Mayo 2018

Cristina Gil Martínez

MÉTODOS DE CLASIFICACIÓN: LDA Y QDA

Apuntes personales sobre Análisis Discriminante Lineal y Cuadrático

CONTENIDO

ANÁLISIS DISCRIMINANTE LINEAL

El análisis discriminante lineal (*Linear Discriminant Analysis o LDA*) es un método alternativo más adecuado a la regresión logística cuando la variable cualitativa tiene más de dos niveles (**K ≥ 2**). Supone también un modelo más estable cuando el tamaño muestral **n** es pequeño y la distribución de los predictores es aproximadamente normal en cada una de sus clases. El propósito del LDA es encontrar la combinación lineal de las variables originales que permita la mejor separación entre grupos de un set de datos.

Teorema de Bayes para la clasificación

El teorema de Bayes es útil para obtener probabilidades condicionales, ya que expresa la probabilidad condicional de un evento aleatorio A dado que otro evento B ya ha ocurrido. Esta probabilidad es igual a la probabilidad conjunta de A y B (*P*(AB)) entre la probabilidad marginal de B (*P*(B)).

$$P(A_j|B) = \frac{\mathrm{P}(A_j \mathrm{B})}{\mathrm{P}(\mathrm{B})} = \frac{\mathrm{P}(A_j)\ x\ P(B|A_j)}{\sum_{i=1}^{n} P\ (A_i)\ x\ P(B|A_j)}$$

donde

P*(A$_j$ | B)** = **probabilidad a *posteriori (*posterior probability*)

P*(A$_j$)** = **probabilidad a *priori (*prior probability*)

En el contexto de la clasificación, mediante el teorema de Bayes se calcula la probabilidad de que la variable respuesta *Y* pertenezca a cada uno de los posibles niveles, dados unos determinados valores de los predictores.

Suponiendo que π_k representa la **probabilidad a *priori*** de que una observación al azar provenga de la clase *k* de la variable respuesta *Y*, y siendo $f_k(X)$ la **función de densidad de probabilidad condicional de *X*** para una observación que proviene de la clase *k*, el teorema de Bayes establece que

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

donde $p_k(x)$ o Pr(Y = *k* |X = *x*) representa la **probabilidad a *posteriori*** de que una observación X = *x* pertenezca a la clase *k*, es decir, la probabilidad dado el valor del predictor (probabilidad condicional). Cada observación se clasificará dentro del nivel que tiene la probabilidad $p_k(x)$ más alta. Además, $f_k(x)$ será mayor

cuanto mayor sea la probabilidad de que la observación de la clase *k* adquiera un valor de X ≈ *x.* Por tanto, el LDA está basado en el clasificador Bayesiano.

Estimar π_k es sencillo si disponemos de una muestra aleatoria de *Y* de la población: simplemente calculamos la fracción de las observaciones que pertenecen a la clase *k*

$$\hat{\pi}_k = n_k/n$$

También deberemos estimar $f_k(x)$ a partir de la muestra, ya que raramente disponemos de los parámetros poblacionales. Esto hace que el clasificador LDA suponga una aproximación al clasificador de Bayes.

LDA con un predictor

A la hora de estimar $f_k(x)$ en un escenario con un solo predictor, deberemos contar con ciertas suposiciones sobre su forma. Suponiendo que asumimos que $f_k(x)$ sigue una **distribución normal**, en un escenario unidimensional, la función de densidad normal tomaría la forma

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)$$

donde μ_k es la media y σ^2_k la varianza para la clase *k,* que han de ser estimados:

$$\begin{aligned}\hat{\mu}_k &= \frac{1}{n_k}\sum_{i:y_i=k} x_i \\ \hat{\sigma}^2 &= \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}(x_i-\hat{\mu}_k)^2\end{aligned}$$

donde ***n*** es el número total de observaciones, y n_k el número de observaciones en la clase *k*.

Si además asumimos que la **varianza en torno a todas las clases *K* es igual** ($\sigma^2_1 = ... = \sigma^2_K$), si incluimos la fórmula anterior de $f_k(x)$ en la ecuación de Bayes, obtenemos

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K}\pi_l \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}$$

Mediante la transformación logarítmica de los términos de la ecuación anterior, e incluyendo las estimaciones de todos los parámetros, podemos obtener una versión más simplificada:

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

donde **$\delta_k(x)$** es el log de $p_k(x)$.

Con esta última ecuación, el clasificador LDA asigna una observación X = *x* a la clase que maximiza $\delta_k(x)$. La **función** discriminante $\delta_k(x)$ es **lineal** respecto de x, por ello el término "lineal" en el nombre del LDA, y por ello generará **límites de decisión lineales**.

LÍMITE DE DECISIÓN DE BAYES VS LDA

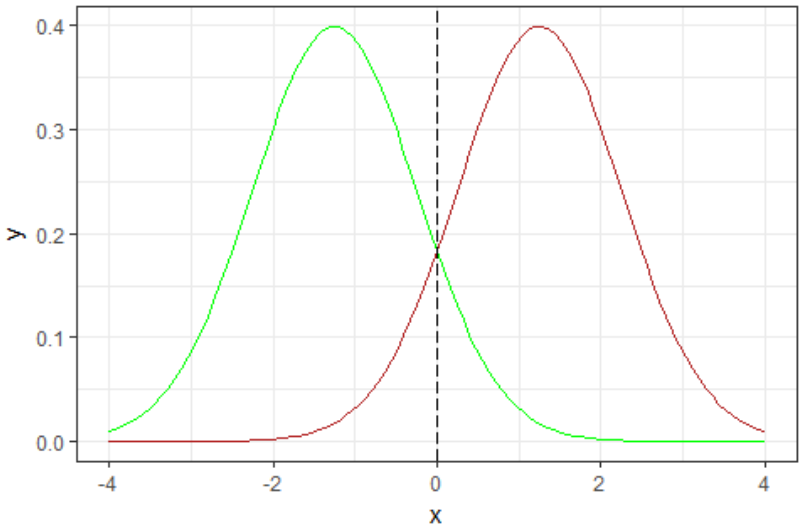
Representemos un ejemplo gráfico para entender los conceptos descritos hasta ahora:

Supongamos que contamos con un predictor con K = 2, cuya función de densidad para cada una de las dos clases proviene de una distribución normal $f_1(x)$ con media $\mu_1 = -1{,}25$ y varianza $\sigma^2_1 = 1$ y $f_2(x)$ con media $\mu_2 = -1{,}25$ y varianza $\sigma^2_2 = 1$. Los parámetros poblacionales en este caso son conocidos, así como su distribución. Además, la varianza es común para las dos clases, condición para la aplicación del LDA. Suponiendo además que la probabilidad a *priori* (π_k) de que una observación sea de una clase u otra es de 0,5 (en este caso $\pi_1 = \pi_2 = 0{,}5$), el límite de decisión de Bayes se podría calcular como

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2}$$

$$x = \frac{-1{,}25 + 1{,}25}{2} = \mathbf{0}$$

```
library(ggplot2)


ggplot(data = data.frame(x = -4:4), aes(x = x)) +
# distribución de densidad de k = 1
stat_function(fun = dnorm,
              args = list(mean = -1.25, sd = 1),
              color = "green3") +
# distribución de densidad de k = 2
stat_function(fun = dnorm,
              args = list(mean = 1.25, sd = 1),
              color = "firebrick") +
# límite de decisión de Bayes
geom_vline(xintercept = 0,
           linetype = "longdash") +
theme_bw()
```



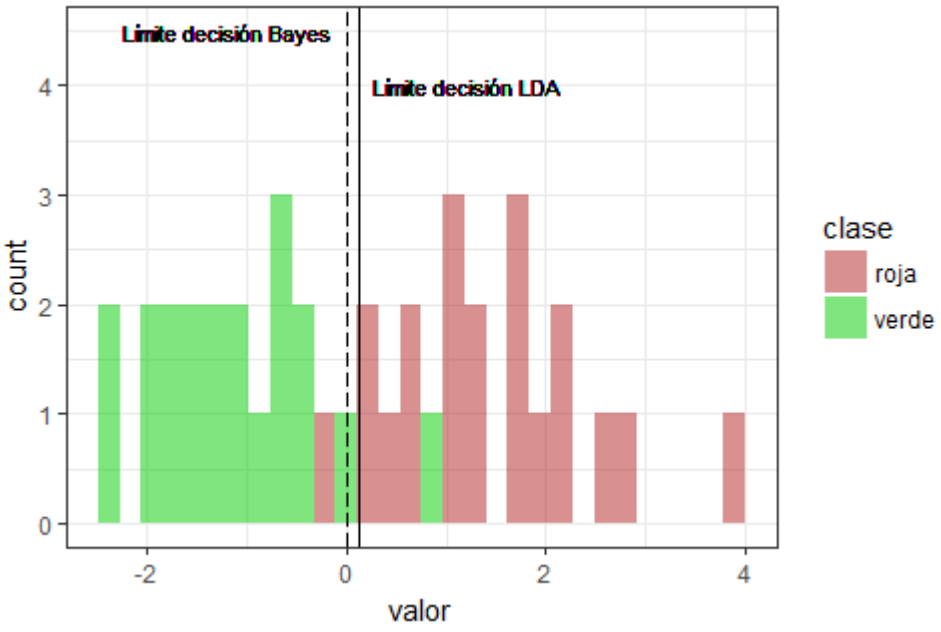
En este caso, el clasificador de Bayes asignará la observación a la clase $k = 1$ (verde) si $x < 0$, y a la clase $k = 2$ (rojo) si $x > 0$.

