PRÁCTICA 1: ANÁLISIS INICIAL DE DATOS MULTIVARIANTES.
ANÁLISIS ESTADÍSTICO MULTIVARIANTE.
GRADO EN CIENCIA E INGENIERÍA DE DATOS.

Sumario: En esta práctica mostramos cómo realizar un estudio previo de los datos multivariantes con los que vamos a trabajar. Esta parte es fundamental y se repetirá en todas las prácticas posteriores. Usaremos algunas de las técnicas vistas en la asignatura de primero.

1. Instalación de R y RStudio

Recordamos que el programa R es un programa de cálculo estadístico gratuito que se completa con **paquetes** elaborados por los propios usuarios. Se puede descargar desde

`http://www.r-project.org`

indicando el sistema operativo.

El programa RStudio es una aplicación (o escritorio) para manejar R (y Python) de una manera más eficiente. Por eso, es muy importante instalar primero R. Rstudio tiene diversas versiones de pago y gratuitas. Para instalar la gratuita podemos ir a

`http://www.rstudio.com/ide/download/desktop`

y descargar *RStudio Desktop*.

Cuando abrimos RStudio nos deben aparecer tres ventanas. La ventana de la izquierda, denominada *Console*, sirve para introducir directamente los comandos de R. En ella también se mostrarán los resultados (salidas) obtenidos al ejecutar uno o varios comandos.

Si queremos guardar los comandos para usarlos en sesiones posteriores (por ejemplo, en los controles), podemos abrir una nueva ventana en la que podremos escribir y ejecutar los comandos de las prácticas. Para hacer esto, en el menú superior seleccionaremos las opciones:

File>New File>R Script

Tras esto, el programa tendrá cuatro ventanas. La nueva ventana, que denominaremos escritorio (o script), se sitúa en la parte superior izquierda. Para realizar una operación en esta ventana la teclearemos `1+1` y pincharemos sobre el icono `Run` (o pulsaremos simultaneamente Ctr+intro o command+intro).

Para guardar los comandos de la ventana escritorio basta pulsar sobre el icono del disquete (o seleccionar en el menú superior File>Save). La primera vez que lo hacemos, el programa nos pedirá el nombre del fichero y la ubicación donde queremos guardarlo. Estos ficheros (denominados scripts) se suelen guardar con la extensión ".R". Por ejemplo, en este caso, el fichero se puede denominar `Practica1.R` (sin tilde). Posteriormente basta con pulsar sobre el icono del disquete para que se guarde. Es muy importante hacer esto a menudo porque si el programa se bloquea perderemos los comandos no guardados. La utilidad de las otras ventanas se mostrará en las secciones siguientes.

2. Modelos multivariantes

Recomendamos trabajar en la ventana de *scripts* para poder guardar el fichero de comandos. Si queremos incluir **comentarios** en esta ventana que no se ejecuten como comando (darían error), debemos poner delante el símbolo #. Por ejemplo, en la primera línea de nuestro escritorio podemos escribir: # Practica 1 y vuestro nombre. En los controles y trabajos podemos pediros que enviéis los scripts por el aula virtual.

Los datos se pueden guardar en diversos formatos en R. Los que están activos (cargados) aparecerán en la ventana superior derecha denominada *Environment*. Al inicio estará vacía y pondrá: *Environment is empty*. Para limpiar esta ventana usaremos el icono de la escoba (los datos no guardados se perderán).

Los datos multivariantes tendrán diversas variables que representaremos como un vector aleatorio $X = (X_1, \ldots, X_k)$. En el caso bivariante podemos usar (X, Y). Los datos obtenidos de este vector se representarán en forma de matriz o de tabla (dataframe) donde los individuos se representan como líneas y en cada columna se representarán los valores en esas variables en cada individuo.

