Análisis discriminate - Práctica 4

José Luis Romero Béjar - Guillermo Arturo Cañadas De la Fuente

Junio, 2024

En esta práctica se ilustra un ejemplo de clasificación con un modelo **discriminante lineal** y otro ejemplo con un **clasificador cuadrático**.

Para realizar esta práctica se deben descargar los siguientes archivos, disponibles en la plataforma PRADO del curso:

- *AD_4_esp.Rmd*

Aspectos tratados:

- Paquetes de R necesarios.
- Exploración gráfica de los datos.
- Supuestos: normalidad y homogeneidad de la varianza.
- Funciones discriminante.
- Visualización de las clasificaciones.

Carga/instalación de paquetes de R necesarios para esta práctica.

El siguiente módulo de código fuente se encarga de cargar, si ya están instalados, todos los paquetes que se utilizarán en esta sesión de R. Si bien un paquete de R se puede cargar en cualquier momento cuando se vaya a utilizar, es recomendable optimizar sus llamadas con este fragmento de código al principio.

Cargar un paquete en una sesión de R **requiere que ya esté instalado**. Si no es así, el primer paso es ejecutar la sentencia:

install.packages("name_of_the_library")

```
########################################
# Loading necessary packages and reason #
########################################

# This is an example of the first installation of a package
# Only runs once if the package is not installed
# Once it is installed this sentence has to be commented (not to run again)
# install.packages("ggplot2")

# Package required to call 'ggplot' function
library(ggplot2)

# Package required to call 'ggarrange' function
library(ggpubr)

# Package required to call 'scatterplot3d' function
```

```r
library(scatterplot3d)

# Package required to call 'melt' function
library(reshape2)

# Package required to call 'mvn' function
library(MVN)

# Package required to call 'boxM' function
library(biotools)

# Package required to call 'partimat' function
library(klaR)

# Package required to call 'summarise' function
library(dplyr)

# Package required to call 'createDataPartition' function
library(caret)
```

Análisis discriminante lineal

Un equipo de biólogos quiere generar un modelo estadístico que permita clasificar a qué especie, a ó b, pertenece un determinado insecto en función de la longitud de sus patas, el diámetro de su abdomen y el de su órgano sexual.

Como **datos de entrenamiento** se han medido estas tres variables (**longitud de las patas, diámetro del abdomen y diámetro del órgano sexual** en milímetros) en 10 individuos de cada una de las dos especies.

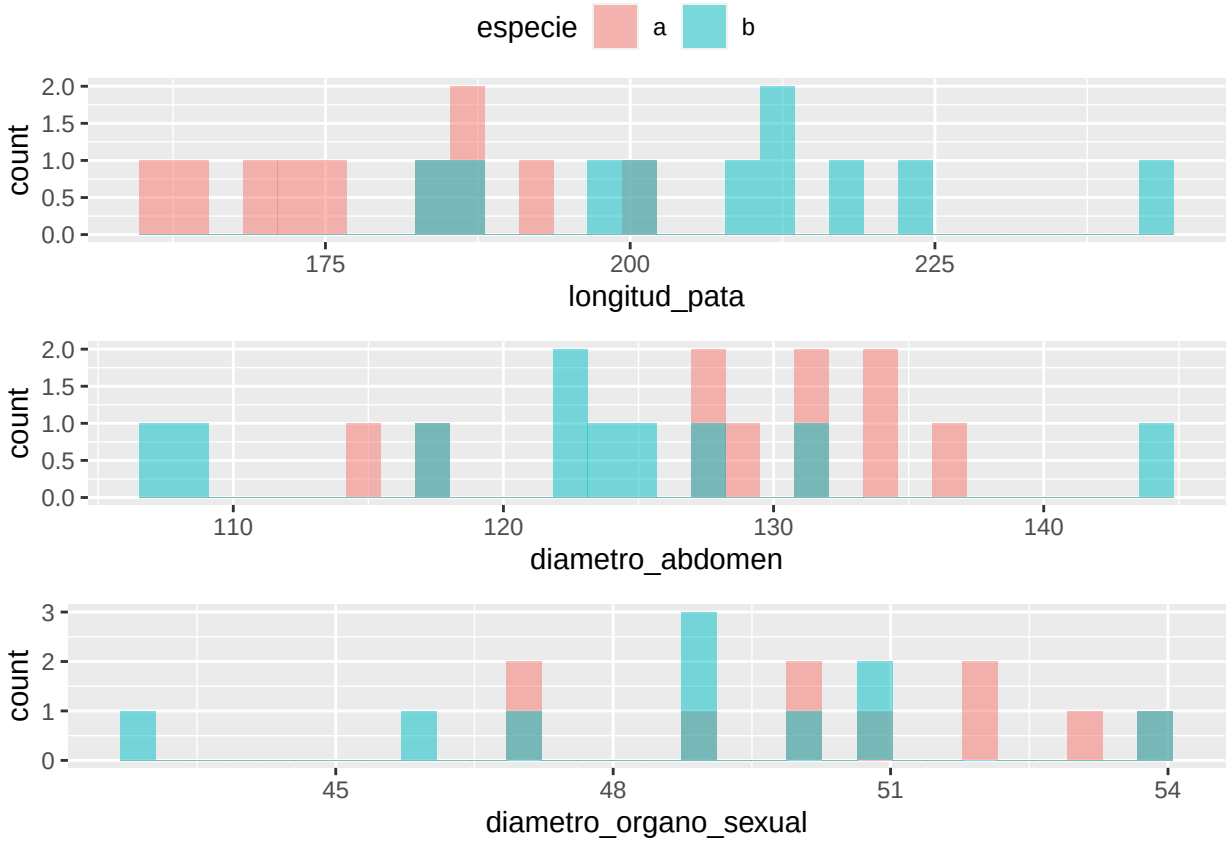
```r
# Response variable
especie<-c("a","a","a","a","a","a","a","a","a","a","b","b","b","b","b","b","b","b","b","b")

# Explanatory variables
longitud_pata<-c(191,185,200,173,171,160,188,186,174,163,186,211,201,242,184,211,
                 217,223,208,199)
diametro_abdomen<-c(131,134,137,127,128,118,134,129,131,115,107,122,144,131,108,
                    118,122,127,125,124)
diametro_organo_sexual<-c(53,50,52,50,49,47,54,51,52,47,49,49,47,54,43,51,49,51,
                          50,46)

# The whole dataset
datos<-data.frame(especie,longitud_pata,diametro_abdomen,diametro_organo_sexual)

# The response variable has to be an object of the class 'factor' of R language
datos$especie<-as.factor(datos$especie)
```

Exploración gráfica de los datos

En primer lugar exploramos como de bien (o mal) clasifica en las especies **cada una de las variables explicativas** consideradas de forma independiente.Para ello dibujamos los histogramas superpuestos. Si **los histogramas se separan**, la variable considerada sería un **buen clasificador individual** para la especie.

```r
p1 <- ggplot(data = datos, aes(x = longitud_pata, fill = especie)) +
      geom_histogram(position = "identity", alpha = 0.5)
```

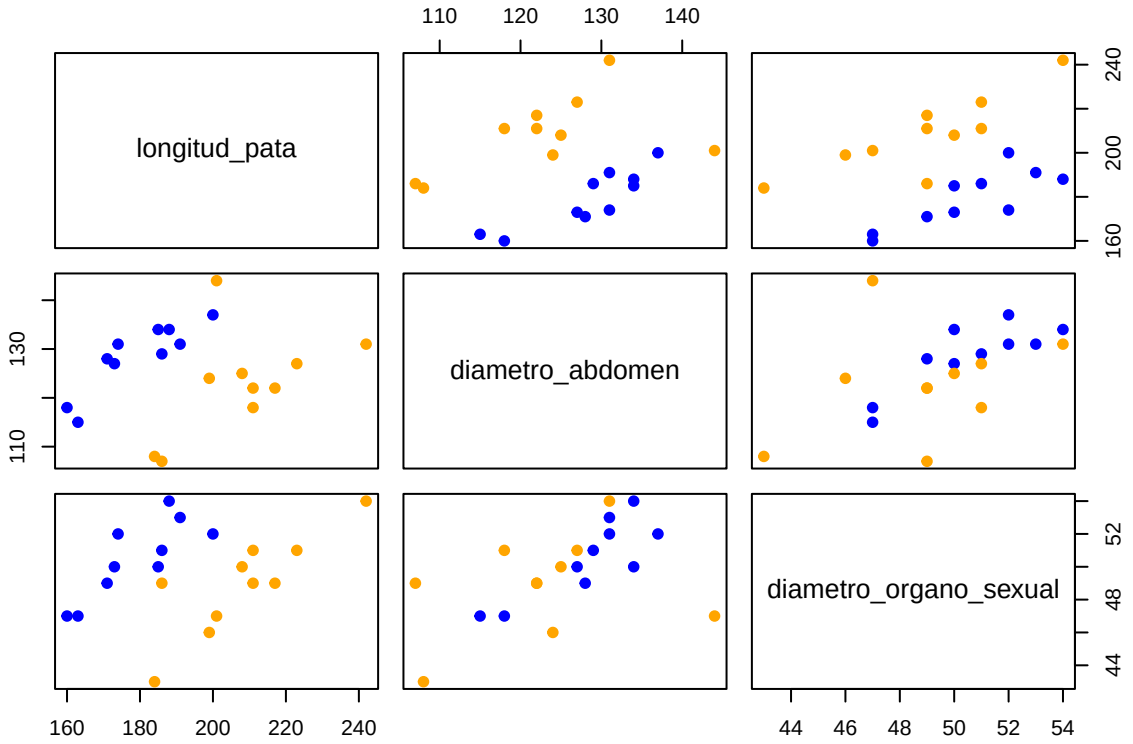
```r
p2 <- ggplot(data = datos, aes(x = diametro_abdomen, fill = especie)) +
      geom_histogram(position = "identity", alpha = 0.5)
p3 <- ggplot(data = datos, aes(x = diametro_organo_sexual, fill = especie)) +
      geom_histogram(position = "identity", alpha = 0.5)

ggarrange(p1, p2, p3, nrow = 3, common.legend = TRUE)
```



En este sentido, parece que la variable **longitud_pata** es la que mejor diferencia entre especies (menor solapamiento).

