Práctica 1

Víctor de Juan

2015-09-30

```
head(iris)
datos <- iris[1:50,-5]
plot(pairs(datos))
```

1

Calcula el vector de medias muestral y las matrices de covarianzas y de correlaciones (cor) muestrales. ¿Entre qué par de variables es más alta la correlación? ¿Qué variable tiene la mayor varianza?

```
datos <- iris[1:50,-5]
m1 = colMeans(datos)
sig1 = cov(datos)
corest = cor(datos)
```

La correlación más alta es la de la anchura del sépalo con respecto de la longitud del sépalo, como podemos ver en la matriz de correlaciones (σ_{12}).

```
print(corest)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000   0.7425467    0.2671758   0.2780984
## Sepal.Width     0.7425467   1.0000000    0.1777000   0.2327520
## Petal.Length    0.2671758   0.1777000    1.0000000   0.3316300
## Petal.Width     0.2780984   0.2327520    0.3316300   1.0000000
```

La mayor varianza corresponde al valor más alto de la diagonal. Es la anchura del sépalo la que toma este valor, como podemos ver en la diagonal.

```
print(diag(sig1))
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##   0.12424898   0.14368980   0.03015918   0.01110612
```

2

Calcula las distancias de Mahalanobis entre cada uno de los lirios y el vector de medias. Representa los datos, usando el color rojo para el 25 % de los lirios más lejanos al vector de medias.

El 25% de los lirios más lejanos al vector de medias son aquellos cuya distancia de mahalanobis es mayor que el tercer quantil.

Lo primero es calcular la distancia de mahalanobis:

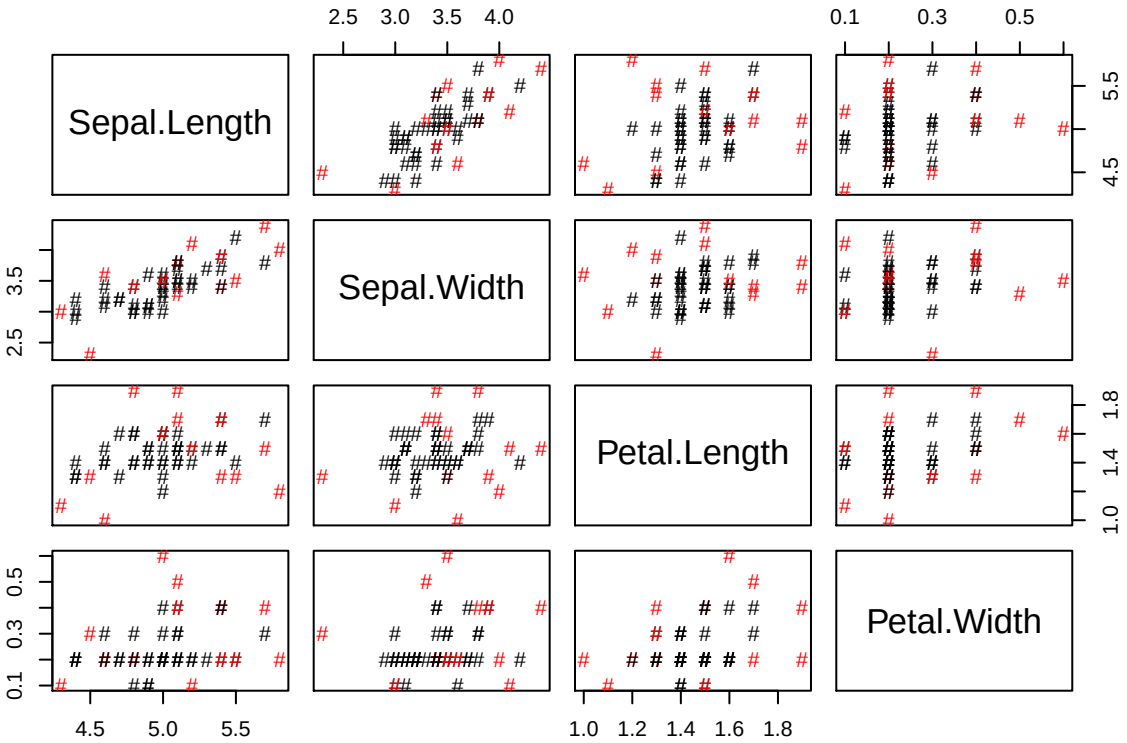
```
distancias <- mahalanobis(datos, m1, sig1)
```

