Introducción a R

Email: [delval@decsai.ugr.es](mailto:delval@decsai.ugr.es)

Alberto Armijo Ruiz

1. Matrices

\* Ejecuta los siguientes comandos.

matrix(data=5, nr=2, nc=2)

matrix(1:6, 2, 3)

matrix(1:6, 2, 3, byrow=TRUE)

> matrix(data=5, nr=2, nc=2)

[,1] [,2]

[1,] 5 5

[2,] 5 5

> matrix(1:6, 2, 3)

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

> matrix(1:6, 2, 3, byrow=TRUE)

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 4 5 6

El primer argumento determina un vector con datos que se van a introducir en la matriz, si se incluyen menos datos que filas y columnas se aplica la regla del reciclaje. Después se especifica el número de filas y el número de columnas; opcionalmente se puede elegir si se quiere introducir los datos por columnas o por filas.

\* Crea un vector z con los 30 primeros números y crea con el una matriz m con 3 filas y 10 columnas.

> z=1:30

> m=matrix(z,nrow=3,ncol=10)

> m

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

[1,] 1 4 7 10 13 16 19 22 25 28

[2,] 2 5 8 11 14 17 20 23 26 29

[3,] 3 6 9 12 15 18 21 24 27 30

\* Escribe la tercera columna en un vector

> vec=m[,3]

> vec

[1] 7 8 9

\* Create in R the matrices

x =3 2l

−1 1

y =1 4 0

0 1 -1

> m\_x = matrix(data=c(3,1,'2l',1),nrow = 2,ncol=2)

> m\_x

[,1] [,2]

[1,] "3" "2l"

[2,] "1" "1"

> m\_y = matrix(data = c(1,0,4,1,0,-1),nrow=2,ncol=3)

> m\_y

[,1] [,2] [,3]

[1,] 1 4 0

[2,] 0 1 -1

Y calcula los efectos de los siguientes comandos

(a) x[1,]

> m\_x[1,]

[1] "3" "2l"

(b) x[2,]

> m\_x[2,]

[1] "1" "1"

(c) x[,2]

> m\_x[,2]

[1] "2l" "1"

(d) y[1,2]

> m\_y[1,2]

[1] 4

(e) y[,2:3]

> m\_y[,2:3]

[,1] [,2]

[1,] 4 0

[2,] 1 -1

\* Transforma la matriz m que creaste en el ejercicio anterior en un array multidimensional. (Pista: averigua lo que puedas de la función dim().)

La función dim() devuelve un vector con las dimensiones del objeto, primero las filas, después las columnas, y el resto de dimensiones de un array.

> dim(m)

[1] 3 10

> m\_array = array(data=m,dim=dim(m))

> m\_array

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

[1,] 1 4 7 10 13 16 19 22 25 28

[2,] 2 5 8 11 14 17 20 23 26 29

[3,] 3 6 9 12 15 18 21 24 27 30

\* Crea un array de 5 x 5 x 2 y rellénalo con valores del 1 al 50. Investiga la función array(). Llama al array x

> x = array(1:50,dim=c(5,5,2))

> x

, , 1

[,1] [,2] [,3] [,4] [,5]

[1,] 1 6 11 16 21

[2,] 2 7 12 17 22

[3,] 3 8 13 18 23

[4,] 4 9 14 19 24

[5,] 5 10 15 20 25

, , 2

[,1] [,2] [,3] [,4] [,5]

[1,] 26 31 36 41 46

[2,] 27 32 37 42 47

[3,] 28 33 38 43 48

[4,] 29 34 39 44 49

[5,] 30 35 40 45 50

\* Dadas las matrices m1 y m2 usa rbind() y cbind() para crear matrices nuevas utilizando estas funciones, llamalas M1 y M2. ¿En que se diferencian las matrices creadas?

m1 <- matrix(1, nr = 2, nc = 2)

m2 <- matrix(2, nr = 2, nc = 2)

> m1 = matrix(1, nrow=2, ncol=2); m1

[,1] [,2]

[1,] 1 1

[2,] 1 1

> m2 = matrix(2, nrow=2, ncol=2); m2

[,1] [,2]

[1,] 2 2

[2,] 2 2

> M1 = rbind(m1,m2); M1

[,1] [,2]

[1,] 1 1

[2,] 1 1

[3,] 2 2

[4,] 2 2

> M2 = cbind(m1,m2); M2

[,1] [,2] [,3] [,4]

[1,] 1 1 2 2

[2,] 1 1 2 2

La diferencia entre la función cbind() y rbind() es como concatena las matrices, vectores, etc. La función cbind() concatena los objetos seleccionados por columnas, es decir, crea una matriz en la que primero están las columnas de la primera matriz y después las columnas de la segunda matriz; en cambio, la función rbind() concactena las filas de ambas matrices, primero las filas de la primera matriz y después las filas de la segunda matrix.