El modelo más usual es el de la distribución normal multivariante. Para calcular su función de densidad debemos cargar el paquete `mvtnorm`. Para cargarlo haremos `library(mvtnorm)`. Si no está instalado, deberemos ir a la ventana inferior derecha, pulsar sobre la pestaña "Packages" y sobre "Install" para instalarlo. Tras esto debemos ejecutar de nuevo `library(mvtnorm)` para cargarlo. Ahora ya podemos teclear:

```
V<-matrix(NA,2,2)
V[1,]<-c(1,1/2)
V[2,]<-c(1/2,1)
mu<-c(0,0)
x<-c(1,1)
dmvnorm(x,mu,sigma=V)
```

El resultado debe ser 0.0943539. Este es el valor de la densidad de una normal en $x = (1, 1)'$ de media $\mu = (0, 0)'$, y matriz de covarianzas V con varianzas iguales a 1 y covarianza $1/2$.

La función de distribución en $(1, 1)'$ se puede calcular con:

```
pmvnorm(lower=-Inf,upper=x,mean=mu,sigma=V)
```

El resultado debe ser $F(1, 1) = \Pr(X < 1, Y < 1) = 0.7452036$. Este comando también se puede usar para calcular (aproximar) las probabilidades en rectángulos dando los límites inferiores y superiores del rectángulo. Por ejemplo, para calcular $\Pr(-1 < X < 1, -1 < Y < 1) = 0.4979718$ haremos

```
pmvnorm(lower=c(-1,-1),upper=x,mean=mu,sigma=V)
```

Para realizar su gráfica podemos hacer:

```
f<-function(x,y)
dmvnorm(data.frame(x,y),mu,sigma=V)
x<-seq(-3,3,length=50)
y<-seq(-3,3,length=50)
z<-outer(x,y,f)
persp(x,y,z,xlab='x',ylab='y',zlab='f(x,y)',col='red')
```

[1]

[1]Nota: Al copiar y pegar estos comandos desde un fichero pdf las comillas pueden no ser reconocidas por R.

La gráfica se puede ver en la Figura 1, izquierda.

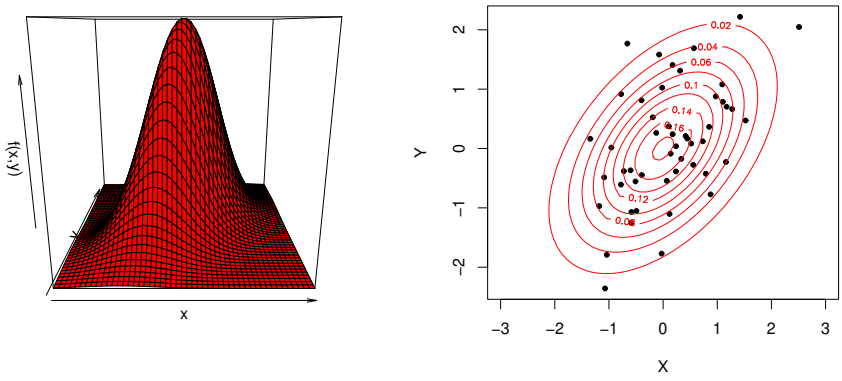

Figura 1: Función de densidad Normal bivariante con medias cero, varianzas uno y correlación 0.5.

Para generar (simular) 50 datos de este modelo haremos:

```
set.seed(123)
d<-rmvnorm(50,mu,V)
```

Recordamos que el primero comando sirve para que todos obtengamos la misma muestra. El primer dato debe ser $(-0.6009522, -0.3673962)$. Podemos representarlos junto con la curvas de nivel de f mediante:

```
plot(d,xlab="X1",ylab="X2",pch=20,xlim=c(-3,3),lim=c(-3,3))
contour(x,y,z,add=TRUE, col='red')
```

El resultado se puede ver en la Figura 1, derecha. En esta gráfica podemos ver como los datos se sitúan alrededor de la media $(0, 0)$ en elipses con diagonal principal en la recta $y = x$. Conforme baja el nivel de f, los datos disminuyen. También observamos una dependencia positiva débil con correlación lineal $\rho = 0.5$. En este modelo X explica linealmente un $\rho^2 = 0.25$ (un 25 %) de Y.

Para calcular algunas medidas descriptivas de los datos simulados haremos: `summary(d)` obteniendo:

	X	Y
Min.	-1.3447	-2.35757
1st Qu.	-0.5666	-0.47686
Median	0.1285	0.10245
Mean	0.1249	0.09652
3rd Qu.	0.6911	0.76620
Max.	2.5095	2.21893

Tabla 1: Resumen de los datos simulados de un normal bivariante.

