Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Estadística

Análisis cluster robusto con contaminación por celdas

Autor:

D. David Población Criado

Tutor:

D. Luis Ángel García Escudero

Agradecimientos

Me gustaría dedicar unas palabras de agradecimiento a todas las personas que me han ayudado a lo largo de la realización de este Trabajo Fin de Grado.

En primer lugar, a los profesores del Grado en Estadística por haberme proporcionado los conocimientos necesarios para comenzar mi carrera profesional.

A Luis Ángel, por la confianza que ha puesto en mí, su constante dedicación y ayuda.

A mi familia, en especial a mis padres, por su apoyo, comprensión y paciencia.

Por último, a todas aquellas personas que me han acompañado durante estos últimos años, compañeros y amigos por estar siempre disponibles para lo que fuera necesario, gracias por tanto.

Resumen

Es frecuente en la práctica habitual de la Estadística contar con medidas atípicas que pueden afectar muy negativamente a los procedimientos estadísticos aplicados. Para resolver esta problemática en problemas de análisis de datos multivariantes, es común recurrir al recorte de observaciones completas. Por tanto, filas enteras de la matriz de datos son recortadas cuando se detecta algún valor atípico en alguna de las celdas o mediciones que conforman dichas filas de la matriz de datos. Desgraciadamente, este tipo de recorte por observaciones también sacrifica la información de las celdas no atípicas en las filas recortadas. En este Trabajo Fin de Grado se han tratado aspectos computacionales para nuevos métodos de Análisis Cluster que buscan recortar solo las celdas atípicas dentro de la matriz de datos. Este enfoque es más adecuado en los problemas de dimensionalidad no necesariamente baja que ocurren en la práctica moderna de la Estadística.

Abstract

It is common in the usual practice of Statistics to have atypical measures that can have a very negative effect on the statistical procedures applied. To solve this problem in multivariate data analysis problems, it is common to resort to trimming entire observations. Thus, entire rows of the data matrix are trimmed when an outlier is detected in any of the cells or measurements that make up these rows of the data matrix. Unfortunately, this type of trimming by observations also sacrifices the information of the non-outlier cells in the trimmed rows. In this Bachelor's Thesis, computational aspects for new Cluster Analysis methods that seek to trim only the outlier cells within the data matrix have been addressed. This approach is best suited to problems of not necessarily low dimensionality that occur in modern statistical practice.

Índice general

Capítulo 1

Introducción

Es bien conocido que unas pocas observaciones atípicas pueden afectar muy negativamente al resultado y a la calidad del ajuste de muchos métodos estadísticos clásicos. La Estadística Robusta proporciona técnicas que "resisten" a estos valores atípicos como es, por ejemplo, "recortar" de manera sistemática y automatizada las observaciones más extremas, eliminándolas del ajuste. Sin embargo, este procedimiento de recorte de observaciones completas puede, en muchas ocasiones, no ser el más adecuado, concretamente si nos encontramos en problemas con una alta dimensionalidad porque se estaría también sacrificando información valiosa de celdas no atípicas que son recortadas junto a las celdas atípicas.

Dado lo frecuente que resulta tratar situaciones con moderada y alta dimensionalidad en la Estadística actual, el presente Trabajo Fin de Grado (en adelante TFG) tratará la aplicación de técnicas de recorte en problemas de Análisis Cluster donde se tratan de eliminar exclusivamente las celdas atípicas de la matriz de datos en lugar de eliminar observaciones completas. Con este propósito, se adaptan técnicas de recortes para problemas de moderada y alta dimensionalidad, en las cuales las observaciones se suponen agrupadas en torno a subespacios afines con dimensiones bastante menores a la dimensión del espacio de partida. Se han tratado en profundidad distintos problemas computacionales asociados a este nuevo enfoque. En primer lugar, se presentará la metodología adoptada procedente de [1], para después realizar una serie de mejoras computacionales y añadir distintas funcionalidades. Estas modificaciones han sido analizadas bajo ciertos esquemas de simulación sobre datos generados artificialmente. Por último, se aplicará el procedimiento a dos conjuntos de datos para mostrar que las modificaciones realizadas tienen un uso en la vida real.

1.1. Herramientas utilizadas

A continuación se enumera el *software* utilizado para la realización del trabajo:

- *R*: lenguaje de programación interpretado en el que se ha realizado la mayor parte del proyecto dado su extendido uso en estadística.
- *C++*: lenguaje de programación de ámbito general, compilado y orientado a objetos. Se ha

utilizado para implementar el algoritmo FAST-LTS por su velocidad de cálculo, lo cual ha sido crítico para el rendimiento general del procedimiento.

- *RStudio*: entorno de desarrollo integrado para programar, depurar y ejecutar código en R.
- *Visual Studio Code*: entorno de desarrollo integrado de ámbito general, utilizado en este trabajo para programar en C++ al tener una interfaz más cómoda en este lenguaje que RStudio.
- *Git*: sistema de control de versiones para llevar un seguimiento de la modificación del código.
- *TeXstudio*: editor de LaTeX utilizado para la escritura de la presente memoria.
- *Mendeley*: gestor bibliográfico para almacenar y procesar las referencias del trabajo, así como mantener toda la documentación ordenada.

1.2. Estructura de la memoria

Este documento sigue la siguiente estructura:

Capítulo 2 - Metodología. Se realiza una revisión de conceptos básicos necesarios para comprender la metodología, como son el Análisis en Componentes Principales o los métodos Análisis Cluster combinados con ideas de la Estadística Robusta. Acto seguido, se explica la metodología de recorte por celdas en Análisis Cluster y el marco teórico que se va a seguir.

Capítulo 3 - Implementación. Tiene un carácter más técnico, en el que se detallan las propuestas de mejora y funcionalidades añadidas a la metodología presentada y cómo se han implementado estas modificaciones en los lenguajes de programación R y C++.

Capítulo 4 - Ejemplos y simulación. Consta de distintos esquemas de simulación para probar que las modificaciones y funcionalidades añadidas son de utilidad sobre datos generados artificialmente. También se analiza el efecto de la elección de diversos parámetros en el funcionamiento de la metodología.

Capítulo 5 - Aplicación sobre datos reales. Se muestra el uso del algoritmo implementado sobre dos conjuntos de datos reales, los primeros sobre la mortalidad de hombres en Francia y el segundo un análisis de las temperaturas correspondientes a diversos puntos de la geografía española.

Capítulo 6 - Conclusiones y líneas futuras. Reflexión final sobre el trabajo desarrollado y planteamiento de futuras líneas de investigación.

Apéndice A - Código fuente. Como apéndice al documento se muestra la programación de las funciones implementadas.

Capítulo 2

Metodología

La metodología en este trabajo surge de combinar ideas de Análisis Cluster [2] y Estadística Robusta [3]. En particular, la aproximación de Análisis Cluster adoptada tiene que ver con enfoques especialmente diseñados para problemas de alta y moderada dimensionalidad en los que se asume que las observaciones se agrupan en torno a G subespacios lineales con dimensiones notablemente menores a la dimensión p del espacio original en el que viven los datos. Esta suposición suele ser razonable cuando existen estructuras de dependencia entre las variables que hacen que la dimensionalidad "intrínseca" del problema en cada uno de los grupos sea reducida y que el modelado de estas dependencias, especificas a los clusters, sea útil a la hora de detectar esos mismos grupos. Este tipo de Análisis Cluster es conocido en la literatura como "subspace clustering" y reduciría al bien conocido método del Análisis de Componentes Principales (ACP) cuando se asume que todas las observaciones están conjuntamente agrupadas en torno a un único subespacio lineal.

Es bien sabido que unas pocas observaciones atípicas pueden afectar muy negativamente a muchos métodos estadísticos clásicos que son típicamente aplicados, obteniéndose resultados poco interesantes e incluso completamente equivocados. Por otro lado, la Estadística Robusta proporciona técnicas alternativas que son más "resistentes" a esas observaciones atípicas y métodos que, en general, son menos dependientes de satisfacer muy exactamente las estrictas suposiciones del modelo asumido.

Entre estas técnicas estadísticas robustas, un enfoque muy aplicado y muy fácil de entender es proceder a "recortar" de forma automatizada las mediciones más extremas (las más susceptibles de ser atípicas) y posteriormente aplicar los procedimientos clásicos sobre las observaciones no recortadas. De esta forma se consigue que posibles observaciones atípicas no afecten al resultado final del procedimiento. Este aproximación a la robustez es conocida como técnicas de "recorte" y la media α-recortada sería un sencillo ejemplo de este enfoque.

Cómo decidir de forma automatizada la mejor forma de recortar observaciones en problemas multivariantes no es un problema sencillo. Tradicionalmente, se considera la posibilidad de recortar observaciones enteras (es decir, filas completas de la matriz de datos). En este TFG trataremos la aplicación de técnicas de recortes en problemas de Análisis Cluster en torno a subespacios, pero eliminando celdas (entradas individuales) de la matriz de datos, sin necesidad de tener que

eliminar observaciones (filas de la matriz) completas.

La primera aproximación teórica a esta nueva metodología de buscar robustez a contaminación por celdas se puede encontrar en [4]. Hay muchos trabajos posteriores en esta interesante línea de trabajo, pero en la mayoría de casos (salvo alguna excepción como puede ser Farcomeni, A. (2014). Snipping for robust k-means clustering under component-wise contamination. Statistics and Computing, 24(6), 907-919.) se asume una única estructura subyacente de datos, es decir, $G = 1$.

En este trabajo se parte de la propuesta realizada en [1] donde se asumen $G > 1$ estructuras diferentes para los clusters y donde las observaciones, previamente al proceso de contaminación, se encuentran agrupadas en torno a G subespacios lineales, quizás de dimensiones q_g, para $g = 1, ..., G$, bastante menores que la dimensión p del problema original y donde se propuso la realizaron de recortes por celdas. Nótese que el caso $G = 1$ sería equivalente a hacer Análisis en Componentes Principales con recortes por celdas, para los que ya existían propuestas en la literatura.

2.1. Conceptos y metodología de partida

Para introducir la metodología y, también, para introducir la notación utilizada, comenzaremos revisando distintos conceptos y metodologías en las que se basa la propuesta metodológica.

2.1.1. Autovalores y autovectores

Dada una matriz cuadrada A de dimensión $p \times p$, se dice que x es un autovector de A si existe $\lambda \in \mathbb{C}$ tal que:

$$Ax = \lambda x \tag{2.1}$$

En este caso, λ se denomina autovalor de A. Los autovectores no son únicos: si $Ax = \lambda x$, entonces $A(cx) = \lambda(cx)$, para algún escalar $c \in \mathbb{C}$. Muchas veces, para solventar este problema se usa una constante de normalización sobre los autovectores, la más común es restringir que $x^t x = \|x\|^2 = 1$, es decir, que tengan longitud unitaria.

De forma equivalente, (2.1) se puede reformular como:

$$(A - \lambda I)x = 0 \tag{2.2}$$

Esto implica que $\det(A - \lambda I) = 0$, lo cual es un polinomio de grado n en λ, llamado polinomio característico de A. Sus raíces pertenecen a $\mathbb{C}$ y pueden ser tanto algunas reales como otras complejas.

El procedimiento es en primer lugar hallar los autovalores de A, resolviendo el polinomio característico. Después, basta resolver 2.2 sustituyendo λ por cada uno de los autovalores obtenidos, lo cual dará lugar a un conjunto denominado espacio característico asociado a λ:

$$V_\lambda = \{x \in \mathbb{C}/(A - \lambda I)x = 0\} \tag{2.3}$$

el cual es un subespacio vectorial en $\mathbb{C}^p$. Finalmente, para obtener los autovectores asociados a λ hay que determinar una base de dicho subespacio. El número máximo de autovectores linealmente independientes asociados a un mismo autovalor viene dado por el espacio característico y se denomina multiplicidad geométrica del autovalor.

Existen diferentes métodos numéricos para obtener de forma aproximada los autovectores y autovalores.

2.1.2. Proyecciones ortogonales y subespacios asociados

Dos vectores $u = (u_1, ..., u_p)^t$ y $v = (v_1, ..., v_p)^t$ que pertenecen al mismo espacio vectorial V son ortogonales si su producto interior en dicho espacio es 0, $\langle u, v\rangle = 0$. El producto interior que se va a usar es el euclídeo en $\mathbb{R}^p$:

$$\langle x, y\rangle = \sum_{i=1}^{p} x_i y_i \tag{2.4}$$

Este concepto es aplicable para subespacios. Sea $W \subset V$, entonces v es ortogonal a W si v es ortogonal a todos los vectores que componen W. El subespacio compuesto por los vectores v que cumplen lo anterior se denomina el complementario ortogonal a W y se denota por $W^{\perp}$. Estos dos subespacios verifican lo siguiente:

- $\dim(W) + \dim(W^{\perp}) = p$
- Para cualquier $y \in \mathbb{R}^p$, dicho vector tiene una descomposición ortogonal única tal que $y = y_1 + y_2$, donde $y_1 \in W$ es la proyección ortogonal e $y_2 \in W^{\perp}$ es la componente ortogonal (2.1)

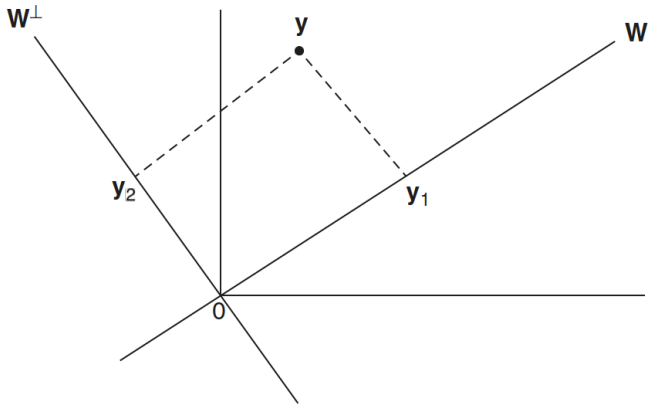


Figura 2.1: Descomposición ortogonal

La proyección Pr_W debe cumplir que:

- Para cualquier $y \in W$, tenemos que $\mathrm{Pr}_W(y) = y$
- Para cualquier $y \in W^{\perp}$, tenemos que $\mathrm{Pr}_W(y) = 0$

Con ello se consigue que, al aplicar dicha proyección a un vector del subespacio, se anule la componente ortogonal y quede únicamente la proyección. Para un subespacio concreto esta matriz de proyección es única y todos sus autovalores valen 0 o 1.

Si W, por abuso de lenguaje, denota a una matriz que contiene por columnas a $\dim(W)$ vectores siendo una base del subespacio W entonces se puede ver que

$$\mathrm{Pr}_W = W(W^tW)^{-1}W^t,$$

y si elegimos una base ortonormal (vectores ortogonales y con norma unitaria) de W entonces

$$\mathrm{Pr}_W(y) = WW^ty.$$

2.1.3. Robustez

Generalmente en los conjuntos de datos que se usan existen valores que son atípicos, los cuales pueden deberse a diversas causas distintas: a la variabilidad de la muestra, a un error experimental,... Dependiendo de la naturaleza de estos se suele optar por eliminarlo o corregirlo (si es posible). Muchos modelos estadísticos se ven altamente influenciados por los valores atípicos dado que intentan ajustar el modelo a todos los datos disponibles, sin hacer distinción entre aquellos que sean atípicos o "outliers" y los que sean correctos.

Para solucionar este posible problema en el ajuste existen una serie de estadísticos que se expondrán a continuación que no son tan sensibles a los valores extremos. De hecho, como ya se ha comentado, los estimadores robustos permiten alejarnos algo de las (rígidas) hipótesis que exigen muchas metodologías e ideas estadísticas, pero obteniendo una eficiencia razonable.

A continuación se presentan, a modo de ejemplo, algunos estimadores clásicos sencillos y sus versiones robustificadas:

2.1.3.1. Media y mediana

Dado $\{x_i\}_{i=1,\dots,n}$ una secuencia de observaciones, la media muestral se define como:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2.5}$$

Por otra parte, la mediana se calcula como:

$$\mathrm{Med}\{x_i\}_{i=1}^{n} = \begin{cases} x_{(\frac{n+1}{2})} & \text{si } n \text{ es impar} \\ \frac{x_{(\frac{n}{2})}+x_{(\frac{n}{2}+1)}}{2} & \text{si } n \text{ es par} \end{cases} \tag{2.6}$$

Ambos estadísticos son una medida de localización para el conjunto de datos, pero con algunas diferencias, sobre todo desde el punto de vista de la robustez. Para ilustrar la diferencia entre ambos estimadores se propondrá un ejemplo sencillo: imaginemos que tenemos una muestra con los datos

$\{1, 2, 3, 5, 7, 9, 10\}$. La media muestral (aritmética) sería de 5.286 y la mediana de 5, ambos valores razonables y muy próximos entre sí. Ahora añadimos un nuevo valor, 1000, cambiando la media por 129.625 y la mediana por 6. Claramente se aprecia que un pequeño cambio en el conjunto de datos ha provocado cambios muy grandes en la media, mientras que la mediana se ha modificado solamente en una unidad. Por ello, la media no es un estadístico robusto mientras que la mediana sí, siempre y cuando no se alteren más de la mitad de los datos, lo cual es improbable.

2.1.3.2. Desviación típica y MAD

Tanto la desviación típica como el *median absolute deviation* (MAD) son estadísticos para medir la dispersión o variabilidad de los datos. La desviación típica se define como:

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{2.7}$$

mientras que el MAD se calcula como:

$$\text{MAD} = 1.4826 \cdot \text{Med}\{|x_i - \text{Med}\{x_i\}_{i=1}^{n}|\}_{i=1}^{n}, \tag{2.8}$$

donde la constante 1.4826 coincide con $1/\Phi^{-1}(\frac{3}{4})$ y garantiza que el valor esperado del MAD coincida con la desviación típica cuando los datos sean normalmente distribuidos.

Fijándonos en la fórmula de la desviación típica, se puede comprobar que al calcular las distancias con respecto a la media estas se elevan al cuadrado, lo que provoca que las desviaciones más extremas tengan muy alta importancia al calcular s. Sin embargo, estas observaciones más extremas tienen un nulo impacto en el MAD, al estar basado en medianas.

2.2. Métodos de clustering y método de k-medias

El Análisis Cluster o los métodos de clustering son técnicas estadísticas orientadas a crear grupos o "clusters" en nuestros datos. Se quieren agrupar n individuos en G grupos, de tal forma que los individuos dentro de un mismo grupo sean lo más "homogéneos" posible y lo más "diferente" posible respecto a los individuos en otros grupos. Los métodos de clustering se suelen así incluir dentro de las técnicas de aprendizaje no supervisado donde etiquetas de grupos no están definidos inicialmente sobre las observaciones y el objetivo es, justamente, crear estas etiquetas.

Dado que la definición de cluster es imprecisa, se pueden diferenciar varios tipos de metodologías, entre los que se encuentran los métodos:

- **Jerárquicos**: la pertenencia de una observación a un grupo condiciona su posible pertenencia en otros niveles superiores. Dependiendo de cómo se construya esta jerarquía, existen dos métodos: *aglomerativos*, donde partimos de tantos clusters como observaciones, los cuales

vamos agrupando; o *divisivos*, donde empezamos teniendo un solo grupo que vamos dividiendo progresivamente. Realmente no se construye una sola partición, sino que dependiendo del nivel de la jerarquía en la que estemos tenemos una partición diferente.

- **Particionales**: la partición en grupos que se obtiene es única mediante la optimización de una función, al contrario que en los jerárquicos que obteníamos una sucesión de particiones encajadas.

En ambos métodos es necesario usar un criterio para realizar estas particiones, por lo que se define una distancia o una disimilaridad entre individuos. En este trabajo usaremos la distancia euclidea.

Para introducir nuestra metodología es especialmente útil conocer el algoritmo de las k-medias, el cual se mencionará posteriormente en distintos puntos y diversas modificaciones de este algoritmo subyacen en muchos de los algoritmos presentados.

El *método de k-medias* es el algoritmo particional (es necesario fijar al principio el número de grupos G a buscar) sin duda más utilizado. Cada observación quedará asignada a un solo grupo y ninguna observación se queda sin clasificar. El objetivo de este algoritmo es obtener G centros óptimos $m_1^*, ..., m_G^*$ tales que

$$\{m_1^*, ..., m_G^*\} = \arg \min_{m_1,...,m_G} \sum_{i=1}^{n} \min_{j=1,...,G} \|x_i - m_j\|^2 \tag{2.9}$$

con $m_1^*, ..., m_G^*$ siendo G puntos en $\mathbb{R}^p$.

El algoritmo de k-medias típicamente para tratar de buscar estos G centroides óptimos $m_1^*, ..., m_G^*$ es el siguiente:

1. Se define G como el número de clusters que queremos obtener y se seleccionan aleatoriamente G centros iniciales $m_1^0, ..., m_G^0$ de los datos originales.

2. Calculamos la distancia elegida entre cada observación de los datos y todos los centros. Cada observación se clasificará en el grupo correspondiente al centro con la menor distancia.

3. Se calcula la media de cada uno de los grupos encontrados para obtener los nuevos centros de cada cluster $m_1^1, ..., m_G^1$.

4. Se repiten los pasos 2 y 3 hasta que se estabilizan las particiones.

Cada observación queda asignada al cluster más cercano, por lo que se cumple que

$$\text{Cluster } J = \{x_i : \|x_i - m_J^*\| \leq \|x_i - m_j\| \text{ para } j \neq J\} \tag{2.10}$$

donde J es el grupo al que hemos asignado x_i.

Este algoritmo presenta una serie de problemas que hace que no sea necesariamente el más adecuado en muchos casos. En primer lugar, tiende a buscar clusters esféricos, por lo que si los datos

no cumplen esta premisa lo más probable es que no encuentre los grupos correctamente. Además, tiende a buscar particiones balanceadas, con el mismo número de elementos por partición, lo cual no siempre es el caso. Otra importante problema es su falta de robustez respecto a observaciones x_i atípicas [5].

2.3. Análisis en Componentes Principales

El Análisis en Componentes Principales (ACP) es una estrategia desarrollada por Pearson por primera vez [6] para encontrar una aproximación óptima en un subespacio lineal aproximante de dimensión inferior q con $q < p$. Después fue modificado por Hotelling [7] como la transformación de la matriz X de datos $n \times p$ en otra matriz de tamaño $n \times q$ de tal forma que las columnas de esta nueva matriz se obtengan mediante combinaciones lineales de las columnas de la matriz original X y de tal forma que esas columnas sean muestralmente incorreladas y ordenadas de forma decreciente en variabilidad. El enfoque de "aproximación" formulado por Pearson va a jugar un papel clave en nuestra presentación.

El ACP trata así de encontrar relaciones entre las distintas variables del conjunto de datos, consiguiendo reducir la dimensionalidad de este principalmente (tiene otras aplicaciones que se detallarán más adelante). Por tanto, no es un procedimiento para predecir una variable a partir de otras, no hay variable respuesta. Es un método que se aplica a variables continuas y suele aplicarse como paso previo a otras técnicas de predicción.

Como hemos comentado, en términos generales, el ACP crea combinaciones lineales con pesos de las variables originales intentando que las nuevas dimensiones expliquen la mayor proporción de varianza posible. La Figura 2.2 muestra un ejemplo básico de datos en $\mathbb{R}^2$, el cual parece que la segunda dimensión no aporta demasiado y podría ser representado únicamente en una dimensión, la dada por el vector u_1. Esto es lo que va a buscar el análisis en componentes principales: la (o las) dirección que recoge la mayor cantidad de información posible en el sentido de la varianza (la separación entre los distintos puntos). Por ello en el ejemplo la varianza de la dimensión dada por u_2 es muy pequeña en comparación con la de u_1. En el caso de 2 dimensiones es fácil de ver gráficamente, el problema es cuando aumentamos el número de dimensiones y no es visible.

Supongamos que X es nuestra matriz de datos $n \times p$ ($X \in \mathbb{R}^{n \times p}$) con p igual al número de variables analizadas (por columnas) para n individuos diferentes en sus n filas que denotamos por $x_i = (x_{i1}, ..., x_{ip})^t$ y x_{ij} denota los elementos o "celdas" de esa matriz X. Se tiene que

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

es la media muestral y

$$S = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^t$$

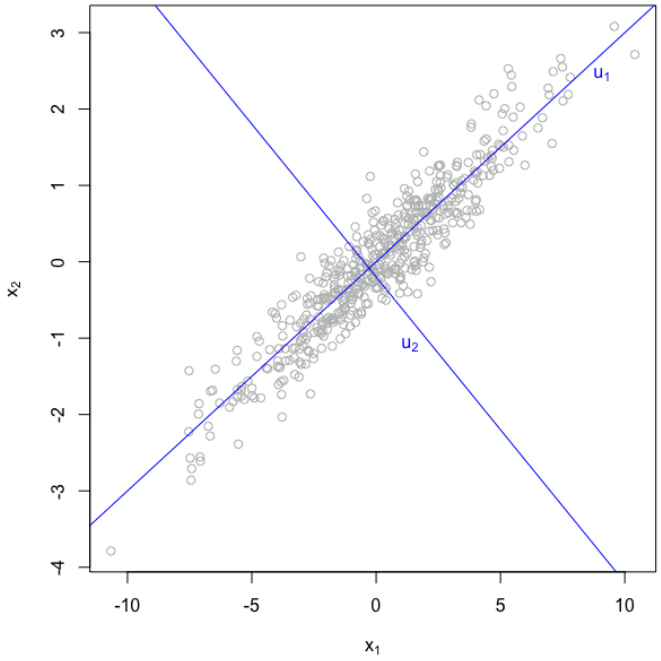


Figura 2.2: Ejemplo con datos bidimensionales

es la matriz de varianzas-covarianzas muestral basada en nuestros datos en X.

Para obtener la mejor aproximación al conjunto de datos buscamos una matriz $B \in \mathbb{R}^{p\times q}$ de vectores ortonormales b_j^t para $j = 1, ..., p$ y una matriz de coordenadas o "scores" (dado que B_q es una matriz ortonormal) que denotamos $A_q \in \mathbb{R}^{n\times q}$ donde a_i, $i = 1, ..., n$, corresponde a la fila i-ésima de la matriz A_q. Por último tenemos un vector $m \in \mathbb{R}^p$ de medias, el cual coincidirá con $\bar{x}$, la media muestral de los datos.

Usando esta notación, la aproximación a la observación i-ésima de X viene dada por

$$\hat{x}_i = m + B_q a_i \tag{2.11}$$

Elemento a elemento se puede reformular de la siguiente forma:

$$\hat{x}_{ij} = m_j + a_i^t b_j \tag{2.12}$$

El subespacio que mejor aproxima a los datos vendrá dado por la minimización de

$$\min_{B_q, A_q, m} \sum_{i=1}^{n} ||x_i - \hat{x}_i||^2 \tag{2.13}$$

donde $||x_i - \hat{x}_i||^2$ es el cuadrado de la distancia de la observación x_i a su proyección ortogonal denotada por $\hat{x}_i$ en el subespacio aproximante determinado por m y las columnas de la matriz B_q. Esta distancia de la observación x_i a su proyección ortogonal en el subespacio aproximante será también denotada, en lo que sigue, por

$$d_i^2(B_q, A_q, m).$$

Es bien sabido, que la solución de este problema de minimización viene dada por los autovectores de la matriz de varianzas-covarianzas S (teniendo en cuenta el tamaño de los autovalores asociados). Sea pues B_q una matriz ortonormal tal que, usando descomposición de valores singulares, verifica:

$$B_q^t S B_q = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_q \end{pmatrix}, \tag{2.14}$$

donde

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_q$$

son los q autovalores mayores de S.

Entonces, la solución de (2.13) está dada por B_q, $m = \bar{x}$ y A_q cuyas filas son $a_i^t = (x_i - m)^t B_q$, $i = 1, ..., n$.

2.4. Metodologías de "Least Trimmed Squares"

El análisis en componentes principales es muy sensible a valores atípicos. Esto no es sorprendente dado que está basado en un procedimiento de minimización de sumas de cuadrados, como eran la media y la regresión, que es bien sabido que son poco robustos. Encaminado a realizar ACP robusto se han propuesto múltiples soluciones como son *Principal Component Pursuit* (PCP), [8], *Stable PCP* [9] o el *Local PCP* [10].

La propuesta que se va a usar serán los "mínimos cuadrados recortados" o, en inglés, *Least Trimmed Squares* (LTS) [11], [12]. En primer lugar se va a exponer desde el punto de vista de la regresión y a continuación se aplicará al caso del ACP.

El modelo de regresión lineal clásico viene dado por el modelo

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \varepsilon_i, \tag{2.15}$$

donde ε_i se supone que son errores independientes normalmente distribuidos y varianza constante (se suele tomar $x_{i1} = 1$ para todo i de tal forma que β_1 sería el "intercept" de la regresión).

El objetivo del algoritmo de mínimos cuadrados clásico (*Ordinary Least Squares* OLS) es minimizar la suma de los residuales al cuadrado, es decir:

$$\min_{\beta} \sum_{i=1}^{n} r_i^2 = \min_{\beta} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2, \tag{2.16}$$

donde $\hat{y}_i = \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + ... + \hat{\beta}_p x_{ip}$ es la predicción de la observación y_i. El problema de este método es, como ya se ha comentado, es que es muy sensible a valores atípicos. Es decir, unas

pocas observaciones extremas pueden variar de forma muy considerable el ajuste.

La regresión LTS modifica la función objetivo basándose en la ordenación de los residuales:

$$(r^2)_{1:n} \leq (r^2)_{2:n} \leq ... \leq (r^2)_{n:n}, \tag{2.17}$$

y usando en la función objetivo únicamente los $h \leq n$ residuales menores:

$$\min_{\beta} \sum_{i=1}^{h} (r^2)_{i:n}. \tag{2.18}$$

Se suele reformular h en términos de la proporción de observaciones eliminadas α teniendo

$$h = \lceil n(1-\alpha) \rceil \tag{2.19}$$

donde α denota el tamaño de recorte.

Para ilustrar la mejora de LTS sobre el algoritmo clásico de mínimos cuadrados (OLS) generaremos un conjunto de datos bidimensionales, algunos de los cuales han sido contaminados modificando una de las variables para que sigan un comportamiento claramente diferenciado (Figura 2.3).

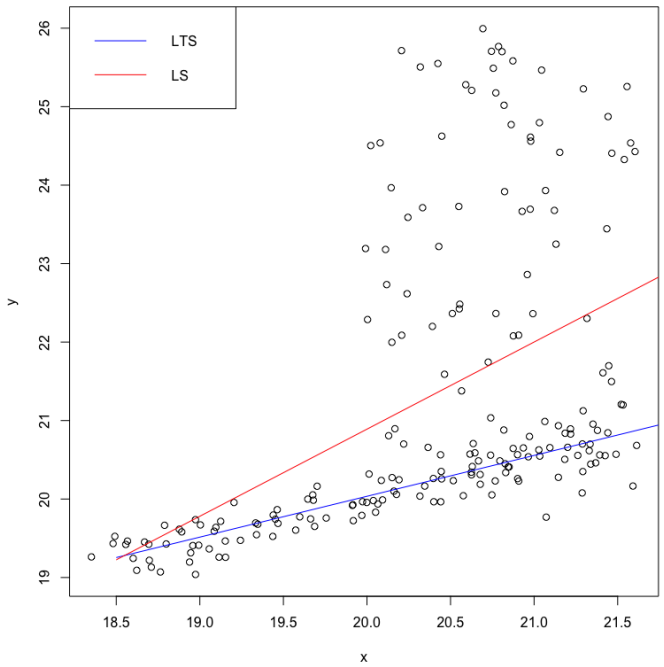


Figura 2.3: Comparación de LTS y LS sobre datos contaminados generados sintéticamente

De forma clara se ve que la regresión OLS no es robusta puesto que queda muy afectada por estos valores atípicos. En cambio la regresión LTS sí que resulta robusta porque no los tiene en cuenta dado que sus residuales son muy grandes. Se ha tomado $\alpha = \frac{1}{2}$, es decir, se ha dejado fuera de la función a minimizar la mitad de los datos con mayor residuo.