\* El operador para el producto de dos matrices es ‘ %\* %’. Por ejemplo, considerando las dos matrices creadas en el ejercicio anterior utilízalo.

> M12 = M1 %\*% M2; M12

[,1] [,2] [,3] [,4]

[1,] 2 2 4 4

[2,] 2 2 4 4

[3,] 4 4 8 8

[4,] 4 4 8 8

\* Usa la matriz M1 del ejercicio anterior y aplica la función t(). ¿qué hace esa función?

> t\_M1 = t(M1); t\_M1

[,1] [,2] [,3] [,4]

[1,] 1 1 2 2

[2,] 1 1 2 2

La función t() calcula la traspuesta de la matriz seleccionada.

\* Ejecuta los siguientes comandos basados en la función diag() sobre las matrices creadas anteriormente m1 y m2. ¿Qué tipo de acciones puedes ejecutar con ella?

La función diag() calcula la diagonal de una matriz, o construye una matriz diagonal. Dependiendo de la información que se le pase a la función, esta realiza diferentes funciones. Si se le pasa una matriz, la función extrae la diagonal de dicha matriz. Si se le pasa un entero, la función devuelve la matriz identidad de dimesiones de dicho número entero. Si se le pasa un vector, la matriz devuelve una matriz donde la diagonal son los elementos del vector. También se puede cambiar el valor de la diagonal asignandole un valor nuevo a dicha diagonal.

> diag(m1)

> diag(m1)

[1] 1 1

> diag(rbind(m1, m2) %\*% cbind(m1, m²))

> diag(rbind(m1,m2) %\*% cbind(m1,m2))

[1] 2 2 8 8

> diag(m1) <- 10

> diag(m1) = 10; diag(m1)

[1] 10 10

> diag(3)

> diag(3)

[,1] [,2] [,3]

[1,] 1 0 0

[2,] 0 1 0

[3,] 0 0 1

> v <- c(10, 20, 30)

> diag(v)

> v=c(10,20,30)

> diag(v)

[,1] [,2] [,3]

[1,] 10 0 0

[2,] 0 20 0

[3,] 0 0 30

> diag(2.1, nr = 3, nc = 5)

> diag(2.1, nrow=3, ncol=5)

[,1] [,2] [,3] [,4] [,5]

[1,] 2.1 0.0 0.0 0 0

[2,] 0.0 2.1 0.0 0 0

[3,] 0.0 0.0 2.1 0 0

\* Ordena la matriz x <- matrix(1:100, ncol=10):

a. en orden descendente por su segunda columna y asigna el resultado a una nueva matrix x1. Pista: función order()

> x1 = x

> x1 = x[order(x[,2],decreasing=T),]

> x1

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

[1,] 10 20 30 40 50 60 70 80 90 100

[2,] 9 19 29 39 49 59 69 79 89 99

[3,] 8 18 28 38 48 58 68 78 88 98

[4,] 7 17 27 37 47 57 67 77 87 97

[5,] 6 16 26 36 46 56 66 76 86 96

[6,] 5 15 25 35 45 55 65 75 85 95

[7,] 4 14 24 34 44 54 64 74 84 94

[8,] 3 13 23 33 43 53 63 73 83 93

[9,] 2 12 22 32 42 52 62 72 82 92

[10,] 1 11 21 31 41 51 61 71 81 91

b. en orden descendente por su segunda fila y asigna el resultado a una nueva matrix x2

> x2 = x

> x2 = x[order(x[2,],decreasing = T),]

> x2

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

[1,] 10 20 30 40 50 60 70 80 90 100

[2,] 9 19 29 39 49 59 69 79 89 99

[3,] 8 18 28 38 48 58 68 78 88 98

[4,] 7 17 27 37 47 57 67 77 87 97

[5,] 6 16 26 36 46 56 66 76 86 96

[6,] 5 15 25 35 45 55 65 75 85 95

[7,] 4 14 24 34 44 54 64 74 84 94

[8,] 3 13 23 33 43 53 63 73 83 93

[9,] 2 12 22 32 42 52 62 72 82 92

[10,] 1 11 21 31 41 51 61 71 81 91

c. Ordena solo la primera columna de x de forma descendentes

> x[,1] = x[order(x[,1],decreasing=T),1]

> x

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

[1,] 10 11 21 31 41 51 61 71 81 91

[2,] 9 12 22 32 42 52 62 72 82 92

[3,] 8 13 23 33 43 53 63 73 83 93

[4,] 7 14 24 34 44 54 64 74 84 94

[5,] 6 15 25 35 45 55 65 75 85 95

[6,] 5 16 26 36 46 56 66 76 86 96

[7,] 4 17 27 37 47 57 67 77 87 97

[8,] 3 18 28 38 48 58 68 78 88 98

[9,] 2 19 29 39 49 59 69 79 89 99

[10,] 1 20 30 40 50 60 70 80 90 100

\* Crea los siguientes vectores:

# Box office Star Wars: In Millions (!) First element: US, Second element:

# Non-US

new\_hope = c(460.998007, 314.4)

empire\_strikes = c(290.475067, 247.9)

return\_jedi = c(309.306177, 165.8)

Los datos se corresponden con las ventas en millones de la trilogía de la guerra de las galaxias. El primer numero corresponde a las ventas en US y el segundo al resto de países.

