UVA-FACULTAD DE CIENCIAS

GRADO EN ESTADISTICA.

TRABAJO FIN DE GRADO.

ANÁLISIS DISCRIMINANTE: UN ESTUDIO DE SIMULACIÓN.

ISABEL ESTÉVEZ.

TUTOR: BONIFACIO SALVADOR.

02/09/2013

INDICE:

Página

1. Introducción

En este proyecto se va a analizar el problema del análisis Discriminante, en el cual se busca clasificar a ciertos individuos en dos o más grupos o poblaciones conocidos. De estos individuos se conocen ciertas características que determinarán el grupo al que pertenecen. La clasificación en cada uno de los grupos se realiza mediante la creación de una regla de clasificación o regla discriminante que permite asignar cada individuo a uno de los grupos.

El objetivo de este proyecto es la comparación de tres reglas discriminantes, mediante el cálculo de los errores obtenidos con cada una de ellas. Esta comparación se realizará mediante simulación.

A lo largo del proyecto se explicará el problema del análisis discriminante, apartado 2, así como las reglas más utilizadas: la Regla de Bayes, utilizada en poblaciones con distribuciones conocidas, explicada en detalle en el apartado 3 junto con la distancia de Mahalanobis, base de ciertos cálculos para la obtención de reglas en poblaciones normales. Estas reglas para poblaciones normales son la Regla Lineal Discriminante de Fisher y la Regla Discriminante Cuadrática, explicadas en el apartado 4 ambas. También se estudia el caso en el que las poblaciones no sean conocidas, utilizando el método de la Regla Núcleo mediante la estimación de la densidad a partir de una función núcleo y una matriz de suavizado, explicada en el apartado 5.

Para hacer las comparaciones de las tres reglas, es necesaria la obtención de los errores que comete cada una de ellas. Por tanto, en el apartado 6 se analizan distintos métodos de obtención de estimadores para la tasa de error efectiva: Tasa de error aparente, método de validación cruzada o cross-validation y método bootstrap.

Para la aplicación de los supuestos, en el apartado 7 se detallan las simulaciones a realizar utilizando cada una de las reglas en distintos escenarios y calculando sus tasas de error. Además, se hace un estudio de los errores estimados comparándolos con los errores "reales" calculados anteriormente. Además, en los apartados 7-f y 7-g se muestran los resultados obtenidos y las conclusiones.

Para finalizar, se adjunta un apéndice, apartado 9, con las funciones creadas para la obtención de los resultados, tablas y gráficos. Éste es precedido de un apartado para la bibliografía, apartado 8.

2. Introducción al Análisis Discriminante.

El análisis discriminante tiene que ver con la relación entre una variable categórica Y y un vector $X = (X_1, \dots, X_p)$ de variables interrelacionadas. Se considera un número finito k de poblaciones, categorías, clases o grupos $\Pi_1, \Pi_2, .., \Pi_k$. La existencia de estas poblaciones se conoce a priori. Cada ítem o individuo se supone que pertenece a una y solo una de las poblaciones. La variable categórica Y determina la pertenencia de cada ítem a una población. El vector p-dimensional X, contiene las p medidas de cada ítem.

El problema consiste en estimar Y en base a X, es decir, conocido el valor del vector X en un ítem o individuo asignar dicho ítem o individuo a una de las k poblaciones.

Denotamos por $p(\Pi_i) = \pi_i$, i=1,2,…,k, las probabilidades a priori, es decir, el conocimiento previo de las probabilidades de pertenencia a cada población.
Una *regla de clasificación* es una aplicación del espacio R^k en el conjunto $\{1,2, \dots, k\}$.Las reglas de clasificación se construyen a partir de las distribuciones del vector X en cada población y de las probabilidades a priori de las poblaciones.

Denotamos por $f_i(.)$ la función de densidad del vector X en la población Π_i , i=1,…,k. Cuando estas densidades son conocidas, la regla de Bayes, (que describiremos más adelante) es la regla óptima, ya que minimiza la probabilidad de clasificación errónea.

Habitualmente en las aplicaciones estas densidades no son conocidas. En estos casos necesitamos información muestral, es decir, un conjunto de ítems o individuos para los que conocemos tanto la variable Y (población a la que pertenecen) como el vector X. Estos datos constituyen la "muestra entrenamiento" que denotamos por X_e. Construiremos reglas de clasificación, basadas en esta información muestral, con las que podemos clasificar nuevos ítems o individuos para los que conozcamos el vector X.

En el caso de densidades desconocidas, y que podamos asumir un modelo paramétrico, la regla óptima es la regla de Bayes sustituyendo los parámetros por los correspondientes estimadores máximo verosímiles.
Si además podemos asumir distribuciones normales en las poblaciones se obtienen dos reglas de clasificación, la regla de Fisher o regla lineal discriminante y la regla cuadrática.
La regla de Fisher se obtiene en el caso en el que las matrices de covarianzas para cada población puedan considerarse iguales, y se obtiene mediante la determinación de una dirección discriminante, la cual separa de manera óptima los grupos. También llegamos al mismo resultado con la regla de Bayes sustituyendo los parámetros desconocidos por sus estimadores máximo verosímiles .
La regla cuadrática se obtiene cuando no podemos suponer la igualdad de las matrices de covarianzas. Esta regla se calcula mediante la regla de mínima distancia, la cual se basa en el cálculo de la regla de Mahalanobis (más adelante explicada) para cada una de las poblaciones. También puede obtenerse, al igual que la lineal, mediante la regla de Bayes

sustituyendo los parámetros desconocidos por sus estimadores máximo verosímiles, ahora teniendo en cuenta que las matrices de covarianzas son distintas.
Sin embargo, no siempre puede asumirse un modelo paramétrico, si nos encontramos en este caso recurriremos a métodos no paramétricos como la Regla Núcleo, la cual nos permitirá crear una regla de clasificación mediante la estimación de las densidades, a partir de una función núcleo y una matriz de suavizado.
El objetivo de este proyecto es la comparación de las tres reglas discriminantes. Para lo cual calcularemos el error cometido por cada una, siendo la mejor regla la que minimice el error obtenido.
El cálculo de este error se realiza mediante simulación, en la cual se tomará un número elevado de individuos provenientes de la misma distribución que la muestra de entrenamiento. Por tanto, de estos datos conocemos la variable Y y el vector X. Estos datos constituyen la 'muestra test'.
La clasificación, con cada una de las tres reglas, de los individuos de la muestra test nos permitirá obtener el error mediante la proporción de individuos mal clasificados. Este error será el error "real".

Sin embargo, en las aplicaciones reales no disponemos de la posibilidad de simular una muestra test, por lo que analizamos tres maneras de estimar el error con la muestra de entrenamiento. Primero, clasificamos la muestra de entrenamiento y calculamos la proporción de individuos mal clasificados. Este error es el llamado "error aparente". Es un error muy sesgado ya que se clasifican los mismos individuos con los que se obtiene la regla discriminante.
Con el fin de disminuir este sesgo, se utilizan los método de "cross-validation" y método bootstrap, los cuales clasifican elementos de la muestra de entrenamiento que no han sido utilizados en la creación de la regla discriminante. Ambos se explican en detalle en el apartado 7.

La comparación de las tres estimaciones de los errores con el error real permitirá elegir el método deseado.

3. Regla de Bayes y distancia de Mahalanobis:

Regla de Bayes:

Clasifica al individuo en la clase cuya probabilidad a posteriori es la más alta.

X_0 se clasifica en Π_1 si y solo si $P(\Pi_1/x= x_0)> P(\Pi_2 /x= x_0)$

$$P(\Pi_1|(x = x_0)) = \frac{P(x=x_0|\Pi_1)P(\Pi_1)}{P(x=x_0|\Pi_1)P(\Pi_1)+P(x=x_0|\Pi_2)P(\Pi_2)} = \frac{\pi_1 f_1(x_0)}{\pi_1 f_1(x_0)+\pi_2 f_2(x_0)}$$

Por tanto, X_0 se clasifica en Π_1 si y solo si

$$P(\Pi_1/x= x_0)> P(\Pi_2 /x= x_0) \leftrightarrow \pi_1 f_1(x_0) > \pi_2 f_2(x_0) \leftrightarrow \frac{f_1(x_0)}{f_2(x_0)} > \frac{\pi_2}{\pi_1}$$

$$\frac{f_1(x_0)}{f_2(x_0)} > \frac{\pi_1}{\pi_2}$$

Tomando logaritmos: $B(x) = \log(f_1(x_o)) - \log(f_2(x_0)) - \log(\frac{\pi_1}{\pi_2})$

Que nos lleva a la Regla de Bayes:

$$x_0 \in \Pi_1 \ Si\ B(x) > 0$$
$$x_0 \in \Pi_2 \ Si\ B(x) < 0$$

Regla óptima de Bayes

Para el caso de k grupos:

$$P(\Pi_i|(x = x_0)) = \frac{P(x = x_0|\Pi_i)P(\Pi_i)}{\sum_{j=1}^{k} P(x = x_0|\Pi_j)P(\Pi_j)}$$

Clasificamos x_0 *en* Π_i, *para la que* $P(\Pi_i|(x = x_0))$ *sea máxima*:

$$x_0 \in \Pi_i \ si\ P(\Pi_i|(x = x_0)) = \max P\left(\Pi_j\middle|(x = x_0)\right) \quad j = 1..k$$

Utilizaremos esta regla para clasificar en dos o más grupos en poblaciones con distribución conocida.

Distancia de Mahalanobis:

La distancia de Mahalanobis M^2 es una medida generalizada de distancia entre dos grupos que tiene en cuenta la posición central (centro de gravedad) y las dispersiones (matrices de productos cruzados o de covarianzas intragrupos) de los grupos.

$$M_{i,j}^2 = (x_i - x_j)' \Sigma_{i,j}^{-1} (x_i - x_j)$$

$\Sigma_{i,j}$: matriz de covarianzas intragrupos.

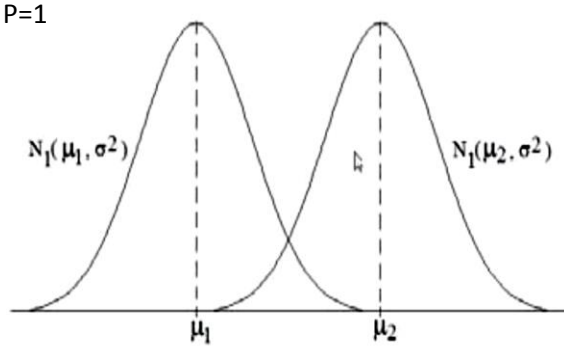
$$\Sigma_{i,j} = cov(x_i, x_j)$$

4. Reglas de clasificación en poblaciones normales:

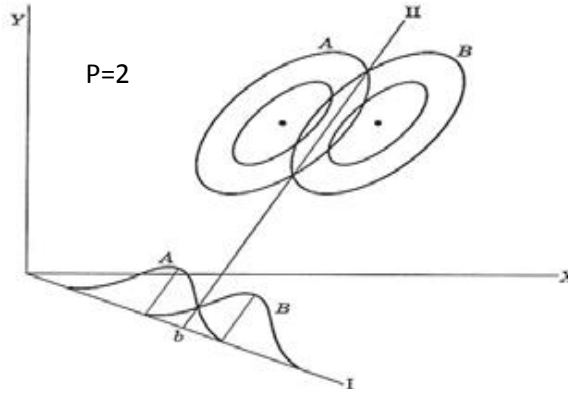
DOS poblaciones $N_p(\mu_i, \Sigma_i)$ $\quad$ $\Sigma's\ iguales$: $\Sigma_1 = \Sigma_2 = \Sigma$

→ ***Función Lineal Discriminante de Fisher*: LDA**

P=1



P=2



Gráficamente: Se busca una dirección óptima sobre la que proyectar los datos de los grupos conocidos y de los que queremos clasificar. Se clasifica en función de qué grupo está más cerca en esa dirección.
Una dirección óptima tiene que separar bien las medias, pero teniendo en cuenta la variabilidad.
El nuevo dato se clasifica dentro del grupo con la media más próxima en la proyección.

Analíticamente: La idea de Fisher consistió en hallar una combinación lineal de las variables originales de la forma $a_1x_1 + \cdots + a_px_p$ y tal que discrimine "el máximo posible" las dos poblaciones. Él mismo definió el criterio de máxima discriminación como maximizar la razón entre la suma de cuadrados *entre* grupos y la suma de cuadrados *dentro* de los grupos sobre la combinación lineal, esto equivale a resolver: $\max\left(\frac{(a'\bar{x}_2 - a'\bar{x}_1)^2}{a'\Sigma a}\right)$

Este problema tiene infinitas soluciones, todas proporcionales a elegir el vector:

$$w = \Sigma^{-1}(\mu_2 - \mu_1)$$

A este vector es lo que Fisher llamó dirección discriminante, ya que nos va a permitir proyectar en ella los datos a clasificar y poder compararlos con los datos que disponemos sobre los grupos.

Densidades conocidas -> μ_i y Σ conocidas

Población 1: $\Pi_1 \quad X \sim N_p(\mu_1, \Sigma)$ **Población 2:** $\Pi_2 \quad X \sim N_p(\mu_2, \Sigma)$
Nuevo elemento de población desconocida: $x_0 \in \Pi_0 \quad x_0 \sim N_p(\mu_0, \Sigma)$
Problema: ¿ $\Pi_0 \equiv \Pi_1$ ó $\Pi_0 \equiv \Pi_2$?

Sean μ_1, μ_2 los vectores de medias de las variables en Π_1, Π_2, respectivamente, y supongamos que la matriz de covarianzas ∑ es común.

La regla lineal discriminante de Fisher clasificará al individuo x_0 en la población más cercana, lo cual se puede traducir a proyectar el vector sobre la dirección discriminante y compararlo con la proyección de la media de medias de las poblaciones, y:

$$\text{Si } w'x_0 < w'\left(\frac{\mu_1+\mu_2}{2}\right) \quad \text{asignamos } x_0 \text{ a } \Pi_1 \qquad (1)$$

$$\text{En caso contrario} \quad \text{asignamos } x_0 \text{ a } \Pi_2$$

$$\text{Siendo: } w = \Sigma^{-1}(\mu_2 - \mu_1)$$

Definimos la función discriminante desarrollando $w'x_0 - w'\left(\frac{\mu_1+\mu_2}{2}\right)$:

$$\boldsymbol{L(x) = \left[x - \frac{1}{2}(\mu_1 + \mu_2)\right]' \Sigma^{-1}(\mu_2 - \mu_1)}$$

Este es el discriminador lineal de Fisher

Y la regla (1) es

$$\boldsymbol{L(x) = \left[x - \frac{1}{2}(\mu_1 + \mu_2)\right]' \Sigma^{-1}(\mu_2 - \mu_1)} \qquad (2)$$

Si $L(x_0) < 0$ asignamos x_0 a Π_1
En caso contrario asignamos x_0 a Π_2

Veamos cómo obtenemos el mismo resultado mediante la regla de Bayes:

Estamos suponiendo que las distribuciones de las poblaciones son normales con distintos vectores de medias pero idéntica matriz de covarianzas, por tanto, su función de densidad es:

$$f_i(x) = \frac{1}{(2\pi)^{\frac{k}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x-\mu_i)'\Sigma^{-1}(x-\mu_i)\right\}$$

La manera óptima es clasificar en la población Π_1 si $\pi_1 f_1(x) > \pi_2 f_2(x)$

Como ambos términos son siempre positivos, tomando logaritmos y sustituyendo *f(x)* por sus expresiones, obtenemos:

$$\log(\pi_1) - \frac{1}{2}(x-\mu_1)'\Sigma^{-1}(x-\mu_1) > \log(\pi_2) - \frac{1}{2}(x-\mu_2)'\Sigma^{-1}(x-\mu_2)$$

Operando, se llega:

$$\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(x-\mu_2) - \frac{1}{2}(x-\mu_1)'\Sigma^{-1}(x-\mu_1) > \log\frac{\pi_2}{\pi_1}$$

Suponiendo las probabilidades a priori iguales y operando adecuadamente, llegamos a la regla (1)

$$\boldsymbol{L(x) = \left[x - \frac{1}{2}(\mu_1 + \mu_2)\right]' \Sigma^{-1}(\mu_1 - \mu_2) > 0 \rightarrow \text{asignamos x a la población } \Pi_1}$$

Si μ_i *y* Σ son desconocidas y asumimos $\Sigma_1 = \Sigma_2 = \Sigma$

Ahora necesitamos la muestra entrenamiento X_e

Se estiman las medias μ_i mediante las medias muestrales:

$$\bar{X}_i = \frac{\sum_{j=1}^{n_i} x_{ei_j}}{n_i}$$

Se estima Σ mediante un estimador pooled a partir de las desviaciones a la media muestral de cada grupo:

$$S_p = \frac{1}{f}\left[\sum_{i=1}^{n_1}(x_i - \bar{X}_1)(x_i - \bar{X}_1)^t + \sum_{i=1}^{n_2}\left(x_{n_1+i} - \bar{X}_2\right)\left(x_{n_1+i} - \bar{X}_2\right)^t\right], \ con\ f = n_1 + n_2 - 2$$

O bien: $S_p = \frac{(n_1-1)S_1+(n_2-1)S_2}{n_1+n_2-2}$

$$Siendo\ S_1, S_2\ las\ matrices\ de\ varianzas - covarianzas\ muestrales$$

Ahora, en esta nueva versión del discriminador de Fisher sustituyo en (2) S_p en lugar de Σ que ahora es desconocido, y las medias μ_i por sus estimadores, las correspondients medias muestrales $\bar{X}_i$:

$$\boldsymbol{L(x) = \left[x - \frac{1}{2}(\bar{X}_1 + \bar{X}_2)\right]' S_p^{-1}(\bar{X}_2 - \bar{X}_1)}$$

Si $L(x) < 0$ asignamos x_0 a Π_1
En caso contrario asignamos x_0 a Π_2

DOS poblaciones $N_p(\mu_i, \Sigma_i)$ $\Sigma's\ diferentes$: $\Sigma_1 \neq \Sigma_2$

→ ***Función Discriminante Cuadrática: QDA***

Densidades conocidas -> μ_i y Σ_i conocidas

Utilizamos el Criterio de Mínima Distancia: asigno a la población de cuya media diste menos.

Regla de Mínima Distancia es:

$x_0 \in \Pi_1$ *si* $M^2(x_0, \mu_1) < M^2(x_0, \mu_2)$:

$$(x_0 - \mu_1)'\Sigma_1^{-1}(x_0 - \mu_1) < (x_0 - \mu_2)'\Sigma_2^{-1}(x_0 - \mu_2) \quad (4)$$

Veamos cómo obtenemos el mismo resultado mediante la regla de Bayes:

Estamos suponiendo que las distribuciones de las poblaciones son normales con distintos vectores de medias y distintas matrices de covarianzas, por tanto, su función de densidad es:

$$f_i(x) = \frac{1}{(2\pi)^{\frac{k}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x - \mu_i)'\Sigma_i{}^{-1}(x - \mu_i)\right\}$$

La manera óptima es clasificar en la población Π_1 si $\pi_1 f_1(x) > \pi_2 f_2(x)$

Como ambos términos son siempre positivos, tomando logaritmos y sustituyendo *f(x)* por sus expresiones, obtenemos:

$$\log(\pi_1) - \frac{1}{2}(x - \mu_1)'\Sigma_1{}^{-1}(x - \mu_1) > \log(\pi_2) - \frac{1}{2}(x - \mu_2)'\Sigma_2{}^{-1}(x - \mu_2)$$

Operando, se llega:

$$\frac{1}{2}(x - \mu_2)'\Sigma_2{}^{-1}(x - \mu_2) - \frac{1}{2}(x - \mu_1)'\Sigma_1{}^{-1}(x - \mu_1) > \log\frac{\pi_2}{\pi_1}$$

Asumiendo que las probabilidades a priori son iguales $\pi_1 = \pi_2$, llegamos a la misma expresión (4):

$$(x_0 - \mu_1)'\Sigma_1^{-1}(x_0 - \mu_1) < (x_0 - \mu_2)'\Sigma_2^{-1}(x_0 - \mu_2) \quad si \quad x_0 \in \Pi_1$$

Si μ_i ó Σ_i desconocidas y NO asumimos $\Sigma_1 = \Sigma_2$

Cuando desconocemos los parámetros, sustituimos por sus estimadores en (4):

$$x_0 \in \Pi_1 \quad si \quad \widehat{M}^2(x_0,\mu_1) < \widehat{M}^2(x_0,\mu_2):$$

$$(x_0 - \bar{X}_1)' S_1^{-1} (x_0 - \bar{X}_1) < (x_0 - \bar{X}_2)' S_2^{-1} (x_0 - \bar{X}_2) \qquad (5)$$

Siendo S_1, S_2 las matrices de varianzas − covarianzas muestrales.
$\bar{X}_1, \bar{X}_2$ las medias muestrales

k poblaciones $N_p(\mu_i, \Sigma_i)$

Asumiendo $\Sigma_1 = \cdots = \Sigma_k = \Sigma$ (desconocidas)

$\bar{X}_i$: *vector media muestral basado en n_i observaciones de la población i.*

S_p: *matriz de covarianzas muestral 'pooled' con* $f = \sum_{i=1}^{k} n_i - k$ $g.l.$

Nuevo elemento: $\boldsymbol{x_0 \sim N_p(\mu_0, \Sigma)}$

Se asigna el nuevo elemento a la población más próxima:
Regla de Discriminación lineal:

$$x_0 \in \Pi_i \ si \ \bar{X}_i^t S_p^{-1} x_0 - \frac{1}{2}\bar{X}_i^t S_p^{-1} \bar{X}_i = \sup_{j=1..k} \bar{X}_j^t S_p^{-1} x_0 - \frac{1}{2}\bar{X}_j^t S_p^{-1} \bar{X}_j \qquad (6)$$

Sin asumir $\Sigma_1 = \cdots = \Sigma_k$ (desconocidas)

Asignamos de forma similar, pero utilizando en la regla (6) cada S_i en lugar del estimador común S_p:

$$x_0 \in \Pi_i \ si \ \bar{X}_i^t S_i^{-1} x_0 - \frac{1}{2}\bar{X}_i^t S_i^{-1} \bar{X}_i = \sup_{j=1..k} \bar{X}_j^t S_j^{-1} x_0 - \frac{1}{2}\bar{X}_j^t S_j^{-1} \bar{X}_j \qquad (7)$$

5. Métodos no paramétricos:

Las reglas de clasificación vistas en las secciones anteriores parten de los supuestos de que las distribuciones son normales multivariantes dentro de cada grupo. Sin embargo, cuando no podemos asumir ningún modelo paramétrico, existen otros métodos para poder realizar clasificaciones sin partir de estos supuestos. Los más conocidos son los métodos de vecinos próximos y los basados en estimación no paramétrica de la densidad, en este proyecto nos centraremos en el segundo:

Estimación de densidades. Estimador núcleo:

Fix y Hodges (1951), proponen el análisis discriminante no-paramétrico como una solución natural al problema cuando no se conocen las funciones de densidad de las poblaciones
El procedimiento empleado seria el mismo, usando en lugar de las funciones de densidad, las funciones estimadas con los dos conjuntos de datos muéstrales iniciales y sustituyendo por estas estimaciones, las funciones de la ecuación correspondiente.
Cuanto mejor sea la estimación de la función de densidad, mejor será la regla discriminante. Un estudio importante acerca de este procedimiento es el contenido en el trabajo de Remme, Habbema y Hermans (1980), en que el método no paramétrico usado para las estimaciones de las densidades en el análisis discriminante es la estimación núcleo. Se estiman las funciones de forma independiente para cada población, utilizando el método de validación cruzada máximo verosímil para obtener un valor del parámetro de suavizado, obteniendo buenos resultados con la estimación núcleo en todos los casos investigados.