A continuación, exploramos qué **pares de variables** separan mejor entre especies.Dibujamos las nubes de puntos bivariadas. Si las nubes de puntos separan bien los colores asignados a cada especie, significa que el par de variables proporcionarían un buen modelo clasificador basado en ellas.

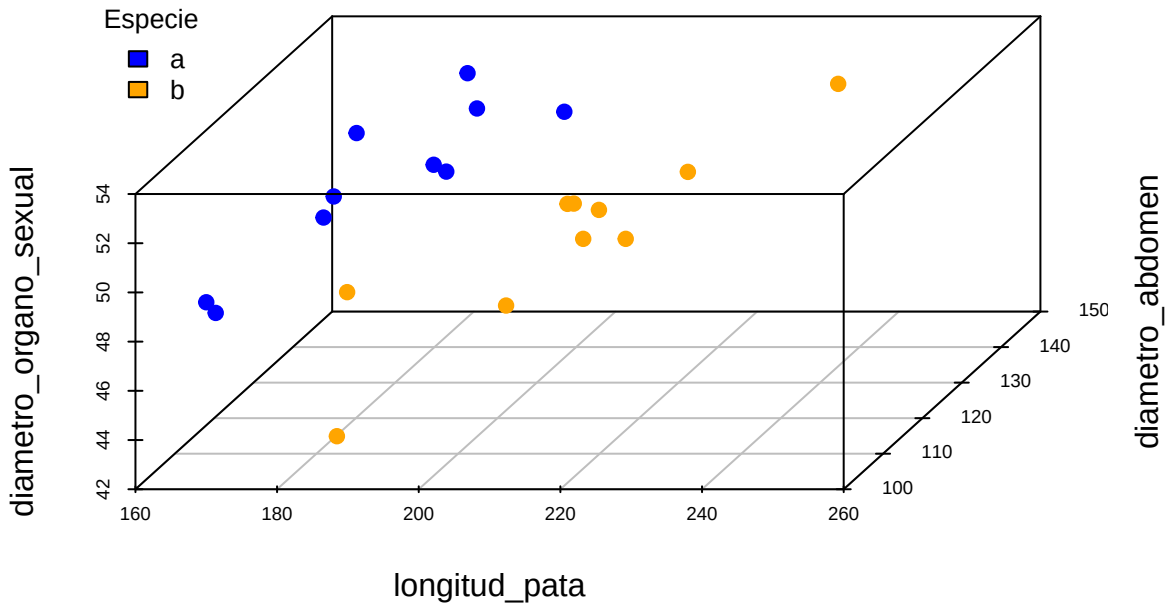
```r
pairs(x = datos[, c("longitud_pata","diametro_abdomen","diametro_organo_sexual")],
      col = c("blue", "orange")[datos$especie], pch = 19)
```



La combinación de variables **diametro_abdomen - longitud_pata** y **longitud_pata - diametro_organo_sexual** parecen separar adecuadamente entre especies.

Finalmente, exploramos si las **tres variables** juntas diferencian adecuadamente entre las dos especies.Dibujamos las nubes de puntos tridimensionales tratando de identificar si la representación gráfica separa las especies según los colores asignados a cada una de ellas.

```
scatterplot3d(datos$longitud_pata, datos$diametro_abdomen, datos$diametro_organo_sexual,
              color = c("blue", "orange")[datos$especie], pch = 19,
              grid = TRUE, xlab = "longitud_pata", ylab = "diametro_abdomen",
              zlab = "diametro_organo_sexual", angle = 65, cex.axis = 0.6)
legend("topleft",
       bty = "n", cex = 0.8,
       title = "Especie",
       c("a", "b"), fill = c("blue", "orange"))
```



Efectivamente, las tres variables de forma simultánea separan perfectamente las dos especies en el espacio tridimensional generado.Tiene sentido plantearse construir un modelo discriminante para la clasificación, por ejemplo.

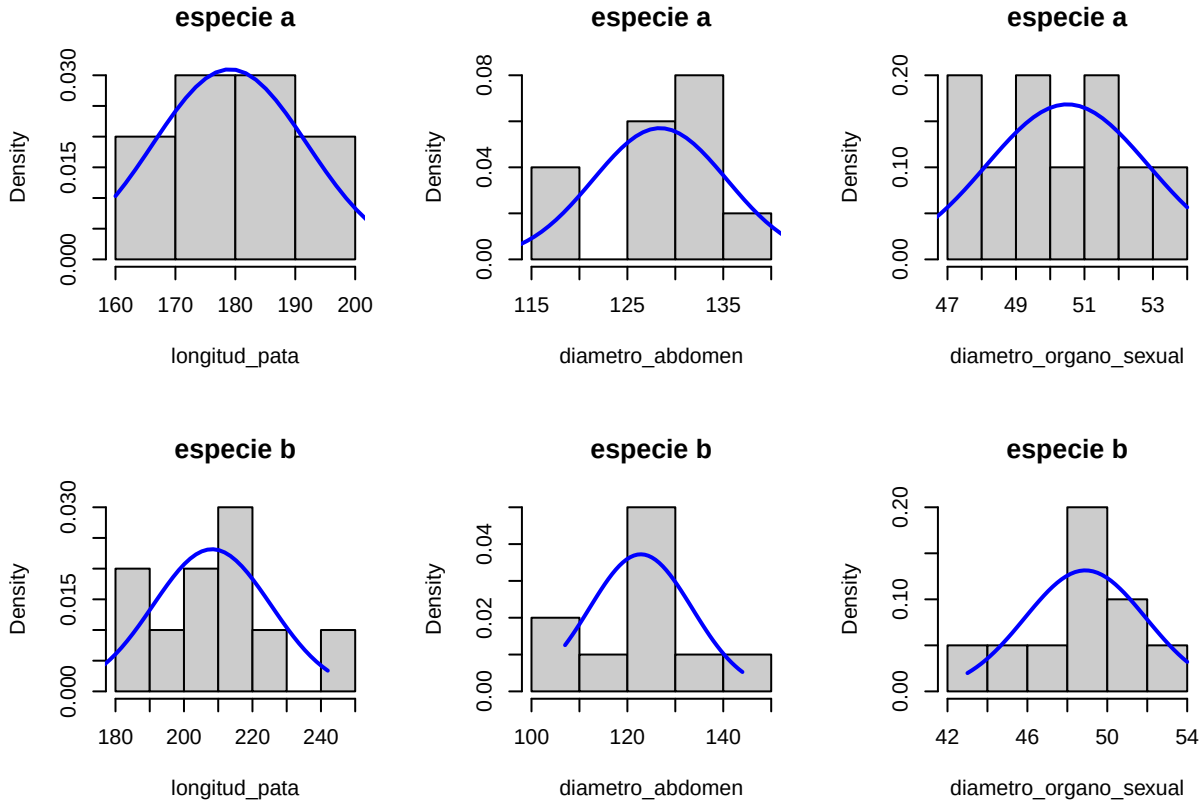
Normalidad univariante y multivariante

Normalidad univariante

A continuación hacemos una exploración gráfica de la **normalidad** de las **distribuciones univariantes** de nuestros predictores representando los **histogramas** y los **gráficos qqplots**.

