Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Estadística

Estimación Robusta y Detección de Atípicos para Datos Tipo Matriz

Alumno: Juan González Arranz
Tutor: Luis Ángel García Escudero
2025

Dedicado a mi familia.

Agradecimientos

Debo dar las gracias a todos los profesores que me han acompañado durante mis estudios, y en especial a mi tutor, Luis Ángel García Escudero, por su paciencia y dedicación en la realización de este trabajo.

Además, quiero agradecer a mis compañeros de clase por su apoyo y compañía durante estos años. Consiguieron hacer más ameno este largo camino. También a la Universidad de Valladolid por darme la oportunidad de formarme y crecer, no solo académicamente, sino también como persona.

Por último, quiero agradecer a mis padres por su apoyo incondicional y por estar siempre a mi lado en los momentos más difíciles de los últimos 5 años en particular, y de toda mi vida en general. Sin su apoyo, no me cabe ninguna duda, nada de esto habría sido posible.

Resumen

En este trabajo se estudia el problema de la estimación robusta y la detección de atípicos en datos tipo matriz. Para ello, se inicia con una breve introducción del algoritmo de estimación MCD para datos p-dimensionales y sus buenas propiedades en situaciones de contaminación con múltiples ejemplos y posibles aplicaciones.

Luego, se explica el concepto de datos matriciales y se estudia el algoritmo MMCD, que permite la aplicación del algoritmo MCD en datos tipo matriz. Se demuestra, además, la eficacia de este algoritmo en la detección de atípicos con un ejemplo práctico en el que se procesa un vídeo.

Para finalizar, se comentan posibles aplicaciones de los conceptos tratados y se explora también una extensión del MMCD, en combinación con el TCLUST, para el agrupamiento de datos de tipo matriz.

Abstract

This work studies the problem of robust estimation and outlier detection in matrix-type data. To this end, it begins with a brief introduction to the MCD estimation algorithm for p-dimensional data and its good properties in contamination scenarios with multiple examples and possible applications.

Then, the concept of matrix data is explained, and the MMCD algorithm, which allows the application of the MCD algorithm to matrix-type data, is studied. The effectiveness of this algorithm in outlier detection is also demonstrated with a practical example in which a video is processed.

Finally, possible applications of the discussed concepts are presented, and an extension of MMCD combined with TCLUST is also explored for the clustering of matrix-type data.

Índice general

Lista de Figuras

Capítulo 1

Introducción

1.1. Contexto y Motivación

En la actualidad, la enorme cantidad y variedad de tipos de datos generados representa un desafío considerable para los científicos de datos. Este volumen masivo y su complejidad creciente no solo abren un amplio abanico de oportunidades para nuevos desarrollos y aplicaciones, sino que también plantean importantes dificultades en términos de capacidad de procesamiento y análisis. Dado que muchos de estos datos están generados por procesos en los que intervienen seres humanos, sujetos a errores, es habitual que estén afectados por ruido, contengan errores o sean incompletos. En otros casos, los datos son producidos de forma automatizada por máquinas cuyos sensores pueden fallar o ser sensibles a interferencias. En cualquier situación, la presencia de datos atípicos es un problema común en el análisis de datos.

La presencia de datos atípicos en un conjunto de datos puede tener consecuencias graves en el análisis de los mismos. La media y la varianza son dos de las medidas estadísticas más utilizadas para resumir un conjunto de datos, y son extremadamente sensibles a la contaminación de las muestras, como se explicará más adelante. Uno de los primeros ejemplos de cómo mejorar la robustez de la media fue la mediana, algo que se estudia en cualquier curso inicial de estadística.

La comunidad científica ha continuado desarrollando multitud de métodos para detectar y tratar estos datos atípicos. Uno de los métodos más utilizados en la actualidad, o por lo menos uno que ha sido ampliamente estudiado, es el estimador MCD (Minimum Covariance Determinant) (Rousseeuw, 1985). Este estimador, además de ser computacionalmente eficiente y sencillo de implementar, es extremadamente robusto frente a la presencia de datos atípicos en los conjuntos de datos.

A su vez, los datos generados por humanos o máquinas no siempre son unidimensionales, sino que pueden ser multidimensionales. En los últimos años esos mismos datos multidimensionales se han convertido en matriciales, como son vídeos, colecciones de imágenes o incluso

observaciones multidimensionales que estén replicadas en q ocasiones para cada sujeto (diferentes localizaciones espaciales, temporales, o condiciones experimentales). La adaptación del estimador MCD a datos matriciales es el problema en el que se centra este trabajo.

1.2. Objetivos

El objetivo principal de este trabajo es estudiar el problema de la estimación robusta y la detección de atípicos en datos de tipo matriz. Para ello, en primer lugar se dará una visión general del MCD y de los datos matriciales, para después estudiar la adaptación del MCD a este tipo de datos y presentar un algoritmo eficiente para su cálculo en el lenguaje de programación R. Por el camino se presentarán multitud de ejemplos y aplicaciones prácticas de estos algoritmos, y se compararán con métodos tradicionales de detección de atípicos.

1.3. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 2 - Algoritmo MCD para datos p-dimensionales: En este capítulo se describirá el estimador MCD, sus propiedades, un algoritmo eficiente para su cálculo y algunas aplicaciones prácticas. Además, al final del capítulo, se comparará su rendimiento en esas mismas aplicaciones frente a otros métodos tradicionales de estimación.

Capítulo 3 - Datos matriciales: Se trata de una corta introducción a los datos matriciales y a las distribuciones típicamente asumidas para este tipo de datos. Se presenta también el algoritmo de estimación mediante *flip-flops* para la estimación de las matrices de covarianzas de un conjunto de datos matriciales, herramienta fundamental para el siguiente capítulo.

Capítulo 4 - MMCD: El capítulo principal. Se presenta el algoritmo MMCD, una adaptación del MCD a datos matriciales. Se describen las ideas principales del algoritmo, las propiedades del estimador MMCD y su algoritmo con las optimizaciones pertinentes para hacer el cálculo computacionalmente eficiente. Se presentan también dos ejemplos: uno con datos simulados y otro con un vídeo, para ilustrar la eficacia de este algoritmo en la detección de atípicos en datos matriciales.

Capítulo 5 - Conclusiones: Además de describir las conclusiones generales del trabajo, se presentan algunas líneas de trabajo futuras que se podrían seguir para continuar con el estudio de estos algoritmos, sus derivados y sus aplicaciones.

Capítulo 6 - Implementación TCLUST: Se presenta una implementación sencilla del TCLUST en R con las modificaciones necesarias para que sea aplicable a datos matriciales, haciendo previamente una breve introducción al TCLUST y sus propiedades.

Apéndice A - Código R: Contiene todo el código escrito en el lenguaje de programación R necesario para reproducir los ejemplos presentados en este trabajo, así como para la implementación de sus algoritmos.

Capítulo 2

Algoritmo MCD para datos p-dimensionales

2.1. Minimum Covariance Determinant

El estimador del Determinante de Mínima Covarianza (MCD, por *Minimum Covariance Determinant*) se diferencia de otros muchos estimadores por ser extremadamente robusto frente a observaciones atípicas para la estimación de la localización y la dispersión de datos multivariantes. Esta resistencia a observaciones atípicas junto con la existencia de un algoritmo computacionalmente muy eficiente lo convierte en un método muy útil y atractivo para multitud de aplicaciones donde se usen datos multivariantes, como por ejemplo en la detección de valores atípicos, en la clasificación de observaciones, o en el agrupamiento de observaciones.

La idea fundamental sobre la que se formula el MCD es la de encontrar el subconjunto de observaciones que minimiza el determinante de la matriz de covarianza calculada solo con las observaciones de ese subconjunto. Otra forma de formularlo y equivalente a la anterior es encontrar el subconjunto de observaciones que maximiza la función de log-verosimilitud de la distribución normal multivariante, como se verá más adelante.

2.1.1. Fundamento teórico

Siendo n el número de observaciones, p el número de variables de un conjunto de datos y reteniendo h observaciones o, alternativamente, recortando una fracción $\alpha = (n-h)/n$ de observaciones con $n/2 \leq h \leq n$. El estimador MCD es la dupla $(\hat{\mu}_0, \hat{\Sigma}_0)$ con:

- $\hat{\mu}_0$ es el vector de medias de las observaciones en el subconjunto de tamaño h que minimiza el determinante de la matriz de covarianza.

- $\hat{\Sigma}_0$ es la matriz de covarianza de las observaciones en el subconjunto de tamaño h que minimiza el determinante de la matriz de covarianza, multiplicada por un factor de consistencia c_0 que será posteriormente introducido.

El parámetro h o el α son los parámetros más importantes en el MCD, al ser los que determinan el tamaño del subconjunto de observaciones que se va a seleccionar, es decir, el número de observaciones *creíbles* a seleccionar. Por las condiciones que se han fijado anteriormente, se puede deducir que es necesario que $h > p$ para poder computar la matriz de covarianza de cualquier h-subconjunto de observaciones. Para evitar ruido excesivo Hubert et al. (2018) recomienda que $n > 5p$. La elección del valor de h afecta enormemente al rendimiento del MCD: cuanto más bajo mayor robustez a costa de pérdida de eficiencia.(Hubert et al., 2018)

El otro parámetro mencionado es el factor de consistencia c_0, necesario para obtener consistencia en la distribución normal. Se puede calcular como:

$$c_0 = \alpha / F_{\chi^2_{p+2}}(q_\alpha)$$

Donde $\alpha = \lim_{n\to\infty} h(n)/n$ y q_α es el α-cuantil de la distribución χ^2_p. Este $c_0 > 1$ hace que la matriz de covarianza de una proporción $1 - \alpha$ de las observaciones, donde se han eliminado observaciones en las colas de la distribución, no subestimen las dispersiones a nivel global de nuestros datos.

En Butler et al. (1993) se ha demostrado la consistencia del estimador *bruto (raw)* MCD tanto para la localización como para la matriz de dispersión en modelos elípticos. En Cator and Lopuhaä (2010) y Cator and Lopuhaä (2012) se demuestra la consistencia y la normalidad asintótica del estimador de la desviación MCD para una gran cantidad de distribuciones.

2.1.2. Demostración de la equivalencia entre la minimización del determinante de la covarianza y la maximización de la log-verosimilitud en el MCD

El estimador MCD se basa en encontrar el subconjunto de observaciones que minimiza el determinante de la matriz de covarianza muestral. Una formulación alternativa, pero equivalente, es la de maximizar la función de log-verosimilitud de una distribución normal multivariante evaluada sobre dicho subconjunto mediante un problema de la siguiente forma donde la función de verosimilitud recortada l se definirá posteriormente:

$$\begin{aligned} \max_{w,\,\mu,\,\Sigma} \quad & l(w, \mu, \Sigma | \mathcal{X}) \\ \text{s.t.} \quad & w_i \in \{0, 1\}, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} w_i = h \end{aligned}$$

Formulación con pesos binarios

Sea $\mathcal{X} = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^p$ nuestro conjunto de datos y $w = (w_1, \ldots, w_n) \in \{0,1\}^n$ un vector de pesos binarios tal que:

$$\sum_{i=1}^{n} w_i = h$$

El subconjunto de tamaño h optimo para el MCD queda determinado por aquellos i tales que $w_i = 1$.

Definimos la media ponderada por

$$\bar{x}_w = \frac{1}{h}\sum_{i=1}^{n} w_i x_i$$

y la matriz de covarianza ponderada por

$$S_w = \frac{1}{h}\sum_{i=1}^{n} w_i (x_i - \bar{x}_w)(x_i - \bar{x}_w)^T$$

Si consideramos la función de log-verosimilitud de una distribución normal multivariante $\mathcal{N}_p(\mu, \Sigma)$, evaluada sobre las observaciones ponderadas

$$\begin{aligned} l(w, \mu, \Sigma | \mathcal{X}) &= \sum_{\{i: w_i = 1\}} \log(\mathcal{N}_p(x_i; \mu, \Sigma)) = \\ &= -\frac{h}{2} p \log(2\pi) - \frac{h}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{n} w_i (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \end{aligned}$$

Si maximizamos esta función respecto a μ y Σ, los mejores μ y Σ en este contexto son (tal como ocurre para el estimador máximo-verosímil de la normal multivariante clásico) obtenidos por

$$\mu = \bar{x}_w \text{ y } \Sigma = S_w$$

Al sustituir estos valores, obtenemos:

$$l(w, \bar{x}_w, S_w | \mathcal{X}) = -\frac{h}{2} p \log(2\pi) - \frac{h}{2}\log|S_w| - \frac{hp}{2}$$

La expresión anterior surge de la propiedad cíclica de la traza, que dice que $\mathrm{tr}(AB) = \mathrm{tr}(BA)$, por lo que

$$\begin{aligned} &\sum_{i=1}^{n} w_i (x_i - \mu)^T S_w^{-1} (x_i - \mu) = \mathrm{tr}\left(\sum_{i=1}^{n} w_i (x_i - \mu)^T S_w^{-1} (x_i - \mu)\right) \\ &= \mathrm{tr}\left(S_w^{-1} \sum_{i=1}^{n} w_i (x_i - \mu)^T (x_i - \mu)\right) = h \cdot \mathrm{tr}(I_p) = hp, \end{aligned}$$

para I_p la matriz identidad en $\mathbb{R}^p$. Por tanto, se tiene

$$l(w, \bar{x}_w, S_w|\mathcal{X}) = \text{constante} - \frac{h}{2} \log |S_w|$$

Por tanto, maximizar la log-verosimilitud recortada es equivalente a minimizar $\log |S_w|$ y, dado que la función logaritmo es monótona creciente, esto es equivalente a minimizar directamente $|S_w|$.

2.1.3. Propiedades del estimador MCD

Robustez y punto de ruptura

Llamamos robustez a la cualidad de un estimador para resistir observaciones atípicas/-desviaciones respecto a las suposiciones asumidas por las técnicas estadísticas sin que afecte a la estimación ni a su calidad.

Una forma de medir la robustez de estimadores es el punto de ruptura *(breakdown value)*: la proporción más pequeña de observaciones en el conjunto de datos que se deben cambiar para afectar al resultado de la estimación. (Rousseeuw and Hubert, 2018). Muchos de los métodos tradicionales se basan en el uso de la media muestral como estimador, que tiene un punto de ruptura de $1/n$; es decir, basta una única observación suficientemente anómala para hacer no operativo cualquier método que se base en esta media muestral. Cualquier observación atípica afecta al resultado. El estimador clásico de desviación típica, $s = \sqrt{\sum_{i=1}^{n} (x_i - \bar{x}^2)/(n-1)}$, sufre de un problema similar: cualquier observación atípica puede afectar muy negativamente al resultado y por tanto su punto de ruptura es 0 %. (Rousseeuw and Hubert, 2018)

Volviendo a los estimadores MCD, el punto de ruptura máximo se obtiene cuando $h = |(n+p+1)/2|$ (Rousseeuw and Hubert, 2018) en el que el punto de ruptura es $(n-p+1)/2$. (Hubert et al., 2018). Este valor es mucho más alto que el $1/n$ comentado anteriormente (incluso en el caso univariante) para la media y la desviación típica y es la razón por la que decimos que el estimador MCD es tan robusto. Un cálculo sencillo permite entender que el punto de ruptura máximo del MCD se alcanza cuando $\alpha = 0.5$, es decir, seleccionando solo un 50 % de las observaciones. Cabría pensar que seleccionar la mitad de las observaciones ($\alpha = 0.5$) es lo ideal, pero en la práctica esto da una muy baja eficiencia (en otras palabras, el estimador tiene una alta varianza).