En la práctica, no conocemos los parámetros poblacionales, aun teniendo cierta seguridad de que X se distribuye de forma normal para cada clase, por lo que no podemos calcular el límite de decisión de Bayes. En este caso, debemos obtener las estimaciones de los parámetros $\mu_1 = ... = \mu_K$, σ^2, y $\pi_1 = ... = \pi_K$ a partir de la muestra.

Supongamos en este caso que disponemos de una muestra aleatoria de 20 observaciones para cada clase ($n_1 = n_2 = 20$). Dado que $n_1 = n_2$, tenemos $\pi_1 = \pi_2$, con lo que en esta situación el límite de decisión corresponderá al punto medio de la media de las dos clases

$$(\hat{\mu}_1 + \hat{\mu}_2)/2$$

```r
set.seed(1303)
muestra.verde <- rnorm(n = 20, mean = -1.25, sd = 1)
muestra.roja <- rnorm(n = 20, mean = 1.25, sd = 1)
datos.muestra <- data.frame(clase = rep(c("verde", "roja"), each = 20),
                            valor = c(muestra.verde, muestra.roja))

ggplot(data = datos.muestra, aes(x = valor, fill = clase)) +
geom_histogram(alpha = 0.5,
               position = "identity") +
scale_fill_manual(values = c("Firebrick", "Green3")) +
geom_vline(xintercept = 0,
           linetype = "longdash") +
geom_vline(xintercept = (mean(muestra.verde) + mean(muestra.roja))/2,
           color = "darkblue") +
geom_text(aes(label = "Límite decisión LDA", x = 1.2, y = 4),
          size = 3,
          colour = "darkblue") +
geom_text(aes(label = "Límite decisión Bayes", x = -1.2, y = 4.5),
          size = 3,
          colour = "black") +
theme_bw()
```



En la figura podemos ver que el límite de decisión LDA está ligeramente desplazado hacia la derecha en relación al límite de decisión de Bayes, pero sigue siendo una aproximación bastante buena.

LDA con múltiples predictores

Podemos extender el LDA para casos con más de un predictor. En este caso ha de asumirse que X = (X_1, X_2, ..., X_p) siguen una distribución **normal multivariante** (los predictores se distribuyen de forma normal cuando se separan para cada clase de la variable que se quiere predecir), con media μ_k específica de la clase y varianza σ^2 común entre clases. Esto también implica la susceptibilidad a posibles *outliers*. Es importante que $p < n$ para evitar problemas de *overfitting*.

Se utiliza el término X ~ N (μ, Σ) para indicar que una variable aleatoria *X* *p*-dimensional sigue una distribución normal multivariante, donde μ es su media y donde **Σ** es su **matriz de covarianza** *p* x *p* común a todas las clases K. La ecuación se define como

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

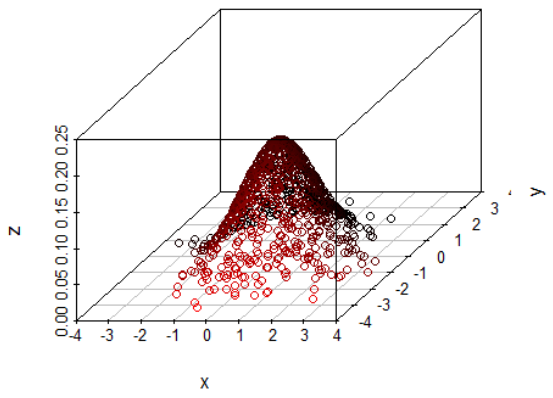
Introduciendo *f(x)* en la ecuación de Bayes, obtenemos un clasificador que asignará la observación X = *x* a la clase para la que se maximice

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log\pi_k$$

De nuevo, desconociendo los parámetros poblacionales no podemos obtener el límite de decisión de Bayes, por lo que tendremos que estimar $\mu_1,...,\mu_k$, $\pi_1,...,\pi_k$ y Σ para obtener los límites de decisión de LDA, que dividen el espacio del predictor en regiones correspondiente a las clases.

Podríamos representar una distribución normal bivariante (2 elementos) de la siguiente manera:

```
library(mvtnorm)
library(scatterplot3d)

sigma.zero <- matrix(c(1,0,0,1), ncol=2)
sigma.zero

##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1

x1000 <- rmvnorm(n=1000, mean=c(0,0), sigma=sigma.zero)
head(x1000, 4)

##             [,1]        [,2]
## [1,]  0.26400733  0.63218681
## [2,] -1.33065099  0.02688882
## [3,]  1.04063632  1.31202380
## [4,] -0.03000208 -0.25002571

scatterplot3d(x1000[,1], x1000[,2],
              dmvnorm(x1000, mean=c(0,0), sigma=sigma.zero),
              highlight=TRUE, xlab = "x", ylab = "y", zlab = "z")
```



Precisión del modelo LDA

El método LDA trata de aproximarse al clasificador de Bayes, el cual comete el menor ratio de error de todos los clasificadores, siempre que se cumpla la condición de normalidad.

Al igual que en la regresión logística, es conveniente obtener la **matriz de confusión** para detectar la cantidad de verdaderos positivos (**sensibilidad**), verdaderos negativos, falsos positivos (**error tipo I**) y falsos

negativos del clasificador LDA, y determinar el error que comete no solo de forma general, sino en cada dirección de predicción (el modelo puede ser mejor o peor prediciendo en una dirección que en otra).

		Clase predicha		
		-	+	Total
Clase verdadera	-	VN	FP	N
	+	FN	VP	P
	Total	N*	P*	N+P

Ratio	
Falsos Positivos	FP/N
Verdaderos Positivos	VP/P
Falsos Negativos	FP/P
Verdaderos Negativos	VN/N

Si nos interesa cometer menos predicciones incorrectas, podemos considerar disminuir el **límite probabilidad a *posteriori*** para el límite de decisión. Por ejemplo, asignar una observación x a la clase k_i si $Pr(K = k_1 \mid X = x) > 0{,}3$, en lugar de $Pr(K = k_1 \mid X = x) > 0{,}5$ (**límite por defecto**). Todo dependerá de qué ratio de error nos interese más, o el coste asociado a cada uno. En resumen, nos interesa conocer el porcentaje de acierto de nuestro clasificador.

La **curva de ROC** es un gráfico útil para representar los dos tipos de errores del modelo (global y por clasificación) para todos los posibles *thresholds*. Una curva de ROC ideal mostraría que para un alto ratio de verdaderos positivos le correspondería un bajo ratio de falsos positivos. El rendimiento global del modelo sobre todos los posibles *thresholds* se corresponde con el **área bajo la curva (AUC)**. Cuanto mayor es AUC, mejor es el clasificador. Un clasificador con AUC = 0,5 no superaría lo esperado por azar (evaluado con datos de test).

Para no obtener un porcentaje de error que subestime el verdadero porcentaje de error que puede cometer el modelo, es importante evaluarlo sobre un grupo de datos de test, y no utilizar los datos de entrenamiento que se han utilizado para entrenar el modelo.

Condiciones para aplicar LDA

Para que el modelo de análisis discriminante lineal se acepte como válido, ha de cumplirse que las observaciones dentro de cada clase siguen una **distribución aproximadamente normal** $N(\mu_k, \sigma^2)$ para casos con un solo predictor, o **normal multivariante** $N(\mu_k, \Sigma)$ para casos con más de un predictor, con media μ_k específica para cada clase y **varianza σ^2 o matriz de covarianza Σ común** entre todas las clases. Ante la falta

de normalidad multivariante, el LDA puede tener cierta robustez, pero es importante tenerlo en cuenta en las conclusiones del análisis.

IMPORTANTE que $p < n$ para evitar problemas de *overfitting*.

Si no se cumple la condición de σ^2 o Σ común, se recurre al Análisis Discriminante Cuadrático.