Aplicando los principios del LTS al Análisis de Componentes Principales, Maronna [13] propuso sustituir la función objetivo (2.13) por la función objetivo robustificada

$$\hat{\sigma}^2_{LTS}(B_q, A_q, m) = \sum_{i=1}^{h} d^2_{(i:n)}(B_q, A_q, m), \tag{2.20}$$

donde

$$d^2_{(1:n)}(B_q, A_q, m) \leq d^2_{(2:n)}(B_q, A_q, m) \leq ... \leq d^2_{(n:n)}(B_q, A_q, m). \tag{2.21}$$

Es decir, se toman las $\lceil n(1-\alpha) \rceil$ distancias más pequeñas para calcular la función objetivo. De esta forma se trata de solucionar el problema de los datos atípicos siempre y cuando esta proporción de atípicos sea menor que α. Nótese que este método procede eliminando o recortando observaciones x_i enteramente.

2.5. Métodos de recorte en Análisis Cluster

El primer enfoque de recortes aplicado al Análisis Cluster fue el trabajo pionero en [14] donde se introdujo el método de *k-medias recortadas*. Este trabajo propuso un procedimiento robusto de clustering basado en el recorte entero de observaciones atípicas. Se habla de un tipo de recorte "imparcial", análogo al tipo de recorte utilizado en el LTS, porque son las propias observaciones las que nos dicen de forma imparcial cual es la fracción α de observaciones que deben ser recortadas [12].

Dada una muestra $\{x_1, ..., x_n\}$ de observaciones en $\mathbb{R}^p$, fijada una proporción $\alpha \in [0, 1)$ de observaciones a recortar, se busca resolver el problema:

$$\min_{Y \subset \{x_1,...,x_n\}, \#Y = \lceil n(1-\alpha) \rceil} \min_{\{m_1,...,m_G\} \in \mathbb{R}^p} \sum_{x_i \in Y} \min_{g=1,...,G} ||x_i - m_g||^2. \tag{2.22}$$

La definición del problema de minimización en (2.22) es bastante parecida a la considerada en el de la definición de las k-medias clásicas dado en (2.9) salvo que ahora una fracción α de observaciones (las observaciones no en Y) se dejan sin asignar a grupos. De hecho, cuando $\alpha = 0$ recuperamos el problema de las k-medias.

El algoritmo de las k-medias recortadas tiene también bastantes similitudes al algoritmo de k-medias pero se aplican pasos de "concentración" como el que será descrito en la Sección 3.7.1 para la regresión LTS (ver detalles en [15]).

Posterioremente, en [16] (ver también [17]) se introdujo una metodología más general conocida como *Trimming approach to Cluster Analysis (TCLUST)*. Nótese que muchas técnicas de Análisis Cluster parecen no tener en cuenta una distribución probabilística subyacente, pero la mayoría de las veces la elección del método afecta al funcionamiento del método imponiendo algún tipo particular de soluciones. Por ejemplo, el algoritmo de k-medias y su versión robustificada, las k-medias recortadas, están diseñadas para la búsqueda de grupos esféricos y del mismo tamaño, por lo que no es útil para datos que no sigan esta premisa.

Al realizar el recorte de observaciones atípicas, el TCLUST tiene en cuenta un modelo subyacente probabilístico para los datos para decidir que partes de la muestra deben de ser descartadas como también se hacía en [18] y [19] en el llamado "spurious outliers model". Sea $f(\cdot, \mu, \Sigma)$ la función de densidad de una normal p-variante de vector de medias μ y matriz de covarianzas Σ, se busca una partición $\{R_0, ..., R_G\}$ del conjunto de índices $\{1, ..., n\}$ y parámetros $\{p_g\}_{g=1}^G$, $\{m_g\}_{g=1}^G$ y $\{S_g\}_{g=1}^G$ maximizando la log-verosimilitud recortada

$$\sum_{g=1}^{G} \sum_{i \in R_g} (\log p_g + \log f(x_i, m_g, S_g)), \tag{2.23}$$

tal que $\#R_0 = \lceil n\alpha \rceil$ y $\sum_{g=1}^G p_g = 1$. R_0 es el conjunto de índices de los valores recortados.

El algoritmo utilizado para realizar esta maximización es una versión recortada de los típicos algoritmos EM aplicados en ajuntes de mixturas y en clustering [20].

El TCLUST también incluye restricciones sobre los tamaños relativos de los autovalores de las matrices $\{S_g\}_{g=1}^G$ para evitar que grupos poco informativos o "espurios", asociados a grupos definidos por muy pocas observaciones casi-colineales, sean detectadas. Estos grupos espurios pueden dar lugar a valores artificialmente grandes de (2.23).

Para comprobar el funcionamiento de TCLUST se comparó en [17] con el método *Partitioning Around Medoids* (PAM) [21]. El PAM es un algoritmo robusto alternativo a las k-medias reemplazando $\|x_i - m_g\|^2$ por $\|x_i - m_g\|$ en la minimización en (2.9). Se ha utilizado el conjunto de datos `geyser2` con $G = 3$ grupos y después se han añadido artificialmente observaciones atípicas para ver cómo funcionan ambos métodos con datos contaminados en la Figura 2.4. En la figura de la izquierda se ve el resultado de PAM con los datos originales, el cual parece ser muy bueno excepto en observaciones que se podían haber considerado atípicas (parte inferior izquierda). El segundo y tercer gráfico muestra el comportamiento de PAM y TCLUST con los mismos datos pero contaminados. Se ve que el PAM no consigue encontrar ya los 3 grupos, sino que es muy influenciado con los atípicos, tanto que muestra un grupo diferenciado de los otros dos. En cambio TCLUST recorta las observaciones atípicas que se salen claramente de la estructura de grupos mayoritaria.

El método TCLUST nos va a servir como inicialización de algunos procedimientos basados en recortes por celdas que serán posteriormente detallados.

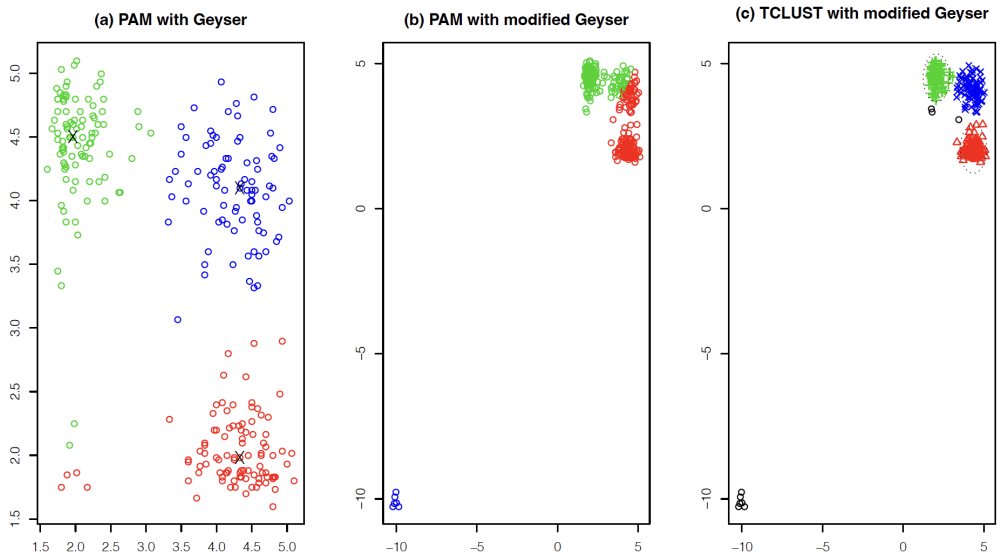


Figura 2.4: Comparación de PAM y TCLUST sobre el conjunto de datos `geyser2` tanto el original como contaminado artificialmente. Obtenido de [17]

2.6. Método de "Robust Linear Grouping"

Una metodología robusta para encontrar clusters de manera particional agrupando en torno a subespacios lineales fue introducida en [22]. Tradicionalmente, la forma más habitual de buscar grupos en Análisis Cluster es realizada en torno a centroides $m_1,...m_G$, clasificando cada observación x_i al centroide que quede más cercano (como sucedía, por ejemplo, con las k-medias). Los métodos de Análisis Cluster basados en subespacios asumen que puedan existir relaciones entre las variables analizadas, con distintos patrones en los clusters e, incluso, dimensiones "intrínsecas" $q_1,...q_G$ (con $q_g < p$ para $g = 1, ..., G$) diferentes en cada uno de los clusters. Por ejemplo, se pueden encontrar estructuras lineales como rectas (dimensión $q_g = 1$) o planos ($q_g = 2$ dimensiones) alrededor de los cuales se distribuyan cada uno de los grupos que conforman el conjunto de observaciones. Se consideran agrupaciones entorno a centroides cuando $q_g = 0$.

En [22] se propuso añadir la robustez a los métodos de Análisis Cluster basados en subespacios usando la metodología de recortes imparciales [12, 14] que ha sido anteriormente presentado.

Dada una muestra $\{x_1, ..., x_n\}$ de observaciones en $\mathbb{R}^p$, fijada una proporción $\alpha \in [0, 1)$ de observaciones a recortar, la dimensión de los distintos subespacios q, $0 \leq q < p$ y el número de grupos a encontrar G, se busca minimizar

$$\min_{Y \subset \{x_1,...,x_n\}, \#Y = \lceil n(1-\alpha) \rceil} \; \min_{\{H_1,...,H_k\} \subset \mathcal{A}_q} \frac{1}{\lceil n(1-\alpha) \rceil} \sum_{x_i \in Y} \min_{g=1,...,G} ||x_i - \mathrm{Pr}_{H_g}(x_i)||^2 \tag{2.24}$$

donde $\mathcal{A}_q := \{H \subset \mathbb{R}^p, H \text{ es el subespacio afín } q\text{-dimensional}\}$ y $\mathrm{Pr}_H(x)$ es la proyección ortogonal de x sobre el subespacio afín H.

En el planteamiento inicial en [22] se asumía que todos los espacios a buscar tenían la misma dimensión intrínseca $q_1 = ... = q_G = q$ y que esta dimensión común era $q = p - 1$.

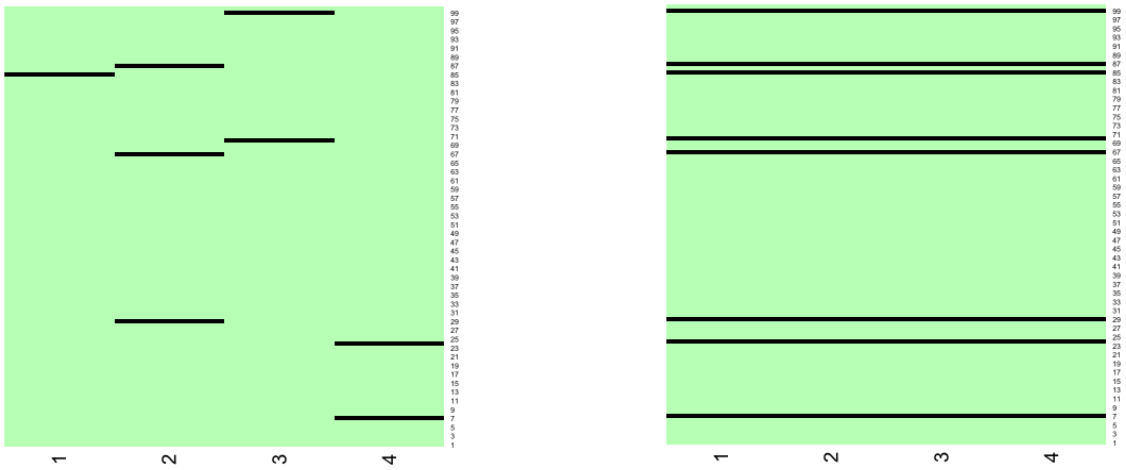
En el trabajo fin de grado [23] se adaptó esta metodología al caso de subespacios de dimensiones

$q_1, ..., q_G$ diferentes para los distintos subespacios afines. El resultado de ese trabajo, junto a su implementación en el lenguaje de programación R, va a ser también utilizado como inicialización para algunos procedimientos que serán descritos posteriormente en esta memoria.

2.7. Recorte por celdas

Como se ha comentado al principio de este capítulo, la metodología que se va a seguir es la del recorte por celdas en vez de por observaciones. Sean $\{x_1, ..., x_n\}$ observaciones de un conjunto de datos, donde cada $x_i = (x_{i1}, ..., x_{ip})^t \in \mathbb{R}^p$ son distintas variables o características medidas en cada observación. En dimensiones bajas, cuando p es pequeño, generalmente es equivalente un procedimiento de recorte u otro pero cuando la dimensión p va aumentando (común en los problemas de hoy en día en la Estadística moderna) la situación cambia. Al eliminar observaciones completas, por existir al menos alguna celda atípica en esas observaciones, se puede eliminar una gran cantidad de información correcta.

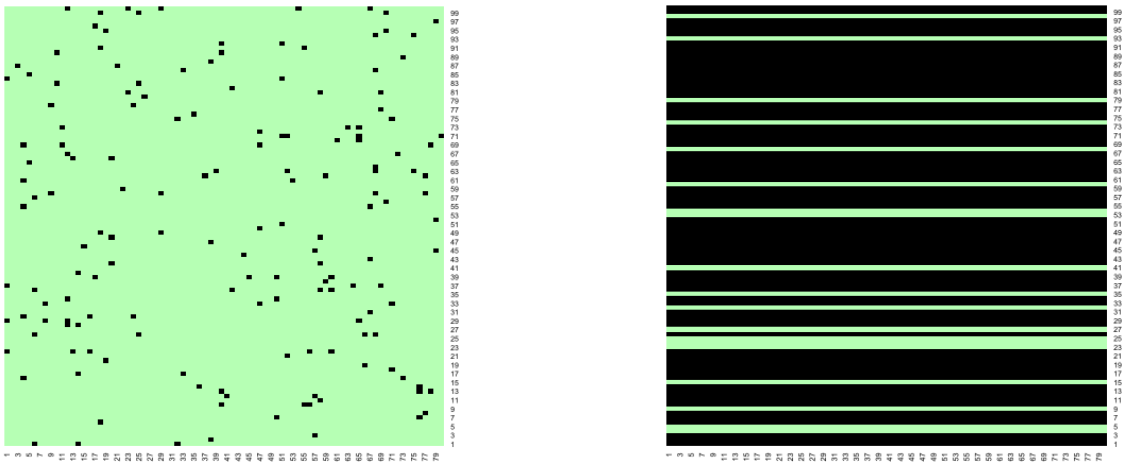
Para ilustrar este problema se va a usar dos matrices de datos X_1 y X_2 generadas sintéticamente a partir de realizaciones independientes en cada una de las celdas de una Normal $N(0, 1)$. Las dimensiones de esas matrices son $X_1 \in \mathbb{R}^{100\times 4}$ y $X_2 \in \mathbb{R}^{100\times 80}$. Ambos tienen el mismo número de observaciones ($n = 100$) pero en el primer caso contamos con $p = 4$ variables y en el segundo con $p = 80$. A estos datos se les ha añadido un ruido artificialmente, contaminando una proporción del 2 % de celdas de forma aleatoria con valores de una $U(-20, 20)$. En la aproximación tradicional de recortes por observaciones, al haber una dimensión contaminada se descarta toda la observación. En cambio, en la metodología de recortes por celdas que vamos a presentar se desean descartar solo las celdas concretas contaminantes.



(a) Recorte por celdas **(b)** Recorte de la observación completa

Figura 2.5: Comparación entre metodologías en X_1 (dimensión baja)

En las Figuras 2.5 y 2.6 se puede ver la diferencia entre las metodologías. Se han marcado de negro las celdas y observaciones recortadas, siguiendo una u otra metodología. En el primer



(a) Recorte por celdas **(b)** Recorte de la observación completa

Figura 2.6: Comparación entre metodologías en X_2 (dimensión alta)

caso no hay demasiada diferencia ya que estamos en una dimensión baja y se recortan únicamente 8 observaciones. Cuando aumentamos la dimensión se ve claramente que la proporción de observaciones recortadas es muy superior, en este caso de 81 casos (de un total de 100).

Una extensión interesante del Least Trimmed Squares (LTS), aplicando al ACP recortes por coordenadas o celdas, fue introducida por [24] y que denotaremos por CooLTS-PCA. Con este procedimiento se pretende paliar la influencia de las celdas atípicas en la estimación del subespacio aproximante proporcionado por el ACP pero recortando por celdas y no observaciones enteras.

De forma similar a lo realizado en (2.20), el CooLTS-PCA pretende minimizar en m, A_q y B_q la expresión

$$\sum_{j=1}^{G} \hat{\sigma}^2_{LTS,j}(B_q, A_q, m) \tag{2.25}$$

(con las condiciones sobre esos elementos que ya fueron comentadas) para

$$\hat{\sigma}^2_{LTS,j}(B_q, A_q, m) = \sum_{i=1}^{\lceil h \rceil} r^2_{(i:n)j} = \sum_{i=1}^{n} w_{ij}(x_{ij} - \hat{x}_{ij})^2 \tag{2.26}$$

donde los pesos w_{ij} han sido definidos de la forma

$$w_{ij} = \begin{cases} 1 & \text{si } r^2_{ij} \leq r^2_{(\lceil h \rceil:n)j} \\ 0 & \text{si } r^2_{ij} > r^2_{(\lceil h \rceil:n)j} \end{cases} \tag{2.27}$$

Es decir, el peso correspondiente a una celda es 0 si esa celda se recorta (no se utiliza para la regresión) o 1 si se considera.

De esta forma se consigue aplicar los mismos principios del Least Trimmed Squares o de recortes imparciales al caso de recortes por celdas. En [25] se realizaron varios experimentos en los que se analizan los métodos basados en celdas frente a los basados en recortes de observaciones completas

aplicados al ACP. En ellos se confirma que los primeros son más útiles en conjuntos de datos con un altos porcentajes de celdas contaminadas. Además, se aplican también a conjuntos de datos reales comprobando que el CooLTS-PCA es también muy útil para señalarnos la posición de estas celdas atípicas en la matriz de datos.

La idea sería generalizar este procedimiento CooLTS-PCA para el caso de G poblaciones heterogéneas, con $G > 1$. Pensamos que esta situación es bastante común en la práctica cuando se cuenta con datos que son el resultado de varios procesos heterogéneos de generación de los datos.

Se van a denotar los distintos clusters con el superíndice g, por lo que, por ejemplo, el vector m en el cluster g quedaría como $m^g \in \mathbb{R}^{p\times g}$. Extendiendo la notación anteriormente presentada, tendríamos que cada observación x_i de cada grupo g quedaría aproximada (partiendo de (2.11)) por

$$\hat{x}_i^g = m^g + B_{q_g}^g a_i^g \tag{2.28}$$

o elemento a elemento (partiendo de (2.12)) por

$$\hat{x}_{ij}^g = m_j^g + (a_i^g)^t b_j^g, \tag{2.29}$$

donde $B_{q_g}^g \in \mathbb{R}^{p\times q_g}$, $A_{q_g}^g \in \mathbb{R}^{n\times q_g}$. Nótese que q_g denota la dimensión intrínseca del g-ésimo subespacio aproximante, las cuales no tienen porqué ser las mismas para todos los clusters.

Para cada celda, se definen residuales del tipo

$$r_{ij}^g(B_{q_g}^g, A_{q_g}^g, m^g) = x_{ij} - \hat{x}_{ij}^g \tag{2.30}$$

Siguiendo lo presentado para el procedimiento CooLTS-PCA, se consideran los pesos w_{ij}^g, $i = 1, ..., n$, $j = 1, ..., p$ y $g = 1, ..., G$, donde cada peso se puede interpretar como la contribución al grupo g de la j-ésima coordenada de la observación x_i y se trata, pues, de minimizar

$$\min_{w_{ij}^g, B_{q_g}^g, A_{q_g}^g, m^g} \sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{g=1}^{G} w_{ij}^g (r_{ij}^g(B_{q_g}^g, A_{q_g}^g, m^g))^2. \tag{2.31}$$

Para que la minimización de (2.31) proporcione resultados razonables, es necesario fijar algunas restricciones para limitar el espacio de búsqueda:

1. Cada observación es asignada a solamente un grupo. Desde el punto de vista del análisis cluster es razonable que todas las celdas de la misma observación se asignen al mismo cluster. Es decir, si $g(i)$ es el grupo asignado a la observación i-ésima, para todos los demás grupos debemos descartar esas celdas. Esto se puede lograr de manera sencilla fijando pesos iguales a 0 a los grupos a los que no se ha asignado globalmente la observación x_i mediante

$$w_{ij}^g = 0 \quad \text{para todo} \quad g \neq g(i). \tag{2.32}$$

2. Dada una fracción de recorte α, podemos ajustar los pesos para que su suma sea aproxima-

damente el número de celdas que no queremos recortar:

$$\sum_{\{i:g(i)=g\}} w_{ij}^g = \lfloor n_g(1-\alpha) \rfloor \quad \text{para todo } g \text{ y } j, \text{ donde} \quad n_g = \#\{g(i) = g\} \tag{2.33}$$

Formulándolo de esa forma obliga a que la cantidad de celdas recortadas en cada variable y grupo sea la misma, lo cual puede no ser útil en ciertos casos donde la contaminación se concentre en unas variables más que en otras o en unos grupos más que en otros. Ya veremos que esto se va a solucionar sin demasiado problema mediante un paso final de "refinamiento".

Alternativamente, se podría modificar (2.33) para que se recorte una proporción α del total de las $n \times p$ celdas mediante

$$\sum_{i=1}^{n} \sum_{j=1}^{p} \sum_{g=1}^{G} w_{ij}^g = \lfloor np(1-\alpha) \rfloor \tag{2.34}$$

Ha sido comprobado empíricamente que es preferible usar las restricciones (2.33) a las restricciones (2.34) dado que obtienen resultados más estables.

2.8. Descripción del método

A continuación se va a revisar el método creado siguiendo la metodología de recorte por celdas aplicado al análisis cluster [1].

2.8.1. Inicialización

En primer lugar es necesario contar con una inicialización razonable de los parámetros que permita aproximarse al mínimo de la función objetivo (2.31). Esta inicialización es crítica dado que una mala inicialización es fácil que nos lleve a quedar atrapados en un mínimo local de la función objetivo. Hay que tener en cuenta que el problema que estamos resolviendo dista generalmente bastante de ser un problema de minimización convexo. Esta necesidad de contar con buenas inicializaciones no es específico de esta metodología, ya que esta problemática surgen con otros muchos métodos de Análisis Cluster robustos y no robustos. En este trabajo se utilizarán dos métodos de Análisis Cluster robusto para ayudar a la inicialización de los parámetros, el TCLUST y el RLG, explicados anteriormente. A partir de ambos métodos conseguimos obtener matrices iniciales $\{B_{q_g}^g\}_{g=1}^G$ y vectores inciales $\{m^g\}_{g=1}^G$.

Tanto TCLUST como RLG nos devuelven directamente un vector de centros para cada uno de los G grupos. En el caso de $B_{q_g}^g$ es algo más laborioso para TCLUST, dado que hay que obtener los autovectores asociados a la matriz de covarianzas resultante. En RLG este paso también es directo ya que el propio método nos proporciona esta información.

Esta etapa de inicialización se puede repetir varias veces o con diferentes métodos de inicialización, que tras su iteración completa, nos puede llevar a resultados diferentes. Esto es especialmente

necesario en el caso de que G y/o p sean elevados.

2.8.2. Actualización de los grupos

Teniendo ya la inicialización de $\{B_{q_g}^g\}_{g=1}^G$ y $\{m^g\}_{g=1}^G$, es necesario inicializar las matrices $\{A_{q_g}^g\}_{g=1}^G$. Incluso con buenas inicializaciones de los dos primeros conjuntos de parámetros, usar mínimos cuadrados OLS para obtener las matrices $A_{q_g}^g$ puede no ser adecuado, dado que la matriz de datos X puede contener valores atípicos que afecten a estas regresiones. Para tratar esta problemática, una opción sería aprovechar los pesos w_{ij}^g para corregirlo, pero en la primera etapa de la inicialización no hay ninguna información sobre pesos w_{ij}^g adecuados.

Como solución se propone aplicar la regresión LTS explicada en la Sección 2.4, la cual es un método robusto que estará poco influenciado por posibles celdas contaminadas. De esta forma conseguimos inicializaciones mejores de $A_{q_g}^g$ y de las asignaciones $g(i)$ a grupos, usando los residuales de esta regresión.

Para tratar de comprender el procedimiento de asignación a grupos, supongamos primero un nivel de recorte $\alpha = 0$. En un algoritmo de k-medias tradicional, los grupos vienen dados por la distancia de cada observación al centro más próximo:

$$g(i) = \arg \min_{g=1,\dots,G} \|x_i - m^g\| \tag{2.35}$$

De forma similar, si buscamos grupos en torno a subespacios, resulta razonable usar

$$g(i) = \arg \min_{g=1,\dots,G} \|x_i - \hat{x}_i^g\| = \arg \min_{g=1,\dots,G} \|x_i - \mathrm{Pr}_{H_j}(x_i)\|,$$

donde $\mathrm{Pr}_{H_j}(x_i) = \hat{x}_i^g = m^g + B_{q_g}^g a_i^g$ es el punto más cercano del g-ésimo subespacio aproximante a la observación x_i. Estos puntos más próximos se pueden obtener realizando regresión y modelando los p valores del vector $x_i - m^g$ como combinación lineal óptima de las q_g columnas de $B_{q_g}^g$. Para alcanzar robustez en estas regresiones, en este procedimiento de recorte por celdas, se va a usar la regresión LTS con un tamaño de recorte α_{LTS}, donde este valor debe ser lo suficientemente grande como para ser superior a la contaminación por celdas máxima que se pueda esperar en cada fila de la matriz de datos.

Si denotamos como $\tilde{a}_i^g$ los coeficientes de aplicar esta regresión y los residuales como

$$\tilde{r}_{ij}^g = (x_{ij} - m^g - (b_j^g)^t \tilde{a}_i^g)^2,$$

definiremos como "distancia" entre x_i y el "punto más próximo" del g-ésimo subespacio aproximante como

$$D_i^g = \sum_{j=1}^{\lceil p(1-\alpha_{\mathrm{LTS}}) \rceil} \tilde{r}_{i(j:p)}^g.$$

Como suele hacerse en técnicas LTS se tienen en cuenta solo las observaciones con residuales más pequeños tratando de que residuales extremos no tengan un efecto dañino en el ajuste. Las

asignaciones a grupos se obtienen mediante la "distancia" mínima

$$g(i) = \arg \min_{g=1,...,G} D_i^g.$$

Adicionalmente, los valores $\tilde{a}_i^g$ proporcionan además una inicialización bastante robusta de $A_{q_g}^g$.

2.8.3. Actualización de los pesos

Teniendo ya inicializaciones previsiblemente razonables de los parámetros $\{A_{q_g}^g\}_{g=1}^G$, $\{B_{q_g}^g\}_{g=1}^G$ y $\{m^g\}_{g=1}^G$, podemos calcular los valores "predichos" de nuestra matriz de datos en esa primera iteración, $\hat{X}$, según (2.29). Con esta estimación podemos calcular los residuos con respecto a los datos originales en X mediante la fórmula (2.30). Para calcular los pesos w_{ij}^g actualizados, usamos el razonamiento de quedarnos con los $[n(1-\alpha)]$ residuos más pequeños y asignamos a cada celda pesos óptimos (paso de concentración) teniendo en cuenta (2.27). Estos pesos w_{ij}^g obviamente deben ser elegidos para cumplir las restricciones (2.33) y (2.34).

2.8.4. Actualización de los parámetros

El siguiente paso sería actualizar los parámetros de forma iterativa con respecto a los pesos w_{ij}^g obtenidos en el paso anterior. Si definimos $L(\{w_{ij}^g, B_{q_g}^g, A_{q_g}^g, m^g\})$ como la suma de los errores cuadráticos con pesos dada por (2.31), es decir, nuestra función objetivo, entonces si derivamos L con respecto a a_i^g, b_j^g y m_j^g tenemos:

$$\begin{aligned}
\frac{\partial}{\partial a_i^g} L(\{w_{ij}^g, B_{q_g}^g, A_{q_g}^g, m^g\}) &= -2 \sum_{j=1}^{p} w_{ij}^g r_{ij}^g (B_{q_g}^g, A_{q_g}^g, m^g) b_j^g \\
\frac{\partial}{\partial b_j^g} L(\{w_{ij}^g, B_{q_g}^g, A_{q_g}^g, m^g\}) &= -2 \sum_{j=1}^{p} w_{ij}^g r_{ij}^g (B_{q_g}^g, A_{q_g}^g, m^g) a_i^g. \\
\frac{\partial}{\partial m_j^g} L(\{w_{ij}^g, B_{q_g}^g, A_{q_g}^g, m^g\}) &= -2 \sum_{j=1}^{p} w_{ij}^g r_{ij}^g (B_{q_g}^g, A_{q_g}^g, m^g)
\end{aligned} \tag{2.36}$$

Si igualamos estas expresiones a 0, obtenemos el siguiente sistema de ecuaciones:

$$\begin{aligned}
\sum_{j=1}^{p} w_{ij}^g (x_{ij} - m_j^g) b_j^g &= \left(\sum_{j=1}^{p} w_{ij}^g b_j^g (b_j^g)^t \right) a_i^g, \qquad i = 1, ..., n \text{ y } g = 1, ..., G \\
\sum_{j=1}^{p} w_{ij}^g (x_{ij} - m_j^g) a_i^g &= \left(\sum_{i=1}^{n} w_{ij}^g a_i^g (a_i^g)^t \right) b_j^g, \qquad j = 1, ..., p \text{ y } g = 1, ..., G \\
\sum_{j=1}^{p} w_{ij}^g (x_{ij} - (a_i^g)^t b_j^g) &= \sum_{i=1}^{n} w_{ij}^g m_j^g, \qquad j = 1, ..., p \text{ y } g = 1, ..., G
\end{aligned} \tag{2.37}$$

Por tanto, con pesos w_{ij}^g fijados, se pueden actualizar los parámetros aplicando simples regre-

siones de mínimos cuadrados con pesos y medias ponderadas.

2.8.5. Refinamiento

Una vez terminado todo el procedimiento iterativo, es posible que exista una cierta cantidad de celdas incorrectamente recortadas por culpa de que se haya fijado el parámetro de recorte α excesivamente grande o que no existan o existan pocos atípicos en una variable j para un grupo g. Si es el caso, estaríamos descartando celdas correctas, las cuales contienen claramente información útil. Por tanto, tiene sentido considerar un último paso de "refinamiento" para recuperar estas observaciones incorrectamente recortadas. Este paso no resulta especialmente complicado haciendo uso de los residuales en las celdas. Este procedimiento será descrito en la Sección 3.5.