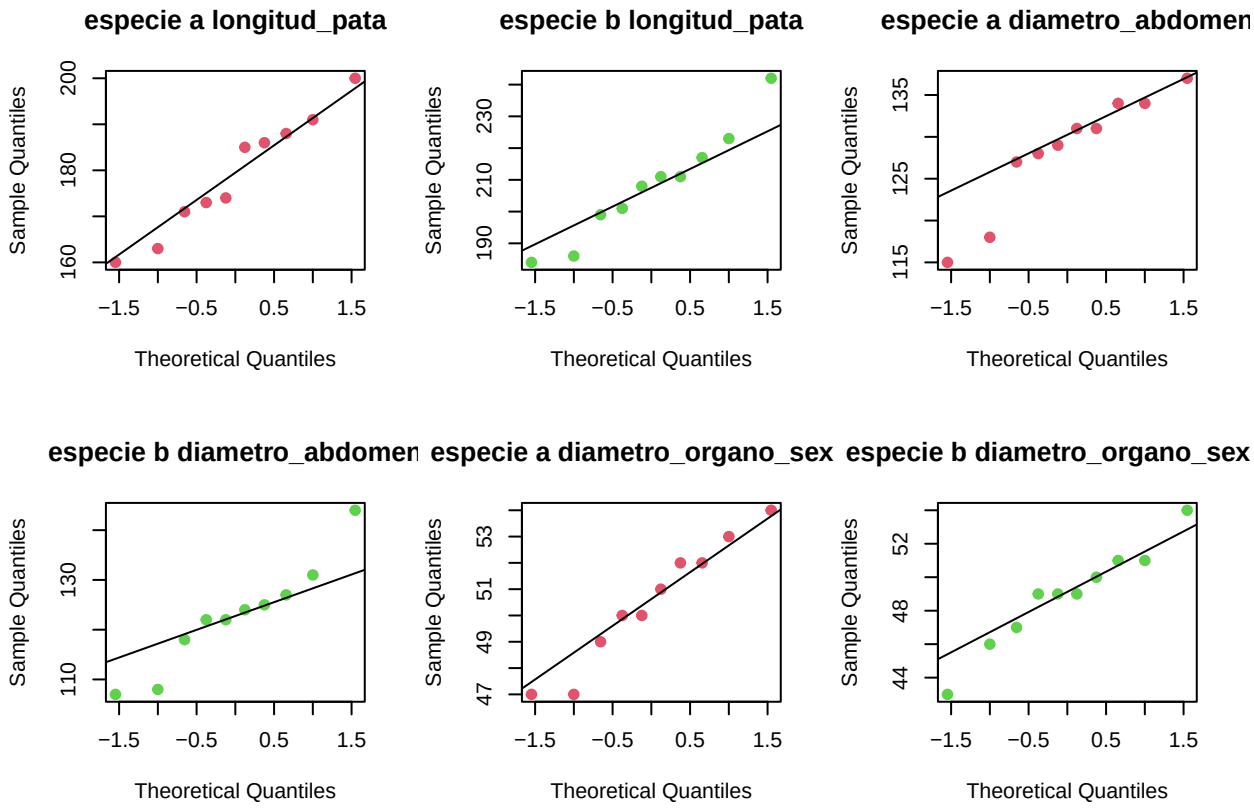
Histogramas univariantes

```
# Histogram representation of each variable for each species
par(mfcol = c(2, 3))
for (k in 2:4) {
  j0 <- names(datos)[k]
  x0 <- seq(min(datos[, k]), max(datos[, k]), le = 50)
  for (i in 1:2) {
    i0 <- levels(datos$especie)[i]
    x <- datos[datos$especie == i0, j0]
    hist(x, proba = T, col = grey(0.8), main = paste("especie", i0), xlab = j0)
    lines(x0, dnorm(x0, mean(x), sd(x)), col = "blue", lwd = 2)
  }
}
```



Gráficos qqplots

```
# Representation of normal quantiles of each variable for each species
par(mfrow=c(2,3))
for (k in 2:4) {
  j0 <- names(datos)[k]
  x0 <- seq(min(datos[, k]), max(datos[, k]), le = 50)
  for (i in 1:2) {
    i0 <- levels(datos$especie)[i]
    x <- datos[datos$especie == i0, j0]
    qqnorm(x, main = paste("especie", i0, j0), pch = 19, col = i + 1)
    qqline(x)
  }
}
```



Este análisis exploratorio puede darnos una idea de la posible distribución normal de las variables univariadas, pero siempre es mejor hacer los respectivos test de normalidad.

Test de normalidad univariantes (Shapiro-Wilk)

Se contrasta la **hipótesis nula** de que los datos **siguen una distribución normal** univariante. Se **rechaza** esta hipótesis si el **p-valor** dado por el test de Shapiro-Wilk es **menor que 0.05**. En otro caso no se rechaza el supuesto de normalidad de los datos.

```
datos_tidy <- melt(datos, value.name = "value")
aggregate(formula = value ~ especie + variable, data = datos_tidy,
          FUN = function(x){shapiro.test(x)$p.value})
```

```
##   especie              variable     value
## 1       a          longitud_pata 0.7763034
## 2       b          longitud_pata 0.7985711
## 3       a       diametro_abdomen 0.1845349
## 4       b       diametro_abdomen 0.5538213
## 5       a diametro_organo_sexual 0.6430844
## 6       b diametro_organo_sexual 0.8217855
```

No se encuentran evidencias de falta de normalidad univariante (p-valor > 0.05).

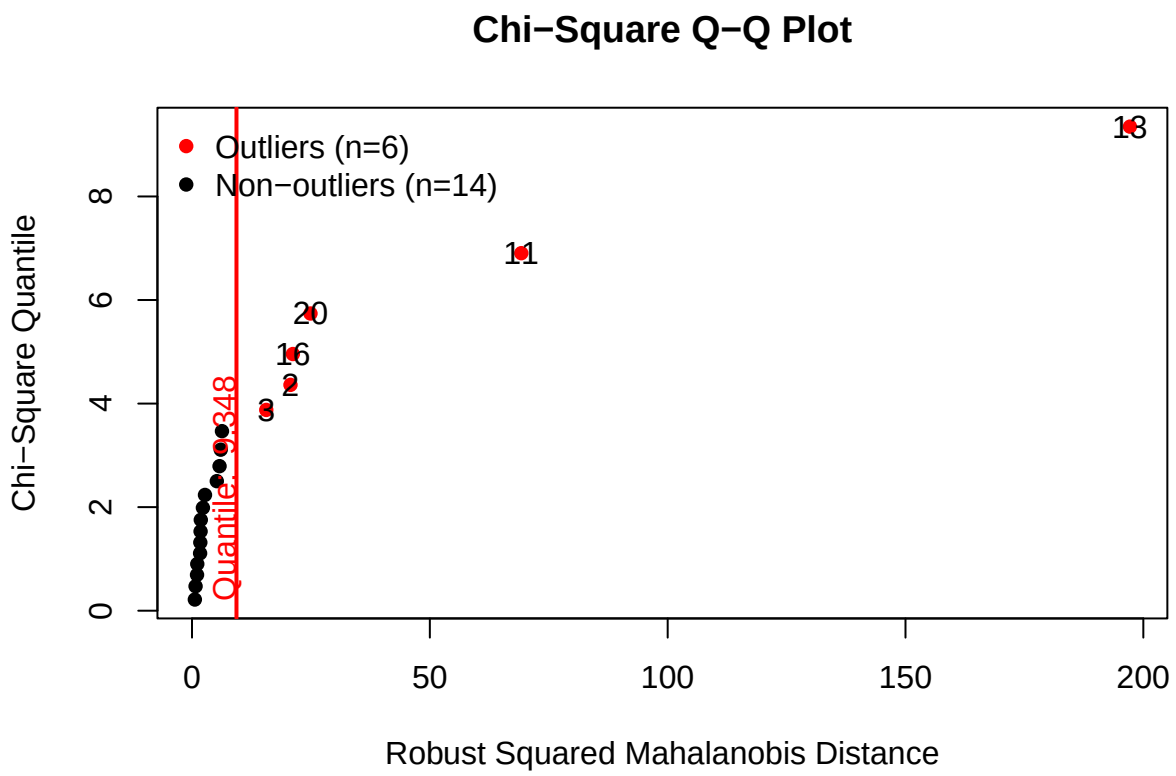
Normalidad multivariante

Test de normalidad multivariante (Mardia, Henze-Zirkler y Royston)

El paquete **MVN** contiene funciones que permiten realizar los tres test que se utilizan habitualmente para **contrastar la normalidad multivariante**.

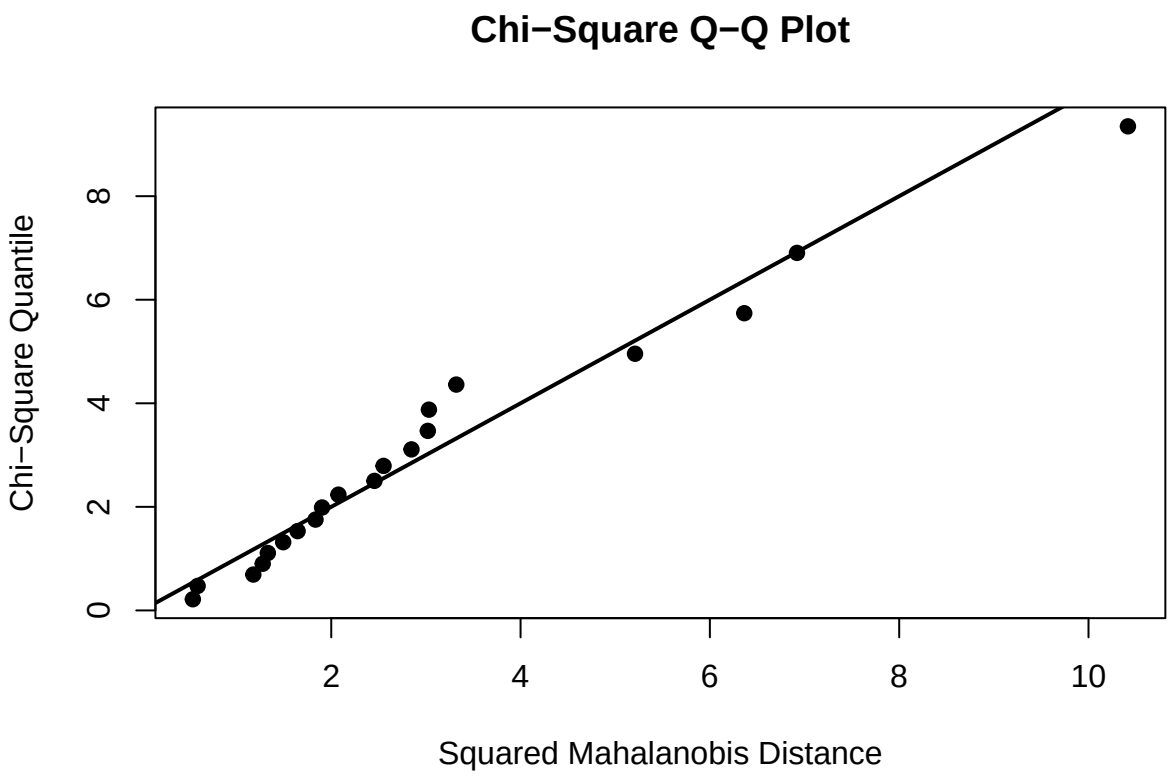
Esta normalidad multivariante puede verse afectada por la presencia de outliers multivariantes. En este paquete también encontramos **funciones para el análisis de outliers multivariantes**.

```
outliers <- mvn(data = datos[,-1], mvnTest = "hz", multivariateOutlierMethod = "quan")
```