Sean $X_1, \dots, X_n$, variables aleatorias independientes e idénticamente distribuidas, con densidad f(.). Definimos el estimador núcleo de la densidad univariante *f(.),* con función núcleo K y parámetro de suavizado h>0

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

Siendo la generalización al modelo multivariante la siguiente:
Sean $X_1, \dots, X_n$, vectores aleatorios, p dimensionales, independientes e idénticamente distribuidos, con densidad p-variante *f(.)*

$$\hat{f}(x) = \frac{1}{n|H|}\sum_{i=1}^{n} K_p(H^{-1}(x - X_i))$$

Donde H es una matriz pxp no singular, llamada matriz de suavizado y |H| es el valor absoluto del determinante de H, $K_p(t)$ es una función núcleo p-dimensional.
A continuación se comentan algunos ejemplos de matrices H, y la utilizada en este proyecto:

- Un cambio de escala global: $H = hI_d, h > 0$. El suavizado es el mismo en todas las direcciones del espacio.
- Un cambio de escala distinto en cada dimensión: $H = diag(h_1, \dots, h_p) \quad h_j > 0$. El suavizado puede ser diferente en cada una de las direcciones coordenadas.

- H matriz no singular. El suavizado puede ser diferente en las diferentes direcciones del espacio.

Una práctica usual es usar un núcleo producto que, dado K un núcleo univariante, se define como $K_p(u_1, \dots, u_p) = \prod_{j=1}^{p} K(u_j)$

Esta última manera, junto con la elección de $H = diag(h_1, \dots, h_p) \quad h_j > 0$ es la más utilizada por los paquetes estadísticos y es la que utilizaremos en este proyecto, tomando como núcleo univariante el núcleo gaussiano. -> $K(u) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}u^2)$

El estimador núcleo de la densidad queda de la forma:

$$\hat{f}(x) = \frac{1}{n \prod_{j=1}^{d} h_j} \sum_{i=1}^{n} \prod_{j=1}^{p} K\left(\frac{x_i - x_{ij}}{h_j}\right) \qquad (7)$$

$$siendo\ h_j\ el\ parámetro\ de\ suavizado\ para\ la\ coordenada\ j$$

La elección de la matriz de suavizado en este proyecto se realiza mediante la minimización de los errores reales.

Se detalla el procedimiento a continuación, para dos o más poblaciones, en el caso multivariante.

Dos poblaciones:

Resumimos el procedimiento:

1. Cálculo de los estimadores núcleo de la densidad para cada uno de los grupos,
$$\widehat{f_1}(\cdot, H_1)$$
$$\widehat{f_2}(\cdot, H_2)$$
2. Clasificar x_0 en el grupo 1 si y solo si,
$$\pi_1 \widehat{f_1}(\cdot, H_1) - \pi_2 \widehat{f_2}(\cdot, H_2) \geq 0$$

$$siendo\ \pi_1\ y\ \pi_2\ las\ probabilidades\ a\ priori\ respectivas.$$

Esta clasificación tiene un inconveniente porque vemos que depende de las matrices de suavizado H_1 y H_2 y no hay una manera óptima para elegir esas matrices. Por tanto, elegiremos los parámetros de suavizado tales que la estimación de la densidad de la forma (7) minimice los errores reales. Para ello se calcula la regla discriminante para una secuencia de valores, centrados en el parámetro de suavizado normal, se obtienen los errores reales para cada parámetro y se escoge el parámetro con el cual se ha obtenido el mínimo error de la secuencia.

K poblaciones:

Dado un conjunto de poblaciones $\Pi_1, \dots, \Pi_k$ y un individuo problema x_0 representado por un vector de observaciones x_0. Supongamos conocidas las probabilidades a priori π_i de que el individuo pertenezca a una de las *k* poblaciones y, siendo las probabilidades del individuo condicionadas a las poblaciones $f_i(\mathrm{x})$. La asignación del individuo problema a una de las poblaciones se efectuará según la regla:

$$\mathrm{x}_0 \in \Pi_i \leftrightarrow P(\Pi_i|\mathrm{x}_0) = máx\{\, P(\Pi_1|\mathrm{x}_0), \dots, P(\Pi_k|\mathrm{x}_0)\,\}$$

Donde las probabilidades condicionadas de las diferentes poblaciones se obtienen a través de la regla de Bayes:

$$P(\Pi_i|\mathrm{x}_0) = \frac{f_i(\mathrm{x}_0)\pi_i}{\sum_{j=1}^{k} f_j(\mathrm{x}_0)\pi_j}$$

Obteniendo las densidades mediante la estimación de la densidad con la función núcleo, al igual que en el caso de dos poblaciones:

$$\hat{f}_i(x) = \frac{1}{n_i \prod_{j=1}^{d} h_j} \sum_{k=1}^{n_i} \prod_{j=1}^{p} K\left(\frac{x_k - x_{kj}}{h_j}\right)$$

$$\text{Con } K(u) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}u^2)$$

$$siendo\ h_j\ el\ parámetro\ de\ suavizado\ para\ la\ coordenada\ j$$

Dado que el objetivo fundamental es la correcta clasificación de los individuos, la elección de la matriz de suavizado *H* se buscará, análogamente a su obtención en el caso de dos poblaciones, de manera que minimice el error de clasificación real, se calculará la regla discriminante para una secuencia de valores, centrados en el parámetro de suavizado normal, se obtendrán los errores reales para cada parámetro y se escogerá el parámetro con el cual se ha obtenido el mínimo error de la secuencia.

6. Valoración de la función predictiva. Errores de clasificación.

Para la valoración de estas reglas discriminantes, vamos a ver la manera de obtener los errores de clasificación de cada una. Es decir, veamos cuántas observaciones en proporción cada regla clasifica en el grupo incorrecto. En la siguiente sección se ven en detalle los métodos utilizados en este proyecto para el cálculo de los errores.

En esta sección vamos a ver tres maneras de obtener el error de clasificación para las reglas discriminantes utilizadas. Para la realización práctica de estos errores, la muestra de la que partimos, que será la que usaremos para el cálculo de las tasas de error, es la muestra de entrenamiento, de la cual conocemos los grupos a los que pertenece cada individuo en ella.

Tasa de error efectiva o probabilidad de clasificación errónea (pce):

En la decisión de identificar la clase a la que pertenece x_0, nos equivocaremos si asignamos x_0 a una población a la que no pertenece. Considerar la regla de clasificación que clasifica x_0 en la población Π_1 si y solo si $x_0 \in A$, siendo A un subconjunto de R^p.

La probabilidad de clasificación errónea (pce) de la regla de clasificación es:

$$\text{pce}=P(2|1)\pi_1 + P(1|2)\pi_2$$

Siendo

$$P(2|1) = \Pr(X \notin A \,/\, \Pi_1) = 1 - \int_A f_1(\mathrm{x})dx \quad \text{y} \quad P(1|2) = \Pr(X \in A \,/\, \Pi_2) = \int_A f_2(\mathrm{x})dx$$

En la práctica no dispondremos de estas probabilidades, por lo que será necesario estimarlas, para ello los siguientes métodos son útiles y serán los empleados en este proyecto.

Errores de clasificación estimados:

Tasa de error aparente o estimación por re-sustitución:

Se obtendrá la tasa de error aparente como el número de individuos, de la muestra entrenamiento mal clasificados, dividido por el número total de individuos.
En la siguiente tabla vemos el número de individuos sabiendo de que población eran originalmente y a qué población han sido clasificados, para dos poblaciones. Por tanto, el error será la suma de los mal clasificados entre el total.
Sea n_{ij}: $elementos\ de\ la\ población\ i\ clasificados\ en\ la\ población\ j$.

		Población clasificada por la regla		Total
		1	2	
Población original	1	n_{11}	n_{12}	n_1
	2	n_{21}	n_{22}	n_2

$\hat{p}(2\backslash 1) = \frac{n_{12}}{n_1}$ Probabilidad de error de la población 1

$\hat{p}(1\backslash 2) = \frac{n_{21}}{n_2}$ Probabilidad de error de la población 2

Estimación de la probabilidad global de error $\rightarrow \hat{p}(error) = \frac{n_{12}+n_{21}}{n_1+n_2}$

Es un estimador muy sesgado, ya que clasificamos las observaciones que ya teníamos. Por tanto, intentaremos disminuir este sesgo con los dos siguientes métodos.

Estimaciones de validación cruzada (Cross validation)

Este método consiste en clasificar cada observación de la muestra entrenamiento con la regla de clasificación basada en el resto de las observaciones.

Partiendo de la muestra entrenamiento $X_{e1}, \dots, X_{en}$, obtendremos el error de validación cruzada con los siguientes pasos:

- Tomo la observación X_{e1} y la separo de los datos. Por lo que mis datos quedan $X_{e2}, \dots, X_{en}$
- Con estos datos $X_{e2}, \dots, X_{en}$ obtengo una regla de clasificación.
- Clasifico la observación X_{e1}
- Repito este proceso para cada una de las observaciones restantes $X_{e2}, \dots, X_{en}$

Por tanto, tengo las n observaciones clasificadas según las n reglas obtenidas con los datos restantes. Calculo la proporción de esta observaciones que han sido mal clasificadas y ese será el estimador del error por validación cruzada.

Corrección mediante bootstrap.

Se trata de una corrección del sesgo de las tasas de error aparentes. Veamos con dos poblaciones el proceso, el cual podría generalizarse fácilmente.

Siguiendo las ideas del bootstrap:

- Extraemos muestras bootstrap tomadas aleatoriamente con reemplazamiento de la muestra de entrenamiento, de la forma:

$$X_1^*, \dots, X_{n_1}^* \; iid \; de \; la \; distribución \; equiprobable \; sobre \; X_{e1}, \dots, X_{en_1}$$

$$X_{n_1+1}^*, \dots, X_{n_2}^* \; iid \; de \; la \; distribución \; equiprobable \; sobre \; X_{e(n_1+1)}, \dots, X_{en_2}$$

- Para cada una de las muestras bootstrap obtengo una regla discriminante.
- Clasifico los elementos de la muestra de entrenamiento que no han sido incluidos en las muestras bootstrap.
- Calculo el error cometido en esta clasificación.
- Repito el proceso un número elevado de veces, N.

Tengo N errores, de los cuales puedo tomar la media y tendré el error bootstrap deseado.

Para k poblaciones o grupos, el proceso es análogo, tomando muestras aleatorias con reemplazamiento de cada grupo y utilizando las reglas discriminantes adecuadas en cada uno de los casos.

7. Parte práctica:

En esta sección se desarrollan los métodos utilizados para las simulaciones de los datos con el fin de validar y comparar las distintas reglas discriminantes.

Como esquema general, el método para la obtención de los errores de cada regla será el siguiente:

1. Creamos la regla de Bayes.
2. Generamos las muestras de entrenamiento de las poblaciones. Para ello vamos a tomar muestras de tamaño n de cada una de las poblaciones.
3. Creamos las tres reglas estimadas: LDA, QDA y la Regla Núcleo, en base a esa muestra de entrenamiento.
4. Generamos las muestras test de las poblaciones. Estas muestras deben ser de tamaño suficientemente grande para que la estimación del error sea suficientemente precisa. Las tomamos de tamaño 2000 en cada población.
5. Clasificamos la muestra test con la regla de Bayes y con cada una de las tres reglas creadas en el paso 3. Y para cada regla obtenemos el error real como la proporción de mal clasificados.

Este esquema lo vamos a realizar en seis escenarios distintos para dos poblaciones con p=2 y para tres escenarios con p=3. Se realizará para tres tamaños muestrales de n=20, n=200 y n=500 el caso bidimensional y para un único tamaño muestral de n=20 en el caso tridimensional. También se realizan 3 escenarios distintos para tres poblaciones con p=2 y para los tamaños muestrales de n=20, n=200 y n=500.

El error obtenido con la regla de Bayes es el error de Bayes (óptimo).El error obtenido con cada una de las tres reglas es el error "real", sin embargo, en las aplicaciones reales no va a ser posible generar la muestra test, por tanto, recurrimos a los errores estimados.
Tras la comparación de los errores en cada uno de los escenarios, con cada tamaño muestral y para cada una de las reglas, vamos a estudiar la efectividad de los errores estimados: realizaremos el error aparente, el error por validación cruzada y el error mediante bootstrap. Se harán las simulaciones de la siguiente manera:
Basado únicamente en la muestra entrenamiento, se calculan cuatro estimadores del error real, para cada una de las tres reglas de clasificación construidas en el paso 2 anterior y en cada escenario considerado.

a. Error aparente: Para cada regla, se calcula como la proporción de elementos de la muestra de entrenamiento mal clasificados.
b. Error cross-validation: Para cada regla, se realiza la validación cruzada correspondiente con los datos de la muestra de entrenamiento. Recordamos los pasos, siendo $X_{e1}, \dots, X_{en}$ la muestra de entrenamiento tomada:
    i. Tomo la observación X_{e1} y la separo de los datos.
    ii. Con los datos restantes $X_{e2}, \dots, X_{en}$ obtengo una regla de clasificación.

iii. Clasifico la observación X_{e1}
iv. Repito este proceso para cada una de las observaciones restantes $X_{e2}, \dots, X_{en}$

Calculo el error cross-validation , como la proporción de observaciones mal clasificadas.

c. Error bootstrap: Para cada regla, se realiza el cálculo del error mediante el método bootstrap para la muestra de entrenamiento. Recordamos el proceso: sean $X_{e11}, \dots, X_{e1n_1}$ y $X_{e21}, \dots, X_{e2n_2}$ las muestras de entrenamiento tomadas en un escenario para las poblaciones 1 y 2 respectivamente:
   i. Extraemos una muestra bootstrap en cada población de la forma: $X_{11}^*, \dots, X_{1n_1}^*$ $iid\ de\ la\ distribución\ equiprobable\ sobre\ X_{e11}, \dots, X_{e1n_1}$ $X_{21}^*, \dots, X_{2n_2}^*$ $iid\ de\ la\ distribución\ equiprobable\ sobre\ X_{e21}, \dots, X_{e2n_2}$
   ii. Basada en la muestra bootstrap se obtienen las tres reglas discriminantes: LDA, QDA y la Regla Núcleo.
   iii. Clasifico las observaciones de la muestra de entrenamiento, que no forman parte de la muestra bootstrap, con cada regla.
   iv. Calculo la proporción de individuos mal clasificados, para cada regla.
   v. Repito el proceso un número elevado de veces, N=200 en R^2 y N=100 en R^3.

Calculo la media de las N proporciones de mal clasificados, correspondientes a cada regla.

Nota: Hay que tener en cuenta que las muestras de entrenamiento deben ser las mismas en cada escenario, tanto para el cálculo del error "real" como para el cálculo de los errores estimados.

Estos errores estimados los comparamos con el error real.

Descripción de escenarios:

Para el desarrollo de la parte práctica se ha trabajado con la simulación de una serie de escenarios que permitirá analizar las diferencias entre errores según los distintos escenarios.

En primer lugar se tiene el caso de dos poblaciones, para dos y para tres dimensiones:

- En el caso bidimensional se trabaja con dos escenarios, el escenario 1 se ha elegido de manera que los datos provengan de poblaciones más confundidas mientras que el escenario 2 contiene datos de poblaciones que están más separadas. A su vez, cada escenario consta de 3 apartados, el apartado 1 contiene datos de poblaciones normales con la misma matriz de varianzas, el apartado 2 contiene datos de poblaciones normales con distinta matriz de varianzas, y, por último, el apartado 3 proviene de una mixtura de normales. La notación utilizada es $E_{i,j}$ con i=1,2 el número del escenario y j=1,2,3 el número del apartado.
- En el caso tridimensional se trabaja con un único escenario que consta de tres apartados de manera análoga al tema bidimensional. Se denota por $S_{1,j}$, con j=1,2,3 el número del apartado.

Para finalizar se tiene el caso de tres poblaciones realizado para dos dimensiones. No se ha realizado la simulación para más dimensiones debido a su complejidad computacional. En este caso también se trabaja con un único escenario que consta de tres apartados de manera análoga al caso bidimensional. Se denota por $M_{1,j}$, con j=1,2,3 el número del apartado.

Por tanto, se consta de 6 escenarios distintos $E_{i,j}$ para dos poblaciones en dos dimensiones, 3 escenarios distintos $S_{1,j}$ para dos poblaciones en tres dimensiones y otros 3 escenarios distintos $M_{1,j}$ para tres poblaciones en dos dimensiones. Con i=1,2 y j=1,2,3.

Escenarios a tener en cuenta.

Distribuciones normales con la misma matriz de covarianzas

$$X|Y=i \sim N_p(\mu_i, \Sigma), \qquad i=1,2$$

$$Escenario\ E11: \quad \mu_1=\begin{pmatrix}0\\0\end{pmatrix}, \qquad \mu_2=\begin{pmatrix}0.5\\0.5\end{pmatrix}, \qquad \Sigma=\mathrm{I}$$

$$Escenario\ E12: \quad \mu_1=\begin{pmatrix}0\\0\end{pmatrix}, \qquad \mu_2=\begin{pmatrix}0.5\\0.5\end{pmatrix}, \quad \Sigma_1=\mathrm{I}\ , \Sigma_2=\begin{pmatrix}1 & 0.3\\0.3 & 1\end{pmatrix}$$

$$Escenario\ S11: \quad \mu_1=\begin{pmatrix}0\\0\\0\end{pmatrix}, \qquad \mu_2=\begin{pmatrix}0.5\\0.25\\0.4\end{pmatrix}, \qquad \Sigma=\mathrm{I}$$

$$X|Y=i \sim N_p(\mu_i, \Sigma), \qquad i=1,2,3$$

$$Escenario\ M11: \quad \mu_1=\begin{pmatrix}0\\0\end{pmatrix}, \qquad \mu_2=\begin{pmatrix}1\\1\end{pmatrix}, \quad \mu_3=\begin{pmatrix}1.4\\1.2\end{pmatrix} \Sigma=\mathrm{I}$$

Distribuciones normales con distinta matriz de covarianzas

$$X|Y=i \sim N_p(\mu_i, \Sigma), \qquad i=1,2$$

$$Escenario\ E21: \quad \mu_1=\begin{pmatrix}0\\0\end{pmatrix}, \qquad \mu_2=\begin{pmatrix}1\\1\end{pmatrix}, \qquad \Sigma=\mathrm{I}$$

$$Escenario\ E22: \quad \mu_1=\begin{pmatrix}0\\0\end{pmatrix}, \qquad \mu_2=\begin{pmatrix}1\\1\end{pmatrix}, \quad \Sigma_1=\mathrm{I}\ , \Sigma_2=\begin{pmatrix}1 & 0.7\\0.7 & 1\end{pmatrix}$$

$$Escenario\ S12: \quad \mu_1=\begin{pmatrix}0\\0\\0\end{pmatrix}, \qquad \mu_2=\begin{pmatrix}0.5\\0.25\\0.4\end{pmatrix}, \quad \Sigma_1=\mathrm{I}\ , \Sigma_2=\begin{pmatrix}1 & 0.3 & 0.7\\0.3 & 1 & 0.4\\0.7 & 0.4 & 1\end{pmatrix}$$

$$X|Y=i \sim N_p(\mu_i, \Sigma_i), \qquad i=1,2,3$$

$$Escenario\ M12: \quad \mu_1=\begin{pmatrix}0\\0\end{pmatrix}, \qquad \mu_2=\begin{pmatrix}1\\1\end{pmatrix}, \qquad \mu_3=\begin{pmatrix}1.4\\1.2\end{pmatrix},$$
$$\Sigma_1=\mathrm{I}\ , \Sigma_2=\Sigma_3=\begin{pmatrix}1 & 0.7\\0.7 & 1\end{pmatrix}$$

Distribuciones no normales . Mixtura de dos normales

$$X|Y=i \sim pN_p(\mu_{1i},\Sigma_{1i}) + (1-p)N_p(\mu_{2i},\Sigma_{2i}), \quad \text{i=1,2}$$

$$Escenario\ E13: \quad \mu_{11}=\begin{pmatrix}0\\0\end{pmatrix} \quad \mu_{12}=\begin{pmatrix}1\\1.5\end{pmatrix}, \quad \mu_{21}=\begin{pmatrix}-1\\-1\end{pmatrix} \qquad \mu_{22}\begin{pmatrix}1\\1\end{pmatrix},$$

$$\Sigma_1 = \mathrm{I}\,, \quad \Sigma_2=\begin{pmatrix}1 & 0.3\\0.3 & 1\end{pmatrix}$$

$$Escenario\ E23: \quad \mu_{11}=\begin{pmatrix}-1.5\\1\end{pmatrix} \quad \mu_{12}=\begin{pmatrix}1\\-1\end{pmatrix}, \quad \mu_{21}=\begin{pmatrix}-1\\-2\end{pmatrix} \qquad \mu_{22}\begin{pmatrix}1\\1\end{pmatrix},$$

$$\Sigma_1 = \mathrm{I}\,, \quad \Sigma_2=\begin{pmatrix}1 & 0.7\\0.7 & 1\end{pmatrix}$$

$$Escenario\ S13: \quad \mu_{11}=\begin{pmatrix}0\\0\\0\end{pmatrix} \quad \mu_{12}=\begin{pmatrix}0.5\\0.25\\0.4\end{pmatrix}, \quad \mu_{21}=\begin{pmatrix}0.2\\0.3\\0.4\end{pmatrix} \quad \mu_{22}=\begin{pmatrix}0.5\\0.25\\0.4\end{pmatrix},$$

$$\Sigma_1 = \mathrm{I} \quad \Sigma_2=\begin{pmatrix}1 & 0.3 & 0.7\\0.3 & 1 & 0.4\\0.7 & 0.4 & 1\end{pmatrix}$$

$$X|Y=i \sim pN_p(\mu_{1i},\Sigma_{1i}) + (1-p)N_p(\mu_{2i},\Sigma_{2i}), \quad \text{i=1,2,3}$$

$$Escenario\ M13: \quad \mu_{11}=\begin{pmatrix}1.4\\1.2\end{pmatrix} \quad \mu_{12}=\begin{pmatrix}-1\\-1\end{pmatrix}, \quad \mu_{21}=\begin{pmatrix}0\\0\end{pmatrix} \qquad \mu_{22}\begin{pmatrix}1\\1.5\end{pmatrix},$$

$$\mu_{31}=\begin{pmatrix}-1\\0\end{pmatrix} \quad \mu_{32}=\begin{pmatrix}0\\-2\end{pmatrix}$$

$$\Sigma_1=\begin{pmatrix}1 & 0.7\\0.7 & 1\end{pmatrix}\,, \quad \Sigma_2=\mathrm{I}$$

En todos los casos p=0.5

Representaciones gráficas y tablas.