Y el tercer quantil del vector de distancias:

```
q3 = quantile(distancias)[4]
```

Por último, pintamos en rojo aquellos datos que sean mayores que el quantil.

```
plot(x = datos, col=ifelse(distancias>q3,'red','black'),pch="#")
```



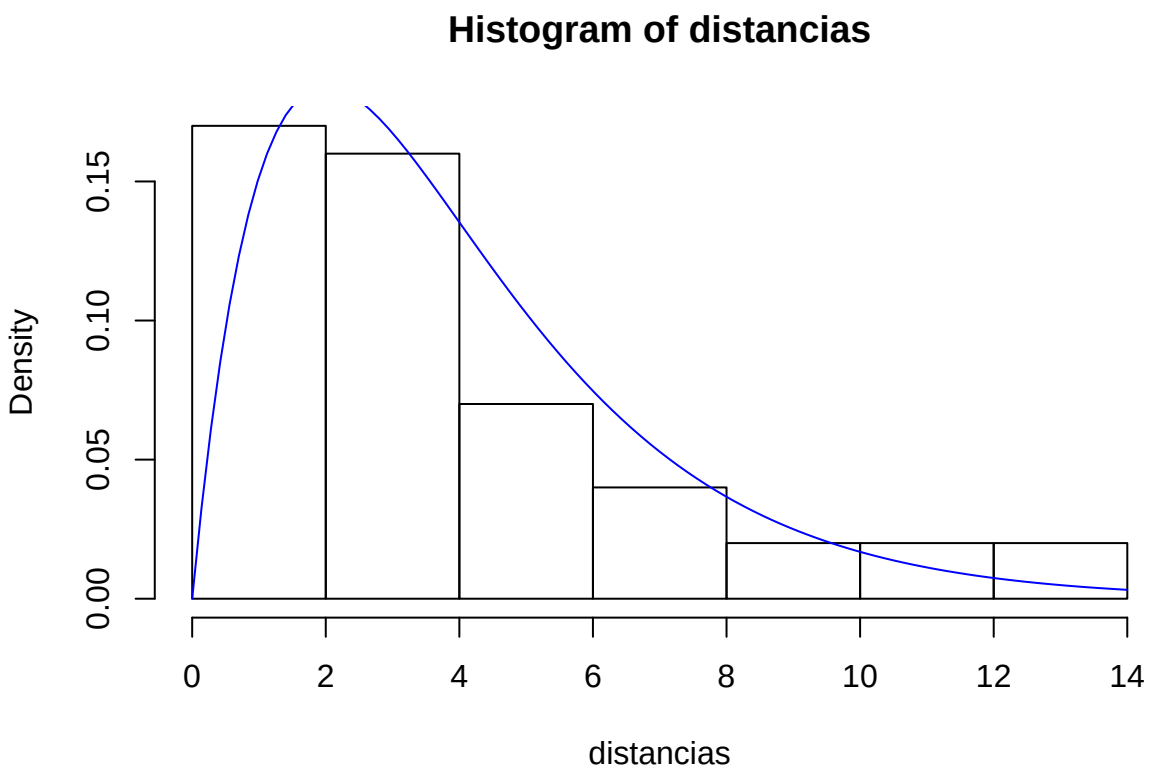
La función ifelse recorre el vector lógico que devuelve la comparación distancias $>$ q3 devolviendo 'red' en caso de encontrar TRUE o 'black' en caso contrario. Estos valores se almacenan como un vector, que utilizamos para colorear gráfico

