Práctica 1

El objetivo de esta primera práctica es que nos comencemos a familiarizar con la manera de trabajar de R y con las principales características de su sintaxis.Seguiremos los pasos siguientes:

- Leeremos un fichero de texto con los datos y lo importaremos a R.
- Usaremos algunos comandos para obtener algunas medidas y gráficos descripitivos elementales.
- Escribiremos una función muy sencilla que simula el lanzamiento de un dado.

Algunos aspectos básicos a tener en cuenta

- R **distingue** mayúsculas y minúsculas.
- Para asignar contenido a un objeto usamos `<-` o `=`. Por ejemplo, tanto `x = 10` como `x <- 10` asignan a `x` el valor `10`.
- Para imprimir por pantalla el contenido de un objeto simplemente escribimos su nombre en la **Consola**. Para ver su estructura (tipo de objeto y un resumen de su contenido y/o componentes) pinchamos en el nombre del objeto en el **Environment**.
- Para usar los comandos escribimos el nombre del comando seguido de sus argumentos **entre paréntesis**. Por ejemplo, `ls()` da una lista de los objetos en el área de trabajo. En este ejemplo concreto, como no usamos argumentos (diferentes a los que el comando tenga por defecto) no escribimos nada en el paréntesis. Otro ejemplo: si ejecutamos `rm(A)`, se borra el objeto `A` del **Environment**.
- Para obtener ayuda usamos el comando `help`. Por ejemplo, `help(mean)` o `?mean` para obtener ayuda sobre el comando `mean` que calcula la media. Otra opción es teclear `mean` en el buscador (icono lupa) de la pestaña `Help`.
- El comando `c` se usa para combinar varios objetos en uno solo. Por ejemplo:

```
x = c(3, 5, 10)     # Crea el vector x = (3, 5, 10)
x
```

```
## [1]  3  5 10
```

```
mean(x) # Calcula la media de x
```

```
## [1] 6
```

- Para escribir cadenas de caracteres da lo mismo delimitar la cadena con doble comilla `"` o con una comilla vertical `'`. Por ejemplo,

```
print("Hola")
```

```
## [1] "Hola"
```

```
print('Hola')
```

```
## [1] "Hola"
```

- Al entrar en R, se considera cierto directorio por defecto como el *directorio de trabajo*. En RStudio este directorio puede cambiarse temporalmente (para la sesión actual) pinchando en **Session ▶ Set Working Directory**, y globalmente (cada vez que arranquemos RStudio) pinchando en **Tools ▶ Global Options**. Resulta conveniente tener un directorio diferente para cada proyecto que realicemos.

- Al salir de R se nos pregunta si queremos guardar el área de trabajo (salvo que se haya desactivado esta opción anteriormente). Si respondemos afirmativamente, se crea un fichero `.RData` que contiene todos los objetos del área de trabajo en el momento de salir. Posteriormente, abriendo este fichero (**Session ▶ Load Worspace**) podremos entrar en R y seguir trabajando con las variables que habíamos definido antes de salir. No tiene especial interés guardar el `.RData` de la sesión, pues sólo guarda objetos definidos en ese instante de la sesión de trabajo, pero no guarda los comandos con los que se generaron. Si hemos escrito todos los comandos de la sesión de trabajo en un R Script (ver punto siguiente), entonces podremos generar los mismos objetos cuantas veces queramos.
- Todos los comandos y código de R se pueden guardar en R Script, un fichero de texto (con extensión `.R`) dentro del directorio de trabajo. El comando `source` se puede usar para ejecutar todo el código contenido en un fichero de texto. Para ejecutar el código línea a línea pincharemos en el botón `Run` del R Script, lo cual ejecuta la línea donde se sitúa el cursor.

Bajar unos datos de la web y describir sus principales características

Antes de poder trabajar con un conjunto de datos debemos asegurarnos de que tiene un formato con el que R pueda trabajar. Es bastante habitual encontrar datos en la web en un fichero de texto (`.txt`,`.dat`), que es un formato universal para cualquier programa de análisis de datos. En Windows, para descargar un fichero de datos de la red al disco duro, pincharemos con el botón derecho del ratón sobre el enlace de los datos y elegiremos la opción `Guardar enlace como`. Entonces guardamos el fichero en el directorio de trabajo.