Se detectan 6 outliers en las observaciones 2, 3, 11, 13, 16 y 20. Sin embargo ninguno de los dos test realizados a continuación encuentran evidencias al 5% de significación de falta de normalidad multivariante.

```
# Royston multivariate normality test
royston_test <- mvn(data = datos[,-1], mvnTest = "royston", multivariatePlot = "qq")
```




```
royston_test$multivariateNormality
```

```
##      Test         H   p value MVN
## 1 Royston 0.4636176 0.9299447 YES
```

```
# Henze-Zirkler multivariate normality test
hz_test <- mvn(data = datos[,-1], mvnTest = "hz")
hz_test$multivariateNormality
```

```
##            Test        HZ    p value MVN
## 1 Henze-Zirkler 0.7870498 0.07666139 YES
```

Homogeneidad de la varianza

Para el análisis de la homogeneidad de la varianza:

- Cuando hay **un solo predictor** el test más recomendable es el **test de Barttlet**, ya utilizado en prácticas anteriores.
- Cuando se emplean **múltiples predictores**, se tiene que contrastar que la matriz de covarianzas es constante en todos los grupos. En este caso es también recomendable comprobar la homogeneidad de la varianza para cada predictor a nivel individual. El test más recomendable es el **test de Box M**, que es una extensión del de Barttlet para escenarios multivariantes. Hay que tener en cuenta que es **muy sensible a que los datos efectivamente se distribuyan según una normal multivariante**. Por este motivo se recomienda utilizar una significación (p-value) <0.001 para rechazar la hipótesis nula.

La **hipóstesis nula** a contrastar es la de **igualdad de matrices de covarianzas** en todos los grupos.

```
boxM(data = datos[, 2:4], grouping = datos[, 1])
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  datos[, 2:4]
## Chi-Sq (approx.) = 9.831, df = 6, p-value = 0.132
```

En este caso no rechazamos la hipótesis nula ya que **p-value = 0.132 > 0.001** y por tanto **asumimos la homogeneidad de varianzas**.

Es importante recordar que para que esta **conclusión sea fiable** debe darse el supuesto de **normalidad multivariante** que, en este caso, se daba.

Función discriminante

Dado que se **satisfacen los supuestos de normalidad multivariante y de homogeneidad de la varianza** tiene sentido ajustar un **modelo de clasificación discriminante lineal**.

La función **lda** del paquete **MASS** ajusta este modelo.

```
modelo_lda <- lda(formula = especie ~ longitud_pata + diametro_abdomen + diametro_organo_sexual,data = d
modelo_lda
```

```
## Call:
## lda(especie ~ longitud_pata + diametro_abdomen + diametro_organo_sexual,
##     data = datos)
##
## Prior probabilities of groups:
##   a   b
## 0.5 0.5
##
## Group means:
##   longitud_pata diametro_abdomen diametro_organo_sexual
## a         179.1            128.4                   50.5
## b         208.2            122.8                   48.9
##
## Coefficients of linear discriminants:
##                                 LD1
## longitud_pata           0.13225339
## diametro_abdomen       -0.07941509
## diametro_organo_sexual -0.52655608
```

La salida de este objeto, nos muestra las **probabilidades a priori** de cada grupo, en este caso 0.5 para cada especie (10/20), las **medias de cada regresor por grupo** y los **coeficientes del modelo de clasificación discrimiante lineal**, que en este caso tendría la forma:

odds = 0.13225339pata - 0.07941509abdomen - 0.52655608organo_sexual

Una vez construido el clasificador podemos clasificar nuevos datos en función de sus medidas sin más que llamar a la función **predict**. Por ejemplo, consideremos un nuevo espécimen cuyas medidas sean: *longitud_pata = 194, diametro_abdomen = 124, diametro_organo_sexual = 49.*

```
nuevas_observaciones <- data.frame(longitud_pata = 194, diametro_abdomen = 124,diametro_organo_sexual =
predict(object = modelo_lda, newdata = nuevas_observaciones)
```

```
## $class
## [1] b
## Levels: a b
##
## $posterior
```

```
##            a         b
## 1 0.05823333 0.9417667
##
## $x
##         LD1
## 1 0.5419421
```

Según la función discriminante, la probabilidad a posteriori de que el espécimen sea de la especie **b** es del 94.2% mientras la de que sea de la especie **a** es tan solo del 5.8%. Por tanto este espécimen sería clasificado en la especie **b**. Del mismo modo, tomando como *cut-point* $\mathbf{p = 0.5}$, dado que la función discriminante toma el valor **0.5419421**, también invitaría a clasificarlo en la especie **b**.

Validación del modelo

Una primera forma básica de **validación del modelo** consiste en tratar de ver cómo de bien clasifica en un conjunto test del que conocemos la especie a la que pertenece cada registro. La forma habitual de proceder es dividir el dataset original en dos subconjuntos: el primero, conocido como **conjunto de entrenamiento**, con aproximadamente el 80% de los registros, que se utilizará para ajustar el modelo; y el segundo, conocido como **conjunto test**, con el 20% de los registros restantes, que será utilizado para la validación cruzada.

Para esta ilustración, sin embargo, tomamos como conjunto de entrenamiento y test al dataset completo, dado que tenemos tan solo 20 registros.

La función **confusionmatrix** del paquete **biotools** realiza una validación cruzada del modelo de clasificación.

```
pred <- predict(modelo_lda, dimen = 1)
confusionmatrix(datos$especie, pred$class)
```

```
##   new a new b
## a    10     0
## b     0    10
```

```
# Classification error percentage
trainig_error <- mean(datos$especie != pred$class) * 100
paste("trainig_error=", trainig_error, "%")
```

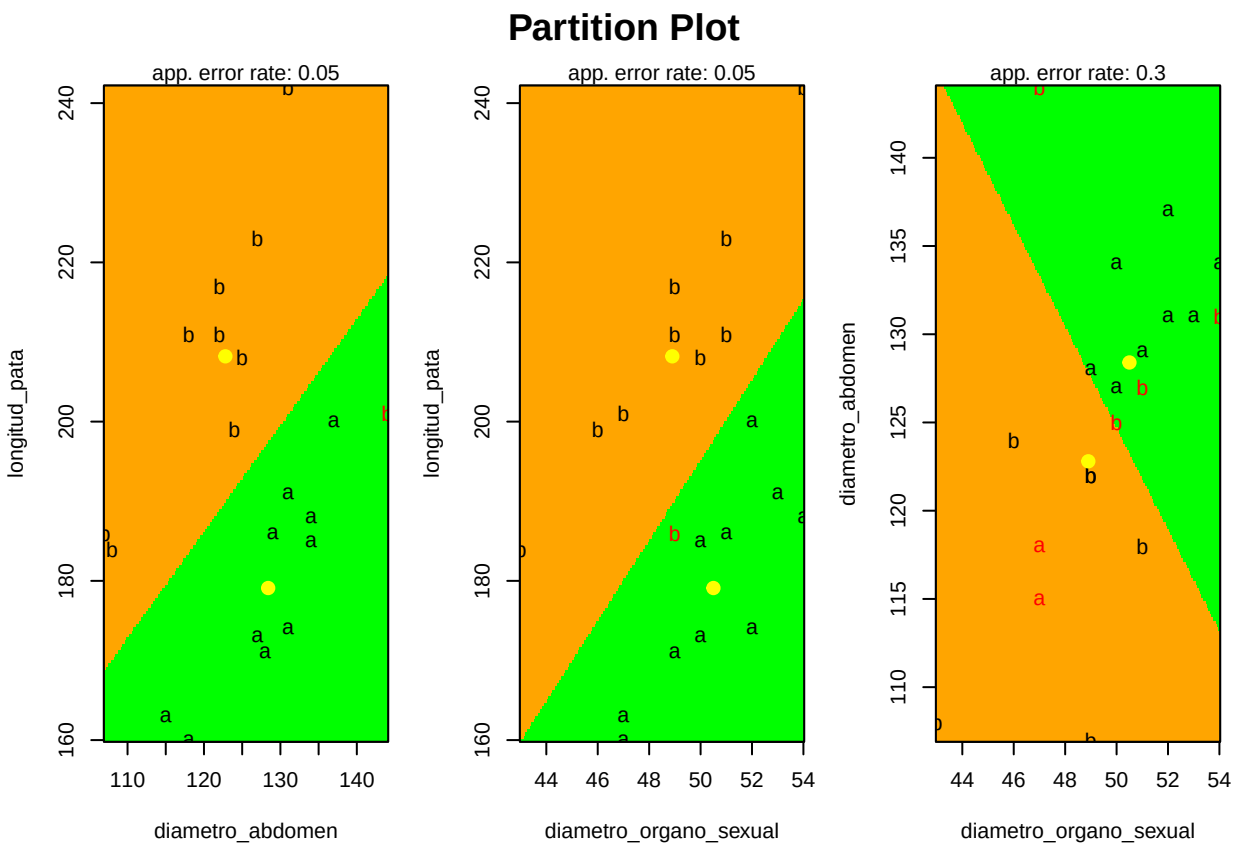
```
## [1] "trainig_error= 0 %"
```

En este caso el **porcentaje de aciertos es del 100%** (no es lo habitual).