ANÁLISIS DISCRIMINANTE CUADRÁTICO

El análisis discriminante cuadrático (*Quadratic Discriminant Analysis* o QDA) supone una alternativa al LDA cuando **cada clase tiene su propia matriz de covarianza** (Σ_k). Al igual que LDA, QDA también asume que las observaciones de cada clase siguen una distribución normal multivariante, así como también introduce las estimaciones de los parámetros en la ecuación del teorema de Bayes para obtener las predicciones.

En este caso, una observación de la clase *k* sigue la forma

$$\mathbf{X \sim N(\mu_k, \Sigma_k)}$$

El clasificador asigna una observación X = *x* a la clase para la que la probabilidad a *posteriori*

$$\begin{aligned} \delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log \pi_k \\ &= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + \log \pi_k \end{aligned}$$

sea mayor.

En este caso los parámetros a estimar a partir de las muestras son μ_k, π_k y Σ_k, y la ecuación no es una **función** lineal de *x*, sino **cuadrática** (de donde deriva el nombre del método "cuadrático"). Debido a esta propiedad, el QDA genera **límites de decisión curvos**, no lineales, por lo que es aplicable para casos en los que la separación entre grupos no es lineal. Por ello también, QDA es mucho más flexible que LDA.

EJEMPLO EN R: LDA Y QDA

En este ejemplo trabajaremos con el set de datos *Auto* del paquete *ISLR* y ajustaremos un modelo de LDA y QDA para predecir si el rendimiento del combustible (*gas mileage*) de un automóvil es alto o bajo en función del resto de predictores del set de datos.

Consideraremos como "alto" un valor mayor a la mediana. Para ello, crearemos una variable binaria a partir de la variable binaria *mpg01* que contenga "1" si el valor de *mpg* está por encima de la mediana, y "0" si se encuentra por debajo. Al hacer esto, será importante codificar la variable respuesta como factor. Como en este caso contamos con un predictor con K = 2, la regresión logística también sería una opción.

1. Análisis exploratorio de los datos

- **head()** -> Muestra las primeras observaciones de un vector, matriz, tabla, data frame o función
- **str()** -> Muestra de manera compacta la estructura interna de un objeto
- **summary()** -> Resumen de los datos
- **cor()** -> Calcula los coeficientes de correlación entre variables (solo acepta vectores numéricos)
- **ggpairs()** -> Combina en un único gráfico diagramas de dispersión, distribución de las variables y los valores de correlación
- **pairs()** -> Matriz con gráficos de dispersión de las variables cuantitativas
- **scatterplot3d()**-> Nube de puntos tridimensional
- **kable()**-> Genera una tabla sencilla a partir de una matriz o data frame
- **ggplot()**

```
library(ISLR)
head(Auto, 3)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
##                       name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
```

```
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
```

```
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 1
4 161 141 54 223 241 2 ...
```

```
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year           origin                      name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
##                                                  (Other)           :365
```

```
attach(Auto)
# Vector de “0”s (rendimiento bajo) con la misma longitud que la variable mpg
mpg01 <- rep(0, length(mpg))
# Sustitución de “0”s por “1”s (rendimiento alto) si mpg > mediana(mpg)
mpg01[mpg > median(mpg)] <- 1
Auto <-  data.frame(Auto, mpg01)

head(Auto, 3)
```

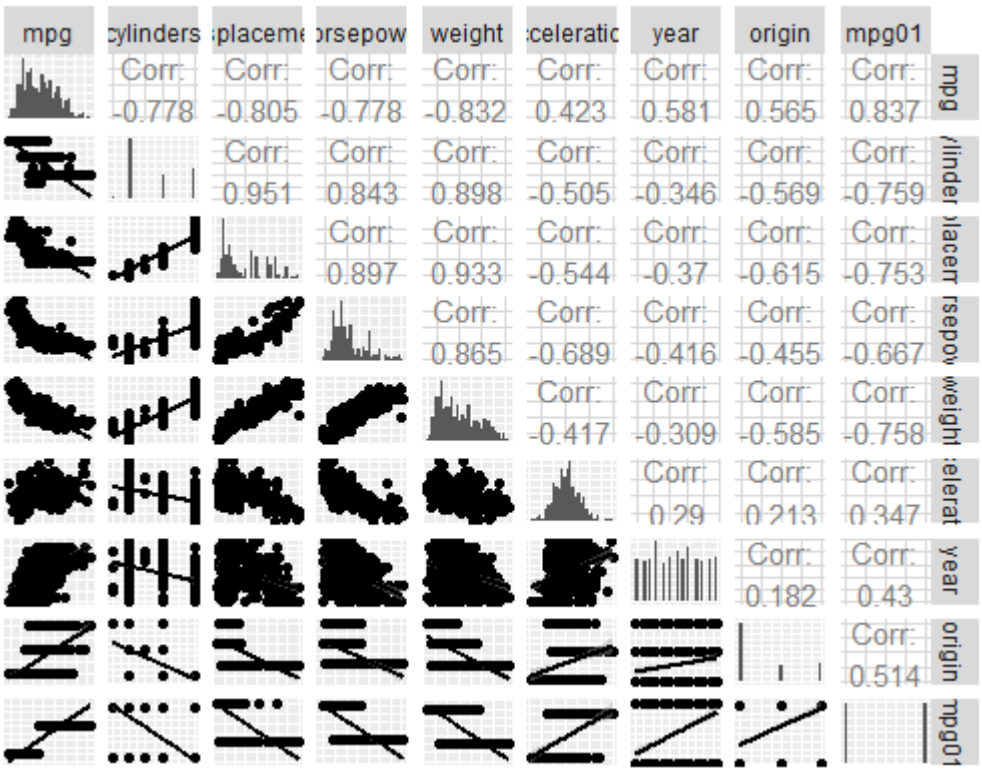
```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
##                        name mpg01
## 1 chevrolet chevelle malibu     0
## 2         buick skylark 320     0
## 3        plymouth satellite     0
```

```
# Correlación entre variables (excluyendo la variable cualitativa “name”)
cor(Auto[, -9], method = "pearson")
```

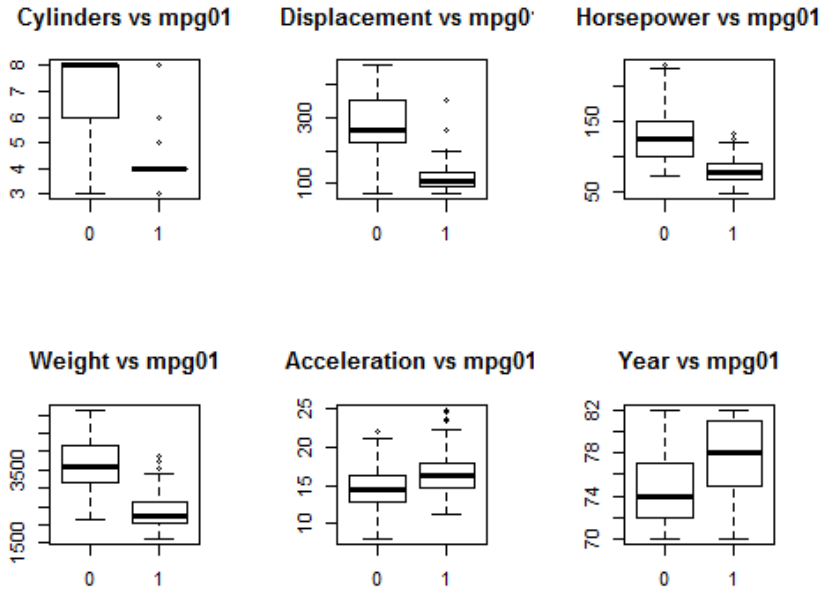
```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
```

```
## mpg01          0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##                acceleration       year     origin      mpg01
## mpg               0.4233285  0.5805410  0.5652088  0.8369392
## cylinders        -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement     -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower       -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight           -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration      1.0000000  0.2903161  0.2127458  0.3468215
## year              0.2903161  1.0000000  0.1815277  0.4299042
## origin            0.2127458  0.1815277  1.0000000  0.5136984
## mpg01             0.3468215  0.4299042  0.5136984  1.0000000
```