Las representaciones se realizan para los casos de poblaciones bidimensionales y para muestras de tamaño pequeño para una mejor visualización de las gráficas.

Se representa para cada uno de los seis escenarios $E_{i,j}$ la muestra de entrenamiento junto con el estimador núcleo de la densidad. Además, se realiza la representación de las tres reglas discriminantes y la regla de Bayes, todas ellas para un tamaño muestral de n=20, ya que para valores más elevados se hace muy difícil la visualización debido a la cantidad de datos.

Para cada escenario se realiza la tabla con los errores obtenidos mediante la simulación, esta tabla contiene, para los tamaños muestrales n=20, n=200 y n=500 los errores cometidos por cada una de las tres reglas discriminantes en cada uno de los escenarios. Los errores que aparecen por filas son: el error real condicionado a la muestra de entrenamiento junto con los errores estimados (el error aparente, el de validación cruzada, el error bootstrap y el error bootstrap 632) y el error no condicionado a la muestra de entrenamiento junto con el error de Bayes (óptimo).

Denotamos en la tabla las tres reglas discriminantes como la regla lineal (Regal 1), cuadrática (Regla 2) y la regla núcleo (Regla 3).

De manera análoga se realiza la tabla para el caso de tres poblaciones en dos poblaciones.

Por último, en el caso de dos poblaciones en tres dimensiones, se ha realizado la misma tabla pero, debido a la enorme carga computacional, sólo se ha realizado para un tamaño muestral de n=20.

Resultados esperados.

Mediante las gráficas y las tablas obtenidas de los escenarios y sus correspondientes apartados se podrá observar las diferencias en los errores cometidos entre poblaciones confundidas y separadas, entre poblaciones normales para matrices de varianzas iguales y distintas y entre poblaciones normales y no normales. Además, se determinará qué regla clasifica con menor error en cada caso.

Los errores que se esperan para poblaciones confundidas deben ser mayores que los cometidos para datos provenientes de poblaciones más separadas. Esto es debido a que las reglas de clasificación podrán diferenciar mejor los elementos de distintas poblaciones cuanto más separadas se encuentren.

Para distintos tamaños muestrales, a medida que se incrementa el tamaño de la muestra el error real condicional debe decrecer hasta estabilizarse en el error asintótico (el error que se cometería para una muestra de infinitos individuos), este error va a ser ligeramente distinto para distintos tamaños muestrales ya que el error se genera partiendo de una muestra del tamaño correspondiente. Para un tamaño muestral de 500 los errores reales

se estabilizan variando únicamente al aumentar de 500 a 1000 en un error del orden de 10^{-3}, por esta razón se han simulado errores hasta un tamaño de n=500.

Los resultados esperados según los distintos escenarios son:

- Densidades normales
  - Con matriz de covarianzas iguales (escenarios $E_{i,1}$, $S_{1,1}$ y $M_{1,1}$) la regla que menor error comete en la clasificación es la regla discriminante lineal.
  - Con matriz de covarianzas distintas (escenarios $E_{i,2}$, $S_{1,2}$ y $M_{1,2}$) la regla que menor error comete es la regla discriminante cuadrática.
- Densidades de las mixturas (escenarios $E_{i,3}$, $S_{1,3}$ y $M_{1,3}$), debe clasificar con menor error la regla núcleo.

Los resultados esperados de los estimadores del error real son:

- El error aparente subestima al error real, esto es debido al gran sesgo que tiene.
- El error de validación cruzada dependiendo de la muestra y del escenario puede subestimar o sobreestimarlo, acercándose más al error real que el aparente.
- Los errores bootstrap y bootstrap 632 subestiman el error real pero en menor medida que el aparente ya que se elimina el sesgo.

Los resultados esperados para los errores no condicionados son que el error de Bayes (al ser el error óptimo) es menor que el error real no condicional, acercándose a él cuanto más grande es el tamaño de la muestra.

Funciones utilizadas para la simulación:

Funciones de R utilizadas.

Las funciones de R utilizadas para la simulación de los datos son las siguientes:

- *rmvnorm:* función de R que genera datos aleatorios de una normal multivariante. Requiere de la librería *mvtnorm.*
- *dmvnorm:* función de R que genera la densidad de datos para la normal multivariante. Requiere de la librería *mvtnorm.*
- *lda:* función de R que realiza un análisis discriminante lineal. Requiere de la librería *MASS.*
- *qda:* función de R que realiza un análisis discriminante cuadrático. Requiere de la librería *MASS.*
- *sm.density:* estimación de la densidad no paramétrica. Requiere de la librería *sm*.
- *contour:* función de R que crea un gráfico de contornos o añade líneas de contorno a gráficos ya existentes. Requiere de la librería *graphics.*

➢ Parámetros comunes a algunas funciones:

- **N**: tamaño de la muestra test.
- **n**: tamaño de la muestra de entrenamiento.
- **mix**: lista con los valores para la creación de la mixtura:
  - $medias: medias.
  - $vars: matrices de las varianzas.
  - $p: primera proporción de la mixtura.

  Si no es el escenario de la mixtura el valor de mix y por defecto es NULL.

- **m**: matriz de medias teóricas. Tamaño pxk, siendo p la dimensión y k el número de poblaciones.
- **v**: lista con las matrices de varianzas de cada población.
- **mue**: muestra de entrenamiento. Salida de la función *m.entr* más adelante explicada.
- **t**: muestra test. Salida de la función mue.test, más adelante descrita.
- **dat**: lista con los datos de las muestras de entrenamiento de cada población.
- **x**: lista con las muestras de entrenamiento de las poblaciones, ordenadas.

Funciones creadas.

- **Generación de muestras:**
  - **m.entr**: función que genera la muestra de entrenamiento.
    - Parámetros: m, n, v, pooled.
      - pooled: parámetro lógico que indica si debe obtenerse la matriz pooled o no para la muestra de entrenamiento.
    - Salida: lista con los siguientes elementos:
      - dat: datos de la muestra.
      - medias: vector de medias muestral.
      - Sigma: matriz de varianzas muestral.
      - Spooled: matriz de varianzas pooled, si pooled=TRUE.

    La simulación de los datos de la muestra de entrenamiento se realiza mediante la función *rmvnorm* de R, descrita anteriormente.

  - **mixt:** función que genera los datos de una mixtura de normales.
    - Parámetros: n, p, m1, m2, S1,S2.
      - p: proporción de la primera parte de la mixtura.
      - m1, m2: medias de la mixtura, en orden.
      - S1, S2: matrices de varianzas de la mixtura, en orden.
    - Salida: vector de datos de la mixtura.

    La simulación de los datos para la creación de la mixtura se realiza mediante la función *rmvnorm* de R.

  - **mue.test:** función que genera los datos de la muestra test.
    - Parámetros: N, m, v, mix.
    - Salida: lista con los datos de las muestras test de cada población.

    La simulación para las normales se realiza con la función *rmvnorm* de R, y para el caso de la mixtura mediante la función *mixt* anteriormente explicada.

- **P-dimensiones y k poblaciones:**

  - **err.lqdak:** función que calcula los errores mediante las reglas discriminantes lineal y cuadrática.
    - Parámetros: n, N, m, v, mix, mue, t.
    - Salida: Una lista con las tablas de clasificación de las predicciones así como los errores de la regla discriminante lineal y la cuadrática.

  Los errores de las reglas discriminantes lineal y cuadrática se calculan mediante las funciones *lda y qda* de R.

  - **Bayesk:** función que calcula el error mediante la regla de Bayes.
    - Parámetros: N, m, v, mix, t.
    - Salida: valor numérico con el error de Bayes.
  - **Asint:** Función que calcula el error asintótico, para una muestra de infinitos individuos.
    - Parámetros: N, m, v, mix, t.
    - Salida: vector con los errores asintóticos calculados con las tres reglas.

  Se utiliza la función *dmvnorm* de R para el caso del cálculo del error de la regla núcleo.

- **2 dimensiones y k poblaciones:**
  - **r.nucleok:** función que calcula el error de la regla núcleo.
    - Parámetros: dat, N, m, v, mix, t.
    - Salida: valor numérico con el error de la regla núcleo.

  El error de la regla núcleo se obtiene con el uso de la función *sm.density* de R.

  - **fisher:** función que calcula las estimaciones mediante la regla discriminante de Fisher.
  - **cuad:** función que calcula las estimaciones mediante la regla discriminante cuadrática.
  - **BayesG:** función que calcula los valores estimados mediante la regla de Bayes.

  Estas tres últimas funciones permiten obtener los valores necesarios para la realización de los gráficos de las reglas discriminantes y la de Bayes. La función *BayesG* utiliza la función *dmvnorm* de R para la obtención de las densidades.

  - **grafs:** función que realiza los gráficos de las reglas discriminantes y de la regla de Bayes.
    - Parámetros: dat, mue, m, v, mix, e.
      - e: número del escenario.
    - Salidas: Gráficos con las reglas discriminantes lineal, cuadrática, núcleo y la regla de Bayes.

Esta función utiliza la función de R *contour* y *sm.density,* así como las funciones arriba explicadas *BayesG, fisher y cuad.*

- **err.aparente2k**: función que calcula el error aparente.
  - Parámetros: x.
  - Salida: vector con los errores aparentes de las tres reglas.
- **err.cross2k:** función que calcula el error mediante validación cruzada.
  - Parámetros: x.
  - Salida: vector con los errores correspondientes de las tres reglas.
- **err.boot2k:** función que calcula el error mediante bootstrap.
  - Parámetros**:** x, M.
    - M**:** número de repeticiones bootstrap.
  - Salida**:** vector con los errores bootstrap de las tres reglas.
- **errores.nocondk:** función que calcula las tablas con todos los errores y los gráficos explicados en el apartado *Representaciones gráficas y tablas,* para el caso de 2-dimensiones con k poblaciones.
  Para ello utiliza las funciones *m.entr, mue.test, mixt, err.lqdak, r.nucleok, err.aparente2k, Bayesk, err.cross2k, err.boot2k, grafs y Asint.*
  - Parámetros: NC, m, v, mmix, vmix, N, nb, ni, a, b, e.
    - NC: Indica si se realizan los errores condicionados (NC=1), o no condicionados siendo el valor el número de veces que se repite para la creación de los errores no condicionados.
    - N: tamaño de la muestra test, por defecto N=2000.
    - nb: número de repeticiones para el bootstrap. Por defecto nb=200.
    - ni: indica el número de elementos que se quieren para la muestra de entrenamiento. Por defecto, ni=20.
    - a, b: valores de los ejes de los gráficos.
    - e: número del escenario.

  Saca por pantalla los gráficos de la estimación de la densidad núcleo con la representación de la muestra de entrenamiento, junto con los gráficos correspondientes a las tres reglas discriminantes y la regla de Bayes. Además, muestra las tablas de errores explicadas en el apartado *Representaciones gráficas y tablas.*

- **P dimensiones y 2 poblaciones:**
  - **err.fn:** función que calcula los errores cometidos por la regla núcleo. Análoga a la función *r.nucleok*.
    - Parámetros: N, m, v, mix, x1, x2, t.
      - x1, x2: muestras de entrenamiento de las poblaciones.
  - **err.apatente:** función que calcula los errores aparentes. Análoga a la función *err.aparente2k.*
    - Parámetros: x.
    - Salida: vector con los errores aparentes de las tres reglas.
  - **err.cross:** función que calcula los errores mediante validación cruzada. Análoga a la función *err.cross2k*.
    - Parámetros: x.
    - Salida: vector con los errores correspondientes de las tres reglas.
  - **err.boot:** función que calcula los errores mediante bootstrap. Análoga a la función *err.boot2k.*
    - Parámetros: x, M.
      - M**:** número de repeticiones bootstrap.
    - Salida: vector con los errores bootstrap de las tres reglas.

  Estas últimas 4 funciones implementan otra función llamada *f.nucleo* que calcula la estimación de los datos mediante la función núcleo por definición.

  - **errores.nocond3:** función que calcula las tablas con todos los errores explicados en el apartado *Representaciones gráficas y tablas,* para el caso de p-dimensiones con 2 poblaciones.
    Para ello utiliza las funciones *m.entr, mue.test, mixt, err.lqdak, err.fn, err.aparente, Bayesk, err.cross, err.boot y Asint.*
    - Parámetros: NC, M1, M2, M5, ni.
      - NC: Indica si se realizan los errores condicionados (NC=1), o no condicionados siendo el valor el número de veces que se repite para la creación de los errores no condicionados.
      - M1, M2, M5: medias para las normales (M1 y M2) y para las mixturas (M1,M2 y M5,M2)
      - ni: indica el número de elementos que se quieren para la muestra de entrenamiento.
    - Salida: Saca por pantalla las tablas de errores explicadas en el apartado *Representaciones gráficas y tablas.*

Resultados.

Escenario E1

$$X|Y = i \sim N_p(\mu_i, \Sigma), \qquad i = 1,2$$

$$Escenario\ E11: \quad \mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mu_2 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, \qquad \Sigma = \mathrm{I}$$

$$X|Y = i \sim N_p(\mu_i, \Sigma_i), \qquad i = 1,2$$

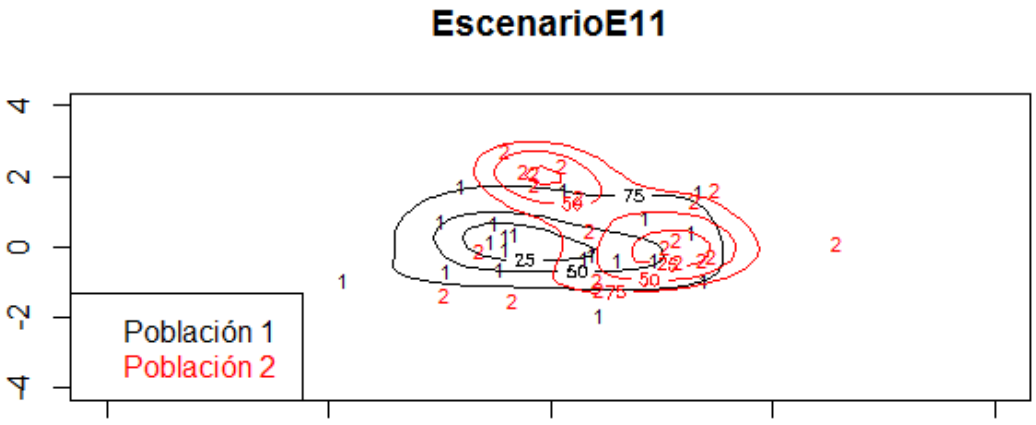
$$Escenario\ E12: \quad \mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mu_2 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, \quad \Sigma_1 = \mathrm{I}\ , \Sigma_2 = \begin{pmatrix} 1 & 0.3 \\ 0.3 & 1 \end{pmatrix}$$

$$X|Y = i \sim pN_p(\mu_{1i}, \Sigma_{1i}) + (1-p)N_p(\mu_{2i}, \Sigma_{2i}), \quad \text{i=1,2}$$

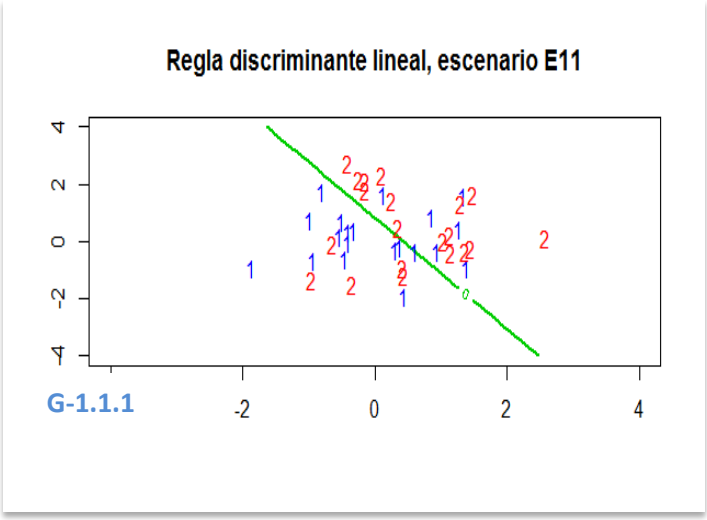
$$Escenario\ E13: \quad \mu_{11} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mu_{12} = \begin{pmatrix} 1 \\ 1.5 \end{pmatrix}, \quad \mu_{21} = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \qquad \mu_{22} \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

$$\Sigma_1 = \mathrm{I}, \quad \Sigma_2 = \begin{pmatrix} 1 & 0.3 \\ 0.3 & 1 \end{pmatrix}$$