Para cada variable, se calculan el mínimo, el primer cuartil, la mediana, la media, el tercer cuartil y el máximo. La matriz de cuasi-covarianzas muestrales se obtiene mediante: `cov(d)`. El resultado

debe ser:

$$S = \begin{pmatrix} 0.6843707 & 0.4133264 \\ 0.4133264 & 0.9909406 \end{pmatrix}$$

La matriz de correlaciones muestrales se calcula con `cor(d)`. La cuasivarianzas también se calculan con `var(d[,1])` y `var(d[,2])` y la cuasi-covarianza con `cov(d[,1],d[,2])`. Compruebe que las fórmulas coinciden con las vistas en teoría. Por ejemplo, la cuasi-covarianza se calcula con

```
mu1<-mean(d[,1])
mu2<-mean(d[,2])
(1/49)*sum( (d[,1]-mu1)*(d[,2]-mu2))
```

Comprobamos como los valores muestrales se parecen a los teóricos pero no mucho ya que el tamaño muestral es pequeño ($n = 50$).

Para realizar un test de normalidad multivariante (Shapiro-Wilk) primero cargaremos este paquete `library(mvnormtest)` y después haremos: `mshapiro.test(t(d))` obteniendo el P-valor 0.6014 que lógicamente conduce a la aceptación de la hipótesis nula $H_0 : (X, Y) \equiv N(\mu, V)$. En algunas técnicas multivariantes esta hipótesis de normalidad es importante y debe ser contrastada. En otras no será necesario.

Por último mostraremos como calcular las distancias de Mahalanobis de los datos a la media (teórica o muestral). Esta distancia tiene en cuenta las diferentes escalas de los datos y sus correlaciones y nos servirá para detectar las observaciones más "raras" (alejadas de la media) que podrían ser observaciones atípicas (outliers) que no provengan de nuestra población o contengan errores. Cuando se pueda, se deberán chequear y, si es posible, corregir o eliminar. En otros casos, se deberán mantener por ser observaciones correctas que hay que tener en cuenta. Para calcular las distancias al cuadrado a ambas medias haremos:

```
dM1<-mahalanobis(d,mu,V)
dM2<-mahalanobis(d,colMeans(d),cov(d))
```

Note que en la práctica las medias y covarianzas serán desconocidas por lo que solo dispondremos de la segunda opción. Si queremos detectar la observación más rara haremos:

```
max(dM1)
which.max(dM1)
max(dM2)
which.max(dM2)
```

En ambos casos la más rara es la observación 49 con distancias al cuadrado a las medias de 7.133321 y 8.659063, respectivamente. Sus medidas son $(2.509470, 2.046512)$ (ver Figura 1). Para ver todas las distancias y ordenarlas podemos hacer:

```
View(data.frame(d[,1],d[,2],dM1,dM2))
```

Así observamos que la segunda observación más alejada de la media es la de la línea 22 con medidas $(-0.6609119, 1.7675419)$.

3. Datos multivariantes

El programa R (y algunos de sus paquetes) dispone de algunos ficheros de datos cargados. Por ejemplo si tecleamos `iris` podemos ver una tabla de datos con medidas de flores iris de distintas

especies. Para ver la fuente de donde se obtienen estos datos podemos hacer `help(iris)`.

Si queremos que estos datos aparezcan en la ventana superior derecha podemos hacer `d<-iris`. De esta forma, observamos que este fichero tiene 150 observaciones de 5 variables. Note que hemos borrado los datos normales de la primera sección al guardar estos datos en `d`. Para ver el tipo de variables pinchamos sobre el icono con un círculo azul y una flecha blanca junto a `d`. De esta forma, podemos ver que hay 4 variables numéricas y otra con tres factores indicando las distintas especies. Los datos se pueden abrir en una ventana nueva con `View(d)` (también se puede hacer pinchando sobre `d`). Pinchando sobre las flechas de cada columna podemos reordenar esta tabla con respecto a cada columna. Así, por ejemplo, podemos comprobar que la flor con la longitud de sépalo más pequeño es la de la fila 14 y pertenece a la especie "setosa".

