Práctica 1

Francisco Javier Mercader Martínez

COMPLETAR 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 datos multivariantes. Esta parte es fundamental y se repetirá en todas las prácticas posteriores. Usaremos algunas de las técnicas vistas en la asignatura de primero.

1) EL MODELO NORMAL MULTIVARIANTE

Para poder trabajar con la función de densidad y para obtener probabilidades con el modelo Normal multivariante, debemos cargar el paquete "mvtnomr".

```
library(mvtnorm)
```

Consideremos un modelo Normal bivariante (X, Y) con vector de medias (0,0), $Var(X) = 1$, $Var(Y) = 1$ y $Cov(X, Y) = 1/2$. Queremos obtener el valor de la función de densidad de dicho modelo en el punto $x = (1, 1)$, lo mismo con la función de distribución y calcular probabilidades en recintos rectangulares.

Primero definimos el vector de medias (mu) y la matriz de covarianzas (V), así como el punto x donde queremos evaluar densidad y distribución:

```
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)
```

```
## [1] 0.0943539
```

1.1) Función de densidad, función de distribución y probabilidades

Evaluamos la función de densidad en $x = (1, 1)$:

```
dmvnorm(x, mean = mu, sigma = V)
```

```
## [1] 0.0943539
```

Evaluamos la función de distribución en $x = (1, 1)$. Indicar que se trata de una aproximación que R permite calcular a través de diferentes algoritmos:

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

```
## [1] 0.7452036
## attr(,"error")
## [1] 1e-15
## attr(,"msg")
```

```
## [1] "Normal Completion"
```

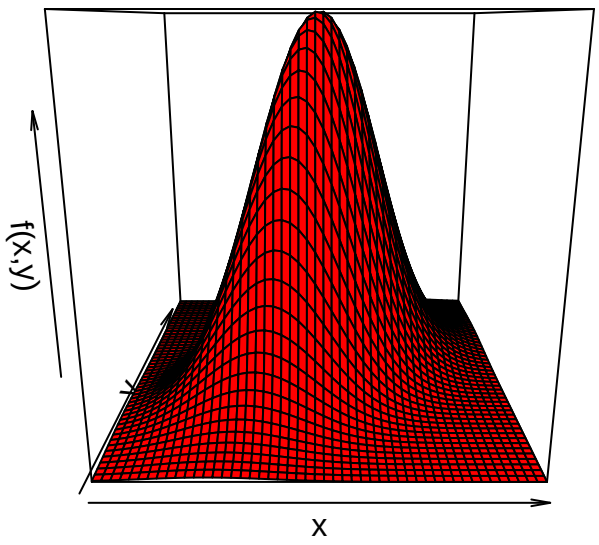
Si queremos calcular la probabilidad en un rectángulo, usaremos la función de distribución indicando los límites inferiores y superiores de cada componente. Por ejemplo, la probabilidad $Pr(-1 < X < 1, -1 < Y < 1)$ se obtendría tecleando:

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

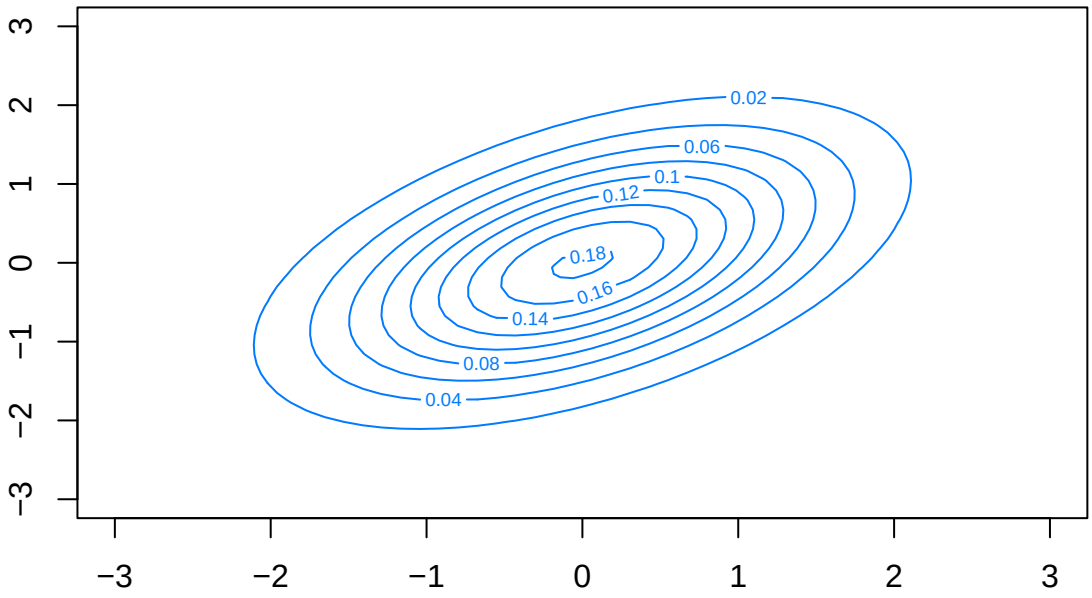
```
## [1] 0.4979718
## attr(,"error")
## [1] 1e-15
## attr(,"msg")
## [1] "Normal Completion"
```