1. Construye la matriz star\_wars\_matrix con esos vectores

> star\_wars\_matrix = matrix(rbind(new\_hope,empire\_strikes,return\_jedi), ncol=2); star\_wars\_matrix

[,1] [,2]

[1,] 460.9980 314.4

[2,] 290.4751 247.9

[3,] 309.3062 165.8

1. Añádele nombres a las columnas y filas de la matriz según las descripciones dadas anteriormente de los datos

> colnames(star\_wars\_matrix)=c('US','Non-US')

> rownames(star\_wars\_matrix)=c('new\_hope','empire\_strikes','return\_jedi')

> star\_wars\_matrix

US Non-US

new\_hope 460.9980 314.4

empire\_strikes 290.4751 247.9

return\_jedi 309.3062 165.8

1. Calcula las ganacias mundiales de cada película y guardalas en un vector que se llame worldwide\_vector.

> worldwide\_vector=c(sum(star\_wars\_matrix[1,]),sum(star\_wars\_matrix[2,]),sum(star\_wars\_matrix[3,]))

> worldwide\_vector

[1] 775.3980 538.3751 475.1062

1. Añade éste ultimo vector como una columna nueva a la matriz star\_wars\_matrix y asigna el resultado a all\_wars\_matrix. Usa para ello la función [cbind()](http://www.rdocumentation.org/packages/base/functions/cbind).

> all\_wars\_matrix = cbind(star\_wars\_matrix,worldwide\_vector)

> colnames(all\_wars\_matrix)=c('US','Non-US','Total'); all\_wars\_matrix

US Non-US Total

new\_hope 460.9980 314.4 775.3980

empire\_strikes 290.4751 247.9 538.3751

return\_jedi 309.3062 165.8 475.1062

1. Calcula las ganancias totals en USA y fuera de USA para las tres películas. Puedes usar para ello la función [colSums()](http://www.rdocumentation.org/packages/base/functions/colSums)

> colSums(star\_wars\_matrix)

US Non-US

1060.779 728.100

1. Calcula la media de ganancias para todas las películas fuera de los estados unidos. Asigna esa media la variable non\_us\_all.

> non\_us\_all = mean(star\_wars\_matrix[,2]); non\_us\_all

[1] 242.7

1. Haz lo mismo pero solo par alas dos primeras películas . Asigna el resultado a la variable non\_us\_some.

> non\_us\_some = mean(star\_wars\_matrix[1:2,2]); non\_us\_some

[1] 281.15

1. Calcula cuantos visitantes hubo para cada película en cada área geográfica. Ya tienes las ganancias totales en star\_wars\_matrix. Asume que el precio de las entradas es de cinco euros/dólares (Nota: el numero total de visitantes para cada pelicula dividido por el precio del ticket te da el numero de visitantes)

> tickets\_sold\_us\_1 = star\_wars\_matrix[1,1]\*1000000/5 ; tickets\_sold\_us\_1

[1] 92199601

> ticket\_sold\_us\_2 = star\_wars\_matrix[2,1]\*1000000/5; ticket\_sold\_us\_2

[1] 58095013

> ticket\_sold\_us\_3 = star\_wars\_matrix[3,1]\*1000000/5; ticket\_sold\_us\_3

[1] 61861235

> tickets\_sold\_non\_us\_1 = star\_wars\_matrix[1,2]\*1000000/5; tickets\_sold\_non\_us\_1

[1] 62880000

> ticket\_sold\_non\_us\_2 = star\_wars\_matrix[2,2]\*1000000/5; ticket\_sold\_non\_us\_2

[1] 49580000

> ticket\_sold\_non\_us\_3 = star\_wars\_matrix[3,2]\*1000000/5; ticket\_sold\_non\_us\_3

[1] 33160000

1. Calcula la media de visitantes en territorio USA y en territorio noUS.

> mean\_visitors\_us = mean(c(ticket\_sold\_us\_2,tickets\_sold\_us\_1,ticket\_sold\_us\_3)); mean\_visitors\_us

[1] 70718617

> mean\_visitors\_non\_us = mean(c(ticket\_sold\_non\_us\_2,tickets\_sold\_non\_us\_1,ticket\_sold\_non\_us\_3)); mean\_visitors\_non\_us

[1] 48540000

1. Subsetting matrices y arrays

\* Como hemos visto en teoría la sintásis para acceder tanto a matrices como a arrays bidimiensionales es la siguiente.

array[rows, columns]

Muchas funciones de R necesitan una matriz como dato de entrada. Si algo no funciona recuerda convertir el objeto a una matriz con la función

as.matrix(iris)

\* Crea un arrayi <- array(c(1:10),dim=c(5,2)). ¿Que información te dan los siguientes comandos?

dim(i);

nrow(i);

ncol(i)

La función dim() devuelve las dimensiones del array, primero devuelve las filas, después devuelve las

columnas, si hay más dimensiones las devuelve también después de estas dos. La función nrow()

devuelve las filas de la matriz. La función ncol() devuelve las columnas de la matriz.