Los datos `notas.txt`

La Comunidad de Madrid evalúa anualmente a los alumnos de sexto de primaria de todos los colegios sobre varias materias. Los datos que vamos a utilizar corresponden a las notas medias obtenidas por los colegios en los años 2009 y 2010 (fuente: diario *El País*) junto con el tipo de colegio (concertado, privado o público)

Bajar los datos y generar un fichero de R

El comando `read.table` lee un fichero de datos de texto y crea un objeto `data.frame` con el que R ya puede trabajar. Un `data.frame` es la estructura de una tabla de datos en R, que comparte muchas de las propiedades de las matrices y las listas. En nuestro ejemplo vamos a usar los siguientes argumentos:

- Entre comillas (por ser una cadena de caracteres) la ruta en la que se encuentra el fichero y el nombre del mismo.
- El argumento `sep` que especifica la separación entre las distintas variables. En este caso, un espacio en blanco (que es la opción por defecto, así que podríamos quitar lo de `sep = ' '`).
- El argumento `dec` que especifica cómo se separa en los números la parte entera y la parte decimal. En este caso, mediante una coma. Por defecto es `dec = "."`.
- El argumento `header` que especifica si la primera fila del fichero contiene (`TRUE`) o no (`FALSE`) los nombres de las variables. En este caso, la primera fila sí contiene los nombres de las variables.
- El argumento `as.is` por defecto toma el valor `TRUE` y el efecto es que las columnas de caracteres (las muestras de variables cualitativas) no se cargan como objeto de tipo factor (esto es importante cuando manejamos datos cualitativos). Para que las columnas de caracteres sí las importe como un factor elegimos `as.is = FALSE`.

```
notas <- read.table('http://verso.mat.uam.es/~amparo.baillo/MatEstI/notas.txt', sep = ' ', dec = ',', he
```

Hemos creado un `data.frame`, llamado `notas`, que contiene los datos en un formato con el que R ya puede trabajar.

Medidas descriptivas elementales

Para ver los nombres de las variables del fichero `notas`, escribimos:

```
names(notas)
```

```
## [1] "tipo"   "nota09" "nota10"
```

Dentro de este fichero cada variable se identifica usando la sintaxis `fichero$variable`. Por ejemplo, las medias de las notas de 2009 y 2010 se obtienen con los comandos:

```
mean(notas$nota09)
```

```
## [1] 6.604877
```

```
mean(notas$nota10)
```

```
## [1] 5.439943
```

o también se pueden calcular ambas simultáneamente con

```
colMeans(notas[,c(2,3)])
```

```
##   nota09   nota10
## 6.604877 5.439943
```

o con

```
apply(notas[,c(2,3)],2,mean)
```

```
##   nota09   nota10
## 6.604877 5.439943
```

Lo que hemos hecho ha sido calcular las medias de las columnas 2 y 3 de `notas` (`notas[,c(2,3)]`) con el comando `apply`, que aplica una función (en este caso `mean`) a lo largo de una dimensión de la matriz (dimensión 1 si la aplica por filas y dimensión 2 si la aplica por columnas). La sintaxis es

`apply(matriz,dimensión,función)`

Las desviaciones típicas de las notas de 2009 y 2010 se obtienen con los comandos:

```
sd(notas$nota09)
```

```
## [1] 1.032083
```

```
sd(notas$nota10)
```

```
## [1] 1.195158
```

o también con

```
apply(notas[,c(2,3)],2,sd)
```

```
##   nota09   nota10
## 1.032083 1.195158
```

El comando `summary` permite calcular algunas medidas descriptivas elementales de *todas* las variables del fichero simultáneamente:

```
summary(notas)
```