Para representar la función de densidad el modelo Normal bivariante con el que estamos trabajando haremos:

```
f<-function(x,y) {
  dmvnorm(data.frame(x,y),mu,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')
```




```
contour(x,y,z,col='#007AFF')
```



1.2) Generar valores aleatorios del modelo Normal multivariante

Para generar (simular) 50 pares de datos de nuestro modelo Normal bivariante haremos:

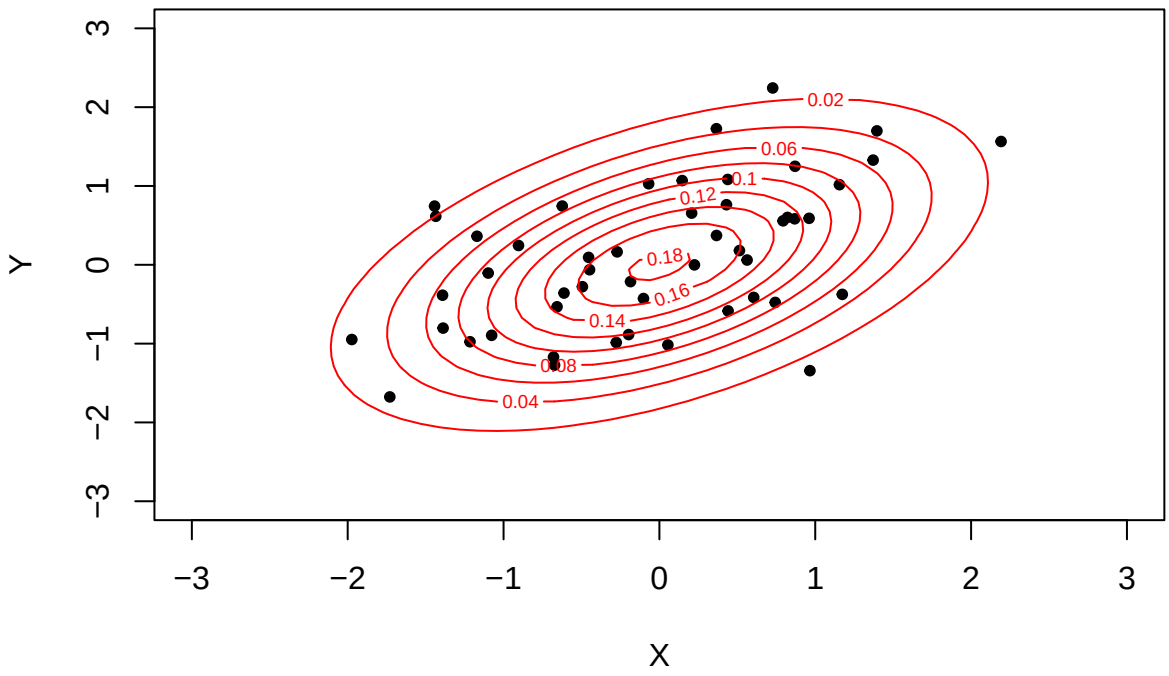
```
library(MASS)
set.seed(123)
d <- mvrnorm(n=50, mu=mu, Sigma = V)
print(d)
```

```
##              [,1]          [,2]
##  [1,] -0.61204541 -0.3587268911
##  [2,] -0.18506618 -0.2136129309
##  [3,]  1.37131623  1.3284457685
##  [4,] -0.62323908  0.7453632002
##  [5,]  0.22485196 -0.0009190298
##  [6,]  0.72705455  2.2435251500
##  [7,]  1.17354155 -0.3752112587
##  [8,] -1.38788204 -0.8032682917
##  [9,] -0.65675914 -0.5329048965
## [10,] -0.49392537 -0.2779838032
## [11,]  0.87026619  1.2499056743
## [12,]  0.56276964  0.0604461883
## [13,]  0.51368195  0.1804745655
## [14,]  0.60514174 -0.4134336478
## [15,]  0.05452307 -1.0172681564
## [16,]  1.39574785  1.6992764915
## [17,]  0.20704627  0.6552560507
```

```
## [18,] -1.72964253 -1.6766383038
## [19,]  0.14625829  1.0685257618
## [20,] -1.43449171  0.6155929730
## [21,] -0.67924687 -1.1702780392
## [22,]  0.96581262 -1.3433562513
## [23,] -1.39141518 -0.3856766544
## [24,] -0.27663794 -0.9858387025
## [25,] -0.19729558 -0.8853041926
## [26,] -1.97350494 -0.9479335706
## [27,]  0.86793137  0.5831583600
## [28,]  0.74318387 -0.4775338398
## [29,] -1.07630724 -0.8950037606
## [30,]  1.15528125  1.0163898921
## [31,]  0.36644676  0.3722109426
## [32,] -0.44817960 -0.0628991996
## [33,]  0.96053158  0.5898715461
## [34,]  0.43829763  1.0826741824
## [35,]  0.82175337  0.6012668071
## [36,]  0.43048897  0.7622709361
## [37,] -0.06871275  1.0281262661
## [38,] -0.27120786  0.1639736313
## [39,] -0.10200565 -0.4279372322
## [40,] -0.90390136  0.2449062569
## [41,] -1.09838582 -0.1048819640
## [42,] -0.45426012  0.0941368351
## [43,] -1.21523125 -0.9764995188
## [44,]  2.19232400  1.5644179277
## [45,]  0.36579955  1.7264520016
## [46,] -0.67251077 -1.2727703578
## [47,] -1.44257500  0.7447579943
## [48,] -1.17044070  0.3621699220
## [49,]  0.79331979  0.5576194270
## [50,]  0.44101072 -0.5854101796
```