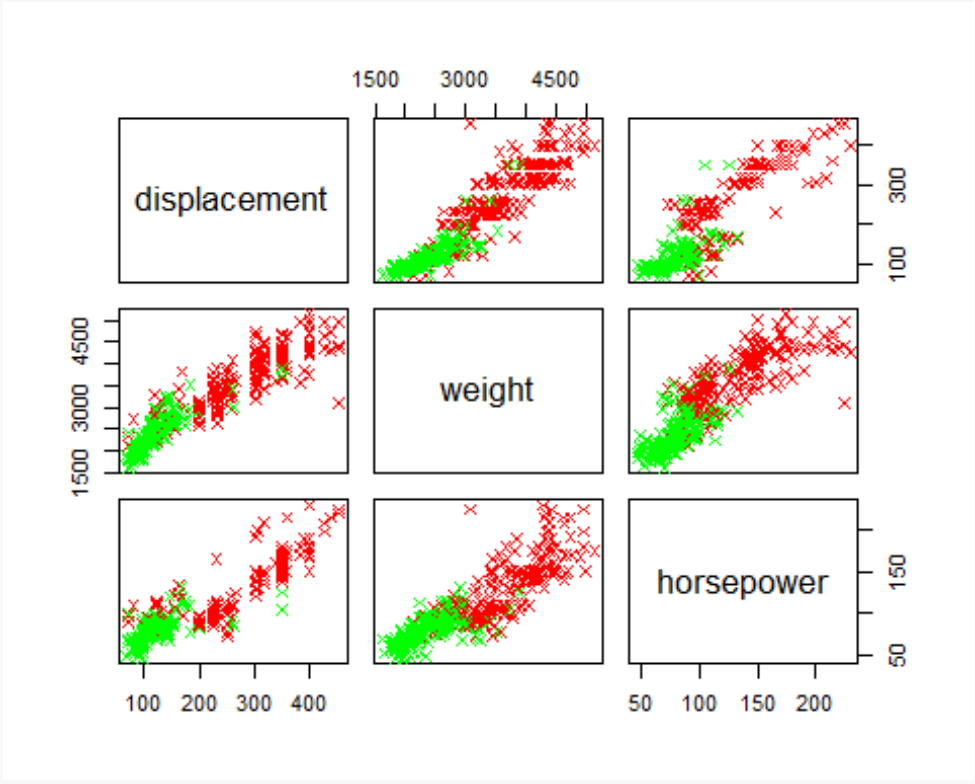
```
library(dplyr)
require(GGally)

ggpairs(select(Auto, - name),
        lower = list(continuous = "smooth"),
        diag = list(continuous = "bar"),
        axisLabels = "none")
```



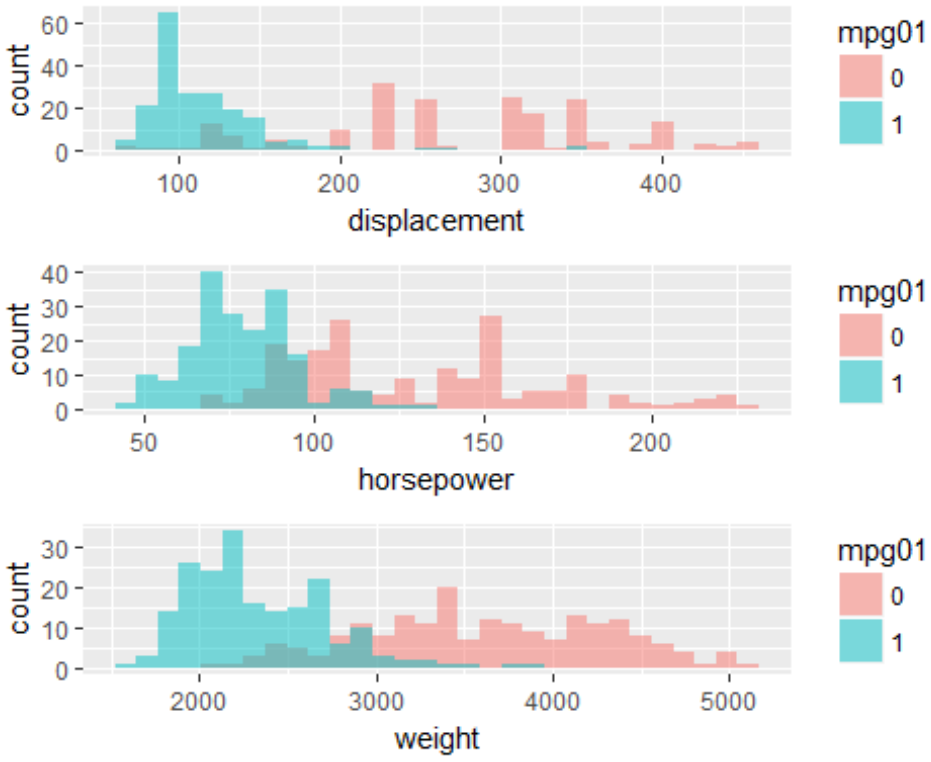

```
par(mfrow=c(2,3))
boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs mpg01")
boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs mpg01")
boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs mpg01")
boxplot(weight ~ mpg01, data = Auto, main = "Weight vs mpg01")
boxplot(acceleration ~ mpg01, data = Auto, main = "Acceleration vs mpg01")
boxplot(year ~ mpg01, data = Auto, main = "Year vs mpg01")
```




```
par(mfrow= c(1,1))
# Asociación entre variables cuantitativas más correlacionadas. Se colorean los ca
sos en función del rendimiento
pairs(x = Auto[, c("displacement", "weight", "horsepower")],
      col= ifelse(Auto$mpg01==0, "red", "green"),
      pch = 4)
```

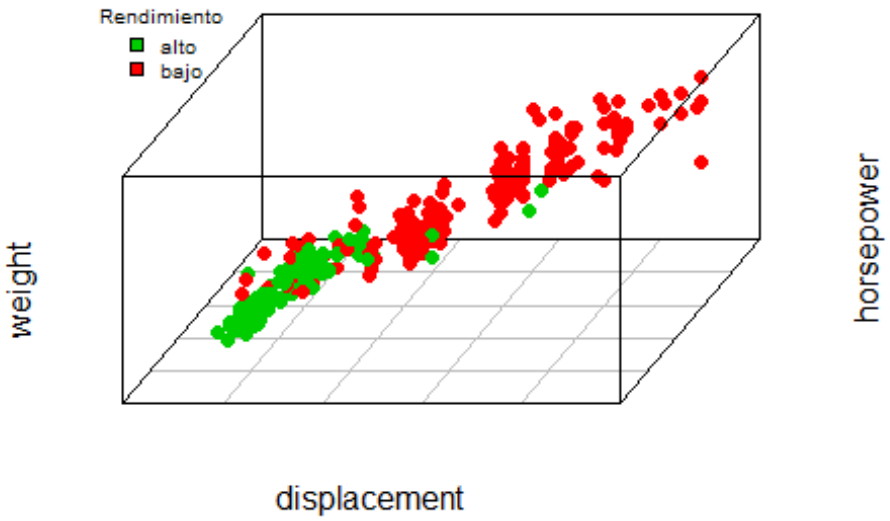


```r
# Representación de distribución en histogramas
library(ggplot2)
library(gridExtra)

p1 <- ggplot(data = Auto, aes(x = displacement)) +
geom_histogram(position = "identity",
               alpha = 0.5,
               aes(fill = as.factor(mpg01)))+
labs(fill = "mpg01")

p2 <- ggplot(data = Auto, aes(x = horsepower)) +
geom_histogram(position = "identity",
               alpha = 0.5,
               aes(fill = as.factor(mpg01)))+
labs(fill = "mpg01")

p3 <- ggplot(data = Auto, aes(x = weight)) +
geom_histogram(position = "identity",
               alpha = 0.5,
               aes(fill = as.factor(mpg01)))+
labs(fill = "mpg01")

grid.arrange(p1, p2, p3)
```




```r
# Representación tridimensional
library(scatterplot3d)
```

```
scatterplot3d(Auto$displacement, Auto$horsepower, Auto$weight, pch = 19,
              grid = TRUE, tick.marks = FALSE, angle = 65,
              xlab = "displacement", ylab = "horsepower", zlab = "weight",
              color = ifelse(Auto$mpg01 == 0, yes = "red", no = "green3"))
legend("topleft", bty = "n", cex = .6,
       title = "Rendimiento", c("alto", "bajo"),
       fill = c("green3", "red"))
```



Las variables que muestran mayor asociación con *mpg01* y que parecen ser útiles en separar los automóviles según el rendimiento (y que por tanto podrían ayudar a predecir esta variable respuesta) son: *displacement*, *weight*, *horsepower* y *cylinders*. Sin embargo, hay que tener en cuenta, según se aprecia en los gráficos, que la clase 0 de menor rendimiento solapa con la clase 1 de mayor rendimiento, lo que puede afectar la capacidad predictiva del modelo.

```
library(dplyr)
library(knitr)
# Subgrupo con las variables seleccionadas para el modelo
Auto2 <- select(Auto, cylinders, displacement, horsepower, weight, mpg01)
Auto2$mpg01 <- as.factor(mpg01)
kable(head(Auto2, 3), align = "c")
```

cylinders	displacement	horsepower	weight	mpg01
8	307	130	3504	0
8	350	165	3693	0
8	318	150	3436	0

