Introducción a la Estadística

Breve descriptivo sobre R

Mayo de 2020 / UTDT

Software estadístico

R es el software de mayor interés y uso dentro de la comunidad Estadística y ramas afines. En primer lugar porque es gratuito y de código abierto; y en segundo lugar porque hay una amplia red de usuarios que contribuyen con paquetes (bloques de programas que resuelven tareas específicas) y soluciones prácticas a problemas concretos.

R será nuestro entorno de programación durante el curso pero también en cursos posteriores. Pero, porqué tengo que aprender a programar? Entre las razones más importantes se encuentran las siguientes:

- **Independencia:** Para poder realizar de manera independiente los cálculos y estimaciones pertinentes sin necesidad de 'expertos' externos.
- **Practicidad:** Cuanto tardarías en, por ejemplo, calcular un promedio de entre 1.000 observaciones a mano?
- **Claridad:** Poner las ideas en términos computacionales ayuda a clarificar los pensamientos y las ideas detrás de los métodos estadísticos propuestos y viceversa.
- **Diversión:** Es una de las tareas más divertidas (siempre que nos motive el problema) dentro del ámbito de la estadística.
- **Estar a la altura de las circunstancias** Hoy (casi) todo el mundo programa en mayor o menor medida, te vas a quedar atrás?