El segundo refinamiento propuesto tendrá que ver con la posibilidad de recortar observaciones, no solo celdas, enteras. Si una observación cuenta con un alto porcentaje de celdas que han sido recortadas se debería poder recortar la observación completa porque la regresión LTS que se ha usado en las primeras fases puede no haber logrado manejar correctamente esta fracción alta de celdas contaminantes. Como criterio del porcentaje de celdas recortadas, se va a considerar que se debe recortar toda la observación x_i si la proporción de celdas recortadas es superior al tamaño α_{LTS} de recorte de LTS, es decir, si sucede que

$$\#\{w_{ij}^{g(i)} = 0 : j = 1, ..., p\}/p > \alpha_{\text{LTS}}. \tag{2.38}$$

2.9. Datos funcionales

Los datos funcionales son un tipo de datos muy utilizados en la Estadística actual dada la facilidad que las nuevas tecnologías tienen para registrar datos de forma continua. Los datos se pueden ver como "funciones" observadas, dando lugar a una disciplina nueva en Estadística para analizar estos datos funcionales conocida como Análisis de Datos Funcionales [26], [27]. Aunque estos datos funcionales pueden tratarse como vectores en un espacio de muy alta dimensionalidad (en teoría, dimensión infinita) tras considerar una discretización de estas funciones, es bien sabido que este enfoque no es el más correcto y que es necesario desarrollar métodos estadísticos que funcionen correctamente con este tipo de datos. Así, la extensión de las técnicas multivariantes más conocidas a este tipo de datos está recibiendo una gran atención recientemente. Este es el caso también del Análisis Cluster [28], [29], [30]. También, obviamente, es necesario disponer de técnicas que sean resistentes a registros atípicos de estas funciones o curvas.

En este trabajo supondremos que las filas de la matriz X incluyen los valores medidos de n funciones al evaluar cada una de estas funciones en p momentos equidistantes en el tiempo, $0 < t_1 < ... < t_p < 1$ en el intervalo $[0, 1]$.

Trabajaremos este enfoque de recorte por celdas en el espacio de dimensión p (generalmente grande) obtenido por la discretización de las curvas, incluso en las partes iterativas del procedimiento, dado que contamos con la aproximación por subespacios lineales en dimensiones $\{q_g\}_{g=1}^{G}$,

con $q_g << p$. No obstante, en la fase de inicialización sí que es necesario trabajar en alguna dimensión inferior, usando bases funcionales y poder realizar un "filtrado" previo de los datos para evitar outliers demasiados grandes. Esta fase inicial requiere suponer una cierta "continuidad" y suavidad en las funciones observadas (salvo los atípicos) y en las funciones subyacentes que generan los subespacios funcionales en torno a los que deberíamos agrupar las funciones. A continuación detallaremos un poco estas técnicas, que insistimos, son solo aplicadas en la fase de inicialización:

1. *B-splines*: Mediante regresiones se van a representar cada una de las funciones (filas de la matriz X) por una representación finito-dimensional en una base de B-splines. Las funciones B-splines son unas funciones spline que tienen el soporte mínimo con respecto a un grado, suavidad y partición del dominio. Cada función, determinados el grado, la suavidad y partición, se puede representar como una combinación lineal de B-splines de los mismos parámetros y, así, cada función admite una representación por P coeficientes, resultado de la regresión con $P << p$.

   Un spline de orden M es una función polinómica definida a trozos de grado $M-1$. Los valores de x donde los polinomios se unen se conocen como nodos, denotados con $t_0, ..., t_M$, tales que $t_0 \leq t_1 \leq ... \leq t_M$. Cuando los nodos son diferentes, las primeras $N-2$ derivadas de cada polinomio son continuas con respecto a cada nodo. Cuando r nodos son iguales, solo las $M-r-1$ derivadas del spline son continuas (con respecto a cada nodo).

   Para una secuencia dada de nodos, existe un único spline $B_{m,M}(x)$ tal que

   $$B_{m,M}(x) = \begin{cases} 0 & \text{si } x < t_m \text{ o } x \geq t_{i+m} \\ \text{distinto de } 0 & \text{en caso contrario} \end{cases}$$

   Si añadimos la restricción $\sum_m B_{m,M}(x) = 1$ para todo x entre el primer y último nodo entonces el factor de escala de $B_{m,M}(x)$ queda fijado.

   Para nuestros datos funcionales, si fijamos ν nodos, la dimensión a la que se reduce es $P = \nu + 4$. Como resultado, las curvas quedan como $\{\tilde{x}, i = 1, ..., n\}$ con $\tilde{x} \in \mathbb{R}^P$ donde $P << p$.

2. *lowess*: El método `lowess` introducido en [31] es un método no paramétrico de suavizado en el que se relajan notablemente las premisas asumidas en la regresión. Se conoce también como regresión local robusta dado que en cada punto x se ajusta robustamente un polinomio con respecto a las observaciones más cercanas, usando pesos. Para determinar qué observaciones utilizar, se toman la proporción `f` de observaciones más próximas a x. El uso de pesos adecuados hace que el `lowess` nos proporcione un suavizado robusto que resulta de gran utilidad para evitar que ciertas celdas atípicas puedan jugar un papel muy importante a la hora de determinar la representación de las curvas en dimensión P.

Seguidamente, en esta fase de inicialización, podemos usar uno de los dos métodos ya comentados, TCLUST o RLG en $\mathbb{R}^P$, con la representación en dimensión P de las curvas en la base de

B-splines para tratar de inicializar las matrices $\{B_{q_g}^g\}_{g=1}^G$ y los vectores $\{m^g\}_{g=1}^G$. Nótese que se TCLUST y RLG, al tratar el problema en $\mathbb{R}^P$, nos devuelve vectores de medias y autovectores también en $\mathbb{R}^P$. No obstante, mediante el producto de estos vectores y autovectores con la matriz obtenida al evaluar las funciones en nuestra base de B-splines en los valores t_1, ..., t_p, es trivial obtener nuevos vectores (ahora sí) en $\mathbb{R}^p$ para inicializar el procedimiento iterativo a realizar en dimensión p.

Capítulo 3

Implementación

En este capítulo se detallará cómo se ha utilizado la metodología descrita en los anteriores capítulos para ser implementada en el lenguaje de programación R. Originalmente, se partía de un código realizado en R bastante "ad hoc" para la obtención de los gráficos acompañando al trabajo [1]. Este código estaba poco refinado, era solo válido para datos funcionales con el algoritmo TCLUST y no resultaba para nada amigable para su implementación práctica en otros conjuntos de datos.

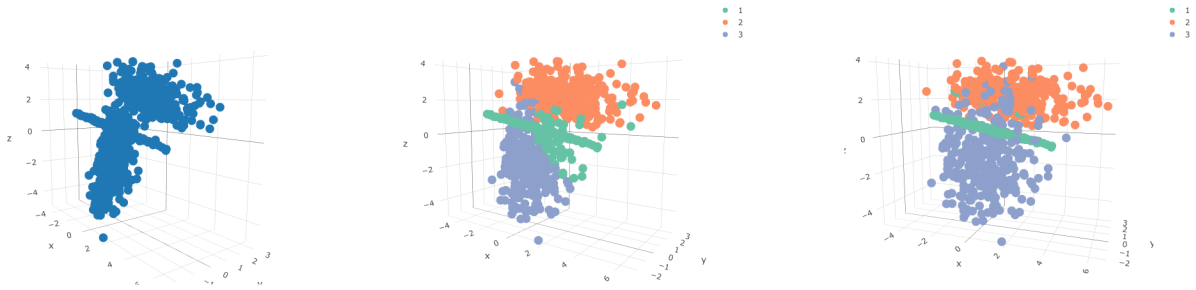
Partiendo de dicho código, se han ido añadiendo numerosas mejoras que se describirán a continuación, para después presentar finalmente el pseudocódigo de los algoritmos utilizados junto con una descripción concreta de los argumentos y salidas de las funciones implementas en R.

3.1. Subespacios de dimensiones diferentes

La primera de las mejoras consideradas ha sido habilitar la posibilidad de realizar agrupaciones en torno a subespacios de dimensiones. En la implementación original, se suponía una dimensión intrínseca común $q_1 = q_2 = ... = q_G = q$. Esta suposición puede no ser acertada cuando las observaciones se agrupen en torno a G subespacios de dimensiones distintas, por ejemplo, cuando esas agrupaciones se tengan conjuntamente en torno a puntos $q_g = 0$, rectas $q_g = 1$, planos $q_g = 2$, etc.

Para ilustrar esta situación, se van a generar $G = 3$ grupos de manera sintética en $\mathbb{R}^3$ donde uno de los clusters esté agrupado en torno a una recta ($q_1 = 1$) y otros dos clusters agrupados en torno a dos planos ($q_2 = 2$ y $q_3 = 2$). Para ello se ha usado la función `genLP` del paquete `T4cluster` [32].

En la Figura 3.1a se pueden ver los datos originales generados mediante `genLP`, en los que ya se pueden verse claramente las tres estructuras, correspondientes a cada uno de los clusters. En la Figura 3.1b se realiza un agrupamiento en torno a rectas $q_1 = q_2 = q_3 = 1$, en el que se aprecia que es insuficiente dado que hay muchas observaciones que quedan mal clasificadas. Al usar planos $q_1 = q_2 = q_3 = 2$ (Figura 3.1c) el ajuste es mejor, aunque de todas formas no es correcto al quedar varias observaciones en la parte superior del gráfico mal clasificadas (se han asignado



(a) Datos originales sin agrupar **(b)** Agrupación en 1 dimensión **(c)** Agrupación en 2 dimensiones

Figura 3.1: Agrupación de la nube de puntos en subespacios. Para cada gráfico se han usado subespacios de la misma dimensión

equivocadamente al grupo 1). En la Figura 3.2 se realiza un ajuste con dimensiones distintas para cada grupo, de la misma forma que los datos originales (dos planos y una recta) fueron generados ($q_1 = 1$ y $q_2 = q_3 = 2$). Dicho ajuste resulta claramente el más adecuado.

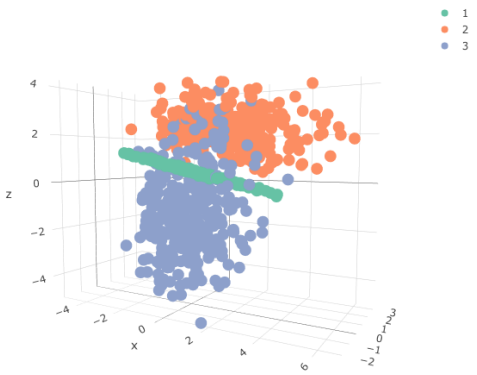


Figura 3.2: Agrupación en clusters de distinta dimensión, uno unidimensional y los otros dos bidimensionales

Para lograr esta modificación, se han realizado varios cambios en el algoritmo, principalmente sobre las matrices $B^g_{q_g}$ y $A^g_{q_g}$. Estas matrices tenían todas dimensión $q \times p$ y $q \times n$, respectivamente, siendo q la dimensión intrínseca común. Al admitir dimensiones q_g diferentes, ya no se puede usar esta estructura en forma de matriz. Para remediarlo, se han usado estructuras de "lista" en R, una por cada grupo, almacenando matrices dentro de cada elemento de la lista, en el primer caso de dimensiones $q_g \times p$ y en el segundo $q_g \times n$. Esta modificación hace que se tenga que cambiar también la forma en la que se accede a los elementos y a su inicialización.

Los datos generados con `genLP` van a ser utilizados en las siguientes secciones para presentar las distintas modificaciones y mejoras aplicadas en la implementación. Cuando sea necesario para la presentación, también se introducirá cierta fracción de celdas contaminantes de forma "dispersa" en los datos.

3.2. Preprocesado de los datos de entrada

Antes de ejecutar el algoritmo, se propone considerar dos potenciales preprocesados de los datos de entrada que pueden llegar a ser muy importantes para que el método proporcione buenos resultados:

- la posibilidad del *escalado robusto* de los datos de entrada buscando una estandarización inicial robusta de las variables
- la aplicación de un sistema de imputación inicial de datos perdidos y de datos atípicos que resulte simple aunque no tenga en cuenta ninguna estructura de G grupos subyacente. Este procedimiento ha sido el método de *Detect Deviating Cells* (DDC) introducido en [33].

La primera de las modificaciones, basada en el escalado de los datos de entrada, es muy común a la hora de aplicar técnicas de Aprendizaje Automático. En muchas ocasiones las distintas variables que componen un conjunto de datos tienen diferentes rangos de valores, lo cual sesga los métodos en gran medida, dando más prioridad a unas variables que a otras. En este caso, en vez de usar el escalado tradicional con la media y la desviación típica muestral clásica como estimadores de localización y escala, se va a utilizar la mediana y el MAD como alternativas más robustas, tal como se mostró en la Sección 2.1.3. Por tanto, para cada celda x_{ij} del conjunto de datos se va a considerar la transformación remplazando x_{ij} por

$$\frac{x_{ij} - \text{Med}\{x_{ij}\}_{i=1}^{n}}{\text{MAD}\{x_{ij}\}_{i=1}^{n}} \tag{3.1}$$

Como ejemplo del impacto de escalar robustamente o no los datos se va a generar un conjunto artificial con `genLP` de datos con $n = 300$ observaciones, 100 observaciones en cada cluster y $p = 10$ y donde la primera variable x_1 ha sido multiplicada por un factor de 100. De esta forma, el rango de las variables se puede ver en la Figura 3.3, viendo que la primera variable se encuentra en el rango $(-20, 20)$ aproximadamente, mientras que las nueve variables restantes oscilan entre $(-0.5, 0.5)$.

En la Figura 3.4 podemos ver las diferencias entre no escalar (Figura 3.4a) y escalar (Figura 3.4b) unos datos simulados y sin contaminación. En el primer caso, prácticamente todas las celdas recortadas pertenecen al primer cluster (el que tenía un rango mayor y diferente al de los otros nueve). En cambio, si escalamos los datos, se encuentra solamente un atípico y no es en la variable primera, por lo que escalar los datos ayuda a considerar todas las variables por igual, sin importar el rango de las mismas.

En el caso concreto de datos funcionales, no se recomienda (por defecto) escalar las variables dado que se podrían perder características importantes de las curvas.

La segunda de las modificaciones consiste en la aplicación del método denominado "Detect Deviating Cells" (DDC) en [33]. Con este procedimiento se realiza un análisis de celdas atípicas teniendo en cuenta las correlaciones por pares de variables usando un método robusto y sin asumir $G > 1$ posibles subpoblaciones en los datos. Una de las principales ventajas de esta metodología

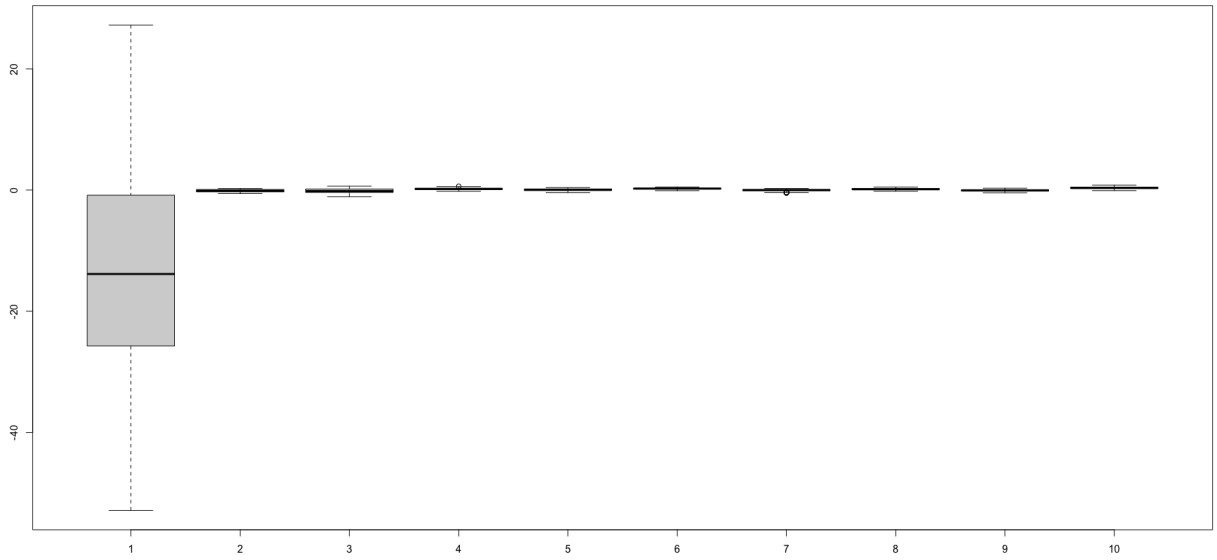
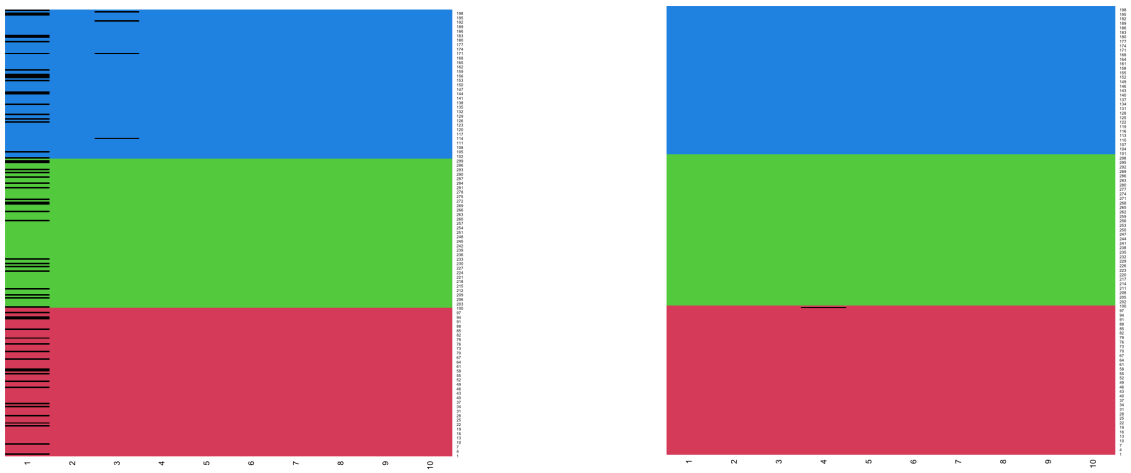


Figura 3.3: Gráfico de cajas de cada variable



(a) Asignación del método a clusters sin escalar **(b)** Asignación del método a clusters escalando

Figura 3.4: Diferencias entre no escalar y escalar los datos en la asignación a grupos

es que proporciona una estimación de las celdas que han sido atípicas y de los valores perdidos o "missing" (`NA`).

En este método DDC, en primer lugar, se estandarizan los datos con estimadores robustos, usando el primero de los pasos de un M-estimador (método robusto que proporciona pesos robustos a las observaciones) para la localización y la escala. Para localizar los valores atípicos se supone que la distribución conjunta de las variables es una (única) normal multivariante con media μ y matriz de varianzas-covarianzas Σ. Partiendo de un método simple de imputación inicial de observaciones perdidas, Los autores recomiendan normalizar las variables que claramente salgan de dicha normalidad, usando por ejemplo el método de Box-Cox. Se define una nueva matriz U de las mismas dimensiones que X, se calculan las correlaciones entre las variables utilizando

la nueva matriz a través de un método robusto llamado `robCorr`. Con estas correlaciones, se consideran variables que están "conectadas" por esa correlación y estas variables conectadas son las que se usarán para estimar "pendientes" robustas de regresiones simples con una función `robSlope` (también descarta valores ausentes, definida también en el anexo de [33]). Para predecir los atípicos se usan las variables que están conectadas según las correlaciones y esas pendientes de forma robusta.

Para ilustrar las diferencias de este método en la detección de valores atípicos por celdas, se utilizó en [33] un conjunto de datos denominado `TopGear` del paquete `robustHD` [34]. En él se analizan 11 variables numéricas objetivas sobre 297 coches del programa inglés de televisión "TopGear". En la Figura 3.5 se muestra la capacidad de detección de celdas atípicas con DDC respecto al uso de técnicas de búsqueda de atípicos por filas, observándose un claro interés por el uso del DDC.

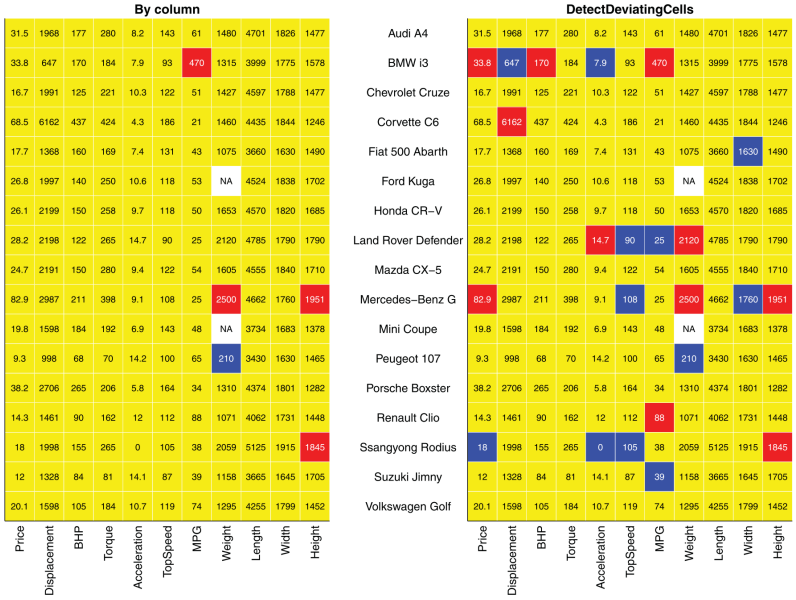


Figura 3.5: Diferencias en la detección de *outliers* de hacerlo por columnas (izquierda) o con DDC (derecha). Obtenido de [33]

El uso de DDC es bastante rápido a nivel computacional, ya que es muy simple por estar basado en simples correlaciones robustas.

3.3. Inicialización con RLG

En la implementación de partida solo se podía inicializar los métodos iterativos por celdas mediante el uso de TCLUST, lo cual puede no ser adecuado, especialmente cuando nos encontremos en una dimensión p relativamente alta por el gran número de parámetros que estimar G matrices de covarianzas, tal como hace TCLUST, implica. Adicionalmente, a nivel formal, el uso de subespacios afines aproximantes está más en línea con la aproximación por subespacios afines que realizamos para aproximar y reconstruir las celdas individuales.

Por ello, se ha considerado contemplar también como método de inicialización el método RLG con un recorte inicial elevado. La implementación eficiente en C++ fue abordada en el TFG [23]. La salida proporcionada por el RLG es muy útil ya que la matriz de centros se puede usar directamente para inicializar los vectores m^g para $g = 1, ..., G$. De la misma forma se puede hacer con los autovectores generando los subespacios afines aproximantes, los cuales se corresponden directamente con las matrices $B^g_{q_g}$ para $g = 1, ..., G$.

En el Capítulo 4 se tratará de ilustrar el impacto al utilizar cada tipo de incialización, TCLUST y RLG, aplicado a distintos conjuntos de datos generados artificialmente.

3.4. Rendimiento computacional

Dado que este método está previsto emplearse en datos con dimensiones posiblemente elevadas, es crítico que el rendimiento computacional sea el adecuado porque si no lo fuera, es posible que el procedimiento no termine en un tiempo razonable. Con este fin, se ha hecho mucho hincapié en mejorar al máximo posible este rendimiento computacional, analizando las partes en el código que pudieran tener un mayor impacto. Estas partes son generalmente las que se repitan muchas veces y/o tengan una mayor complejidad computacional.

La mayor parte de las operaciones del algoritmo son matriciales, que de por sí no deberían consumir demasiado tiempo. En cambio, hay dos partes que tienen un impacto considerable en el rendimiento. Una de estas partes críticas detectadas es la regresión LTS, que debe realizarse $n\times$`loop1` veces. El LTS es de por sí un método iterativo, que puede ralentizar mucho el procedimiento y, consecuentemente, las regresiones con pesos que se realizan para actualizar $B^g_{q_g}$, $A^g_{q_g}$ y m^g.

Con respecto a la regresión LTS, la implementación inicial usaba la función `ltsReg` de la librería `robustbase` [35], la cual no permitía ejecutar la regresión en dimensiones bajas, lo que suponía un problema dado que el procedimiento propuesto debería funcionar en todo tipo de dimensiones. Además, dicha función está programada en R y FORTRAN, lo que provocaba que los resultados no fueran lo suficientemente adecuados en términos de velocidad computacional. Dados ambos factores, se optó por implementar desde el principio este algoritmo en C++ en vez de en R. La elección de este lenguaje de programación responde a las siguientes razones:

- *Rapidez:* R no es suficientemente rápido, dado que fue concebido para hacer el análisis de datos y operaciones estadísticas sencillas y no para ser un lenguaje de altas prestaciones computacionales. Por ello, R está pensado para que sea lo más sencillo para el programador y no está optimizado a nivel computacional. A modo de ejemplo, no se optimiza el uso de la memoria y cuando se modifica un array se realiza una copia de este, lo cual impacta negativamente en su rendimiento computacional. A esto se añade ser un lenguaje interpretado en vez de compilado, es decir, para ejecutar código R no hace falta traducirlo en una fase previa a lenguaje máquina, sino que se realiza sobre la marcha.

- *Integración con R:* Existen varios paquetes que nos permiten convertir variables de C++ a R y viceversa, así como funciones u objetos. En esta implementación se han usado dos:

  - `Rcpp`: es el encargado de realizar la definición y conversión entre ambos lenguajes. Por ejemplo, para crear una función en C++ que pueda ser usada en R, basta con añadir el siguiente comentario al principio de la función:

    $$// \ [[Rcpp::export]]$$

    Y después se puede compilar el archivo fuente `.cpp` con `sourceCpp` en R. Hay que tener cuidado a la hora de definir los argumentos y la salida de la función, dado que tienen que pertenecer al paquete Rcpp; de otra forma daría un error.

  - `RcppArmadillo`: proporciona acceso a la librería Armadillo de C++ a través de R. Esta librería es especialmente útil para realizar operaciones de álgebra lineal y de computación científica. Por ejemplo, es posible calcular autovalores, autovectores o descomponer matrices (SVD, QR, ...), entre otras funciones.

- *Portabilidad:* El lenguaje C++ es ampliamente usado por poder ser compilado de muchas formas diferentes (gcc, clang, ...) y en plataformas totalmente distintas. Con ello podemos crear código C++ y ejecutarlo en cualquier tipo de procesador (x64, x32, ARM, MIPS, ...) y en cualquier sistema operativo (GNU/Linux, macOS, Microsoft Windows, ...).

Para comprobar la diferencia de velocidad entre ambas implementaciones, se realizaron una serie de pruebas sobre el mismo ordenador con distintos conjuntos de datos generados de forma aleatoria, de distintas dimensiones $p \in \{10, 50, 100, 300\}$. En la Tabla 3.1 se muestran los resultados (en segundos): en todos los casos el código C++ gana en términos de velocidad al de R y cuanto mayor es la dimensión más se notan estas diferencias llegando al extremo de tardar más de 4 horas en terminar en R cuando en C++ tardaba menos de un minuto. Con dimensiones superiores a 500, el código en R no termina en un tiempo prudencial (de días), en cambio en C++ consigue terminar algo más de 2 minutos.

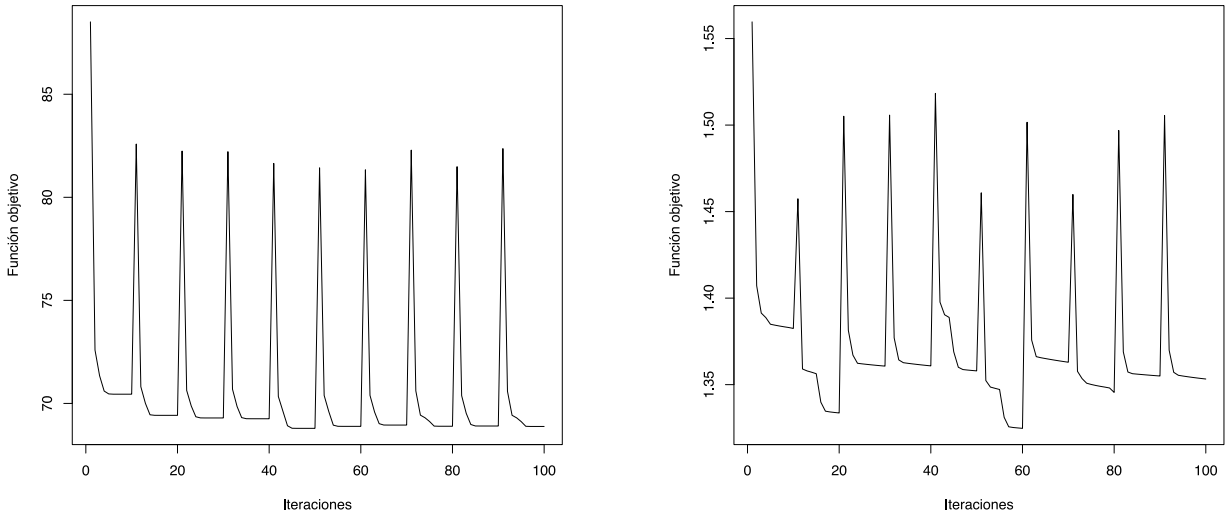
Por otra parte, la versión de R está optimizada para conjuntos de datos con más observaciones (n superior) pero dado el uso que se le va a hacer en el procedimiento no es necesario tener dicha mejora.

	$p = 10$	$p = 50$	$p = 100$	$p = 300$	$p = 500$
`R`	0.062	1.485	6.348	46.878	-
`C++`	0.031	0.283	0.914	29.623	141.346

Tabla 3.1: Diferencias en la velocidad de la implementación de LTS sobre R y C++ (tiempos en segundos) con el mismo número de inicializaciones

El programa implementado en C++ debe permitir realizar el ajuste LTS sin término de "intercept", para poder resolver las regresiones necesarias.

Viendo la diferencia de velocidad entre ambas implementaciones, se hizo lo mismo con el resto del algoritmo, implementarlo totalmente en C++ la versión original. En este caso la diferencia era mínima, por lo que se decidió dejarlo y continuar en R dado que no suponía gran ventaja, y a la hora de desarrollar y realizar cambios, R ofrece un entorno más amigable.



(a) **(b)**

Figura 3.6: Evolución de la función de pérdida en dos escenarios diferentes

Otra de las mejoras propuestas consiste en quedarse con el mejor valor de la función objetivo en vez de con el último. Como se puede ver en la Figura 3.6 existen principalmente dos tipos de comportamientos en la evolución de la función de pérdida. En el primero de ellos (Figura 3.6a), el decrecimiento es suave y el último valor corresponde con el mejor. En cambio, en la situación de la Figura 3.6b se producen "saltos" dependiendo de como se vayan "acomodando" la asignación de observaciones. De ahí la importancia de los hiperparámetros `loops1` y `loops2`. El primero, correspondiente al bucle externo, provee una inicialización diferente en cada iteración de las asignaciones y de los primeros "scores", lo que ayuda a no estancarse en mínimos locales y poder explorar varias soluciones. El segundo, correspondiente al bucle interno, permite iterar sobre la solución inicial.

3.5. Paso de refinamiento

Como se introdujo en la Sección 2.8.5, existen dos etapas de refinamiento que contemplan la recuperación de celdas incorrectamente recortadas y la eliminación de observaciones o filas de la matriz de datos completas.