Recordemos que podemos trabajar con los datos de cada columna usando el símbolo $. Por ejemplo, `mean(d$Sepal.Length)` nos proporcionará la media de esta variable (5.843333). También podemos usar $d[,1]$ como en la sección anterior. Para calcular el vector de medias y la matriz de covarianzas muestrales haremos

```
mu<-colMeans(d[,1:4])
mu
V<-cov(d[,1:4])
V
```

Para calcular varias medidas descriptivas sobre todas las variables podemos hacer `summary(d)` obteniendo:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min.	4.300	2.000	1.000	0.100	setosa 50
1st Qu.	5.100	2.800	1.600	0.300	versicolor 50
Median	5.800	3.000	4.350	1.300	virginica 50
Mean	5.843	3.057	3.758	1.199	
3rd Qu.	6.400	3.300	5.100	1.800	
Max.	7.900	4.400	6.900	2.500	

Tabla 2: Resumen de los datos en iris

Al detectar que la última variable es cualitativa, R simplemente proporciona el conteo de cada uno de los casos (factores). Al tener distintas especies, los datos pueden provenir de poblaciones diferentes por lo que debemos calcular las diferentes medidas en cada grupo. En este fichero tenemos 3 muestras de tamaño 50 de cada especie. Por ejemplo, las medias de la primera columna se deben calcular con

```
tapply(d$Sepal.Length,d$Species,mean)
```

Así obtenemos las medias 5.006 (setosa), 5.936 (versicolor) y 6.588 (virginica) para la primera variable. Para calcular el resto de medidas podemos hacer:

```
tapply(d$Sepal.Length,d$Species,summary)
```

Las diferencias entre las distintas especies se pueden visualizar con las gráficos caja-bigote que representan los cuartiles (cajas) y los valores extremos (bigotes). Para ello haremos:

```
boxplot(d$Sepal.Length~d$Species,xlab='Especies',ylab='Long.sepalo')
```

Así se obtiene la gráfica de la Figura 2, izquierda. Observamos claras diferencias entre las distintas especies. Las flores setosa tienen sépalos más cortos y las virginica más largos. En esta última especie se aprecia un posible dato atípico (outlier) que podría pertenecer a la especie setosa.

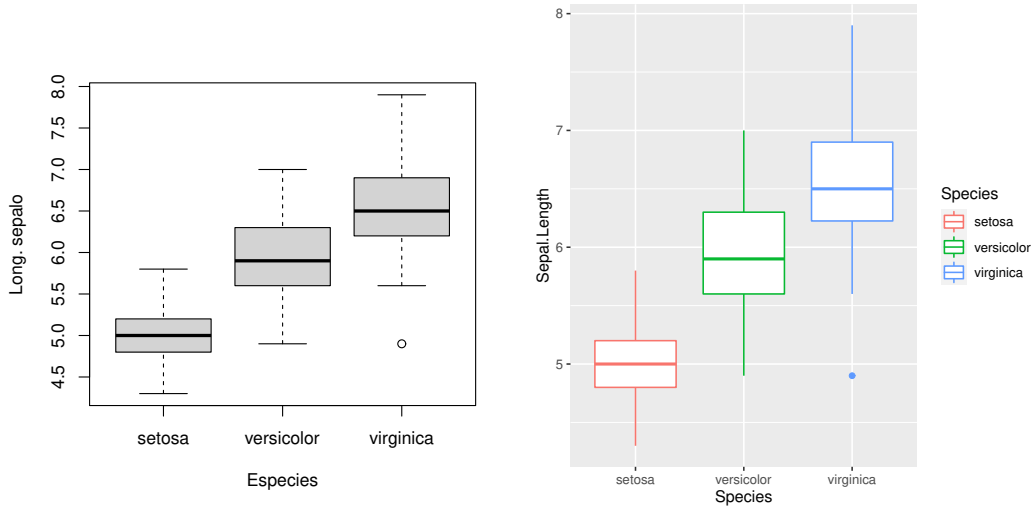

Figura 2: Gráficos caja-bigote para la variable "Sepal.Length" por especies.