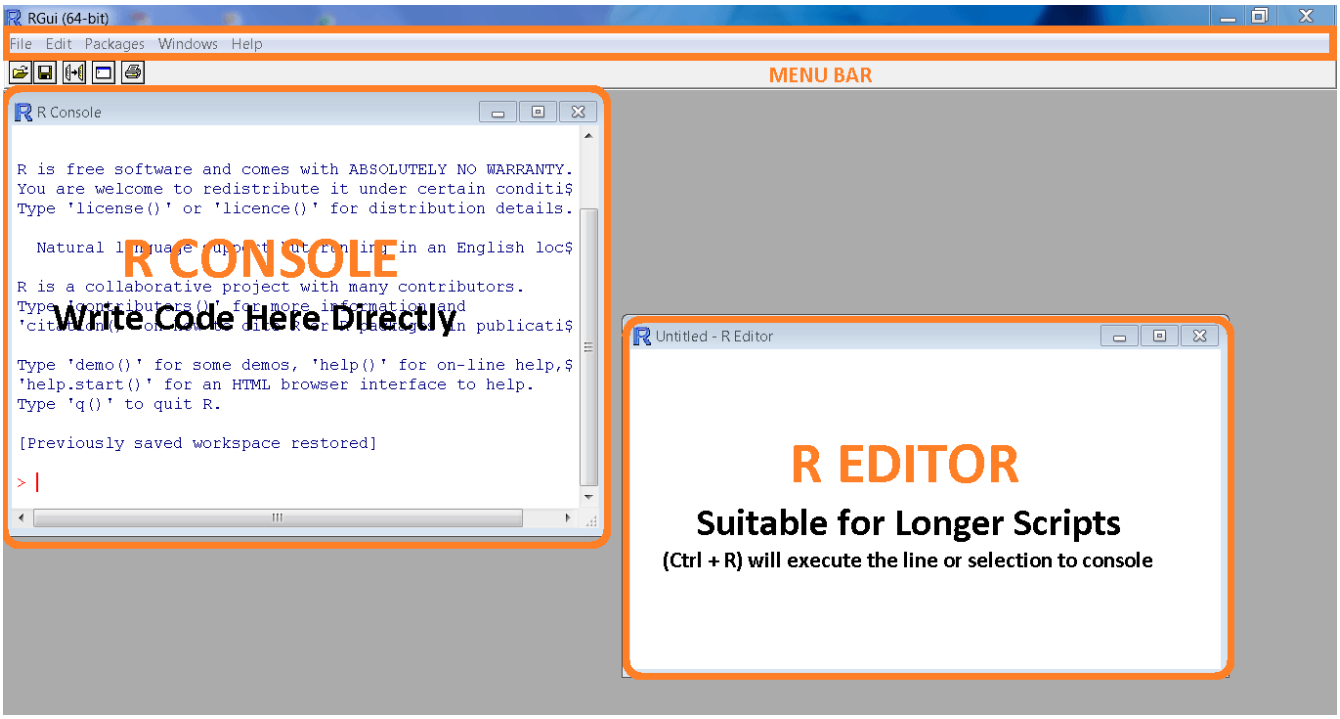
Descarga e instalación del programa

Se puede utilizar R en su plataforma original o desde RStudio (también gratuito). Este último tiene ventanas mejor organizadas y cuenta con más opciones gráficas. En cualquier caso, debes tener siempre instalado R en tu computadora. El RStudio es solo una carcasa que sirve para visualizar de manera un poco más amistosa el entorno.

Vamos a instalar en primer término el R en tu ordenador. Sigue las instrucciones:

- pincha en el link http://cran.r-project.org (puedes hacer click sobre el link si estas en tu computadora)
- pincha *download R for Windows* (si utilizas linux o alguna otra plataforma elige el fichero de descarga correspondiente)
- pincha en *base*
- elige *Download R X.x.x for Windows* (la versión más actualizada que figure en la web) y elige las opciones por defecto en los cuadros de instalación.

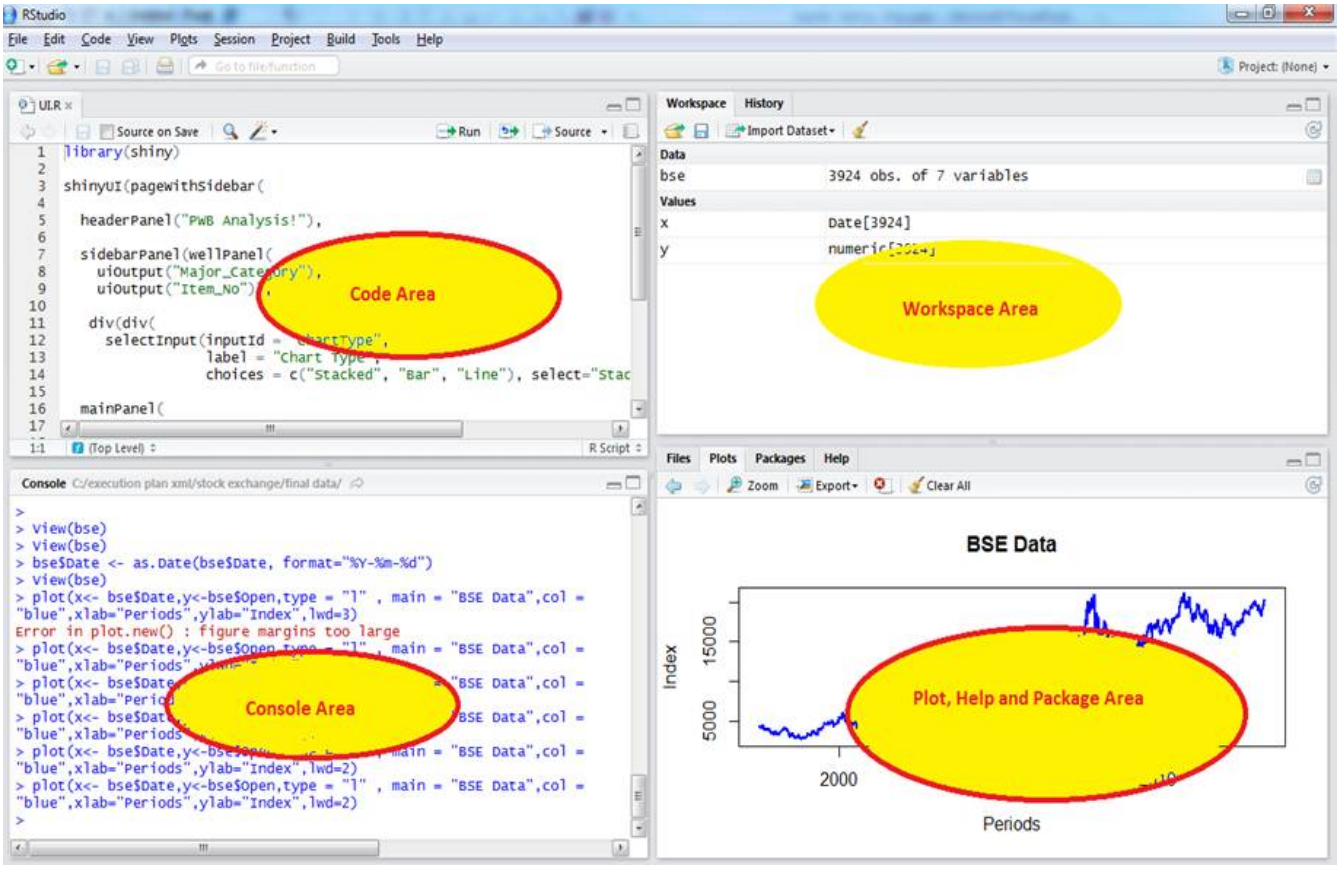
Una vez hayas instalado R en tu ordenador, busca el icono de R en el escritorio y haz doble click. Verás una ventana como esta:



Esta es la interfaz básica de R. Tiene disponible una consola en donde puedes ejecutar comandos y rutinas, y puedes abrir un *script* para ir guardando las líneas de código que generas (en caso de que quieras volver ejecutar las rutinas en otro momento). Puedes utilizar el R en este formato sin problemas, sin embargo te recomendamos que utilices RSutdio como interfaz de programación. Para ello debes instalar R-studio. Haz un click en el link anterior y sigue las instrucciones:

- abre http://www.rstudio.com (clikea sobre el link si estas en tu computadora)
- click *Download Rstudio*
- click *Download Rstudio Desktop*
- elige la versión adecuada para tu sistema operativo
- descarga e instala el programa eligiendo las respuestas por defecto en las ventanas de instalación.

Una vez finalizada la instalación haz doble-click en el icono de R-studio. Podrás ver las siguientes ventanas:



- Consola: aquí es donde escribimos los comandos que R interpreta y que nos permiten hacer análisis estadístico.
- Editor: en algunas ocasiones nos interesa utilizar reiteradas veces los comandos que R ejecuta. Esta ventana sirve para ir guardando los comandos que vamos ejecutando en R. Si quieres ejecutar algunos de los comandos guardados, simplemente los seleccionas en esta ventana y presionas CTRL + R, de esta manera los comandos seleccionados se ejecutan en la consola de R.
- Workspace: aquí podrás observar los elementos que R mantiene en su memoria.
- Files / Plots / Packages / Help: desde esta ventanas podrás visualizar los gráficos creados, abrir ficheros, ver que paquetes tienes instalados y recurrir a la ayuda del programa.

Ya estamos listos para empezar, manos a la obra!

Básicos de R

R puede ser utilizado como una calculadora (aunque es un poco naive usarlo solo para esto). Veamos que podemos hacer en la ventana de comandos:

```
3*2+2
```

```
## [1] 8
```

```
5^2
```

```
## [1] 25
```

```
sqrt(2)
```

```
## [1] 1.414214
```

```
exp(2)
```

```
## [1] 7.389056
```

```
sin(1/2)
```

```
## [1] 0.4794255
```

```
pi
```

```
## [1] 3.141593
```

```
log(3)
```

```
## [1] 1.098612
```

```
abs(-1)
```

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

Para obtener ayuda respecto de las funciones de R puedes escribir en la consola el comando *help('normbre de la función que deseas consultar')*, prueba escribiendo: help('Arithmetic')

Obviamente que no vamos a utilizar R solo para hacer cálculos aritméticos básicos. Para poder trabajar con mayor desenvoltura vamos a necesitar hacer uso de *estructuras de datos* un poco más complejas.