En el primero de los refinamientos posible relativo a la *recuperación de celdas incorrectamente recortadas* se van a utilizar los residuos escalados o estandarizados respecto al subespacio aproximante del grupo al que se ha sido globalmente asignada cada observación x_i para determinar

cuáles son las celdas que se debieran recuperar. El residuo en la celda (i, j) vendrá dado por

$$x_{ij} - \hat{x}_{ij} = x_{ij} - \hat{x}_{ij}^{g(i)},$$

con $g(i)$ siendo el grupo al que que ha sido globalmente asignada la observación x_i.

Para realizar el escalado robusto de los residuales, se ha usado estimadores Med_j^g y el MAD_j^g que están calculados con la mediana y el MAD obtenidos considerando solo los residuales correspondientes a la variable j en las observaciones asignadas al cluster $g(i)$. Se considera entonces unos residuales estandarizados robustamente y que son definidos mediante

$$r_{ij} = \frac{x_{ij} - \hat{x}_{ij} - \text{Med}_j^g}{\text{MAD}_j^g}. \tag{3.2}$$

Se han usado estimadores robustos en vez de las medias y la desviaciones típicas muestrales dado que esos estimadores no son robustos.

Estos residuales serán también usados para medir la "magnitud" del grado de "atipicidad", en términos de desviaciones típicas, de las celdas, con el fin de visualizar más cómodamente dicho grado.

Se ha comprobado experimentalmente que los residuos correspondientes a cada variable y grupo no siguen una distribución Normal, teniendo colas más pesadas como se puede ver en la Figura 3.7, donde se han representado los residuales estandarizados para el resultado de aplicar el procedimiento a datos generados con `genLP`. Este fenómeno ya fue observado en [24]. Por tanto, al no poder suponer normalidad en los residuos tampoco se pueden usar los percentiles de la Normal estandar (o los percentiles de la χ_1^2 si se usan los residuales al cuadrado) para detectar atípicos y recuperar celdas atípicas incorrectamente recortadas.

A pesar de esta problemática, estos residuales estandarizados sí tienen localización próxima a 0 y medida de desviación típica robusta igual a 1. Como consecuencia, sí tiene sentido recuperar las celdas con residuales estandarizadas que estén menos alejados de un número fijado, que denotamos por `recover`, de desviaciones típicas estimadas. La elección de ese parámetro `recover` controla el grado de permisibilidad al recuperar observaciones. Con un valor de `recover` elevado se recuperan más celdas (se recortan menos celdas).

Como ilustración de esta idea, la Figura 3.7 muestra dos elecciones posibles del parámetro `recuperar` correspondientes a `recover=3` en color azul y a `recover=5` en rojo. En la primera de ellas, que suele ser la usual cuando suponemos normalidad, se comprueba que fuera de sus límites se descarta un número muy grande de observaciones, por lo que en principio no parece la más adecuada. En cambio, usando `recover=5` se tiene que el número de residuales que se deja fuera es muy inferior y más acorde con la contaminación que tenía el conjunto de datos original, por lo que sería más correcta. No obstante, recalcar que dependiendo del conjunto de datos y la aplicación final del procedimiento este parámetro puede ser ajustado en una dirección u otra.

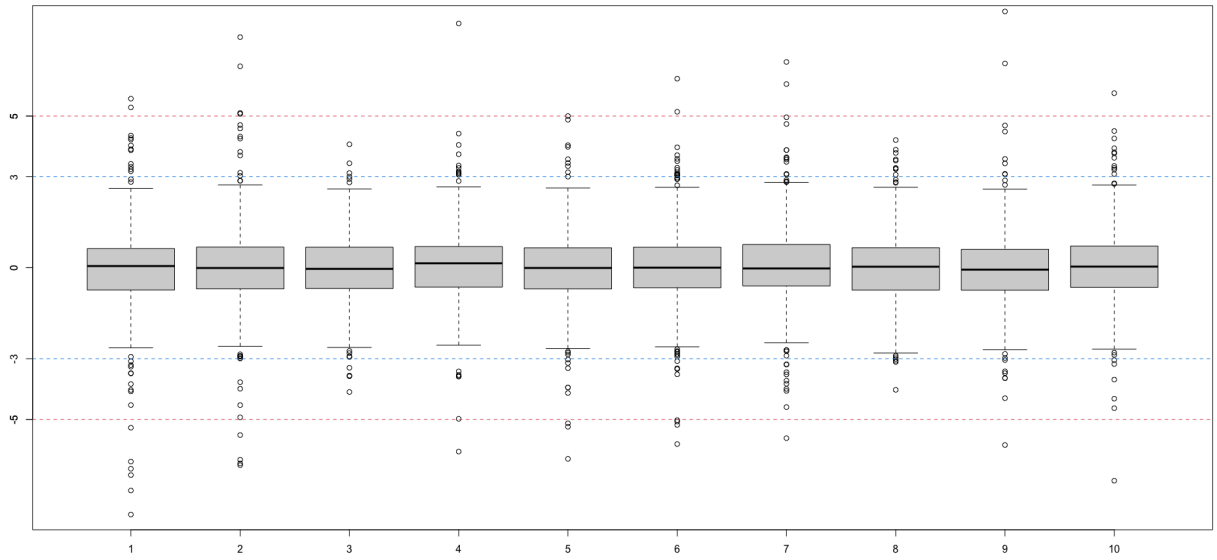


Figura 3.7: Gráfico de cajas de los residuos estandarizados por variable

El segundo paso de refinamiento corresponde a la *eliminación de observaciones enteras* si el número de celdas recortadas en una misma observación es muy alto. Para determinar si es alto o no este número de celdas, se utiliza el parámetro α_{LTS}, el mismo usado para la regresión LTS (el resto de detalles fueron ya expuestos en la Sección 2.8.5).

3.6. Funcionalidad gráfica

Para representar visualmente los resultados, se han desarrollado una serie de funciones tanto para el método propuesto en este trabajo de recortes por celdas. También se ha desarrollados funcionalidades gráficas para el RLG tratando la salida de la programación en [23].

3.6.1. Función `plot.cell.rlg`

A la hora de crear los resúmenes gráficos es necesario distinguir dos situaciones relativas a la naturaleza de los datos, si son funcionales o no, ya que interesarán posiblemente distintos tipos de representaciones. En primer lugar se detallan los desarrollados en ambos casos, tratando posteriormente de forma particular los gráficos a mayores interesantes para los datos funcionales.

3.6.1.1. Gráfico de residuales por celdas estandarizados

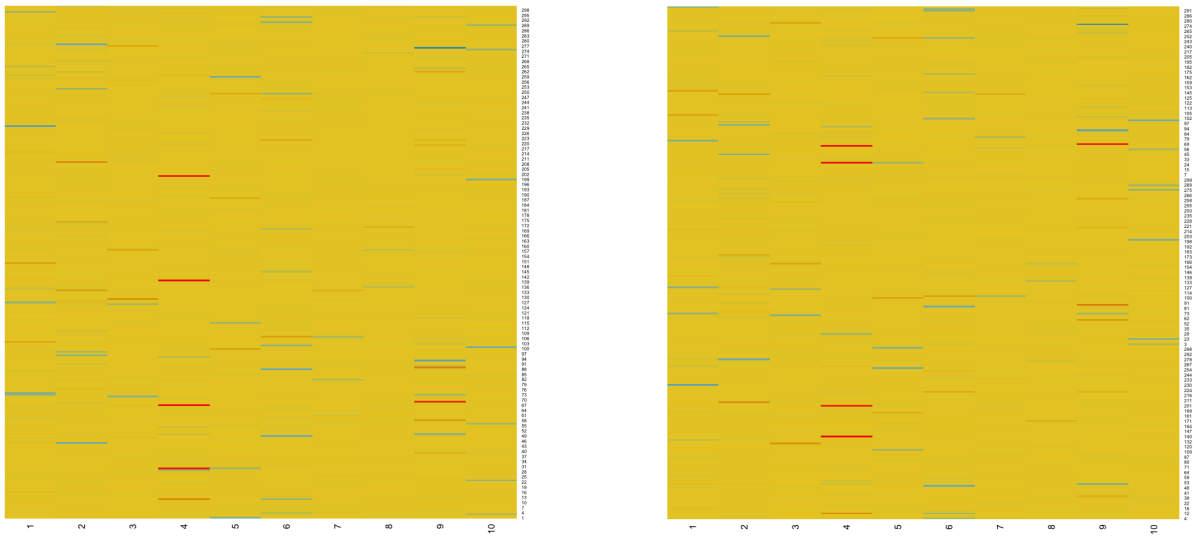
La representación de los residuos es útil puesto que nos proporciona una medida sobre el residual observado en cada celda particular respecto al ajuste por el correspondiente subespacio aproximante. Para ello se utilizarán los residuales estandarizados definidos en (3.2), los cuales dan lugar a una matriz $n \times p$. Tiene sentido considerar una representación tipo "heatmap" de estos

residuos estandarizados y, para una mejor visualización, se ha decidido usar una escala de colores concreta, donde los azules son los que están más alejados por ser pequeños y en rojo los que resultan elevados. Se ha pretendido que los valores de residuales tipificados en el intervalo $[-3, 3]$, es decir, valores no especialmente "atípicos" tengan colores "amarillentos".

Figura 3.8: Paleta de colores "Zissou1" continua con 256 tonos diferentes

La escala utilizada consta de 256 colores, generados a través de la paleta "Zissou1" (Figura 3.8) continua definida en el paquete `wesanderson` [36].

Una funcionalidad interesante puede ser agrupar las observaciones por clusters, para que así los residuos de cada grupo queden contiguos y se pueda comprobar de forma más sencilla si hay grupos que han quedado mejor o peor aproximados a los subespacios. En la Figura 3.9 se muestra un ejemplo con datos simulados de los residuos, tanto sin ordenar (Figura 3.9a) como ordenados (Figura 3.9b). En este caso, la reordenación no presenta una ventaja clara por la forma en que la contaminación ha sido generada, pero en otros casos puede ser de gran utilidad.

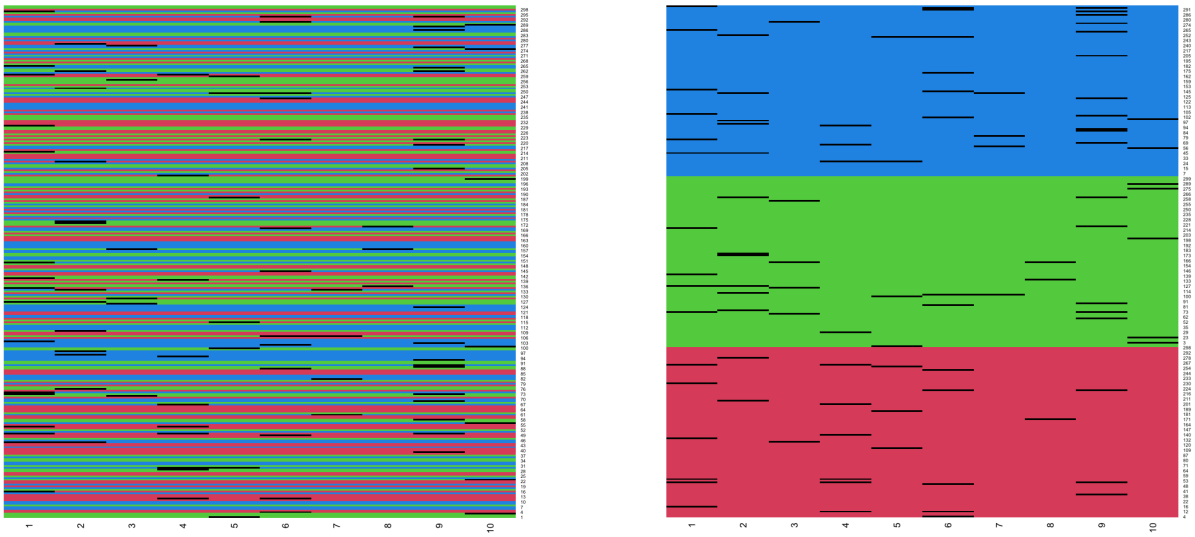


(a) Sin ordenar **(b)** Ordenados por clusters

Figura 3.9: Representación de los residuos estandarizados

3.6.1.2. Gráfico de asignación a clusters y celdas recortadas

Para comprobar de forma gráfica la asignación de cada observación a los clusters como se ha definido en la Ecuación 2.35, se ha realizado un "heatmap" de la matriz de dimensiones $n \times p$ en el que cada fila únicamente está asignada a un grupo, denotado por colores (sin una escala en concreto, usando en orden los definidos por R desde el 2). En la Figura 3.10 además son señaladas las celdas que han sido recortadas usando color negro.



(a) Sin ordenar **(b)** Ordenados por clusters

Figura 3.10: Asignación a grupos

De igual forma que en el caso anterior, se pueden ordenar las observaciones por cluster. En este caso es aún más interesante dado que nos da una muestra del tamaño de cada uno de los grupos y las observaciones que han sido clasificadas a cada cluster.

3.6.1.3. Gráfico de "*Scores*"

Los *scores* en los G clusters están dados por las matrices $A^g_{q_g}$ para $g = 1, ..., G$. Estos scores proporcionan las coordenadas de las observaciones x_i al ser representadas (mediante proyecciones ortogonales) en el subesapcio aproximante. La representación de los scores por grupos (Figura 3.11) es interesante por múltiples motivos. De forma análoga a lo que sucede en el ACP clásico, las observaciones que se muestran próximas en este gráfico, además de estar en el mismo grupo, toman valores parecidos en las variables analizadas al tener en cuenta la estructura de dependencia detectada en su grupo. Además, las observaciones que tengan scores más alejados indican que son observaciones atípicas dentro de dicho cluster.

El tamaño de estas coordenadas en $A^g_{q_g}$ es también informativo porque, en el algoritmo, se forzaba a que las columnas de las matrices $B^g_{q_g}$ tuvieran norma unitaria.

Dado que el número de clusters G puede ser elevado, para una correcta disposición y visualización en pantalla se ha usado la función `n2mfrow`, que indica las filas y columnas óptimas para la pantalla concreta que se deben de usar.

3.6.1.4. Gráfico de "*Loadings*"

Los loadings están determinados por las matrices $B^g_{q_g}$ para $g = 1, ..., G$, y las columnas de estas matrices proporcionaban los vectores unitarios que se usan para generar los subespacios aproximantes. Estos loadings, permiten ver el peso que se proporciona a cada variable en la obtención de los scores y proporcionan, por tanto, una interpretabilidad de los scores representados

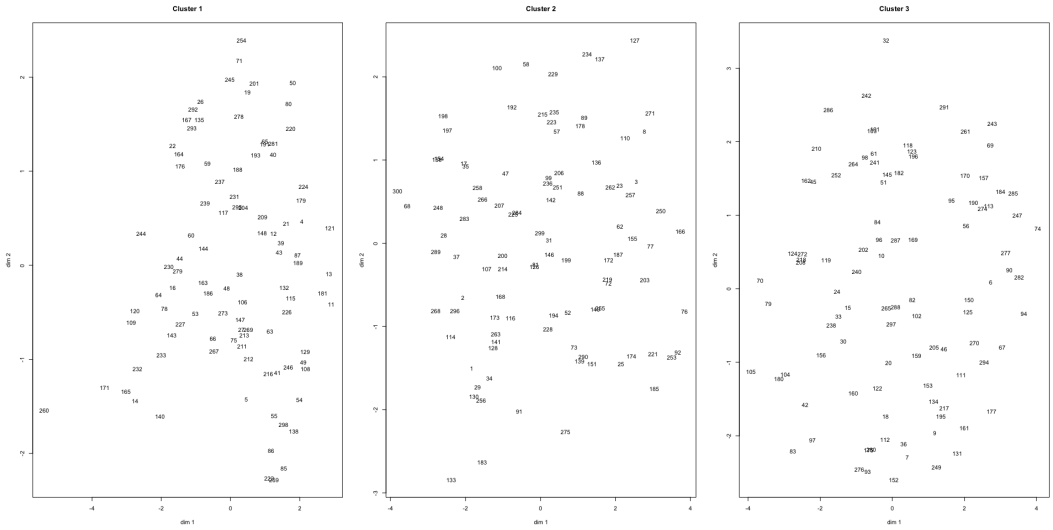


Figura 3.11: Representación de los *scores* por grupos

anteriormente, de forma análoga a lo que sucede en el ACP clásico.

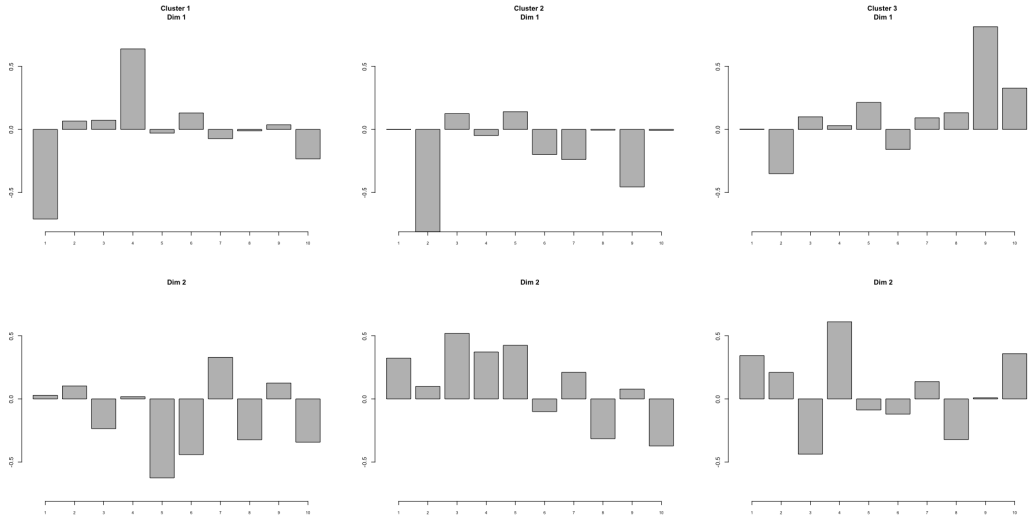


Figura 3.12: Representación de los *loadings* por dimensión intrínseca y grupo

Para cada uno de los gráficos de la Figura 3.12, en el eje de la x se representan los pesos para las p variables del conjunto de datos, mientras que en el eje de la y se representan los valores de b_i^j (todos en la misma escala para facilitar la comparación). Este gráfico es especialmente útil comprobar el signo y el peso para cada variable en cada variable por cluster. De esta forma podemos resaltar las variables a las que el procedimiento ha dado mayor peso en el subespacio aproximante.

Para cada cluster se realiza una serie de gráficos, dependiendo de la dimensión del subespacio aproximante q_g. Si esta dimensión q_g es mayor que 3, por interpretabilidad, se realizan un máximo de 3 gráficos por grupo. De nuevo, dependiendo del número de clusters y la dimensión del subespacio de cada uno de los grupos, se usará la función `n2mfrow` para optimizar la representación visual de estos gráficos.

Datos funcionales

A continuación se detallan los gráficos a mayores de los anteriores que se obtendrían si los datos de entrada son funcionales.

3.6.1.5. Gráfico de asignación de curvas a grupos

Dado que ahora las variables explicativas tienen un sentido temporal, donde $t_1 \leq t_2 \leq ... \leq t_p$ son los distintos tiempos en los que se ha medido una característica concreta, el resumen gráfico natural a hacer es usar el eje x como escala del tiempo y en el eje y representar los valores de la característica medidas a lo largo del tiempo. En este caso, las distintas curvas también están asignadas a grupos, por lo que se pueden usar colores para distinguir los clusters. A la hora de marcar los atípicos, que vuelven a ser celdas concretas, se hará marcando un punto negro en el instante concreto que ha ocurrido el atípico.

Para ilustrar la metodología en el caso funcional en esta sección, se han usado de datos funcionales simulados con diversos tipos de contaminación. Concretamente, se ha usado $G = 2$ grupos de curvas generados con funciones de media dadas por las funciones

$$f_1(x) = 5 + 10\sin(4\pi x)e^{-2x} + 5\sin(\pi\frac{x}{3}) + 2\cos(2\frac{\pi}{2}) \text{ y}$$
$$f_2(x) = 10 + 10\cos(4\pi x).$$

A estas funciones se les ha añadido una variabilidad extra o distintos "modos de variación" que justifican el enfoque de subespacios, mediante las funciones:

$$\begin{aligned} \rho_{1,1}(x) &= \sqrt{2}\cos(2\pi x) \qquad \rho_{1,2}(x) = \sqrt{2}\sin(2\pi x) \\ \rho_{2,1}(x) &= \sqrt{2}\sin(2\pi x) \qquad \rho_{2,2}(x) = \sqrt{2}\cos(2\pi x) \end{aligned} \tag{3.3}$$

Así, finalmente las funciones en el primer grupo vendrán dadas por realizaciones de la función

$$f_1(x) + r_1\rho_{1,1}(x) + r_2\rho_{1,2}(x),$$

y de la forma

$$f_2(x) + r_3\rho_{2,1}(x) + r_2\rho_{2,2}(x),$$

para las funciones del segunda grupo donde r_1 son realizaciones aleatorias independientes de una distribución $N(0, 3)$, r_2 de una distribución $N(0, 2)$ y r_3 de una distribución $N(0, 4)$, e independientes entre sí. Estos dos modos de variación en cada grupo, dados por las funciones $\rho_{1,1}$, $\rho_{1,2}$, $\rho_{2,1}$ y $\rho_{2,2}$ hacen que las dimensiones "intrínsecas" del subespacio funcional aproximante sean $q_1 = q_2 = 2$.

Para discretizaciones de estas funciones en una rejilla de tamaño p, se sumará también errores aleatorios en esas p evaluaciones de las curvas en esas evaluaciones independientes y con distribución común $N(0, 0.5)$. Finalmente, se ha seguido un mecanismo de generación de valores atípicos en las funciones de forma "dispersa". Este esquema de simulación será la base de un esquema de

simulación más general que será presentado en la Sección 4.2.

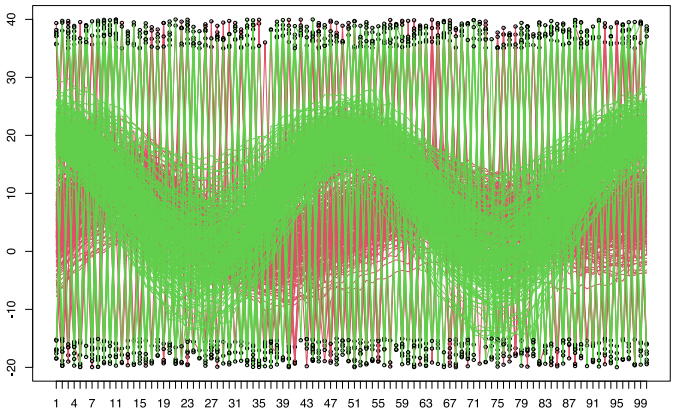


Figura 3.13: Asignación a grupos de las diferentes curvas, representación conjunta, con celdas recortadas como círculos negros.

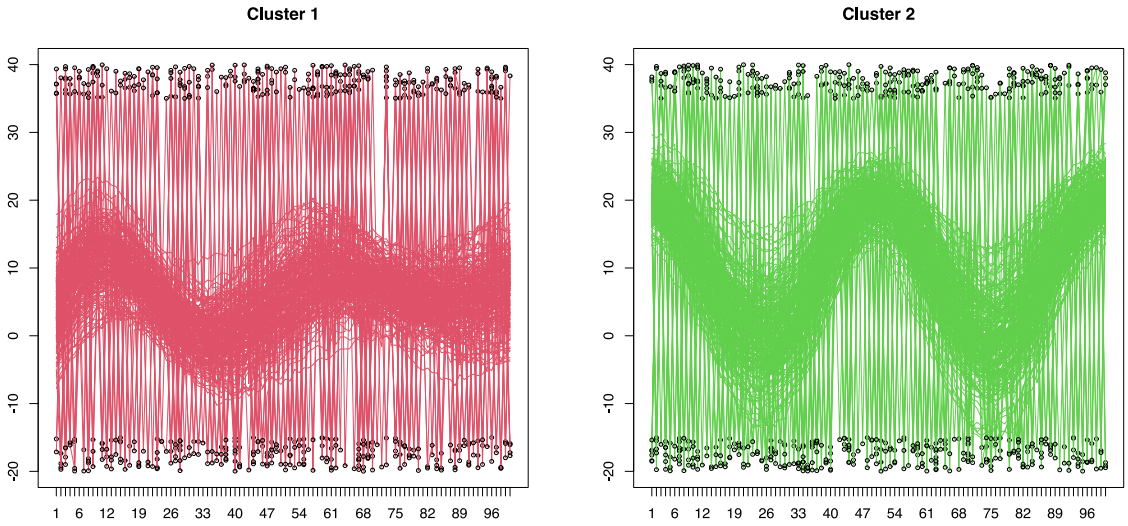


Figura 3.14: Asignación a grupos de las diferentes curvas al separar por grupos, con celdas recortadas como círculos negros.

La Figura 3.13 muestra la asignación a clusters encontrada, con los valores atípicos marcados mediante círculos negros. No se aprecia demasiado bien al haber bastantes curvas y superponerse, por el tipo particular de contaminación generada. Puede ser más interesante la representación en la Figura 3.14, en la cual se han separado los distintos grupos para que sea visualmente más sencillo distinguirlos, con los valores atípicos de las funciones también marcados como círculos negros.

3.6.1.6. Gráfico de curva medias y modos de variación en grupos

Otra representación interesante para ver la tendencia de cada grupo en el caso funcional es representar la curva media en cada instante de tiempo (media puntual) de las curvas que fueran asignadas a un cluster. En la Figura 3.15 se muestra un ejemplo con los datos anteriores, en los cuales se ve claramente la diferencia entre ambos clusters y se ve que se aproximan bastante bien las funciones f_1 y f_2 de medias.

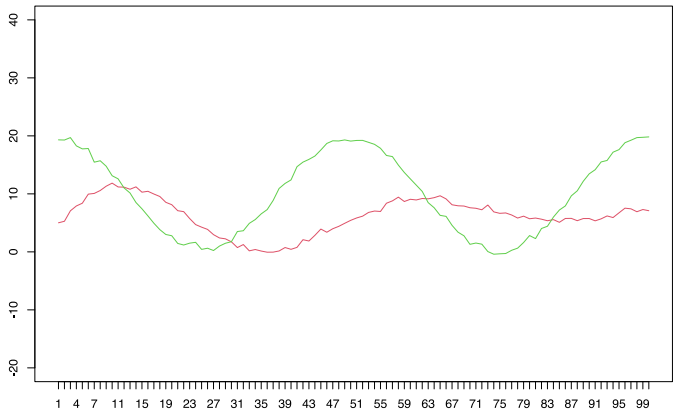


Figura 3.15: Curvas medias por grupos

En el caso de los datos funcionales, podemos ver fácilmente como el gráfico de "loadings" es útil para ver cómo los términos de las matrices $B^g_{q_g}$ aproximan a los "modos de variación" en las funciones $\rho_{1,1}$ $\rho_{1,2}$, $\rho_{2,1}$ y $\rho_{2,2}$. Recordemos que en esta simulación los modos fueron prefijados en (3.3). En la Figura 3.16 se puede comprobar que se las aproximaciones dadas por las columnas de las matrices $B^g_{q_g}$ aproximan muy correctamente estas funciones senos y cosenos para cada grupo y dimensión.

3.6.2. Función `plot.rlg`

Se ha aprovechado el trabajo realizado en la obtención de los gráficos resultantes de `plot.cell.rlg` para introducir procedimientos gráficos que permitan visualizar cómodamente los resultados de aplicar el procedimiento RLG (con recortes por observaciones). La mayor parte de los gráficos se corresponden, por tanto, con algunos ya especificados anteriormente. En particular, esto sucede con los gráficos de asignación a clusters (Sección 3.6.1.2), los de "scores" (Sección 3.6.1.3) y "loadings" (Sección 3.6.1.4) pero con pequeñas diferencias. Por ejemplo, en el gráfico de la asignación a grupos, al ser un método que recorta observaciones en vez de celdas, únicamente se marcan las filas recortadas en vez de hacerlo celda a celda. Adicionalmente, la implementación de partida en C++ no proporcionaba los scores, por lo que se ha precisado su cálculo explícito para su posterior representación gráfica.

Se ha obtenido también un "scree plot" con los autovalores asociados a la representación en

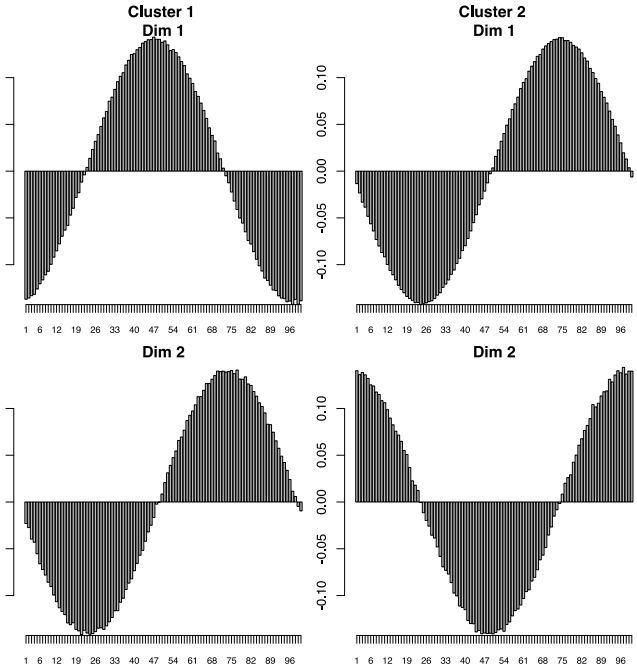


Figura 3.16: Aproximación a los modos de variación en grupos por las columnas de las matrices $B_{q_g}^g$

componentes principales (asociados a los autovectores que proporcionan la base ortonormal del subespacio afín aproximante). En el eje de la x se representan el número de dimensiones recogidas (como máximo p) para los posibles subespacios aproximantes en cada grupo y en el eje de la y se representa la variabilidad recogida por cada una de esas dimensiones. Este gráfico es útil para determinar dimensiones intrinsecas q_g razonables en los G clusters. Nótese también que las observaciones atípicas, en virtud al recorte aplicado mediante el RLG, no van a afectar a la determinación de estas dimensiones intrínsecas.

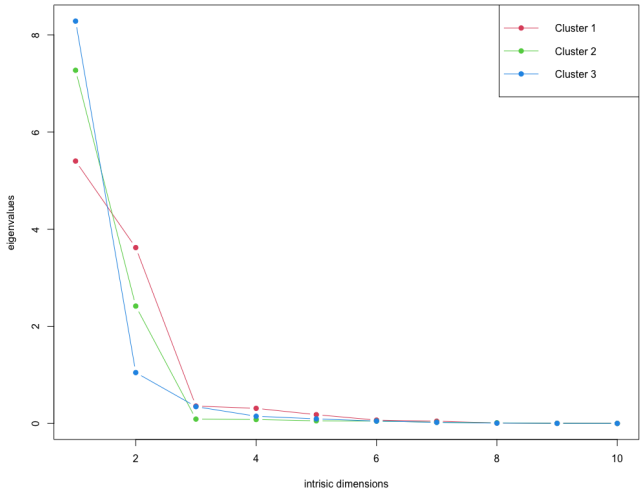


Figura 3.17: Gráfico "scree plot" por grupos asociado a la salida de RLG

Se muestra en la Figura 3.17 un ejemplo con los datos no funcionales. En este caso concreto,

se puede ver que los autovalores son todos próximos a 0 cuando se consideran dimensiones en cualquiera de los clusters estrictamente mayores que 2, mientras que estos autovalores son mayores para dimensiones 2 o menores. Este gráfico "scree plot" sugiere por tanto la elección $q_1 = q_2 = q_3 = 2$ como dimensiones intrinsecas en este problema. Vemos que las dimensiones intrisecas coinciden exactamente con las dimensiones reales que fueron usadas para generar los datos.