Visualización de las clasificaciones

Desde el punto de vista geométrico, el análisis discriminante lineal separa el espacio mediante una recta. En este sentido, la función **partimat** del paquete **klaR** permite representar los **límites de clasificación** de un modelo discriminante lineal o cuadrático para cada par de predictores. Cada color representa una región de clasificación acorde al modelo, **se muestra el centroide** de cada región y el **valor real de las observaciones**.

```
partimat(especie ~ longitud_pata + diametro_abdomen + diametro_organo_sexual,
         data = datos, method = "lda", prec = 200,
         image.colors = c("green", "orange"),
         col.mean = "yellow",nplots.vert =1, nplots.hor=3)
```



A diferencia de la clasificación con las tres variables explicativas que tiene un error del 0%, al considerar la clasificación según cada par de variables, se cometen errores mayores, aunque como se intuía desde el principio **las parejas *diametro_abdomen - longitud_pata* y *diametro_organo_sexual - longitud_pata* cometen poco error en la clasificación (5%)** mientras el error con la pareja *diametro_abdomen - diametro_organo_sexual* se dispara al 30%.

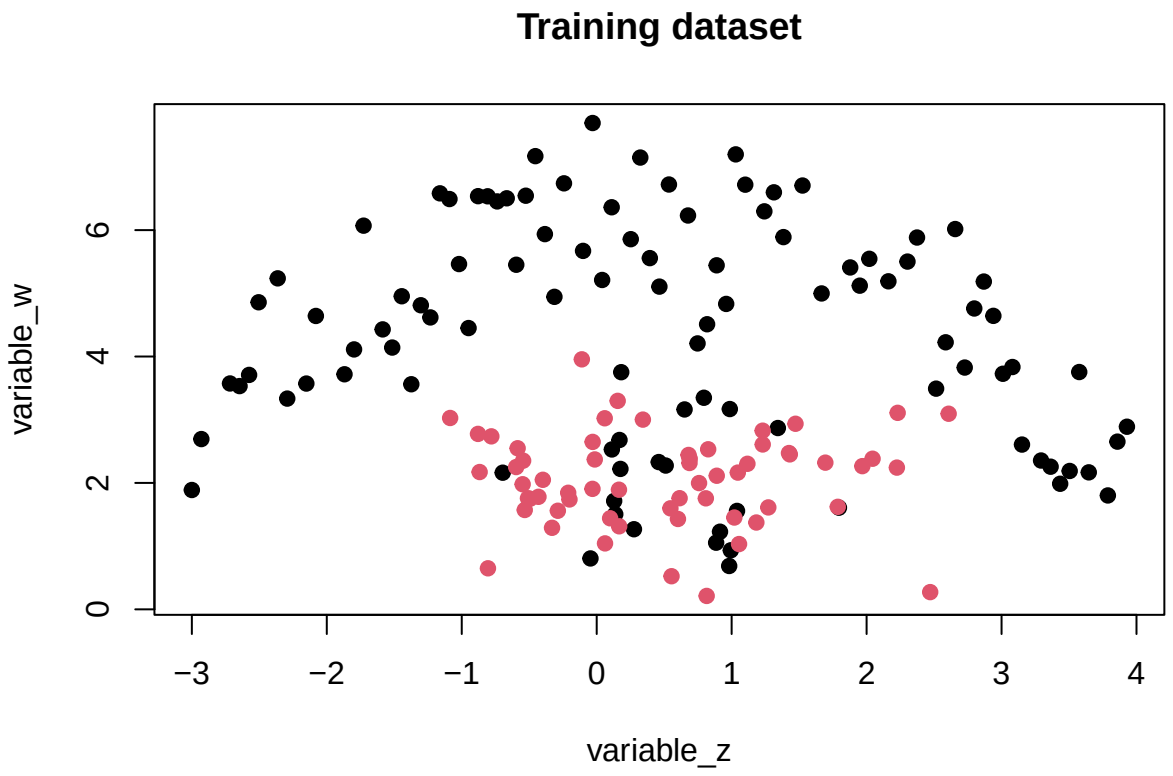
Análisis discriminante cuadrático

A continuación se ilustra un ejemplo de análisis discriminante cuadrático mediante un conjunto de **datos de entrenamiento simulados**. Se simulan 200 registros de dos variables explicativas, *variable_z y variable_w*, que clasifican en dos grupos, *grupo A y grupo B*.

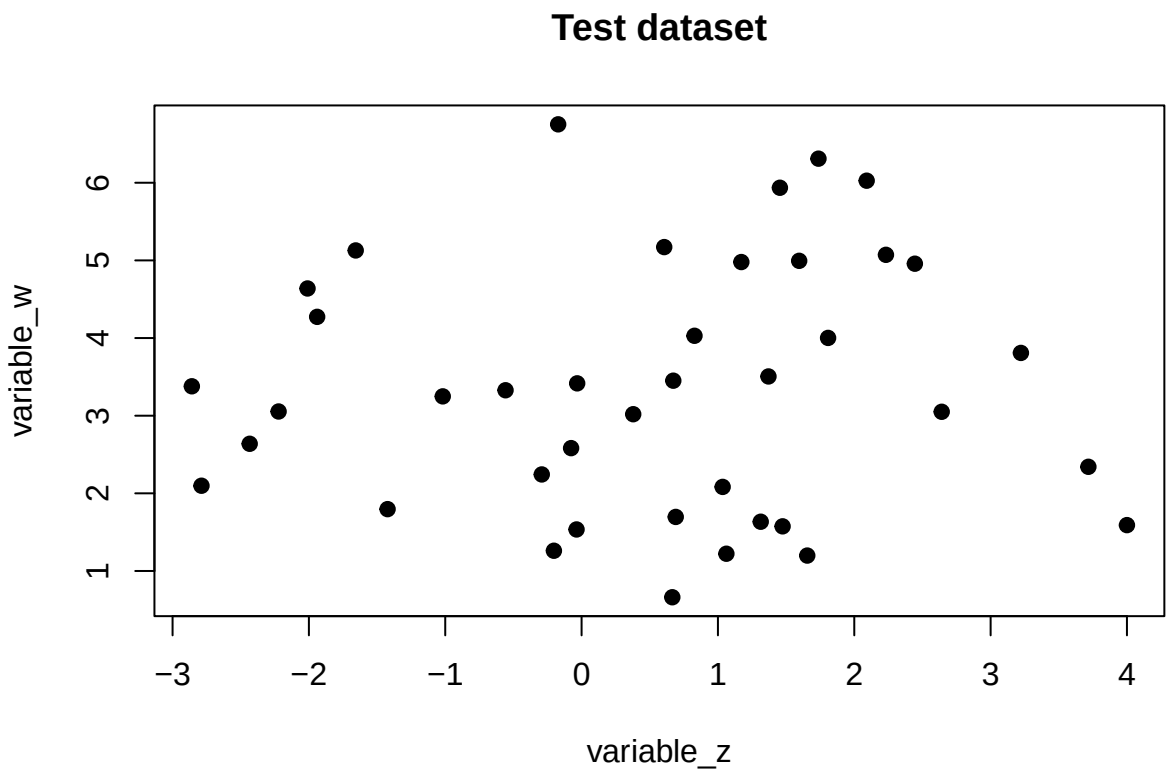
En este ejemplo sí **se va a trabajar con una partición del dataset, como conjunto de entrenamiento,** sobre el que realizaremos el análisis discriminante cuadrático y **otra partición, como conjunto test,** con el que realizaremos la validación del modelo.

```
set.seed(3)
grupoA_x <- seq(from = -3, to = 4, length.out = 100)
grupoA_y <- 6 + 0.15 * grupoA_x - 0.3 * grupoA_x^2 + rnorm(100, sd = 1)
grupoA <- data.frame(variable_z = grupoA_x, variable_w = grupoA_y, grupo = "A")

grupoB_x <- rnorm(n = 100, mean = 0.5, sd = 0.8)
grupoB_y <- rnorm(n = 100, mean = 2, sd = 0.8)
grupoB <- data.frame(variable_z = grupoB_x, variable_w = grupoB_y, grupo = "B")

datos <- rbind(grupoA, grupoB)
```

```r
datos$grupo <- as.factor(datos$grupo)

# Partitioning the dataset: training (80%) + test (20%)
trainIndex<-createDataPartition(datos$grupo,p=0.80)$Resample1
datos_train<-datos[trainIndex,]
datos_test<-datos[-trainIndex,]

plot(datos_train[, 1:2], col = datos$grupo, pch = 19, main="Training dataset")
```