\* Crea un array de dimensiones 5 filas y dos columnas y rellénalo con valores del 1-5 y del 5 al 1

> arr<- array(c(1:5,5:1),dim=c(5,2))

> arr

[,1] [,2]

[1,] 1 5

[2,] 2 4

[3,] 3 3

[4,] 4 2

[5,] 5 1

\* ¿Qué hace el comando x[i] ¿. Comprueba que tienes en x antes

> x;x[i]

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

[1,] 10 11 21 31 41 51 61 71 81 91

[2,] 9 12 22 32 42 52 62 72 82 92

[3,] 8 13 23 33 43 53 63 73 83 93

[4,] 7 14 24 34 44 54 64 74 84 94

[5,] 6 15 25 35 45 55 65 75 85 95

[6,] 5 16 26 36 46 56 66 76 86 96

[7,] 4 17 27 37 47 57 67 77 87 97

[8,] 3 18 28 38 48 58 68 78 88 98

[9,] 2 19 29 39 49 59 69 79 89 99

[10,] 1 20 30 40 50 60 70 80 90 100

[1] 51 62 73 84 95

\* ¿y el comando x[i] <- 0?

> x[i] = 0;x

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

[1,] 10 11 21 31 41 0 61 71 81 91

[2,] 9 12 22 32 42 52 0 72 82 92

[3,] 8 13 23 33 43 53 63 0 83 93

[4,] 7 14 24 34 44 54 64 74 0 94

[5,] 6 15 25 35 45 55 65 75 85 0

[6,] 5 16 26 36 46 56 66 76 86 96

[7,] 4 17 27 37 47 57 67 77 87 97

[8,] 3 18 28 38 48 58 68 78 88 98

[9,] 2 19 29 39 49 59 69 79 89 99

[10,] 1 20 30 40 50 60 70 80 90 100

El comando asigna el valor 0 a las posiciones definidas por el array ‘i’.

\* Descárgate el fichero array\_datos.txt de PRADO (Datos/) e impórtalo en tu work space de R teniendo en cuenta que es un texto tabulado. Después crea un documento con los mismos datos pero en formato csv en vez de tab separated.

> arr\_tab = read.delim('array\_datos.txt',header = T)

> str(arr\_tab)

'data.frame': 3 obs. of 3 variables:

$ edad : int 20 22 19

$ peso : int 65 70 68

$ altura: int 174 180 170

> write.csv(arr\_tab,'salida\_datos.csv')

1. Factors

\* Dado x = c(1, 2, 3, 3, 5, 3, 2, 4, NA), ¿cuáles son los levels de factor(x)?

1. 1, 2, 3, 4, 5
2. NA
3. 1, 2, 3, 4, 5, NA

> factor(c(1,2,3,3,5,2,4,NA))

[1] 1 2 3 3 5 2 4 <NA>

Levels: 1 2 3 4 5

La respuesta correcta sería la a)1,2,3,4,5.

\* Dado  x <- c(11, 22, 47, 47, 11, 47, 11) y la ejecución de la sentencia

factor(x, levels=c(11, 22, 47), ordered=TRUE) ¿cuál es el cuarto elemento de la salida?

a. 11  
b. 22  
c. 47

> x <- c(11, 22, 47, 47, 11, 47, 11)

> factor(x,levels=c(11,22,47),ordered = T)

[1] 1 2 3 3 5 2 4 <NA>

Levels: 1 2 3 4 5

El cuarto elemento de la salida es el 47.

\* Para el factor  z <- c("p", "a" , "g", "t", "b"), reemplaza el tercer elemento de z por "b".

a. factor(z[3]) <- "b"  
b. levels(z[3]) <- "b"  
c. z[3] <- "b"

> z <- c("p", "a" , "g", "t", "b")

> z[3] <- "b"

> z

[1] "p" "a" "b" "t" "b"

La respuesta correcta sería la c. c) z[3]<-’b’

\* Dado z <- factor(c("p", "q", "p", "r", "q")) escribe una expresión de R que cambie el level "p" a "w"

> levels(z)

[1] "p" "q" "r"

> levels(z)[levels(z)=='p'] = 'w'

> z

[1] w q w r q

Levels: w q r

\* Usa el dataset “iris”

* escribe la expresión necesaria para convertir la variable “Sepal.Length” en un factor con cinco niveles (levels) . Pista( mira la función table() y la función cut().