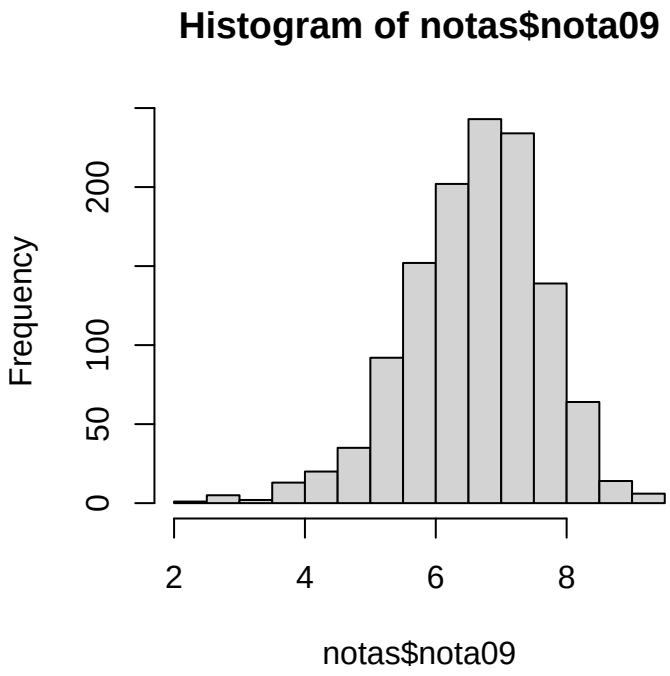
```
##          tipo         nota09          nota10
##  concertado:392   Min.   :2.170   Min.   : 0.000
##  privado   : 98   1st Qu.:5.970   1st Qu.: 4.810
##  publico   :732   Median :6.685   Median : 5.497
```

```
##                  Mean   :6.605   Mean   : 5.440
##                  3rd Qu.:7.350   3rd Qu.: 6.250
##                  Max.   :9.340   Max.   :10.530
```

Algunos gráficos

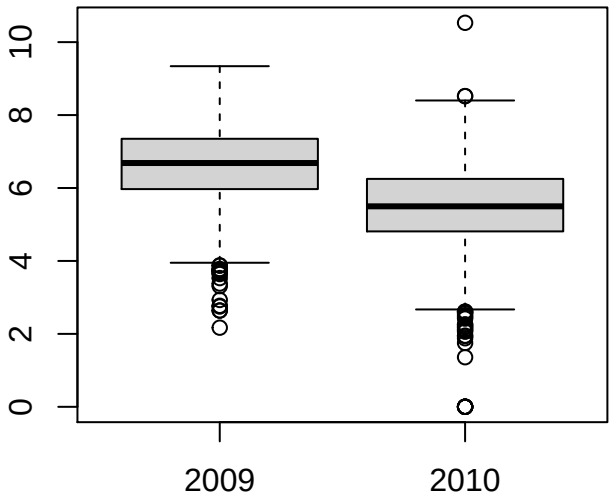
Un histograma de las notas de 2009. La distribución es bastante simétrica aunque muestra una ligera asimetría a la izquierda.

```
hist(notas$nota09)
```



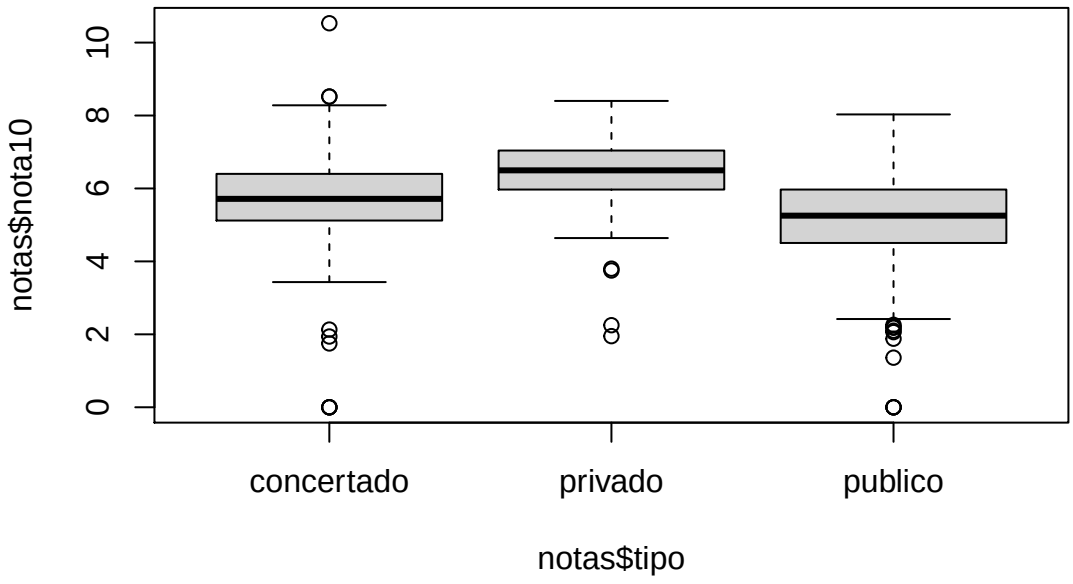
Para comparar conjuntos de datos un diagrama de caja puede ser más útil que un histograma. Por ejemplo, para comparar las notas de 2009 con las de 2010:

```
boxplot(notas$nota09, notas$nota10, names=c("2009","2010"))
```