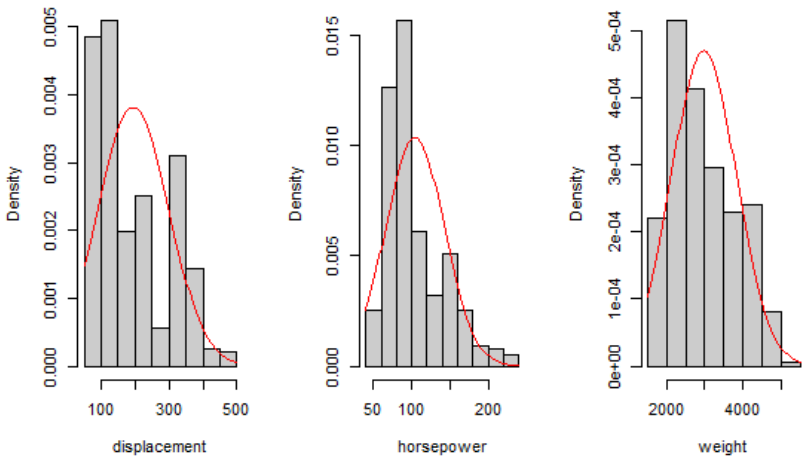
2. Análisis de normalidad (univariante y multivariante) y homogeneidad de varianza

Combinando resultados numéricos de distintos tests junto con métodos gráficos (histogramas, *Q-Q plots*) podremos llegar a conclusiones más robustas.

NORMALIDAD UNIVARIANTE

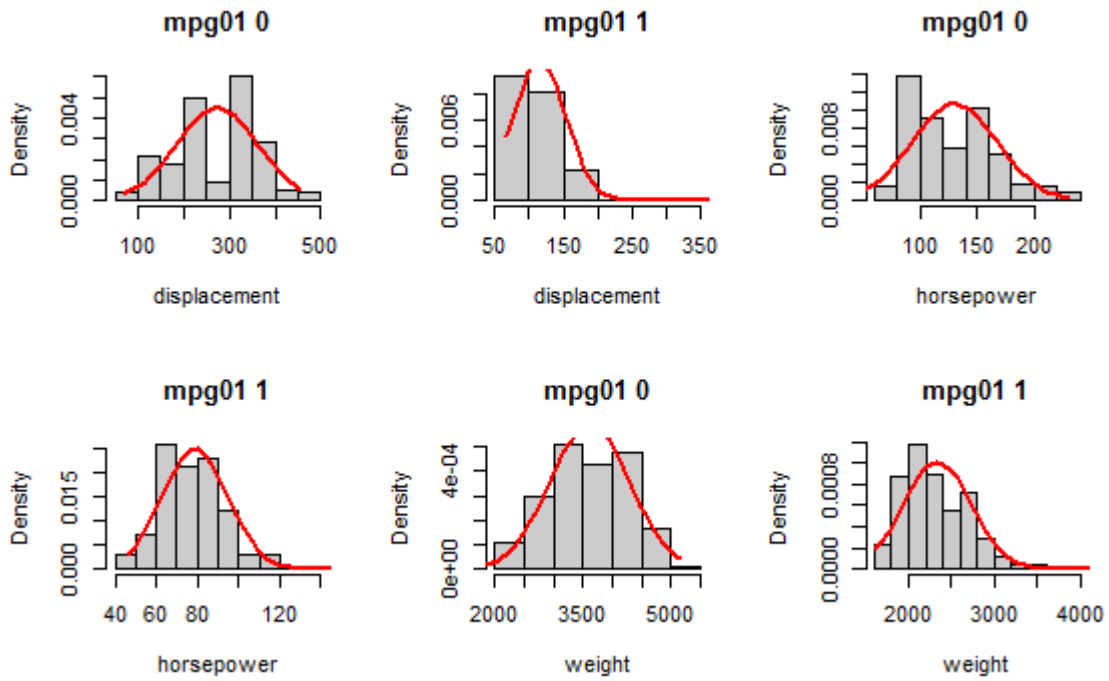
- **hist()** y **curve()**/**lines()** -> Histograma y curva
- **qqnorm()** y **qqline()**-> Gráfico Q-Q normal de los valores de "y" y línea teórica normal
- **shapiro.test()**-> Test de normalidad de Shapiro-Wilk

```r
par(mfrow= c(1,3))

# Distribución "displacement"
hist(Auto2$displacement, proba = T, col = grey(0.8), main = "")
curve(dnorm(x, mean=mean(Auto2$displacement), sd=sd(Auto2$displacement)),
      add=TRUE,
      col = "red")
# Distribución "horsepower"
hist(Auto2$horsepower, proba = T, col = grey(0.8), main = "")
curve(dnorm(x, mean=mean(Auto2$horsepower), sd=sd(Auto2$horsepower)),
      add=TRUE,
      col = "red")
# Distribución "weight"
hist(Auto2$weight, proba = T, col = grey(0.8), main = "")
curve(dnorm(x, mean=mean(Auto2$weight), sd=sd(Auto2$weight)),
      add=TRUE,
      col = "red")
```



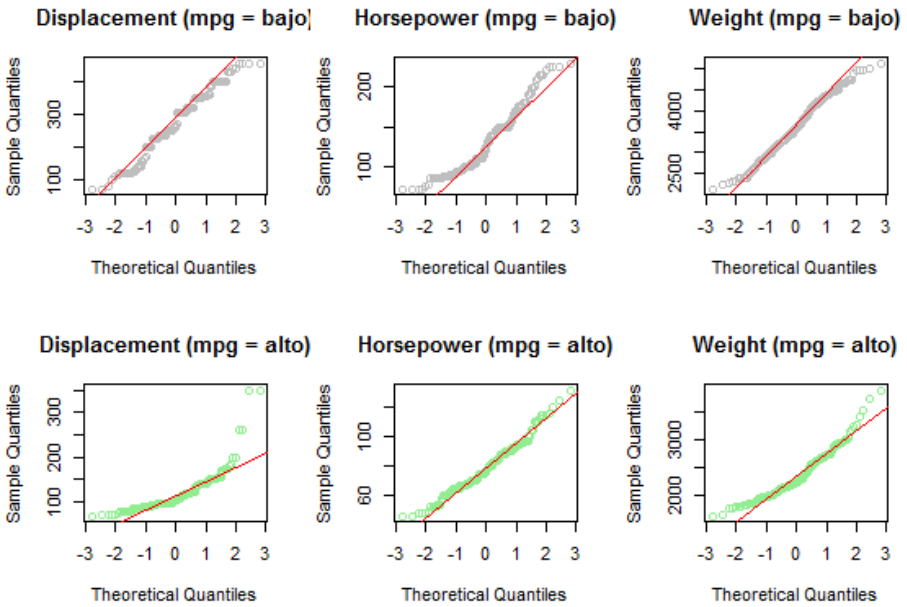
Los histogramas anteriores se pueden representar de forma automática para cada *mpg01* con un *for loop*:

```
par(mfrow=c(2,3))

for (k in 2:4) {
     v <- names(Auto2)[k]
     for (i in 1:2) {
         l <- levels(Auto2$mpg01)[i]
         x <- Auto2[Auto2$mpg01 == l, v]
         hist(x, proba = T, col = grey(0.8), main = paste("mpg01", l), xlab = v)
         x0 <- seq(min(Auto2[, k]), max(Auto2[, k]), le = 50)
         lines(x0, dnorm(x0, mean(x), sd(x)), col = "red", lwd = 2)
}
}
```