> table(cut(iris$Sepal.Length,breaks = 5))

(4.3,5.02] (5.02,5.74] (5.74,6.46] (6.46,7.18] (7.18,7.9]

32 41 42 24 11

* escribe la expresión necesaria para generar una tabla de frecuencias con dos filas y tres columnas . Las filas deben referirse a si la variable “Sepal.length” es menor que 5 y las columnas a las diferentes expecies. El resultado debe ser:

> menor = table(iris[iris$Sepal.Length < 5,"Species"])

> mayor = table(iris[iris$Sepal.Length >=5, "Species"])

> tabla = rbind(menor, mayor)

> tabla

setosa versicolor virginica

menor 20 1 1

mayor 30 49 49

setosa versicolor virginica

FALSE 30 49 49

TRUE 20 1 1

\* El factor responses se define como:

responses <- factor(c("Agree", "Agree", "Strongly Agree", "Disagree", "Agree")),

sin embargo nos damos cuenta que tiene un nuevo nivel, "Strongly Disagree", que no estaba presente cuando se creó. Añade el nuevo nivel al factor y conviértelo en un factor ordenado de la siguiente forma:

Levels: Strongly Agree < Agree < Disagree < Strongly Disagree

> responses <- factor(c("Agree","Agree","Strongly Agree","Disagree","Agree"))

> responses

[1] Agree Agree Strongly Agree Disagree Agree

Levels: Agree Disagree Strongly Agree

> levels(responses) = c("Strongly Agree", "Agree", "Disagree", "Strongly Disagree")

> responses

[1] Strongly Agree Strongly Agree Disagree Agree Strongly Agree

Levels: Strongly Agree Agree Disagree Strongly Disagree

\* Dado el factor:

x <- factor(c("high", "low", "medium", "high", "high", "low", "medium"))

escribe la expresión en R que permita dar valores numéricos únicos para los distintos niveles (levels) de x según el siguiente esquema:

level high => value 1  
level low => value 2  
level medium => value 3

Pista: investiga la función unique() y los parámetros de data.frame()

> x <- factor(c("high", "low", "medium", "high", "high", "low", "medium"))

> x

[1] high low medium high high low medium

Levels: high low medium

> unique(x)

[1] high low medium

Levels: high low medium

> niveles = 1:length(unique(x))

> x\_numerico = x; levels(x\_numerico) = niveles; x\_numerico

[1] 1 2 3 1 1 2 3

Levels: 1 2 3

1. Acceso y selección de secciones de un data frames

La sintaxis general para acceder a un data frame es

my\_frame[rows, columns]

\* Vamos a trabajar con un ejemplo que viene por defecto en la instalación de R USArrests. Este data frame contiene la información para cada estado Americano de las tasas de criminales (por 100.000 habitantes). Los datos de las columnas se refieren a Asesinatos, violaciones y porcentaje de la población que vive en áreas urbanas. Los datos son de 1973. Contesta a las siguientes preguntas sobre los datos

- Las dimensiones del dataframe

> dim(USArrests)

[1] 50 4

- La longitud del dataframe (filas o columnas)

> nrow(USArrests)

[1] 50

- Numero de columnas

> ncol(USArrests)

[1] 4

- ¿Cómo calcularías el número de filas?

> nrow(USArrests)

[1] 50

- Obtén el nombre de las filas y las columnas para este data frame

> colnames(USArrests)

[1] "Murder" "Assault" "UrbanPop" "Rape"

> rownames(USArrests)

[1] "Alabama" "Alaska" "Arizona" "Arkansas" "California" "Colorado" "Connecticut"

[8] "Delaware" "Florida" "Georgia" "Hawaii" "Idaho" "Illinois" "Indiana"

[15] "Iowa" "Kansas" "Kentucky" "Louisiana" "Maine" "Maryland" "Massachusetts"

[22] "Michigan" "Minnesota" "Mississippi" "Missouri" "Montana" "Nebraska" "Nevada"

[29] "New Hampshire" "New Jersey" "New Mexico" "New York" "North Carolina" "North Dakota" "Ohio"

[36] "Oklahoma" "Oregon" "Pennsylvania" "Rhode Island" "South Carolina" "South Dakota" "Tennessee"

[43] "Texas" "Utah" "Vermont" "Virginia" "Washington" "West Virginia" "Wisconsin"

[50] "Wyoming"

- échale un vistazo a los datos, por ejemplo a las seis primeras filas

> USArrests[1:6,]

Murder Assault UrbanPop Rape

Alabama 13.2 236 58 21.2

Alaska 10.0 263 48 44.5

Arizona 8.1 294 80 31.0

Arkansas 8.8 190 50 19.5

California 9.0 276 91 40.6

Colorado 7.9 204 78 38.7

- Ordena de forma decreciente las filas de nuestro data frame según el porcentaje de población en el área urbana. Para ello investiga la función order () y sus parámetros.