La elección del valor de h (o de α) es por tanto una tarea muy importante en el MCD, ya que se trata de encontrar un equilibrio entre robustez y eficiencia. Para alcanzar un resultado con una alta robustez y eficiencia se puede aplicar un paso de pesado (Lopuhaä, 1999; Lopuhaä and Rousseeuw, 1991), que consiste en redefinir los dos estimadores de la siguiente forma:

$$\hat{\mu}_{MCD} = \frac{\sum_{i=1}^{n} w(d_i^2) x_i}{\sum_{i=1}^{n} w(d_i^2)}$$

$$\hat{\Sigma}_{MCD} = c_1 \frac{1}{n} \sum_{i=1}^{n} w(d_i^2)(x_i - \hat{\mu}_{MCD})(x_i - \hat{\mu}_{MCD})'$$

Donde $d_i = d(x, \hat{\mu}_0, \hat{\Sigma}_0)$ es la distancia de Mahalanobis definida como:

$$d(x, \hat{\mu}, \hat{\Sigma}) = \sqrt{(x - \hat{\mu})' \hat{\Sigma}^{-1} (x - \hat{\mu})}$$

$w(\cdot)$ es una función de pesado apropiada y la constante c_1 es un factor de consistencia adicional. Típicamente se define w como una función que toma 1 cuando la distancia robusta d_i está por debajo del valor de corte $\sqrt{\chi^2_{p,0.975}}$ y 0 en caso contrario. (Hubert et al., 2018)

$$w(d^2) = I(d^2 \leq \chi^2_{p,0.975})$$

Como apunte adicional, se puede construir una matriz de correlación robusta a partir de la matriz de covarianza robusta del MCD (Hubert et al., 2018) de la siguiente forma:

$$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii} s_{jj}}}$$

Donde r_{ij} es la correlación robusta entre las variables X_i y X_j y s_{ij} es el elemento (i, j) de la matriz de covarianza MCD.

Función de influencia

La función de influencia de un estimador es otra medida de la robustez. Esta función mide el efecto que tiene una fracción infinitesimal de contaminación en el estimador colocados en un punto concreto. (Hubert et al., 2018). La función de influencia del estimador MCD tanto para la variante bruta como la variante con pesos se ha calculado en Cator and Lopuhaä (2012) y Croux and Haesbroeck (1999), llegando a la conclusión de que resulta estar acotada, otra buena propiedad del estimador MCD que indica su robustez. El efecto de una observación atípica cualquiera (o un conjunto limitado de ellas) tiene un efecto limitado en el estimador MCD. Esto contrasta por ejemplo con el estimador clásico de la media que tiene una función de influencia no acotada y puede verse muy afectado a nivel infinitesimal por contaminación puntual.

En particular, para una población normal multivariante "estándar" con media 0 y matriz de covarianzas la matriz identidad la función de influencia del estimador MCD para la localización resulta ser 0 para todas aquellas observaciones x que cumplan $||x||^2 > \chi^2_{p,\alpha}$. (Hubert et al., 2018) Es decir, las observaciones atípicas lejanas ni siquiera influyen en el estimador. En cuanto al estimador de la desviación, para los elementos no diagonales de la matriz de covarianza ocurre lo mismo, y para los elementos en la diagonal la función de influencia es constante cuando $||x||^2$ es suficientemente grande, manteniendo el carácter acotado de la función de influencia para el estimador MCD.

Equivarianza afín

Esta propiedad del estimador MCD tanto para la localización como para la desviación indica que para cada matriz no singular cuadrada de dimensión p que llamaremos A, y

cualquier vector columna de dimensión p que llamaremos b, se cumple que:

$$\hat{\mu}_M CD(XA' + 1_n b') = \hat{\mu}_M CD(X)A' + b$$

$$\hat{\Sigma}_M CD(XA' + 1_n b') = A\hat{\Sigma}_M CD(X)A'$$

Siendo 1_n un vector de la forma $(1, 1, ..., 1)'$ de dimensión n. La demostración de esta propiedad (Hubert et al., 2018) se basa en el hecho de que para cada subconjunto H de $(1, 2, ...)$ de tamaño h y su conjunto de datos correspondiente X_H, el determinante de la matriz de covarianza de los datos transformados es igual a:

$$|S(X_H A)| = |AS(X_H)A'| = |A|^2 |S(X_H)|$$

Por lo que transformar un h-subconjunto de determinante mínimo da un h-subconjunto $X_H A'$ de determinante mínimo entre todos los h-subconjuntos de los datos transformados XA' y su matriz de covarianza se transforma de la misma forma. La equivarianza afín del estimador bruto MCD para la localización se da por la equivarianza de la media muestral.

Como detalle adicional, las distancias entre observaciones calculadas mediante los estimadores MCD *(distancias robustas)* son invariantes afines: se mantienen iguales independientemente de las transformaciones de los datos. Esto implica que el estimador MCD con pesos es equivariante afín también.

Esta propiedad, en resumen, significa que los datos se pueden rotar, trasladar y escalar sin peligro de afectar al diagnóstico de valores atípicos.

2.2. Algoritmo FastMCD

Todas las ventajas explicadas hasta el momento del estimador MCD se ven empañadas por un gran problema: el estimador MCD exacto es computacionalmente muy costoso al requerir evaluar $\binom{n}{h}$ subconjuntos de observaciones de tamaño h. Este problema es computacionalmente prohibitivo para tamaños de muestra n incluso bastante pequeños. Con el fin de solventar este problema se han propuesto varios algoritmos para calcular el estimador MCD de forma más eficiente, siendo el algoritmo FastMCD propuesto en Rousseeuw and Driessen (1999) uno de los más populares y eficientes, pese a garantizar únicamente una solución aproximada. Este algoritmo será el que se use a lo largo de este documento para calcular el estimador MCD.

2.2.1. C-Step

La parte fundamental del algoritmo FastMCD es el algoritmo C-Step *(Concentration Step o Paso de Concentración).*

Algoritmo C-Step (Rousseeuw and Driessen, 1999)

Dados los estimadores iniciales $\hat{\mu}_{\text{old}}$ y $\hat{\Sigma}_{\text{old}}$ de un conjunto inicial H_{old} de tamaño h:

1. Computar las distancias $d_{\text{old}}(i) = d(x_i, \hat{\mu}_{\text{old}}, \hat{\Sigma}_{\text{old}})$ para cada $i = 1, 2, ..., n$.
2. Ordenar las distancias de menor a mayor, obteniendo una permutación π tal que $d_{\text{old}}(\pi(1)) \leq d_{\text{old}}(\pi(2)) \leq ... \leq d_{\text{old}}(\pi(n))$.
3. Definir el conjunto $H_{\text{new}} = \{\pi(1), \pi(2), ..., \pi(h)\}$.
4. Computar los nuevos estimadores:
   - $\hat{\mu}_{\text{new}} = \sum_{i \in H_{\text{new}}} \frac{x_i}{h}$
   - $\hat{\Sigma}_{\text{new}} = \sum_{i \in H_{\text{new}}} \frac{(x_i - \hat{\mu}_{\text{new}})(x_i - \hat{\mu}_{\text{new}})'}{h-1}$

En Rousseeuw and Driessen (1999) se prueba que $\det(\hat{\Sigma}_{\text{new}}) \leq \det(\hat{\Sigma}_{\text{old}})$ con igualdad solo si $\hat{\Sigma}_{\text{old}} = \hat{\Sigma}_{\text{new}}$. Por tanto, iterando el algoritmo C-Step se garantiza que la secuencia de determinantes de las matrices de covarianza de los subconjuntos es decreciente y converge en un número finito de pasos al menos a un mínimo local de la función objetivo.

En la práctica esta convergencia se da de forma rápida, aunque no se garantiza que el estimador final $\hat{\Sigma}_{\text{new}}$ sea el mínimo global de la función objetivo MCD porque la función objetivo no necesariamente tiene un único mínimo absoluto y, de hecho, suele presentar varios mínimos locales. Con el fin de tratar de aumentar la probabilidad de encontrar el mínimo global el algoritmo FastMCD propone realizar varias ejecuciones del algoritmo C-Step con distintos conjuntos iniciales H aleatorios y quedarse con el mejor conjunto final de entre todas las ejecuciones.

2.2.2. Construcción del conjunto inicial

La elección del conjunto inicial H es un paso crucial en el algoritmo FastMCD porque debe garantizar la máxima aleatoriedad posible para aumentar las probabilidades de encontrar el mínimo global. La primera alternativa que se podría proponer sería obtener directamente unos subconjuntos aleatorios de tamaño h de entre todas las observaciones. Sin embargo, esta alternativa no se usa en la práctica por tener una alta probabilidad de seleccionar observaciones atípicas en el conjunto inicial. (Hubert et al., 2018)

El método empleado por FastMCD funciona de la siguiente forma:

Construcción del conjunto inicial (Hubert et al., 2018)

1. Construir un conjunto inicial J de tamaño $p+1$ de observaciones seleccionadas aleatoriamente, siendo p el número de variables.
2. Computar la media y covarianza de las observaciones en J, obteniendo $\hat{\mu}_0$ y $\hat{\Sigma}_0$.
3. Computar las distancias $d_0^2(i) := d^2(x_i, \hat{\mu}_0, \hat{\Sigma}_0)$ para cada $i = 1, 2, ..., n$.
4. Ordenar las distancias.
5. Construir el conjunto inicial H_1 con las h observaciones con menor distancia d_0.

Este método tiene una menor probabilidad de seleccionar observaciones atípicas frente a comenzar directamente con h observaciones. Esos conjuntos de tamaño $p + 1$ (mínimo para poder calcular una matriz de covarianzas en problemas p-dimensionales) se denominan "conjuntos elementales".

2.2.3. Optimizaciones del algoritmo FastMCD

El algoritmo FastMCD va más allá de limitarse a ejecutar el algoritmo C-Step varias veces con distintos conjuntos iniciales. En Hubert et al. (2018) se revisan las principales optimizaciones que aplica el algoritmo FastMCD para mejorar su eficiencia:

- **Reducción de C-Steps:** Dado que cada paso C-Step requiere el cálculo de la matriz de covarianza de un subconjunto de observaciones, su determinante y las distancias correspondientes, reducir el número de C-Steps incrementa sustancialmente la eficiencia del algoritmo.
  Resulta que muchas de las iteraciones que llevan al mínimo global normalmente tienen un determinante de la matriz de covarianza muy pequeño con el segundo C-Step. Por tanto, el algoritmo FastMCD aplica únicamente dos C-Steps a cada conjunto inicial y selecciona los 10 subconjuntos con menor determinante para continuar con el proceso. Solamente se iterará hasta la convergencia sobre esos 10 subconjuntos.
- **Partición del Conjunto de Datos:** El cálculo sobre matrices es costoso y los conjuntos de datos enormes pueden hacer que el algoritmo sea muy lento. Si el valor de n es muy alto FastMCD particiona el conjunto de datos, operando por separado en cada partición y combinando los resultados al final.

2.3. Ejemplo de aplicación

Este ejemplo usará datos simulados con el fin de mostrar el funcionamiento del estimador MCD y su robustez frente a observaciones atípicas. Para ello se usará el paquete **robustbase** (Maechler et al., 2024) de R, que implementa el algoritmo FastMCD. El código usado para este ejemplo se encuentra en el Apéndice A.1.1.

En un primer lugar, se generan 500 observaciones de una distribución normal multivariante de dimensión $p = 10$ con media:

$$\mu = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Y matriz de covarianza:

$$\Sigma = \begin{pmatrix} 1.5 & 1 & \dots & 1 \\ 1 & 1.5 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1.5 \end{pmatrix}$$

Luego, se añaden 50 observaciones como contaminación generadas de la misma forma pero con la única diferencia siendo que su media pasa a ser:

$$\mu = \begin{pmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \end{pmatrix}$$

La matriz de covarianza para la contaminación es la misma que para las observaciones normales.

Finalmente, se juntan estas observaciones en un único conjunto de datos de 550 observaciones en las que se sabe que las 50 últimas son las contaminadas. Con estos datos, se realiza la estimación de la media y la matriz de covarianza usando el estimador MCD y los estimadores tradicionales y posteriormente se calculan las distancias de Mahalanobis respecto al centro de la muestra con ambos tipos de estimadores. Los resultados se muestran en el siguiente gráfico:

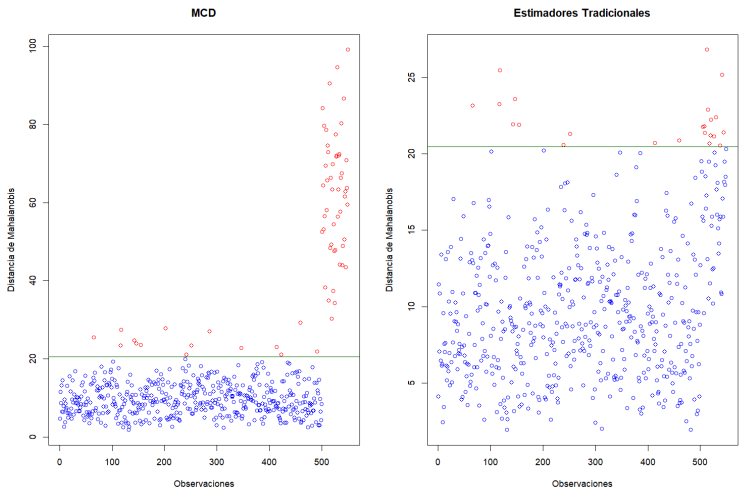


Figura 2.1: Distancias de Mahalanobis al cuadrado con MCD y Método Clásico

Se puede observar que el estimador MCD es capaz de detectar sin problema el conjunto de observaciones de contaminación (las 50 observaciones más a la derecha en ambos gráficos)

y las clasifica como observaciones atípicas. Los estimadores tradicionales modifican por completo su resultado e imposibilitan la detección de estas observaciones atípicas, que quedan mayoritariamente clasificadas como observaciones normales. Esto es un claro ejemplo de la robustez del estimador MCD frente a observaciones atípicas.

La línea verde en ambos gráficos marca el límite de tolerancia del 97.5 % para las distancias de Mahalanobis, es decir, las observaciones que cumplen lo siguiente pueden clasificarse seguramente como atípicas:

$$\mathrm{MD}^2(X_i, \hat{M}, \hat{\Sigma}) > \chi^2_{p;0.975}$$

2.4. Otros métodos relacionados con el MCD

A continuación se mostrarán métodos muy relacionados con el MCD, usando paquetes de R fácilmente disponibles. Como detalle, la fracción de observaciones que forman parte de la muestra seleccionada por el método MCD, α, suele ser en estos paquetes 0.75 o 0.5. En los ejemplos a continuación se usará el valor de α por defecto de cada uno.

2.4.1. Regresión Lineal

Un modelo de regresión lineal asume que existen una o varias variables explicativas que se usan para predecir una variable respuesta, con la siguiente fórmula:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon_i \text{ para } i = 1, 2, ..., n$$

siendo los coeficientes β_j los parámetros del modelo y ϵ_i los términos de error. Los errores se presuponen como independientes y siguiendo una distribución normal con media 0 y varianza σ^2. El método clásico de estimación de los coeficientes β_j es el de mínimos cuadrados (LS) que minimiza la suma de los cuadrados de los residuos. Sin embargo, este método es muy sensible a observaciones atípicas y puede dar lugar a estimaciones sesgadas. (Rousseeuw and Leroy, 1987)

Un método alternativo de estimación de estos coeficientes y que sin embargo es robusto a observaciones atípicas es el de Mínimos Cuadrados Recortados (Least Trimmed Squares, LTS) propuesto en Rousseeuw (1984). Este método selecciona un subconjunto de observaciones de tamaño h que minimiza la suma de los cuadrados de sus residuos y calcula los coeficientes β sobre este subconjunto. En otras palabras, el problema pasa a ser:

$$\min_{\beta} \sum_{i=1}^{h} r^2(\pi(i))$$

donde π es una permutación tal que $r^2(\pi(1)) \leq r^2(\pi(2)) \leq ... \leq r^2(\pi(n))$.