Estos gráficos se pueden mejorar usando el paquete `tidyverse` (ver prácticas de primero). Para cargarlo haremos `library(tidyverse)`. Si no está instalado, deberemos ir a la ventana inferior derecha, pulsar sobre la pestaña "Packages" y sobre "Install" para instalarlo. Tras esto debemos ejecutar de nuevo `library(tidyverse)` para cargarlo. Ahora ya podemos teclear:

```
d%>%
ggplot(aes(x=Species,y=Sepal.Length)) +
geom_boxplot(aes(color=Species))
```

De esta forma se obtiene la gráfica de la Figura 2, derecha. Practique realizando estos gráficos para las otras variables. Estos gráficos demuestran que en realidad tenemos $k = 4$ variables medidas en tres poblaciones (especies) distintas. Por esto el test de normalidad multivariante puede fallar. Así, con

```
mshapiro.test(t(d[,1:4]))
```

se obtiene un P-valor de 0.02342 que nos conduce a rechazar la normalidad de los datos conjuntos con las tres especies. Para separar los datos de las tres especies podemos usar:

```
list <- split(d,d$Species)
list2env(list,.GlobalEnv)
```

De esta forma se crean tres tablas de datos (una para cada especie) con los nombres de cada especie. A pesar de ésto, si realizamos los test de normalidad para cada especie, podemos comprobar que la única que pasa el test es la especie setosa con un P-valor de 0.07906 (si tomamos $\alpha = 0.05$ lo pasa por poco, es decir hay sospechas de que tampoco es normal). Para la especie virginica podríamos sospechar que la falta de normalidad se debe a la observación atípica que está en la línea 107. Si la eliminamos con

```
mshapiro.test(t(virginica[c(1:6,8:50),1:4]))
```

el P-valor es 0.008909 y la normalidad sigue fallando.

Tras estudiar las variables por separado podemos estudiarlas juntas para ver sus diferencias y correlaciones. Las gráficas de puntos (scatter plots) se pueden obtener con: `plot(d[,1:4])`. El resultado se puede ver en la Figura 3, izquierda. Se aprecian correlaciones positivas y grupos en muchas gráficas. Para ver si estos grupos se deben a las distintas especies podemos hacer:

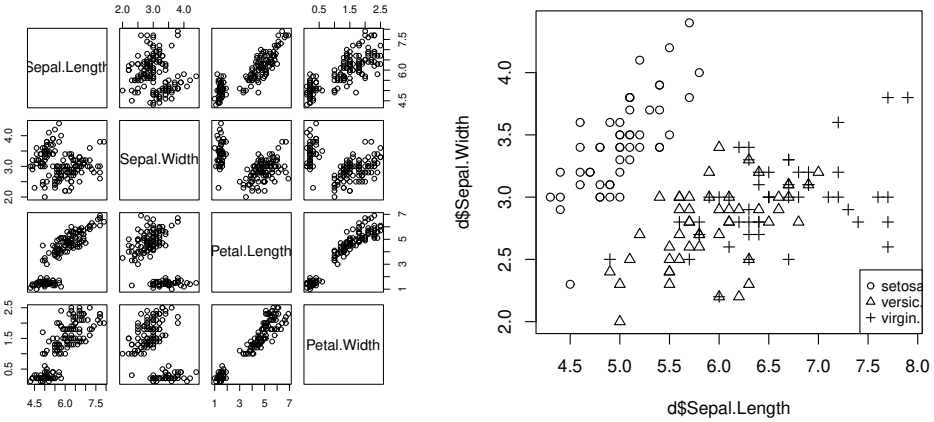

Figura 3: Gráficos de puntos de todas las variables numéricas de iris.

```
plot(d$Sepal.Length,d$Sepal.Width,pch=as.integer(d$Species))
legend('bottomright',legend=c('setosa','versic.','virgin.'),pch=1:3,cex=0.8)
```

En este gráfico, que se puede ver en la Figura 3, derecha, observamos claras diferencia entre los grupos y observaciones atípicas en todos ellos. El grupo setosa aparece en la parte superior izquierda claramente separado de los otros dos que están más mezclados. Analice los gráficos correspondientes a los pétalos. Note que los grupos se asignan por orden alfabético y que el comando `cex=0.8` sirve para que la leyenda sea más pequeña y no tape puntos de los datos.