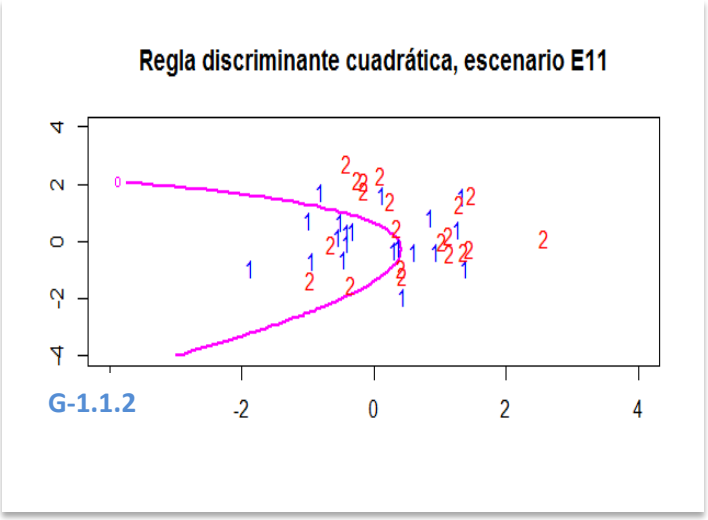
p=0.5



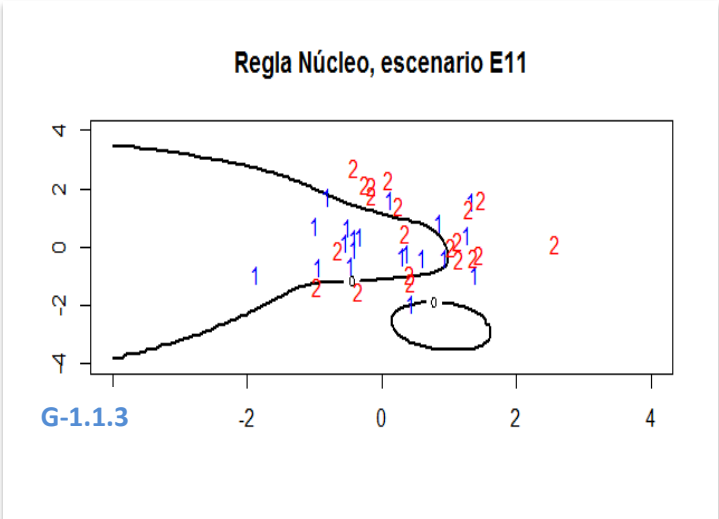
G-1.1. Densidades Escenario E 1.1



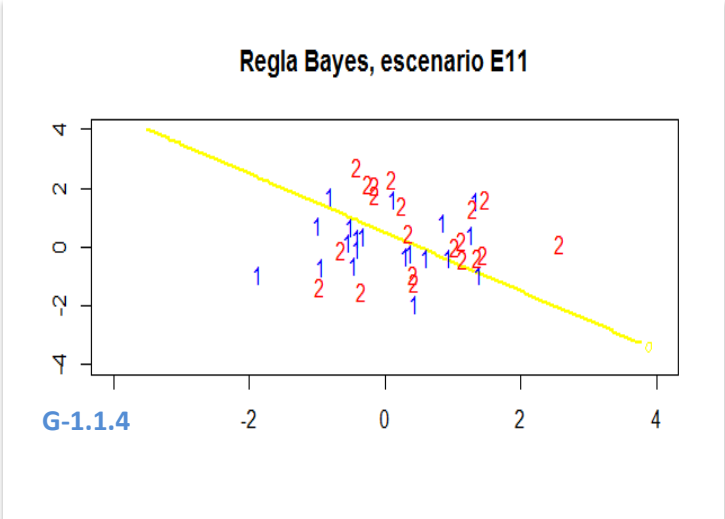
G-1.1.1



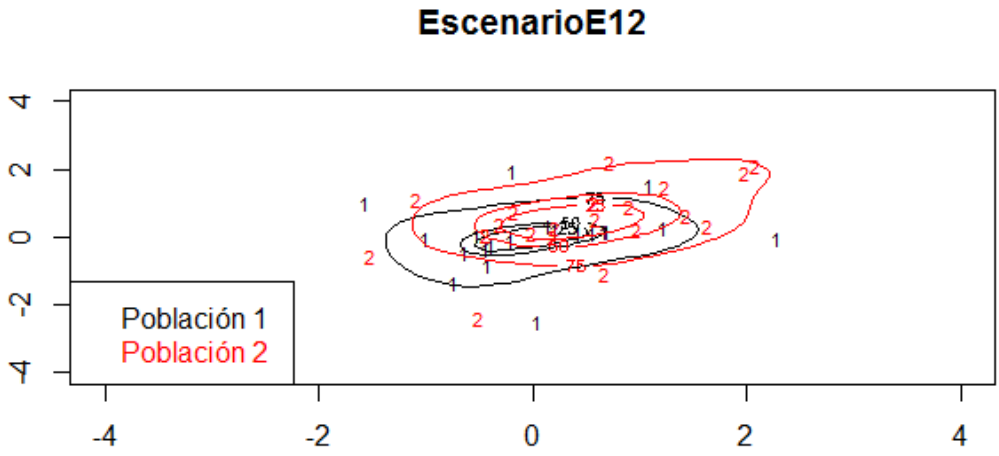
G-1.1.2



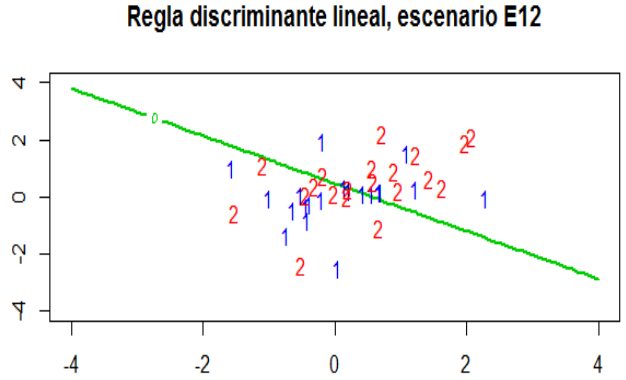
G-1.1.3



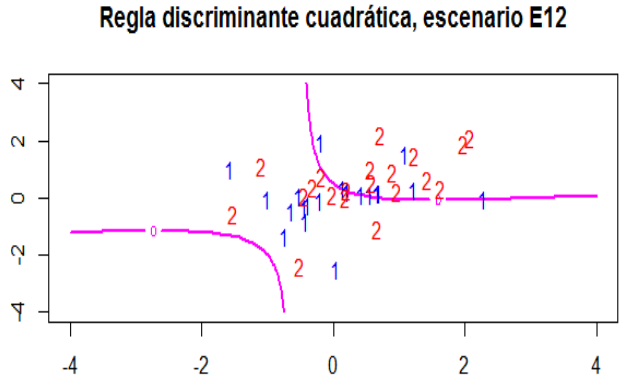
G-1.1.4



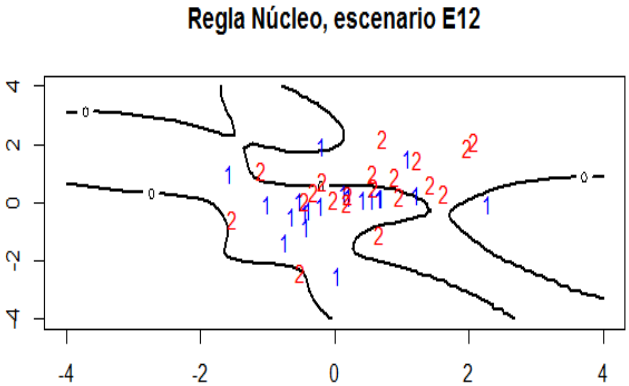
G-1.2. Densidades Escenario 1.2



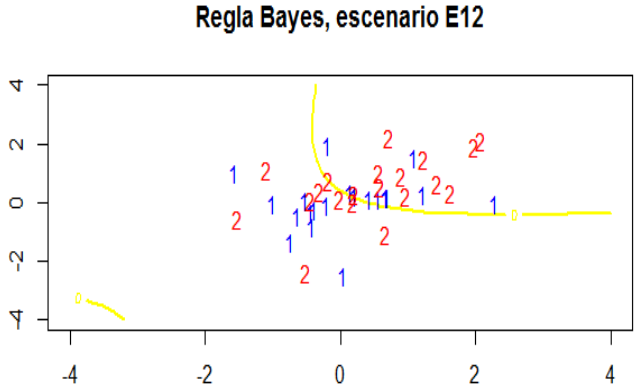
G-1.2.1



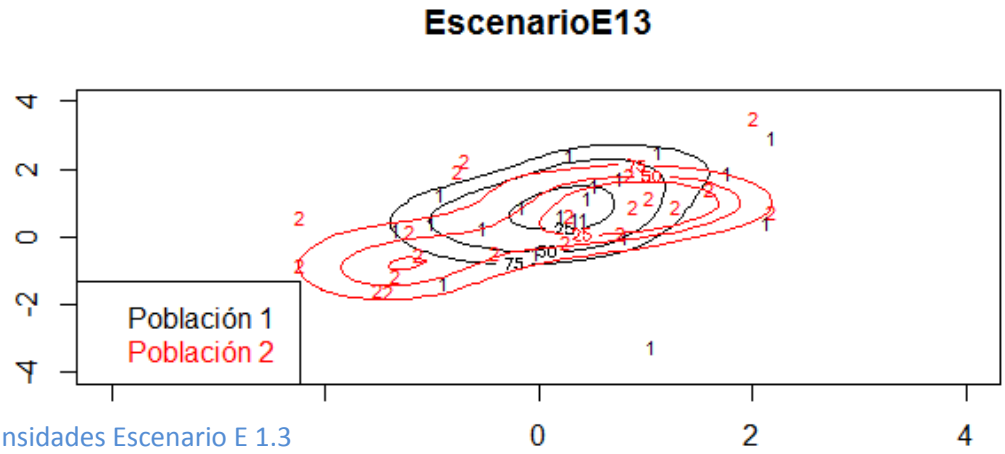
G-1.2.2



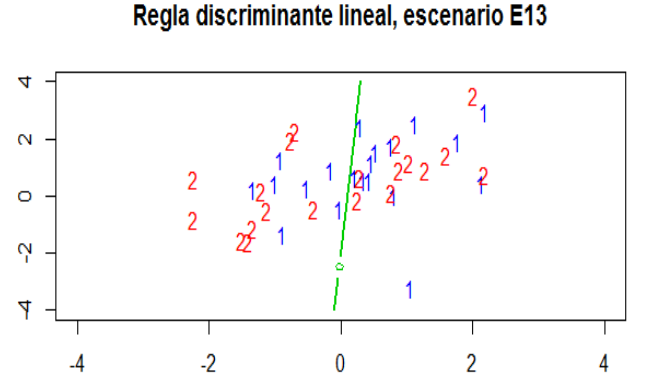
G-1.2. 3



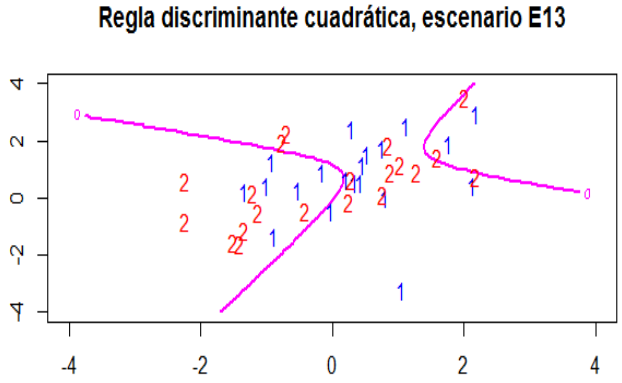
G-1.2. 4



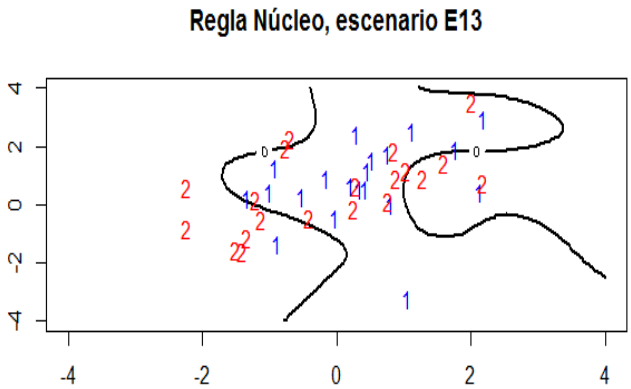
G-1.3. Densidades Escenario E 1.3



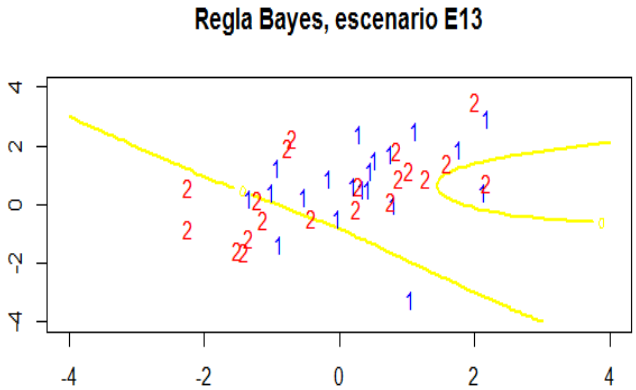
G-1.3. 1



G-1.3. 2



G-1.3.3



G-1.3.4

		ESCENARIO E1								
		Errores Condicionados a la muestra de entrenamiento								
		n=20			**n=200**			**n=500**		
		Regla 1	Regla 2	Regla 3	Regla 1	Regla 2	Regla 3	Regla 1	Regla 2	Regla 3
ESCENARIO E 1-1	Real Cond.	0.3632	0.3760	0.4235	0.3612	0.3838	0.3972	0.3675	0.3645	0.4122
	Aparente0	0.3000	0.4000	0.4250	0.4075	0.3800	0.4075	0.3650	0.3480	0.4180
	Cross Val.	0.3000	0.5250	0.5000	0.4075	0.3825	0.4125	0.3710	0.3510	0.4200
	Bootstrap	0.4035	0.4423	0.5482	0.4107	0.3855	0.4069	0.3689	0.3500	0.4179
	Boot632	0.3654	0.4267	0.5029	0.4096	0.3835	0.4071	0.3675	0.3493	0.4179
	Asintotico	0.3528	0.3728	0.4058	0.3645	0.3838	0.3990	0.3710	0.3612	0.4115
ESCENARIO E 1-2	Real Cond.	0.3855	0.3672	0.4345	0.3665	0.3708	0.3767	0.3720	0.3612	0.3868
	Aparente	0.3500	0.3750	0.5000	0.4125	0.3800	0.3700	0.3610	0.3400	0.3790
	Cross Val.	0.4000	0.5000	0.6000	0.4250	0.3925	0.3900	0.3620	0.3420	0.3820
	Bootstrap	0.4197	0.4978	0.5444	0.4191	0.3905	0.3912	0.3698	0.3413	0.3894
	Boot632	0.3941	0.4526	0.5281	0.4167	0.3866	0.3834	0.3665	0.3408	0.3856
	Asintotico	0.3528	0.3728	0.4058	0.3645	0.3838	0.3990	0.3710	0.3612	0.4115
ESCENARIO E 1-3	Real Cond.	0.4025	0.4242	0.3880	0.3748	0.3900	0.3735	0.3768	0.3610	0.3532
	Aparente	0.2250	0.2500	0.2750	0.3675	0.3550	0.3450	0.3520	0.3280	0.3380
	Cross Val.	0.4500	0.4250	0.4500	0.4250	0.4350	0.3750	0.3710	0.3430	0.3650
	Bootstrap	0.4432	0.4820	0.4998	0.4517	0.4183	0.4020	0.3806	0.3508	0.3676
	Boot632	0.3629	0.3966	0.4171	0.4207	0.3950	0.3810	0.3700	0.3424	0.3567
	Asintotico	0.3528	0.3652	0.3752	0.3645	0.3692	0.3545	0.3710	0.3578	0.3575
		Errores NO Condicionados a la muestra de entrenamiento								
		n=20			**n=200**			**n=500**		
E 1-1	Real NO Cond.	0.37990	0.39190	0.41184	0.36265	0.37068	0.40150	0.36284	0.37150	0.40145
	Bayes	0.36188								
E 1-2	Real NO Cond.	0.39902	0.40308	0.41103	0.36462	0.36355	0.38211	0.36382	0.36317	0.38246
	Bayes	**0.36212**								
E 1-3	Real NO Cond.	0.41577	0.41784	0.40616	0.37464	0.37191	0.37147	0.37021	0.370045	0.36836
	Bayes	0.36077								

T1. Tabla de errores. Escenario E1

En el escenario 1 se ve cómo las densidades de las poblaciones elegidas están muy confundidas, lo que conllevará errores más elevados. En los gráficos G1.i.1-G1.i.4 se representan las tres reglas discriminantes junto con la regla de Bayes (óptima) para cada sub-escenario. Se observa que la completa separación de las poblaciones es prácticamente imposible, lo que hace muy difícil de analizar las reglas gráficamente.

En la tabla T1 tenemos los errores para el escenario 1. En esta tabla vemos que, como se esperaba, para el escenario E 1-1 la regla que mejor clasifica es la Regla 1, para el escenario E 1-2 es la Regla 2, y para el escenario E 1-3 es la Regla 3. Se pueden ver pequeñas discrepancias, por ejemplo para n=200 para el escenario E 1-2 la regla que menor error tiene es la regla 1, esto puede ser debido a la tremenda aleatoriedad de los datos que generamos. Sin embargo, se comprueba que los errores al aumentar la muestra tienden a estabilizarse en torno al error asintótico.

Para un tamaño muestral aproximado de 500 los errores reales se estabilizan variando únicamente al aumentar de 500 a 1000 en un error del orden de 10^{-3}. Generalmente este error es menor a medida que aumenta el tamaño.

El error aparente generalmente subestima al error real, esto es debido al gran sesgo que tiene.

El error de validación cruzada dependiendo de la muestra y del escenario puede subestimar o sobreestimarlo, acercándose más al error real.

Por otro lado, los errores bootstrap y bootstrap 632 subestiman el error real pero en menor medida que el aparente ya que se elimina el sesgo mediante esta técnica.

En cuanto a los errores no condicionados, el error de Bayes (al ser el error óptimo) es generalmente menor que el error real No condicional, acercándose cada vez más cuanto más grande es el tamaño de la muestra. Vemos en la tabla T1 que esto sucede así, siendo en el escenario 1.3 donde más discrepancias podemos ver.

Escenario E2

$$X|Y = i \sim N_p(\mu_i, \Sigma), \qquad i = 1,2$$

Escenario E21: $\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mu_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \Sigma = \mathrm{I}$

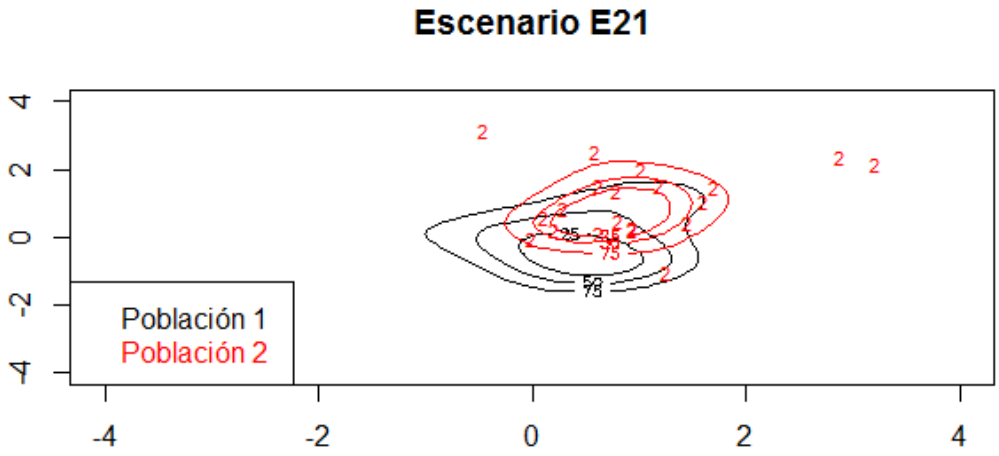
$$X|Y = i \sim N_p(\mu_i, \Sigma_i), \qquad i = 1,2$$

Escenario E22: $\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mu_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \Sigma_1 = \mathrm{I}\ , \Sigma_2 = \begin{pmatrix} 1 & 0.7 \\ 0.7 & 1 \end{pmatrix}$

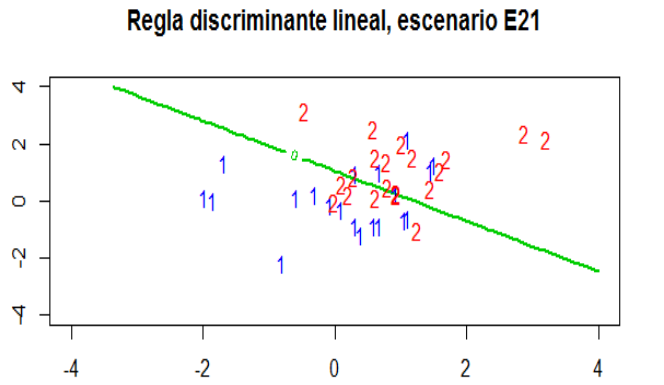
$$X|Y = i \sim pN_p(\mu_{1i}, \Sigma_{1i}) + (1-p)N_p(\mu_{2i}, \Sigma_{2i}), \quad \text{i=1,2}$$

Escenario E23: $\mu_{11} = \begin{pmatrix} -1.5 \\ 1 \end{pmatrix} \quad \mu_{12} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \mu_{21} = \begin{pmatrix} -1 \\ -2 \end{pmatrix} \qquad \mu_{22} \begin{pmatrix} 1 \\ 1 \end{pmatrix},$

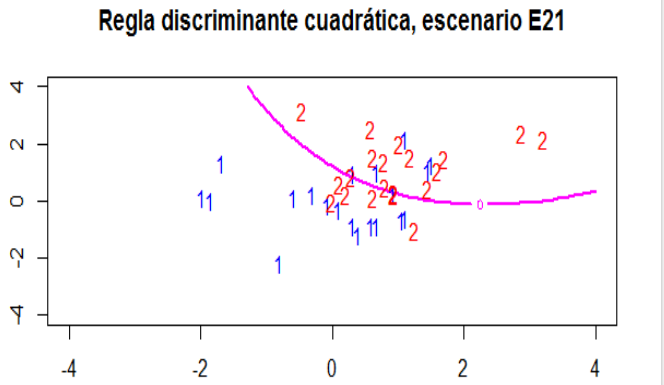
$$\Sigma_1 = \mathrm{I}\,, \quad \Sigma_2 = \begin{pmatrix} 1 & 0.7 \\ 0.7 & 1 \end{pmatrix}$$