Representemos los datos simulados junto con las curvas de nivel de la densidad teórica:

```
plot(d,xlab="X",ylab="Y",pch=20,xlim=c(-3,3),ylim=c(-3,3))
contour(x,y,z,add=TRUE,col="red")
```



En esta gráfica podemos ver como los datos se sitúan alrededor de la media (0, 0) en elipses con diagonal principal en la recta $y = x$.

Podemos calcular medidas descriptivas de los datos simulados, que deben tomar valores parecidos a los parámetros teóricos del modelo:

```
summary(d)
```

```
##        V1                   V2
##  Min.   :-1.973505   Min.   :-1.67664
##  1st Qu.:-0.668573   1st Qu.:-0.51906
##  Median :-0.007095   Median : 0.07729
##  Mean   :-0.043410   Mean   : 0.10300
##  3rd Qu.: 0.696576   3rd Qu.: 0.72238
##  Max.   : 2.192324   Max.   : 2.24353
```

```
cov(d) # matriz de cuasi-covarianzas muestrales
```

```
##           [,1]      [,2]
## [1,] 0.8739270 0.4379678
## [2,] 0.4379678 0.8218432
```

```
cor(d) # matriz de correlaciones muestrales
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.5167852
## [2,] 0.5167852 1.0000000
```

1.3) Test de normalidad mutivariante (Shapiro-Wilk)

Como ocurre en el caso univariante, también podemos usar un test para contrastar si un conjunto de datos multivariante se puede asumir Normal multivariante. Para ello, necesitamos cargar el paquete "mvnormtest".

```
library(mvnormtest)
```

Y aplicamos la función de R para llevar a cabo el test:

```
mshapiro.test(t(d))  #Necesitamos trasponer los datos del dataframe porque lee
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.98999, p-value = 0.946
```

```
                    #las variables por filas en lugar de por columnas
```

Observar que el $p - valor = 0.6014$, lo que conduce a aceptar que los datos provienen de una distribución Normal (lo cual sabemos que es cierto, pues se generaron a partir de dicho modelo).

1.4) Distancia de Mahalanobis

Veamos cómo calcular la **distancia de Mahalanobis al cuadrado** de cada fila de datos al vector de medias (teórico 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.

```
dM1<-mahalanobis(d, center = mu, cov = V) #dist Mahalanobis al cuadrado
                                          #usando parámetros teóricos del modelo

dM2<- mahalanobis(d, center = colMeans(d), cov = cov(d))                    #dist Mahalano
                                        #usando parámetros muestrales del modelo

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

Si queremos detectar la observación más rara haremos:

```
max(dM1)
```

```
## [1] 5.379774
```

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

```
## [1] 22
```

```
max(dM2)
```

```
## [1] 7.491946
```

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

```
## [1] 22
```

```
d[49,]
```

```
## [1] 0.7933198 0.5576194
```

2) ESTUDIO DESCRIPTIVO DE DATOS MULTIVARIANTES

Cargamos el fichero iris de R y vemos su estructura:

```r
d2 <- iris
str(d2)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Hacemos resumen descriptivo global:

```r
summary(d2)
```