Las ideas del MCD se aplican en este nuevo método de estimación de la misma forma que en el MCD: todo radica en la elección del subconjunto y de su tamaño h, que será siempre

mayor al 50 % de las observaciones. La desviación típica de los residuos se puede estimar como:

$$\hat{\sigma}_{LTS} = c^2_{h,n} \sum_{i=1}^{h} r^2(\pi(i))/h$$

donde $c^2_{h,n}$ es una constante que hace que este estimador sea consistente en distribuciones de error gaussianas. Este estimador $\hat{\sigma}_{LTS}$ es importante puesto que se usa tanto para calcular los residuos estandarizados $(r_i/\hat{\sigma}_{LTS})$ como para aplicar un paso de pesado a cada observación con estos residuos estandarizados y mejorar la eficiencia del estimador LTS.

El código en R presente en el Apéndice A.1.2 y cuyo resultado se muestra a continuación es un ejemplo sencillo de cómo aplicar el método LTS mejora la regresión lineal clásica ofreciendo una mayor resistencia a observaciones atípicas. Para ello se usa la librería **robustbase** (Maechler et al., 2024) y su conjunto de datos `starsCYG`, al que se le han modificado algunas observaciones para que sea un mejor ejemplo.

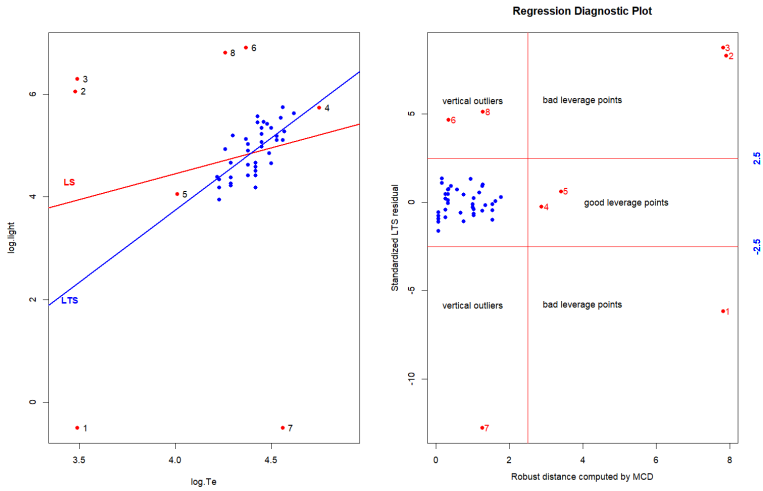


Figura 2.2: Regresión clásica vs. LTS y gráfico de residuos de LTS

Como se puede ver en la Figura 2.2, la regresión LTS es capaz de ajustarse a los datos de forma más precisa que la regresión clásica, que se ve afectada por la presencia de observaciones atípicas.

A fin de comparar los resultados de la regresión LTS con los de la regresión clásica, se ha realizado un análisis de residuos estandarizados para ambas regresiones. En la Figura 2.3 se muestran los residuos estandarizados de ambos modelos, y se puede visualizar que la regresión LTS identifica y clasifica mejor las observaciones atípicas. En particular, los atípicos 6, 8 y 4 ni siquiera se marcan como atípicos, y el atípico 2 pasa de ser *bad leverage point* en la regresión LTS a *good leverage point* en la regresión clásica. Es decir, la regresión clásica está tan influenciada por esta observación que pasa a considerarse positiva para el ajuste del modelo. Esto es un claro ejemplo de cómo el método LTS es más robusto que el método

clásico, y de cómo el MCD y sus ideas principales pueden ser utilizados para mejorar la regresión lineal clásica y analizar observaciones atípicas (Rousseeuw and Hubert, 2018).

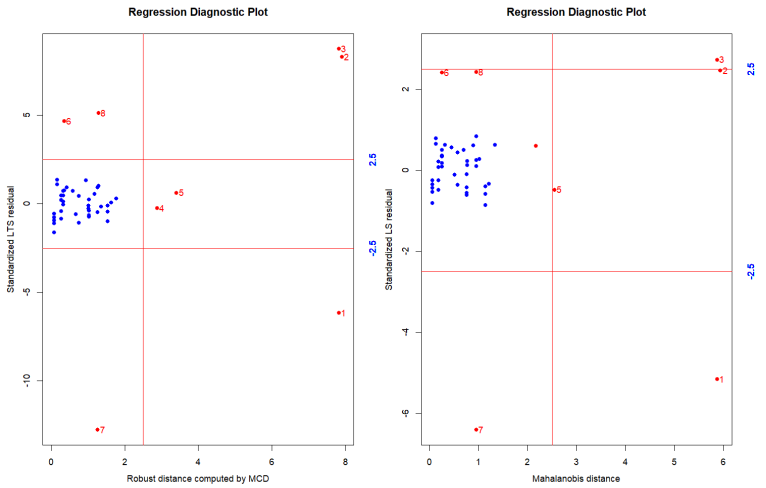


Figura 2.3: Gráficos de residuos de LTS y LS

2.4.2. Análisis de Componentes Principales

El análisis de componentes principales (*Principal Component Analysis* o PCA) es un método de reducción de dimensiones que trata de explicar la estructura de la covarianza de un conjunto de datos en un número pequeño de componentes. El primer componente principal es la dirección en la que la varianza de los datos es máxima, el segundo componente principal es la dirección ortogonal al primero y que maximiza nuevamente la varianza, y así sucesivamente. (Peña, 2002) El enfoque clásico de PCA usa la matriz de covarianza común para calcular los componentes principales, lo que lo hace muy sensible a observaciones atípicas, como ya se ha mencionado anteriormente. Con el fin de mitigar este problema se han propuesto multitud de soluciones, siendo una de ellas el uso del estimador MCD en lugar de la matriz de covarianza habitual.

El ejemplo a continuación usa el conjunto de datos `longley`, con 7 variables. Se comparará el resultado de aplicar PCA con la matriz de covarianza clásica y con la matriz de covarianza robusta MCD. Para el cálculo de la matriz de covarianza MCD se ha usado la librería **rrcov** (Todorov and Filzmoser, 2009) y para el diagnóstico la librería **chemometrics** (Filzmoser, 2023). El código está en el Apéndice A.1.3.

Como se puede ver en la Figura 2.4, el PCA con la matriz de covarianza MCD detecta un mayor número de observaciones distantes al resto. En ambos casos no hay ni *good leverage points* ni *bad leverage points*, pero aún así este análisis sirve para poder señalar estas observaciones extraordinarias y analizarlas con más detalle después teniendo en cuenta que aunque sean observaciones lejanas están bien modelizadas.

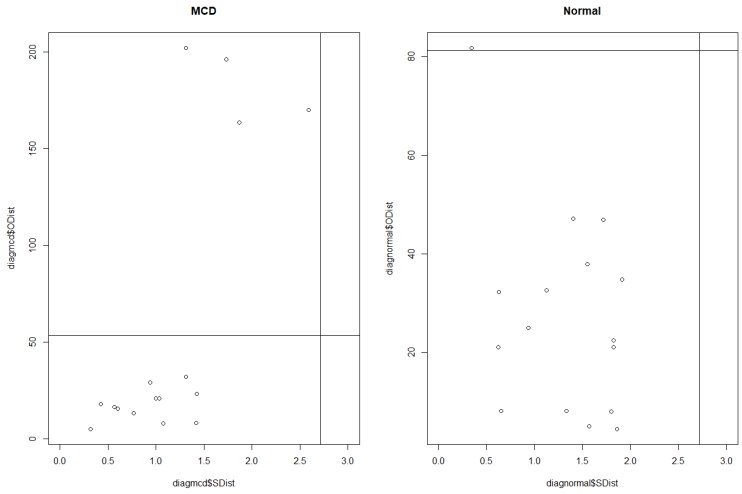


Figura 2.4: Mapa de outliers para los dos PCA

Este método tiene la principal limitación de necesitar que el número de observaciones sea mayor que el número de variables, por lo que otros métodos como el *Projection Pursuit* o el *ROBPCA*, un híbrido entre los otros dos métodos, han sido propuestos para superar esta limitación (Rousseeuw and Hubert, 2018).

2.4.3. Clasificación (Análisis de Discriminante Lineal)

La clasificación de observaciones es una tarea común en Estadística y trata de asignar una observación a una de varias clases posibles. El análisis de discriminante lineal (LDA) es un método de clasificación que busca obtener una función lineal que separe cada par de clases de observaciones, no solo con el fin de clasificar bien las observaciones conocidas sino de servir como clasificador de nuevas observaciones. El LDA utiliza la matriz de covarianza muestral, pero ahora es posible cambiar esta matriz de covarianza por la estimada por el MCD con el fin de aumentar la robustez de este método. (Rousseeuw and Hubert, 2018)

El código en el Apéndice A.1.4 muestra un ejemplo de cómo aplicar el LDA con la matriz de covarianza tradicional y con la matriz de covarianza MCD en el conjunto de datos `pottery` de la librería **MASS** (Venables and Ripley, 2002) (aunque se modifica ligeramente en este ejemplo acentuando algunos atípicos con el fin de mostrar mejor las ventajas del MCD). Para poder extraer un gráfico en dos dimensiones se usarán solo como variables explicativas el contenido de Calcio y de Magnesio, y como variable respuesta el origen de la cerámica, con dos clases posibles: *Attic* o *Eritrean*. Para el LDA con el método MCD se ha usado la librería **rrcov** (Todorov and Filzmoser, 2009) y su función *Linda*.

Como se puede observar en la Figura 2.5, el LDA con MCD clasifica mejor y es capaz de definir las fronteras de ambas clases (*Attic* en rojo, *Eritrean* en verde) de una forma mucho más precisa y menos influenciada por las observaciones atípicas, que con ambos métodos

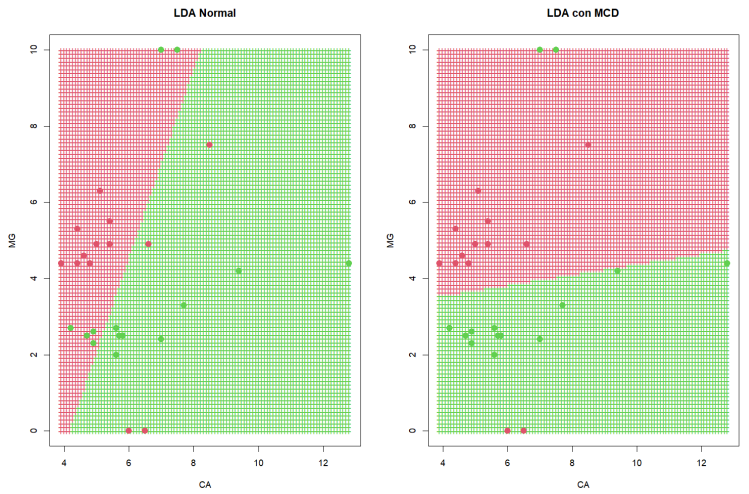


Figura 2.5: Comparación de LDA con matriz de covarianza clásica y MCD

son mal clasificadas. En caso de disponer de nuevas observaciones es de esperar que el LDA con MCD se comporte mejor por haber definido la frontera con menos influencia de estas observaciones extraordinarias.

2.4.4. Clustering (k-medias recortadas)

El clustering consiste en el agrupamiento de observaciones en grupos o clusters en base a su similitud sin conocimiento previo de esas clases a las que podrían pertenecer. Uno de los algoritmos más populares es *k-medias*, que busca agrupar las observaciones en k grupos minimizando la suma de los cuadrados de las distancias de cada observación al centroide (media) de su grupo. (Peña, 2002) Su dependencia de la media lo hace ser poco robusto, así que se han propuesto multitud de alternativas. Una de ellas es el *k-medias recortadas* introducido en Cuesta-Albertos et al. (1997), inspirado en el MCD y el LTS comentados anteriormente. Este método selecciona un subconjunto de observaciones de tamaño h que minimiza la suma de los cuadrados de las distancias de las observaciones al centroide de su grupo. Uno de los efectos colaterales de este método es que habrá observaciones que no pertenezcan a ningún grupo (concretamente hasta $n - h$ observaciones no son asignadas a ningún grupo) lo que facilita automáticamente una forma de detectar observaciones atípicas.

El código del Apéndice A.1.5 muestra un ejemplo aplicando al conjunto de datos `geyser2` tanto el algoritmo k-medias clásico como el k-medias recortadas de la librería **tclust**. (Fritz et al., 2012)

Como se puede ver en la Figura 2.6, el k-medias recortadas detecta los grupos sin ser afectado por las seis observaciones atípicas en la esquina inferior izquierda, a diferencia del k-medias clásico. Nótese que todas aquellas observaciones que no parecen asignarse a ningún grupo se han dejado sin clasificar (circunferencias negras).

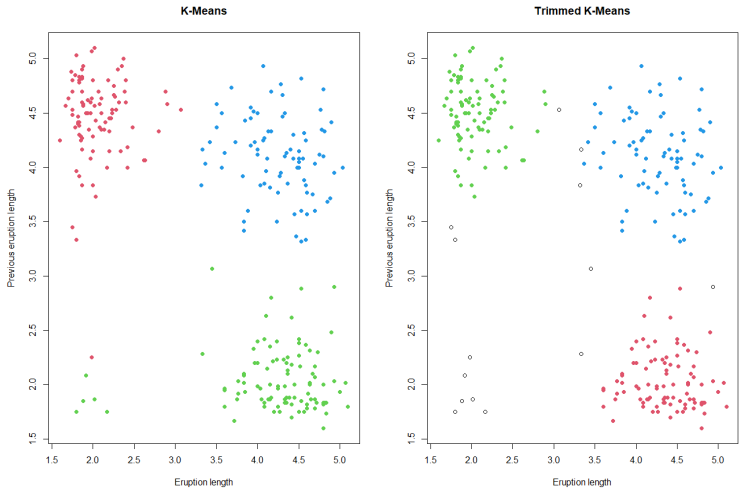


Figura 2.6: Comparación de k-medias y k-medias recortadas

Capítulo 3

Datos matriciales

3.1. Introducción

Hasta ahora se ha trabajado con distribuciones univariantes o multivariantes en este trabajo. Sin embargo, en la estadística actual es común trabajar con datos matriciales, es decir, distribuciones en las que cada observación es una matriz. Ejemplos de este tipo de uso serían n imágenes de resolución $p \times q$ o n experimentos en los que se miden p variables en q réplicas. El enfoque tradicional ha sido transformar estas matrices en vectores de dimensión pq y trabajar con métodos multivariantes para vectores tradicionales. Se debe tener en cuenta que esta vectorización da lugar a n observaciones de vectores de dimensión muy alta, lo que puede llevar a problemas en múltiples métodos estadísticos.

Un nuevo enfoque es modelar las n observaciones como matrices y asumir que se originan de una distribución matriz-variante. En general se suele trabajar con la familia de distribuciones matriz-elípticas. Esta familia de distribuciones se trata de una clase de distribuciones parametrizada por:

- La matriz de medias: $\mathbf{M} \in \mathbb{R}^{p \times q}$.
- Covarianza de las filas: $\Sigma^{row} \in PDS(p)$.
- Covarianza de las columnas: $\Sigma^{col} \in PDS(q)$.
- Función generadora de la densidad: $g : [0, \infty) \to \mathbb{R}$

Donde $PDS(a)$ con $a \in \mathbb{N}$ denota la clase de todas las matrices $a \times a$ simétricas y definidas positivas.