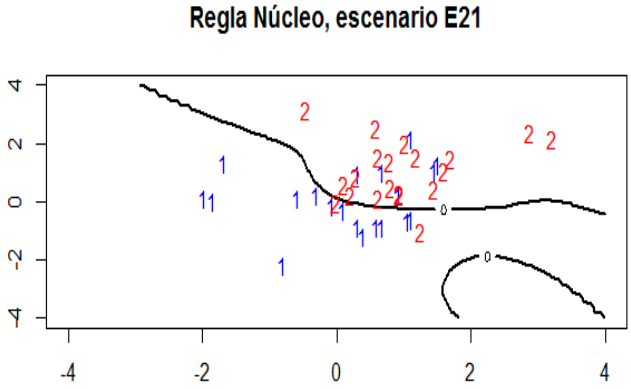
G-2.1. Densidades Escenario E 2.1



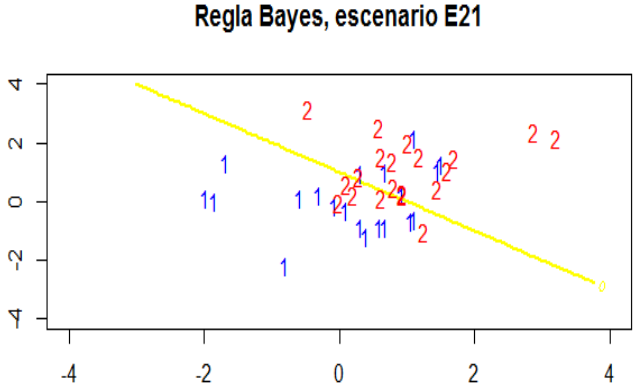
G-2.1.1



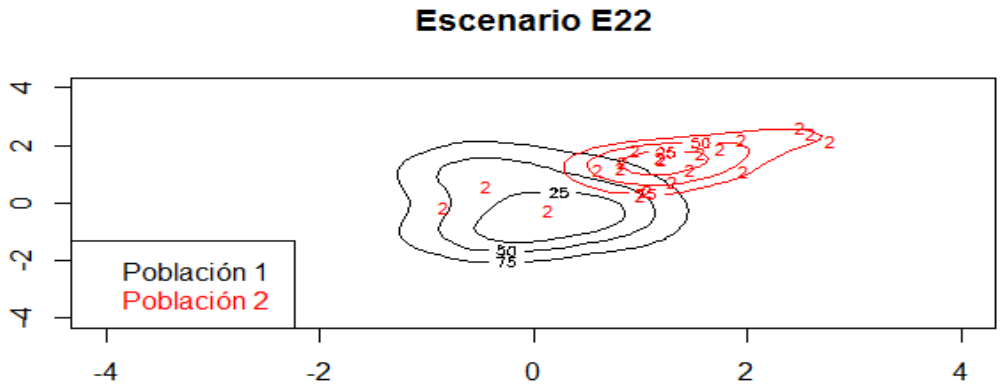
G-2.1.2



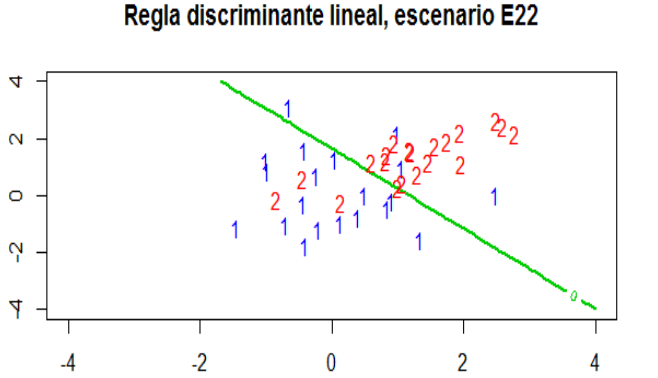
G-2.1.3



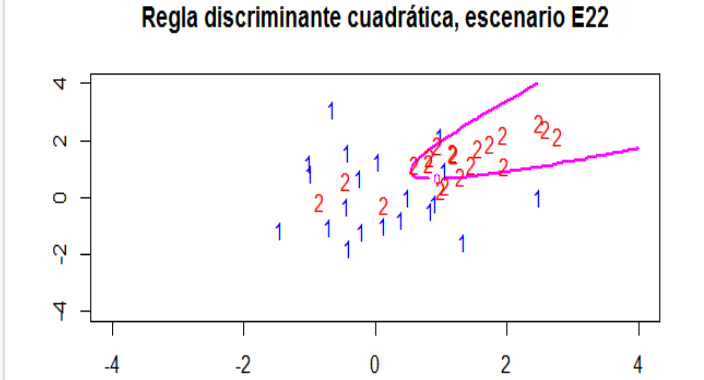
G-2.1.4



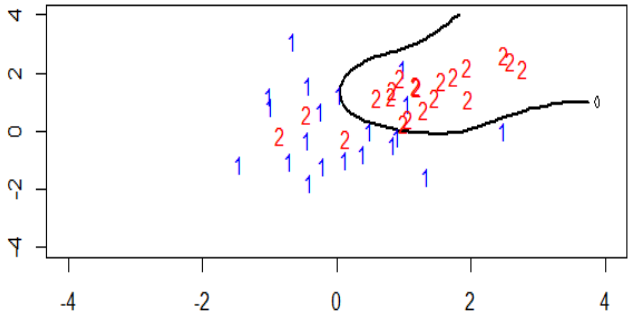
G-2.2. Densidades Escenario E 2.2



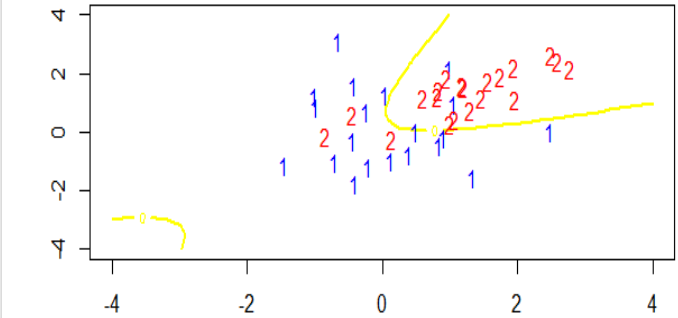
G-2.2.1



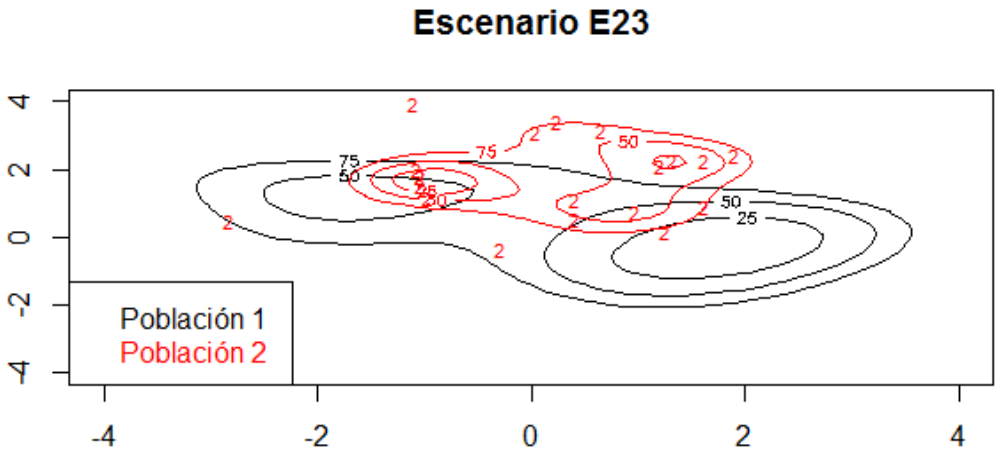
G-2.2.2



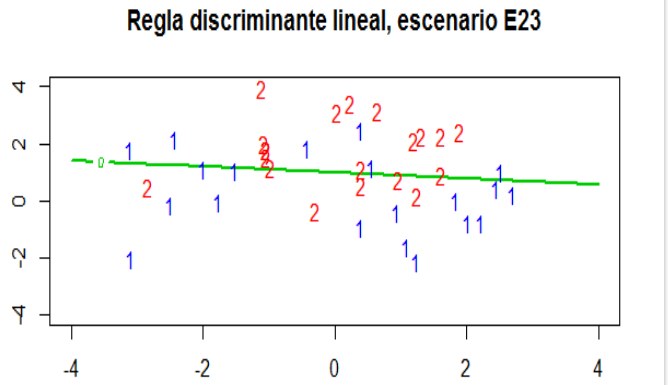
G-2.2.3



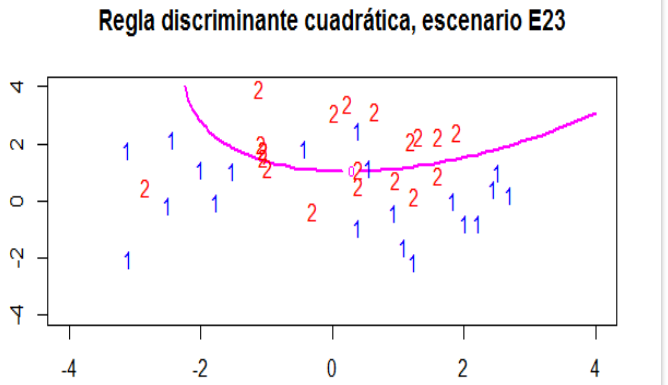
G-2.2.4



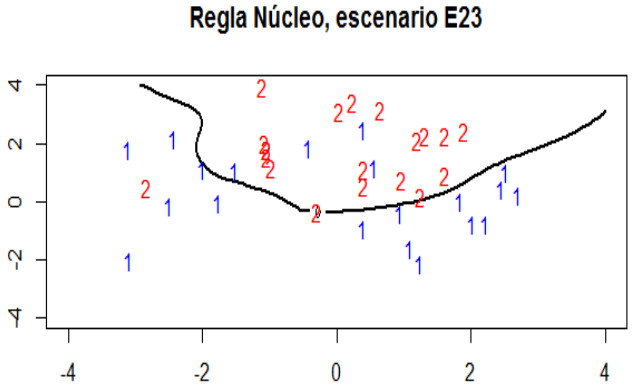
G-2.3. Densidades Escenario 2.3



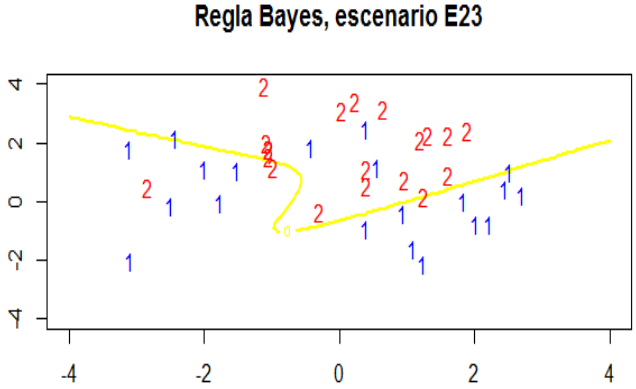
G-2.3.1



G-2.3.2



G-2.3.3



G-2.3.4

		ESCENARIO E2								
		Errores Condicionados a la muestra de entrenamiento								
		n=20			**n=200**			**n=500**		
		Regla 1	Regla 2	Regla 3	Regla 1	Regla 2	Regla 3	Regla 1	Regla 2	Regla 3
ESCENARIO E 2-1	Real Cond.	0.2490	0.2605	0.2790	0.2442	0.2812	0.2662	0.2468	0.2642	0.2700
	Aparente0	0.2750	0.1750	0.2750	0.2625	0.2800	0.2700	0.2360	0.2690	0.2650
	Cross Val.	0.3500	0.2000	0.3750	0.2675	0.2800	0.2825	0.2360	0.2700	0.2680
	Bootstrap	0.3223	0.2351	0.3296	0.2667	0.2763	0.2748	0.2371	0.2736	0.2721
	Boot632	0.3049	0.2130	0.3095	0.2652	0.2777	0.2731	0.2367	0.2719	0.2695
	Asintótico	0.2440	0.2612	0.3162	0.2412	0.2782	0.3240	0.2460	0.2630	0.3208
ESCENARIO E 2-2	Real Cond.	0.2580	0.2552	0.2552	0.2435	0.2375	0.2552	0.2460	0.2378	0.2537
	Aparente	0.2750	0.1500	0.1750	0.2650	0.2475	0.2775	0.2350	0.2430	0.2440
	Cross Val.	0.3250	0.1500	0.2500	0.2700	0.2525	0.2875	0.2370	0.2430	0.2460
	Bootstrap	0.3704	0.1725	0.2698	0.2696	0.2466	0.2815	0.2358	0.2427	0.2498
	Boot632	0.3353	0.1642	0.2349	0.2679	0.2469	0.2801	0.2355	0.2428	0.2477
	Asintótico	0.2440	0.2612	0.3162	0.2412	0.2782	0.3240	0.2460	0.2630	0.3208
ESCENARIO E 2-3	Real Cond.	0.2705	0.2428	0.2302	0.2465	0.2445	0.2112	0.2508	0.2390	0.2105
	Aparente	0.2000	0.1250	0.1250	0.2375	0.2300	0.2025	0.2350	0.2370	0.1950
	Cross Val.	0.2000	0.1750	0.2000	0.2650	0.2500	0.2175	0.2380	0.2470	0.2040
	Bootstrap	0.3443	0.2175	0.2631	0.2795	0.2474	0.2245	0.2431	0.2516	0.2089
	Boot632	0.2912	0.1834	0.2123	0.2641	0.2410	0.2164	0.2401	0.2462	0.2038
	Asintótico	0.2440	0.2390	0.2040	0.2412	0.2390	0.2017	0.2460	0.2368	0.2060
		Errores NO Condicionados a la muestra de entrenamiento								
		n=20			**n=200**			**n=500**		
E 2-1	Real NO Cond.	0.25364	0.28107	0.26939	0.24062	0.26903	0.26371	0.24126	0.26696	0.26208
	Bayes	0.24080								
E 2-2	Real NO Cond.	0.26522	0.25703	0.26861	0.24114	0.23874	0.25275	0.24127	0.23712	0.25028
	Bayes	0.23668								
E 2-3	Real NO Cond.	0.28028	0.28021	0.24845	0.24638	0.24445	0.21513	0.24383	0.24072	0.20855
	Bayes	0.20249								

T2. Tabla de errores. Escenario E2

En este escenario 2, se puede ver en las representaciones gráficas que las poblaciones están más separadas, y por esa razón, en general los errores en el escenario 1 son más elevados que en el escenario 2.

A pesar de ello, las poblaciones no son totalmente disjuntas, lo que sigue haciendo difícil la comparación de las reglas discriminantes gráficamente, representadas en los gráficos G2.i.1-G2.i.4 junto con la regla de Bayes (óptima) para cada apartado.

En la tabla T2 tenemos los errores para el escenario 2. En esta tabla vemos que, como se esperaba, para el escenario E 2-1 la regla que mejor clasifica es la Regla 1, para el escenario E 2-2 es la Regla 2, y para el escenario E 2-3 es la Regla 3. Se observa que los errores se van acercando cada vez más al error asintótico según aumenta el tamaño muestral.

Al igual que en la tabla T1 observamos que el error aparente subestima al error real debido al gran sesgo que tiene.

El error de validación cruzada dependiendo de la muestra y del escenario puede subestimar o sobreestimar el error real, acercándose más a él que el error aparente.

Por otro lado, los errores bootstrap en este caso vemos que generalmente sobreestiman el error real, y el error bootstrap 632 se hace más al error real debido a la corrección realizada.

En cuanto a los errores no condicionados, el error de Bayes (al ser el error óptimo) es generalmente menor que el error real No condicional, acercándose cada vez más cuanto más grande es el tamaño de la muestra. Vemos en la tabla T2 que esto sucede así, siendo en el escenario 1.3 donde más discrepancias podemos ver.

Nótese que para este escenario, los resultados esperados se cumplen con más precisión y en más casos que en el escenario 1, esto es debido a que las reglas discriminantes van a funcionar mejor en poblaciones separadas.

3 dimensiones

Escenario S1

$$X|Y = i \sim N_p(\mu_i, \Sigma), \qquad i = 1,2$$

$$Escenario\ S11: \quad \mu_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \mu_2 = \begin{pmatrix} 0.5 \\ 0.25 \\ 0.4 \end{pmatrix}, \qquad \Sigma = \mathrm{I}$$

$$X|Y = i \sim N_p(\mu_i, \Sigma_i), \qquad i = 1,2$$

$$Escenario\ S12: \quad \mu_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \mu_2 = \begin{pmatrix} 0.5 \\ 0.25 \\ 0.4 \end{pmatrix}, \quad \Sigma_1 = \mathrm{I}\ , \Sigma_2 = \begin{pmatrix} 1 & 0.3 & 0.7 \\ 0.3 & 1 & 0.4 \\ 0.7 & 0.4 & 1 \end{pmatrix}$$

$$X|Y = i \sim pN_p(\mu_{1i}, \Sigma_{1i}) + (1-p)N_p(\mu_{2i}, \Sigma_{2i}), \quad \text{i=1,2}$$

$$Escenario\ S13: \quad \mu_{11} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \mu_{12} = \begin{pmatrix} 0.5 \\ 0.25 \\ 0.4 \end{pmatrix}, \quad \mu_{21} = \begin{pmatrix} 0.2 \\ 0.3 \\ 0.4 \end{pmatrix} \quad \mu_{22} = \begin{pmatrix} 0.5 \\ 0.25 \\ 0.4 \end{pmatrix},$$

$$\Sigma_1 = \mathrm{I} \quad \Sigma_2 = \begin{pmatrix} 1 & 0.3 & 0.7 \\ 0.3 & 1 & 0.4 \\ 0.7 & 0.4 & 1 \end{pmatrix}$$

En los dos casos p=0.5

		3 DIMENSIONES		
		Errores Condicionados		
		n=20		
		Regla 1	Regla 2	Regla 3
ESCENARIO S1-1	Real Cond.	0.3935	0.5210	0.4545
	Aparente0	0.3000	0.4500	0.3250
	Cross Val.	0.3500	0.6500	0.6250
	Bootstrap	0.3395	0.5510	0.5350
	Boot632	0.3250	0.5138	0.4577
	Asintótico	0.3635	0.3695	0.4580
ESCENARIO S1-2	Real Cond.	0.4255	0.4735	0.4855
	Aparente	0.2000	0.1750	0.2500
	Cross Val.	0.4500	0.3500	0.5000
	Bootstrap	0.3934	0.3352	0.4773
	Boot632	0.3222	0.2763	0.3937
	Asintótico	0.3635	0.3930	0.4580
ESCENARIO S1-3	Real Cond.	0.4240	0.4745	0.4705
	Aparente	0.1000	0.1000	0.1000
	Cross Val.	0.4750	0.3000	0.5750
	Bootstrap	0.1455	0.1135	0.1806
	Boot632	0.1288	0.1085	0.1510
	Asintótico	0.3635	0.3175	0.4540
		Errores NO Condicionados		
		n=20		
ES1-1	Real NO Cond.	0.40385	0.41620	0.47440
	Bayes	0.36475		
ES1-2	Real NO Cond.	0.42507	0.35764	0.48334
	Bayes	0.31200		
ES1-3	Real NO Cond.	0.43842	0.39581	0.48358
	Bayes	0.44647		

T3. Tabla de errores. Escenario S1

En este apartado se analizan los resultados obtenidos para el caso tridimensional de dos poblaciones. La simulación se ha realizado, en este caso, únicamente para un tamaño muestral de 20 individuos ya que la carga computacional de este tipo de simulaciones es de cerca del 300% en comparación con el caso bidimensional.

Las representaciones gráficas de las reglas discriminantes en este caso no son factibles debido a la dificultad de representación de las reglas en 3D. Se puede ver cómo los resultados son análogos a los obtenidos en el caso bidimensional, aunque obtenemos más discrepancias con los resultados esperados debido a la elección de las poblaciones y a la complejidad de los datos.

En la tabla T3 tenemos los errores para el escenario S1. En esta tabla vemos que hay más discrepancias con los resultados esperados, para el escenario S 1-1 la regla que mejor clasifica es la Regla 1, sin embargo también lo es para los escenarios E 2-2 y E 2-3.

Debido a la carga computacional no se han realizado los errores para los tamaños muestrales n=200 y n=500, por tanto no es posible hacer la comparación a medida que crece el tamaño muestral. Sin embargo, comprobamos que para datos más complejos los resultados esperados no se cumplen con tanta precisión como para un menor número de poblaciones.

3 poblaciones:

Escenario M1

$$X|Y = i \sim N_p(\mu_i, \Sigma), \qquad i = 1,2,3$$

$$Escenario\ M11: \quad \mu_1 = \begin{pmatrix}0\\0\end{pmatrix}, \qquad \mu_2 = \begin{pmatrix}1\\1\end{pmatrix}, \quad \mu_3 = \begin{pmatrix}1.4\\1.2\end{pmatrix} \ \Sigma = \mathrm{I}$$

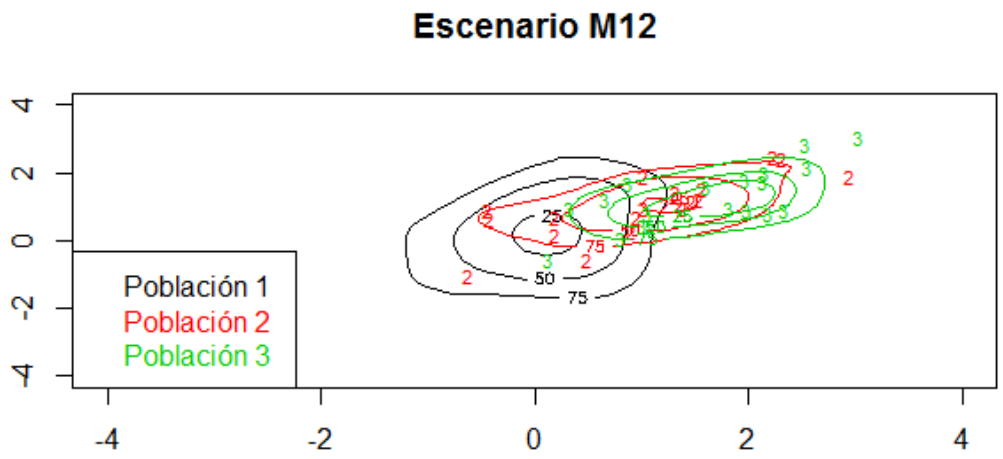
$$X|Y = i \sim N_p(\mu_i, \Sigma_i), \qquad i = 1,2,3$$

$$Escenario\ M12: \quad \mu_1 = \begin{pmatrix}0\\0\end{pmatrix}, \qquad \mu_2 = \begin{pmatrix}1\\1\end{pmatrix}, \qquad \mu_3 = \begin{pmatrix}1.4\\1.2\end{pmatrix},$$
$$\Sigma_1 = \mathrm{I} \ , \Sigma_2 = \Sigma_3 = \begin{pmatrix}1 & 0.7\\0.7 & 1\end{pmatrix}$$

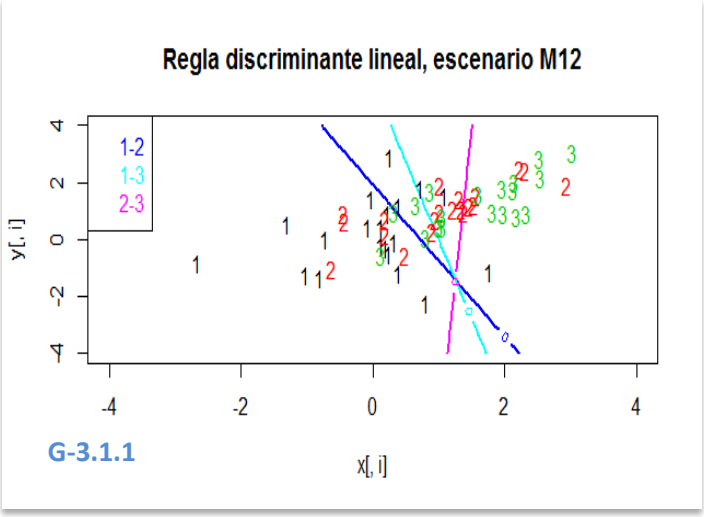
$$X|Y = i \sim pN_p(\mu_{1i}, \Sigma_{1i}) + (1-p)N_p(\mu_{2i}, \Sigma_{2i}), \quad \text{i=1,2,3}$$

$$Escenario\ M13: \quad \mu_{11} = \begin{pmatrix}1.4\\1.2\end{pmatrix} \quad \mu_{12} = \begin{pmatrix}-1\\-1\end{pmatrix}, \quad \mu_{21} = \begin{pmatrix}0\\0\end{pmatrix} \qquad \mu_{22} \begin{pmatrix}1\\1.5\end{pmatrix},$$
$$\mu_{31} = \begin{pmatrix}-1\\0\end{pmatrix} \quad \mu_{32} = \begin{pmatrix}0\\-2\end{pmatrix}$$
$$\Sigma_1 = \begin{pmatrix}1 & 0.7\\0.7 & 1\end{pmatrix} \ , \ \Sigma_2 = \mathrm{I}$$