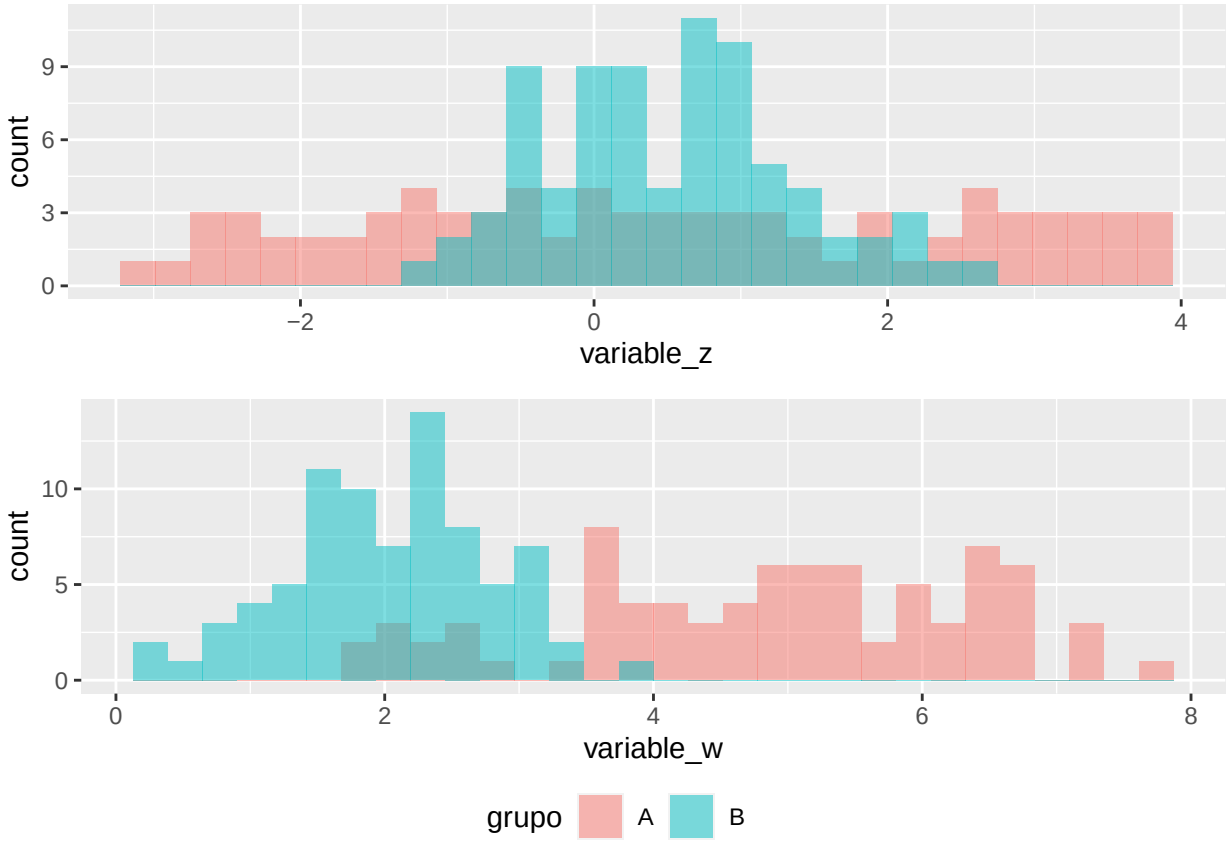

```r
plot(datos_test[, 1:2], col = datos$grupo, pch = 19, main="Test dataset")
```



Exploración gráfica de los datos

En primer lugar exploramos como de bien o mal clasifica en los grupos **cada una de las dos variables** consideradas de forma independiente.

```
p1 <- ggplot(data = datos_train, aes(x = variable_z, fill = grupo)) +
      geom_histogram(position = "identity", alpha = 0.5)
p2 <- ggplot(data = datos_train, aes(x = variable_w, fill = grupo)) +
      geom_histogram(position = "identity", alpha = 0.5)
ggarrange(p1, p2, nrow = 2, common.legend = TRUE, legend = "bottom")
```



En este sentido, parece que la **variable_w** es la que mejor diferencia entre grupos (menor solapamiento).

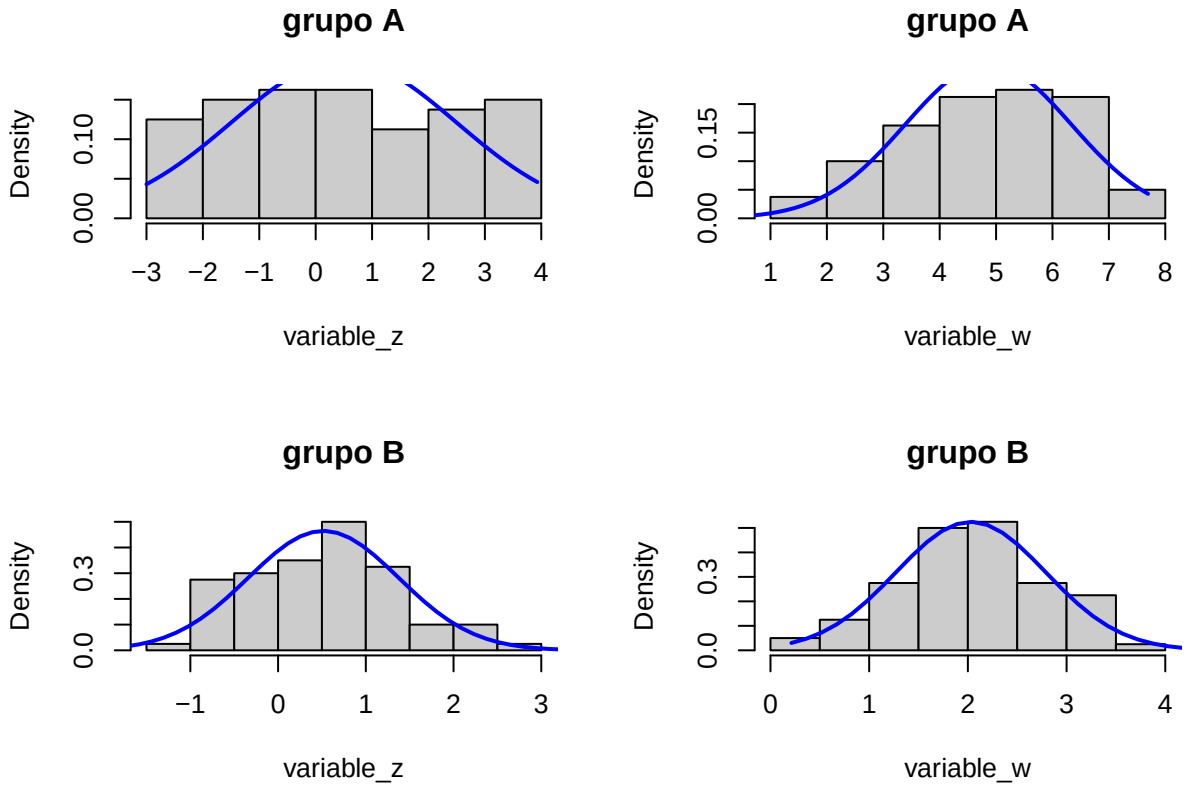
Normalidad univariante y multivariante

Normalidad univariante

A continuación hacemos una exploración gráfica de la **normalidad** de las **distribuciones univariantes** de nuestros predictores representando los **histogramas** y los **gráficos qqplots**.

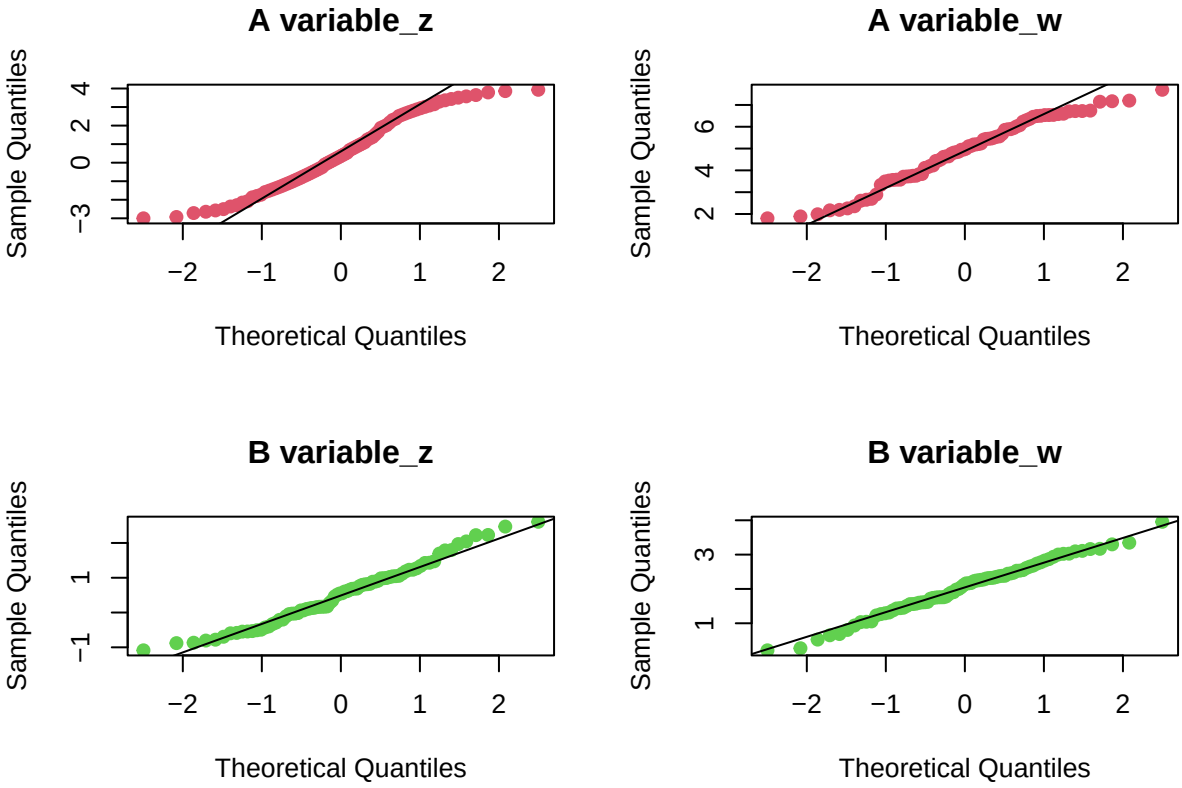
Histogramas individuales

```
# Histogram representation of each variable for each group
par(mfcol = c(2, 2))
for (k in 1:2) {
  j0 <- names(datos_train)[k]
  x0 <- seq(min(datos_train[, k]), max(datos_train[, k]), le = 50)
  for (i in 1:2) {
    i0 <- levels(datos_train$grupo)[i]
    x <- datos_train[datos_train$grupo == i0, j0]
    hist(x, proba = T, col = grey(0.8), main = paste("grupo", i0),
    xlab = j0)
    lines(x0, dnorm(x0, mean(x), sd(x)), col = "blue", lwd = 2)
  }
}
```



Gráficos qqplots

```
# Representation of normal quantiles of each variable for each group
par(mfcol = c(2, 2))
for (k in 1:2) {
  j0 <- names(datos_train)[k]
  x0 <- seq(min(datos_train[, k]), max(datos_train[, k]), le = 50)
  for (i in 1:2) {
    i0 <- levels(datos_train$grupo)[i]
    x <- datos_train[datos_train$grupo == i0, j0]
    qqnorm(x, main = paste(i0, j0), pch = 19, col = i + 1)
    qqline(x)
  }
}
```



Este análisis exploratorio puede darnos una idea de la posible distribución normal de las variables univariadas, pero siempre es mejor hacer los respectivos test de normalidad.