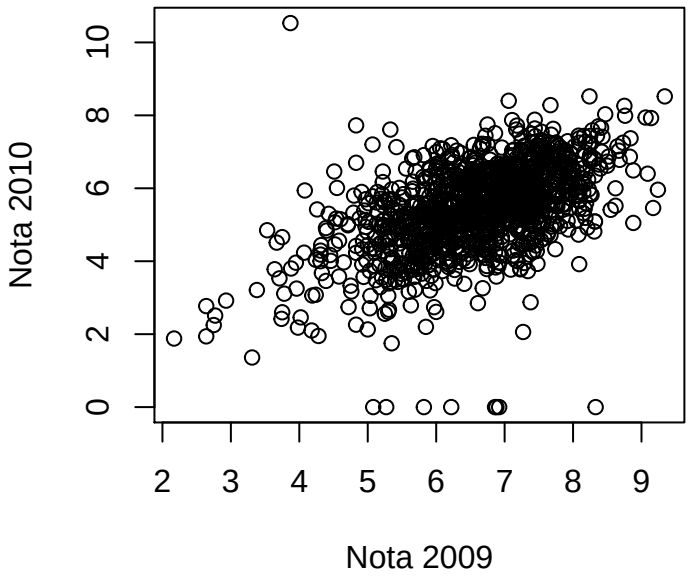
Vemos que en 2010 las notas tienden a ser más bajas. También podemos comparar las notas de 2010 para cada tipo de colegio. Aunque las diferencias no son muy grandes, los centros privados tienden a tener notas más altas que los concertados, y éstos más altas que los públicos.

```
boxplot(notas$nota10 ~ notas$tipo)
```



Para estudiar si hay relación entre las notas de los dos años podemos representar la correspondiente nube de puntos:

```
plot(notas$nota09, notas$nota10,xlab = "Nota 2009", ylab = "Nota 2010")
```



Se observa una relación positiva moderada. Numéricamente, esta relación se puede cuantificar mediante la covarianza y la correlación:

```
cov(notas$nota09, notas$nota10)
```

```
## [1] 0.650412
```

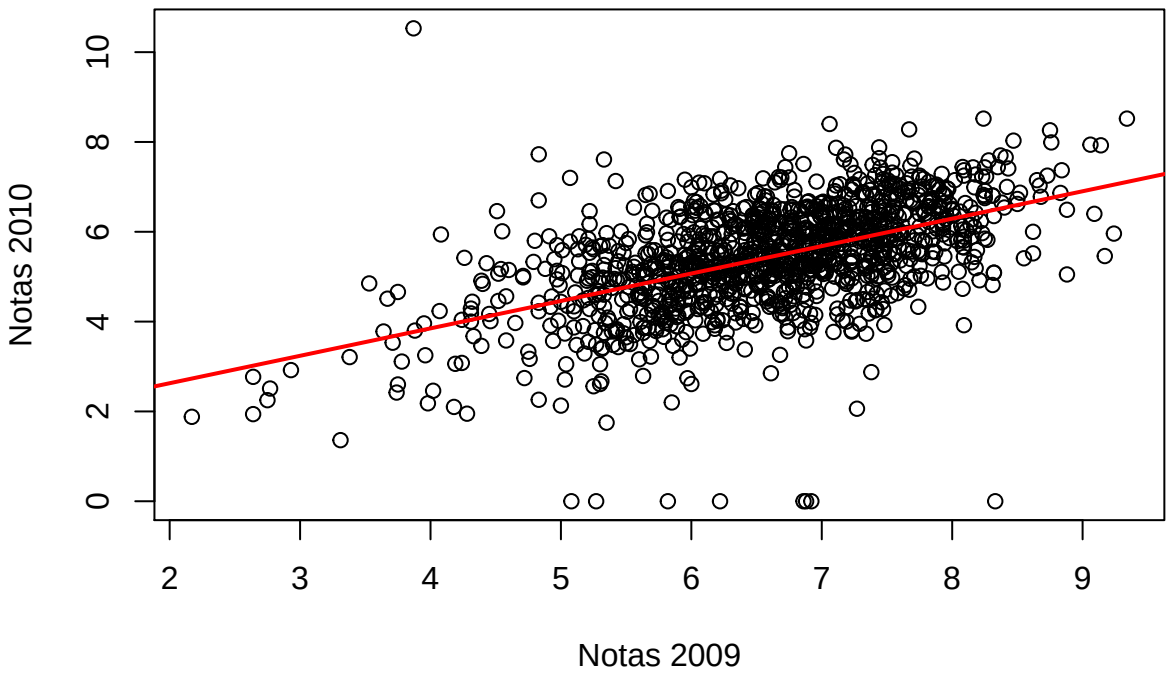
```
cor(notas$nota09, notas$nota10)
```

```
## [1] 0.5272887
```