Este gráfico puede ser también especialmente útil para ser realizado antes de ejecutar `cell.rlg`, tras considerar un recorte suficientemente grande en RLG para eliminar todas las celdas más atípicas y que podrían distorsionar la estimación de las dimensiones intrínsecas. Así, nos proporciona una recomendación razonable de las dimensiones intrínsecas q_g subyacentes a usar en el posterior procedimiento de recorte por celdas.

3.7. Pseudocódigo

3.7.1. LTS

El algoritmo implementado en C++ es el denominado FAST-LTS para mejorar el rendimiento de LTS [37] y se muestra en el Algoritmo 1.

Algoritmo 1 FAST-LTS

Input: $X \in \mathbb{R}^{n \times p}$ matriz de datos con filas x_i; $Y \in \mathbb{R}^p$ vector con la variable dependiente con valores y_i; $\alpha \in [0, 1]$ tamaño de recorte, `nsamp` número de iteraciones en la inicialización, `n_best` número de etapas de concentración

Output: $b^* \in \mathbb{R}^p$ coeficientes estimados; H^* índices de las observaciones no recortadas; Q^* mejor valor de la función objetivo; residuos r_i^*

$n \leftarrow$ número de observaciones (filas) de X
$p \leftarrow$ número de variables (columnas) de X
$h \leftarrow \text{máx}(p, \lceil n(1-\alpha) \rceil)$
for $l \leftarrow 1$, `nsamp` **do** ▷ Inicialización de los coeficientes
$H_0 \leftarrow$ vector de p observaciones aleatorias de un total de n (no hacen falta $p+1$ porque es una regresión sin intercept)
$X_{H_0} \leftarrow$ matriz tomada de X con los índices de H_0
$Y_{H_0} \leftarrow$ vector tomado de Y con los índices de H_0
$b \leftarrow \arg\text{mín}_{b \in \mathbb{R}^p} ||Y_{H_0} - X_{H_0} b||^2$
for $j \leftarrow 1, 2$ **do** ▷ sólo 2 etapas de concentración
$r_i^2 \leftarrow (y_i - x_i^t b)^2$
$H \leftarrow \{i : r_i^2 \leq r_{(h)}^2\}$
$X_H \leftarrow$ matriz tomada desde X con los índices de H
$Y_H \leftarrow$ vector tomado desde Y con los índices de H
$b \leftarrow \arg\text{mín}_{b \in \mathbb{R}^p} ||Y_H - X_H b||^2$
end for

$b[l] \leftarrow b$
$Q[l] \leftarrow \sum_{i=1}^{h} r_{(i)}^2$
end for
$Q \leftarrow Q[\texttt{n\_best}]$ ▷ `n_best` valores con los menores valores de la función objetivo
$\tilde{b} \leftarrow b[\texttt{n\_best}]$
for $m \leftarrow 1, \texttt{n\_best}$ **do** ▷ Etapas de concentración hasta converger de las soluciones más prometedoras
$b_1 \leftarrow \tilde{b}(m)$
do
$b_0 \leftarrow b_1$
$r_i^2 \leftarrow (y_i - x_i^t b_1)^2$
$H \leftarrow \{i : r_i^2 \leq r_{(h)}^2\}$
$X_H \leftarrow$ matriz tomada desde X con los índices de H
$Y_H \leftarrow$ vector tomado desde Y con los índices de H
$b_1 \leftarrow \arg\min_{b_1 \in \mathbb{R}^p} ||Y_H - X_H b||^2$
while $\sum_{j=1}^{p} |b_0 - b_1| < 10^{-8}$
if $Q^* > \sum_{i=1}^{h} r_{(i)}^2$ **then** ▷ Guardar los mejores según la función objetivo
$b^* \leftarrow b_1$
$H^* \leftarrow H$
$Q^* \leftarrow \sum_{i=1}^{h} r_{(i)}^2$
$(r_i^2)^* \leftarrow r_i^2$
end if
end for
return $(b^*, H^*, Q^*, (r^2)^*)$

3.7.2. `cell.rlg`

Explicados los distintos pasos del método, se presenta el pseudocódigo del algoritmo iterativo (Algoritmo 2) que integra todos los pasos implementados en la función `cell.rlg` en R.

Algoritmo 2 Método propuesto `cell.rlg`

Input: $X \in \mathbb{R}^{n\times p}$ matriz de datos; $q_g \in \mathbb{N}^G$ dimensiones intrínsecas de cada grupo; `init_method` método de inicialización; $\alpha \in [0,1]$ proporción de recorte para los pesos; $\alpha_{\text{LTS}} \in [0,1]$ proporción de recorte de LTS; $\alpha_{\text{INIT}} \in [0,1]$ proporción de recorte para el método de inicialización; `loop1` número de bucles externos; `loop2` número de bucles internos; `nstart` número de inicializaciones aleatorias; `do_ddc` indicador de realizar DDC; `scale` indicador para hacer escalado, `refinamiento` indicador para hacer la etapa de refinamiento, `recover` número de desviaciones típicas a recuperar; `functional` indicador si son datos funcionales, `f` argumento para lowess; `knots` nodos internos que definen la representación con B-splines

Output: `X` matriz de entrada, q_g dimensiones intrínsecas de cada grupo, `init` método de inicialización, `functional` indicador si son datos funcionales, $\widehat{x}^*_{ij}$ matriz de predicciones en las celdas; $\hat{s}^*$ valor de la función objetivo; $(A^g_{q_g})^*$ loadings; $(B^g_{q_g})^*$ scores; $(m^g)^*$ centros; $g(i)$ asignación de cada observación; w^*_{ij} indicador de recorte 0-1 de cada celda

$n \leftarrow$ número de observaciones (filas) de X
$p \leftarrow$ número de variables (columnas) de X
if `scale` **then**
  Realizar el escalado robusto con media y MAD por filas
end if
if `do_ddc` **then**
  $X \leftarrow \text{DDC}(X)$
end if
if `functional` **then**
  $X\texttt{.s} \leftarrow lowess(X, \texttt{f})$
  $P \leftarrow \texttt{knots} + 4$ ▷ B-splines, se reduce la dimensión a P
  $B_{m,M} \leftarrow$ Matriz con la evaluación de los B-splines en los nodos con $\texttt{df} = P$
  for $i \leftarrow 1, n$ **do**
    $b \leftarrow \arg\min_{b\in\mathbb{R}^p} ||X\texttt{.s}_i - B_{m,M}b||^2$
    $x_i \leftarrow b$
  end for
end if

if `init` = TCLUST **then** ▷ Inicialización de los parámetros
$m^g \leftarrow$ centros obtenidos de TCLUST
$B^g_{q_g} \leftarrow$ autovectores de las matrices de dispersión obtenidas con TCLUST
else if `init` = RLG **then**
$m^g \leftarrow$ centros obtenidos de RLG
$B^g_{q_g} \leftarrow$ matriz U con los q_g autovectores óptimos devueltos por RLG
end if
if `functional` **then** ▷ Recuperación de la dimensión p de partida
$m^g \leftarrow B_{m,M} m^g$
$B^g_{q_g} \leftarrow B_{m,M} B^g_{q_g}$
end if
for $it_1 \leftarrow 1$, `loop1` **do** ▷ Bucle externo
$A^g_{q_g} \leftarrow$ coeficientes de LTS ▷ Inicialización de $A^g_{q_g}$ y $g(i)$ con LTS
Suma de los $\lceil n(1-\alpha_{\text{LTS}})\rceil$ residuales menores del LTS menores
$g(i)$ dependiendo de esas sumas de menores residuales del LTS
for $it_2 \leftarrow 1$, `loop2` **do** ▷ Bucle interno
$\widehat{X} \leftarrow m^g + B^g_{q_g} A^g_{q_g}$
$r^2_{ij} \leftarrow (x_{ij} - \widehat{x}_{ij})^2$
$w_{ij} \leftarrow I_{r^2_{ij} \leq r^2_{(h:n)j}}$ ▷ Actualización de los pesos
$\sum_{j=1}^p w^g_{ij}(x_{ij} - m^g_j) b^g_j = \left(\sum_{j=1}^p w^g_{ij} b^g_j (b^g_j)^t\right) a^g_i, \qquad i = 1, ..., n \text{ y } g = 1, ..., G$ ▷ Actualización de los parámetros
$\sum_{j=1}^p w^g_{ij}(x_{ij} - (a^g_i)^t b^g_j) = \sum_{i=1}^n w^g_{ij} m^g_j, \qquad j = 1, ..., p \text{ y } g = 1, ..., G$
$\sum_{j=1}^p w^g_{ij}(x_{ij} - m^g_j) a^g_i = \left(\sum_{i=1}^n w^g_{ij} a^g_i (a^g_i)^t\right) b^g_j, \qquad j = 1, ..., p \text{ y } g = 1, ..., G$
$\hat{s} \leftarrow \sum_{i,j} w^{g(i)}_{ij} r^2_{ij}$ ▷ Cálculo de la función objetivo
end for
if $\hat{s} < \hat{s}^*$ **then** ▷ Guardar los mejores según la función objetivo
$\hat{s}^* \leftarrow \hat{s}$
$(A^g_{q_g})^* \leftarrow A^g_{q_g}$
$(B^g_{q_g})^* \leftarrow B^g_{q_g}$
$(m^g)^* \leftarrow m^g$
$\widehat{x}^*_{ij} \leftarrow \widehat{x}_{ij}$
$g(i)^* \leftarrow g(i)$
$(w_{ij})^* \leftarrow w^{g(i)}_{ij}$
end if
end for
Estandarizar los residuos por variables y grupos de forma robusta en $\tilde{r}_{ij}$ ▷ Residuos estandarizados
if `refinamiento` **then**
$w^*_{ij} \leftarrow I_{\{w_{ij}=1\}} \cup \{|\tilde{r}_{ij}| < \texttt{recover}\}$ ▷ Recuperar celdas
$w^*_{ij} \leftarrow 0$ para los i tales que $\#\{w^{g(i)}_{ij} = 0 : j = 1, ..., p\}/p > \alpha_{\text{LTS}}$ ▷ Eliminar observaciones completas
end if
return (`X`, q_g, `init`, `functional`, $\widehat{x}^*_{ij}$, $\hat{s}^*$, $(A^g_{q_g})^*$, $(B^g_{q_g})^*$, $(m^g)^*$, $g(i)^*$,w^*_{ij})

3.8. Especificación de las funciones

Para cerrar este capítulo, se detallarán los argumentos y las salidas que devuelven las funciones desarrolladas: `cell.rlg`, `plot.cell.rlg` y `plot.rlg`.

3.8.1. `cell.rlg`

3.8.1.1. Argumentos

Los únicos argumentos obligatorios son `x` y `d`.

Argumento	**Descripción**
`x`	Matriz o *dataframe* $n \times p$ con los datos de entrada
`d`	Vector con las dimensiones intrínsecas q_g de cada grupo $g \in \{1, ..., G\}$
`init.method`	Algoritmo para la inicialización de las matrices $B_{q_g}^g$ y m^g. Posibles valores: `tclust` o `rlg`. Valor por defecto `rlg`
`alpha`	Proporción de celdas inicialmente recortadas. Debe de ser un número real en el rango $[0, 1]$. Valor por defecto 0.1
`alpha.lts`	Proporción de las celdas a recortar por el método LTS en las regresiones robustas por filas. Debe de ser un número real en el rango $[0, 1]$. Valor por defecto 0.25
`alpha.init`	Proporción de observaciones a recortar al inicializar con TCLUST o RLG, dependiendo del método escogido. Debe de ser un número real en el rango $[0, 1]$. Valor por defecto 0.2
`loops1`	Número de bucles externos a ejecutar. Valor por defecto 4
`loops2`	Número de bucles internos a ejecutar. Valor por defecto 10
`nstart`	Número de inicializaciones aleatorias del algoritmo de inicialización (TCLUST o RLG) a realizar. Valor por defecto $\sum_{g=1}^{G} q_g \times 40$ (donde G es el número de clusters)
`do.ddc`	Indicador de si se debe aplicar el algoritmo DDC antes de inicializar los pesos. Valor por defecto `FALSE`
`scale`	Indicador para realizar o no un escalado robusto a la matriz de entrada. Valor por defecto `TRUE`
`refinamiento`	Indicador de si se debe realizar la etapa de refinamiento. Valor por defecto `TRUE`

`recover`	Número de desviaciones típicas a recuperar el refinamiento. Solo tiene validez si se ha marcado hacer `refinamiento`. Valor por defecto 10
`functional`	Indicador de datos funcionales. Valor por defecto `FALSE`
`f`	Correspondiente al argumento `f` para el suavizado con lowess. Solo tiene validez si se ha marcado que los datos son funcionales. Valor por defecto 1/5
`knots`	Número de nodos internos que definen el B-spline. Solo tiene validez si se ha marcado que los datos son funcionales. Valor por defecto 4

Tabla 3.2: Argumentos de entrada de la función `cell.rlg`

3.8.1.2. Valor que devuelve

Devuelve un objeto S3 de la clase `cell.rlg` con los atributos mostrados en la Tabla 3.3.

Atributo	**Descripción**
`x`	Matriz o dataframe $n \times p$ con los datos de entrada
`d`	Vector con las dimensiones intrínsecas q_g de cada grupo $g \in \{1, ..., G\}$
`init`	Objeto con el resultado del método de inicialización seleccionado
`functional`	Indicador de datos funcionales
`xpred`	Matriz $n \times p$ con los datos estimados $\widehat{x}_{ij}$
`residuals`	Residuos estandarizados r_{ij} robustos
`shat`	Valor de la función objetivo
`A`	Lista con las matrices de *Scores* $A^g_{q_g}$
`B`	Lista con las matrices de *Loadings* $B^g_{q_g}$
`m`	Lista con las vectores de medias m^g
`cluster.cell`	Asignación al cluster concreto de cada celda. Los clusters se definen de 1 en adelante. El valor 0 indica que dicha celda ha sido recortada
`cluster`	Asignación a clusters de cada observación $g(i)$. Los clusters se definen de 1 en adelante. El valor 0 indica que dicha observación ha sido recortada totalmente

`cluster.cell.raw`	Asignación al cluster concreto de cada celda antes de la etapa de refinamiento. Los clusters se definen de 1 en adelante. El valor 0 indica que dicha celda ha sido recortada
`cluster.raw`	Asignación a clusters de cada observación $g(i)$ antes de la etapa de refinamiento. Los clusters se definen de 1 en adelante. El valor 0 indica que dicha observación ha sido recortada totalmente
`evo`	Valores con la evolución de la función de pérdida en cada iteración

Tabla 3.3: Valores devueltos por la función `cell.rlg`

3.8.2. `plot.cell.rlg`

El único argumento obligatorio es `obj`. La forma de invocar la función es directamente mediante `plot(obj)`.

Atributo	**Descripción**
`obj`	Objeto S3 de clase `cell.rlg`
`which`	Gráficos a realizar, puede tomar los siguientes valores (solo uno de ellos): • `all`: todos los gráficos posibles • `residuals`: residuos por celda (ver Sección 3.6.1.1) • `clusters`: asignación a clusters (ver Sección 3.6.1.2) • `scores` (ver Sección 3.6.1.3) • `loadings` (ver Sección 3.6.1.4) Además, si los datos de entrada son funcionales, es posible realizar: • `curves`: asignación de curvas a grupos (ver Sección 3.6.1.5) • `curves.splitted`: asignación de curvas a grupos separados • `mean.curves`: curva media de cada grupo (ver Sección 3.6.1.6)
`sort`	Indicador si se deben ordenar por clusters las observaciones. Solo válido para los gráficos `residuals` y `clusters`

Tabla 3.4: Argumentos de entrada de la función `plot.cell.rlg`

3.8.3. `plot.rlg`

El único argumento obligatorio es `obj`. La forma de invocar la función es directamente mediante `plot(obj)`.

Atributo	**Descripción**
`obj`	Objeto S3 de clase `rlg`
`which`	Gráficos a realizar, puede tomar los siguientes valores (solo uno de ellos): • `all`: todos los gráficos posibles • `clusters`: asignación a clusters (ver Sección 3.6.1.2) • `scores` (ver Sección 3.6.1.3) • `loadings` (ver Sección 3.6.1.4) • `eigenvalues` "Scree plot" (ver Sección 3.6.2)
`sort`	Indicador si se deben ordenar por clusters las observaciones. Solo válido para los gráficos `residuals` y `clusters`

Tabla 3.5: Argumentos de entrada de la función `plot.rlg`

Capítulo 4

Ejemplos y simulación

Para comprobar que las modificaciones y funcionalidades implementadas funcionan correctamente y analizar la dependencia de la metodología de diversas características, se han realizado una serie de simulaciones con datos artificiales tanto para datos no funcionales como funcionales.

Para todos estos conjuntos de datos simulados se han medido una serie de métricas como son el "tiempo de ejecución" y el "valor de la función objetivo". Está claro que un menor tiempo de computación y un menor valor de la función objetivo (problema definido por mínimos) son preferible. También se considerará una métrica que denotaremos por "SSE" y que dependerá de la suma de los errores al cuadrado (SSE). Este último criterio se basa en

$$SSE = \sum_{(i,j)\in\mathcal{G}} (x_{ij} - \hat{x}_{ij})^2$$

donde $\mathcal{G}$ serán los índices (i, j) con residuales más pequeños una vez que se permite descartar los residuales más grandes y que podrían corresponder a la fracción ε_0 de celdas contaminadas (no se pretenden ajustar las celdas atípicas y sí las celdas que no lo son). Para ser más precisos, si

$$\tilde{r}^2_{(1)} \leq \tilde{r}^2_{(2)} \leq .... \leq \tilde{r}^2_{(n\cdot p)}$$

fueran los $n \cdot p$ residuales $r^2_{ij} = (x_{ij} - \hat{x}_{ij})^2$ ordenados de menor a mayor entonces

$$\mathcal{G} = \{(i, j) : r^2_{ij} \leq \tilde{r}^2_{(np(1-\varepsilon_0))}\}.$$

Evidentemente, el cálculo del SSE requiere de conocer la fracción verdadera de celdas contaminadas ε_0, pero esto es conocido en el caso simulado ya que nosotros controlamos dicha contaminación. Si el método ha ajustado correctamente, los residuos considerados para el cálculo del estadístico SSE deberían ser pequeños, al corresponderse con celdas no contaminadas y dejar fuera los residuales al cuadrado grandes de las celdas que se desvían más del comportamiento mayoritario. Por tanto, es también preferible alcanzar valores de SSE pequeños.

Finalmente, en algunos de los casos, se ha calculado también el error de clasificación, como la fracción de observaciones que son asignadas a grupos incorrectos o incorrectamente recortadas.

Cada simulación se ha replicado $B = 100$ veces para así tener una medida de la variabilidad, con semillas fijadas a la hora de generar los datos, permitiendo así la reproducibilidad de los resultados de la experimentación.

4.1. Datos no funcionales

Para el caso de *datos no funcionales*, las simulaciones realizadas parten de una función definida en R utilizada para simplificar la generación de datos mediante una nueva función denominada `genMN(mdim, ndim, n, var, sep_means)` modificando la función `genLP` del paquete `T4cluster`. Esta función modificada sirve para generar $G = 3$ clusters entorno a subespacios de dimensión `mdim` en un espacio global de dimensión `ndim`. Es posible especificar el número de datos por grupo, con el argumento `n`, así como la varianza de los datos de errores generados a partir de una distribución Normal y la separación entre los distintos centros mediante el parámetro `sep_means`.

1. *Método de inicialización*: El objetivo es comprobar el funcionamiento de RLG frente a TCLUST como procedimiento para inicializar los parámetros según la dimensión del conjunto de datos. Para ello se han generado conjuntos de datos de dimensiones $p \in \{10, 40\}$ con $G = 3$ clusters distintos de dimensión intrínseca $q_1 = q_2 = q_3 = 2$. Cada cluster contiene 100 observaciones y en ambos métodos de inicialización se han mantenido los mismos valores por defecto en el número de inicializaciones aleatorias.

2. *Dimensión intrínseca*: Para verificar el impacto que tiene escoger uno u otro valor de q_g para cada grupo, se han generado unos datos $n = 600$, $p = 10$, $G = 3$ con $(q_1, q_2, q_3) = (1, 2, 3)$. Se han probado los siguientes conjuntos de dimensiones intrínsecas (q_1, q_2, q_3) dados por $(1, 1, 1)$, $(2, 2, 2)$, $(3, 3, 3)$, $(4, 4, 4)$ y $(1, 2, 3)$. En todos ellos se ha utilizado el mismo número `nsamp=1000` para que fueran comparables.

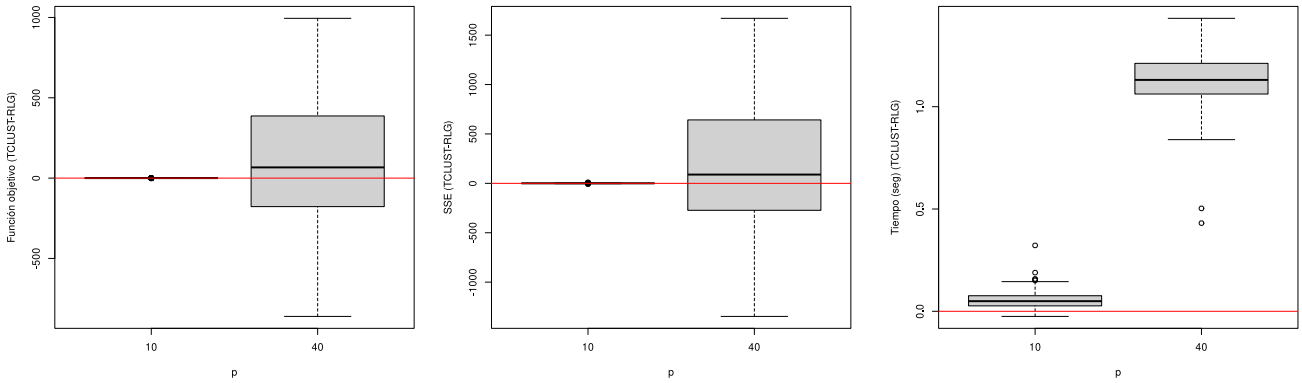
3. *Contaminación y refinamiento*: Se quiere constatar la influencia de las observaciones contaminadas junto al valor α especificado. Es decir, ver el comportamiento si α es menor que la contaminación introducida, así como la mejora que se podría obtener con un α (preventivamente) mayor al necesario pero aplicando las etapas de refinamiento. Para ello, se ha generado un conjunto de datos de $n = 300$ con $G = 3$ (100 observaciones por grupo), $p = 10$ y $q_1 = q_2 = q_3 = 2$. A estas celdas se les ha introducido un ruido reemplazando una fracción $\varepsilon_0 = 0.05$ del total de celdas por celdas atípicas distribuidas según $U(-3, 3)$ independientes. Se han probado tamaños de recorte $\alpha \in \{0, 0.05, 0.10, 0.15, 0.20\}$.

4.1.1. Método de inicialización

Por la forma en que se realiza el experimento, nos encontramos con datos pareados. Para cada conjunto de datos simulado se han ejecutado ambos métodos de inicialización, por lo que parece

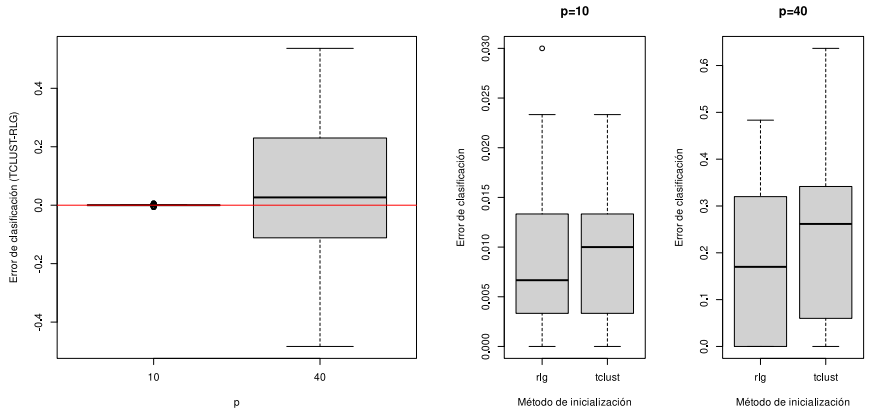
razonable calcular la diferencia en las métricas de uno menos el otro. En los resultados que se exponen a continuación, se ha restado el resultado de las métricas anteriormente presentadas al aplicar TCLUST menos el resultado de aplicar RLG, sobre el mismo conjunto de datos (TCLUST−RLG).

En las Figuras 4.1a y 4.1b se han realizado unos diagramas de cajas con la diferencia de los valores obtenidos en la función objetivo y SSE para cada simulación ($B = 100$). Ambos parecen funcionar de manera similar cuando la dimensión es baja pero al aumentarla se percibe que TCLUST se comporta quizás mejor, al desplazarse ligeramente la mediana de las diferencias a valores positivos.



(a) Función objetivo **(b)** SSE **(c)** Tiempo de ejecución

Figura 4.1: Diagramas de caja de la diferencia entre TCLUST y RLG frente a la dimensión p



(a) Diferencia **(b)** Separados

Figura 4.2: Comparación de los errores de clasificación para los métodos de inicialización según la dimensión p

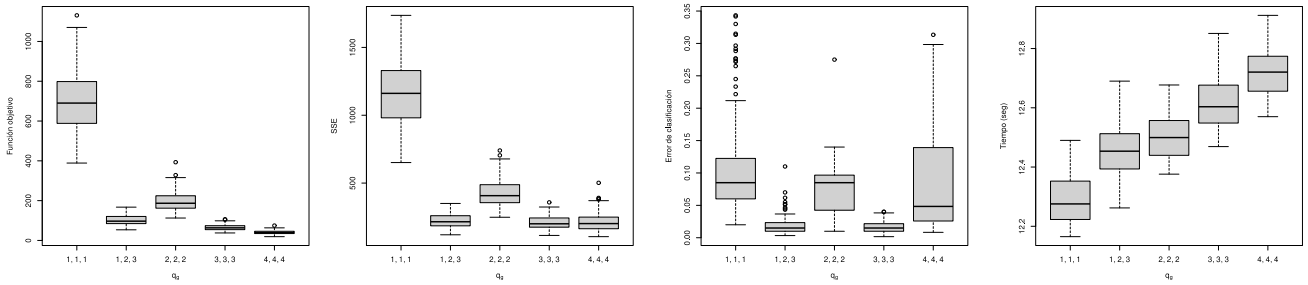
Como comparación de esta posible diferencia, es de utilidad comparar las tasas de los errores de clasificación en la Figura 4.2. En la Figura 4.2a se aprecia de nuevo que TCLUST funciona ligeramente peor. Para comprobar la escala de esta diferencia se ha realizado un par de gráficos, uno por dimensión p, con ambos métodos de inicialización y que aparece en la Figura 4.2b. Ambos

procedimientos se diferencian muy poco cuando $p = 10$, pero la diferencia es más notable al aumentar p.

Al fijarnos en el tiempo de ejecución, la Figura 4.1c sí que parece mostrar más grandes diferencias. TCLUST tarda un tiempo bastante superior a RLG. Para dimensión $p = 10$ estas diferencias no son grandes, pero en el caso $p = 40$ la diferencia es más grande.

Resumiendo lo visto, parece que hay una clara ventaja al usar RLG con dimensiones mayores, tanto si nos fijamos en el tiempo computacional pero también en la calidad de los resultados, con un error de clasificación y SSE inferiores.

4.1.2. Dimensión intrínseca



(a) Función objetivo **(b)** SSE **(c)** Error de clasificación **(d)** Tiempo de ejecución

Figura 4.3: Diagramas de caja frente a los valores de q_g

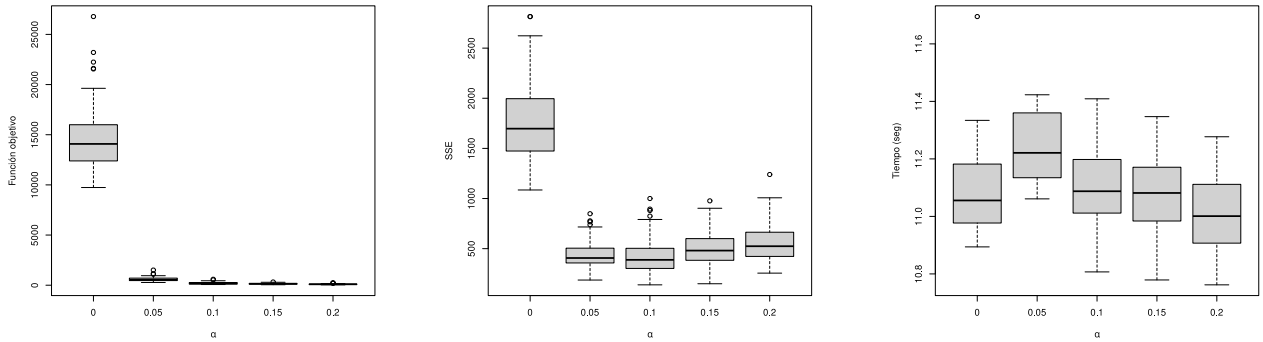
Los datos reales se han simulado usando como dimensiones intrínsecas diferentes $(q_1, q_2, q_3) = (1, 2, 3)$. En primer lugar, si nos fijamos en los tiempos de ejecución (Figura 4.3d) parece que se van claramente incrementando con la dimensión intrínseca, como era lógico esperar. Tarda menos buscar rectas, después buscar planos, etc.

Sobre el gráfico de la función objetivo en Figura 4.3a, se comprueba claramente que usando como dimensión intrínseca $(q_1, q_2, q_3) = (1, 1, 1)$ se obtienen resultados bastante pobres, al ser mucho más alta la dimensión en algunos subespacios. Buscando subespacios de dimensión $(q_1, q_2, q_3) = (2, 2, 2)$ mejoran los resultados, aunque siguen siendo significativamente inferiores al resto de los casos. Con subespacios de dimensiones más altas, $(q_1, q_2, q_3) = (3, 3, 3)$ y $(4, 4, 4)$, parece obtenerse mejores resultados en la función de pérdida y en el SSE (Figura 4.3b). Este resultado es razonable dado que el caso $(q_1, q_2, q_3) = (1, 2, 3)$ está incluido en $(q_1, q_2, q_3) = (3, 3, 3)$ y este a su vez en $(q_1, q_2, q_3) = (4, 4, 4)$.

Por último, al fijarnos en los errores de clasificación en Figura 4.3c, sí que se aprecian diferencias entre $(q_1, q_2, q_3) = (1, 2, 3)$, $(q_1, q_2, q_3) = (3, 3, 3)$ y $(q_1, q_2, q_3) = (4, 4, 4)$. Los mejores resultados según esta métrica se corresponden, como era de esperar, con la dimensión intrínseca generada, aunque en el caso de aproximar con subespacios de dimensiones ligeramente superiores se obtienen también resultados similares. En el caso de subespacios $(q_1, q_2, q_3) = (4, 4, 4)$ se aprecia un incremento de la variabilidad, el cual es posible que se deba porque es necesario realizar más iteraciones al ser un modelo a ajustar más complejo.