Estructuras de datos

Vectores

Vamos a crear vectores (y guardarlos en la memoria de R) utilizando el comando concatenar *c()*. Aquí debajo dejamos algunos ejemplos muy simples

```
#Creamos y guardamos dos vectores en la memoria de R
digitos = c(0,1,2,3,4,5,6,7,8,9)
print(digitos)
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```
dias = c('lu','mar','mier','jue','vie')
print(dias)
```

```
## [1] "lu"   "mar"  "mier" "jue"  "vie"
```

Va a ser de mucha utilidad acceder a elementos específicos de un vector que tengas guardado en la memoria, por ejemplo:

```
#Accediendo a elementos específicos de un vector:
digitos[1]
```

```
## [1] 0
```

```
dias[1:3]
```

```
## [1] "lu"   "mar"  "mier"
```

```
digitos[c(1,3,7)]
```

```
## [1] 0 2 6
```

```
dias[-5]
```

```
## [1] "lu"   "mar"  "mier" "jue"
```

También podemos evaluar condiciones lógicas sobre los elementos de un vector, por ejemplo:

```
digitos > 3
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
dias == 'vie'
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE
```

```
(digitos > 3 & digitos <= 7)
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
```

```
(dias == 'vie' | dias == 'lu')
```

```
## [1]  TRUE FALSE FALSE FALSE  TRUE
```

Algunos comandos útiles que sirven para crear, manipular y trabajar con vectores:

```
#Creando vectores con valores consecutivos y repetidos
a = seq(0,1,length.out = 5)
a
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
b = rep(1, 5)
b
```

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

```
#Operaciones algebráicas con vectores
2*a + 1
```

```
## [1] 1.0 1.5 2.0 2.5 3.0
```

```
a + b
```

```
## [1] 1.00 1.25 1.50 1.75 2.00
```

```
sqrt(b)
```

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

```
exp(a)
```

```
## [1] 1.000000 1.284025 1.648721 2.117000 2.718282
```

```
sum(a)
```

```
## [1] 2.5
```

```
prod(c(1:4)) # Producto de los elementos del vector (1,2,3,4)
```

```
## [1] 24
```

```
a^b
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
cumsum(a) # Suma acumulativa de los elementos de a
```

```
## [1] 0.00 0.25 0.75 1.50 2.50
```

```r
max(a) # Máximo
```

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

```r
min(a) # Mínimo
```

```
## [1] 0
```

```r
sort(a,decreasing = F) # Ordenamos de mayor a menor.
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```r
#Concatenando vectores en un vector
a_union_b = c(a,b)
a_union_b
```

```
##  [1] 0.00 0.25 0.50 0.75 1.00 1.00 1.00 1.00 1.00 1.00
```

```r
#Algunas funciones estadísticas básicas en R
mean(a_union_b)
```

```
## [1] 0.75
```

```r
var(a_union_b)
```

```
## [1] 0.1388889
```

```r
sd(a_union_b)
```

```
## [1] 0.372678
```

```r
range(a_union_b)
```

```
## [1] 0 1
```

```r
quantile(a_union_b, 0.75)
```

```
## 75%
##   1
```

```r
length(a_union_b)
```

```
## [1] 10
```

```r
# Otras funciones
floor(a) # Redondeo 'hacia abajo'
```

```
## [1] 0 0 0 0 1
```

```r
ceiling(a) # Redondeo 'hacia arriba'
```

```
## [1] 0 1 1 1 1
```

```r
round(a,1) # Redondeo 'a 1 decimal' (1 es un parámetro de la función round que puedo cambiar!)
```