El valor de la correlación es moderadamente positivo, tal y como habíamos observado al mirar la nube de puntos.

Podemos calcular la recta de regresión de las notas de 2010 sobre las de 2009:

```
y = notas$nota10
x = notas$nota09
recta = lm(y~x)
plot(x,y,xlab="Notas 2009",ylab="Notas 2010")
abline(recta,lwd=2,col="red")
```



Podemos ver todos los parámetros que se pueden modificar en un dibujo escribiendo en la consola `?par`.

Una función muy sencilla

Podemos escribir también nuestras propias funciones. En este apartado vamos a ver un ejemplo muy simple. Se trata de escribir una función que simule los resultados de n tiradas de un dado. Por defecto $n = 50$. El resultado es un vector que contiene las frecuencias de cada posible resultado. Abrimos el editor de RStudio (**File ▶ New File ▶ R script**) y escribimos:

```
Dado <- function(n = 50){
#
# Genera n lanzamientos de un dado
# y devuelve la tabla de frecuencias
#
    lanzamientos <- sample(1:6,n,rep=TRUE)   # Selecciona con reemplazamiento n números en 1,2,...,6
    frecuencias <- table(lanzamientos)          # Obtiene la tabla de frecuencias
    return(frecuencias)
}
```

Una vez guardado el fichero que contiene la función podemos usar `source("nombre del fichero.R")` para cargar la función en el área de trabajo y poder usarla o, alternativamente, copiar y pegar el código anterior en la consola. Una vez hecho esto, ya podemos usar la función. Por ejemplo, supongamos que hemos guardado la función en el fichero `Dado.R`.

```
source("Dado.R")
Dado(1000)
```

```
## lanzamientos
##   1   2   3   4   5   6
## 142 174 201 159 165 159
```

```
Dado()
```

```
## lanzamientos
##  1  2  3  4  5  6
##  7  3  7 11 10 12
```

Gráficos

Los comandos gráficos en R se dividen en dos tipos:

- **Comandos de alto nivel**: Son comandos que siempre abren una nueva ventana gráfica. Por ejemplo, `plot`, `hist`, `boxplot`, `pairs`
- **Comandos de bajo nivel**: Son comandos que permiten añadir elementos (puntos, líneas, texto, etc.) a un gráfico ya existente.