3

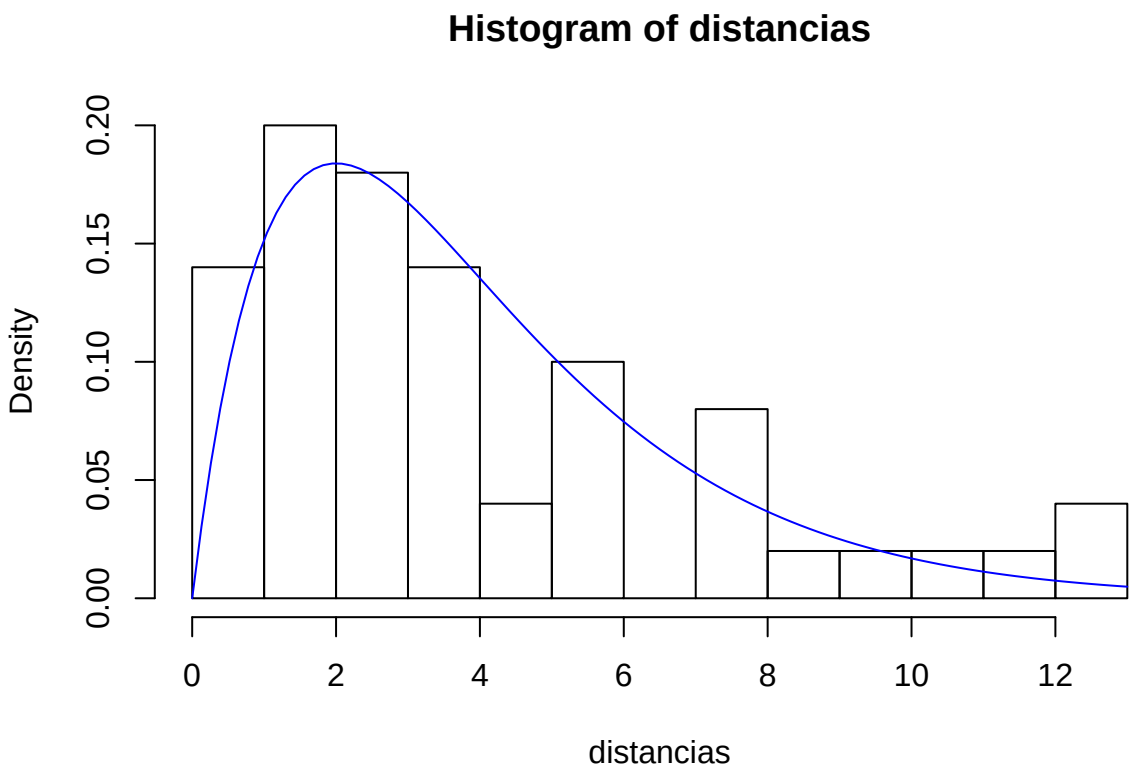
Representa un histograma de las distancias y compáralo con la función de densidad de una variable χ^2 con 4 grados de libertad.

Comparamos la diferencia, dependiendo de las barras de histograma elegidas:

```
hist(distancias, breaks = 8, freq=FALSE)
curve( dchisq(x, df=4), col='blue', add=TRUE)
```