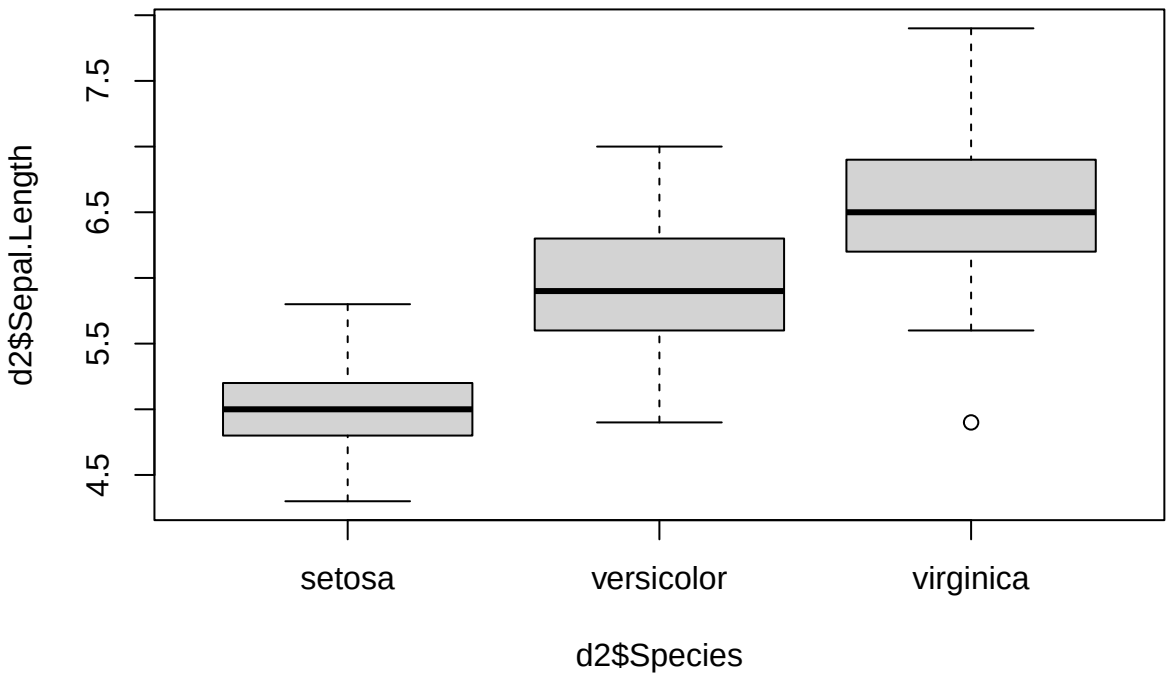
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057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

Resumen numérico por especies para la variable longitud del sépalo:

```r
tapply(d2$Sepal.Length, d2$Species, summary)
```

```
## $setosa
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.300   4.800   5.000   5.006   5.200   5.800
##
## $versicolor
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.900   5.600   5.900   5.936   6.300   7.000
##
## $virginica
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.900   6.225   6.500   6.588   6.900   7.900
```

```r
boxplot(d2$Sepal.Length ~ d2$Species)
```