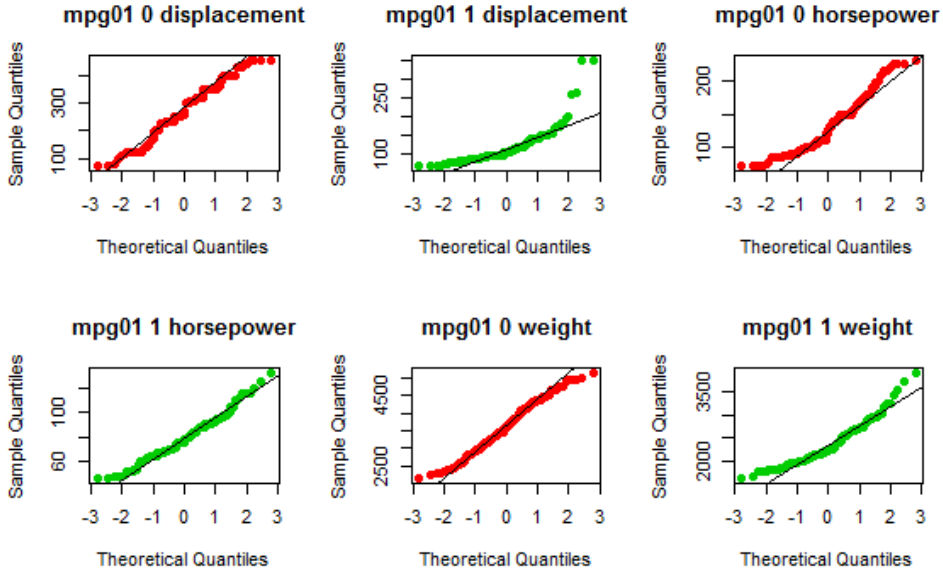

```
par(mfrow= c(2,3))

# Gráficos Q-Q de cada variable para la clase mpg01 = 0 (bajo rendimiento)
qqnorm(Auto2[mpg01 == 0, 2],
       col = "grey",
       main = "Displacement (mpg = bajo)")
qqline(Auto2[mpg01 == 0, 2], col = "red")

qqnorm(Auto2[mpg01 == 0, 3],
       col = "grey",
       main = "Horsepower (mpg = bajo)")
qqline(Auto2[mpg01 == 0, 3], col = "red")

qqnorm(Auto2[mpg01 == 0, 4],
       col = "grey",
       main = "Weight (mpg = bajo)")
qqline(Auto2[mpg01 == 0, 4], col = "red")
```

```r
# Gráficos Q-Q de cada variable para la clase mpg01 = 1 (alto rendimiento)
qqnorm(Auto2[mpg01 == 1, 2],
       col = "lightgreen",
       main = "Displacement (mpg = alto)")
qqline(Auto2[mpg01 == 1, 2], col = "red")

qqnorm(Auto2[mpg01 == 1, 3],
       col = "lightgreen",
       main = "Horsepower (mpg = alto)")
qqline(Auto2[mpg01 == 1, 3], col = "red")

qqnorm(Auto2[mpg01 == 1, 4],
       col = "lightgreen",
       main = "Weight (mpg = alto)")
qqline(Auto2[mpg01 == 1, 4], col = "red")
```



Se puede volver a automatizar esta representación mediante un *for loop*:

```r
par(mfrow=c(2,3))
for (k in 2:4) {
v <- names(Auto2)[k]
for (i in 1:2) {
l <- levels(Auto2$mpg01)[i]
x <- Auto2[Auto2$mpg01 == l, v]
qqnorm(x, main = paste("mpg01", l, v), pch = 19, col = i + 1)
qqline(x)
}
}
```



NOTA: La representación de histogramas y cuantiles anterior también se pueden obtener con la función *mvn*, cuyo uso se ejemplifica más adelante.

```
library(reshape2)
library(knitr)
library(dplyr)


# Contraste de normalidad Shapiro-Wilk para cada variable en cada rendimiento
Auto2.tidy <- melt(Auto2, value.name = "valor")

kable(Auto2.tidy %>% group_by(mpg01, variable) %>% summarise(p_value_Shapiro.test
= shapiro.test(valor)$p.value), align = "c")
```

mpg01	variable	p_value_Shapiro.test
0	cylinders	0.0000000
0	displacement	0.0010302
0	horsepower	0.0000002
0	weight	0.0554957
1	cylinders	0.0000000
1	displacement	0.0000000
1	horsepower	0.0029960
1	weight	0.0000003

Conclusión:

En conjunto hay evidencias de falta de normalidad univariante en todas las variables empleadas como predictores, menos la variable *weight* para la clase *mpg01* = 0 (rendimiento bajo).

NORMALIDAD MULTIVARIANTE

- **`mvn()`** -> Función del paquete MVN, que incluye argumentos para llevar a cabo tests y gráficos de normalidad multivariante, detección de *outliers* multivariantes, tests y gráficos de normalidad univariante

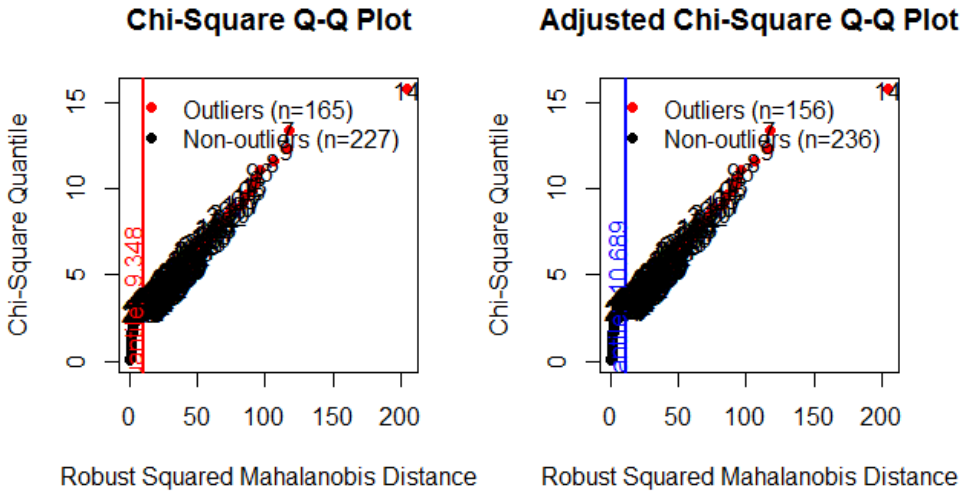
Además de la normalidad univariante, se requiere evaluar la normalidad multivariante. La presencia de valores atípicos puede ser causa de no cumplir esta condición. Por ello, es conveniente verificar si los datos tienen ***outliers*** **multivariantes** (valores extremos para combinaciones de variables) antes de comenzar con el análisis multivariante. Con el paquete MVN podemos evaluar la normalidad multivariante con tres de los test comúnmente utilizados, como el de *Mardia, Royston y Henze-Zirkler*, así como identificar los *outliers* multivariantes que puedan influir en el contraste.

```
library(MVN)
```

Detección de outliers multivariantes:

La el argumento *multivariateOutlierMethod* emplea el método de cuantiles basado en la distancia de Mahalanobis y la distancia de Mahalanobis ajustada.

```
par(mfrow = c(1, 2))
# Distancia de Mahalanobis
outliers <- mvn(data = Auto2[,2:4], multivariateOutlierMethod = "quan")
# Distancia ajustada de Mahalanobis
outliers.adj <- mvn(data = Auto2[,2:4], multivariateOutlierMethod = "adj")
```

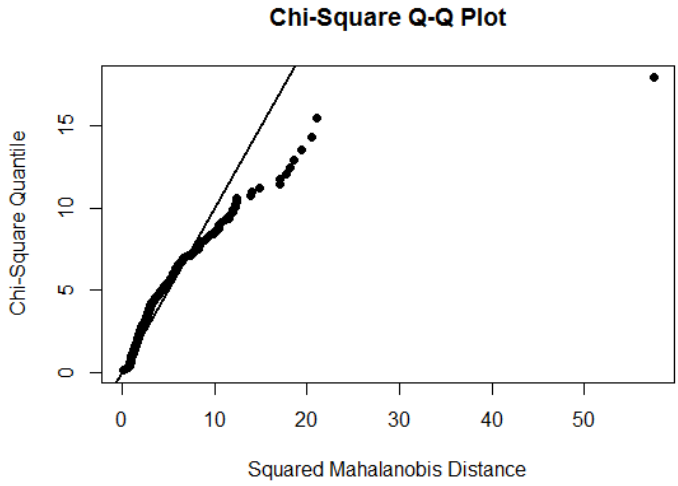


En torno al 40% de las observaciones se consideran *outliers* multivariantes (excluyendo la variable continua discreta *cylinders*), un porcentaje muy alto. Hay que tener en cuenta de lo anterior que la normalidad univariante no se cumple.

Test MVN de Royston:

NOTA: no se aconseja usar este test si los datos cuentan con más de 5000 o menos de 3 observaciones, ya que depende del test de *Shapiro Wilk*.

```
royston <- mvn(data = Auto2[,-5], mvnTest = "royston", multivariatePlot = "qq")


names(royston)

## [1] "multivariateNormality" "univariateNormality"   "Descriptives"


royston

## $multivariateNormality
##      Test        H      p value MVN
## 1 Royston 128.4618 9.127413e-29  NO
##
## $univariateNormality
##           Test     Variable Statistic   p value Normality
## 1 Shapiro-Wilk  cylinders      0.7507  <0.001      NO
## 2 Shapiro-Wilk displacement    0.8818  <0.001      NO
## 3 Shapiro-Wilk  horsepower     0.9041  <0.001      NO
## 4 Shapiro-Wilk    weight       0.9415  <0.001      NO
##
## $Descriptives
##                n        Mean    Std.Dev Median  Min  Max    25th    75th
## cylinders    392    5.471939   1.705783    4.0    3    8    4.00    8.00
## displacement 392  194.411990 104.644004  151.0   68  455  105.00  275.75
## horsepower   392  104.469388  38.491160   93.5   46  230   75.00  126.00
## weight       392 2977.584184 849.402560 2803.5 1613 5140 2225.25 3614.75
##                    Skew   Kurtosis
## cylinders    0.5042273 -1.4038700
## displacement 0.6963083 -0.7949853
## horsepower   1.0790191  0.6541069
## weight       0.5156160 -0.8253788
```



*El uso de la función *mvn* también nos da a conocer que la normalidad univariante para cada variable, sin distinguir entre clases de *mpg01* tampoco se cumple.