> ord\_usarrests = USArrests[order(USArrests[,"UrbanPop"],decreasing = T),]

> ord\_usarrests

Murder Assault UrbanPop Rape

California 9.0 276 91 40.6

New Jersey 7.4 159 89 18.8

Rhode Island 3.4 174 87 8.3

New York 11.1 254 86 26.1

Massachusetts 4.4 149 85 16.3

Hawaii 5.3 46 83 20.2

Illinois 10.4 249 83 24.0

Nevada 12.2 252 81 46.0

Arizona 8.1 294 80 31.0

Florida 15.4 335 80 31.9

Texas 12.7 201 80 25.5

Utah 3.2 120 80 22.9

Colorado 7.9 204 78 38.7

Connecticut 3.3 110 77 11.1

Ohio 7.3 120 75 21.4

Michigan 12.1 255 74 35.1

Washington 4.0 145 73 26.2

Delaware 5.9 238 72 15.8

Pennsylvania 6.3 106 72 14.9

Missouri 9.0 178 70 28.2

New Mexico 11.4 285 70 32.1

Oklahoma 6.6 151 68 20.0

Maryland 11.3 300 67 27.8

Oregon 4.9 159 67 29.3

Kansas 6.0 115 66 18.0

Louisiana 15.4 249 66 22.2

Minnesota 2.7 72 66 14.9

Wisconsin 2.6 53 66 10.8

Indiana 7.2 113 65 21.0

Virginia 8.5 156 63 20.7

Nebraska 4.3 102 62 16.5

Georgia 17.4 211 60 25.8

Wyoming 6.8 161 60 15.6

Tennessee 13.2 188 59 26.9

Alabama 13.2 236 58 21.2

Iowa 2.2 56 57 11.3

New Hampshire 2.1 57 56 9.5

Idaho 2.6 120 54 14.2

Montana 6.0 109 53 16.4

Kentucky 9.7 109 52 16.3

Maine 2.1 83 51 7.8

Arkansas 8.8 190 50 19.5

Alaska 10.0 263 48 44.5

South Carolina 14.4 279 48 22.5

North Carolina 13.0 337 45 16.1

South Dakota 3.8 86 45 12.8

Mississippi 16.1 259 44 17.1

North Dakota 0.8 45 44 7.3

West Virginia 5.7 81 39 9.3

Vermont 2.2 48 32 11.2

- ¿Podrías añadir un segundo criterio de orden?, ¿cómo?

- Muestra por pantalla la columna con los datos de asesinato

> USArrests[,"Murder"]

[1] 13.2 10.0 8.1 8.8 9.0 7.9 3.3 5.9 15.4 17.4 5.3 2.6 10.4 7.2 2.2 6.0 9.7 15.4 2.1 11.3 4.4 12.1 2.7 16.1 9.0 6.0

[27] 4.3 12.2 2.1 7.4 11.4 11.1 13.0 0.8 7.3 6.6 4.9 6.3 3.4 14.4 3.8 13.2 12.7 3.2 2.2 8.5 4.0 5.7 2.6 6.8

- Muestra las tasas de asesinato para el segundo, tercer y cuarto estado

> USArrests[2:4,"Murder"]

[1] 10.0 8.1 8.8

- Muestra las primeras cinco filas de todas las columnas  
> USArrests[1:5,]

Murder Assault UrbanPop Rape

Alabama 13.2 236 58 21.2

Alaska 10.0 263 48 44.5

Arizona 8.1 294 80 31.0

Arkansas 8.8 190 50 19.5

California 9.0 276 91 40.6

- Muestra todas las filas para las dos primeras columnas

> USArrests[,1:2]

Murder Assault

Alabama 13.2 236

Alaska 10.0 263

Arizona 8.1 294

Arkansas 8.8 190

California 9.0 276

Colorado 7.9 204

Connecticut 3.3 110

Delaware 5.9 238

Florida 15.4 335

Georgia 17.4 211

Hawaii 5.3 46

Idaho 2.6 120

Illinois 10.4 249

Indiana 7.2 113

Iowa 2.2 56

Kansas 6.0 115

Kentucky 9.7 109

Louisiana 15.4 249

Maine 2.1 83

Maryland 11.3 300

Massachusetts 4.4 149

Michigan 12.1 255

Minnesota 2.7 72

Mississippi 16.1 259

Missouri 9.0 178

Montana 6.0 109

Nebraska 4.3 102

Nevada 12.2 252

New Hampshire 2.1 57

New Jersey 7.4 159

New Mexico 11.4 285

New York 11.1 254

North Carolina 13.0 337

North Dakota 0.8 45

Ohio 7.3 120

Oklahoma 6.6 151

Oregon 4.9 159

Pennsylvania 6.3 106

Rhode Island 3.4 174

South Carolina 14.4 279

South Dakota 3.8 86

Tennessee 13.2 188

Texas 12.7 201

Utah 3.2 120

Vermont 2.2 48

Virginia 8.5 156

Washington 4.0 145

West Virginia 5.7 81

Wisconsin 2.6 53

Wyoming 6.8 161

- Muestra todas las filas de las columnas 1 y 3

> USArrests[,c(1,3)]