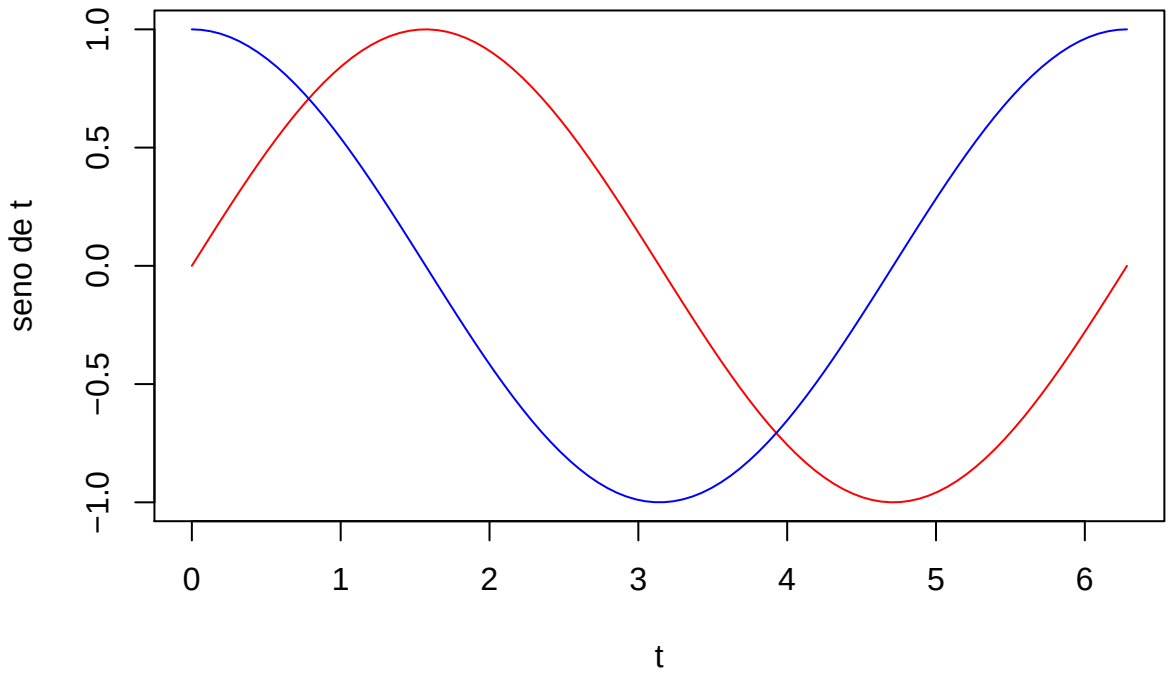
```
## [1] 0.0 0.2 0.5 0.8 1.0
```

Pincha en este párrafo para acceder a una carta de referencia sobre comandos básicos de R, donde también enconstrarás algunas instrucciones gráficas de R que pueden ser de utilidad:

```r
t = seq(0,2*pi,length.out = 100)
f_t = sin(t)
# Gráficos de linea standard
plot(t, f_t, type='l', xlab='t', ylab = 'seno de t', col = 'red')
```

```
g_t = cos(t)

points(t, g_t, type='l', col = 'blue')
```



En R puedes crear estructuras de datos mucho más complejas (y ricas) como matrices, array's, data-frames o listas. A continuación haremos una breve intro a alguna de estas estructuras, como *cargar datos externos* en R (que almacenaremos en un data-frame), y veremos que comandos utilizar para hacer estadística descriptiva con la información guardada.

Ejercicios en clase:

1- Crea un vector que contenga las primeras 10 potencias de 2.

2- Grafica la función anterior, es decir $f(n) = 2^n$, para $n = 1, ..., 10$.

Matrices y Data frames

Una matriz es un arreglo bidimensional de (generalmente) números.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

El conjunto de las matrices de tamaño $n \times m$ se representa como $\mathcal{M}^{n \times m}$. El tamaño de una matriz siempre se da con el número de filas primero y el número de columnas después. En el caso de la matriz $\mathbf{A}$ del ejemplo anterior tiene n-filas y n-columnas $\mathbf{A} \in \mathcal{M}^{n \times n}$) y estas matrices se llaman matrices *cuadradas* (porque tienen el mismo número de filas que de columnas). En estadística tendrás que trabajar con este tipo de matrices en muchos contextos.

Las matrices pueden entenderse en R como objetos bi-dimensionales y por lo tanto habrá dos indices para indicar la posición de cada elemento en este objeto: una para indexar a que *fila* y el otro para indexar a que *columna* pertenece el objeto en cuestión.