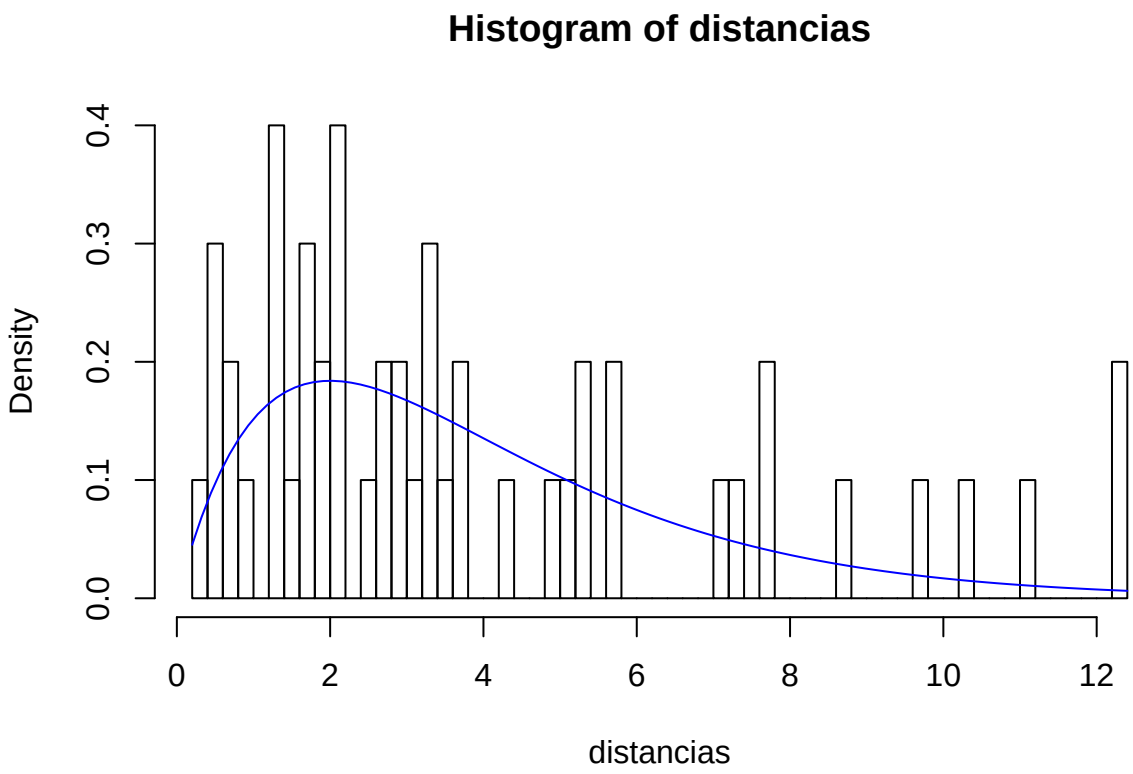

```
hist(distancias, breaks = 16, freq=FALSE)
curve( dchisq(x, df=4), col='blue', add=TRUE)
```



En el caso de 8 breaks, parece que el ajuste es mejor. En cambio, cuando elegimos 16 breaks, se producen algunas situaciones atípicas, como el pico al final del vector, de los lirios que están a una distancia de mahalanobis mayor de 12.

Esto puede deberse a que en un vector de 50 elementos, representar el histograma de frecuencias con 16 breaks es demasiado. Si llevamos el caso al extremo, observamos más la diferencia.

```
hist(distancias, breaks = 50, freq=FALSE)
curve( dchisq(x, df=4), col='blue', add=TRUE)
```


Histogram of distancias
Density
0.0
0.1
0.2
0.3
0.4
0
2
4
6
8
10
12
distancias

4

Genera 100 observaciones con distribución normal bidimensional con vector de medias el origen y matriz de covarianzas:

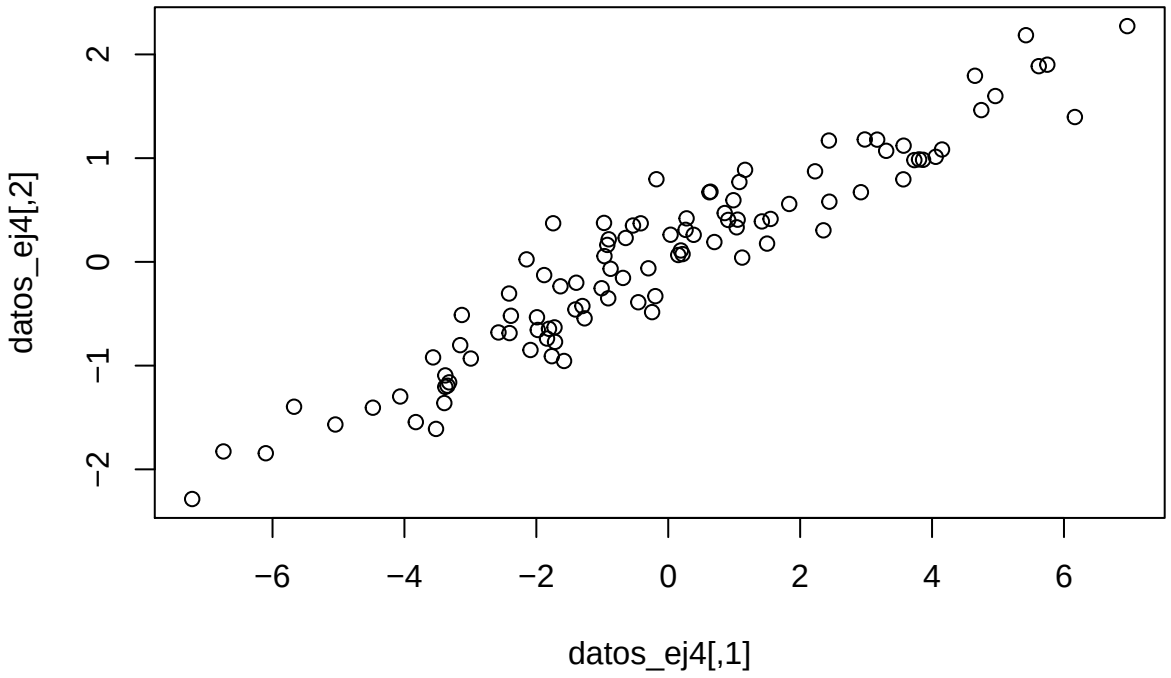
$$sig = \begin{pmatrix} 10 & 3 \\ 3 & 1 \end{pmatrix}$$