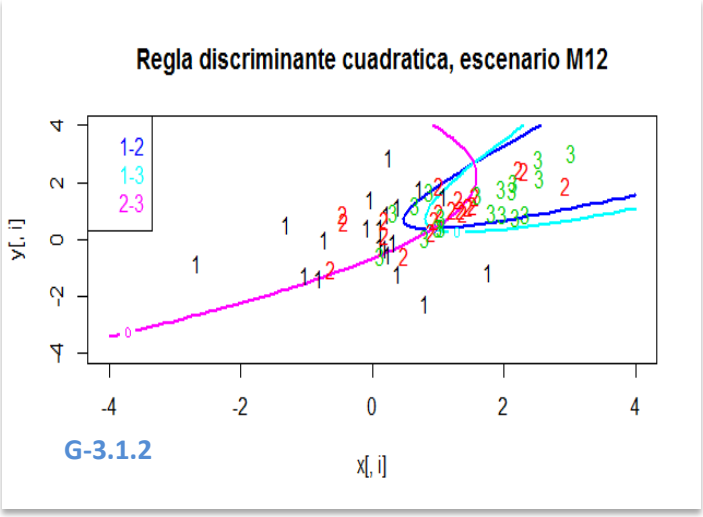
En los dos casos p=0.5



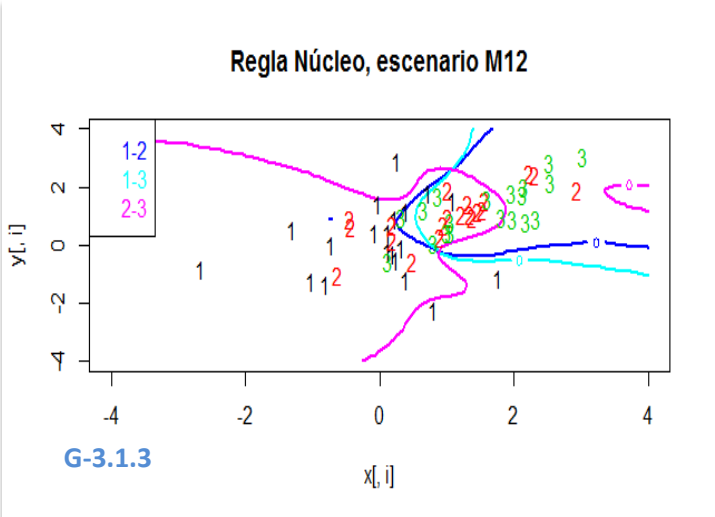
G-3.1. Densidades Escenario M12. 3 poblaciones



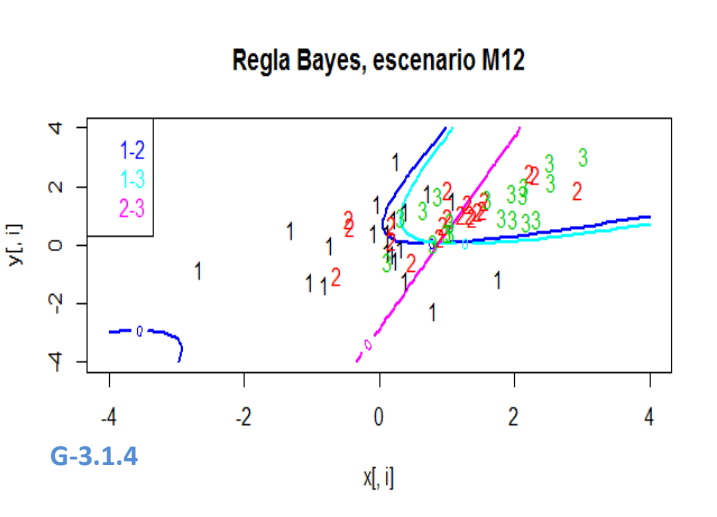
G-3.1.1



G-3.1.2

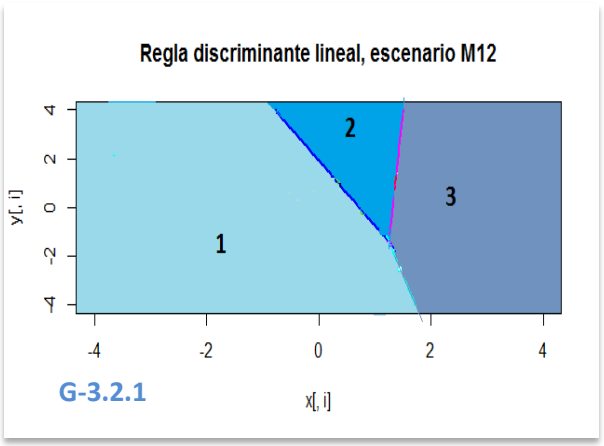


G-3.1.3

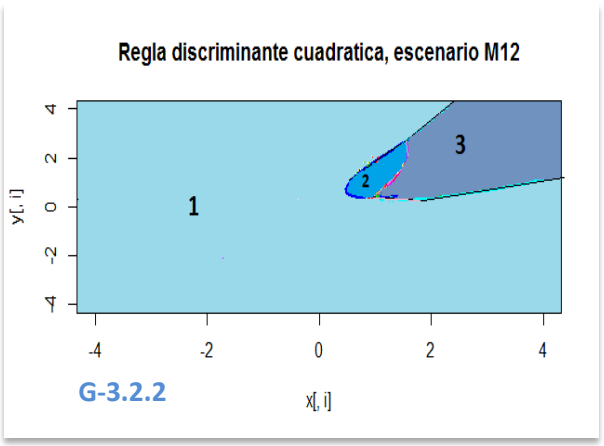


G-3.1.4

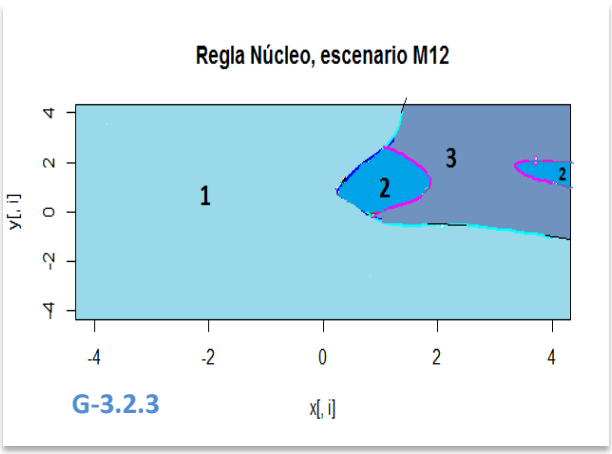
División del espacio según la población:



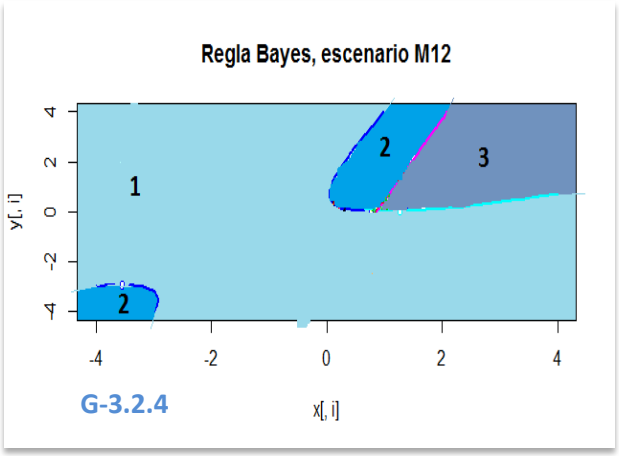
G-3.2.1



G-3.2.2



G-3.2.3



G-3.2.4

		3 poblaciones								
		Errores Condicionados a la muestra de entrenamiento								
		n=20			n=200			n=500		
		Regla 1	Regla 2	Regla 3	Regla 1	Regla 2	Regla 3	Regla 1	Regla 2	Regla 3
ESCENARIO M 1-1	Real Cond.	0.4672	0.4568	0.5182	0.4440	0.4632	0.5167	0.4463	0.4540	0.5227
	Aparente0	0.4167	0.4333	0.5167	0.4100	0.4850	0.5100	0.4393	0.4527	0.5140
	Cross Val.	0.5167	0.5000	0.5667	0.4200	0.4983	0.5150	0.4413	0.4580	0.5153
	Bootstrap	0.4999	0.4635	0.5562	0.4231	0.4787	0.5092	0.4409	0.4572	0.5166
	Boot632	0.4693	0.4524	0.5417	0.4183	0.4810	0.5095	0.4403	0.4556	0.5157
	Asintótico	0.4412	0.4523	0.5285	0.4383	0.4617	0.5177	0.4415	0.4505	0.5242
ESCENARIO M 1-2	Real Cond.	0.5050	0.4405	0.5137	0.4455	0.4507	0.4725	0.4470	0.4307	0.4770
	Aparente	0.4000	0.4000	0.4333	0.4083	0.4583	0.4750	0.4407	0.4287	0.4713
	Cross Val.	0.5333	0.4833	0.5500	0.4150	0.4717	0.4850	0.4440	0.4327	0.4767
	Bootstrap	0.4942	0.4674	0.5372	0.4300	0.4657	0.4789	0.4436	0.4307	0.4744
	Boot632	0.4595	0.4426	0.4990	0.4220	0.4630	0.4774	0.4425	0.4300	0.4733
	Asintótico	0.4412	0.4570	0.5285	0.4383	0.4667	0.5177	0.4415	0.4587	0.5242
ESCENARIO M 1-3	Real Cond.	0.4937	0.4510	0.4948	0.4548	0.4572	0.4605	0.4550	0.4348	0.4445
	Aparente	0.3500	0.3667	0.3167	0.3950	0.4250	0.4017	0.4153	0.4187	0.4213
	Cross Val.	0.5500	0.4000	0.5333	0.4417	0.4867	0.4350	0.4567	0.4420	0.4460
	Bootstrap	0.5493	0.4964	0.5515	0.4508	0.4889	0.4470	0.4530	0.4404	0.4533
	Boot632	0.4759	0.4487	0.4651	0.4303	0.4654	0.4303	0.4391	0.4324	0.4415
	Asintótico	0.4412	0.4287	0.4295	0.4383	0.4455	0.4262	0.4415	0.4290	0.4292
		Errores NO Condicionados a la muestra de entrenamiento								
		n=20			n=200			n=500		
E 1-1	Real NO Cond.	0.45434	0.48621	0.52563	0.43642	0.46286	0.51796	0.43475	0.46186	0.51685
	Bayes	0.43346								
E 1-2	Real NO Cond.	0.46686	0.46959	0.49696	0.43896	0.44239	0.47611	0.43551	0.44134	0.47621
	Bayes	0.43976								
E 1-3	Real NO Cond.	0.48109	0.49353	0.49051	0.44783	0.45157	0.45116	0.44138	0.44687	0.44521
	Bayes	0.43063								

T4. Tabla de errores. Escenario M1

Para el caso bidimensional de tres poblaciones, se ve que los errores son muy grandes debido a la cercanía de las poblaciones. A pesar de ello, los resultados generales son análogos a los obtenidos en el caso de dos poblaciones.

En la tabla T4 tenemos los resultados para el escenario M1. En esta tabla se cumple con los resultados esperados para el escenario M 1-1 la regla que mejor clasifica es la Regla 1, y para el M 1-2 la que mejor clasifica es la Regla 2, sin embargo, también lo es para el escenario M 1-3. A pesar de ello, se observa que el error cometido por la regla 3 en el escenario M1-3 disminuye considerablemente a medida que aumenta el tamaño muestral, y a mayor velocidad que el resto de reglas, por lo que podemos suponer que la regla que mejor clasifique en el escenario M 1-3 será la Regla 3 para un tamaño muestral más elevado.

Los gráficos G3.1.i muestran la representación de un único escenario (el apartado 2, de datos provenientes de poblaciones normales con distinta matriz de covarianzas) para mostrar la complejidad de representación de las reglas discriminantes ya que la división del espacio en k partes (3 en este caso) no es factible con las herramientas y el programa utilizado. Por tanto, la única manera que se ha encontrado es representar las reglas para cada par de poblaciones y posteriormente dividir el espacio con las coincidencias entre poblaciones. Se muestra en los gráficos G3.2.i la representación de cómo debería quedar dividido el espacio para este caso. Esta representación se ha realizado con el programa *paint* de Windows.

Como dificultad añadida está la carga computacional que conlleva el cálculo de los errores para tres poblaciones.

Conclusiones.

En general los resultados esperados se cumplen para las poblaciones elegidas. Para poblaciones normales con la misma matriz de covarianzas, la regla que clasifica con menor error es la regla lineal discriminante (Regla 1). Para poblaciones normales con distinta matriz de covarianzas, la regla que clasifica con menor error es la regla cuadrática (Regla 2). Y, para poblaciones no normales, en este caso mixtura de normales, la regla que menor error comete es la regla núcleo (Regla 3). En este proyecto se comprueba que en todos los casos realizados para un tamaño muestral de 500, los errores no condicionados cumplen con lo esperado. El único caso para el que no se cumple es el caso tridimensional, en el cual no se ha podido realizar para un tamaño mayor de 20, vemos que para poblaciones no normales, escenario S 1-3, el error de la regla cuadrática (Regla 2) es el menor.

En cuanto a los errores reales, a medida que aumenta el tamaño muestral éste decrece hasta estabilizarse en un valor que es el valor del error real para un tamaño muestral infinito.

El error aparente subestima el error real debido al sesgo. Se intenta corregir el sesgo de este error mediante el error de validación cruzada y el error bootstrap, ambos dos se aproximan mejor al error real, sobrestimándolo generalmente. El error corregido bootstrap632 combina el error aparente con el error bootstrap, por lo que por lo general es el que mejor aproxima al error real.

En cuanto a los errores reales no condicionados, estos decrecen a medida que aumenta el tamaño muestral hasta estabilizarse en un valor. Estos errores reales se comparan con los errores obtenidos mediante la regla de Bayes, que, al ser óptima, los errores de Bayes son menores que el error real no condicionado, siendo más próximos cuanto mayor sea el tamaño muestral.

El orden del error que cometen las reglas discriminantes es tanto mayor cuanto más confundidas estén las poblaciones. En el caso bidimensional con dos poblaciones, en el escenario 1, los resultados oscilan entre 0.34 y 0.42. Mientras que en el escenario 2, donde las poblaciones están más separadas, oscilan entre 0.21 y 0.28. Con tres poblaciones los errores oscilan entre 0.43 y 0.53. Y, en el caso tridimensional, oscilan entre 0.39 y 0.52.

Las diferencias dentro de cada apartado entre las reglas son, a lo sumo, de un 10% de diferencia. Por ejemplo, en el escenario E1, tomando como referencia un tamaño muestral de 500, los errores no condicionados para las reglas 1, 2 y 3 respectivamente han sido aproximadamente 0.36, 0.37, 0.40 para el apartado E 1-1; 0.364, 0.363, 0.38 para el apartado E 1-2; y 0.37, 0.37, 0.36 para el apartado E 1-3.

Mientras que en el escenario E2, dos poblaciones más separadas entre sí, los errores para las reglas 1,2 y 3 respectivamente han sido aproximadamente 0.24, 0.27, 0.26 para el apartado E 2-1; 0.24, 0.23, 0.25 para el apartado E 2-2; y 0.243, 0.24, 0.20 para el apartado E 2-3. Siendo estos errores alrededor de un 25% menores que el escenario E1.

Para el caso de tres poblaciones, los errores para las reglas 1,2 y 3 respectivamente han sido aproximadamente 0.43, 0.46, 0.51 para el escenario M 1- 1; 0.43, 0.44, 0.47 para el escenario M 1-2; y 0.441, 0.446, 0.445 para el escenario M 1-3. Estos errores se verían reducidos si las poblaciones estuviesen más separadas. Vemos que no se cumplen los resultados esperados ya que en los tres escenarios la regla que menor error comete es la 1. Esto puede ser debido a la poca separación de las poblaciones y a la complejidad al añadir una población más.

Por último para el caso de dos poblaciones de datos tridimensionales, tomando como referencia n=20, los errores no condicionados para las reglas 1, 2 y 3 respectivamente han sido aproximadamente 0.40, 0.41, 0.47 para el apartado S 1-1; 0.42, 0.35, 0.48 para el apartado S 1-2; y 0.44, 0.39, 0.48 para el apartado S 1-3. Los errores vemos que son bastante elevados y en el apartado S 1-3 no se cumplen los resultados esperados, esto puede ser debido a la complejidad de los datos y/o al tamaño muestral reducido.

Hay que tener en cuenta que toda simulación realizada en este proyecto conlleva una tremenda aleatoriedad, sobre todo cuando se trata de los errores condicionados. Por lo tanto, es fácil que los resultados esperados no se cumplan en algunos casos. Además, la elección de las poblaciones influye enormemente en el resultado, siendo más sencillo que se cumplan los resultados esperados en poblaciones disjuntas. Cuanto más confundidas estén las poblaciones, más difícil será de diferenciar entre ellas lo que hará más incierta la elección de la regla que mejor clasifica. Además, al trabajar con más de una dimensión complica los resultados y la aplicación de las reglas discriminantes.

8. Bibliografía.

Delicado, Pedro (2008). Curso de Modelos no Paramétricos, Departament d'Estadística i Investigació Operativa, Universitat Politécnica de Catalunya.

Grané, Aurea. Análisis discriminante y Clasificación. Departamento de Estadística, Universidad Carlos III de Madrid.)

Dr. Acuña Fernández, Edgar (2000). Notas de Análisis Discriminante. Departamento de Matemáticas, Universidad de Puerto Rico en Mayaguez.

Miñarro, Antonio (1998). Estimación no paramétrica de la función de densidad. Barcelona.

Cuadras, Carles M. (2012). Nuevos métodos de análisis multivariante. Barcelona.

Justel, Ana. Docencia: Tema Técnicas de Análisis Discriminante. Universidad Autónoma de Madrid, Madrid.

González de Garibay, Valentín. Docencia: Análisis Multivariante, tema Análisis Discriminante. Universidad de Valladolid, Valladolid.

R tips pages. University of British Columbia.
https://www.zoology.ubc.ca/~schluter/R/multivariate/

9. Apéndice.

Generación de muestras:

```
m.entr=function(m,v,n,pooled=T){
 L=list()
 S=list()
 k=length(n)
 p=nrow(m)
 for (i in 1:k){
  dat=rmvnorm(n[i],m[,i],v[[i]])
  med=apply(dat,2,mean)
  var=cov(dat)
  if(pooled==T)
   S[[i]]=var
  L[[i]]=list(dat=dat,medias=med,Sigma=var)
   }
 names(L)=paste("Muestra" ,1:k)
 if(pooled==T){
  Sp=matrix(0,p,p)
  for (i in 1:k){
   Sp=Sp+(n[i]-1)*S[[i]]
  }
  Sp=Sp/(sum(n)-k)
  L[[k+1]]=list(Spooled=Sp)
 }
 return(L)
}
```

```
mixt=function(n,p,m1,m2,s1,s2){
 n1=round(p*n)
 n2=n-n1
 a1=rmvnorm(n1,m1,s1)
 a2=rmvnorm(n2,m2,s2)
 return( mix=rbind(a1,a2))
}

mue.test=function(N,m,v,mix){
 k=ncol(m)
 t=list()
 if (!is.null(mix)){
  m=v=list()
  pm=0
  for( i in 1:k){
   m[[i]]=mix[[i]]$medias
   v[[i]]=mix[[i]]$vars
   pm[i]=mix[[1]]$p
   t[[i]]=mixt(N,pm[i],m[[i]][,1],m[[i]][,2],v[[i]][[1]],v[[i]][[2]])
  }  }
  else{
  for (i in 1:k)
   t[[i]]=rmvnorm(N,m[,i],v[[i]])
 }
 return(t) }
```

P-dimensiones y k poblaciones.

```
err.lqdak=function(n,N=2000,m,v,mue,mix=NULL,t){
 k=ncol(m)
 p=nrow(m)
 E=matrix(0,n*k,p);  J=matrix(0,N*k,p)
 a=list()
 X=matrix(0,n,k);  Y=matrix(0,N,k)
  #Bucles para colocar los datos para poder realizar la regla con
 #las funciones lda y qda,con la muestra de entrenamiento.
 #Cada columna de la matriz E contiene las n primeras filas población 1, las
 #n segundas filas población 2 etc..., E tiene p filas, la dimensión.
 for (i in 1:k)
   a[[i]]=mue[[i]]$dat
 for (i in 1:p){
   for (j in 1:k)
   X[,j]=a[[j]][,i]
  E[,i]=c(X)
 }


 d=rep(1:k,each=n)
 z=lda(E,d)
 z1=qda(E,d)
```

Cont. función err.lqdak

```
 #Colocación de la muestra test para su clasificación y el cálculo
  #de errores con las funciones lda y qda.
  #J es análoga a la matrriz E pero para la muestra test,contenida en t.

  for (i in 1:p){
    for (j in 1:k)
    Y[,j]=t[[j]][,i]
   J[,i]=c(Y)
  }

  #Tablas y errores:
  class=rep(1:k,each=N)
  pt=predict(z,J)
  tab1=table(class, pt$class)
  elda=(sum(tab1)-sum(diag(tab1)))/(N*k)

  pt1=predict(z1,J)
  tab2=table(class, pt1$class)
  eqda=(sum(tab2)-sum(diag(tab2)))/(N*k)

  return(c(list(tabla.lda=tab1,tabla.qda=tab2),err.lda=elda,err.qda=eqda))
}
```

```
Bayesk=function(N=2000,m,v,mix=NULL,t){
 d=list()
 k=length(t)
 contD=matrix(0,N,k)

 #Si estamos en el tercer escenario Ei3 entonces mix no será nulo, en este caso
 #tenemos que hacer la densidad de la mixtura ->lista d
 if (!is.null(mix)){
  m=v=list()
  pm=0
  for( i in 1:k){
    m[[i]]=mix[[i]]$medias
    v[[i]]=mix[[i]]$vars
    pm[i]=mix[[1]]$p
  }
  for(i in 1:k){
    d[[i]]=matrix(0,N,k)
    for (r in 1:k)
     d[[i]][,r]=pm[i]*(dmvnorm(t[[i]],m[[r]][,1],v[[r]][[1]])) +(1-
pm[i])*dmvnorm(t[[i]],m[[r]][,2],v[[r]][[2]])
  }
 }
```

