2] [,3] [,4] [,5]
[1,]    3    2    4    0    1
[2,]    4    2    3    1    3
[3,]    3    6    5    0    6
> #completando por filas:
> for (i in 1:k1){
+   for (j in 1:k2){
+     mat2[i,j]<-x[k2*(i-1)+j]}
+ }
> #calcular la matrix traspuesta:
> mattrasp<-matrix(0,k2,k1)
> mat3<-mat2
> for (i in 1:k2){
+   for (j in 1:k1){
+     mattrasp[i,j]<-mat3[j,i]}
+ }
> transpuesta<-mattrasp;transpuesta
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
> mat2.transpose()
Error in mat2.transpose() :
  no se pudo encontrar la función "mat2.transpose"
> t(mat2)
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
> transpuesta<-mattrasp;transpuesta
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
> |

R C:\Users\usuario\Desktop\programacion R\practica4R (1).R - Editor R
#d) Determinar la regresion lineal entre las variables y representar los resultados
lm1<-lm(y~x)
abline(lm1)

#Ejercicio 3

#Calcular una matriz de k1 filas por k2 columnas cuyas componentes sean números
#según una distribución de Poisson de parámetro Lambda. Calcular la traspuesta
#resultados usando la función t de R.

k1<-readline(prompt=("introduce  k1 "))
k1<-as.integer(k1)
k2<-readline(prompt=("introduce  k2 "))
k2<-as.integer(k2)
lambda<-readline(prompt=("introduce  lambda "))
lambda<-as.integer(lambda)

x<-rpois(k1*k2,lambda)
#forma facil:matrix completa por columnas
mat<-matrix(x, nrow=k1, ncol=k2);mat
#forma de construir la matriz a partir de un vector:
mat2<-mat;mat2
#completando por filas:
for (i in 1:k1){
  for (j in 1:k2){
    mat2[i,j]<-x[k2*(i-1)+j]}
}

#calcular la matrix traspuesta:
mattrasp<-matrix(0,k2,k1)
mat3<-mat2
for (i in 1:k2){
  for (j in 1:k1){
    mattrasp[i,j]<-mat3[j,i]}
}

transpuesta<-mattrasp;transpuesta
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
t(mat2)
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
```

```
RGui (32-bit)
Archivo  Editar  Paquetes  Ventanas  Ayuda

R Console
> for (i in 1:k1){
+   for (j in 1:k2){
+     mat2[i,j]<-x[k2*(i-1)+j]}
+ }
> #calcular la matrix traspuesta:
> mattrasp<-matrix(0,k2,k1)
> mat3<-mat2
> for (i in 1:k2){
+   for (j in 1:k1){
+     mattrasp[i,j]<-mat3[j,i]}
+ }
> transpuesta<-mattrasp;transpuesta
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
> mat2.transpose()
Error in mat2.transpose() :
  no se pudo encontrar la función "mat2.transpose"
> t(mat2)
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
> transpuesta<-mattrasp;transpuesta
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
> |

R C:\Users\usuario\Desktop\programacion R\practica4R (1).R - Editor R
#d) Determinar la regresion lineal entre las variables y representar los resultados
lm1<-lm(y~x)
abline(lm1)

#Ejercicio 3

#Calcular una matriz de k1 filas por k2 columnas cuyas componentes sean números
#según una distribución de Poisson de parámetro Lambda. Calcular la traspuesta
#resultados usando la función t de R.

k1<-readline(prompt=("introduce  k1 "))
k1<-as.integer(k1)
k2<-readline(prompt=("introduce  k2 "))
k2<-as.integer(k2)
lambda<-readline(prompt=("introduce  lambda "))
lambda<-as.integer(lambda)

x<-rpois(k1*k2,lambda)
#forma facil:matrix completa por columnas
mat<-matrix(x, nrow=k1, ncol=k2);mat
#forma de construir la matriz a partir de un vector:
mat2<-mat;mat2
#completando por filas:
for (i in 1:k1){
  for (j in 1:k2){
    mat2[i,j]<-x[k2*(i-1)+j]}
}

#calcular la matrix traspuesta:
mattrasp<-matrix(0,k2,k1)
mat3<-mat2
for (i in 1:k2){
  for (j in 1:k1){
    mattrasp[i,j]<-mat3[j,i]}
}

transpuesta<-mattrasp;transpuesta
t(mat2)
     [,1] [,2] [,3]
[1,]    3    6    1
[2,]    4    4    0
[3,]    3    3    1
[4,]    2    5    3
[5,]    2    0    6
```