Si preferimos diferenciar los grupos por colores podemos hacer:

```
plot(d$Sepal.Length,d$Sepal.Width,pch=20,xlab='Long. sépalo',ylab='Anchura sépalo')
points(versicolor$Sepal.Length,versicolor$Sepal.Width,pch=20,col='red')
points(virginica$Sepal.Length,virginica$Sepal.Width,pch=20,col='blue')
legend('bottomright',legend=c('setosa negro','versic. rojo','virgin. azul'),cex=0.5)
```

Los gráficos para los pétalos se hacen de forma similar. Los resultados se pueden ver en la Figura 4. De nuevo observamos que las flores setosa aparecen muy separadas de las de las otras especies. Los pétalos tienen una clara correlación positiva en todos los grupos y también separan a las de las especies versicolor y virginica.

En estos gráficos no se aprecian observaciones atípicas. Para detectarlas podemos usar la distancia de Mahalanobis a la media. Al usar datos reales tendremos que usar el vetores de medias y la matriz de varianzas muestrales (ya que las teóricas son desconocidas). Por ejemplo, podemos hacer:

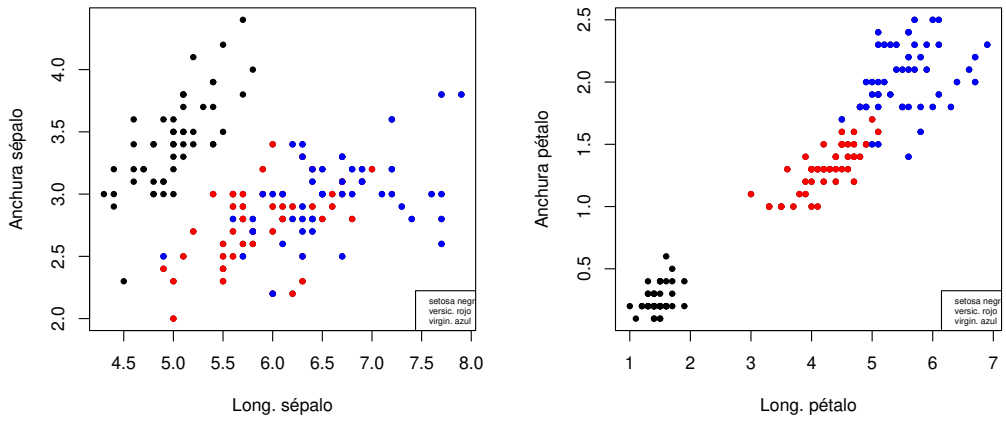

Figura 4: Gráficos de puntos de todas las variables numéricas de iris por grupos.

```
Msetosa<-mahalanobis(setosa[,1:4],colMeans(setosa[,1:4]),cov(setosa[,1:4]))
sort(Msetosa)
```

Así, observamos que las observaciones más raras de la especie setosa son las de las líneas 42 y 44 con distancias al cuadrado iguales a 12.3276387 y 12.3100577, respectivamente. La más cercana a la media es la de la línea 8 con 0.3434392. Estudie las distancias en las otras especies. Si queremos señalar a la flor 42 en el gráfico de la izquierda añadiremos el comando:

```
text(d[42,1],d[42,2],'42',cex=0.8, pos=3)
```

La verosimilitud de un dato en un modelo estadístico será el valor de la función de densidad (función de probabilidad si es un modelo discreto) en ese punto. Por ejemplo, si asumimos distribuciones normales en los tres grupos, la verosimilitud de la flor 42 en el grupo setosa se calculará con:

```
mu1<-colMeans(setosa[,1:4])
V1<-cov(setosa[,1:4])
dmvnorm(d[42,1:4],mu1,V1)
```

Se debe obtener 0.03666644. Calcule su verosimilitud en las otras especies y compruebe que son más pequeñas.