En conclusión, queda claro que si se conocen las dimensiones intrínsecas de antemano se obtiene la mejor solución. No obstante, si no son conocidas, es mejor utilizar el procedimiento con subespacios de dimensión más alta pero próximos a la real. Como solución a la elección de las dimensiones intrínsecas se propone realizar un "scree plot". Para ello se puede usar el método RLG con la función desarrollada "plot.rlg" (Sección 3.8.3) y seleccionar el número de dimensiones con las que se explica mayor variabilidad.

4.1.3. Contaminación y refinamiento



(a) Función objetivo **(b)** SSE **(c)** Tiempo de ejecución

Figura 4.4: Diagramas de caja frente a los valores de α. Se han eliminado los valores más atípicos para una mejor visualización

En todas las simulaciones realizadas el número de celdas reales contaminadas ha sido del 5 %. Al probar varios valores del tamaño de recorte se puede comprobar que si indicamos al método un número inferior de celdas a quitar, el resultado es bastante pobre en términos de la función de pérdida y del SSE. En cambio, si fijamos α en el mismo valor a la contaminación real ε_0 el rendimiento es el mejor de todos los probados. En cambio, cuando aumentamos el tamaño de recorte α se incrementa algo el SSE pero este incremento es muy bajo en comparación con recortar de menos. Por tanto, es preferible recortar de más que de menos y mejor cuanto más nos ajustemos a la contaminación real. Esto seguramente se deba al paso de refinamiento, que recupera una gran cantidad de celdas que no debieron ser recortadas.

Sobre los tiempos de ejecución, no hay diferencias significativas en usar un tamaño de recorte u otro, lo cual es previsible dado que las etapas que recorre el algoritmo son las mismas (con la misma complejidad).

4.2. Datos funcionales

Dada la naturaleza específica de los datos funcionales, se han propuesto las siguientes simulaciones:

1. *Método de inicialización*: Al aplicar una representación con B-splines, el algoritmo no es demasiado dependiente de la dimensión p (número de evaluaciones de la función) y sí lo es del número de nodos `knots` utilizados en la representación B-splines. Por ello, los datos se han generado en una dimensión de $p = 50$ y después se han probado los valores de `knots` $\in \{4, 10, 20\}$, con $n = 200$ curvas y con $G = 2$ clusters (100 curvas por grupo) distintos de dimensión intrínseca $q_1 = q_2 = 2$ (para todos la misma).

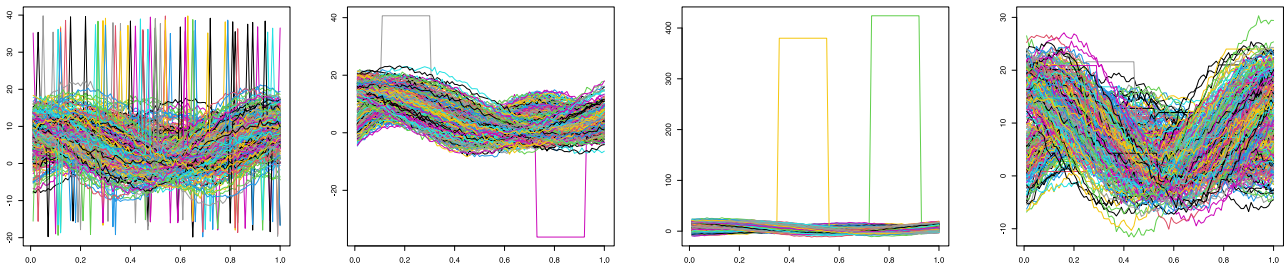
2. *Tipos de contaminación*: Se han generado varios esquemas de contaminación diferentes, los cuales son:

   a) Se seleccionan 3 % de celdas aleatorias para perturbar; cada una de las elegidas se reemplazan con un valor negativo o positivo (fuera del rango de los datos) con probabilidad 0.5 (Figura 4.5a).

   b) La contaminación se hace seleccionando el 15 % de observaciones; para cada una de ellas se perturban un 20 % de coordenadas consecutivas con valores atípicos pero no extremos (Figura 4.5b). La cantidad total de celdas a contaminar es del 3 %.

   c) Similar al segundo caso solo que a las celdas modificadas se les da un valor extremo y positivo (unas 10 veces mayor). En este caso, también la cantidad de celdas a contaminar es del 3 % (Figura 4.5c).

   d) Similar al segundo y tercer caso excepto que los valores nuevos están dentro del rango de los datos, porque lo que se hace es dejar la función fija sin actualizar for una fracción del 20 % de observaciones consecutivas y generando una fracción total del 3 % de celdas contaminantes (Figura 4.5d).



(a) **(b)** **(c)** **(d)**

Figura 4.5: Distintos episodios de contaminación para datos funcionales

Al igual que se hizo en el caso de los datos no funcionales, se ha creado una función llamada `genFUN` extendiendo la explicada en la Sección 3.6.1.4 pero con algunos cambios en valores que antes estaban fijos, con el fin de ganar aleatoriedad y que no se generen siempre las mismas funciones. Las funciones de medias y modos de variación son ahora de la forma:

$$f_1(x) = a_1 + a_2 \sin(a_3 \pi x) e^{-a_4 x} + a_1 \sin(\pi \frac{x}{3}) + a_4 \cos(a_4 \frac{\pi}{2}) \text{ y}$$
$$f_2(x) = a_2 + a_2 \cos(4\pi x).$$

$$\rho_{1,1}(x) = \sqrt{a_4} \cos(a_4 \pi x) \qquad \rho_{1,2}(x) = \sqrt{a_4} \sin(a_4 \pi x)$$
$$\rho_{2,1}(x) = \sqrt{a_4} \sin(a_4 \pi x) \qquad \rho_{2,2}(x) = \sqrt{a_4} \cos(a_4 \pi x)$$

$$f_1(x) = a_1 + a_2 \sin(a_3 \pi x) e^{-a_4 x} + a_1 \sin(\pi \frac{x}{3}) + a_4 \cos(a_4 \frac{\pi}{2}) \text{ y}$$
$$f_2(x) = a_2 + a_2 \cos(4\pi x).$$

Así, finalmente las funciones en el primer grupo vendrán dadas por realizaciones de funciones de la forma:

$$f_1(x) + r_1 \rho_{1,1}(x) + r_2 \rho_{1,2}(x),$$

y de la forma

$$f_2(x) + r_3 \rho_{2,1}(x) + r_2 \rho_{2,2}(x),$$

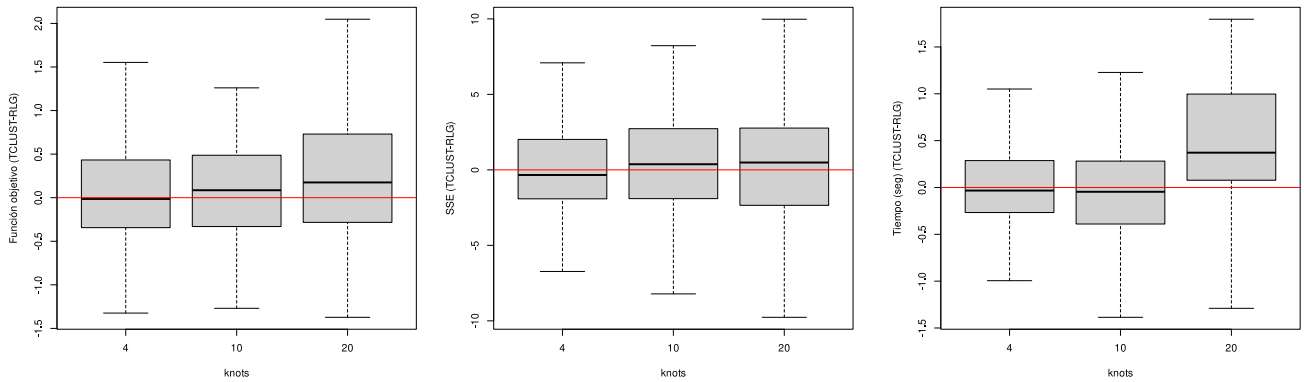
donde los términos a_1, a_2, a_3, a_4 son realizaciones aleatorias independientes de distribuciones $U(0,5)$, $U(5,10)$, $U(0,4)$, $U(0,2)$. Recordemos además que r_1 eran realizaciones aleatorias independientes de la distribución $N(0,3)$, r_2 de la distribución $N(0,2)$ y r_3 de la distribución $N(0,4)$. También se han sumado unos errores aleatorios con normales independientes de menor variabilidad en los puntos donde se realiza la evaluación de las curvas.

4.2.1. Método de inicialización

De igual forma que con los datos no funcionales, se ha calculado la diferencia de inicializar con TCLUST menos RLG al tener datos pareados, por aplicarse sobre el mismo conjunto de datos funcionales en cada ocasión $B = 100$.

En la Figura 4.6 se muestran las diferencias observadas para cada una de las métricas. Para un número de nodos bajo, parece no haber diferencias significativas entre ambos métodos y parece que la función objetivo en `knots` $= 4$ pueda ser algo mejor en RLG pero para el SSE parece ser inferior el de TCLUST. No obstante, las diferencias se empiezan a apreciar más a favor de RLG cuando incrementamos el número de nodos hasta `knots` $= 20$, tanto en la función objetivo como en el SSE. Para dicho número de nodos también parece que TCLUST tarda más en ejecutarse, aunque con una diferencia menos marcada que en el caso no funcional.

De nuevo y tal como ocurría en el caso no funcional parece que RLG arroja mejores resultados, pero con datos funcionales esta diferencia resulta menor, al menos, si el numero de `knots` internos no es muy grande.



(a) Función objetivo **(b)** SSE **(c)** Tiempo de ejecución

Figura 4.6: Diagramas de caja de la diferencia entre TCLUST y RLG frente a los nodos de B-splines

4.2.2. Contaminación y refinamiento

De la misma forma que ocurría al analizar la contaminación y el refinamiento en datos no funcionales, algo similar aparece en el caso de los datos funcionales según la función objetivo (Figura 4.7). En todos los casos al recortar más celdas se reduce la pérdida, aunque con diferencias entre ellos. En los tres primeros esquemas de contaminación se reduce más bruscamente hasta el penúltimo valor de α, no habiendo demasiada diferencia entre escoger el 5 % o el 20 %. En cambio, en el último caso se reduce de manera parecida con cada valor de α, progresivamente.

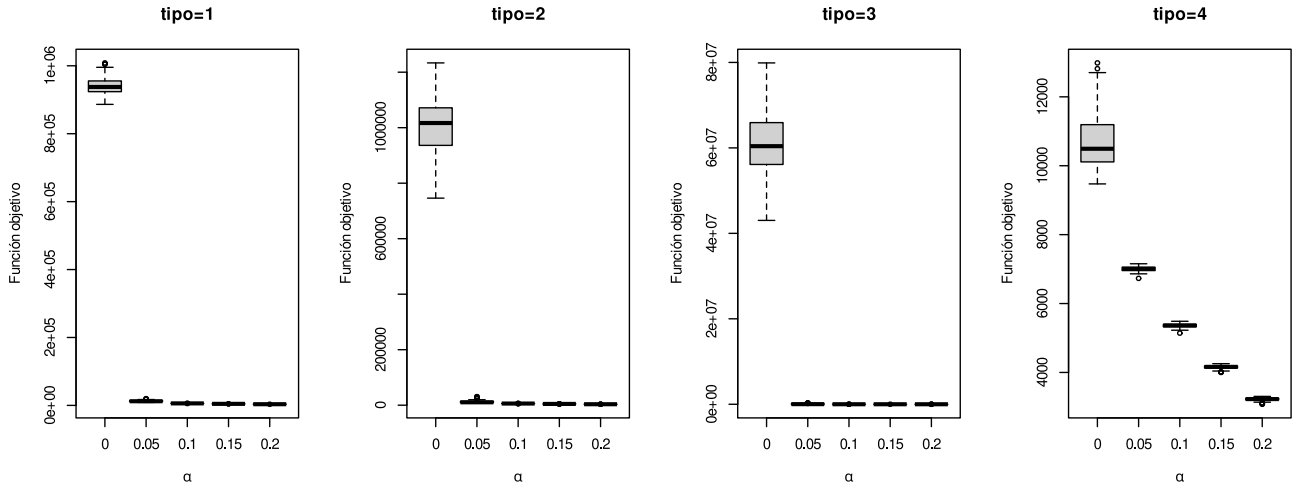


Figura 4.7: Diagramas de caja de la función objetivo frente al tamaño de recorte para distintos esquemas de contaminación

Al comparar los distintos valores de recorte para el SSE se obtienen también resultados bastante similares en todos los casos en la Figura 4.8. El SSE parece que siempre disminuye al aumentar α en los tres primeros esquemas de simulación, aunque se use un tamaño de recorte superior

al de la contaminación. El último esquema de simulación es en el único que no se cumple esto, incrementándose el SSE cuando se utilizan valores de recorte superiores al necesario ε_0, siendo en algún caso incluso peor que no recortar. Si recordamos cómo se había generado la contaminación, esta última situación correspondía a cambiar valores por una constante pero dentro del rango de los datos. Lo que parece estar ocurriendo es que esas observaciones se han recortado pero no han podido ser recuperadas por la etapa de refinamiento, al estar muy cerca de valores no atípicos.

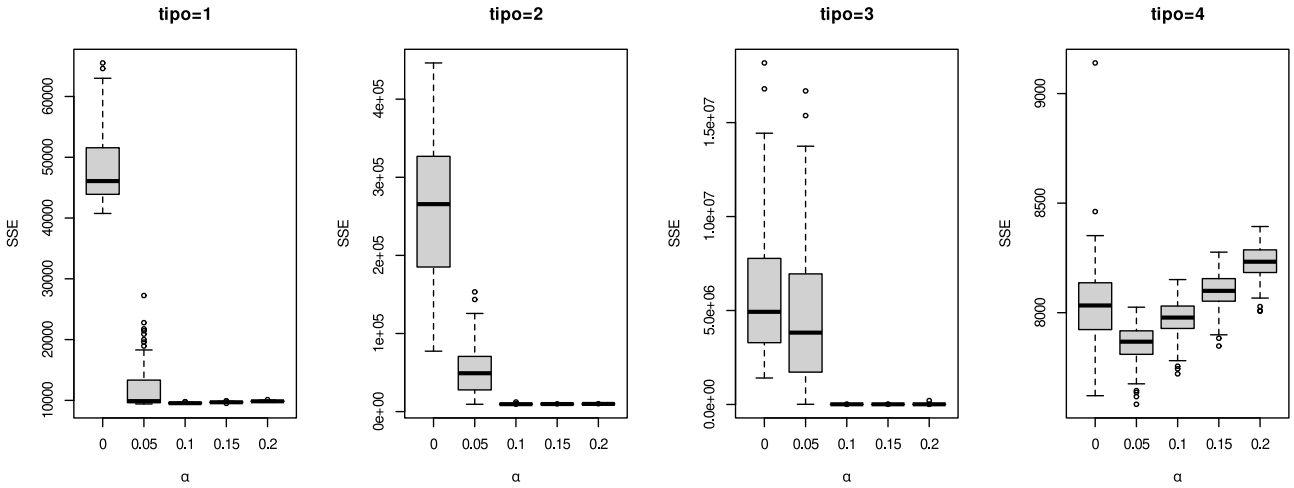


Figura 4.8: Diagramas de caja del SSE frente al tamaño de recorte para distintos esquemas de contaminación

Con respecto a los errores de clasificación en la Figura 4.9, en todos los casos se mejora al incrementar el tamaño de recorte. Este comportamiento es el esperado dado que cuantas más celdas se eliminen más probabilidad hay de que las que sobrevivan el recorte no sean atípicas y contribuyan solo a una mejor clasificación. Cabe destacar el mal comportamiento que surge si no se recorta nada $\alpha = 0$ en datos contaminados, llegando en ocasiones a tasas de mala clasificación cercanas al 50 % (tercer esquema de simulación).

Por último, los tiempos de ejecución son bastante parecidos en todos ellos (Figura 4.10), incrementándose cuando se recortan más celdas. En todos ellos hay varios puntos atípicos, correspondientes a conjuntos de datos que han tardado más de lo habitual (comparando con los demás).

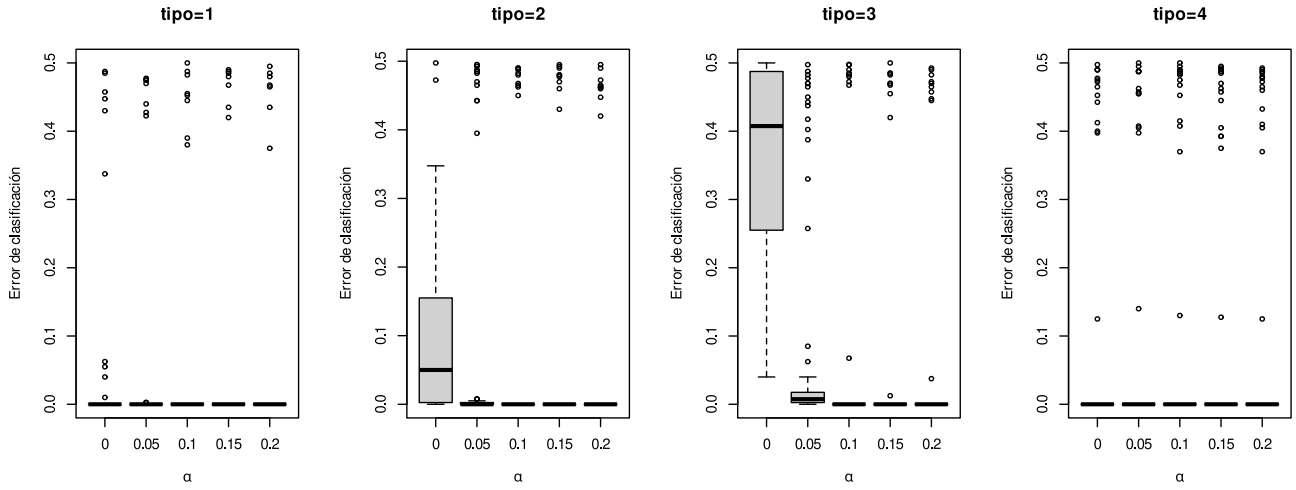


Figura 4.9: Diagramas de caja del error de clasificación frente al tamaño de recorte para distintos esquemas de contaminación

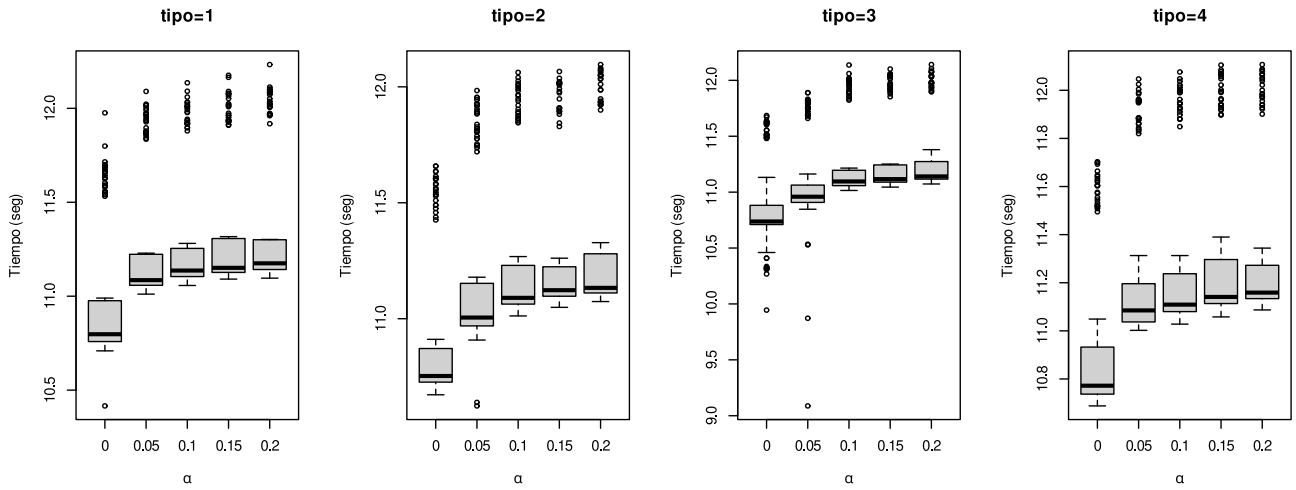


Figura 4.10: Diagramas de caja del tiempo frente al tamaño de recorte para distintos esquemas de contaminación

Capítulo 5

Aplicación sobre datos reales

A continuación se expondrán dos situaciones basadas en datos reales en las que se usará el algoritmo propuesto. El objetivo fundamental es comprobar que las modificaciones realizadas funcionan correctamente, por lo que se van a utilizar los mismos dos conjuntos de datos en [1].

5.1. Mortalidad en Francia

5.1.1. Descripción del conjunto de datos

En el primer conjunto de datos se analizará la tasa de mortalidad en Francia entre los años 1816 y 2006 para edades de 0 a 99 años en hombres. Los datos han sido extraídos de la "Human Mortality Database" y están disponibles en el paquete `demography` de R [38].

Estamos ante un ejemplo de datos funcionales, donde cada curva corresponde a la evolución de la tasa de mortalidad en cada año de la historia de Francia en el periodo de 1816 y 2006, por lo que $n = 191$, y en cada grupo de edad entre 0 y 99 años de edad, por lo que $p = 100$. La variable que se ha medido por grupo de edad y año es la tasa de mortalidad, definido como la proporción de personas que fallecen respecto a la población total, agrupados por edad. Es muy común aplicar una transformación logarítmica para mejorar su visualización, al ser valores generalmente bajos y asimétricos.

En la Figura 5.1 se muestra una representación gráfica de los datos. Cada línea corresponde con una curva, es decir, un año. En el eje x se representa cada grupo de edad y en el eje y el valor del logaritmo de la mortalidad. Se ha asignado un color distinto a cada curva utilizando el año de la observación, de tal forma que colores más claros de la curva se corresponden con años menos recientes. A simple vista se pueden apreciar dos clusters. Los años más antiguos tienen una tasa de mortalidad claramente superiores a los tasas de mortalidad de los años más recientes. La diferencia entre los grupos se encuentra en el final de la Segunda Guerra Mundial, después de la cual hubo grandes mejoras en la calidad de vida y por ello es un punto de inflexión que se ve reflejado en las tasas de mortalidad.

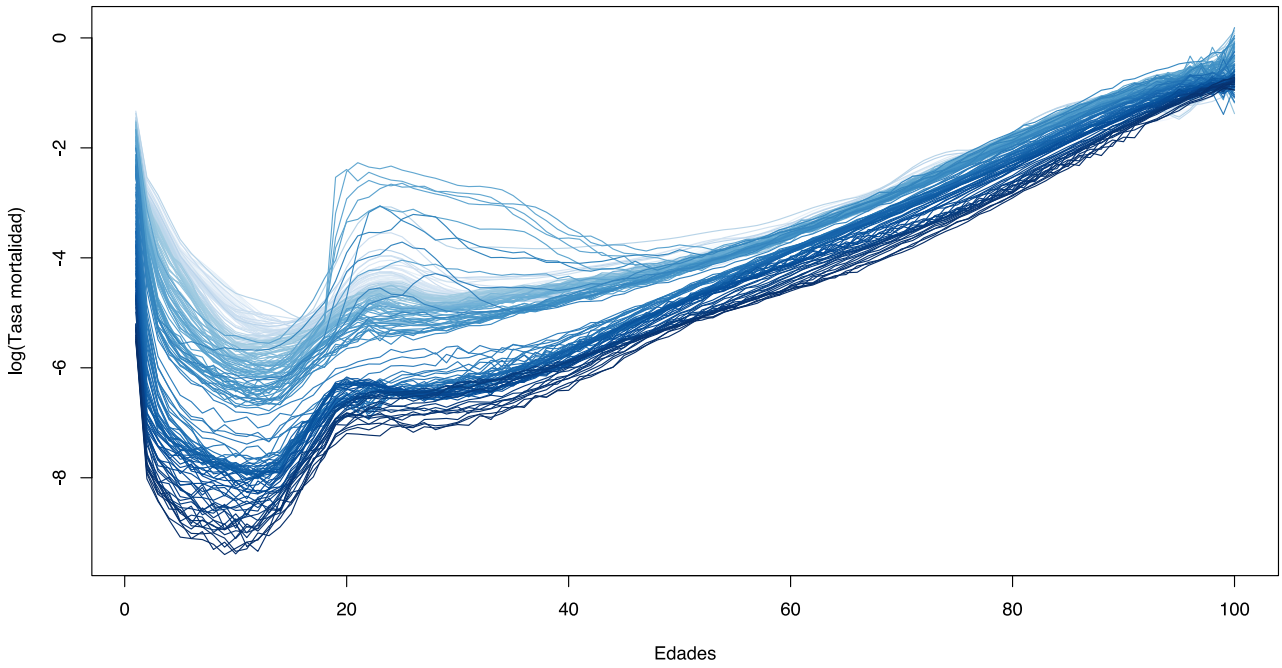


Figura 5.1: Tasas de mortalidad (escala logarítmica) en varones desde 1816 hasta 2006 en Francia

5.1.2. Resultados

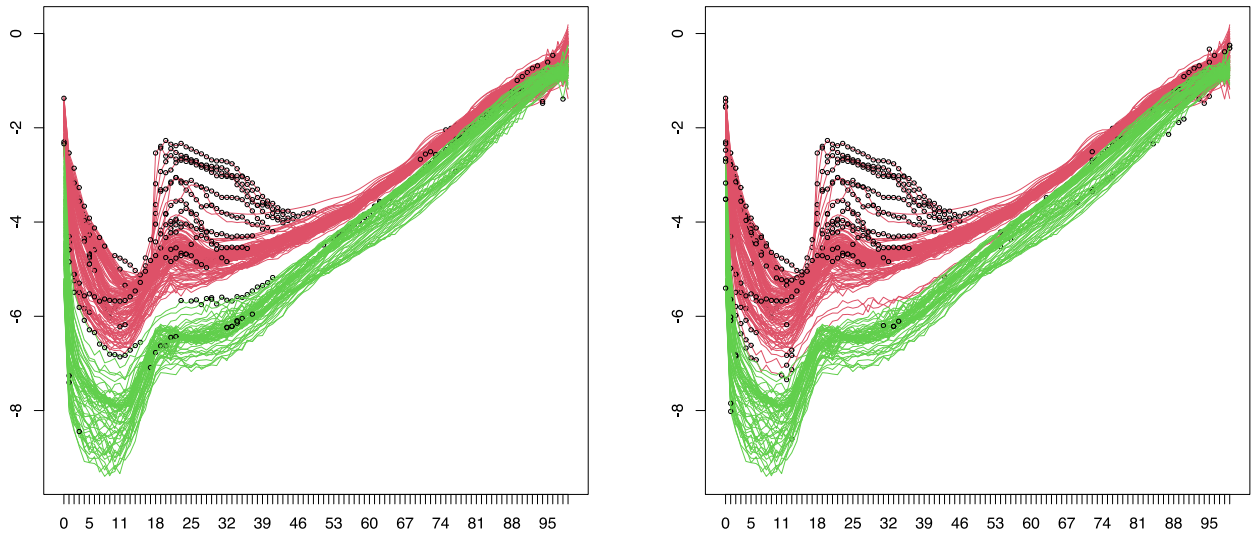
Se han probado ambos métodos de inicialización, tanto TCLUST como RLG. Para TCLUST se han usado los mismos parámetros que los propuestos en [1] y se han realizado pruebas con distintos parámetros para RLG. Para ambos métodos de inicialización, se toman dimensiones intrínsecas $q_1 = q_2 = 2$, como en [1].

Si comparamos la asignación a grupos, ambos encuentran correctamente la diferencia entre curvas correspondientes a años antes del final de la Segunda Guerra Mundial y los de después, con la salvedad de que el resultado de inicializar con TCLUST no incluye los años que están en el límite en el grupo superior pero RLG sí lo hace. Estas observaciones corresponden con un período entreguerras, por lo que realmente no queda claro el límite y podrían clasificarse en cualquiera de los dos grupos.

Para comprobar qué ajuste es mejor podemos ver el valor que ha alcanzado la función de pérdida para cada método de inicialización (Tabla 5.1). Como se intuía, ambos son muy parecidos, aunque en este caso quizás un poco mejor TCLUST.

TCLUST	RLG
31.14361	31.50215

Tabla 5.1: Función objetivo según el método de inicialización



(a) TCLUST **(b)** RLG

Figura 5.2: Asignación de cada curva a grupos

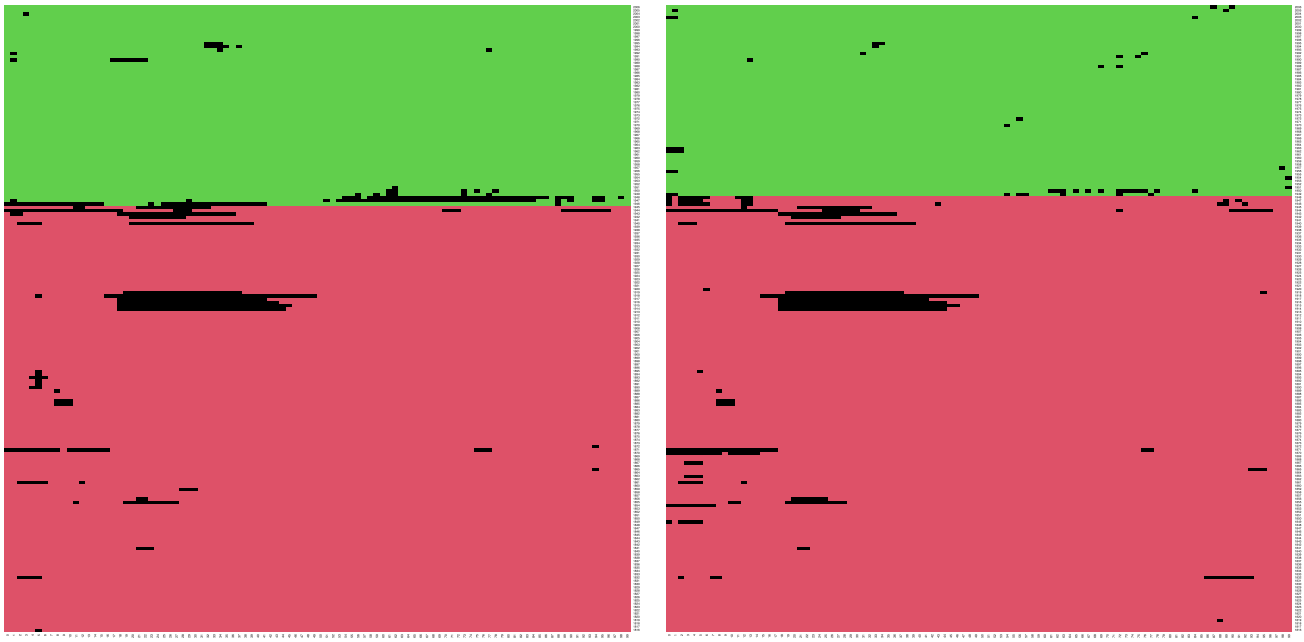
Se puede visualizar de manera cómoda las asignaciones a cada uno de los grupos, con el gráfico mostrado en la Figura 5.3, en el que aparecen marcadas en negro las celdas recortadas, correspondientes a celdas con residuos altos (después de la etapa de refinamiento). Se pueden comprobar si estos residuos corresponden a atípicos por exceso o por defecto (la mortalidad obsevada era mayor que la prevista en ese grupo) a través de la Figura 5.4.