Murder UrbanPop

Alabama 13.2 58

Alaska 10.0 48

Arizona 8.1 80

Arkansas 8.8 50

California 9.0 91

Colorado 7.9 78

Connecticut 3.3 77

Delaware 5.9 72

Florida 15.4 80

Georgia 17.4 60

Hawaii 5.3 83

Idaho 2.6 54

Illinois 10.4 83

Indiana 7.2 65

Iowa 2.2 57

Kansas 6.0 66

Kentucky 9.7 52

Louisiana 15.4 66

Maine 2.1 51

Maryland 11.3 67

Massachusetts 4.4 85

Michigan 12.1 74

Minnesota 2.7 66

Mississippi 16.1 44

Missouri 9.0 70

Montana 6.0 53

Nebraska 4.3 62

Nevada 12.2 81

New Hampshire 2.1 56

New Jersey 7.4 89

New Mexico 11.4 70

New York 11.1 86

North Carolina 13.0 45

North Dakota 0.8 44

Ohio 7.3 75

Oklahoma 6.6 68

Oregon 4.9 67

Pennsylvania 6.3 72

Rhode Island 3.4 87

South Carolina 14.4 48

South Dakota 3.8 45

Tennessee 13.2 59

Texas 12.7 80

Utah 3.2 80

Vermont 2.2 32

Virginia 8.5 63

Washington 4.0 73

West Virginia 5.7 39

Wisconsin 2.6 66

Wyoming 6.8 60

- Muestra solo las primeras cinco filas de las columnas 1 y 2

> USArrests[1:5,1:2]

Murder Assault

Alabama 13.2 236

Alaska 10.0 263

Arizona 8.1 294

Arkansas 8.8 190

California 9.0 276

- Extrae las filas para el índice Murder

> USArrests[,"Murder"]

[1] 13.2 10.0 8.1 8.8 9.0 7.9 3.3 5.9 15.4 17.4 5.3 2.6 10.4 7.2 2.2 6.0 9.7 15.4 2.1 11.3 4.4 12.1 2.7 16.1 9.0 6.0

[27] 4.3 12.2 2.1 7.4 11.4 11.1 13.0 0.8 7.3 6.6 4.9 6.3 3.4 14.4 3.8 13.2 12.7 3.2 2.2 8.5 4.0 5.7 2.6 6.8

Vamos con expresiones un poco mas complicadas:…

-¿Que estado tiene la menor tasa de asesinatos? ¿qué línea contiene esa información?, obtén esa informaciónn

> USArrests[which.min(USArrests[,"Murder"]),]

Murder Assault UrbanPop Rape

North Dakota 0.8 45 44 7.3

> which.min(USArrests[,"Murder"])

[1] 34

¿Que estados tienen una tasa inferior al 4%?, obtén esa informaciónn

> USArrests[which(USArrests[,"Murder"] < 4.0),]

Murder Assault UrbanPop Rape

Connecticut 3.3 110 77 11.1

Idaho 2.6 120 54 14.2

Iowa 2.2 56 57 11.3

Maine 2.1 83 51 7.8

Minnesota 2.7 72 66 14.9

New Hampshire 2.1 57 56 9.5

North Dakota 0.8 45 44 7.3

Rhode Island 3.4 174 87 8.3

South Dakota 3.8 86 45 12.8

Utah 3.2 120 80 22.9

Vermont 2.2 48 32 11.2

Wisconsin 2.6 53 66 10.8

¿Que estados estan en el cuartil superior (75) en lo que a poblacion en zonas urbanas se refiere?

> USArrests[which(USArrests[,"UrbanPop"] >= 75),]

Murder Assault UrbanPop Rape

Arizona 8.1 294 80 31.0

California 9.0 276 91 40.6

Colorado 7.9 204 78 38.7

Connecticut 3.3 110 77 11.1

Florida 15.4 335 80 31.9

Hawaii 5.3 46 83 20.2

Illinois 10.4 249 83 24.0

Massachusetts 4.4 149 85 16.3

Nevada 12.2 252 81 46.0

New Jersey 7.4 159 89 18.8

New York 11.1 254 86 26.1

Ohio 7.3 120 75 21.4

Rhode Island 3.4 174 87 8.3

Texas 12.7 201 80 25.5

Utah 3.2 120 80 22.9

\* Vamos a trabajar con otro dataframe. Descarga el fichero student.txt de la plataforma PRADO, almacena la información en una variable llamada “students”. Ten en cuenta que los datos son tab-delimited y tienen un texto para cada columna. Comprueba que R ha leído correctamente el fichero imprimiendo el objeto en la pantalla

> students

> students

height shoesize gender population

1 181 44 male kuopio

2 160 38 female kuopio

3 174 42 female kuopio

4 170 43 male kuopio

5 172 43 male kuopio

6 165 39 female kuopio

7 161 38 female kuopio

8 167 38 female tampere

9 164 39 female tampere

10 166 38 female tampere

11 162 37 female tampere

12 158 36 female tampere

13 175 42 male tampere

14 181 44 male tampere

15 180 43 male tampere

16 177 43 male tampere

17 173 41 male tampere

-Imprime solo los nombres de la columnas

> colnames(students)