Test de normalidad univariantes (Shapiro-Wilk)

```
# Shapiro-Wilk normality test for each variable in each group
datos_tidy <- melt(datos_train, value.name = "valor")
datos_tidy %>%
  group_by(grupo, variable) %>%
  summarise(p_value_Shapiro.test = round(shapiro.test(valor)$p.value,5))
```

```
## # A tibble: 4 x 3
## # Groups:   grupo [2]
##   grupo variable   p_value_Shapiro.test
##   <fct> <fct>                     <dbl>
## 1 A     variable_z               0.0091
## 2 A     variable_w               0.0447
## 3 B     variable_z               0.206
## 4 B     variable_w               0.872
```

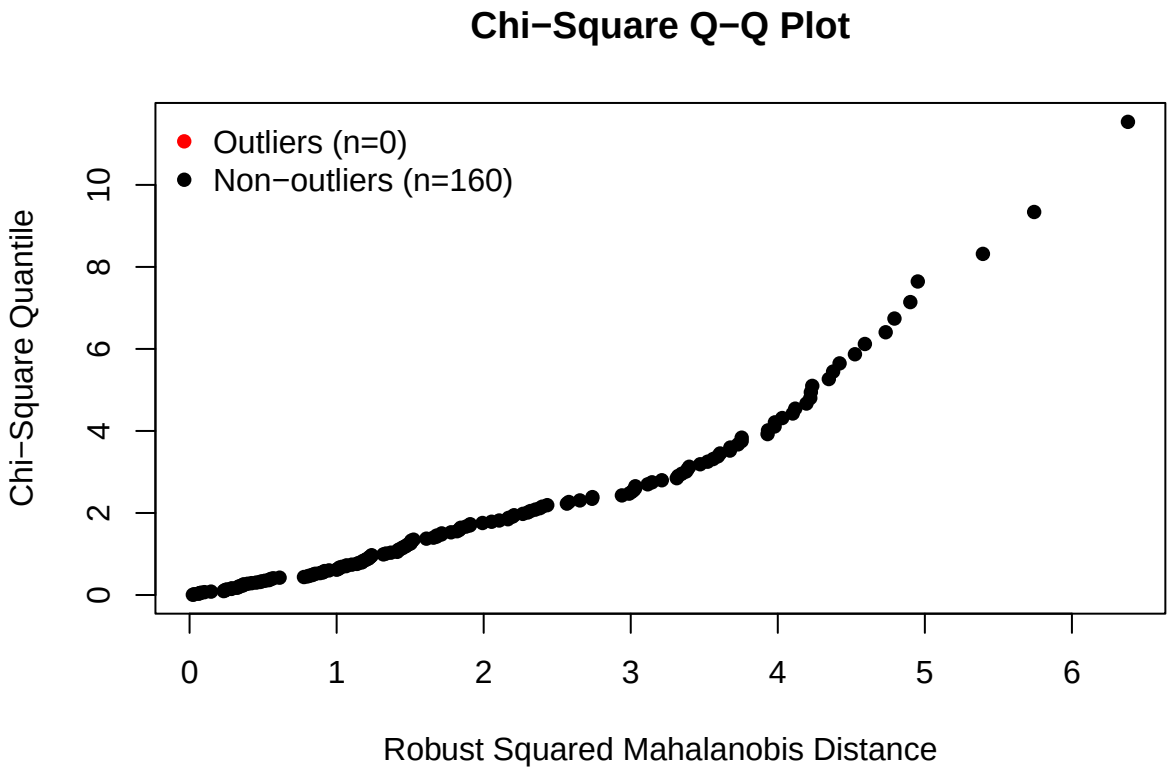
Las variables **z** y **w** no se distribuyen según una ley normal para el **grupo A** (p-valor <0.05).

Test de normalidad multivariante (Mardia, Henze-Zirkler y Royston)

El paquete **MVN** contiene funciones que permiten realizar los tres test que se utilizan habitualmente para **contrastar la normalidad multivariante**.

Esta normalidad multivariante puede verse afectada por la presencia de outliers. En este paquete también encontramos **funciones para el análisis de outliers**.

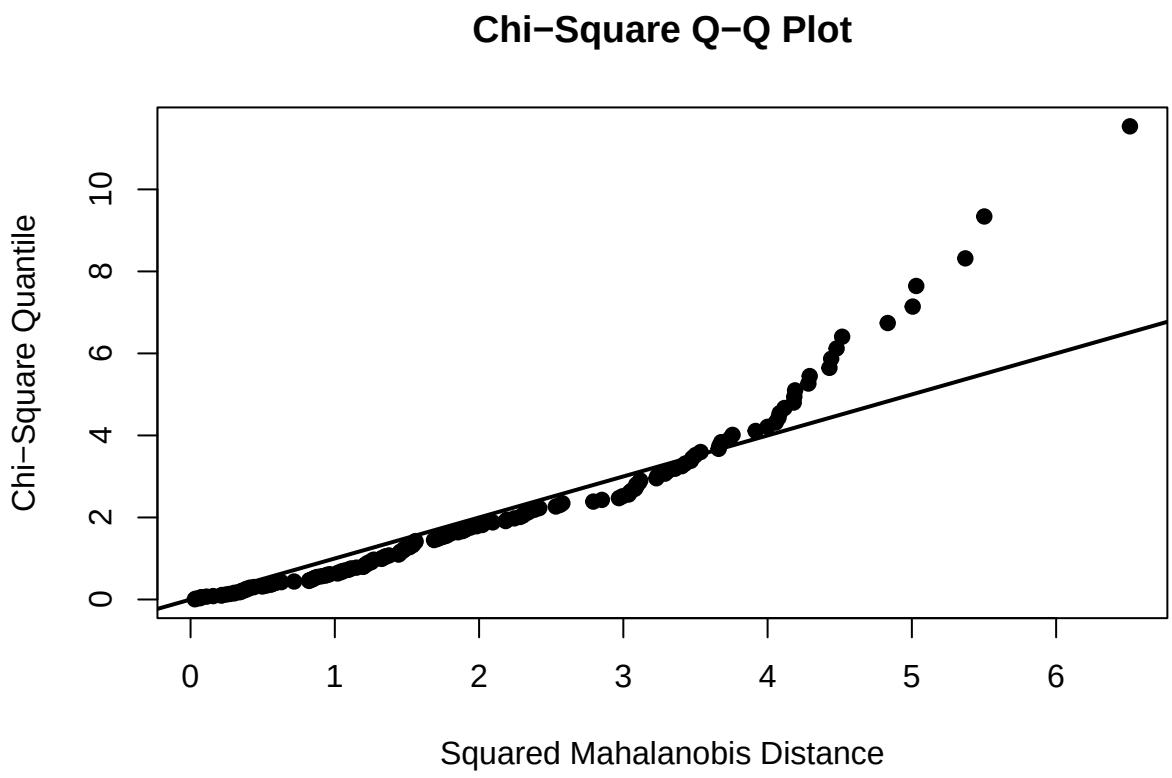
```r
outliers <- mvn(data = datos_train[,-3], mvnTest = "hz", multivariateOutlierMethod = "quan")
```



En este caso no se detectan outliers. Los dos test realizados a continuación encuentran evidencias, al 5% de significación, de falta de normalidad multivariante.

Si bien es cierto que, tal y como se deduce de las salidas siguientes, **no se cumple el supuesto de normalidad multivariante**, el análisis **discriminante cuadrático tiene cierta robustez** en este caso, aunque conviene tenerlo en cuenta en las conclusiones del análisis.

```r
# Royston multivariate normality test
royston_test <- mvn(data = datos_train[,-3], mvnTest = "royston", multivariatePlot = "qq")
```




```
royston_test$multivariateNormality
```

```
##      Test        H      p value MVN
## 1 Royston 23.06759 9.792823e-06  NO
```

```
# Henze-Zirkler multivariate normality test
hz_test <- mvn(data = datos_train[,-3], mvnTest = "hz")
hz_test$multivariateNormality
```

```
##            Test       HZ      p value MVN
## 1 Henze-Zirkler 4.951702 6.526013e-11  NO
```

Homogeneidad de la varianza

Se procede como la *Sección 1.3* anterior.

La **hipóstesis nula** a contrastar es la de **igualdad de matrices de covarianzas** en todos los grupos.

```
boxM(data = datos_train[, -3], grouping = datos_train[, 3])
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  datos_train[, -3]
## Chi-Sq (approx.) = 81.024, df = 3, p-value < 2.2e-16
```

En este caso rechazamos la hipótesis nula ya que **p-value $<$ 0.001** y por tanto **asumimos la NO homogeneidad de varianzas**.