4. Guardar y leer objetos y datos

Recordamos algunos comandos vistos en asignaturas anteriores. En primer lugar hay que comentar que el programa tiene un directorio activo que es donde realizará todas las operaciones por defecto (si no se le indica otra cosa). Si queremos cambiarlo debemos seleccionar en el menú superior:

Session>Set Working Directory>Choose Directory

También se puede hacer con el comando `setwd('c:/nombre')` indicando la "ruta" completa del disco *c*. En la pestaña "Files" de la ventana inferior derecha, podemos ver los ficheros disponibles en este directorio.

Si queremos guardar el objeto `d` (constante, vector, dataframe, etc.) en ese directorio usaremos

```
dump('d','DatosPractica1.R')
```

De esta forma, hemos guardado el dataframe `d` en ese fichero. Ahora podemos borrarlo con `rm(d)`. Tecleando `d` vemos que este objeto ahora no contiene ningún dato. Para recuperarlo podemos usar

```
source('DatosPractica1.R')
```

Se guardará en `d` por lo que si ese objeto tiene algo, lo borrará. Otra forma de hacer esto es pinchar en la ventana inferior derecha sobre el archivo. De esta forma se abre una nueva ventana (script) y si marcamos el comando y lo ejecutamos con `Run`, se cargará ese objeto.

Para ver el conjunto de objetos activos usaremos `ls( )`. Si queremos guardarlos todos usaremos

```
dump(ls(),'ObjetosP1.R')
```

Ahora podemos borrar todos los objetos usando la escoba o `rm(list=ls())` y recuperarlos haciendo: `source('ObjetosP1.R')`.

Cuando nos salimos de RStudio, el programa nos pregunta si queremos guardar los objetos activos. Si le decimos que sí, cuando volvamos al programa, podemos cargarlos marcando y pinchando sobre el fichero denominado ".RData" (del directorio de trabajo). Recordemos que los datos/objetos también se pueden guardar en la ventana del escritorio (script) o con la opción: *File>Open File*.

El programa RStudio también puede leer ficheros en otros formatos con la opción

File>Import Dataset

Esta es la mejor opción para leer ficheros de otros programas (Excel, SPSS, SAS, etc.) aunque no siempre la lectura es exitosa.

Podemos guardar y leer los datos de `d` como un fichero *txt* con

```
write.table(d, file='Prueba.txt')
d1<-read.table(file='Prueba.txt', header=TRUE)
```

Aquí hemos elegido un objeto nuevo `d1` para leerlos.

Para leer ficheros *.txt* y *.csv* también podemos utilizar las funciones `read.csv` y `read.csv2`. En estas opciones hay que indicar el fichero, *header=TRUE o FALSE* según si la primera fila contiene o no encabezamientos (nombres de las columnas), *sep=* indicando cómo se separan los datos en el fichero (espacios en blanco, tabulador, coma, punto y coma, etc.), *dec=* indicando qué carácter se utiliza para la separación de los decimales y *fill=* indicando si el fichero tiene celdas en blanco (TRUE). Por ejemplo, podemos leer el fichero: *binary.csv* (descargar del aula virtual) haciendo:

```
datos<- read.csv('binary.csv',header=TRUE)
```

Recuerde que primero debemos descargar este fichero en el directorio activo. Para cambiar el directorio activo usar: Session>Set Working Directory>Choose Directory. Este fichero se usará en prácticas posteriores.

Para guardar un objeto haremos:

```
dump('d','datos1.R')
```

(se guardará en el directorio activo). Para recuperarlo haremos

```
source('datos1.R')
```

(si hay algo en el objeto `d` se borrará y se sustituirá por lo guardado).

Por último, para leer los ficheros con extensión `.rda` usarermos

```
load('decatlon.rda')
```

Los datos se han guardado en el objeto `d`. También se pueden leer con: File>Open File.

5. Ejercicios

Practique leyendo los otros ficheros proporcionados en el aula virtual y realizando un estudio preliminar de algunos de ellos. También puede practicar con los ficheros proporcionados por R.

En todas las prácticas posteriores se realizará un estudio preliminar de los datos antes de aplicarles una técnica de estadística multivariante.