Ambos métodos recortan un número similar de celdas, aunque TCLUST elimina alguna más. Entre los recortes comunes, se puede percibir una región entre 1917 y 1920 aproximadamente que afecta a muchos grupos de edad, desde los 14 años hasta los 50 aproximadamente. Si lo relacionamos con la Historia, en esos años empezó la Primera Guerra Mundial y se expandió la gripe española, causas muy posibles de este incremento de mortalidad. Otra región con crecimiento del número de celdas recortadas es entre los años 1939 y 1944, correspondientes al período de la Segunda Guerra Mundial, por ello la mayor parte de las celdas recortadas se encuentran entre la población joven, que sería la que fallecería más notablemente en los frentes de guerra.

Otro resumen gráfico interesante de analizar es el de los "scores" y los "loadings" (Figura 5.5), cuya interpretación se parece a la del ACP. Si nos centramos en uno de ellos, por ejemplo en el de TCLUST (Figuras 5.5a y 5.5b), podemos observar que para el primer grupo y su primera dimensión tienen valores más grandes (negativos) los grupos de edad jóvenes, que van disminuyendo hasta los 18 años y después vuelven a disminuir hasta los 40 años aproximadamente. Relacionando este dato con el gráfico de los "scores", para el cluster 1, valores más negativos sobre el eje x indican una mortalidad mayor para los grupos de edad jóvenes (dado que ambos valores son negativos en el producto a realizar, dicho producto sería positivo). Hay dos años que sobresalen siguiendo estos criterios, 1871 y 1944, correspondientes a la pérdida de Francia en las guerra franco-prusiana y al

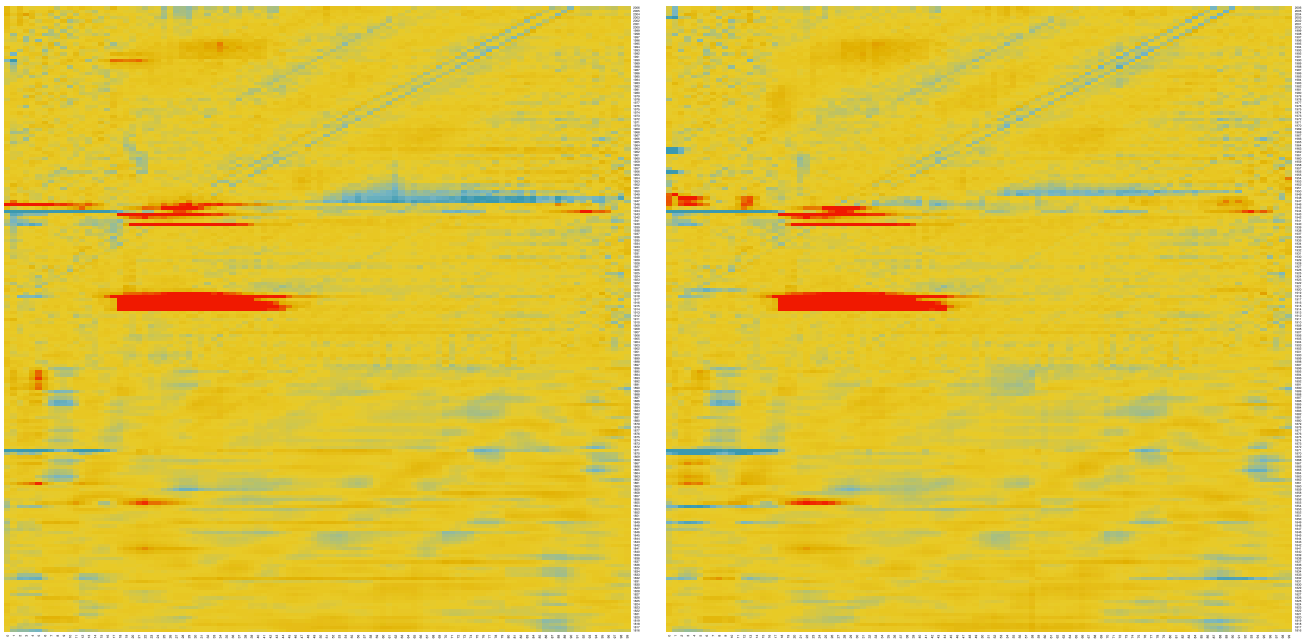
final de la Segunda Guerra Mundial en Francia, respectivamente. Si nos fijamos en valores menos atípicos, podemos comprobar que se encuentran en la parte negativa (y por tanto mayor mortalidad en edades tempranas) los años anteriores a la Primera Guerra Mundial, por lo que seguramente sea debido a una alta mortalidad consecuencia de partos en peores condiciones sanitarias.

De manera análoga, se puede interpretar el segundo de los grupos para la primera dimensión. En el caso de la segunda dimensión (para el cluster 2) se muestran en el gráfico de los "loadings" valores más grandes también para la gente joven, pero en este caso, en mayor medida la franja de los 16-25 años. Al interpretar los scores con dicha información, valores más grandes del eje y corresponden a mayor mortalidad en esa franja de edad. Estos años son los años 70 y 80, por lo que una posible explicación al incremento de la mortalidad juvenil podría ser un alto consumo de heroína por parte de algunos jóvenes, y la mortalidad extra asociada a ese consumo, en esos años en Francia.



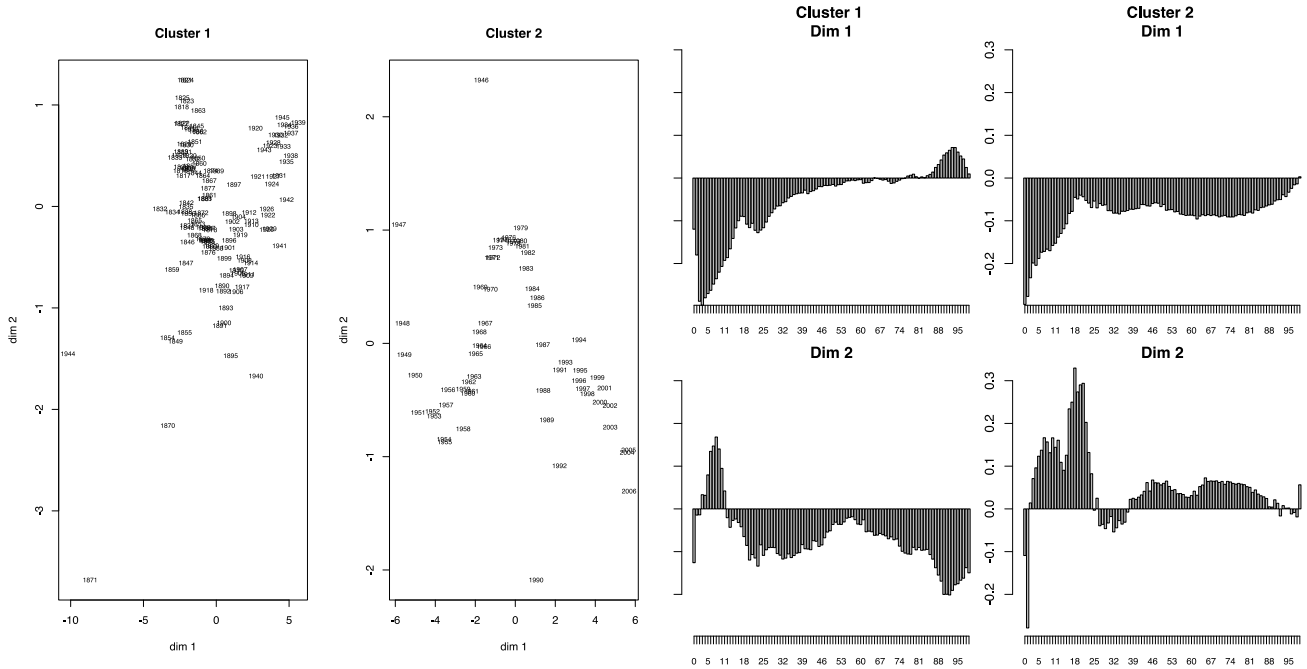
(a) TCLUST **(b)** RLG

Figura 5.3: Asignación a grupos



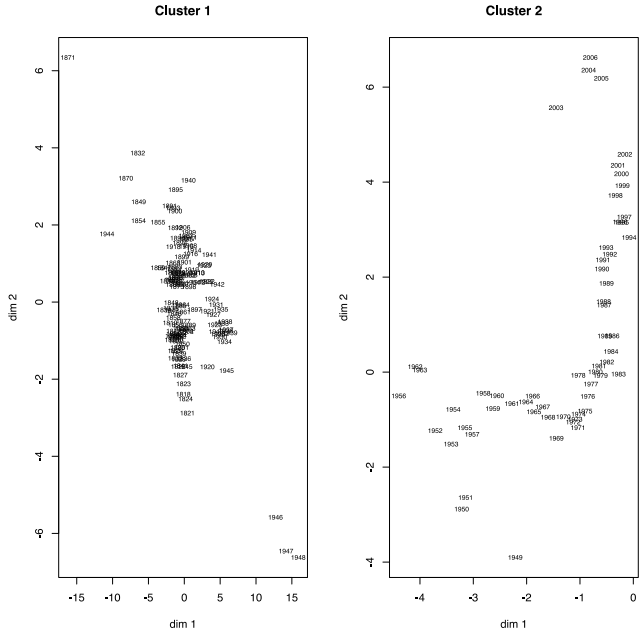
(a) TCLUST **(b)** RLG

Figura 5.4: Residuos por celda

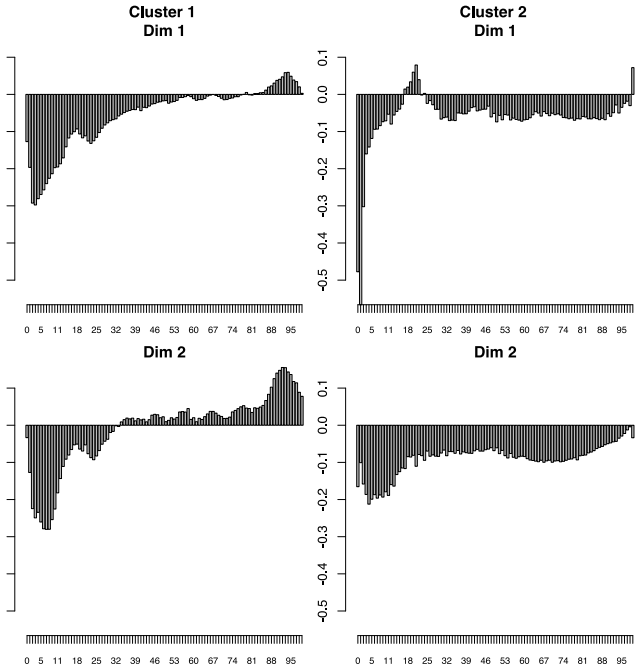


(a) *Scores* TCLUST

(b) *Loadings* TCLUST



(c) *Scores* RLG



(d) *Loadings* RLG

Figura 5.5: Gráficos de *scores* y *loadings*

5.2. Temperaturas medias en España

5.2.1. Descripción del conjunto de datos

El conjunto de datos que se va a utilizar contiene información sobre la temperatura diaria registrada en $n = 83$ estaciones meteorológicas de España. Los datos han sido extraídos del portal de Datos Abiertos de la Agencia Estatal de Meteorología (AEMET) y contienen información sobre los años 2007, 2008, 2009, lo que supone un total de $p = 1096$. Estamos de nuevo ante un ejemplo de datos funcionales, donde cada curva corresponde con cada una de las $n = 83$ estaciones.

A estos datos se les ha aplicado las siguientes contaminaciones artificiales:

- Se han seleccionado 100 observaciones consecutivas durante el otoño de 2007 en Huelva y dos grupos de 50 días (también consecutivos) de Oviedo. A estos días se les ha reemplazado su valor por 0^{o}C, de esta forma se puede simular que la estación no funcionaba correctamente.

- Aleatoriamente se han seleccionado el 1 % de días (no consecutivos) de estaciones al azar a los que a la mitad se les ha sustituido por realizaciones aleatorias e independientes de una $U(40, 45)$ y a la otra mitad por realizaciones de una $U(-2, 0)$. Este comportamiento simula también fallos en la estación, registrando valores extremos, tanto muy altos como muy bajos. Esta contaminación aleatoria difiere de la contaminación considerada en [1] y hace que el problema sea de una mayor complejidad.

En la Figura 5.6 se muestran las temperaturas medias de cada estación a lo largo de dichos años con la contaminación descrita. Se percibe un componente estacional, que se repite cada año. En los meses de invierno las temperaturas son más bajas, subiendo progresivamente hasta verano y de nuevo disminuyendo en el siguiente invierno. Esta componente es más brusca en ciertas curvas que en otras, pudiendo diferenciar un par de grupos a simple vista.

Se han tomado estaciones de todos los puntos de la geografía española, incluyendo las Islas Baleares, Melilla y las Islas Canarias. En la Figura 5.7 se muestra un mapa con todas las estaciones seleccionadas junto con la imagen LiDAR (altitud), realizado gracias al paquete `mapSpain` de R [39]. De esta forma cabe pensar que dependiendo de la situación geográfica y de la altura de la estación las temperaturas se comportarán de forma diferente, ya que son dos factores que afectan en gran medida a las temperaturas.

5.2.2. Resultados

Se han probado los dos métodos de inicialización, TCLUST y RLG, cada uno de ellos con $q_g = (2, 2, 2, 2)$. Atendiendo al agrupamiento de las curvas (Figura 5.8) y al mapa representando dicha agrupación (Figura 5.9), parece que la solución dada por TCLUST es más interpretable: encuentra los grupos de forma razonable según la geografía de las estaciones. En cambio RLG agrupa las del centro de la península con las de Canarias, lo cual no parece demasiado lógico: Valladolid dista mucho de tener un clima similar a La Palma, por ejemplo.

Si analizamos en detalle los grupos dados por TCLUST, nos encontramos que:

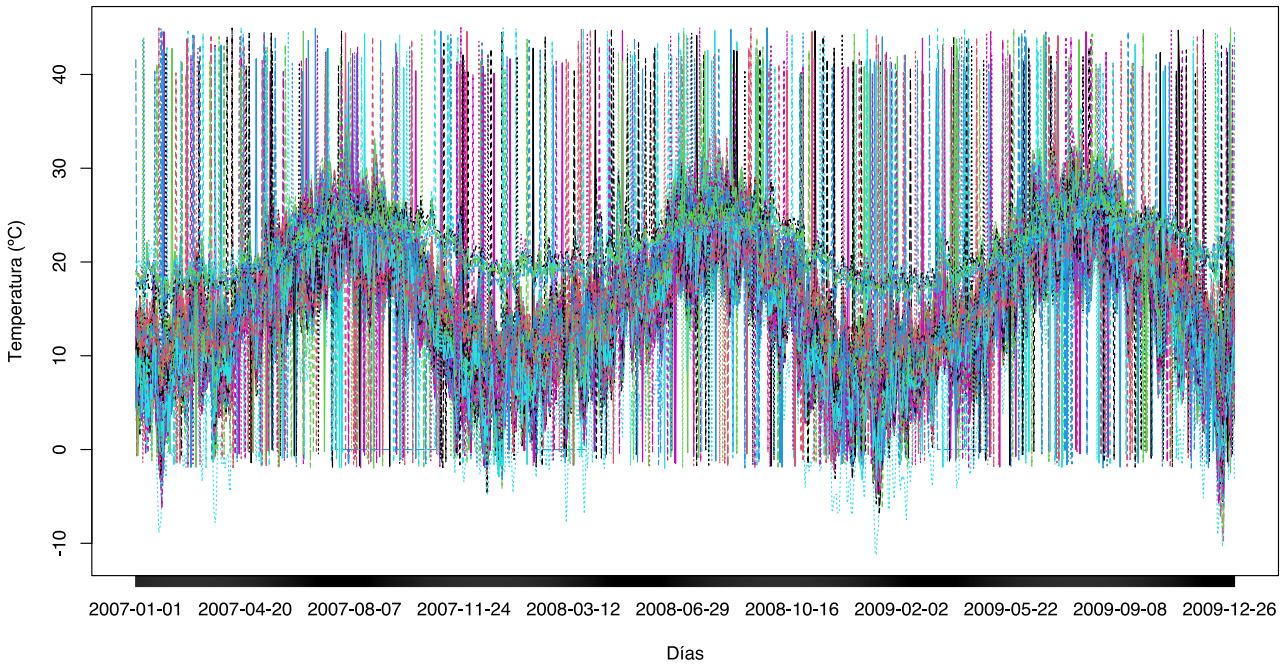


Figura 5.6: Serie temporal con las temperaturas desde el 1 de enero de 2007 hasta el 31 de diciembre de 2009 en 83 estaciones meteorológicas de España con contaminación artificial añadida

1. *Rojo*. Mesetas norte y sur junto con Granada y parte del valle del Ebro. Son zonas correspondientes a un clima mediterráneo continentalizado, extremo tanto en los meses de verano como en invierno.

2. *Verde*. Mediterráneo, Andalucía (excepto Granada y Rota), Islas Baleares, Extremadura y parte del valle del Ebro. Temperaturas altas en verano pero más suaves en invierno, sin heladas.

3. *Azul oscuro*. Litoral vasco, cantábrico y gallego. Temperaturas más suaves durante todo el año debido a la influencia del mar.

4. *Azul claro*. Islas Canarias excepto Tenerife Norte. Clima tropical.

Para observar el comportamiento sobre la contaminación, nos podemos fijar en las figuras 5.11 y 5.12. En ambas aparece correctamente identificado el primer tipo de contaminación de observaciones consecutivas para Huelva y Oviedo. Además, los residuos están marcados como azul, por lo que son menores de lo que cabría esperar (dado que se ha reemplazado las observaciones verdaderas positivas con valores de 0^{o}C).

Con respecto a la contaminación dispersa del 1 %, para visualizar si se detecta correctamente, se puede ver la Figura 5.13 en la que se muestran tres gráficos. El primer gráfico muestra la contaminación real, el segundo muestra las celdas recortadas por cada procedimiento y el tercero

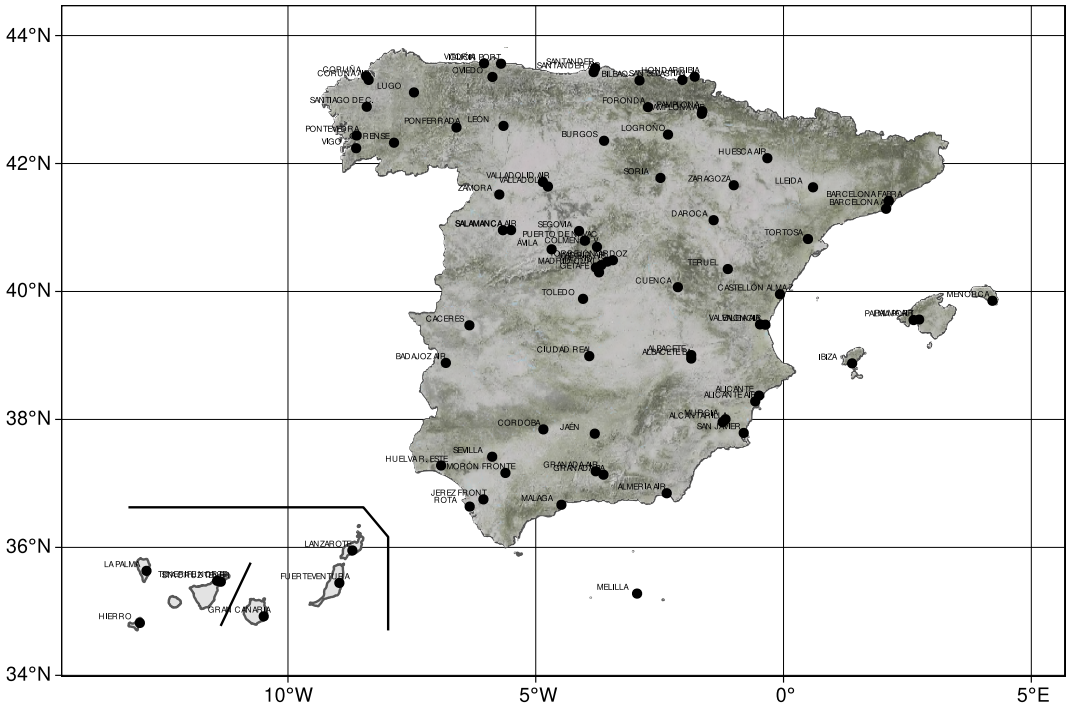


Figura 5.7: Ubicaciones de las estaciones seleccionadas

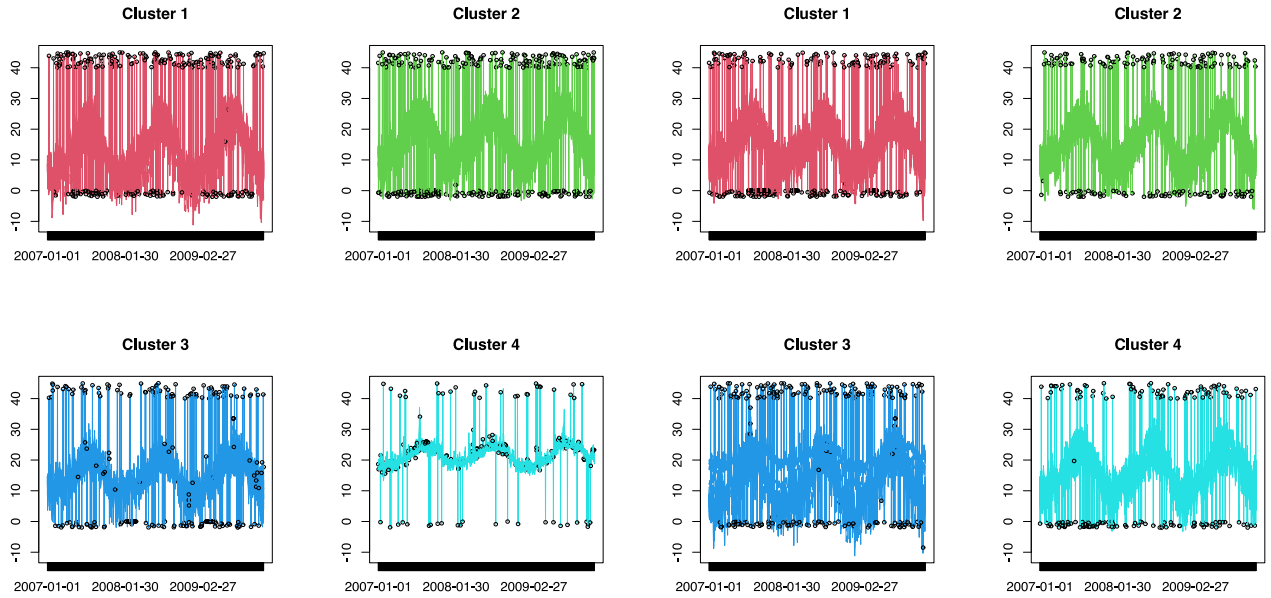
nos muestra las celdas en las que nos hemos equivocado porque son celdas que no ha recortado pero estaban contaminadas o ha recortado y no estaban contaminadas. A la vista de estos gráficos, parece que RLG detecta mejor las celdas contaminadas y que TCLUST no lo hace del todo bien, sobre todo en la asignación a cluster de las Islas Canarias. Seguramente, el hecho de que falle más en este grupo de curvas sea debido a que no hay demasiadas observaciones, tan solo 6.

Para comprobar la calidad del ajuste, en la Tabla 5.2 se muestra el valor de la función de objetivo al que ha llegado el algoritmo usando un método de inicialización o el otro. RLG minimiza más la función objetivo que TCLUST, por lo que, en principio, sería preferible. Parece contradictorio que, aún teniendo mejor ajuste y una detección de celdas atípicas también mejor, sin embargo, a la hora de interpretar queden mejores resultados con TCLUST con nuestro conocimiento del problema. Indagando en la forma de asignar observaciones a grupos de ambos algoritmos, este comportamiento seguramente se deba a que RLG, al asignar ciertas observaciones al subespacio aproximante, porque la distancia a ese subespacio sea menor, no parece tener en cuenta la estructura del cluster, cuestión que TCLUST sí hace.

TCLUST	RLG
56292.87	42600.94

Tabla 5.2: Función objetivo según el método de inicialización

Para ilustrar esta situación, en la Figura 5.10 se han representado dos grupos distintos de observaciones provenientes de dos normales multidimensionales en $\mathbb{R}^2$. De forma natural, se aprecian los dos clusters y de esa misma forma los encuentra TCLUST. Sin embargo, RLG al aproximar



(a) TCLUST **(b)** RLG

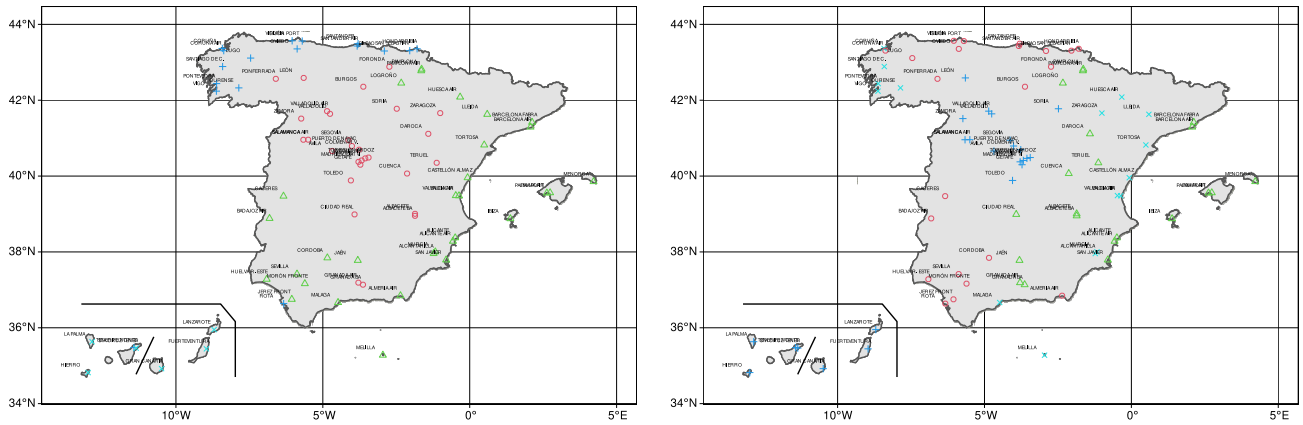
Figura 5.8: Asignación de cada curva a grupos

subespacios, en este caso alrededor de rectas, no consigue encontrar la estructura de grupos correctamente y parte de las observaciones de un grupo las asigna al otro dado que aproxima a la recta más cercana.

Por último, si nos fijamos en el gráfico de "scores" y "loadings" (Figura 5.14) podemos extraer información de los grupos formados. En los "scores" (Figura 5.14a) se pueden ver las estaciones meteorológicas más atípicas dentro de cada cluster. En el primer cluster, por ejemplo, se muestran en las esquinas el "Puerto de Navacerrada" y "Zaragoza", donde ambas estaciones tienen un clima algo diferente al de la meseta. El "Puerto de Navacerrada" es una zona montañosa y el "Zaragoza" podría agruparse cercano al cluster del Mediterráneo. Además, las estaciones que se parecen entre ellas están cercanas en ese gráfico de "scores", como Santander y Gijón; Vitoria y Oviedo (de interior ambas); Huesca, Pamplona y Logroño, etc.

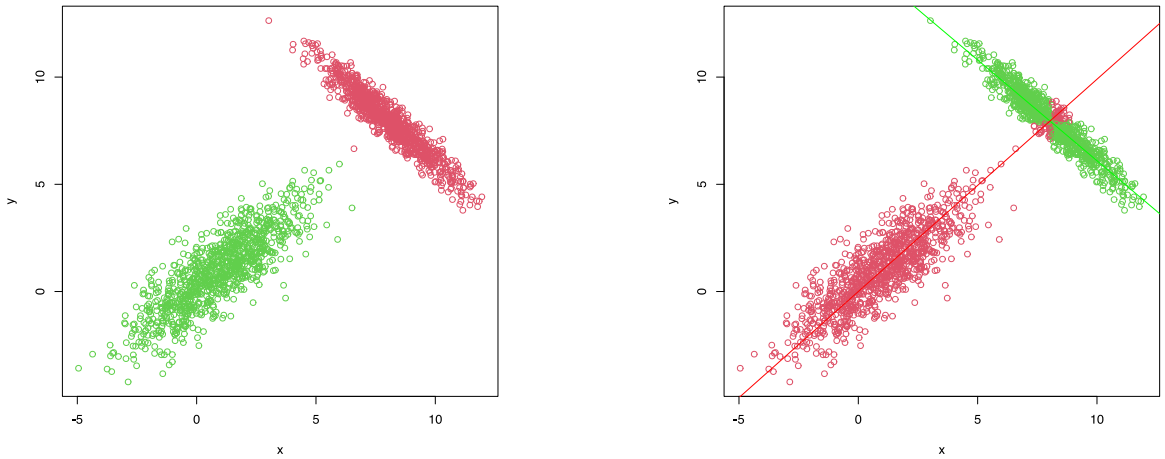
Si además usamos la información que nos proporcionan los "loadings" (Figura 5.14b), podemos ver que en la primera dimensión del primer cluster todos son valores positivos. Por tanto, observaciones que se encuentren más a la derecha (valores positivos) del eje x de los "scores" corresponden a estaciones más cálidas. Estas coinciden en gran medida con la submeseta sur y Granada. Por otro lado, las coordenadas del lado negativo del mismo eje corresponden a temperaturas más bajas (comparadas con las del mismo cluster) como pueden ser Valladolid, León y Salamanca.