[1] "height" "shoesize" "gender" "population"

-Llama a la columna height solo

> students$height

[1] 181 160 174 170 172 165 161 167 164 166 162 158 175 181 180 177 173

> students[,"height"]

[1] 181 160 174 170 172 165 161 167 164 166 162 158 175 181 180 177 173

-¿Cuantas observaciones hay en cada grupo?. Utiliza la función table(). Este commando se puede utilizar para crear tablas cruzadas (cross-tabulation)

El comando “table(students)” crea 4 tablas donde se compara el tamaño del pie con la altura y se consideran el sexo y población de los individuos, primero se compara mujeres de kuopio, después hombres de kuopio; tras esto se comparan mujeres y hombres de tampere por ese orden.

-Crea nuevas variables a partir de los datos que tenemos. Vamos a crear una variable nueva “sym” que contenga M si el genero es masculino y F si el genero es femenino. Busca en la ayuda información sobre la función ifelse(). Crea una segunda variable “colours” cuyo valor será “Blue” si el estudiante es de kuopio y “Red” si es de otro sitio.

> students$sym = ifelse(students$gender == "male","M","F")

> students$colours = ifelse(students$population == "kuopio","Blue","Red")

- Con los datos anteriores de height y shoesize y las nuevas variables crea un nuevo data.frame que se llame students.new

> students.new = data.frame(students$height,students$shoesize,sym,colours); students.new

students.height students.shoesize sym colours

1 181 44 M Blue

2 160 38 F Blue

3 174 42 F Blue

4 170 43 M Blue

5 172 43 M Blue

6 165 39 F Blue

7 161 38 F Blue

8 167 38 F Red

9 164 39 F Red

10 166 38 F Red

11 162 37 F Red

12 158 36 F Red

13 175 42 M Red

14 181 44 M Red

15 180 43 M Red

16 177 43 M Red

17 173 41 M Red

- Comprueba que la clase de student.new es un dataframe

> class(students.new)

[1] "data.frame"

- Crea dos subsets a partir del dataset student. Dividelo dependiendo del sexo. Para ello primero comprueba que estudiantes son hombres (male). Pista: busca información sobre la función which.

> which(students$gender=="male")

[1] 1 4 5 13 14 15 16 17

> which(students$gender=="female")

[1] 2 3 6 7 8 9 10 11 12

-Basándote en esa selección dada por which() toma solo esas filas del dataset student para generar el subset stundent.male

> students.male = students[which(students$gender=="male"),]; students.male

height shoesize gender population

1 181 44 male kuopio

4 170 43 male kuopio

5 172 43 male kuopio

13 175 42 male tampere

14 181 44 male tampere

15 180 43 male tampere

16 177 43 male tampere

17 173 41 male tampere

- Repite el procedimiento para seleccionar las estudiantes mujeres (females)

> students.female = students[which(students$gender == "female"),]; students.female

height shoesize gender population

2 160 38 female kuopio

3 174 42 female kuopio

6 165 39 female kuopio

7 161 38 female kuopio

8 167 38 female tampere

9 164 39 female tampere

10 166 38 female tampere

11 162 37 female tampere

12 158 36 female tampere

- Utiliza la function write.table() para guarder el contenido de student.new en un archivo.

> write.table(students.new,"student\_new.txt")

\* Accede al dataset “women”.

* Primero confirma que los datos están ordenados de forma creciente según la altura (height) y el peso (weight) sin mirar los datos

> is.unsorted(women$height)

[1] FALSE

> is.unsorted(women$weight)

[1] FALSE

* Crea una nueva variable “bmi”. Este valor responde a la siguiente fórmula: BMI = ( Weight in Pounds / (Height in inches) x (Height in inches) ) x 703

> women$bmi = ((women$weight\*2.20462)/((women$height\*2.54)\*\*2))\*703

> women$bmi

[1] 7.549988 7.568585 7.577083 7.585019 7.599069 7.599771 7.665643 7.675889 7.741687 7.807828 7.874243 7.940852 8.007563 8.074274

[15] 8.212275

* Ordena el dataframe por el valor de bmi y lpor orden alfabético de la variable name

investiga las funciones is.unsorted(), sort() and order()

> women = women[order(women$bmi),]

> women

height weight bmi

13 70 154 7.549988

12 69 150 7.568585

14 71 159 7.577083

11 68 146 7.585019

10 67 142 7.599069

15 72 164 7.599771

9 66 139 7.665643

8 65 135 7.675889

7 64 132 7.741687

6 63 129 7.807828

5 62 126 7.874243

4 61 123 7.940852

3 60 120 8.007563

2 59 117 8.074274

1 58 115 8.212275