Representa la nube de puntos generados, su vector de medias y su matriz de covarianzas.

```
library(MASS)
set.seed(199) #Establecemos una semilla para generar siempre los mismos números aleatorios.

n <-100
m <- c(0,0)
sig <-matrix(c(10,3,3,1),2,2)
datos_ej4 <- mvrnorm(n,m,sig)

plot(datos_ej4)
```




```
colMeans(datos_ej4)
```

```
## [1] -0.14773939  0.02996675
```

```
cov(datos_ej4)
```

```
##          [,1]      [,2]
## [1,] 8.767555 2.6724149
## [2,] 2.672415 0.9032635
```

Podemos observar que la generación de los valores aleatorios no coincide exactamente con el valor esperado para la media (0,0) ni para la matriz de covarianzas, aunque está cerca. Aumentando el número de datos generados, conseguiríamos una mayor precisión

5

Para la misma distribución del apartado anterior, calcula el valor esperado teórico de la segunda coordenada respecto a la primera. Si no lo conocieras y solo dispusieras de los datos generados, ¿Cómo lo estimarías? Calcula el valor resultante para el estimador que has propuesto.

a) Queremos calcular $\mathbb{E}(X_2|X_1)$. Para ello, utilizamos las fórmulas:

$$m_{2.1} = m_2 + sig_{21}sig_{11}^{-1}(X_1 - m_1)$$
$$\Sigma_{2.1} = sig_{22} - sig_{21}sig_{11}^{-1}sig_{12}$$

Utilizando los datos del enunciado, y utilizando la simetría de sig obtenemos:

```
m_aux = m[1] + sig[2]*sig[1]^(-1)
print(m_aux)
```

```
## [1] 0.3
```

```
sig_aux= sig[4] - sig[2]*sig[1]^(-1)*sig[2]
print(sig_aux)
```

```
## [1] 0.1
```

Con lo que $m_{2.1} = 0.3X_1$ y $sig_{2.1} = 0.1$

b) Por otro lado, el valor obtenido a partir de las observaciones lo calculamos:

```
sig = cov(datos_ej4)
m = colMeans(datos_ej4)
m_aux = m[1] + sig[2]*sig[1]^(-1)
m_aux_2 = m[1] + sig[2]*sig[1]^(-1)*m[1]

sig_aux = sig[4] - sig[2]*sig[1]^(-1)*sig[2]

print(c(m_aux,m_aux_2))
```

```
## [1]  0.1570679 -0.1927714
```

```
print(sig_aux)
```

```
## [1] 0.08869184
```

Con lo que la media condicionada a partir de las observaciones sería: $m_{2.1} = m_{aux}X_1 - m_{aux_2} = 0.15X_1 + 0.2$ y la varianza es 0.08.