Cont. función Bayesk:

```
 else{
   #Si estamos en los otros escenarios, se introduce en m y v las
   #medias de las poblaciones y varianzas
   #correspondientes al escenario que estemos.
   #Se calcula la densidad multivariante de las normales de cada población
   #en la lista d
   for(i in 1:k){
     d[[i]]=matrix(0,N,k)
     for (r in 1:k)
       d[[i]][,r]=dmvnorm(t[[i]],m[,r],v[[r]])
   } }
   #Errores: vemos si el máximo es el correspondiente a la densidad de la
 #población de la que procede.
 #ContD=1 si se comete error.
   for(i in 1:k){
   for(j in 1:N){
     if(max(d[[i]][j,])==d[[i]][j,i])    contD[j,i]=0      else contD[j,i]=1
   } }
 bayd=mean(apply(contD,2,sum)/N)
 return(bayd)
 }
```

```
Asint=function(N=2000,m,v,mix=NULL,t){
 S=list()
 p=nrow(m)
 k=ncol(m)
 Sp=matrix(0,p,p)
 Fis=cuad=list()
 d=mu=list()
 contF=contC=matrix(0,N,k)
 contD=matrix(0,N,k)
 if (!is.null(mix)){
  m=v=list()
  pm=0
  for( i in 1:k){
    m[[i]]=mix[[i]]$medias
    v[[i]]=mix[[i]]$vars
    pm[i]=mix[[1]]$p
     }
  for(i in 1:k){
    d[[i]]=matrix(0,N,k)
    S[[i]]=(v[[i]][[1]]+v[[i]][[2]])/2
    mu[[i]]=apply(m[[i]],1,mean)
 for (r in 1:k)
      d[[i]][,r]=pm[i]*(dmvnorm(t[[i]],m[[r]][,1],v[[r]][[1]])) + (1-
pm[i])*dmvnorm(t[[i]],m[[r]][,2],v[[r]][[2]])
     } }
```

Cont. función Asint:

```
 else{
   for(i in 1:k){
     S[[i]]=v[[i]]
     mu[[i]]=m[,i]
     d[[i]]=matrix(0,N,k)
     for (r in 1:k)
       d[[i]][,r]=dmvnorm(t[[i]],m[,r],v[[r]])    } }
 for(i in 1:k)
   Sp=Sp+S[[i]]
   Sp=Sp/k
  for(i in 1:k){
   Fis[[i]]=cuad[[i]]=matrix(0,N,k)
   for(r in 1:k)
  for( j in 1:N){
      Fis[[i]][j,r]=mu[[r]]%*%solve(Sp)%*%t[[i]][j,]-1/2*mu[[r]]%*%solve(Sp)%*%mu[[r]]
      cuad[[i]][j,r]=mu[[r]]%*%solve(S[[r]])%*%t[[i]][j,]-1/2*mu[[r]]%*%solve(S[[r]])%*%mu[[r]] }
   for(j in 1:N){
    if(max(Fis[[i]][j,])==Fis[[i]][j,i])   contF[j,i]=0       else contF[j,i]=1
    if(max(cuad[[i]][j,])==cuad[[i]][j,i])   contC[j,i]=0       else contC[j,i]=1
    if(max(d[[i]][j,])==d[[i]][j,i])    contD[j,i]=0      else contD[j,i]=1     } }
  bayf=mean(apply(contF,2,sum)/N)
  bayc=mean(apply(contC,2,sum)/N)
  bayd=mean(apply(contD,2,sum)/N)
  return(c(fis=bayf,cuad=bayc,nucl=bayd))  }
```

2 dimensiones y k poblaciones.

```
r.nucleok=function(N=2000,dat,m,v,mix=NULL,t){
  k=ncol(m)
 dens=f=cont=matrix(0,N,k)
 e=0
 #matriz dens contiene las densidades estimadas con la función sm.density
 #aplicadas a cada muestra test, cada población una columna,
 #se realiza la suma por filas contenida en el vector den que se utiliza para
 #calcular los valores de la ecuación para k poblaciones de la teoria y hallar los errores.
 #Para los errores cont se hará 1 si el máximo de cada fila coincide con el
 #valor de la fila correspondiente a la poblacion de la que proviene el dato.
 for (i in 1:k){
  for (j in 1:k)
   dens[,j]=diag(sm.density(x=dat[[j]],display="none",eval.points=t[[i]])$estimate)
  den=apply(dens,1,sum)
  for(j in 1:k)
   f[,j]=dens[,j]/den
  for(j in 1:N)
   if(max(f[j,])==f[j,i])  cont[j,i]=0      else cont[j,i]=1
  }
 e=mean(apply(cont,2,sum)/N)
return(e)
}
```

```
fisher=function(x,mu1,mu2,Sp)
  L=t(x-1/2*(mu1+mu2))%*%solve(Sp)%*%(mu1-mu2)


cuad=function(x,mu1,mu2,S1,S2)
  L=t(x-mu1)%*%solve(S1)%*%(x-mu1)-t(x-mu2)%*%solve(S2)%*%(x-mu2)



BayesG=function(N=2000,m,v,mix=NULL,t){
    k=ncol(m)
    if (!is.null(mix)){
    m=v=list()
    pm=0
    for( i in 1:k){
      m[[i]]=mix[[i]]$medias
      v[[i]]=mix[[i]]$vars
      pm[i]=mix[[1]]$p
    }
d=matrix(0,nrow(t),k)
    for (r in 1:k)
      d[,r]=pm[i]*(dmvnorm(t,m[[r]][,1],v[[r]][[1]])) +(1-pm[i])*dmvnorm(t,m[[r]][,2],v[[r]][[2]])     }
  else{
    d=matrix(0,nrow(t),k)
    for (r in 1:k)
      d[,r]=dmvnorm(t,m[,r],v[[r]])    }
  return(d) }
```

```
grafs=function(dat,mue,m,v,mix,e){
 dens=difden=list(list())
 x=y=matrix(0,50,3)
 k=ncol(m[[1]])
 p=nrow(m[[1]])
 leg=0
 i=1
 for(j in 1:(k-1))
  for(l in (j+1):k){
   leg[i]=paste(j,"-",l,sep="")
   i=i+1    }
 nc=length(leg)
 if(k==2)    if(p==2) esc="E" else esc="S"
 else esc="M"
 for (i in 1:3){
  #Representación de cada población, con la leyenda
  plot(-4:4,-4:4,type="n",xlab=" ",ylab=" ")
  dens[[i]]=list()
  dens[[i]][[1]]=sm.density(dat[[i]][[1]]$dat,add=T,pch="1",display="slice",xlim=a,ylim=b,col=1)
  title(paste("Escenario ",esc,e,i,sep=""))
  for(j in 2:k){
   pch=as.character(j)
   dens[[i]][[j]]=sm.density(dat[[i]][[j]]$dat,add=T,display="slice",xlim=a,ylim=b,col=j)
   points(dat[[i]][[j]]$dat,pch=pch,col=j,cex=0.7)
  }
  legend("bottomleft",paste("Población",1:k),text.col=1:k)
```

Cont. función grafs

```
 #Diferencia de densidades para la regla núcleo
   for(j in 1:(k-1)){
    difden[[j]]=list()
    for(l in (j+1):k)
     difden[[j]][[l]]=dens[[i]][[j]]$est-dens[[i]][[l]]$est    }
   x[,i]=dens[[i]][[1]]$eval.points[,1]
   y[,i]=dens[[i]][[2]]$eval.points[,2]
   z=w=ba=list(list(matrix(0,length(x[,i]),length(y[,i]))))
   n1=nrow(x)
   n2=nrow(y)
   #Cálculo de las funciones a representar: fisher, cuadrática y Bayes.
   #Se calculan 50x50 valores correspondientes a los puntos donde se evalúan
   #las densidades estimadas.    #Se calculan por parejas entre poblaciones para
   #poder representarlo si hay mas de dos poblaciones.    Siendo la primera lista una población, que
se              #representará con las poblaciones
   #restantes contenidas en una segunda lista dentro de esta. Es decir,
   #z[[1]][[2]]: datos de la población 1 con la 2
   #Fisher ->función fisher()
   for(r in 1:(k-1)){
    z[[r]]=list()
    for(l in (r+1):k){
     z[[r]][[l]]=matrix(0,n1,n2)
     for (q in 1:n1)
      for (j in 1:n2)
       z[[r]][[l]][q,j]=fisher(c(x[q,i],y[j,i]),mu1=mue[[i]][[r]]$medias,
                               mu2=mue[[i]][[l]]$medias,Sp=mue[[i]][[k+1]]$Spooled)    }  }
```

Cont. función grafs (2)

```
#Cuadratica-> función cuad()
for(r in 1:(k-1)){
 w[[r]]=list()
 for(l in (r+1):k){
  w[[r]][[l]]=matrix(0,n1,n2)
  for (q in 1:n1)
   for (j in 1:n2)
    w[[r]][[l]][q,j]=cuad(c(x[q,i],y[j,i]),mu1=mue[[i]][[r]]$medias,
                mu2=mue[[i]][[l]]$medias,S1=mue[[i]][[r]]$Sigma,S2=mue[[i]][[l]]$Sigma)  } }

t=matrix(0,n1*n2,2)
aux=0

#Bayes -> función bayesk
if(i==3){
 r=1
 for (q in 1:n1)
  for (j in 1:n2){
   t[r,]=c(x[q,i],y[j,i])
   r=r+1      }
 bay=BayesG(t=t,m=m[[2]],v=v[[2]],mix=mix)
 for(r in 1:(k-1)){
  ba[[r]]=list()
  for(l in (r+1):k){
   aux=bay[,r]-bay[,l]
   ba[[r]][[l]]=matrix(aux,n1,n2,byrow=T)      }   }  }
```

Cont. función grafs (3)

```
else{
    r=1
    for (q in 1:n1)
     for (j in 1:n2){
       t[r,]=c(x[q,i],y[j,i])
       r=r+1       }
    bay=BayesG(t=t,m=m[[i]],v=v[[i]],mix=NULL)
    for(r in 1:(k-1)){
     ba[[r]]=list()
     for(l in (r+1):k){
       aux=bay[,r]-bay[,l]
       ba[[r]][[l]]=matrix(aux,n1,n2,byrow=T)       }     }    }


  if(k==2){
   contour(x[,i],y[,i],difden[[1]][[2]],levels=c(0),lwd=2,col=1)
    points(dat[[i]][[1]]$dat,pch="1",col="blue")
    points(dat[[i]][[2]]$dat,pch="2",col="red")
    title(paste("Regla Núcleo, escenario ",esc,e,i,sep=""))
    contour(x[,i],y[,i],z[[1]][[2]],levels=c(0),lwd=2,col=3)
    points(dat[[i]][[1]]$dat,pch="1",col="blue")
    points(dat[[i]][[2]]$dat,pch="2",col="red")
    title(paste("Regla discriminante lineal, escenario ",esc,e,i,sep=""))
     contour(x[,i],y[,i],w[[1]][[2]],levels=c(0),lwd=2,col=6)
    points(dat[[i]][[1]]$dat,pch="1",col="blue")
    points(dat[[i]][[2]]$dat,pch="2",col="red")
    title(paste("Regla discriminante cuadrática, escenario ",esc,e,i,sep=""))
```

Cont. función grafs (4)

```
 contour(x[,i],y[,i],ba[[1]][[2]],levels=c(0),lwd=2,col=7)
    points(dat[[i]][[1]]$dat,pch="1",col="blue")
    points(dat[[i]][[2]]$dat,pch="2",col="red")
    title(paste("Regla Bayes, escenario ",esc,e,i,sep=""))     }


  else{
    plot(x[,i],y[,i],type="n")
    q=4
    for(j in 1:(k-1))
     for(l in (j+1):k){
       contour(x[,i],y[,i],difden[[j]][[l]],add=T,levels=c(0),lwd=2,col=q)
       q=q+1       }
    for(j in 1:k){
     pch=as.character(j)
     points(dat[[i]][[j]]$dat,pch=pch,col=j)     }
    legend("topleft",leg,text.col=seq(4,4+(nc-1),))
    title(paste("Regla Núcleo, escenario ",esc,e,i,sep=""))
  plot(x[,i],y[,i],type="n")  ;     q=4
    for(j in 1:(k-1))
     for(l in (j+1):k){     contour(x[,i],y[,i],z[[j]][[l]],add=T,levels=c(0),lwd=2,col=q)
       q=q+1       }
    for(j in 1:k){
     pch=as.character(j)
     points(dat[[i]][[j]]$dat,pch=pch,col=j)     }
    legend("topleft",leg,text.col=seq(4,4+(nc-1)))
```

Cont. función grafs (5)

```
  title(paste("Regla discriminante lineal, escenario ",esc,e,i,sep=""))


    q=4;      plot(x[,i],y[,i],type="n")
    for(j in 1:(k-1))
     for(l in (j+1):k){
       contour(x[,i],y[,i],w[[j]][[l]],add=T,levels=c(0),lwd=2,col=q)
       q=q+1       }
    for(j in 1:k){
     pch=as.character(j)
     points(dat[[i]][[j]]$dat,pch=pch,col=j)     }
    legend("topleft",leg,text.col=seq(4,4+(nc-1)))
    title(paste("Regla discriminante cuadratica, escenario ",esc,e,i,sep=""))


    plot(x[,i],y[,i],type="n")
    q=4
    for(j in 1:(k-1))
     for(l in (j+1):k){
       contour(x[,i],y[,i],ba[[j]][[l]],add=T,levels=c(0),lwd=2,col=q)
       q=q+1
     }
    for(j in 1:k){
     pch=as.character(j)
     points(dat[[i]][[j]]$dat,pch=pch,col=j)
    }
    legend("topleft",leg,text.col=seq(4,4+(nc-1),))
    title(paste("Regla Bayes, escenario ",esc,e,i,sep=""))   } } }
```

```
err.aparente2k=function(x){
 k=length(x)
 n=nrow(x[[1]])
 p=ncol(x[[1]])
 E=matrix(0,n*k,p);
 X=matrix(0,n,k);
 #Análogo a la función err.lqdak pero con las muestras de entrenamiento
 for (i in 1:p){
  for (j in 1:k)
  X[,j]=x[[j]][,i]
 E[,i]=c(X)  }
 d=rep(1:k,each=n)
 z=lda(E,d)
 z1=qda(E,d)
 pt=predict(z,E)
 tab1=table(d, pt$class)
 elda=(sum(tab1)-sum(diag(tab1)))/(n*k)

 pt1=predict(z1,E)
 tab2=table(d, pt1$class)
 eqda=(sum(tab2)-sum(diag(tab2)))/(n*k)
 dens=f=cont=matrix(0,n*k,k)
 e=0
 #Análogo a la función r.nucleok pero con la muestra de entrenamiento
 for (i in 1:k){
   for (j in 1:k)
    dens[,j]=diag(sm.density(x=x[[j]],display="none",eval.points=x[[i]])$estimate)
  den=apply(dens,1,sum)
```

Cont. función err.aparente2k

```
  for(j in 1:k)
    f[,j]=dens[,j]/den
  for(j in 1:nrow(f))
    if(max(f[j,])==f[j,i])  cont[j,i]=0      else cont[j,i]=1   }
 e=mean(apply(cont,2,sum)/(n*k))
 return(c(lda=elda,qda=eqda,fn=e))  }
```

```
err.cross2k=function(x){
 e=0
 k=length(x)
 n=nrow(x[[1]])
 p=ncol(x[[1]])
 pt=pt1=0
 E=G=matrix(0,n*k,p);
 X=matrix(0,n,k);
#Análogo a la funcion err.lqdak, la matriz E.
 for (i in 1:p){
  for (j in 1:k)
  X[,j]=x[[j]][,i]
 E[,i]=c(X)  }
 N=nrow(E)
```

Cont. función err.cross2k

```
#Bucle que toma G matriz con todas las observaciones menos la fila i-ésima,
#x0 observación i-ésima
#d: contiene los valores 1:k para la clasificacion de las poblaciones
for(i in 1:N){
 G=E[-i,]
 x0=E[i,]
 d=rep(1:k,each=n)
 n0=0
 #Bucle para tomar los tamaños de las poblaciones una vez quitada la
 #observación, será n=20 la poblacion a la que no pertenezca la fila i, y
 #n=19 la población a la que pertenezca. Con esto podemos crear d0
 # que tiene los valores 1:k n veces cada poblacion
 #para la clasificación de las poblaciones
 for(j in 1:k)
 n0[j]=length(G[d[-i]==j,1])
 d0=rep(1:k,n0)
 #Se crean las reglas con G y se clasifica con predict, la observación x0
 z=lda(G,d0)
 z1=qda(G,d0)
 pt[i]=predict(z,x0)$class
 pt1[i]=predict(z1,x0)$class  }
#Calculo de los errores
 d=rep(1:k,each=n);  tab1=table(d, pt)
 elda=(sum(tab1)-sum(diag(tab1)))/(n*k)
 tab2=table(d, pt1)
 eqda=(sum(tab2)-sum(diag(tab2)))/(n*k)
 x=list() ;  cont=0
```

Cont. función err.cross2k (2)

```
for(i in 1:N){
    dens=f=0
    x0=matrix(E[i,],1,2)
    G=E[-i,]
    d0=d[-i]
  for (j in 1:k){
    x[[j]]=G[d0==j,]
    dens[j]=sm.density(x=x[[j]],display="none",eval.points=x0)$estimate    }
  den=sum(dens)
  for(j in 1:k)
    f[j]=dens[j]/den
  if(max(f)==f[d[i]])   cont[i]=0       else cont[i]=1
  }
    for( i in 1:k)
   e[i]=sum(cont[d==i])/n
  err=mean(e)
 return(c(lda=elda,qda=eqda,fn=err))    }
```

```
err.boot2k=function(x,M){
  k=length(x)
  n=nrow(x[[1]])
  p=ncol(x[[1]])
  nj=1:n
  elda=eqda=err=0
  w=matrix(0,n,k)

  for (r in 1:M){
   b=no=x0=E=E0=list()
   #Se toma la muestra bootstrap
    for( j in 1:k){
      w[,j]=sample(nj,replace=T) #muestra con reempl de las filas
      b[[j]]=x[[j]][w[,j],]  #tomamos las filas correspondientes de x
      no[[j]]=nj[!nj%in%w[,j]]  #filas que no se han incluído en la muestra
      x0[[j]]=x[[j]][no[[j]],]  #se toman las filas correspondientes de x
    }
   #Errores lda y qda
    E=b[[1]]
    n0=length(no[[1]])
    E0=x0[[1]]
   #Preparamos los datos en matrices
       for(j in 2:k){
       E0=rbind(E0,x0[[j]])
       E=rbind(E,b[[j]])
       n0[j]=length(no[[j]])          }
```

Cont. función err.boot2k

```
    d=rep(1:k,each=n);       z=lda(E,d)
    z1=qda(E,d)
    pt=predict(z,E0)
    d0=rep(1:k,n0)
    tab1=table(d0, pt$class)
    elda[r]=(sum(tab1)-sum(diag(tab1)))/(sum(n0))
    pt1=predict(z1,E0)
    tab2=table(d0, pt1$class)
    eqda[r]=(sum(tab2)-sum(diag(tab2)))/(sum(n0))
   e=0
    for (i in 1:k){
     dens=f=matrix(0,nrow(x0[[i]]),k)
     cont=0
    for (j in 1:k)
       dens[,j]=diag(sm.density(x=b[[j]],display="none",eval.points=x0[[i]])$estimate)
     den=apply(dens,1,sum)
     for(j in 1:k)
      f[,j]=dens[,j]/den
     for(j in 1:nrow(f))
      if(max(f[j,])==f[j,i])   cont[j]=0    else cont[j]=1
    e[i]=sum(cont)/nrow(f)       }
    err[r]=mean(e)       }
  melda=mean(elda)
  meqda=mean(eqda)
  merr.fn=mean(err)
  return(c(lda=melda,qda=meqda,efn=merr.fn))   }
```

```
errores.nocondk=function(NC,m,v,mmix,vmix,N=2000,nb=200,ni=20,a,b,e){

 E1=E2=E3=Ea1=Bay1=Bay2=Bay3=Ea2=Ea3=Ecv1=Ecv2=Ecv3=
Eb1=Eb2=Eb3=Eb61=Eb62=Eb63=matrix(0,3,NC)

 As1=As2=As3=matrix(0,3,NC)


 for (r in 1:NC){

  k=ncol(m[[1]])

  p=nrow(m[[1]])

  n=rep(ni,k)

  mue=dat=mix=mumix=Smix=list(list())

  #creación de las muestras de entrenamiento, lista mue y dat con los valores de estas muestras
para los escenarios 1 y 2

  for(i in 1:2){

   mue[[i]]=m.entr(m=m[[i]],v=v[[i]],n)

   dat[[i]]=list()

   for(j in 1:k)

    dat[[i]][[j]]=list(dat=mue[[i]][[j]]$dat)

  }


  #Escenario 3, creación de las mixturas

  for(i in 1:k){

   mix[[i]]=list(dat=mixt(n[k],p=0.5,mmix[[i]][,1],mmix[[i]][,2],vmix[[i]][[1]],vmix[[i]][[2]]))

   mumix[[i]]=apply(mix[[i]]$dat,2,mean)

   Smix[[i]]=cov(mix[[i]]$dat)

  }

  Spmix=(Smix[[1]]*(n[k]-1)+Smix[[2]]*(n[k]-1))/(n[k]*2-2)

  dat[[3]]=mix

  mue[[3]]=list()
```