De forma más general, una matriz aleatoria X con distribución absolutamente continua sigue una distribución matriz-elíptica $\mathcal{ME}(\mathbf{M}, \Sigma^{row}, \Sigma^{col}, g)$ si su densidad es de la forma:

$$f(X) = det(\Sigma^{row})^{-\frac{q}{2}} det(\Sigma^{col})^{-\frac{p}{2}} g(tr(\Omega^{col}(X - \mathbf{M})'\Omega^{row}(X - \mathbf{M})))$$

con $\Omega^{row} = (\Sigma^{row})^{-1}$ y $\Omega^{col} = (\Sigma^{col})^{-1}$, llamadas *matrices de precisión.*

Una característica importante de las distribuciones matriz-elípticas es que se pueden relacionar fácilmente con sus homólogas multivariantes. Se cumple la siguiente relación entre observaciones matriciales y las mismas observaciones vectorizadas:

$$X \sim \mathcal{ME}(\mathbf{M}, \Sigma^{row}, \Sigma^{col}, g) \Leftrightarrow vec(X) \sim \mathcal{E}(\mathbf{M}, \Sigma^{col} \otimes \Sigma^{row}, g)$$

siendo $\otimes$ el producto de Kronecker definido como:

Producto de Kronecker

Sea A una matriz de dimensión $p \times q$ y B una matriz de dimensión $r \times s$. Entonces el producto de Kronecker se define como:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1q}B \\ \vdots & \ddots & \vdots \\ a_{p1}B & \cdots & a_{pq}B \end{bmatrix}$$

Con a_{ij} el elemento de la fila i y columna j de la matriz A.

3.2. Distribución normal matricial

Al igual que en el caso univariante y multivariante, la distribución normal es la distribución más relevante y estudiada. En el caso matricial, se tiene la distribución normal matricial $\mathcal{MN}(\mathbf{M}, \Sigma^{row}, \Sigma^{col})$ con densidad:

$$f(X|\mathbf{M}, \Sigma^{row}, \Sigma^{col}) = \frac{exp(-\frac{1}{2}tr(\Omega^{col}(X-\mathbf{M})'\Omega^{row}(X-\mathbf{M})))}{(2\pi)^{\frac{pq}{2}} det(\Sigma^{col})^{\frac{p}{2}} det(\Sigma^{row})^{\frac{q}{2}}}$$

La estimación de los parámetros normalmente requeriría vectorizar las observaciones correspondientes y estimar un vector de medias y una matriz de covarianzas de dimensión pq, potencialmente muy grande. Sin embargo, se puede usar la relación mostrada en la sección anterior y aprovechar el producto de Kronecker para simplificar la estimación. De esta forma, se pueden estimar las matrices de covarianzas de filas y columnas por separado y con dimensión mucho menor ($p \times p$ y $q \times q$ respectivamente). El estimador de máxima verosimilitud para cada uno de los parámetros satisface que (Dutilleul, 1999):

$$\hat{\mathbf{M}} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

$$\hat{\Sigma}^{row} = \frac{1}{qn}\sum_{i=1}^{n}(X_i - \hat{\mathbf{M}})(\hat{\Omega}^{col})(X_i - \hat{\mathbf{M}})'$$

$$\hat{\Sigma}^{col} = \frac{1}{pn}\sum_{i=1}^{n}(X_i - \hat{\mathbf{M}})'(\hat{\Omega}^{row})(X_i - \hat{\mathbf{M}})$$

En Soloveychik and Trushin (2016) se demuestra que para n observaciones independientes igualmente distribuidas de una distribución continua matriz-variante $p \times q$ no hay un máximo único de la función de verosimilitud matriz-normal si $n < \text{máx}(\frac{p}{q}, \frac{q}{p})$ y que existe un máximo único casi seguro si $n \geq \lfloor \frac{p}{q} + \frac{q}{p} \rfloor + 2$. Obtener los estimadores de máxima verosimilitud para las matrices de covarianza en la práctica es un problema complicado de resolver, así que Dutilleul (1999) propuso un algoritmo de estimación iterativo (*flip-flop*) en el que se alterna entre el cálculo de las matrices de covarianza de filas y columnas. El algoritmo funciona de forma que converge al máximo único desde cualquier inicialización simétrica definida positiva de cualquiera de las dos matrices de covarianzas, siempre que $n \geq \lfloor \frac{p}{q} + \frac{q}{p} \rfloor + 2$. (Soloveychik and Trushin, 2016)

3.3. Algoritmo de estimación mediante flip-flops

El algoritmo descrito en Dutilleul (1999) es sencillo de implementar. A continuación se realizará una demostración del mismo en R. Para la generación de los datos se ha usado la librería *MixMatrix* que contiene una función para generar matrices aleatorias con distribución normal matricial. (Thompson, 2024) Esta librería contiene también funciones para la estimación de los parámetros de la distribución normal matricial, pero no se usarán con el fin de demostrar la facilidad de implementación del algoritmo flip-flop.

El primer paso es cargar la librería y generar los datos. Se generan 100000 observaciones de una distribución normal matricial de dimensiones $p = 3$ y $q = 2$ con media:

$$\mathbf{M} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Matriz de covarianza de filas:

$$\Sigma^{row} = \begin{bmatrix} 8 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 8 \end{bmatrix}$$

Y matriz de covarianza de columnas:

$$\Sigma^{col} = \begin{bmatrix} 8 & 0.5 \\ 0.5 & 0.18 \end{bmatrix}$$

El código en R usado para este capítulo se encuentra en el Apéndice A.2.

La estimación de la media se pude realizar con la función *apply* de R, mientras que para la estimación de las matrices de covarianza se implementa el algoritmo *flip-flop*. El código que se presenta en el apéndice está bastante optimizado para mejorar su eficiencia en vistas a su uso intensivo en capítulos posteriores, pero aún así se ha intentado mantener su claridad.

El algoritmo se ejecuta hasta que la norma de la diferencia entre las matrices de covarianza de dos iteraciones consecutivas sea menor que 10^{-9}.

Tras la ejecución de estos pasos se obtienen los siguientes resultados:

$$\hat{\Sigma}^{row} = \begin{bmatrix} 1.0015233 & 0.124166 & 0.1227287 \\ 0.1241660 & 1.002148 & 0.1227710 \\ 0.1227287 & 0.122771 & 0.9963304 \end{bmatrix} \text{ y } \hat{\Sigma}^{col} = \begin{bmatrix} 63.827634 & 3.995207 \\ 3.995207 & 1.436552 \end{bmatrix}$$

Queda claro que estas matrices no se parecen a los parámetros originales pero, tal y como se explica en Dutilleul (1999), la función generadora de la distribución normal multivariante no se ve afectada si se cambia Σ^{row} por $a\Sigma^{row}$ y Σ^{col} por $\frac{1}{a}\Sigma^{col}$ con $a > 0$. En este caso controlado es posible calcular esa constante a y comprobar que los parámetros estimados son correctos y eso es precisamente lo que se hace en las últimas líneas del código. Con una constante $a = 0.1251904$ los parámetros estimados son mucho más cercanos a los parámetros originales:

$$\hat{\Sigma}^{row} = \begin{bmatrix} 8.0000000 & 0.9918173 & 0.9803366 \\ 0.9918173 & 8.0049888 & 0.9806743 \\ 0.9803366 & 0.9806743 & 7.9585202 \end{bmatrix} \text{ y } \hat{\Sigma}^{col} = \begin{bmatrix} 7.9906075 & 0.5001616 \\ 0.5001616 & 0.1798426 \end{bmatrix}$$

El algoritmo de estimación mediante *flip-flops*, por su sencillez y buen funcionamiento, será la base para la implementación del algoritmo MMCD en el siguiente capítulo.

Capítulo 4

MMCD

4.1. Planteamiento del MMCD

Explicado ya el algoritmo general MCD y el concepto de los datos tipo matriz, solo queda unir los dos conceptos en el MMCD *(Matrix Minimum Covariance Determinant)*. El MMCD permite obtener equivalentes a los estimadores definidos en el capítulo anterior pero con una mayor robustez.

El MMCD se fundamenta en la maximización de la función de log-verosimilitud basada en la distribución matriz-normal. Al igual que en el MCD, se busca identificar un subconjunto de h observaciones ($n/2 \leq h \leq n$) con el menor determinante de la matriz de covarianzas; o equivalentemente, con la mayor log-verosimilitud. (Mayrhofer et al., 2025)

4.1.1. Función de log-verosimilitud

Usando unos pesos $w = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n$ para una muestra $\mathfrak{X} = (X_1, \ldots, X_n)$ de tamaño n independiente e idénticamente distribuida según una $\mathcal{MN}(M, \Sigma^{row}, \Sigma^{col})$ se puede formular la función con pesos de la log-verosimilitud, $l(w, M, \Sigma^{row}, \Sigma^{col}|\mathfrak{X})$, definida como:

$$-\frac{1}{2}\sum_{i=1}^{n} w_i \left(p\ln(\det(\Sigma^{col})) + q\ln(\det(\Sigma^{row})) + \mathrm{MMD}^2(X) + pq\ln(2\pi)\right)$$

donde $\mathrm{MMD}^2(X)$ es la distancia de Mahalanobis matricial al cuadrado definida como:

$$\mathrm{MMD}^2(X) = tr(\Omega^{col}(X-M)'\Omega^{row}(X-M)) \text{ siendo } \Omega^{col} = (\Sigma^{col})^{-1} \text{ y } \Omega^{row} = (\Sigma^{row})^{-1}$$

Los pesos a escoger serán binarios, $w_i \in \{0, 1\}$, con la restricción de que $\sum_{i=1}^{n} w_i = h$. La tarea entonces a realizar en el MMCD es encontrar las observaciones $H \subset (1, \ldots, n)$ con

$|H| = h$ en las que $w_i = 1$ si $i \in H$ y $w_i = 0$ si $i \notin H$. El problema de optimización se puede reescribir como:

$$\begin{aligned} \max_{w,\, M,\, \Sigma^{row},\, \Sigma^{col}} \quad & l(w, M, \Sigma^{row}, \Sigma^{col} | \mathfrak{X}) \\ \text{s.t.} \quad & w_i \in \{0,1\}, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} w_i = h \end{aligned}$$

4.1.2. Estimadores MMCD

Definido el problema de optimización, se define un problema equivalente al anterior pero con mayor utilidad, al presentar directamente los estimadores.

Estimadores MMCD (Mayrhofer et al., 2025)

Dada una muestra $\mathfrak{X} = (X_1, \ldots, X_n)$, con $n/2 \leq h \leq n$ y $h \geq \lfloor p/q + q/p \rfloor + 2$ independiente e idénticamente distribuida según una $\mathcal{MN}(M, \Sigma^{row}, \Sigma^{col})$, la obtención de un máximo de la función de log-verosimilitud con pesos es equivalente a la minimización de:

$$\ln(\det(\hat{\Sigma}_H^{col} \otimes \hat{\Sigma}_H^{row})) = p \ln(\det(\hat{\Sigma}_H^{col})) + q \ln(\det(\hat{\Sigma}_H^{row}))$$

a lo largo de todos los subconjuntos $H \subset \{1, \ldots, n\}$ con $|H| = h$. Se pueden obtener los estimadores MMCD de la siguiente manera:

- $\hat{M}_H = \frac{1}{h} \sum_{i \in H} X_i$
- $\hat{\Sigma}_H^{row} = \frac{1}{qh} \sum_{i \in H} (X_i - \hat{M}_H) \hat{\Omega}_H^{col} (X_i - \hat{M}_H)'$
- $\hat{\Sigma}_H^{col} = \frac{1}{ph} \sum_{i \in H} (X_i - \hat{M}_H)' \hat{\Omega}_H^{row} (X_i - \hat{M}_H)$

Donde $\hat{\Omega}_H^{col} = (\hat{\Sigma}_H^{col})^{-1}$ y $\hat{\Omega}_H^{row} = (\hat{\Sigma}_H^{row})^{-1}$.

Como apunte final, los estimadores *brutos (raw)* MMCD se definen como:

$$(\hat{M}_{H*}, \hat{\Sigma}_{H*}^{row}, \hat{\Sigma}_{H*}^{col}) = \underset{\substack{\hat{M}_H, \hat{\Sigma}_H^{row}, \hat{\Sigma}_H^{col} \\ H \subset \{1,\ldots,n\}, |H| = h}}{\arg\max} \quad p \ln(\det(\hat{\Sigma}_H^{col})) + q \ln(\det(\hat{\Sigma}_H^{row}))$$

4.2. Propiedades de los estimadores MMCD

A continuación se presentan las propiedades de los estimadores MMCD siguiendo Mayrhofer et al. (2025). Las demostraciones no se incluyen en este documento, pero se pueden encontrar en el artículo original.

4.2.1. Equivarianza Matriz-Afín

Suponiendo una matriz aleatoria $X \sim \mathcal{ME}(M, \Sigma^{row}, \Sigma^{col}, g)$, funciones lineales de ésta siguen también una distribución matriz-elíptica. Esto implica que para matrices constantes como $A \in \mathbb{R}^{r\times p}, \text{rank}(A) = r \leq p$, $B \in \mathbb{R}^{q\times s}, \text{rank}(B) = s \leq q$ y $C \in \mathbb{R}^{r\times s}$, la matriz aleatoria $Z = AXB + C$ sigue una distribución matriz-elíptica con parámetros:

$$Z \sim \mathcal{ME}(AMB + C, A\Sigma^{row}A', B\Sigma^{col}B', g)$$

Los estimadores correspondientes se transforman de la misma forma que los parámetros. Es decir, teniendo un muestra $\mathfrak{X} = (X_1, \dots X_n)$ con sus estimadores MMCD $\hat{M}_{\mathfrak{X}}, \hat{\Sigma}^{row}_{\mathfrak{X}}, \hat{\Sigma}^{col}_{\mathfrak{X}}$, la muestra aleatoria $\mathfrak{Z} = (AX_1B + C, \dots, AX_nB + C)$ tiene estimadores:

$$\hat{M}_{\mathfrak{Z}} = A\hat{M}_{\mathfrak{X}}B + C, \quad \hat{\Sigma}^{row}_{\mathfrak{Z}} = A\hat{\Sigma}^{row}_{\mathfrak{X}}A', \quad \hat{\Sigma}^{col}_{\mathfrak{Z}} = B\hat{\Sigma}^{col}_{\mathfrak{X}}B'$$

Al igual que se hace en Mayrhofer et al. (2025), es importante recalcar que esta propiedad se da en el contexto matriz-variante, y en general no para matrices vectorizadas. En ese caso solo se cumple para transformaciones con la estructura de Kronecker: $vec(AXB + C) = (B' \otimes A)\text{vec}(X) + \text{vec}(C)$

Equivarianza Matriz-Afín de los Estimadores MMCD

Dada una muestra $\mathfrak{X} = (X_1, \dots X_n)$ de matrices $p \times q$ donde $X_i \sim \mathcal{ME}(M_{\mathfrak{X}}, \Sigma^{row}_{\mathfrak{X}}, \Sigma^{col}_{\mathfrak{X}}, g)$ y $\mathfrak{Z} = (Z_1, \dots, Z_n)$ siendo la transformación afín de $\mathfrak{X}$ de forma que $Z_i = AX_iB + C$ con $A \in \mathbb{R}^{p\times p}, B \in \mathbb{R}^{q\times q}$ con A, B invertibles y $C \in \mathbb{R}^{p\times q}$, se cumple:

- Los estimadores MMCD son equivariantes matriz-afines.
- Las distancias de Mahalanobis son iguales, $\text{MMD}^2(Z_i, \hat{M}_{\mathfrak{Z}}, \hat{\Sigma}^{row}_{\mathfrak{Z}}, \hat{\Sigma}^{col}_{\mathfrak{Z}}) = \text{MMD}^2(Z_i, \hat{M}_{\mathfrak{X}}, \hat{\Sigma}^{row}_{\mathfrak{X}}, \hat{\Sigma}^{col}_{\mathfrak{X}})$

4.2.2. Punto de ruptura

En Mayrhofer et al. (2025) se presentan resultados para obtener el punto de ruptura tanto para el estimador de localización $\hat{M}$ como para los estimadores de covarianza de forma conjunta. Lo más relevante en cuanto a esta propiedad es la relación que guarda el punto de ruptura del MMCD con el punto de ruptura del MCD.