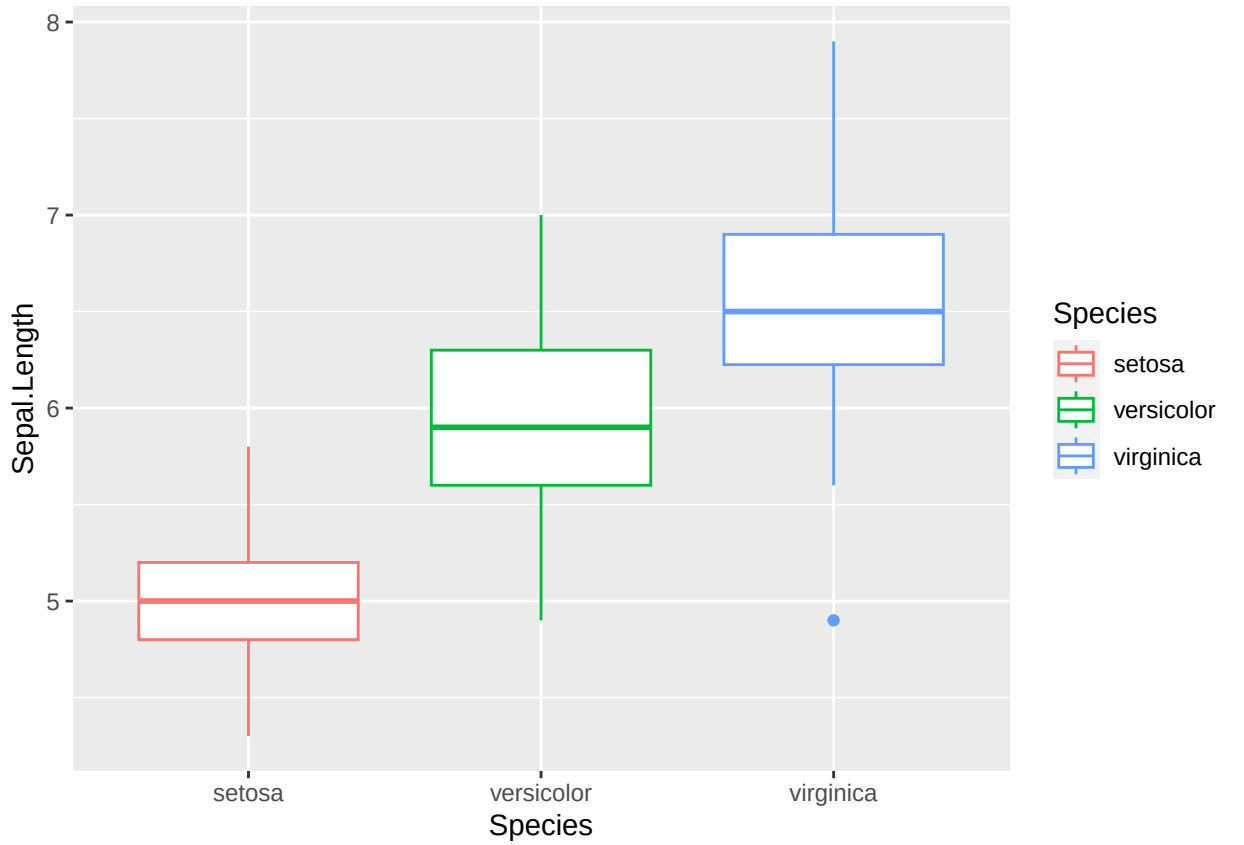


Recordar que también se puede realizar con la colección de paquetes "tidyverse":

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
d2 %>%
  ggplot(aes(x = Species, y = Sepal.Length)) +
  geom_boxplot(aes(color = Species))
```



Estos gráficos demuestran que en realidad tenemos k = 4 variables medidas en tres poblaciones (especies) distintas.

Podemos realizar el test de normalidad multivariante sin distinguir las 3 poblaciones:

```
mshapiro.test(t(d2[,1:4]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.97935, p-value = 0.02342
```

O bien podemos realizar dicho test para cada especie. Primero podemos separar las especies en distintos dataframes:

```
list <- split(d2, d2$Species)
list2env(list, .GlobalEnv)
```

```
## <environment: R_GlobalEnv>
```

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

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.95878, p-value = 0.07906
```

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

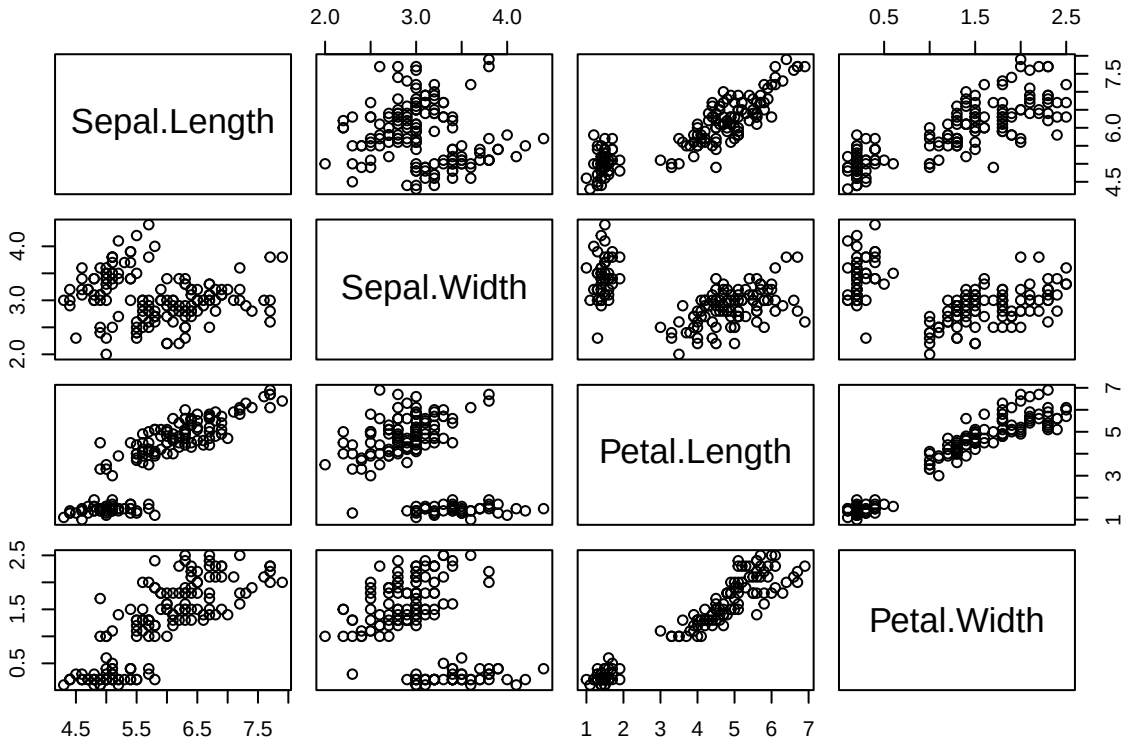
```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.93043, p-value = 0.005739
```