Con el comando **matrix(*datos*, *nrow*, *ncol*, byrow = F)** en R creamos matrices. La estructura general

```
x <- c(1,2,3,4)
matrix(x, ncol = 2, nrow =2, byrow = F) # byrow = F -> coloca los datos x columnas.
```

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

```
matrix(x, ncol = 2, nrow =2, byrow = T) # byrow = T -> coloca los datos x filas.
```

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

En los contextos con datos *multivariantes* se suelen utilizar matrices para representar los datos extraídos de una población. Por ejemplo:

```
peso   <- c(60,75,82,78,69,50)
altura <- c(151,174,181,175,164,141)
Datos = matrix(c(peso,altura), ncol = 2, byrow = F) # byrow = F -> coloca los datos x columnas.
Datos
```

```
##      [,1] [,2]
## [1,]   60  151
## [2,]   75  174
## [3,]   82  181
## [4,]   78  175
## [5,]   69  164
## [6,]   50  141
```

```
colnames(Datos) = c('Peso','Altura') # Podemos colocar nombre a las columnas.
head(Datos,3) # Con el comando head('matriz','# filas') visualizamos las # primeras filas.
```

```
##      Peso Altura
## [1,]   60    151
## [2,]   75    174
## [3,]   82    181
```

Operaciones algebraicas con Matrices:

Al igual que con los vectores, para sumar dos matrices (sin que el R de un error) debemos tener en cuenta la restricción de que ambas matrices tengan el mismo tamaño (es decir, el mismo número de filas y columnas). Recordemos que para poder hacer el **producto entre matrices** AB necesitamos que la cantidad de columnas de A coincidan con el número de filas de B (en otro caso R nos dirá que no se puede hacer el producto). Veamos algunos ejemplos

```
A = matrix(c(1:9),ncol=3)
print(A)
```

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

```
B = matrix(c(9:1),ncol=3)
print(B)
```

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

```
A + B # Suma entre matrices
```

```
##      [,1] [,2] [,3]
## [1,]   10   10   10
## [2,]   10   10   10
## [3,]   10   10   10
```

```
A%*%B # Producto entre matrices
```

```
##      [,1] [,2] [,3]
## [1,]   90   54   18
## [2,]  114   69   24
## [3,]  138   84   30
```

```
A*B   # Producto 'elemento a elemento'
```

```
##      [,1] [,2] [,3]
## [1,]    9   24   21
## [2,]   16   25   16
## [3,]   21   24    9
```

```
A^2 # Cuadrado de los elementos de A
```

```
##      [,1] [,2] [,3]
## [1,]    1   16   49
## [2,]    4   25   64
## [3,]    9   36   81
```

```
sqrt(A) # Raíz de los elementos de  A
```

```
##          [,1]     [,2]     [,3]
## [1,] 1.000000 2.000000 2.645751
## [2,] 1.414214 2.236068 2.828427
## [3,] 1.732051 2.449490 3.000000
```

```
log(A)  # Logaritmo de los elementos de  A
```

```
##           [,1]     [,2]     [,3]
## [1,] 0.0000000 1.386294 1.945910
## [2,] 0.6931472 1.609438 2.079442
## [3,] 1.0986123 1.791759 2.197225
```

```
exp(A) # Exponencial de los elementos de  A
```

```
##           [,1]      [,2]     [,3]
## [1,]  2.718282  54.59815 1096.633
## [2,]  7.389056 148.41316 2980.958
## [3,] 20.085537 403.42879 8103.084
```

```
sin(A) # Seno de los elementos de  A
```

```
##           [,1]       [,2]      [,3]
## [1,] 0.8414710 -0.7568025 0.6569866
## [2,] 0.9092974 -0.9589243 0.9893582
## [3,] 0.1411200 -0.2794155 0.4121185
```