Cont. función errores.nocondk

```
 for(i in 1:k)
     mue[[3]][[i]]=list(medias=mumix[[i]],Sigma=Smix[[i]])
   mue[[3]][[k+1]]=list(Spooled=Spmix)
ea1=ea2=ea3=eb1=eb2=eb3=ecv1=ecv2=ecv3=bay1=bay2=bay3=as1=as2=as3=0
   e1=e2=e2=e3=list()
   m1=m[[1]]
   v1=v[[1]]
   mue1=mue[[1]]
    mix=list()
   for(i in 1:k)
     mix[[i]]=list(medias=mmix[[i]],vars=vmix[[i]],p=0.5)
   t=list()
   for (i in 1:length(m))
    t[[i]]=mue.test(N=N,m[[i]],v[[i]],mix=NULL)
   t[[3]]=mue.test(N=N,m[[2]],v[[2]],mix=mix)
   #errores escenario 1
   e1=err.lqdak(n=ni,N=N,mue=dat[[1]],m=m1,v=v1,mix=NULL,t=t[[1]])
   e11=e1$err.lda
   e12=e1$err.qda
   dat1=list()
   for( i in 1:k)
    dat1[[i]]=dat[[1]][[i]]$dat
   e13=r.nucleok(N=N,dat=dat1,m=m1,v=v1,mix=NULL,t=t[[1]])
   bay1=Bayesk(m=m1,v=v1,mix=NULL,t=t[[1]])
   as1=Asint(m=m1,v=v1,mix=NULL,t=t[[1]])
   ea1=err.aparente2k(dat1);     ecv1=err.cross2k(dat1);    eb1=err.boot2k(dat1,nb)
```

Cont. función errores.nocondk (2)

```
  ####Escenario 2
  m2=m[[2]]
  v2=v[[2]]
  mue2=mue[[2]]
  #errores escenario 2
  e2=err.lqdak(n=ni,N=N,mue=dat[[2]],m=m2,v=v2,mix=NULL,t=t[[2]])
  e21=e2$err.lda
  e22=e2$err.qda
 dat2=list()
  for( i in 1:k)
    dat2[[i]]=dat[[2]][[i]]$dat
  e23=r.nucleok(N=N,dat=dat2,m=m2,v=v2,mix=NULL,t=t[[2]])
  ea2=err.aparente2k(dat2)
  as2=Asint(m=m2,v=v2,mix=NULL,t=t[[2]])
  bay2=Bayesk(m=m2,v=v2,mix=NULL,t=t[[2]])
  ecv2=err.cross2k(dat2)
  eb2=err.boot2k(dat2,nb)


  #####Escenario 3
 #Errores escenario 3
  e3=err.lqdak(n=ni,N=N,mue=dat[[3]],m=m2,v=v2,mix=mix,t=t[[3]])
  e31=e3$err.lda
  e32=e3$err.qda
  dat3=list()
  for( i in 1:k)
    dat3[[i]]=dat[[3]][[i]]$dat
```

Cont. función errores.nocondk (3)

```
  e33=r.nucleok(N=N,dat=dat3,m=m2,v=v2,mix=mix,t=t[[3]])
  ea3=err.aparente2k(dat3)
  as3=Asint(m=m2,v=v2,mix=mix,t=t[[3]])
  bay3=Bayesk(m=m2,v=v2,mix=mix,t=t[[3]])
  ecv3=err.cross2k(dat3)
  eb3=err.boot2k(dat3,nb)

 ########################Tablas##############################
  E1[,r]=c(e11,e12,e13)
  E2[,r]=c(e21,e22,e23)
  E3[,r]=c(e31,e32,e33)
  Ea1[,r]=ea1
  Ea2[,r]=ea2
  Ea3[,r]=ea3
  Ecv1[,r]=ecv1
  Ecv2[,r]=ecv2
  Ecv3[,r]=ecv3
  Eb1[,r]=eb1
  Eb2[,r]=eb2
  Eb3[,r]=eb3
  Eb61[,r]=eb1*(0.632)+(1-0.632)*ea1
  Eb62[,r]=eb2*(0.632)+(1-0.632)*ea2
  Eb63[,r]=eb3*(0.632)+(1-0.632)*ea3
  Bay1[,r]=c(bay1,bay2,bay3)
  Bay2[,r]=c(bay1,bay2,bay3)
  Bay3[,r]=c(bay1,bay2,bay3)
```

Cont. función errores.nocondk (4)

```
As1[,r]=as1
   As2[,r]=as2
   As3[,r]=as3  }
 #Gráficos si tenemos una muestra de entrenamiento unicamente
 if ((NC==1)&&(ni<30)&&(p<3)) grafs(dat=dat,mue=mue,m=m,v=v,mix=mix,e)
 #TABLAS, medias de las NC muestras de entrenamiento
 ER=cbind(apply(E1,1,mean),apply(E2,1,mean),apply(E3,1,mean))
 AS=cbind(apply(As1,1,mean),apply(As2,1,mean),apply(As3,1,mean))
 if(NC>1){
   BAY=cbind(apply(Bay1,1,mean),apply(Bay2,1,mean),apply(Bay3,1,mean))
   R=paste("R",1:3,sep="")
   E=c("Real NO Cond.","Bayes")
   T1=T2=T3=matrix(0,2,3,dimnames=list(E,R))
   for( i in 1:3){ T1[1,i]=ER[1,i]
                   T1[2,i]=BAY[1,i]    }
   for( i in 1:3){ T2[1,i]=ER[2,i]
                   T2[2,i]=BAY[2,i]    }
   for( i in 1:3){ T3[1,i]=ER[3,i]
                   T3[2,i]=BAY[3,i]    }
       return(list(E1=T1,E2=T2,E3=T3))          }

 else{
 EA=cbind(apply(Ea1,1,mean),apply(Ea2,1,mean),apply(Ea3,1,mean))
 ECV=cbind(apply(Ecv1,1,mean),apply(Ecv2,1,mean),apply(Ecv3,1,mean))
 EB=cbind(apply(Eb1,1,mean),apply(Eb2,1,mean),apply(Eb3,1,mean))
 EB6=cbind(apply(Eb61,1,mean),apply(Eb62,1,mean),apply(Eb63,1,mean))
```

Cont. función errores.nocondk (5)

```
R=paste("R",1:3,sep="")
E=c("Real Cond.","Aparente","Cross Val.","Bootstrap","Boot632","Asintotico")
 Tab1=Tab2=Tab3=matrix(0,6,3,dimnames=list(E,R))

 for( i in 1:3){ Tab1[1,i]=ER[1,i]
                 Tab1[2,i]=EA[1,i]
                 Tab1[3,i]=ECV[1,i]
                 Tab1[4,i]=EB[1,i]
                 Tab1[5,i]=EB6[1,i]
                 Tab1[6,i]=AS[1,i]  }
 for( i in 1:3){ Tab2[1,i]=ER[2,i]
                 Tab2[2,i]=EA[2,i]
                 Tab2[3,i]=ECV[2,i]
                 Tab2[4,i]=EB[2,i]
                 Tab2[5,i]=EB6[2,i]
                 Tab2[6,i]=AS[2,i]  }
 for( i in 1:3){ Tab3[1,i]=ER[3,i]
                 Tab3[2,i]=EA[3,i]
                 Tab3[3,i]=ECV[3,i]
                 Tab3[4,i]=EB[3,i]
                 Tab3[5,i]=EB6[3,i]
                 Tab3[6,i]=AS[3,i]  }
 return(list(E1=round(Tab1,4),E2=round(Tab2,4),E3=round(Tab3,4)))    }    }
```

P dimensiones y 2 poblaciones.

```
err.fn=function(N,m,mix=NULL,v,x1,x2,t){

 h=(hnorm(x1)+hnorm(x2))/2
 #Cálculo de la función núcleo.
 #x: vector a clasificar
 #xe: matriz con la muestra de entrenamiento.
 f.nucleo=function(x){
 aux=0
 n=nrow(xe)
 H=diag(h)
 for(i in 1:n) aux=aux+prod(dmvnorm(solve(H)%*%(x-xe[i,])))
 fun=1/(n*abs(det(H)))*aux
 return(fun)
 }

 xe=x1
  dens11=apply(t[[1]],1,f.nucleo)
  dens21=apply(t[[2]],1,f.nucleo)
  xe=x2
  dens12=apply(t[[1]],1,f.nucleo)
  dens22=apply(t[[2]],1,f.nucleo)
 difden1=dens11-dens12
 difden2=dens21-dens22
  e1=sum(difden1<0)/N
 e2=sum(difden2>=0)/N
 err.fn=(e1+e2)/2
 return(err.fn)
 }
```

```
err.aparente=function(x){

 k=length(x)
 n=nrow(x[[1]])
 p=ncol(x[[1]])
 h=(hnorm(x[[1]])+hnorm(x[[2]]))/2
 E=matrix(0,n*k,p);
 X=matrix(0,n,k);
 for (i in 1:p){
   for (j in 1:k)
   X[,j]=x[[j]][,i]
  E[,i]=c(X)  }
 d=rep(1:k,each=n)
 z=lda(E,d)
 z1=qda(E,d)
 pt=predict(z,E)
 tab1=table(d, pt$class)
 elda=(sum(tab1)-sum(diag(tab1)))/(n*k)
 pt1=predict(z1,E)
 tab2=table(d, pt1$class)
 eqda=(sum(tab2)-sum(diag(tab2)))/(n*k)

 #Error función núcleo
 f.nucleo<-function(x){
 aux=0
 n=nrow(xe)
 H=diag(h)
 for(i in 1:n) aux=aux+prod(dmvnorm(solve(H)%*%(x-xe[i,])))
 fun=1/(n*abs(det(H)))*aux
 return(fun)  }
  xe=x[[1]]
  dens11<-apply(xe,1,f.nucleo)
   dens21=apply(x[[2]],1,f.nucleo)
   xe=x[[2]]
   dens12=apply(x[[1]],1,f.nucleo)
   dens22=apply(xe,1,f.nucleo)
 difden1=dens11-dens12;   difden2=dens21-dens22
 N1=nrow(x[[1]]);   N2=nrow(x[[2]])
  e1=sum(difden1<0)/N1 ;  e2=sum(difden2>=0)/N2
 err.fn=(e1+e2)/2
 return(c(lda=elda,qda=eqda,fn=err.fn))   }
```

```
err.cross=function(x){

 k=length(x)
 n=nrow(x[[1]])
 p=ncol(x[[1]])
 h=(hnorm(x[[1]])+hnorm(x[[2]]))/2
 pt=pt1=difden=0
 E=G=matrix(0,n*k,p);
 X=matrix(0,n,k);
for (i in 1:p){
  for (j in 1:k)
  X[,j]=x[[j]][,i]
 E[,i]=c(X)
 }
 N=nrow(E)
  for(i in 1:N){
  G=E[-i,]
  x0=E[i,]
  n0=rep(N/k,k)
  for( j in 1:(k-1))
    if((N/k*j)<i&&(N/k*(j+1))>i) n0[j+1]=N/k-1 else n0[1]=N/k-1
  d=rep(1:k,n0)
  z=lda(G,d)
  z1=qda(G,d)
  pt[i]=predict(z,x0)$class
  pt1[i]=predict(z1,x0)$class    }
 d=rep(1:k,each=n)
 tab1=table(d, pt)
 elda=(sum(tab1)-sum(diag(tab1)))/(n*k)
 tab2=table(d, pt1)
 eqda=(sum(tab2)-sum(diag(tab2)))/(n*k)
 #Error función núcleo
 f.nucleo=function(x){
 aux=0
 n=nrow(xe)
 H=diag(h)
 for(i in 1:n) aux=aux+prod(dmvnorm(solve(H)%*%(x-xe[i,])))
 fun=1/(n*abs(det(H)))*aux
 return(fun)
 }
```

Cont. función err.cross

```
 for(i in 1:N){
   x0=E[i,]
   G=E[-i,]
   if (i<(n+1)){
    xe=G[1:(n-1),]
    d1=f.nucleo(x0)
    xe=G[n:(N-1),]
    d2=f.nucleo(x0)
   }
   else{
    xe=G[1:n,]
    d1=f.nucleo(x0)
    xe=G[(n+1):(N-1),]
    d2=f.nucleo(x0)
   }
   difden[i]=d1-d2
 }

 e1=sum(difden[1:n]<0)/n
e2=sum(difden[(n+1):N]>=0)/n
err.fn=(e1+e2)/2
return(c(lda=elda,qda=eqda,fn=err.fn))
}
```

```
err.boot=function(x,M){

  k=length(x)
  n=nrow(x[[1]])
  p=ncol(x[[1]])
  x1=x[[1]]
  x2=x[[2]]
  n1=1:nrow(x1)
  n2=1:nrow(x2)
  h=(hnorm(x1)+hnorm(x2))/2

 #Error función núcleo
  f.nucleo=function(x){
  aux=0
  n=nrow(xe)
  H=diag(h)
  for(i in 1:n) aux=aux+prod(dmvnorm(solve(H)%*%(x-xe[i,])))
  fun=1/(n*abs(det(H)))*aux
  return(fun)
  }
  elda=eqda=err.fn=0
  for (j in 1:M){

    #muestra bootstrap
     w1=sample(n1,replace=T)
     b1=x1[w1,]
     w2=sample(n2,replace=T)
     b2=x2[w2,]
     no1=n1[!n1%in%w1]
     x01=x1[no1,]
     no2=n2[!n2%in%w2]
     x02=x2[no2,]
    #Errores lda y qda
     E=rbind(b1,b2)
     d=rep(1:k,each=n)
     z=lda(E,d)
     z1=qda(E,d)
     n0=c(length(no1),length(no2))
     E0=rbind(x01,x02)
     pt=predict(z,E0)
     d0=rep(1:k,n0)
```

Cont. función err.boot

```
  tab1=table(d0, pt$class)
    elda[j]=(sum(tab1)-sum(diag(tab1)))/(sum(n0))
    pt1=predict(z1,E0)
    tab2=table(d0, pt1$class)
    eqda[j]=(sum(tab2)-sum(diag(tab2)))/(sum(n0))

   #Error función núcleo
    xe=b1
     dens11=apply(x01,1,f.nucleo)
     dens21=apply(x02,1,f.nucleo)
    xe=b2
     dens12=apply(x01,1,f.nucleo)
     dens22=apply(x02,1,f.nucleo)

    difden1=dens11-dens12
    difden2=dens21-dens22
    N1=length(x01)
    N2=length(x02)
     e1=sum(difden1<0)/N1
    e2=sum(difden2>=0)/N2
    err.fn[j]=(e1+e2)/2
 }
  melda=mean(elda)
  meqda=mean(eqda)
  merr.fn=mean(err.fn)
  return(c(lda=melda,qda=meqda,efn=merr.fn))
}
```

```
errores.nocond3=function(NC=200,M1,M2,M5,ni){
 E1=E2=E3=Ea1=Ea2=Ea3=Ecv1=Ecv2=Ecv3=Eb1=Eb2=Eb3=Eb61=Eb62=Eb63=
Bay1=Bay2=Bay3=matrix(0,3,NC)
 As1=As2=As3=matrix(0,3,NC)
 m1=cbind(M1,M2)
 v1=list(I,I)
 n1=rep(ni,2)
 m2=cbind(M1,M2)
 v2=list(I,A)
 n2=rep(ni,2)

 for (r in 1:NC){
  ea1=ea2=ea3=eb1=eb2=eb3=ecv1=ecv2=ecv3=0
  e1=e2=e2=e3=list()
  mix1=mixt(ni,0.8,M1,M2,I,A)
  mix2=mixt(ni,0.5,M5,M2,I,A)
  mmix=list()
  mmix[[1]]=list(dat=mix1)
  mmix[[2]]=list(dat=mix2)
  mix=list(list(medias1=cbind(M1,M2),vars1=list(I,A),p=0.5),
       list(medias2=cbind(M5,M2),vars2=list(I,A),p=0.5))

  t=list()
  t[[1]]=mue.test(N=1000,m1,v1,mix=NULL)
  t[[2]]=mue.test(N=1000,m2,v2,mix=NULL)
  t[[3]]=mue.test(N=1000,m2,v2,mix=mix)
```

Cont. función errores.nocond3

```
#Escenario 1
mue1=m.entr(m=m1,v=v1,n1)
dat11=mue1[[1]]$dat
dat12=mue1[[2]]$dat
e1=err.lqdak(n=ni,N=1000,mue=mue1,m=m1,v=v1,mix=NULL,t=t[[1]])
e11=e1$err.lda
e12=e1$err.qda
e13=err.fn(N=1000,m=m1,v=v1,x1=dat11,x2=dat12,mix=NULL,t=t[[1]])
x1=list(dat11,dat12)
ea1=err.aparente(x1)
bay1=Bayesk(m=m1,v=v1,mix=NULL,t=t[[1]])
ecv1=err.cross(x1)
eb1=err.boot(x1,200)
as1=Asint(N=1000,m=m1,v=v1,mix=NULL,t=t[[1]])

#Escenario 2
mue2=m.entr(m=m2,v=v2,n2)
dat21=mue2[[1]]$dat
dat22=mue2[[2]]$dat
e2=err.lqdak(n=ni,N=1000,mue=mue2,m=m2,v=v2,mix=NULL,t=t[[2]])
e21=e2$err.lda
e22=e2$err.qda
e23=err.fn(N=1000,m=m2,v=v2,x1=dat21,x2=dat22,mix=NULL,t=t[[2]])
x2=list(dat21,dat22)
ea2=err.aparente(x2)
bay2=Bayesk(m=m2,v=v2,mix=NULL,t=t[[2]])
```

Cont. función errores.nocond3 (2)

```
   ecv2=err.cross(x2)
   eb2=err.boot(x2,200)
   as2=Asint(N=1000,m=m2,v=v2,mix=NULL,t=t[[2]])
   e3=err.lqdak(n=ni,N=1000,mue=mmix,m=m2,v=v2,mix=mix,t=t[[3]])
   e31=e3$err.lda
   e32=e3$err.qda
   e33=err.fn(N=1000,m=0,v=0,mix=mix,x1=mix1,x2=mix2,t=t[[3]])
   x3=list(mix1,mix2)
   ea3=err.aparente(x3)
   bay3=Bayesk(m=m2,v=v2,mix=mix,t=t[[3]])
   ecv3=err.cross(x3)
   eb3=err.boot(x3,200)
   as3=Asint(N=1000,m=m2,v=v2,mix=mix,t=t[[3]])
#Tablas
   E1[,r]=c(e11,e12,e13)
   E2[,r]=c(e21,e22,e23)
   E3[,r]=c(e31,e32,e33)

   Ea1[,r]=ea1
   Ea2[,r]=ea2
   Ea3[,r]=ea3

   Ecv1[,r]=ecv1
   Ecv2[,r]=ecv2
   Ecv3[,r]=ecv3
```

Cont. función errores.nocond3 (3)

```
  Eb1[,r]=eb1

  Eb2[,r]=eb2

  Eb3[,r]=eb3


  Eb61[,r]=eb1*(0.632)+(1-0.632)*ea1

  Eb62[,r]=eb2*(0.632)+(1-0.632)*ea2

  Eb63[,r]=eb3*(0.632)+(1-0.632)*ea3

  Bay1[,r]=c(bay1,bay2,bay3)

  Bay2[,r]=c(bay1,bay2,bay3)

  Bay3[,r]=c(bay1,bay2,bay3)

 As1[,r]=as1

  As2[,r]=as2

  As3[,r]=as3  }


 ER=cbind(apply(E1,1,mean),apply(E2,1,mean),apply(E3,1,mean))

 AS=cbind(apply(As1,1,mean),apply(As2,1,mean),apply(As3,1,mean))


 if(NC>1){

  BAY=cbind(apply(Bay1,1,mean),apply(Bay2,1,mean),apply(Bay3,1,mean))

  R=paste("R",1:3,sep="")

  E=c("Real NO Cond.","Bayes")

  T1=T2=T3=matrix(0,2,3,dimnames=list(E,R))

  for( i in 1:3){ T1[1,i]=ER[1,i]

                  T1[2,i]=BAY[1,i]    }

  for( i in 1:3){ T2[1,i]=ER[2,i]

                  T2[2,i]=BAY[2,i]    }
```

Cont. función errores.nocond3 (4)

```
  for( i in 1:3){ T3[1,i]=ER[3,i]
                  T3[2,i]=BAY[3,i]    }


   return(list(E1=T1,E2=T2,E3=T3))           }
  else{
   EA=cbind(apply(Ea1,1,mean),apply(Ea2,1,mean),apply(Ea3,1,mean))
   ECV=cbind(apply(Ecv1,1,mean),apply(Ecv2,1,mean),apply(Ecv3,1,mean))
   EB=cbind(apply(Eb1,1,mean),apply(Eb2,1,mean),apply(Eb3,1,mean))
   EB6=cbind(apply(Eb61,1,mean),apply(Eb62,1,mean),apply(Eb63,1,mean))
   R=paste("R",1:3,sep="")
   E=c("Real Cond.","Aparente","Cross Val.","Bootstrap","Boot632","Asintotico")
   Tab1=Tab2=Tab3=matrix(0,6,3,dimnames=list(E,R))


   for( i in 1:3){ Tab1[1,i]=ER[1,i]
                   Tab1[2,i]=EA[1,i]
                   Tab1[3,i]=ECV[1,i]
                   Tab1[4,i]=EB[1,i]
                   Tab1[5,i]=EB6[1,i]
                   Tab1[6,i]=AS[1,i]    }
   for( i in 1:3){ Tab2[1,i]=ER[2,i]
                   Tab2[2,i]=EA[2,i]
                   Tab2[3,i]=ECV[2,i]
                   Tab2[4,i]=EB[2,i]
                   Tab2[5,i]=EB6[2,i]
                   Tab2[6,i]=AS[2,i]
   }
```

Cont. función errores.nocond3 (5)

```
  for( i in 1:3){ Tab3[1,i]=ER[3,i]
                  Tab3[2,i]=EA[3,i]
                  Tab3[3,i]=ECV[3,i]
                  Tab3[4,i]=EB[3,i]
                  Tab3[5,i]=EB6[3,i]
                  Tab3[6,i]=AS[3,i]
  }
  return(list(E1=round(Tab1,4),E2=round(Tab2,4),E3=round(Tab3,4)))    }    }
```