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

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.93414, p-value = 0.007955
```

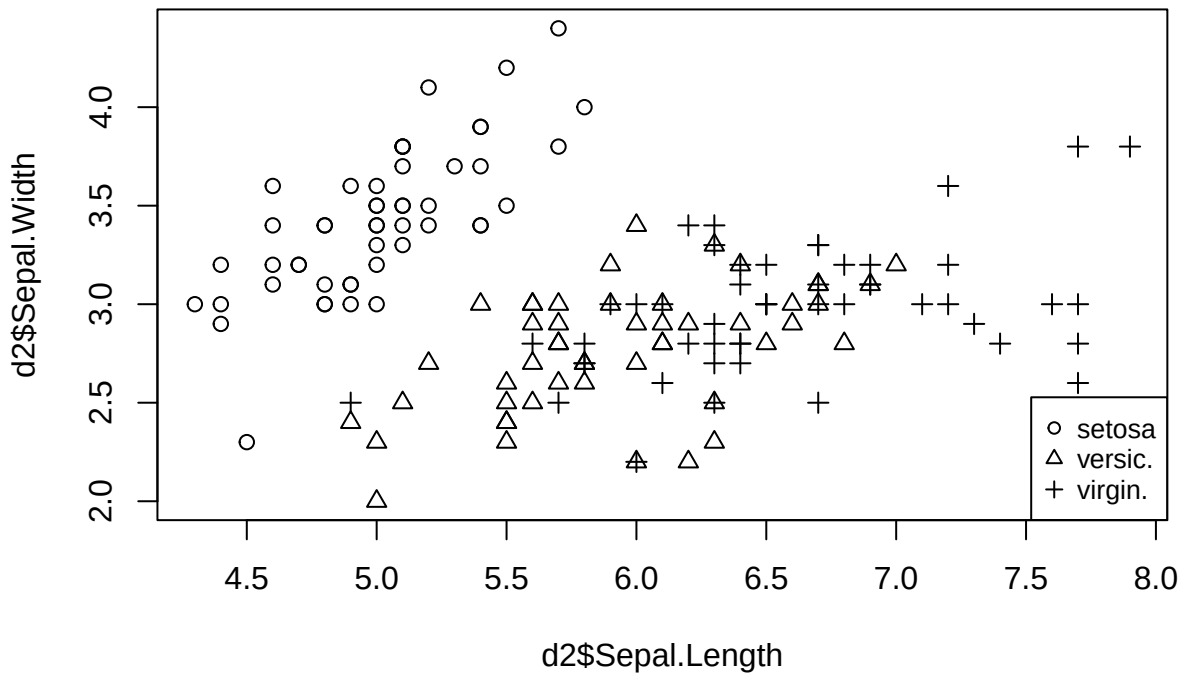
Hacemos las nubes de puntos por pares de variables, sin distinguir especies:

```
plot(d2[, 1:4])
```



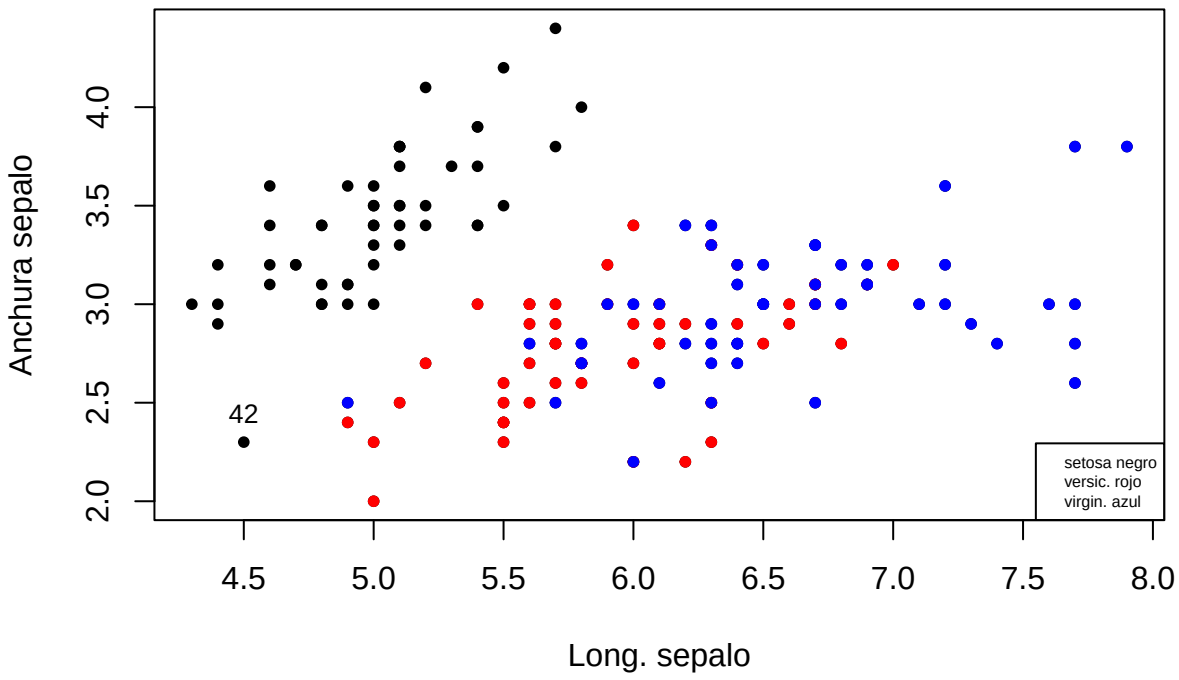
Y ahora algún ejemplo distinguiendo especies:

```
plot(d2$Sepal.Length,d2$Sepal.Width,pch=as.integer(d2$Species))
legend('bottomright',legend=c('setosa','versic.','virgin.'),pch=1:3,cex=0.8)
```



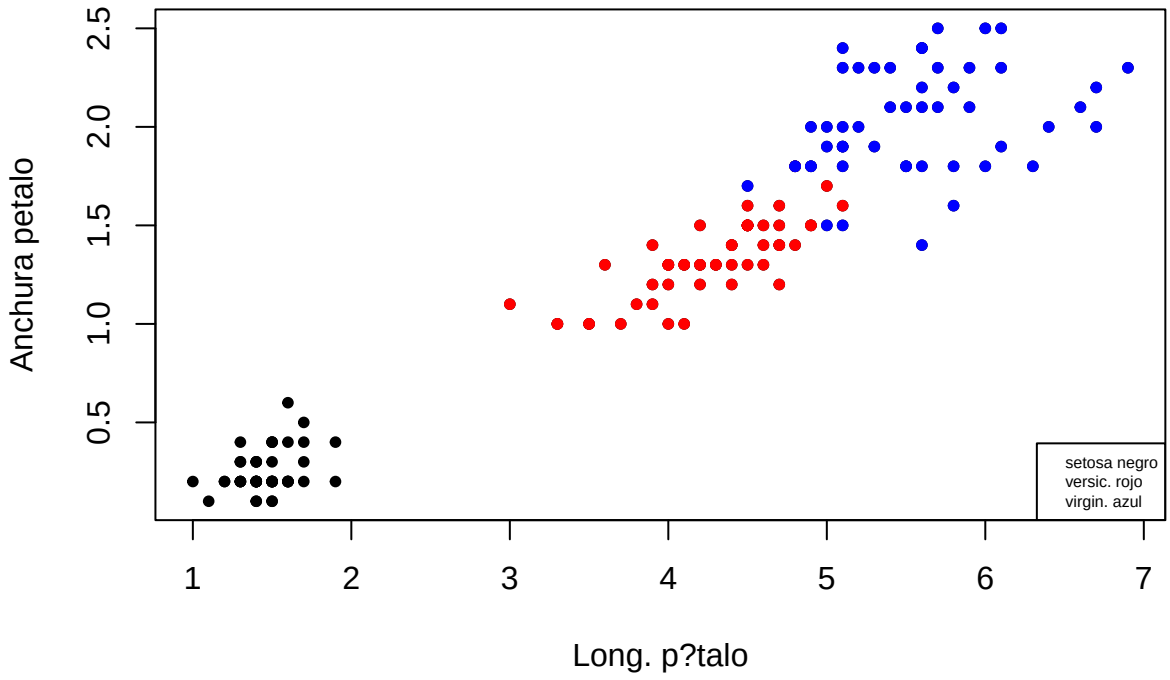
Otro ejemplo distinguiendo especies:

```
plot(d2$Sepal.Length,d2$Sepal.Width,pch=20,xlab='Long. sepalo',ylab='Anchura sepalo')
points(versicolor$Sepal.Length,versicolor$Sepal.Width,pch=20,col='red')
points(virginica$Sepal.Length,virginica$Sepal.Width,pch=20,col='blue')
text(d2[42,1],d2[42,2],'42',cex=0.8,pos=3)
legend('bottomright',legend=c('setosa negro','versic. rojo','virgin. azul'),cex=0.5)
```



Otro más:

```
plot(d2$Petal.Length,d2$Petal.Width,pch=20,xlab='Long. p?talo',ylab='Anchura petalo')
points(versicolor$Petal.Length,versicolor$Petal.Width,pch=20,col='red')
points(virginica$Petal.Length,virginica$Petal.Width,pch=20,col='blue')
legend('bottomright',legend=c('setosa negro','versic. rojo','virgin. azul'),cex=0.5)
```



Calculemos la **distancia de Mahalanobis al cuadrado* para los datos de cada especie respecto a su vector de medias muestral.

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

```
##         8         1        50        40        18         5        28
##  0.3434392  0.4491138  0.4947559  0.5893473  0.6362547  0.7616854  0.8291292
##        49        35         3        29        48        20        41
##  1.2527278  1.2669810  1.2843351  1.3230148  1.4888677  1.6124127  1.6848472
##         4        11        31        12         2        30        46
##  1.7062070  1.8909526  1.9945061  2.0148794  2.0810942  2.1743957  2.1946518
##        27        22        47        13         9        38        10
##  2.5256872  2.7223552  2.7557423  2.9473331  2.9964765  3.0856490  3.2000859
##        39        36         7         6        26        43        32
##  3.2702461  3.3018243  3.4241961  3.7126474  3.7705104  4.2010386  4.8891115
##        19        34        21        37        17        14        24
##  5.1857747  5.2480239  5.3490587  5.7212698  5.7423687  7.0402099  7.2303753
##        16        33        45        25        15        23        44
##  7.6538032  7.6992784  8.6011598  9.7479738 10.2220770 11.0444280 12.3100577
##        42
## 12.3276387
```

```
Mversicolor <- mahalanobis(versicolor[,1:4], colMeans(versicolor[,1:4]), cov(versicolor[,1:4]))
sort(Mversicolor)
```

```
##       100        93        95        98        83        79        97
##  0.3783153  0.4976204  0.7543141  0.7725618  0.8492779  0.9725275  1.0676174
```

```
##         92         70         90         81         75         64         72
##  1.2944391  1.4082478  1.6242992  1.6366303  1.7597667  1.7677621  1.9491202
##         89         52         55         87         62         82         59
##  2.2875030  2.3973078  2.4558493  2.4805283  2.6553116  2.7631832  2.9112176
##         76         57         56         67         53         77         54
##  2.9199945  3.1179808  3.2527816  3.5278668  3.5631825  3.7736387  3.9178610
##         96         66         60         80         94         78         58
##  4.2302567  4.2747500  4.3762012  4.4510573  4.4743536  4.7325800  4.7361110
##         91         65         86         88         85         73         51
##  5.3823881  5.5411161  5.5893623  5.7578365  5.9027716  6.0671797  6.0916551
##         63         61         68         74         84         71         99
##  6.1610471  6.4308470  6.5262626  7.1337117  8.0889349  8.5146136 10.2907920
##         69
## 12.4894655
```

```
Mvirginica <- mahalanobis(virginica[,1:4], colMeans(virginica[,1:4]), cov(virginica[,1:4]))
sort(Mvirginica)
```

```
##        103        113        112        129        148        117        121
##  0.8333777  0.8501882  0.8882618  1.0711906  1.1118083  1.1395765  1.2650472
##        125        144        105        133        116        102        143
##  1.2983426  1.3352612  1.5855332  1.7500469  1.8765129  1.8944347  1.8944347
##        104        138        141        140        128        122        150
##  2.1093471  2.1692940  2.1721288  2.3114137  2.5594497  2.6139404  2.6910812
##        111        139        145        124        126        127        131
##  2.8049311  3.0674703  3.1401967  3.1866510  3.1886575  3.3050360  3.3292942
##        109        114        108        149        147        106        134
##  3.4851203  3.6451283  3.6681851  3.9418920  4.0076980  4.1651756  4.2847010
##        110        137        146        130        115        136        120
##  4.4487424  4.4726436  4.5454882  5.0318804  5.6123019  6.3042090  6.4187616
##        123        107        142        135        101        118        132
##  6.6556667  7.9943130  8.3131469  8.7844642  8.8020901  9.4800318 10.8263574
##        119
## 13.6690945
```

Observamos que la fila 42 es la que tiene mayor distancia, dentro de la especie setosa.

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(setosa[42, 1:4], mean=mu1, sigma = V1)
```

```
##         42
## 0.03666644
```