Test MVN de Mardia:

El test de *Mardia* calcula los coeficientes de asimetría y curtosis y la correspondiente significancia estadística, combinando ambos para dar un resultado global de normalidad multivariante (MVN):

```
mardia <- mvn(data = Auto2[,-5], mvnTest = "mardia")


mardia$multivariateNormality
```

```
##              Test        Statistic              p value Result
## 1 Mardia Skewness 415.192051331094 1.43400374737941e-75     NO
## 2 Mardia Kurtosis 16.5440533744995                    0     NO
## 3             MVN             <NA>                 <NA>     NO
```

Test MVN de Henze-Zirkler:

```
hz <- mvn(data = Auto2[,-5], mvnTest = "hz")


hz$multivariateNormality
```

```
##            Test      HZ p value MVN
## 1 Henze-Zirkler 12.3492       0  NO
```

Conclusión:

Todos los test llevados a cabo muestran evidencias significativas (α = 0,05) de que los datos no siguen una distribución normal multivariante (podríamos deducir que el problema se debe a la falta de normalidad univariante), por lo que habrá que tener en cuenta que tendrá implicaciones en la precisión del LDA/QDA.

MATRICES DE COVARIANZA

- **boxM()** -> Realiza el test M de Box (1949) para determinar la homogeneidad de las matrices de covarianza obtenidas a partir de datos normales multivariados según uno o más factores de clasificación. Tiene como hipótesis nula que las matrices de covarianza son iguales. IMP: sensible a la falta de normalidad multivariante

```
library(biotools)

## ---
## biotools version 3.1

boxM(data = Auto2[, 1:4], grouping = Auto2[, 5])

##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  Auto2[, 1:4]
## Chi-Sq (approx.) = 292.01, df = 10, p-value < 2.2e-16
```

El test M de Box muestra evidencias significativas de que la matriz de covarianza no es constante para todos los grupos, lo que haría apropiado aplicar QDA en lugar de LDA, pero en este caso ante la falta de normalidad multivariante en los datos, el test podría haberse visto afectado por ello.

3. Cálculo de la función discriminante

A modo de ejemplo y a pesar de la falta de normalidad multivariante e igualdad entre matrices de covarianza, ajustaremos los modelos LDA y QDA.

Primero dividiremos el set de datos en entrenamiento (80%) para ajustar el modelo, y en test (20%) para evaluarlo.

```r
set.seed(1)
entrenamiento <- sample(x = nrow(Auto), size = nrow(Auto)*0.8, replace = FALSE)
# Subgrupo de datos de entrenamiento
Auto.train <- Auto[entrenamiento,]
# Subgrupo de datos de test
Auto.test <- Auto[-entrenamiento,]


# Comprobamos que la suma de observaciones de cada subgrupo iguala al set de datos
 original
nrow(Auto.train)
```

```
## [1] 313
```

```r
nrow(Auto.test)
```

```
## [1] 79
```

```r
nrow(Auto.train) + nrow(Auto.test) == nrow(Auto)
```

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

PAQUETE MASS

- **lda()** -> Función para ajustar el modelo discriminante lineal

```r
library(MASS)


# Modelo LDA con los datos de entrenamiento
modelo.lda <- lda(formula = mpg01 ~ cylinders + displacement + horsepower + weight
, data = Auto.train)


modelo.lda
```

```
## Call:
## lda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto.train)
##
## Prior probabilities of groups:
##        0        1
## 0.514377 0.485623
##
## Group means:
##   cylinders displacement horsepower   weight
## 0  6.677019     269.8323  128.71429 3601.379
## 1  4.210526     117.4145   78.57895 2350.526
##
## Coefficients of linear discriminants:
##                          LD1
## cylinders    -0.4183687107
## displacement -0.0017457149
## horsepower    0.0028180950
## weight       -0.0009283838
```

El modelo calcula automáticamente las probabilidades a *priori* (π_0 = 0,514, π_1 = 0,485), y el promedio de cada predictor dentro de cada clase, usados por el modelo como estimadores de μ_k. Los coeficientes proporcionan la combinación de los predictores (- 0,4183**cylinders* - 0,0017**displacement* + 0,0028 – 0,0009) para generar los discriminantes lineales para cada una de las observaciones de entrenamiento. En este caso, como la variable respuesta solo cuenta con dos clases, sólo se genera una discriminante lineal (LD1).

```
# Modelo QDA con los datos de entrenamiento
modelo.qda <- qda(formula = mpg01 ~ cylinders + displacement + horsepower + weight
, data = Auto.train)


modelo.qda
```

```
## Call:
## qda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto.train)
##
## Prior probabilities of groups:
##         0         1
## 0.514377 0.485623
##
## Group means:
##   cylinders displacement horsepower   weight
## 0  6.677019     269.8323  128.71429 3601.379
## 1  4.210526     117.4145   78.57895 2350.526
```

A diferencia del LDA, el QDA no contiene los coeficientes de los discriminantes lineales, puesto que el clasificador QDA se basa en una función cuadrática de los predictores, no lineal.

PAQUETE CARET

- **`train()`** -> Función para ajuste de modelos

El paquete *caret* contiene funciones aplicadas a *machine learning*. A modo de ejemplo, se ajusta sólo el modelo LDA con este paquete:

```
library(caret)

modelo.lda.caret <- train(as.factor(mpg01) ~ cylinders + displacement + horsepower
 + weight,
                          method ='lda',
                          data=Auto.train)
```

```
modelo.lda.caret
```

```
## Linear Discriminant Analysis
##
## 313 samples
##   4 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 313, 313, 313, 313, 313, 313, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.876216  0.7519744
```

4. Evaluación del modelo

- **predict()**-> Predicciones a partir de los resultados de diversas funciones de ajuste de modelos
- **table()**-> Tabla de contingencia de los conteos en cada combinación de niveles de factores

Modelo LDA:

```
# Evaluación del modelo LDA con los datos de test
lda.pred <-  predict(object = modelo.lda, newdata = Auto.test)


names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

La función *predict()* ha calculado las probabilidades a *posteriori* de que cada nueva observación pertenezca a la clase con menor rendimiento (*mpg01* = 0). A continuación, calculamos el porcentaje de observaciones incorrectamente clasificadas comparando la predicción con la verdadera clase del set de datos de test, obteniendo así el *test error rate* del modelo discriminante:

```
# Matriz de confusion del modelo LDA
table(lda.pred$class, Auto.test$mpg01, dnn = c("Clase predicha", "Clase real"))
```

```
##               Clase real
## Clase predicha  0  1
##              0 32  1
##              1  3 43
```

```r
# Aplicando un threshold del 50% a las probabilidades a posterior podemos recrear las predicciones del modelo.
sum(lda.pred$posterior[,1] >= 0.5)
```

```
## [1] 33
```

```r
sum(lda.pred$posterior[,1] < 0.5)
```

```
## [1] 46
```

```r
# Test error rate del modelo LDA
mean(lda.pred$class != Auto.test$mpg01)
```

```
## [1] 0.05063291
```

```r
# Porcentaje de aciertos del modelo LDA
mean(lda.pred$class == Auto.test$mpg01)
```

```
## [1] 0.9493671
```

El *test error rate* es muy bajo con este modelo (5%). En prácticamente el 95% de los casos las observaciones son correctamente predichas.

NOTA: Podríamos cambiar el *threshold* de decisión si las predicciones no fueran lo bastante buenas en el sentido que nos interesa.

Modelo QDA:

```r
qda.pred <-  predict(object = modelo.qda, newdata = Auto.test)

table(qda.pred$class, Auto.test$mpg01, dnn = c("Clase predicha", "Clase real"))
```

```
##                Clase real
## Clase predicha  0  1
##               0 32  4
##               1  3 40
```