Punto de ruptura de los Estimadores MMCD vs. Estimadores MCD

Ambos puntos de ruptura coinciden si $p = 1$ y/o $q = 1$ y además los estimadores MMCD tienen un punto de ruptura más elevado (es decir, hacen falta más perturbaciones en los datos para afectar a los estimadores) que los estimadores MCD aplicados a las mismas muestras vectorizadas siempre que $p \geq 2$ y $q \geq 2$.

Siendo $d = \lfloor \frac{p}{q} + \frac{q}{p} \rfloor$. El punto de ruptura máximo obtenible en el MMCD es $\frac{1}{n}\lfloor \frac{(n-d)}{2} \rfloor$ y se obtiene si $h = \lfloor \frac{(n-d-2)}{2} \rfloor$.

4.2.3. Consistencia en Distribuciones Elípticas

Escalando los estimadores MMCD para la covarianza tanto de filas como de columnas con un *factor de consistencia* específico para cada distribución, se puede obtener consistencia para distribuciones elípticas.

Consistencia de los Estimadores MMCD

Siendo $X_1, \ldots, X_n$ una muestra aleatoria de una distribución $\mathcal{ME}(M, \Sigma^{row}, \Sigma^{col}, g)$ con covarianzas positivas definidas. Con los estimadores MMCD correspondientes $(\hat{M}, \hat{\Sigma}^{row}, \hat{\Sigma}^{col})$ se cumple:

$$||\hat{M} - M|| \xrightarrow{c.s.} 0$$

$$||c(\alpha)\hat{\Sigma}^{col} \otimes \hat{\Sigma}^{row} - \Sigma^{col} \otimes \Sigma^{row}|| \xrightarrow{c.s.} 0$$

Donde $c(\alpha), \alpha = \frac{h}{n} \in [0.5, 1]$ es un factor de consistencia específico para cada distribución del tipo a los considerados en Croux and Haesbroeck (1999).

Para los estimadores MMCD *brutos* se explica que para obtener consistencia en el modelo normal se debe usar el factor de consistencia:

$$c(\alpha) = \frac{\alpha}{F_{\chi^2_{pq+2}}(\chi^2_{\alpha;pq})}$$

4.2.4. Estimador MMCD *repesado*

Los estimadores MMCD *brutos* tienen máxima robustez cuando alrededor de la mitad de las observaciones son descartadas, tal y como se ha explicado en la sección del punto de ruptura. Al igual que con el MCD, disminuir de forma tan drástica el número de observaciones a utilizar reduce enormemente la eficiencia en el modelo normal.

Aumentar el número de observaciones de forma controlada, sin añadir observaciones atípicas, lograría aumentar su eficiencia sin afectar negativamente a su robustez, así que en Lopuhaä and Rousseeuw (1991) (ver, también, Maronna et al., 2019) se propuso un procedimiento de *repesado* en un único paso. Este procedimiento se aplica a los estimadores MMCD *brutos* definiendo estimadores de máxima verosimilitud con pesos que dependen de las distancias de Mahalanobis de esos estimadores MMCD *brutos*.

Estimadores MMCD con pesos

Dada una muestra $\mathfrak{X}$ de tamaño n independiente e idénticamente distribuida según una distribución continua $p \times q$ matriz-variante, en la que $d = \lfloor \frac{p}{q} + \frac{q}{p} \rfloor$, con $p, q \in \mathbb{N}$ y $p \geq 2, q \geq 2$. Los estimadores MMCD *brutos* serán $\hat{M}$, $\hat{\Sigma}^{row}$ y $\hat{\Sigma}^{col}$. Los estimadores MMCD *repesados* se obtienen de la siguiente manera:

- $\tilde{M} = \frac{1}{\sum_{i=1}^{n} w(\mathrm{MMD}(X_i))} \sum_{i=1}^{n} w(\mathrm{MMD}(X_i))$
- $\tilde{\Sigma}^{row} = \frac{1}{q \sum_{i=1}^{n} w(\mathrm{MMD}(X_i))} \sum_{i=1}^{n} w(\mathrm{MMD}(X_i))(X_i - \tilde{M})\tilde{\Omega}^{col}(X_i - \tilde{M})'$
- $\tilde{\Sigma}^{col} = \frac{1}{p \sum_{i=1}^{n} w(\mathrm{MMD}(X_i))} \sum_{i=1}^{n} w(\mathrm{MMD}(X_i))(X_i - \tilde{M})'\tilde{\Omega}^{row}(X_i - \tilde{M})$

Donde $w : [0, \infty) \to [0, \infty)$ es una función de pesos no-creciente y acotada que cumple $w(\mathrm{MMD}(X_i)) > 0$ para al menos $\lfloor \frac{n+d+2}{2} \rfloor$ observaciones y para grandes distancias tiende a 0, es decir, $w(\mathrm{MMD}(X_i)) = 0$ *si* $\mathrm{MMD}(X_i) > c_1 > 0$.

En Mayrhofer et al. (2025) además se demuestra que el estimador MMCD *repesado* mantiene el punto de ruptura del MMCD *bruto*.

4.3. Algoritmo MMCD

4.3.1. Cambios fundamentales respecto al MCD

El algoritmo *Fast-MCD* será la base para el algoritmo MMCD. Las diferencias principales se deben al hecho de que no se pueden estimar las dos matrices de covarianza de forma conjunta, sino que se debe incorporar el algoritmo de *flip-flop* visto en el anterior capítulo en el paso del C-Step.

Suponiendo que tenemos una muestra aleatoria matriz-variante $\mathfrak{X} = (X_1, \ldots, X_n)$ con $X_i \in \mathbb{R}^{p \times q}$. Asumamos que en este momento tenemos un subconjunto de tamaño h tal que $H_{old} \subset \{1, \ldots, n\}$ y $|H_{old}| = h > \lfloor p/q + q/p \rfloor + 2$. El C-Step funciona de la siguiente manera:

C-Step del Algoritmo MMCD (Mayrhofer et al., 2025)

1. Se calculan los estimadores de máxima verosimilitud basándose únicamente en las observaciones del subconjunto H_{old} usando el algoritmo de *flip-flops*: $(\hat{M}_{H_{old}}, \hat{\Sigma}^{row}_{H_{old}}, \hat{\Sigma}^{col}_{H_{old}})$.
2. Se calculan las distancias de Mahalanobis al cuadrado para cada $i = 1, \ldots, n$ tal que $d_i^2(H_{old}) = \text{MMD}^2(X_i, \hat{M}_{H_{old}}, \hat{\Sigma}^{row}_{H_{old}}, \hat{\Sigma}^{col}_{H_{old}})$.
3. Se calcula el nuevo subconjunto H_{new}:
   a) Se ordenan las distancias de Mahalanobis al cuadrado de menor a mayor, obteniendo una permutación π de $\{1, \ldots, n\}$ tal que $d^2_{\pi(1)}(H_{old}) \leq \ldots \leq d^2_{\pi(n)}(H_{old})$.
   b) Se define el nuevo subconjunto de tamaño h como $H_{new} = \{\pi(1), \ldots, \pi(h)\}$.
4. Se actualizan los estimadores MMCD con el nuevo subconjunto H_{new}.

En Mayrhofer et al. (2025) se explica y demuestra que todos los pasos del C-Step son no-decrecientes respecto a la verosimilitud, por lo que el determinante de covarianza debe converger en un número finito de subconjuntos explorados. La solución final se alcanza cuando se obtenga el mismo determinante de covarianza en dos subconjuntos consecutivos, aunque al igual que en el caso multivariante no se garantiza encontrar un óptimo global: esto se trata de solucionar en la implementación final del algoritmo MMCD.

4.3.2. Visión general del algoritmo

El algoritmo MMCD engloba el C-Step de la sección anterior con otros pasos adicionales para obtener los mejores estimadores posibles. A continuación se presentan las ideas principales explicadas en Mayrhofer et al. (2025):

- **Inicialización**: Dado que la sucesión de C-Steps no garantiza encontrar un óptimo global, se debe realizar la sucesión con conjuntos iniciales diferentes. Para aumentar la probabilidad de obtener un subconjunto inicial sin una alta proporción de observaciones atípicas, estos conjuntos iniciales tienen un tamaño que es el mínimo h posible, $\lfloor p/q + q/p \rfloor + 2$, aunque no necesariamente sea el mismo h que se utilizará más adelante en el algoritmo. Se obtienen entonces m subconjuntos iniciales de este tamaño, cuyo número típicamente se fija en $m = 500$. Para casos en los que el número de filas o columnas sean muy diferentes es recomendable aumentar m.
- **Descarte**: En un principio, se realizan únicamente 2 iteraciones de C-Step (Rousseeuw and Driessen, 1999) y estimación de máxima verosimilitud (el algoritmo de *flip-flop*) (Werner et al., 2008) para cada uno de los m subconjuntos iniciales. Con únicamente dos pasos de cada uno basta para poder detectar los subconjuntos con menor determinante de covarianza y descartar el resto. Normalmente se mantienen de los m conjuntos iniciales únicamente los 10 con menor determinante de covarianza, y sobre estos sí que se realizan iteraciones del C-Step y de estimación hasta la convergencia.

- **Reescalado**: Los estimadores MMCD *brutos* se reescalan usando el factor de consistencia $c(\alpha) = \frac{\alpha}{F_{\chi^2_{pq+2}}(\chi^2_{\alpha;pq})}$ tal y como se explicó en la sección de propiedades de los estimadores MMCD.

- **Repesado**: Usando los estimadores reescalados se calculan los estimadores *repesados* MMCD con las ecuaciones vistas en la sección de propiedades del MMCD. Los pesos a utilizar serán los siguientes:

$$w(\mathrm{MMD}(X_i)) = \begin{cases} 1 \text{ si } i \in H \text{ ó } \mathrm{MMD}^2(X_i) < \chi^2_{pq;0,975} \\ 0 \text{ en otro caso} \end{cases}$$

  Siendo H el conjunto de observaciones seleccionadas en los pasos anteriores.

- **Reescalado del Repesado**: Los estimadores MMCD *repesados* se reescalan de nuevo con el factor de consistencia $c(\tilde{\alpha}) = c(\frac{\tilde{h}}{n})$ donde $\tilde{h}$ es el número de observaciones con el peso igual a 1.

- **División de observaciones (opcional)**: En casos donde n sea muy grande el coste computacional puede ser muy elevado al necesitarse calcular las distancias de Mahalanobis entre todas las observaciones. Una solución propuesta por Rousseeuw and Driessen (1999) es dividir el conjunto de datos en subconjuntos más pequeños y obtener los 10 estimadores iniciales en cada uno de ellos. Luego, se unen los conjuntos y se itera sobre los estimadores iniciales de cada subconjunto hasta el final.

4.3.3. Implementación en R

Especificados todos los detalles del MMCD solo queda presentar el núcleo de este trabajo y su principal resultado: la implementación en el lenguaje de programación R. El algoritmo en su conjunto se presenta en el Apéndice A.3.1, organizado por sus principales funciones. Estas funciones son las siguientes:

- `matrixMahalanobis`: Función auxiliar para calcular distancias de Mahalanobis para matrices.

- `MLE`: Estimación por máxima verosimilitud. Contiene algunos sutiles cambios para maximizar el rendimiento del algoritmo MMCD en su conjunto. En esta versión se proporciona una forma de controlar el número de ejecuciones del algoritmo y poder limitar así a 2 iteraciones la estimación inicial de todos los conjuntos.

- `C_STEP`: Implementación del C-Step del algoritmo MMCD. También consta de un parámetro opcional `max_iter` para controlar el número de iteraciones del algoritmo en las exploraciones iniciales. Como optimización importante adicional, no se devuelven los estimadores en esas exploraciones iniciales: nunca serían usados y ocuparían espacio innecesario en memoria.

- `MMCD`: Núcleo principal del algoritmo MMCD. Se encarga de realizar la exploración inicial, descartar los subconjuntos con mayor determinante de covarianza y realizar las

iteraciones finales sobre los subconjuntos seleccionados. La función ha sido diseñada con el objetivo de ser muy clara en sus pasos a la vez que se saca el máximo provecho de la paralelización de las ejecuciones de las funciones anteriores.

4.4. Ejemplos de Aplicaciones

Una vez se ha explicado el algoritmo MMCD con detalle y se ha implementado en R, solo queda demostrar su utilidad y rendimiento con algunos ejemplos prácticos. En esta última sección se presentan dos ejemplos de aplicación, uno con datos simulados y otro con datos reales y utilidad más práctica.