Es importante recordar que para que esta **conclusión sea fiable** debe darse el supuesto de **normalidad**

multivariante, que en este caso no se da. De hecho cuando no existe distribución normal multivariante este test siempre sale significativo y por tanto no es fiable.

Función discriminante

Pese a que no se verifica el supuesto de normalidad multivariante, teniendo en cuenta que las varianzas no son homogéneas, **se procede a ajustar un modelo discriminante cuadrático** porque es robusto frente a la falta de este supuesto, aunque hay que tenerlo presente ante la posiblidad de obtener resultados inesperados.

La función **qda** del paquete **MASS** realiza la clasificación.

```
modelo_qda <- qda(grupo ~ variable_z + variable_w, data = datos_train)
modelo_qda
```

```
## Call:
## qda(grupo ~ variable_z + variable_w, data = datos_train)
##
## Prior probabilities of groups:
##   A   B
## 0.5 0.5
##
## Group means:
##   variable_z variable_w
## A  0.5000000   4.862263
## B  0.5170937   2.030475
```

La salida de este objeto, nos muestra las **probabilidades a priori** de cada grupo, en este caso 0.5 y las **medias de cada regresor por grupo**.

Una vez construido el clasificador podemos clasificar nuevos datos en función de sus medidas sin más que llamar a la función **predict**. Por ejemplo, **vamos a clasificar todas las observaciones del dataset test**.

```
nuevas_observaciones <- datos_test
predict(object = modelo_qda, newdata = nuevas_observaciones)
```

```
## $class
##  [1] A A A A A A A A A A A A A A A A A A A A A B B A B B B B B B B A A B B B B B B
## [39] B A
## Levels: A B
##
## $posterior
##              A            B
## 3   0.99572804 4.271960e-03
## 4   0.88076443 1.192356e-01
## 9   0.87488971 1.251103e-01
## 12  0.91643185 8.356815e-02
## 15  0.99961804 3.819638e-04
## 16  0.99753642 2.463576e-03
## 20  0.99992181 7.819121e-05
## 41  0.99999997 3.024630e-08
## 52  0.99911537 8.846308e-04
## 60  0.99808348 1.916519e-03
## 64  0.99999270 7.302331e-06
## 66  0.99880428 1.195717e-03
## 68  0.99999951 4.908639e-07
## 69  0.93604603 6.395397e-02
## 73  0.99999816 1.842343e-06
```

```
## 75  0.99968839 3.116069e-04
## 78  0.99963898 3.610191e-04
## 89  0.99441459 5.585415e-03
## 96  0.96333973 3.666027e-02
## 100 0.97329074 2.670926e-02
## 105 0.03509319 9.649068e-01
## 107 0.01724841 9.827516e-01
## 108 0.57241519 4.275848e-01
## 115 0.08518801 9.148120e-01
## 118 0.45297454 5.470255e-01
## 136 0.11209239 8.879076e-01
## 137 0.04346586 9.565341e-01
## 138 0.01808141 9.819186e-01
## 143 0.05273681 9.472632e-01
## 144 0.18804503 8.119550e-01
## 150 0.86695494 1.330451e-01
## 155 0.60687367 3.931263e-01
## 161 0.44664763 5.533524e-01
## 166 0.02205248 9.779475e-01
## 167 0.04044205 9.595580e-01
## 169 0.04370428 9.562957e-01
## 173 0.02323925 9.767607e-01
## 187 0.48079614 5.192039e-01
## 192 0.01986393 9.801361e-01
## 195 0.79768574 2.023143e-01
```

Por ejemplo, según la función discriminante, la probabilidad a posteriori de que el primer registro del conjunto test esté en el **grupo A** es del 99,6% mientras la de que esté en el **grupo B** es inferior al 0.1%. Por tanto este dato sería clasificado en el **grupo A**. Se razona del mismo modo con el resto de elementos del conjunto test.

Validación del modelo

La función **confusionmatrix** del paquete **biotools** realiza una validación cruzada del modelo de clasificación.

```
pred <- predict(object = modelo_qda, newdata = datos_test)
confusionmatrix(datos_test$grupo, pred$class)
```

```
##   new A new B
## A    20     0
## B     4    16
```

```
# Classification error percentage
trainig_error <- mean(datos_test$grupo != pred$class) * 100
paste("trainig_error=", trainig_error, "%")
```

```
## [1] "trainig_error= 10 %"
```

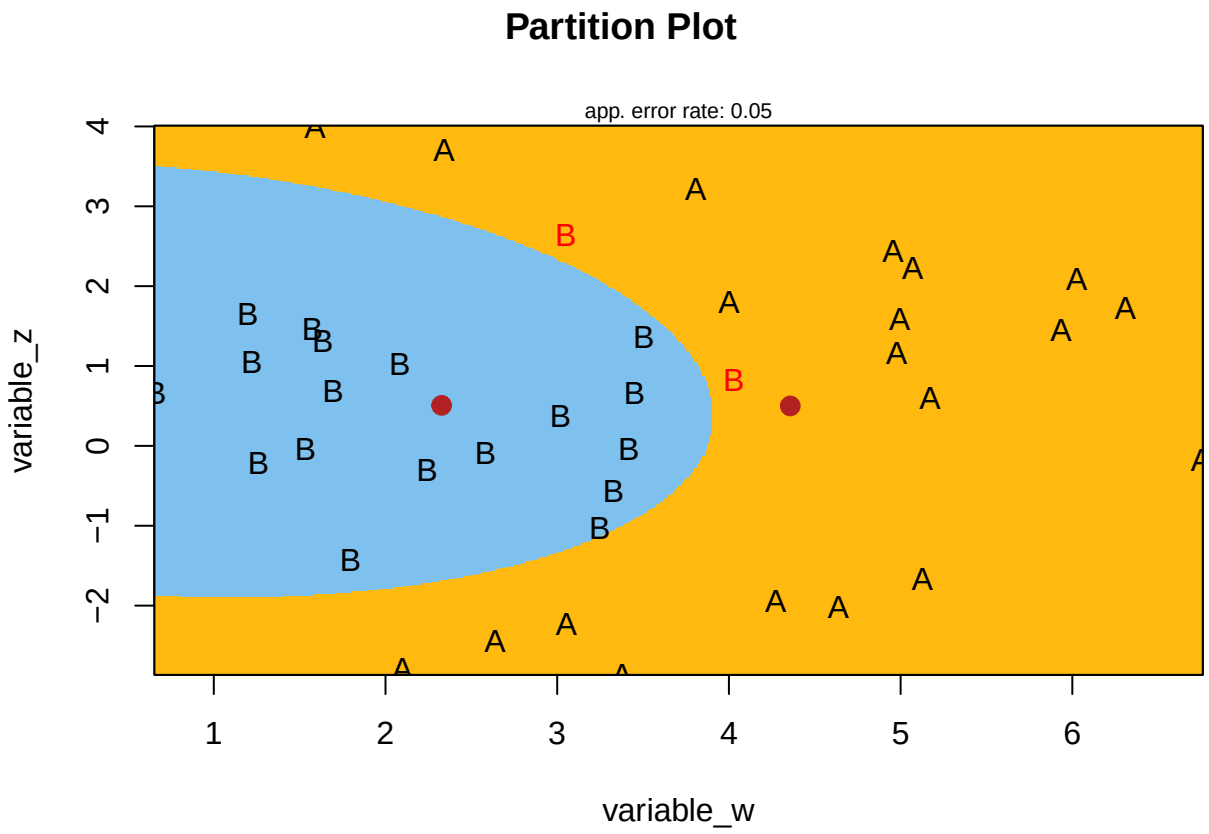
En este caso el porcentaje de aciertos es del 90%.

Visualización de las clasificaciones

La función **partimat** del paquete **klaR** permite representar los **límites de clasificación** de un modelo discriminante lineal o cuadrático para cada par de predictores. Cada color representa una región de clasificación acorde al modelo, **se muestra el centroide** de cada región y el **valor real de las observaciones**.

```
partimat(formula = grupo ~ variable_z + variable_w, data = datos_test,
         method = "qda", prec = 400,
```

```
        image.colors = c("darkgoldenrod1", "skyblue2"),
        col.mean = "firebrick")
```



LDA vs. QDA

El clasificador más adecuado depende de las implicaciones que tenga asumir que todos los grupos comparten una matriz de covarianza común ya que este supuesto puede producir un sesgo en las clasificaciones o producir varianzas altas.

- LDA produce **límites de decisión lineales**, lo que se traduce en **menor flexibilidad** y por lo tanto menor problema de varianza.
- QDA produce **límites cuadráticos** y por lo tanto curvos, lo que aporta mayor flexibilidad permitiendo ajustarse mejor a los datos, menor sesgo pero mayor riesgo de varianza.
- En términos generales, **LDA tiende a conseguir mejores clasificaciones que QDA cuando hay pocas observaciones con las que entrenar al modelo**, escenario en el que evitar la varianza es crucial.
- Si se dispone de una **gran cantidad de observaciones de entrenamiento** o si no es asumible que existe una matriz de covarianza común entre clases, **QDA es más adecuado**.
- Si se dispone de **p predictores**, calcular una matriz de **covarianza común requiere estimar p(p+1)/2 parámetros**, mientras que **calcular una matriz diferente para cada grupo requiere de Kp(p+1)/2** (K es el número de grupos de la variable respuesta). Para valores de **p muy altos**, la elección del método puede estar limitada por la **capacidad computacional**.