```r
mean(qda.pred$class != Auto.test$mpg01)
```

```
## [1] 0.08860759
```

El *test error rate* del modelo QDA es ligeramente superior (8,8%), por lo que en este caso optaríamos por el LDA, que cuenta con más porcentaje de aciertos globales.

PAQUERE CARET

- **confusionMatrix()**-> Calcula una tabulación cruzada de clases observadas y predichas con estadísticos asociados

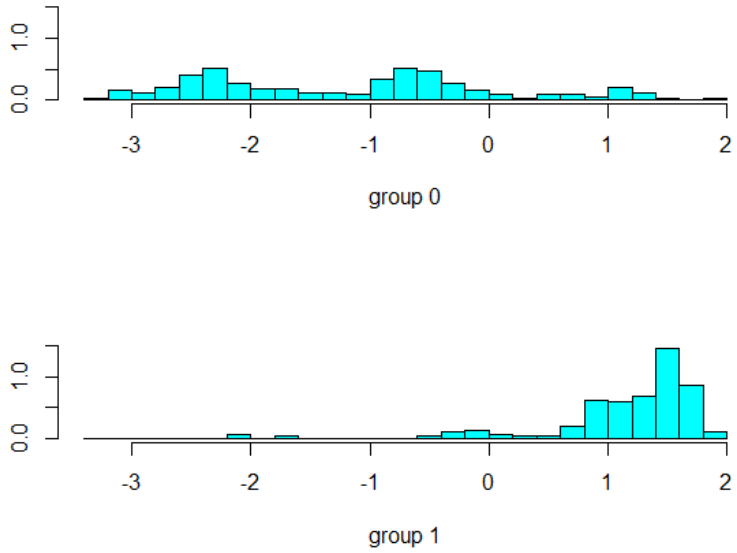
```
library(caret)


confusionMatrix(as.factor(Auto.test$mpg01), predict(modelo.lda.caret, Auto.test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 32  3
##          1  1 43
##
##                Accuracy : 0.9494
##                  95% CI : (0.8754, 0.986)
##     No Information Rate : 0.5823
##     P-Value [Acc > NIR] : 1.196e-13
##
##                   Kappa : 0.8968
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9697
##             Specificity : 0.9348
##          Pos Pred Value : 0.9143
##          Neg Pred Value : 0.9773
##              Prevalence : 0.4177
##          Detection Rate : 0.4051
##    Detection Prevalence : 0.4430
##       Balanced Accuracy : 0.9522
##
##        'Positive' Class : 0
##
```

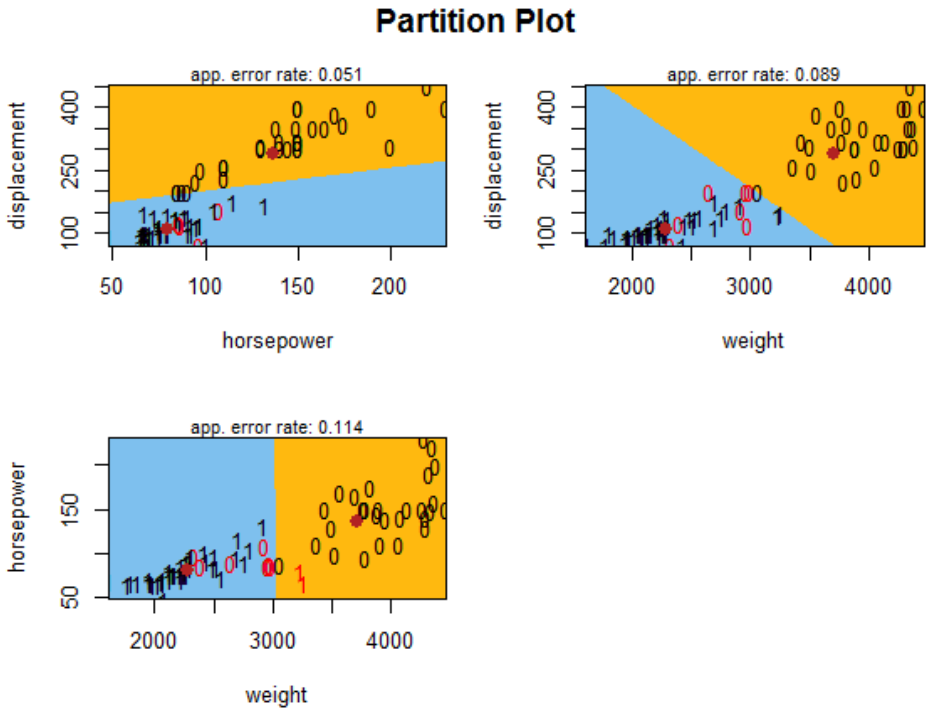
5. Visualización del modelo

- **partimat()**-> Proporciona una matriz de figuras múltiples que muestra la clasificación de las observaciones basadas en métodos de clasificación (por ejemplo, lda, qda) para cada combinación de dos variables. Además, se muestran los límites de clasificación y las tasas de error aparentes se dan en cada título
- **plot()**

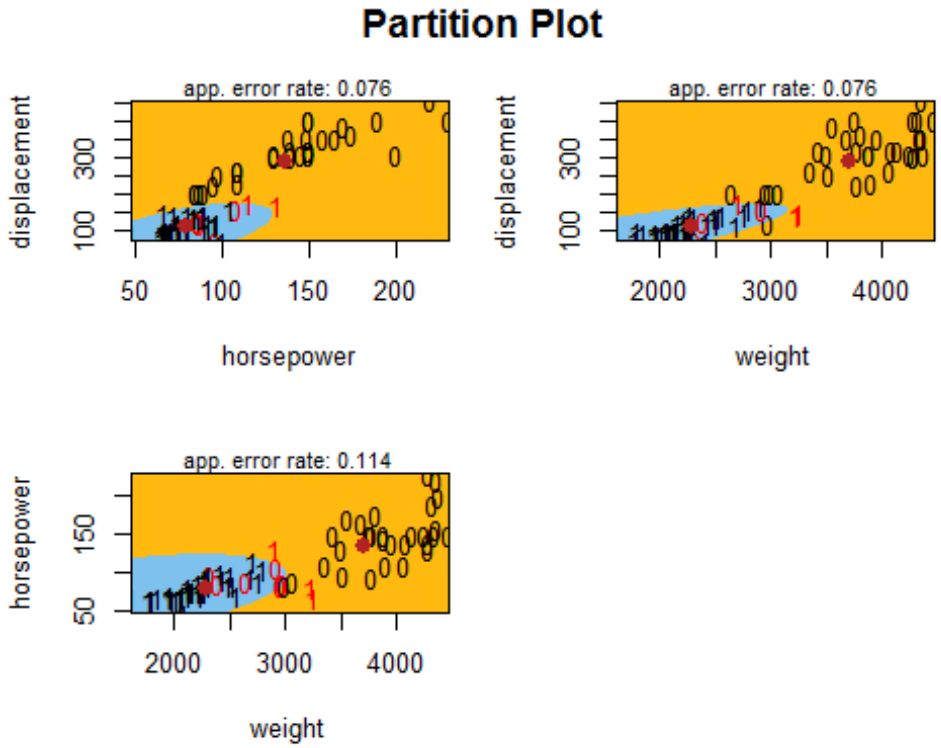
```r
# LDA respecto a los datos de entrenamiento usados en el modelo
plot(modelo.lda)
```




```r
library(klaR)
# Representación del LDA respecto a los datos de test
partimat(formula = as.factor(mpg01) ~ displacement + horsepower + weight,
         data = Auto.test, method = "lda", prec = 400,
         image.colors = c("darkgoldenrod1", "skyblue2"),
         col.mean = "firebrick",
         nplots.vert = 2)
```



```r
# Representación del QDA respecto a los datos de test
partimat(formula = as.factor(mpg01) ~ displacement + horsepower + weight,
         data = Auto.test, method = "qda", prec = 400,
         image.colors = c("darkgoldenrod1", "skyblue2"),
         col.mean = "firebrick", nplots.vert = 2)
```



En base a las representaciones podemos observar que la función discriminante consigue separar mejor la clase 1, existiendo superposición respecto a la clase 0.

COMPARACIÓN ENTRE LDA Y QDA

LDA es un método mucho menos flexible que QDA y sufre de menos varianza. Ello puede suponer una mejora en la predicción, pero hay un inconveniente: si la asunción del LDA de que todas las clases comparten la misma matriz de covarianza no es correcta en realidad, el LDA puede sufrir un alto *bias* o sesgo. Visto de otra manera, LDA suele ser mejor que QDA si contamos con relativamente pocas observaciones de entrenamiento y reducir la varianza es importante. Por el contrario, se recomienda QDA si el set de observaciones de entrenamiento es muy grande y la varianza del clasificador no supone un problema, o si el supuesto de una matriz de covarianza común entre las clases claramente no se cumple.

Si el verdadero límite de Bayes es lineal, LDA será una aproximación más precisa que QDA. Si por el contrario no es lineal, QDA será una mejor opción.

BIBLIOGRAFÍA

An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)

Andrea J., Amón I., Técnicas para detección de outliers multivariantes. Revista en Telecomunicaciones e Informática, Vol. 3, No. 5 p. 11-25 (2013)

Korkmaz S., Goksuluk D., Zararsiz G., MVN: An R Package for Assessing Multivariate Normality. Trakya University

http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/AMult/tema6am.pdf