4.4.1. Ejemplo con Datos Simulados

Este ejemplo trata de demostrar la robustez del algoritmo MMCD frente a datos atípicos. Se generan 100 observaciones con filas $p = 30$ y columnas $q = 40$ de una distribución matriz-normal con una matriz de medias:

$$M = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{pmatrix}$$

Y unas matrices de covarianza de filas y columnas:

$$\Sigma^{row} = \Sigma^{col} = \begin{pmatrix} 0.5 & 0 & \dots & 0 \\ 0 & 0.5 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0.5 \end{pmatrix}$$

A continuación se añaden 10 observaciones atípicas con una matriz de medias:

$$M = \begin{pmatrix} 2 & \dots & 2 \\ \vdots & \ddots & \vdots \\ 2 & \dots & 2 \end{pmatrix}$$

Y unas matrices de covarianza como las siguientes:

$$\Sigma^{row} = \Sigma^{col} = \begin{pmatrix} 1.2 & 0.2 & \dots & 0.2 \\ 0.2 & 1.2 & \dots & 0.2 \\ \vdots & \vdots & \ddots & \vdots \\ 0.2 & 0.2 & 0.2 & 1.2 \end{pmatrix}$$

Por último, se añaden otras 10 observaciones atípicas con una matriz de medias:

$$M = \begin{pmatrix} -2 & \dots & -2 \\ \vdots & \ddots & \vdots \\ -2 & \dots & -2 \end{pmatrix}$$

Y unas matrices de covarianza como las siguientes:

$$\Sigma^{row} = \Sigma^{col} = \begin{pmatrix} 5 & 0 & \dots & 0 \\ 0 & 5 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

Se espera que el algoritmo MMCD sea capaz de detectar estas observaciones atípicas y descartarlas en el proceso de estimación de los parámetros; consiguiendo actuar como si no estuvieran presentes en la muestra. Una vez estimadas las matrices de media y covarianzas, se calculan las distancias con cada observación respecto al centro de la muestra y se dibujan en el siguiente gráfico usando tanto MMCD como el método clásico:

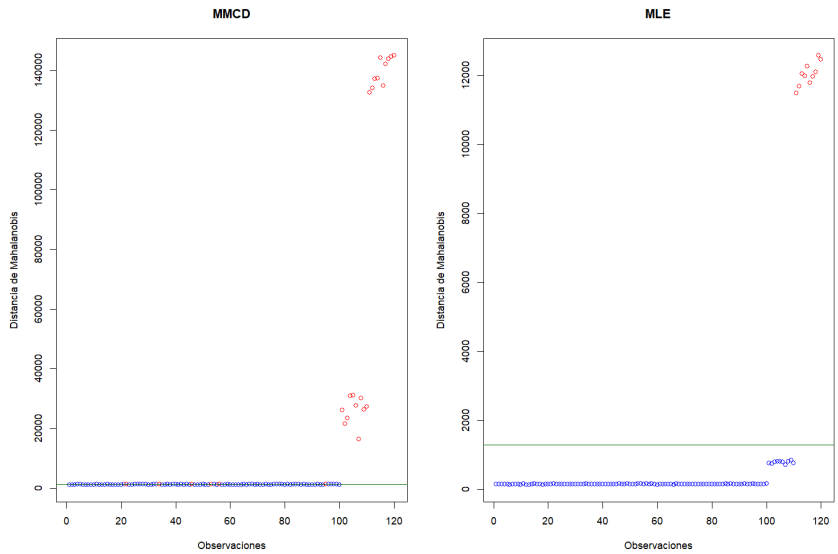


Figura 4.1: Distancias de Mahalanobis al cuadrado con MMCD y Método Clásico

La línea verde indica el intervalo de confianza del 97.5 % para las distancias, estando fijada en $\chi^2_{pq;0.975}$. Marca como atípicas (de color rojo) las observaciones cuya distancia de Mahalanobis al cuadrado cumpla:

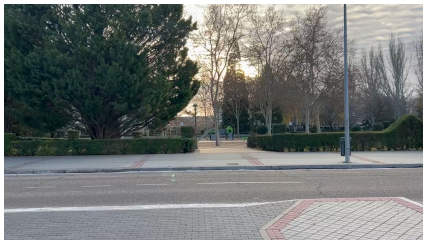
$$\text{MMD}^2(X_i, \hat{M}, \hat{\Sigma}^{row}, \hat{\Sigma}^{col}) > \chi^2_{pq;0.975}$$

Esta línea verde muestra cómo claramente el método MMCD permite omitir por completo las observaciones atípicas, marcando el intervalo sin influencia de esas observaciones. Cabe destacar que el método tradicional no es capaz de detectar el primer grupo de observaciones atípicas, a diferencia del MMCD. Está claro que el método MMCD es mucho más robusto frente a todo tipo de contaminaciones e incluso a combinaciones de éstas.

4.4.2. Ejemplo con Datos Reales - Vídeo

La gran robustez del MCD aplicada a datos matriciales permite nuevas aplicaciones que antes resultarían poco esperables. En este ejemplo se tratará de demostrar la utilidad del algoritmo MMCD en la detección de eventos en un vídeo.

El vídeo en cuestión es una grabación propia de una calle de Valladolid. Durante la mayoría de los 25 segundos no ocurre nada, puesto que hay muy poco trafico. En un momento dado, un autobús pasa por la calle y se muestra en el vídeo. Las dos imágenes a continuación ilustran la situación más común y cuando pasa el autobús:



(a) Situación normal

(b) Momento en el que pasa el autobús

Figura 4.2: Fotogramas del vídeo

Para el tratamiento del vídeo de forma computacionalmente asumible, se ha reducido a una resolución de 256×144p y a blanco y negro, aunque el mismo concepto podría ser aplicable con mayores calidades sin mayor complejidad. El mismo procedimiento que en el ejemplo anterior, visible en el Apéndice A.3.2, permite calcular las distancias de cada fotograma al centro de la muestra y crear el siguiente gráfico:

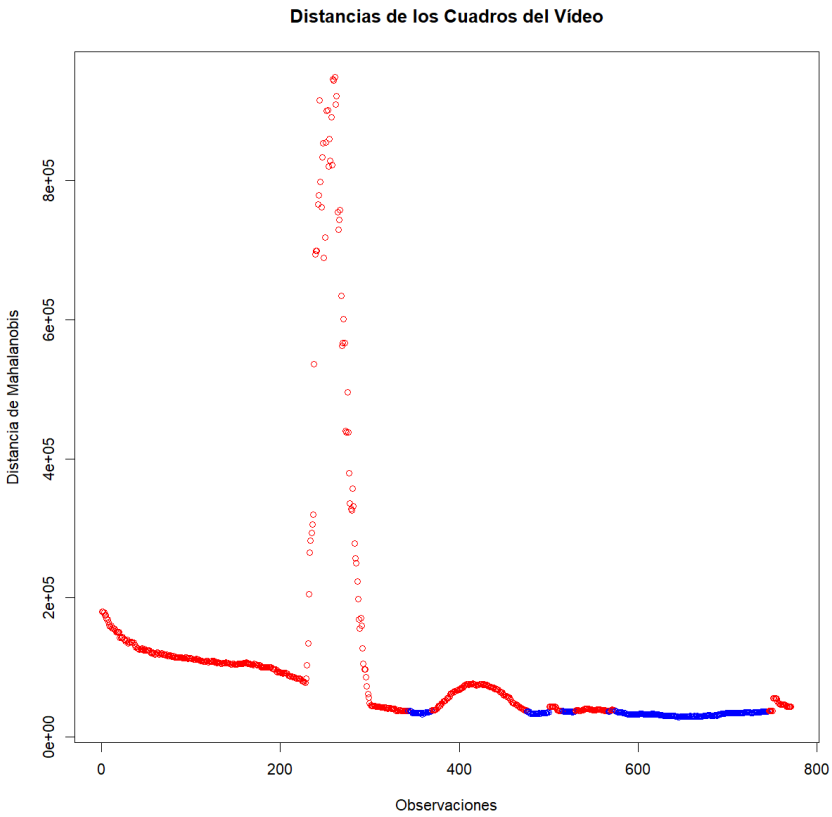


Figura 4.3: Distancias en el vídeo de demostración

De todos los diferentes puntos en el gráfico, destaca el momento en el que pasa el autobús de forma clara, en forma de pico. El resto de puntos atípicos (rojos) son probablemente debidos a pequeños movimientos en la grabación u otros pequeños cambios. Es importante recalcar el hecho de que estos otros eventos también son detectados; y permite que incluso con un evento tan perturbador de la muestra se pueda seguir analizando el vídeo con normalidad en el resto de la grabación. Esto es consecuencia, nuevamente, de la robustez del algoritmo MMCD.

Con este ejemplo queda demostrada la gran utilidad del MMCD para detectar eventos atípicos incluso en grabaciones muy cortas, algo que puede ser de ayuda en cámaras de vigilancia con estrictas limitaciones de memoria, por ejemplo.

Capítulo 5

Conclusiones

En un mundo donde los datos están permanentemente presentes en nuestras vidas y las cantidades de datos generadas son cada vez mayores, la necesidad de métodos de análisis de estos mismos datos eficientes y robustos es cada vez mayor. En este trabajo se ha estudiado el problema de la estimación robusta y la detección de atípicos en datos tipo matriz, como son las imágenes o los vídeos que se pueden encontrar en cualquier momento del día a día.

La adaptación del MCD a datos matriciales otorga una muy útil herramienta para la estimación y la detección de atípicos en este tipo de datos. Un ejemplo mostrado en este trabajo muestra la eficacia de esta adaptación del algoritmo en la detección de atípicos en imágenes y vídeos y sus aplicaciones pueden ir mucho más allá de lo comentado en este trabajo.

A continuación se presentan otras posibles líneas de trabajo futuras que se podrían seguir para continuar con el estudio de estos algoritmos, sus derivados, aplicaciones o incluso únicamente sus ideas generales:

5.1. Líneas de trabajo futuras

5.1.1. Aplicaciones comunes

Algunas de esas posibles aplicaciones podrían ir en la línea de las ya presentadas en el capítulo 2, como son la creación de modelos, el análisis de componentes principales, la clasificación o el agrupamiento de datos (como se trata en el capítulo siguiente, con una breve introducción al TCLUST); pero aplicadas a datos matriciales. El futuro de este tipo de algoritmos es prometedor y su estudio puede aportar grandes avances en el campo del análisis de datos.

5.1.2. Aplicaciones en imágenes

Otra línea de trabajo futura sería la aplicación de estos algoritmos en imágenes. En este trabajo se ha presentado un ejemplo simple de detección de atípicos en imágenes en forma de vídeo, pero es sencillo imaginar aplicaciones más complejas de estos algoritmos en imágenes, como puede ser la detección de tumores en imágenes médicas. Sería necesaria una gran colección de imágenes con tumores y sin tumores para poder ajustar los parámetros del algoritmo y tratar esas observaciones con tumores como atípicas, por ejemplo.

5.1.3. Aplicaciones en vídeos

Por último, se podría estudiar la aplicación de estos algoritmos en vídeos. En este trabajo se ha presentado un ejemplo sencillo de detección de atípicos en un vídeo en el que se detectaba el movimiento de un autobús en una calle. Sería interesante estudiar la aplicación de estos algoritmos en vídeos más complejos, como puede ser la detección de objetos en movimiento en un vídeo de vigilancia. Así, se podría evitar almacenar cantidades enormes de vídeo y solo guardar aquellos fragmentos en los que se detecte movimiento.

Capítulo 6

Algoritmo TCLUST para matrices

6.1. Introducción

Es fácil pensar en la posibilidad de utilizar las ideas generales del MMCD para realizar un agrupamiento de datos en diferentes grupos. En este algoritmo, el TCLUST, cobra una especial relevancia la idea de los pasos de concentración (C-Step), discutidos anteriormente.

Hay una gran cantidad de literatura sobre la conexión entre el Fast-MCD y el algoritmo de K-Medias, como es el algoritmo de K-Medias Recortadas (Trimmed K-Means) (García-Escudero et al., 2003). Usar distancias de Mahalanobis en lugar de euclídeas es otra mejora fácilmente implementable, aunque trae consigo problemas como el hecho de que los grupos (clusters) grandes absorben a los pequeños (Maronna and Jacovkis, 1974). Esto genera la necesidad de fijar más restricciones en el algoritmo, ya sea implícita o explícitamente. El algoritmo TCLUST (Fritz et al., 2013) controla las diferencias relativas de los grupos usando las diferencias de los autovalores de las matrices de covarianza.

Tradicionalmente el principal problema del TCLUST era el enorme tiempo de ejecución necesario para realizar el agrupamiento, dado que se necesita resolver un problema de optimización con un enorme número de restricciones en cada paso de concentración (C-Step). En Fritz et al. (2013) se propone una solución a este problema, replanteando el algoritmo de optimización y reduciendo considerablemente el número de funciones y restricciones a evaluar en cada C-Step.

Este Trabajo Fin de Grado también incluye en el Apéndice A.4 una implementación original del TCLUST en R con las modificaciones necesarias para que sea aplicable a datos matriciales, aunque se trata de una versión sencilla en la que no se tienen en cuenta las restricciones como las de los autovalores. Esto implica que en algunos casos particulares como en los que hay contaminación colineal, el resultado podría ser subóptimo. A cambio, se permite una mayor simplicidad en el algoritmo, mayor velocidad de ejecución y una mayor facilidad de implementación.

6.2. Algoritmo TCLUST

6.2.1. Descripción del Problema

Antes de proceder a la explicación del algoritmo, conviene definir el problema con detalle. Dada una muestra $\mathcal{X} = (x_1, \ldots, x_n)$ de tamaño n y una función de densidad de una distribución normal matricial $\mathcal{MN}(M, \Sigma^{row}, \Sigma^{col})$, el *Problema de Agrupamiento Robusto Restringido* para un nivel de recorte α es el siguiente:

Buscar una partición $R_0, R_1, \ldots, R_k$ de índices $\{1, \ldots, n\}$ con $\#R_0 = \lceil n\alpha \rceil$. R_0 es el grupo al que se asignan las observaciones descartadas. Se definen también los centros $m_1, \ldots, m_k$ en $\mathbb{R}^{p \times q}$ y matrices de covarianza tanto de filas como de columnas para cada partición, $\Sigma_1^{row}, \ldots, \Sigma_k^{row}$ y $\Sigma_1^{col}, \ldots, \Sigma_k^{col}$ respectivamente. Además, deberán definirse los pesos $p_1, \ldots, p_k$ con $p_j \in [0, 1]$ y $\sum_{j=1}^{k} p_j = 1$ tal que se maximice:

$$\sum_{j=1}^{k} \sum_{i \in R_j} \log(p_j \mathcal{MN}(x_i; m_j, \Sigma_j^{row}, \Sigma_j^{col}))$$

En función de diversas decisiones de las restricciones, como la elección de pesos o la forma de las matrices de covarianza, se pueden obtener diferentes algoritmos de agrupamiento. En este trabajo se estudiará el algoritmo TCLUST sin restricciones adicionales.

6.2.2. Algoritmo

Teniendo definidas las siguientes funciones, que representan la verosimilitud de que una observación pertenezca al grupo k y la verosimilitud máxima de una observación, respectivamente:

$$D_k(x; \theta) = \log(\pi_k \phi(x; \theta_k)) = \log(\pi_k) - \frac{q}{2} \log(|\Sigma_k^{row}|) - \frac{p}{2} \log(|\Sigma_k^{col}|) - \frac{1}{2} MMD^2(x, \theta_k) - pq \log(\pi)$$

$$D(x, \theta) = \max_{1 \leq k \leq K} D_k(x; \theta)$$

Además, se fijará el número de observaciones que se mantienen (y que no se asignan al *grupo 0*) como:

$$h = \lfloor n(1 - \alpha) \rfloor$$

Siendo α una constante que, por ejemplo, puede fijarse en 0.05. Con todo esto definido, solo queda explicar las dos fases del algoritmo claramente diferenciadas, aunque el bucle principal es común en ambas.

Inicialización

En esta fase, se fijan los pesos en $\pi_i^{(0)} = 1/k$. Luego se realizan un gran número de inicializaciones `nstart`, en las que para cada grupo se eligen $\lfloor p/q + q/p \rfloor + 2$ observaciones

al azar y el resto se asignan al grupo 0, el de las observaciones descartadas. Para cada grupo se obtienen los estimadores para las matrices de medias y covarianza de filas y columnas mediante el método MLE, que junto a los pesos se agrupan de la siguiente forma para la iteración $l-1$:

$$\theta^{(l-1)} = (\{\pi_k^{(l-1)}\}_{k=1}^K, \{\mu_k^{(l-1)}\}_{k=1}^K, \{\Sigma^{row}{}_k^{(l-1)}\}_{k=1}^K, \{\Sigma^{col}{}_k^{(l-1)}\}_{k=1}^K)$$

Bucle Principal

Para cada una de las inicializaciones, se itera sobre el algoritmo de actualización un pequeño número de veces. En este algoritmo, se calcula la verosimilitud máxima para cada observación y se ordena de la siguiente forma:

$$D(x_{1:n}, \theta^{(l-1)}) \leq D(x_{2:n}, \theta^{(l-1)}) \leq \dots D(x_{n:n}, \theta^{(l-1)})$$

Las observaciones con menor verosimilitud máxima (y por tanto, menor verosimilitud en cualquier grupo) son descartadas y asignadas al *grupo 0*:

$$R_0^{(l)} = \{i : D(x_{i:n}, \theta^{(l-1)}) \leq D(x_{n-h:n}, \theta^{(l-1)})\}$$

El resto de observaciones se asignan al grupo que les otorga esa verosimilitud máxima:

$$R_k^{(l)} = \{i \notin R_0^{(l)} \vee D_k(x_i, \theta^{(l-1)}) = D(x_i, \theta^{(l-1)})\}$$

Luego, se calculan de nuevo los estimadores de cada grupo $\theta_k^{(l)}$ usando MLE y las asignaciones de las observaciones definidas en $R_k^{(l)}$. Los pesos en cada iteración serán $\pi_k^{(l)} = \frac{\#R_k^{(l)}}{h}$.