Gráficos: comandos de alto nivel

- Los argumentos que más se usan son:

Uso	Argumento
Tipo de gráfico (puntos, líneas)	`type='l'`
Leyendas de los ejes	`xlab=`, `ylab=`
Rango de los ejes	`xlim=c(min,max)`, `ylim=c(min,max)`
Título	`main=`

Gráficos: comandos de bajo nivel

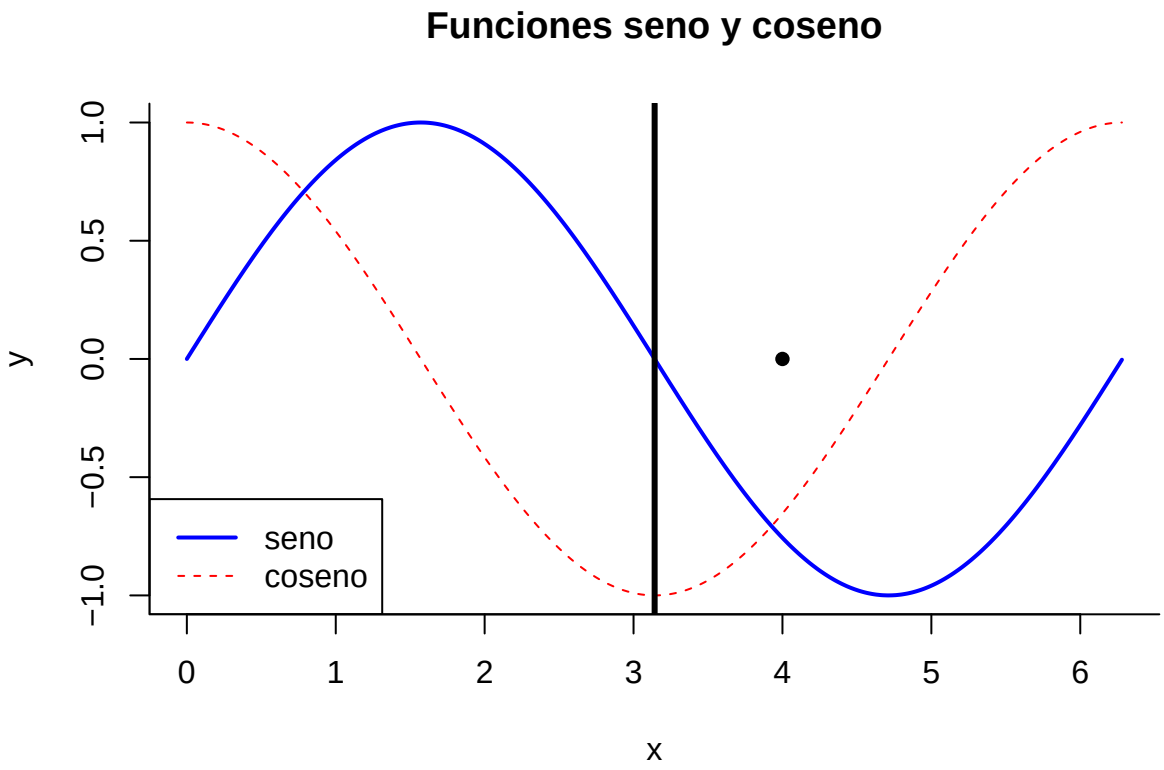
Los que más se usan son:

Uso	Commando
Añadir puntos (x_i, y_i)	`points(x,y)`
Añadir líneas	`lines(x,y)`
Dibuja una recta $y = a + bx$	`abline(a,b)`
Dibuja una recta $y = c$	`abline(h=c)`
Dibuja una recta $x = c$	`abline(v=c)`
Dibuja la recta de regresion	`abline(lm(y~x))`
Añade título	`title()`
Añade una leyenda en (x, y)	`legend(x,y,legend)`

Ejecuta el siguiente código línea a línea para tratar de entender lo que hace.

```
x <- seq(0, 2*pi, 0.01)
y <- sin(x)
plot(x,y, t='l', col='blue', lwd=2, bty='l')
lines(x, cos(x), col='red', lty=2)
```

```
abline(v=pi, lwd=3)
points(4,0,pch=16)
legend('bottomleft', c('seno','coseno'),lty=c(1,2),lwd=c(2,1),col=c('blue','red'))
title('Funciones seno y coseno')
```



Más información sobre cómo modificar gráficos (títulos, ejes, etc) se puede encontrar, por ejemplo, en `https://en.wikibooks.org/wiki/R_Programming/Graphics`. Además, para los frikis de R, hay un versátil paquete de R llamado `ggplot2` que hace unos gráficos fantásticos, pero es complicado de manejar (yo no sé usarlo, pero mi compañero José Ramón Berrendero está en proceso de dominarlo).