Las funciones de *suma*, *valores absolutos*, *maximos*, *mínimos* también se pueden emplear con una matriz (compruébalo tu mismo con los anteriores ejemplos). Para acceder a *elementos* de una matriz, al igual que con los vectores, usamos el comando [***filas,columnas***]. La única diferencia es que ahora necesitamos especificar tanto las filas como las columnas a las que queremos acceder:

```
A[2,1] # Nos devuelve el elemento que ocupa la posición de la segunda fila y la primera columna.
```

```
## [1] 2
```

Al igual que con los vectores, podemos utilizar de varias maneras el operador:

```
C[-1,c(1,3)] # Quitamos las filas 1 y nos quedamos todas las columnas 1 y 3
```

```
## Error in C[-1, c(1, 3)]: objeto de tipo 'closure' no es subconjunto
```

Ejercicios en clase:

- 3. Construye en R la siguientes matrices:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{bmatrix} \qquad B = \begin{bmatrix} 6 & 5 & 4 \\ 1 & 2 & 3 \\ 9 & 8 & 7 \end{bmatrix}$$

  y luego verifica (y explica) que resultado obtienes al aplicar las siguientes funciones:

- c(A)
- A <= 5
- A > B
- t(B)
- diag(A)
- rowSums(B)
- colSums(A)
- apply(A, 1, max)
- apply(A, 2, max)
- apply(B, 1, mean)
- apply(B, 2, sd)

Data Frames

Los **data frames** son una estructura de datos que generaliza a las matrices, en el sentido en que las columnas (variables a menudo) pueden ser de diferente tipo entre sí (no todas numéricas, por ejemplo). Sin embargo, todos los elementos de una misma columna deben ser del mismo tipo. Al igual que las filas y columnas de una matriz, todos los elementos de un data frame deben ser de la misma longitud. De este modo, pueden usarse funciones tales como *dim*, *colnames*, *rownames*, etc, sobre un data frame como si se tratara de una matriz. Los datos de un data frame pueden ser accedidos como elementos de una matriz o de una lista.

Para introducir los dataframes, partiremos de la matriz de pesos y alturas construidas al principio:

```r
peso   <- c(60.1,75.5,82.9,78.5,69.6,50.4)
altura <- c(151,174,181,175,164,141)
region<-c('IV','V','Metropolitana','V','Metropolitana','V')

Datos = data.frame(peso, altura, region)

print(Datos)
```

```
##   peso altura        region
## 1 60.1    151            IV
## 2 75.5    174             V
## 3 82.9    181 Metropolitana
## 4 78.5    175             V
## 5 69.6    164 Metropolitana
## 6 50.4    141             V
```

```r
str(Datos) # En el mismo objeto 'Datos' hay diferentes tipos de datos.
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ peso  : num  60.1 75.5 82.9 78.5 69.6 50.4
##  $ altura: num  151 174 181 175 164 141
##  $ region: Factor w/ 3 levels "IV","Metropolitana",..: 1 3 2 3 2 3
```

Para acceder a los diferentes elementos dentro de un data frame podemos utilizar el mismo comando que en el caso de las matrices, sin embargo y desde el punto de vista práctico cuando trabajamos con datos almacenados en *data-frames* será mas cómodo hacer referencia a las columnas *por su nombre* (que en general se refieren al nombre de cada una de las variables de tu data-set). Para ello necesitamos indicarle a R que queremos trabajar de esta forma utilizando el comando **attach()** sobre el data frame y luego extraer las variables por su nombre. Veamos un ejemplo:

```r
# Queremos extraer la columna de altura del data--frame:
Datos[,2]
```

```
## [1] 151 174 181 175 164 141
```

```r
# Alternativamente
Datos$altura
```

```
## [1] 151 174 181 175 164 141
```

```r
# o alternativamente:
attach(Datos)
altura
```

Ejercicios:

4- Crea el anterior data frame en tu sesión de R y calcula la media de las columnas peso y altura.

5- Emplea el comando *summary(Datos)*. Analiza la información brindada.

6- Investiga para sirve el comando **detach()** en R. Puedes ayudarte con los tutoriales online de R o simplemente utilizando la ayuda en linea de R: *?detach* o *help(detach)*.