Este proceso se itera un número limitado de veces en la fase de inicialización o hasta que se llegue a la estabilidad de los grupos en la fase final, medida mediante la función que se trata de optimizar:

$$\sum_{k=1}^{K} \sum_{i \in R_k} \log(\pi_k^{(l)} \phi(x_i; \theta_k^{(l)}))$$

Solución final

Para cada una de las inicializaciones, se registra su valor de la función objetivo y se mantienen solo las 10 mejores. Estas 10 soluciones se ejecutan de nuevo en el bucle principal, pero esta vez con un número de iteraciones mucho mayor. La solución final será la que tenga el mayor valor de la función objetivo.

6.3. Ejemplo con datos simulados

Para la demostración de la eficacia del algoritmo TCLUST para el agrupamiento de datos matriciales se propone el siguiente ejemplo extremadamente sencillo. Se plantean 3 grupos de 100 observaciones cada uno, con forma $p = 10$ y $q = 18$, obtenidas de distribuciones normales matriciales con medias distintas aunque covarianzas iguales. Las matrices de covarianza tienen la siguiente forma:

$$\Sigma^{row} = \Sigma^{col} = \begin{pmatrix} 0.5 & 0.25 & 0.25\ldots & 0.25 \\ 0.25 & 0.5 & \ldots & 0.25 \\ \vdots & \vdots & \ddots & \vdots \\ 0.25 & 0.25 & 0.25 & 0.5 \end{pmatrix}$$

Estas observaciones se generan con matrices de medias con todos sus elementos 0, 2 o -2 para cada grupo. No se mezclan, de forma que sus índices servirán para evaluar la eficacia del algoritmo.

El resultado de ejecutar el código disponible en el Apéndice A.4 es el siguiente:

- Grupo 1: 47 10 36 19 96 21 35 4 3 11 18 46 89 98 62 5 6 24 70 53 39 63 94 9 34 90 65 87 43 42 100 71 66 88 84 31 14 32 7 85 79 13 67 58 27 97 75 41 80 92 77 44 2 95 50 26 52 33 37 76 73 20 38 93 28 22 81 55 69 91 78 74 8 99 86 64 60 72 54 82 49 48 30 29 61 45 23 1 59 83 40 57 68 25 15 51 56
- Grupo 2: 240 210 254 241 258 249 215 212 273 293 239 225 246 213 289 251 296 276 209 272 274 260 252 253 297 233 257 256 291 294 244 235 211 269 285 266 204 280 268 259 216 223 255 284 263 230 231 205 279 242 270 295 217 227 202 264 281 250 228 243 292 287 265 214 288 221 277 201 267 236 206 247 234 224 218 283 299 220 300 286 238 248 237 229 226 222 278 219 262 290 207 261 275 208
- Grupo 3: 158 107 115 159 128 162 174 161 163 125 177 179 121 154 171 114 129 194 180 150 148 139 103 134 151 141 101 170 133 172 116 146 143 189 166 165 191 147 181 109 167 196 132 168 173 138 190 123 127 104 142 193 155 199 182 192 145 169 183 176 130 102 188 157 137 131 126 140 118 135 200 152 175 178 112 110 195 187 185 164 113 136 156 184 122 108 106 124 119 198 120 149 105 197
- Grupo 0 - observaciones descartadas: 160 186 271 298 17 245 117 111 144 232 282 16 12 203 153

Como se puede observar fácilmente, el algoritmo ha conseguido detectar y agrupar, en su mayoría, las observaciones de forma muy correcta: en el grupo 1 se han asignado las primeras 100 observaciones, en el grupo 3 las siguientes 100 y en el grupo 2 las últimas 100, tal y como fueron generadas.

Bibliografía

R. W. Butler, P. L. Davies, and M. Jhun. Asymptotics for the minimum covariance determinant estimator. *The Annals of Statistics*, 21(3):1385 – 1400, 1993. URL https://doi.org/10.1214/aos/1176349264.

E. A. Cator and H. P. Lopuhaä. Asymptotic expansion of the minimum covariance determinant estimators. *Journal of Multivariate Analysis*, 101(10):2372–2388, 2010. URL https://www.sciencedirect.com/science/article/pii/S0047259X10001284.

E. A. Cator and H. P. Lopuhaä. Central limit theorem and influence function for the mcd estimators at general multivariate distributions. *Bernoulli*, 18(2), 2012. URL http://dx.doi.org/10.3150/11-BEJ353.

C. Croux and G. Haesbroeck. Influence function and efficiency of the minimum covariance determinant scatter matrix estimator. *Journal of Multivariate Analysis*, 71(2):161–190, 1999. URL https://www.sciencedirect.com/science/article/pii/S0047259X99918390.

J. A. Cuesta-Albertos, A. Gordaliza, and C. Matrán. Trimmed k-means: an attempt to robustify quantizers. *The Annals of Statistics*, 25(2):553 – 576, 1997. URL https://doi.org/10.1214/aos/1031833664.

P. Dutilleul. The mle algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123, 1999. URL https://doi.org/10.1080/00949659908811970.

P. Filzmoser. *chemometrics: Multivariate Statistical Analysis in Chemometrics*, 2023. URL https://CRAN.R-project.org/package=chemometrics. R package version 1.4.4.

H. Fritz, L. A. García-Escudero, and A. Mayo-Iscar. tclust: An r package for a trimming approach to cluster analysis. *Journal of Statistical Software*, 47(12):1–26, 2012. URL https://www.jstatsoft.org/index.php/jss/article/view/v047i12.

H. Fritz, L. A. García-Escudero, and A. Mayo-Iscar. A fast algorithm for robust constrained clustering. *Computational Statistics & Data Analysis*, 61:124–136, 2013. URL https://www.sciencedirect.com/science/article/pii/S0167947312004203.

L. A. García-Escudero, A. Gordaliza, and C. Matrán. Trimming tools in exploratory data analysis. *Journal of Computational and Graphical Statistics*, 12(2):434–449, 2003. URL http://www.jstor.org/stable/1391203.

M. Hubert, M. Debruyne, and P. J. Rousseeuw. Minimum covariance determinant and extensions. *WIREs Computational Statistics*, 10(3):e1421, 2018. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.1421.

H. P. Lopuhaä. Asymptotics of reweighted estimators of multivariate location and scatter. *The Annals of Statistics*, 27(5):1638 – 1665, 1999. URL https://doi.org/10.1214/aos/1017939145.

H. P. Lopuhaä and P. J. Rousseeuw. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *The Annals of Statistics*, 19(1):229 – 248, 1991. URL https://doi.org/10.1214/aos/1176347978.

M. Maechler, P. J. Rousseeuw, C. Croux, V. Todorov, A. Ruckstuhl, M. Salibian-Barrera, T. Verbeke, M. Koller, E. L. T. Conceicao, and M. Anna di Palma. *robustbase: Basic Robust Statistics*, 2024. URL http://robustbase.r-forge.r-project.org/. R package version 0.99-4-1.

R. Maronna and P. M. Jacovkis. Multivariate clustering procedures with variable metrics. *Biometrics*, 30(3):499–505, 1974. URL http://www.jstor.org/stable/2529203.

R. A. Maronna, R. D. Martin, V. J Yohai, and M. Salibián-Barrera. *Robust statistics: theory and methods (with R)*. John Wiley & Sons, 2019.

M. Mayrhofer, U. Radojičić, and P. Filzmoser. Robust covariance estimation and explainable outlier detection for matrix-valued data. *Technometrics*, pages 1–23, 2025. URL https://doi.org/10.1080/00401706.2025.2475781.

D. Peña. *Análisis de datos multivariantes*. McGraw-Hill España Cambridge, 2002.

P. J. Rousseeuw. Least median of squares regression. *Journal of The American Statistical Association*, 79:871–880, 1984.

P. J. Rousseeuw and K. Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1999.10485670.

P. J. Rousseeuw and M. Hubert. Anomaly detection by robust statistics. *WIREs Data Mining and Knowledge Discovery*, 8(2):e1236, 2018. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1236.

P. J. Rousseeuw and A. Leroy. *Robust Regression & Outlier Detection*. John Wiley, New York, 1987. ISBN 9780471852339.

P.J. Rousseeuw. Multivariate estimation with high breakdown point. In W. Grossmann, G. Pflug, I. Vincze, and W. Wertz, editors, *Mathematical Statistics and Applications*, pages 283–297. Reidel, 1985.

I. Soloveychik and D. Trushin. Gaussian and robust kronecker product covariance estimation: Existence and uniqueness. *Journal of Multivariate Analysis*, 149:92–113, 2016. URL https://www.sciencedirect.com/science/article/pii/S0047259X16300070.

G. Thompson. *MixMatrix: Classification with Matrix Variate Normal and t Distributions*, 2024. URL https://CRAN.R-project.org/package=MixMatrix. R package version 0.2.8.

V. Todorov and P. Filzmoser. An object-oriented framework for robust multivariate analysis. *Journal of Statistical Software*, 32(3):1–47, 2009. URL `https://www.jstatsoft.org/index.php/jss/article/view/v032i03`.

B. Venables and B. Ripley. *Modern Applied Statistics With S*. Springer, 2002.

K. Werner, M. Jansson, and P. Stoica. On estimation of covariance matrices with kronecker product structure. *Signal Processing, IEEE Transactions on*, 56:478–491, 2008.

Apéndice A

Código

A.1. Código utilizado en el Capítulo 2

A.1.1. Estimación con MCD

```
library(MASS)
library(robustbase)

# Muestra
mu <- matrix(1, nrow = 10, ncol = 1)
covarianza <- diag(10)*.5 + 1
x <- mvrnorm(500, mu, covarianza)

# Contaminacion
mu <- matrix(8, nrow = 10, ncol = 1)
covarianza <- diag(10)*.5 + 1
x2 <- mvrnorm(50, mu, covarianza)

x <- rbind(x, x2)

mcd <- covMcd(x)

# Comparar medias
colMeans(x)
mcd$center

# Comparar varianzas
cov(x)
mcd$cov


# Graficas de distancias
par(mfrow=c(1,2))

# Con MCD
d <- mahalanobis(x, mcd$center, mcd$cov)
plot(d, col=ifelse(d < qchisq(.975, df=10), "blue", "red"), xlab="Observaciones",
     ylab="Distancia de Mahalanobis",
      main = "MCD")
abline(h=qchisq(.975, df=10), col="darkgreen")


```

```r
# Con el metodo tradicional
d <- mahalanobis(x, colMeans(x), cov(x))
plot(d, col=ifelse(d < qchisq(.975, df=10), "blue", "red"), xlab="Observaciones",
     ylab="Distancia de Mahalanobis",
     main = "Estimadores Tradicionales")
abline(h=qchisq(.975, df=10), col="darkgreen")
```

A.1.2. Regresión Lineal

```r
library(robustbase)
data("starsCYG")

starsCYG[20, 2] <- -0.5
starsCYG[7, 1] <- 4.5
starsCYG[11, 1] <- 4.75
starsCYG[9, 2] <- 6.8
starsCYG[1, 2] <- 6.9
starsCYG[2, 2] <- -0.5

index_outliers <- c(20, 30, 34, 11, 14, 1, 2, 9)

par(mfrow=c(1,2))
# Outliers al comienzo
starsCYG <- rbind(starsCYG[index_outliers,], starsCYG[-index_outliers,])
colores <- c(rep("red", 8), rep("blue", nrow(starsCYG)-8))

# Regresion
plot(starsCYG$log.Te, starsCYG$log.light, xlim=c(3.4, 4.9), pch=21, col=colores,
     bg=colores, xlab="log.Te", ylab="log.light")
text(starsCYG[1:8,1], starsCYG[1:8,2], 1:8, pos=4)
abline(lm(log.light~log.Te, data=starsCYG), col="red", lwd=2)
abline(ltsReg(log.light~log.Te, data=starsCYG), col="blue", lwd=2)
text(3.45, 2, substitute(paste(bold("LTS"))), col="blue")
text(3.45, 4.3, substitute(paste(bold("LS"))), col="red")

# Diagnostico
plot(lts <- ltsReg(log.light~log.Te, data=starsCYG), which="rdiag", pch=21, col=colores,
     bg=colores)
text(4, 5.8, "bad leverage points")
text(4, -5.8, "bad leverage points")
text(5.2, 0, "good leverage points")
text(1, 5.8, "vertical outliers")
text(1, -5.8, "vertical outliers")

plot(lts <- ltsReg(log.light~log.Te, data=starsCYG), which="rdiag", pch=21, col=colores,
     bg=colores, classic=T)
```

A.1.3. Análisis de Componentes Principales

```r
library(rrcov)
library(chemometrics)

pcamcd <- princomp(longley, covmat=covMcd(longley))
pcanormal <- princomp(longley)

dmcd <- pcaDiagplot(longley, pcamcd, plot=FALSE)
dnormal <- pcaDiagplot(longley, pcanormal, plot=FALSE)

plot(dmcd$SDist, dmcd$ODist, xlim=c(0,3), main="MCD",
     xlab="Distancia Score", ylab="Distancia Ortogonal")
abline(v=dmcd$critSD)
abline(h=dmcd$critOD)

plot(dnormal$SDist, dnormal$ODist, xlim=c(0,3), main="Normal",
     xlab="Distancia Score", ylab="Distancia Ortogonal")
abline(v=dnormal$critSD)
abline(h=dnormal$critOD)
```

A.1.4. Análisis de Discriminante Lineal

```r
library(MASS)
library(rrcov)
data("pottery")

x <- pottery[,c("CA", "MG")]
grupo <- pottery$origin

x[3,] <- c(6, 0)
x[9,] <- c(6.5, 0)
x[16,] <- c(7, 10)
x[23,] <- c(7.5, 10)

normal <- lda(x, grupo)
mcd <- Linda(x, grupo, method="mcd")

nuevo <- data.frame(expand.grid(seq(min(x[,1]),max(x[,1]),
        length=100),seq(min(x[,2]),max(x[,2]),length=100)))
names(nuevo) <- c("CA", "MG")

par(mfrow=c(1,2))

pred <- as.numeric(predict(normal,nuevo)$class)
plot(nuevo,col=as.numeric(pred)+1,pch="+",xlab="CA",
     ylab="MG",main="LDA Normal")
points(x,pch=19,col=as.numeric(grupo)+1, cex=1.5)

pred <- as.numeric(predict(mcd,nuevo)@classification)
plot(nuevo,col=as.numeric(pred)+1,pch="+",xlab="CA",
     ylab="MG",main="LDA con MCD")
points(x,pch=19,col=as.numeric(grupo)+1, cex=1.5)
```

A.1.5. Trimmed K-Means

```r
library(tclust)
data(geyser2)

kmedias <- kmeans(geyser2, centers=3)
plot(geyser2, col=kmedias$cluster+1, main="K-Means", pch=19)

trimmed <- tkmeans(geyser2, k=3)
plot(geyser2, col=trimmed$cluster+1, main="Trimmed K-Means",
     pch=ifelse(trimmed$cluster==0, 1, 19))
```

A.2. Código utilizado en el Capítulo 3

```r
library(MixMatrix)

n <- 100000
p <- 3
q <- 2

mu <- matrix(1:6,nrow=p,ncol=q)
sigma_row <- 7 * diag(p) + 1
sigma_col <- matrix(c(8,.5,.5,.18),nrow=q)

x <- rmatrixnorm(n, mu, U=sigma_row, V=sigma_col, list=T)

mu_hat <- apply(simplify2array(x), 1:2, mean)
mu_hat

sig_r_hat <- diag(p)
sig_c_hat <- Reduce("+", lapply(x, function(y)
  t(y-mu_hat) %*% solve(sig_r_hat) %*% (y-mu_hat))) / (n*p)
```

```

continuar <- TRUE

while(continuar){
  omega_c <- solve(sig_c_hat)
  sig_r_hat_new <- Reduce("+", lapply(x, function(y)
    (y-mu_hat) %*% omega_c %*% t(y-mu_hat))) / (n*q)

  omega_r <- solve(sig_r_hat_new)
  sig_c_hat_new <- Reduce("+", lapply(x, function(y)
    t(y-mu_hat) %*% omega_r %*% (y-mu_hat))) / (n*p)

  if (norm(sig_r_hat_new - sig_r_hat, type="F") < 1e-9 &&
      norm(sig_c_hat_new - sig_c_hat, type="F") < 1e-9){
    continuar <- FALSE
  }

  sig_r_hat <- sig_r_hat_new
  sig_c_hat <- sig_c_hat_new
}


a <- sig_r_hat[1,1]/sigma_row[1,1]

sig_r_hat <- sig_r_hat / a
sig_c_hat <- sig_c_hat * a
```

A.3. Código utilizado en el Capítulo 4

A.3.1. MMCD

```
library(MixMatrix)
library(doParallel)
registerDoParallel(cores=detectCores()/2)


MLE <- function(x, max.iter = 2147483647){
  n <- length(x)
  p <- nrow(x[[1]])
  q <- ncol(x[[1]])

  mu_hat <- apply(simplify2array(x), 1:2, mean)
  centrados <- lapply(x, function(y) y-mu_hat)

  sig_r_hat_old <- diag(p)
  sig_c_hat_old <- Reduce("+", lapply(centrados, function(y)
    t(y) %*% y)) / (n*p)


  for(i in 1:max.iter){
    omega_c <- solve(sig_c_hat_old)
    sig_r_hat <- Reduce("+", lapply(centrados, function(y)
      y %*% omega_c %*% t(y))) / (n*q)

    omega_r <- solve(sig_r_hat)
    sig_c_hat <- Reduce("+", lapply(centrados, function(y)
      t(y) %*% omega_r %*% y)) / (n*p)

    if ((norm(sig_r_hat - sig_r_hat_old, type="F") < 1e-9 &&
        norm(sig_c_hat - sig_c_hat_old, type="F") < 1e-9)){
      break
    }

    sig_r_hat_old <- sig_r_hat
    sig_c_hat_old <- sig_c_hat
  }
```

```r
  return(list(mu_hat = mu_hat, sigma_row = sig_r_hat,
              sigma_col = sig_c_hat))
}

# Traza de una matriz
tr <- function(z) sum(diag(z))

# Distancias de Mahalanobis al Cuadrado
matrixMahalanobis <- function(x, mu, sigma_row, sigma_col) {
  omega_row <- solve(sigma_row)
  omega_col <- solve(sigma_col)
  centrados <- lapply(x, function(y) y-mu)

  return(sapply(centrados, function(y) tr(omega_col %*%
                                          t(y) %*%
                                            omega_row %*% y)))
}


C_STEP <- function(x, H, epsilon = 1e-9, max.iter = 2147483647){
  n <- length(x)
  p <- nrow(x[[1]])
  q <- ncol(x[[1]])

  estimadoresOld <- MLE(x[H], max.iter = max.iter)

  detcol <- determinant(estimadoresOld$sigma_col)
  detrow <- determinant(estimadoresOld$sigma_row)
  deltaOld <- p*detcol$modulus*detcol$sign +
    q*detrow$modulus*detcol$sign

  h <- floor((n + floor(p/q + q/p) + 2) / 2)

  for(i in 1:max.iter){
    H <- order(matrixMahalanobis(x, estimadoresOld$mu_hat,
                                 estimadoresOld$sigma_row,
                                 estimadoresOld$sigma_col))[1:h]
    estimadores <- MLE(x[H], max.iter = max.iter)

    detcol <- determinant(estimadoresOld$sigma_col)
    detrow <- determinant(estimadoresOld$sigma_row)
    delta <- p*detcol$modulus*detcol$sign +
      q*detrow$modulus*detcol$sign

    if(abs(deltaOld - delta) < epsilon){
      break
    }
    estimadoresOld <- estimadores
    deltaOld <- delta
  }
  if(max.iter == 2){
    return(list(delta = delta, H = H))
  } else{
    return(list(delta = delta, H = H,
                estimadores = estimadores))
  }
}


f_aux <- c("C_STEP", "MLE", "matrixMahalanobis", "tr")

MMCD <- function(x) {
  n <- length(x)
  p <- nrow(x[[1]])
  q <- ncol(x[[1]])
  h <- floor((n + floor(p/q + q/p) + 2) / 2)
  alpha <- h/n

  # Conjuntos Iniciales
  tam_iniciales <- floor(p/q + q/p) + 2
```

```
  iniciales <- foreach(k = 1:500, .export = f_aux,
                       .inorder = FALSE) %dopar%{
      return(C_STEP(x, sample(1:n, tam_iniciales), max.iter=2))
  }
  mejores <- order(sapply(iniciales, function(x) x$delta))[1:10]

  # Iteraciones Hasta Convergencia
  finales <- foreach(l = mejores, .export = f_aux) %dopar% {
    return(C_STEP(x, iniciales[[l]]$H))
  }
  j <- which.min(sapply(finales, function(x) x$delta))

  # Escalado
  f_consist <- alpha * pchisq(qchisq(alpha, df=p*q), df=p*q+2)
  d <- matrixMahalanobis(x, finales[[j]]$estimadores$mu_hat,
                         f_consist *
                           finales[[j]]$estimadores$sigma_row,
                         finales[[j]]$estimadores$sigma_col)
  H <- union(finales[[j]]$H, which(d < qchisq(.975, df=p*q)))

  # Repesado
  resultados <- MLE(x[H])
  alpha_tilde <- length(H)/n
  f_repeso <- alpha_tilde / pchisq(qchisq(alpha_tilde, df=p*q),
                                   df=p*q+2)

  resultados$sigma_row <- f_repeso * resultados$sigma_row
  resultados$sigma_col <- f_repeso * resultados$sigma_col

  return(resultados)
}


### OUTLIERS ###
# Datos normales
n <- 100
p <- 30
q <- 40

mu <- matrix(0,nrow=p,ncol=q)
sigma_col <- diag(q)*.5
sigma_row <- diag(p)*.5

x <- rmatrixnorm(n, mu, U=sigma_row, V=sigma_col, list=T)

# Datos outliers
mu <- matrix(2,nrow=p,ncol=q)
sigma_row <- diag(p) + .2
sigma_col <- diag(q) + .2

x_outliers <- rmatrixnorm(10, mu, U=sigma_row, V=sigma_col, list=T)

# Otros datos outliers
mu <- matrix(-2,nrow=p,ncol=q)
sigma_row <- 5*diag(p)
sigma_col <- 5*diag(q)

x_outliers2 <- rmatrixnorm(10, mu, U=sigma_row, V=sigma_col, list=T)

# Juntamos los datos
x <- c(x, x_outliers,x_outliers2)

# Aplicamos MMCD para obtener estimadores
mmcd_outliers <- MMCD(x)

par(mfrow=c(1,2))

# Calculamos distancias
d <- matrixMahalanobis(x, mmcd_outliers$mu_hat, mmcd_outliers$sigma_row,
    mmcd_outliers$sigma_col)
```

```
plot(d, col=ifelse(d < qchisq(.975, df=p*q), "blue", "red"), xlab="Observaciones",
     ylab="Distancia de Mahalanobis",
      main = "MMCD")
abline(h=qchisq(.975, df=p*q), col="darkgreen")


# Con el metodo tradicional
estimadores <- MLE(x)
d <- matrixMahalanobis(x, estimadores$mu_hat, estimadores$sigma_row,
     estimadores$sigma_col)
plot(d, col=ifelse(d < qchisq(.975, df=p*q), "blue", "red"), xlab="Observaciones",
     ylab="Distancia de Mahalanobis",
      main = "MLE")
abline(h=qchisq(.975, df=p*q), col="darkgreen")
```

A.3.2. Vídeo

```
library(magick)

video2List <- function(input){
  video <- image_convert(image_read_video(input, fps=NULL), type="grayscale")
  video <- lapply(video, image_data)
  video <- lapply(video, drop)

  return(foreach(i=1:length(video)) %dopar% {
    return(apply(video[[i]], 2, function(x) strtoi(x,base = 16)))
  })
}

imagenes <- video2List("videos/video2_supersmall.mp4")

mmcd <- MMCD(imagenes)

# Calculamos distancias
d <- matrixMahalanobis(imagenes, mmcd$mu_hat, mmcd$sigma_row, mmcd$sigma_col)
plot(d, col=ifelse(d < qchisq(.975, df=256*144), "blue", "red"), xlab="Observaciones",
     ylab="Distancia de Mahalanobis", main="Distancias de los Cuadros del Video")
abline(h=qchisq(.975, df=256*144), col="green")
```

A.4. Código utilizado en el Capítulo 6

```
library(MixMatrix)
library(doParallel)
registerDoParallel(cores=detectCores()/2)


# Funcion objetivo a maximizar
f_obj <- function(x, pesos, estimadoresGrupos){
  return(sum(unlist(delta(x, pesos, estimadoresGrupos)$deltaMax)))
}

# Calcula los deltas de todas las observaciones para un grupo
deltaGrupo <- function(x, peso, mu, sigmaRow, sigmaCol){
  p <- nrow(x[[1]])
  q <- ncol(x[[1]])

  detcol <- determinant(sigmaCol)
  detrow <- determinant(sigmaRow)

  return(log(peso) - q/2 * detrow$modulus*detrow$sign - p/2 * detcol$modulus*detcol$sign
    -
    1/2 * matrixMahalanobis(x, mu, sigmaRow, sigmaCol) - p*q*log(pi))
}

# Delta global, devuelve para cada observacion su delta maximo y el grupo con el que
```

```
# lo consigue.
delta <- function(x, pesos, estimadoresGrupos){
  k <- length(pesos)

  deltas <- vector("list", k)
  for(j in 1:k){
    deltas[[j]] <- deltaGrupo(x, pesos[[j]], estimadoresGrupos[[j]]$mu,
        estimadoresGrupos[[j]]$sigma_row, estimadoresGrupos[[j]]$sigma_col)
  }

  result <- list()
  # Para cada observacion, su deltaMax y su grupo asignado es:
  for(i in 1:length(x)){
    deltas_i <- sapply(deltas, function(delta) delta[[i]])
    result$deltaMax[i] <- max(deltas_i)
    result$grupo[i] <- which.max(deltas_i)
  }

  return(result)
}

# TCLUST - Bucle - El mismo en ambas fases
tclustUpdate <- function(x, pesos, estimadoresGrupos, alpha, max.iter = 10){
  n <- length(x)
  p <- nrow(x[[1]])
  q <- ncol(x[[1]])
  h <- floor(n*(1-alpha))
  k <- length(pesos)

  objetivoOld <- f_obj(x, pesos, estimadoresGrupos)
  for(iter in 1:max.iter){
    deltas <- delta(x, pesos, estimadoresGrupos)

    # Eliminamos las h observaciones con menor deltaMax
    grupo_0 <- order(deltas$deltaMax)[1:(n-h)]
    restoIndex <- order(deltas$deltaMax)[(n-h+1):n]

    # Para el resto, las asignamos al grupo con mayor delta
    grupos <- vector("list", k)

    for(i in restoIndex){
      asignacion <- deltas$grupo[i]
      grupos[[asignacion]] <- c(grupos[[asignacion]], i)
    }

    # Actualizar estimadores
    pesos <- sapply(grupos, function(y) length(y)/h)   # Comentar si pesos sin
        actualizar 1/k
    for(i in 1:k){
      estimadoresGrupos[[i]] <- MLE(x[grupos[[i]]], max.iter = max.iter)
    }

    # Calcular funcion objetivo
    objetivo <- f_obj(x, pesos, estimadoresGrupos)
    if(abs(objetivoOld - objetivo) < 1e-9){
      break
    }
    objetivoOld <- objetivo
  }
  return(list(obj = objetivo, grupos = grupos, pesos = pesos, estimadoresGrupos =
      estimadoresGrupos, grupo_0 = grupo_0))
}

# TCLUST - Funcion Principal
f_aux <- c("delta", "deltaGrupo", "f_obj", "tclustUpdate","C_STEP", "MLE",
    "matrixMahalanobis", "tr")
tclust <- function(x, k = 3, alpha = .05, nstart = 4000){
  n <- length(x)
  p <- nrow(x[[1]])
  q <- ncol(x[[1]])
```

```
  tamanoGrupoInicial <-  floor(p/q + q/p) + 2

  # Fase 1 - Conjuntos Iniciales
  iniciales <- foreach(inicial = 1:nstart, .export = f_aux, .inorder = FALSE) %dopar%{
    pesos <- rep(1/k, k)
    grupos <- list()
    estimadoresGrupos <- list()

    asignaciones <- sample(c(rep(seq(1,k), each = tamanoGrupoInicial), rep(0, n -
        tamanoGrupoInicial*k)))
    for(i in 1:k){
      grupos[[i]] <- which(asignaciones == i)
      estimadoresGrupos[[i]] <- MLE(x[grupos[[i]]], max.iter = 4)
    }

    return(tclustUpdate(x, pesos, estimadoresGrupos, alpha, max.iter = 4))
  }

  # Fase 2 - 10 mejores soluciones
  mejores <- order(sapply(iniciales, function(x) x$obj), decreasing = TRUE)[1:10]

  finales <- foreach(l = mejores, .export = f_aux) %dopar% {
    return(tclustUpdate(x, iniciales[[l]]$pesos, iniciales[[l]]$estimadoresGrupos,
        alpha))
  }

  # Encontrar el mejor
  j <- which.max(sapply(finales, function(x) x$obj))

  assig <- rep(0,n)
  for (i in 1:k){
    assig[finales[[j]]$grupos[[i]]] <- i
  }

  return(list(obj = finales[[j]]$obj, grupos = finales[[j]]$grupos,
              pesos = finales[[j]]$pesos, estimadores = finales[[j]]$estimadoresGrupos,
              grupo_0 = finales[[j]]$grupo_0,assig=assig))
}


### EJEMPLO ###
set.seed(1)

p <- 10
q <- 18

mu <- matrix(0,nrow=p,ncol=q)
sigma_row <- 0.25*(diag(p) + 1)
sigma_col <- 0.25*(diag(q) + 1)
x <- rmatrixnorm(100, mu, U=sigma_row, V=sigma_col, list=T)

mu <- matrix(2,nrow=p,ncol=q)
sigma_row <- 0.25*(diag(p) + 1)
sigma_col <- 0.25*(diag(q) + 1)
x <- c(x, rmatrixnorm(100, mu, U=sigma_row, V=sigma_col, list=T))

mu <- matrix(-2,nrow=p,ncol=q)
sigma_row <- 0.25*(diag(p) + 1)
sigma_col <- 0.25*(diag(q) + 1)
x <- c(x, rmatrixnorm(100, mu, U=sigma_row, V=sigma_col, list=T))

a <- tclust(x)

# plot(delta(x, a$pesos, a$estimadores)$grupo)
# plot(delta(x, a$pesos, a$estimadores)$deltaMax)

table(a$assig,rep(1:3,each=100))
```