Por último, en la Figura 5.15, se han representado dos gráficos en los que se muestran en negro las observaciones reales (contaminadas) y en rojo la predicción realizada por el procedimiento. La Figura 5.15a corresponde con el resultado del método utilizando un tamaño de recorte $\alpha = 0.1$, en el que se aprecia que se predicen bastante bien los posibles valores de la temperatura, sin ser negativamente influidos por la contaminación y teniendo en cuenta la información del cluster al



(a) TCLUST **(b)** RLG

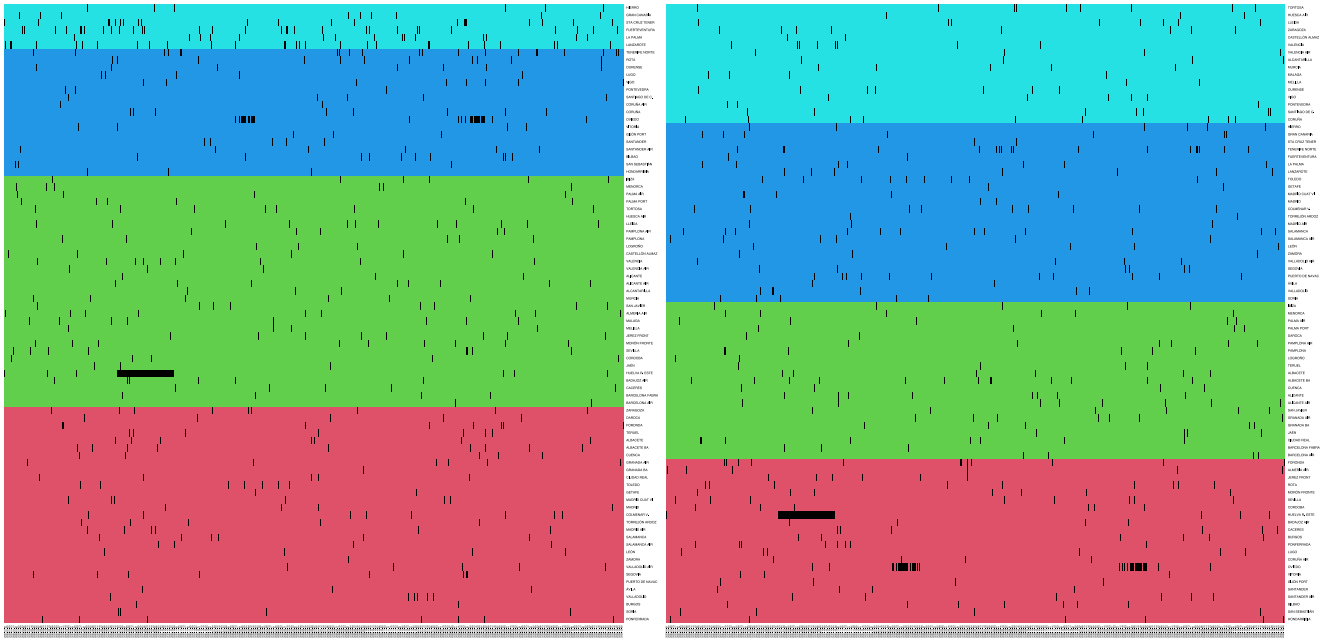
Figura 5.9: Mapa con la asignación a clusters de cada estación dependiendo del método de inicialización



(a) TCLUST **(b)** RLG

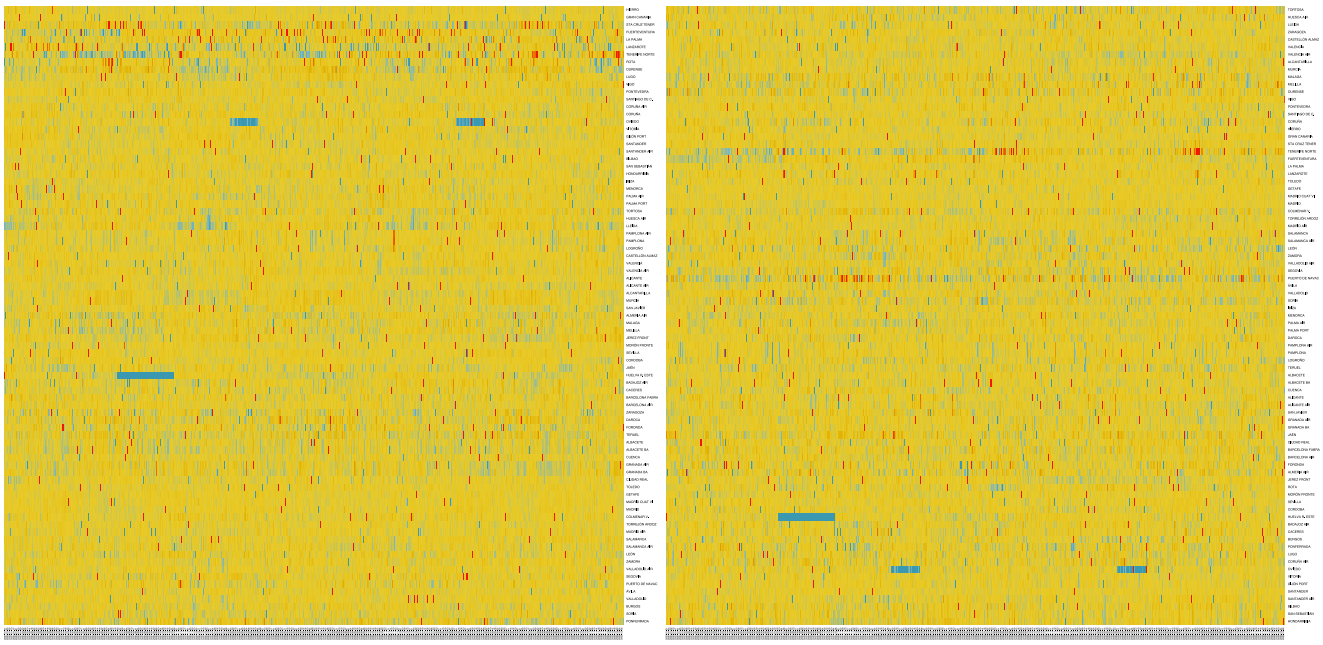
Figura 5.10: Ejemplo ilustrativo de la diferencia al asignar observaciones a clusters con TCLUST o RLG

que pertencen y la información sobe la observación proporcionada por las celdas no contaminadas en dicha observación. Por otro lado, en la Figura 5.15b, se ha ejecutado el método sin aplicar ningún tipo de recorte $\alpha = 0$ y, claramente, se nota el efecto negativo de las celdas contaminantes, especialmente en Huelva.



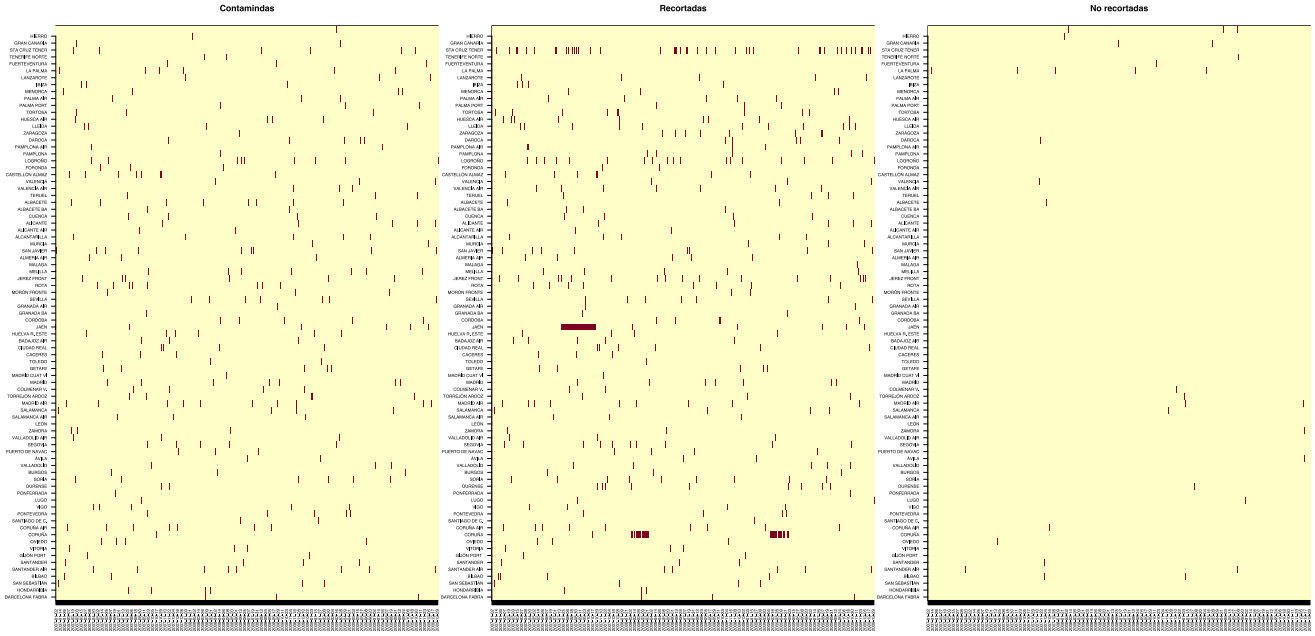
(a) TCLUST **(b)** RLG

Figura 5.11: Asignación a grupos

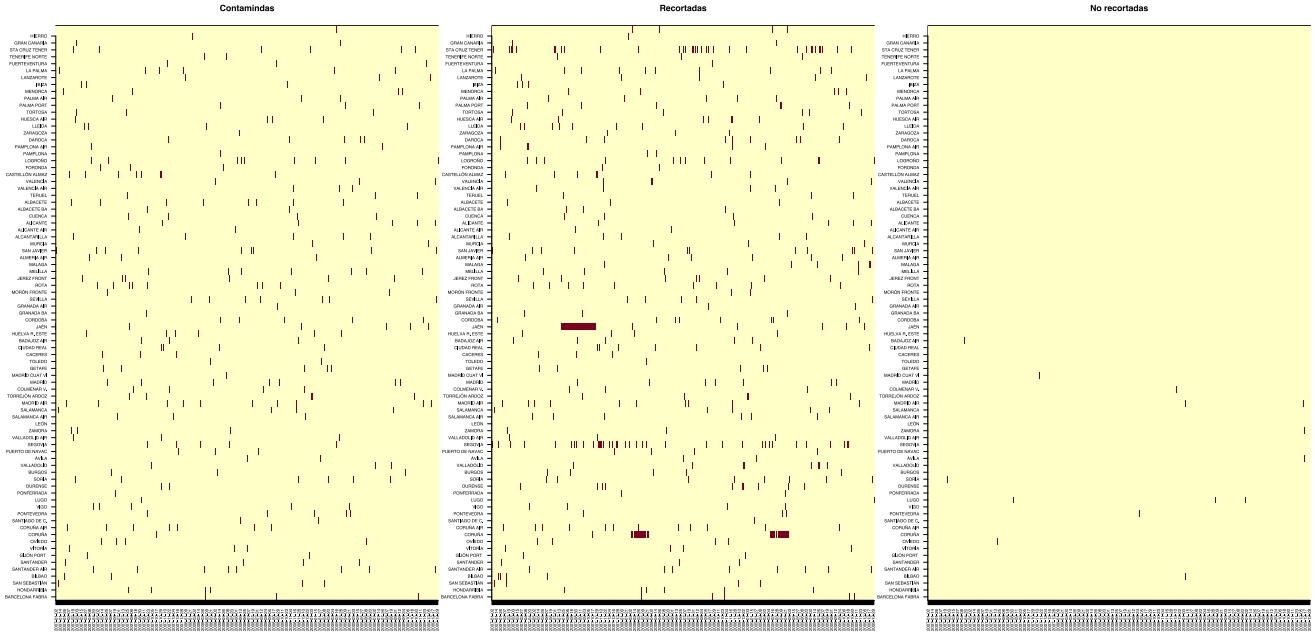


(a) TCLUST **(b)** RLG

Figura 5.12: Residuos por celda

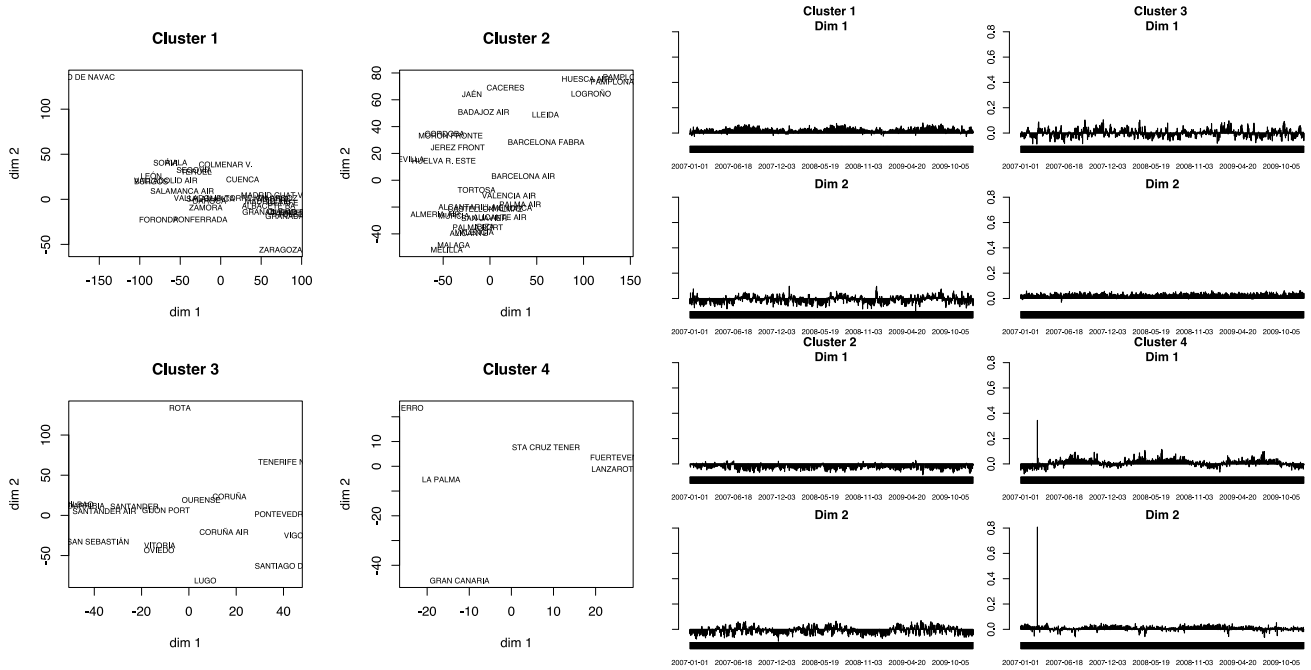


(a) TCLUST



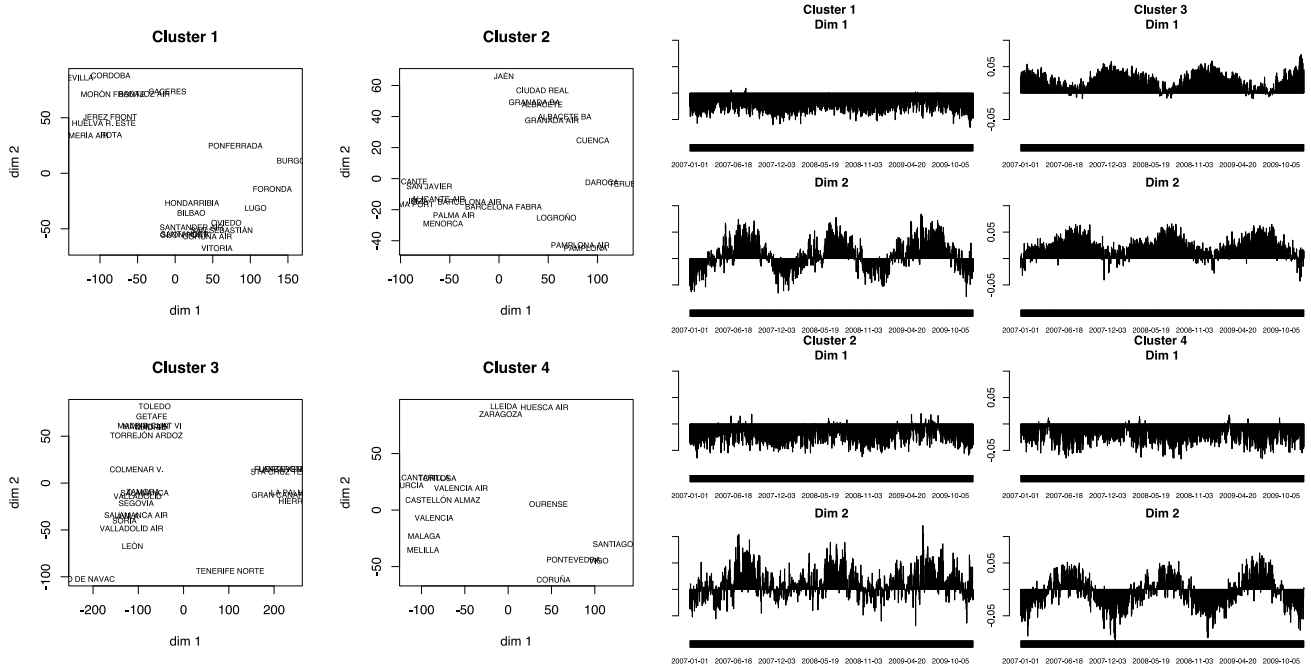
(b) RLG

Figura 5.13: Comparación de celdas contaminantes y recortadas



(a) Scores con TCLUST

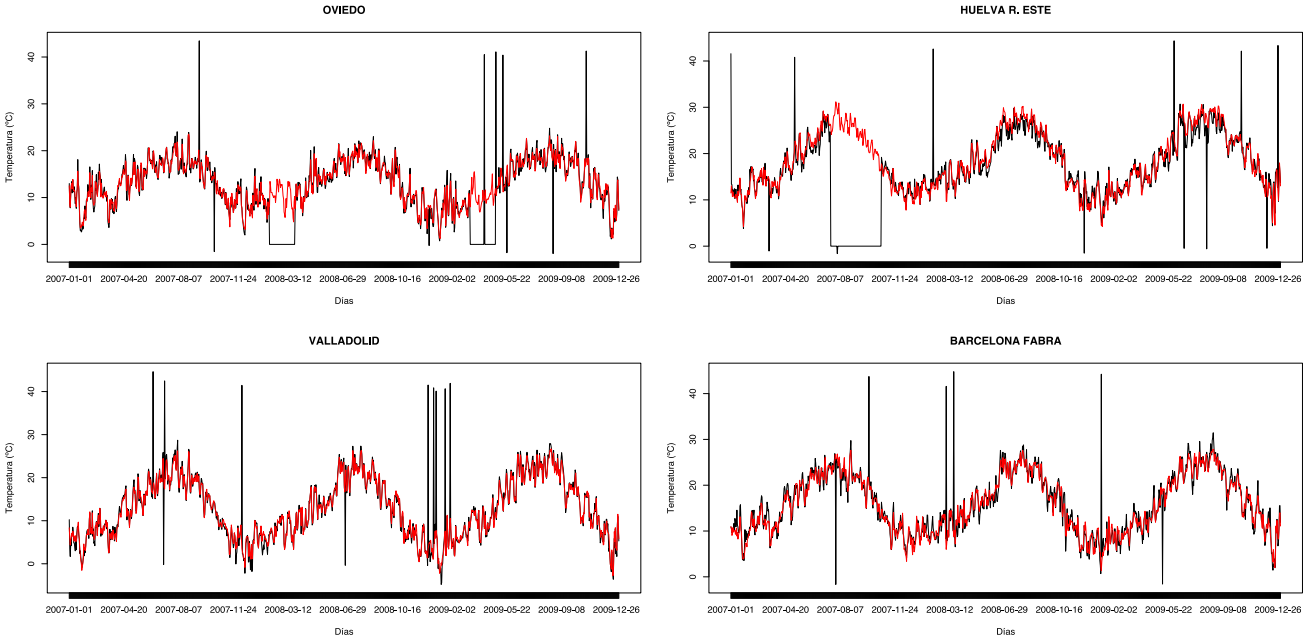
(b) Loadings con TCLUST



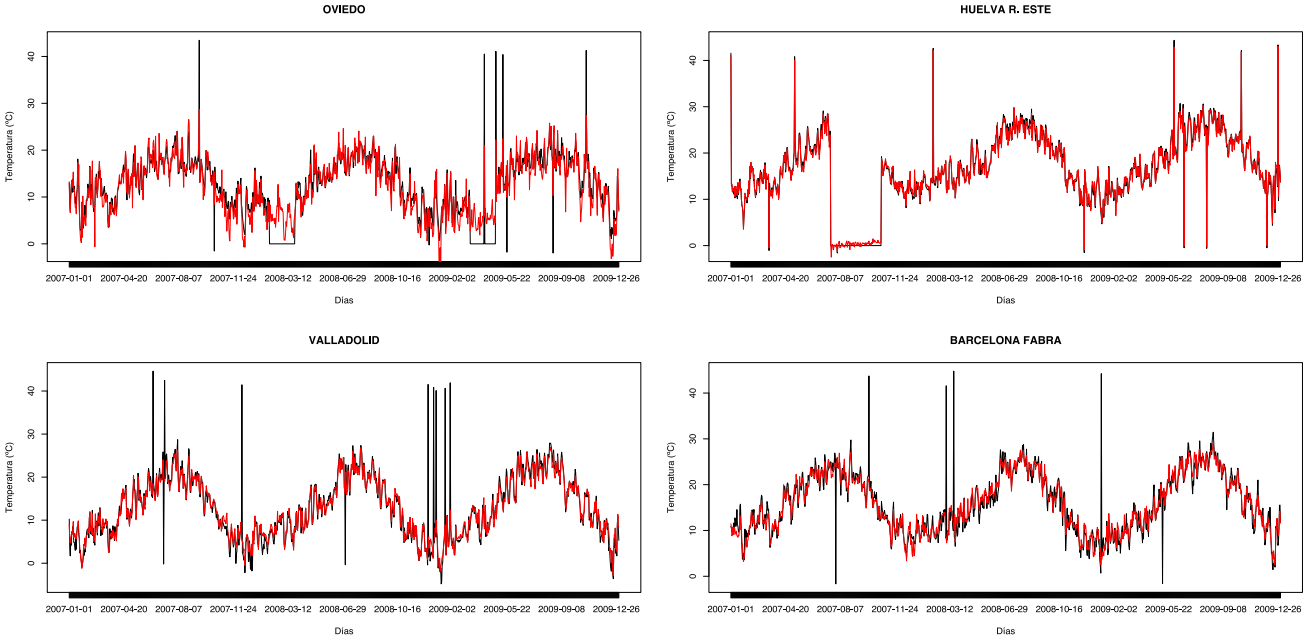
(c) Scores con RLG

(d) Loadings con RLG

Figura 5.14: Gráficos de "scores" y "loadings"



(a) TCLUST con $\alpha = 0.1$



(b) TCLUST con $\alpha = 0$

Figura 5.15: Gráfico comparando los valores reales (incluyendo valores contaminantes) en línea negra y los valores predichos `xpred` inicializando el método de recortes por celdas con el resultado de TCLUST en línea roja

Capítulo 6

Conclusiones y líneas futuras

En este Trabajo Fin de Grado se ha analizado una metodología para realizar Análisis Cluster permitiendo recorte por celdas, en lugar del típico recorte por observaciones, para proporcionar robustez frente a celdas atípicas en la matriz de datos. Después, se han propuesto varias mejoras sobre el algoritmo inicial, proporcionando mayor flexibilidad, rendimiento y diversos resúmenes gráficos.

A través de distintos escenarios de simulación se han mostrado las mejoras que se obtienen después de modificar el método con las propuestas y nuevas funcionalidades especificadas. En primer lugar, inicializando a través de un método de búsqueda de subespacios aproximantes como es RLG, en vez de hacerlo con TCLUST, parece mejorarse de forma evidentemente el funcionamiento de la técnica, especialmente en dimensiones más elevadas. Al añadir la posibilidad de especificar dimensiones intrínsecas diferentes se gana flexibilidad y se evita sobreajustar innecesariamente, dado que puede que cada cluster se ajuste a una estructura de dimensionalidad diferente de los demás. Por último se ha estudiado cómo se modifica el ajuste dependiendo del tamaño de recorte. Parece resultar preferible recortar más celdas de las estrictamente necesarias y posteriormente recuperararlas con el método automatizado propuesto. En la simulación, se ha analizado el caso de los datos funcionales, tanto dependiendo del método de inicialización como añadiendo diferentes tipos de contaminación.

Se han aplicado las distintas alternativas en dos conjuntos de datos reales, donde la inicialización con TCLUST proporcionaba una mayor interpretabilidad pero RLG conseguía minimizar más la función objetivo y detectaba mejor la contaminación incluida (en los datos de la AEMET). Es por ello que en el programa final se ha contemplado la posibilidad de inicializar con ambos métodos de clustering robusto.

En base a este trabajo es posible marcar futuras líneas de investigación, cuyo objetivo final sería la publicación de un paquete estadístico en CRAN ("The Comprehensive R Archive Network") para el lenguaje R.

Otra línea de trabajo futuro sería investigar la posibilidad de tratar datos perdidos o *missing* `NA` de forma eficiente. Podría pensarse en los datos perdidos como observaciones "atípicas" pero en las que su posición (i, j) sería conocida de antemano. En este trabajo se ha añadido el método DDC

que estima dichas celdas perdidas y permite la ejecución de la metodología con datos perdidos. No obstante, sería preferible modificar en cada paso las matrices w_{ij}^g de forma que sean siempre fijadas como $w_{ij}^g = 0$ para esas celdas perdidas. Ese peso igual a 0 haría que esas celdas fueran siempre "recortadas" y que no sean nunca utilizadas en la actualización de parámetros. Sin embargo, esto hace que haya que cambiar notablemente la implementación de gran parte del algoritmo y su implementación en R y faltaría analizar su funcionamiento a nivel práctico.

Por otra parte, para mejorar el rendimiento, se pueden realizar dos modificaciones. La primera de ellas, después de estudiar el tiempo que tarda en ejecutarse cada parte del código implementado, sería programar la regresión con pesos realizada en el paso de actualización de parámetros en C++. Pensamos que esto conseguiría una disminución considerable del tiempo de ejecución (actualmente se utiliza la orden `lm` programada en R). La segunda sería paralelizar partes del código en R, para que dichas partes de la computación se realicen de forma paralela en vez de secuencial aprovechando los distintos hilos de los procesadores modernos.

Apéndice A

Código fuente

A.1. cell.rlg

```
#' Robust cluster analysis with cellwise contamination
#'
#' @param x data frame or matrix with the input data
#' @param d vector with the intrinsec dimension of each group
#' @param init.method either "tclust" or "rlg"
#' @param alpha trimming level for the weights
#' @param alpha.lts trimming level for LTS
#' @param alpha.init trimming level for TCLUST or RLG
#' @param loops1 number of external loops
#' @param loops2 number of internal loops
#' @param nstart number of random inicializations of TCLUST or RLG
#' @param do.ddc whether to perform DDC before running the proposed method
#' @param scale whether to scale the input data
#' @param refinamiento whether to do the final improvement step
#' @param recover number of standard deviations to recover in the first stage
#' of the final improvement step
#' @param functional whether if the input data is functional. If true, perform
#' lowess and B-spline representation before running the algorithm
#' @param f the smoothing parameter in lowess
#' @param knots if functional, number of internal knots in the B-splines
#' representation
#'
#' @return S3 object of class "cell.rlg". The [plot.cell.rlg()] method can be
#' used to plot it.
#' It contains the following components:
#' \item{x}{matrix with the input data}
#' \item{d}{vector with the intrinsec dimension of each group}
#' \item{init}{object returned by TCLUST or RLG}
#' \item{functional}{whether if the input data is functional}
#' \item{xpred}{predicted data values}
#' \item{residuals}{standarized residuals}
#' \item{shat}{target function best value}
#' \item{A}{optimal loadings matrices}
#' \item{b}{optimal scores matrices}
#' \item{m}{optimal mean functions}
#' \item{cluster.cell}{optimal cell assigments}
#' \item{cluster}{optimal observation assigments}
#' \item{cluster.cell.raw}{optimal cell assigments before improvement step}
#' \item{cluster.raw}{optimal observation assigments before improvement step}
#' \item{evo}{Evolution of the loss}
#'
#' @references L.A. Garcia-Escudero, D. Rivera-Garcia, A. Mayo-Iscar and
#' J. Ortega (2021), \emph{Cluster analysis with cellwise trimming and
#' applications for the robust clustering of curves}, Information Sciences,
#' vol. 573, 100-124
```

```
46
47  library("tclust")
48  library("splines")
49  library("cellWise")
50  library("Rcpp")
51
52  sourceCpp("./src/script_lts.cpp")
53  source('./R/rlg.r')
54
55  cell.rlg <- function(x, alpha=0.1, d, alpha.init=0.2, alpha.lts=0.25,
56  f=1/5, knots=4,loops1=4, loops2=10, init.method="rlg",
57  functional=FALSE, do.ddc=FALSE,nstart=NULL,refinamiento=T,
58  recover=10,scale=TRUE){
59    x <- as.matrix(x)
60    if(is.null(nstart)){nstart <- sum(d)*40}
61    if (functional){scale=FALSE}
62    # Initialization of variables and matrices ----
63    K <- length(d)
64    n <- nrow(x)
65    p <- ncol(x)
66    W <- array(NA, c(n, p, K))
67    W.reduc <- W
68    m <- matrix(0, nrow = p, ncol = K)
69    cls <- rep(1, n)
70    rij <- matrix(0, nrow = length(x), ncol = K)
71    rijk <- array(NA, c(n, p, K))
72    m <- matrix(0, nrow = p, ncol = K)
73    r <- matrix(0, nrow = n, ncol = K)
74    A <- list()
75    b <- list()
76    i <- 1
77    for (qi in d) {
78      Ai <- array(NA, dim = c(qi, n))
79      bi <- array(NA, dim = c(qi, p))
80      A[[i]] <- Ai
81      b[[i]] <- bi
82      i <- i + 1
83    }
84    xhat <- array(NA, dim = c(n, p, K))
85    #####
86    # An equispaced grid to work in the [0,1] interval
87    s <- (1:p)/p
88    if(scale == TRUE){
89      standard <- function(x){(x-median(x,na.rm=TRUE))/mad(x,na.rm=TRUE)}
90      x <- apply(x,2,standard)
91    }
92    if(do.ddc){
93      XX <- DDC(x)$Xest
94    }else{
95      XX <- x
96    }
97    if(functional){
98      # Smoothing with "lowess" ----
99      lowess.mod <- function(x){lowess(x,f=f,iter=20)$y}
100     X.smooth <- t(apply(x,1,lowess.mod))
101     # B-splines fitting (additional smoothing and dimensionality reduction) ----
102     # Total number of knots
103     knots <- knots
104     df <- knots+4
105
106     # XX is the "small dimensional" representation of data for initialization
107     # B-splines
108     XX <- matrix(rep(0,n*df),nrow=n)
109     bb <- bs(s, df=df,intercept=TRUE)
110     for (u in 1:n){
111       fit <- lm(X.smooth[u,] ~ -1+bb)
112       XX[u,] <- fit$coefficients
113     }
114   }
115   # Initializing the m vectors and the B matrices ----
116   # TCLUST robust clustering method applied on XX for initialization ----
```

```
  # (only onerandom initialization based on TCLUST is applied but other robust
  # clustering can be applied too)
  if(init.method == "tclust"){
    a <- tclust(XX,k=K,alpha=alpha.init,nstart=nstart,restr.fact=10,warnings=0)

    ## Mean vectors
    m <- a$centers

    ## B matrices
    for (k in 1:K){
      qi <- d[k]
      b[[k]] <- t(eigen(a$cov[,,k])$vectors[,1:qi])
    }
  }
  # RLG  method applied on XX for initialization ----
  else if (init.method == "rlg"){
    a <- rlg(XX, d=d,nstart=nstart, alpha=alpha.init)
    b <- lapply(a$U, t)
    m <- a$centers
  }
  # Functional case ----
  if(functional){
    mt <- matrix(0, nrow = p, ncol = K)
    bt <-list()
    i <- 1
    for (qi in d) {
      bi <- array(NA, dim = c(qi, p))
      bt[[i]] <- bi
      i <- i + 1
    }
    for (k in 1:K){
      ## Means
      mt[,k] <- bb[,]%*%m[,k]
      ## Autofunction
      for (j in 1:d[k]){
        bt[[k]][j,]<- bb[,]%*%b[[k]][j,]
      }
    }
    b <- bt
    m <- mt
  }
  # External and internal loops ----
  # Initializing target function with a very high value
  shat.opt <- 10^10
  # Evolution of the loss
  evo <-c()
  # Abort computation if troubles
  abort <- 1
  # Best values
  best.shat <- Inf
  best.values <- list()
  # Starting the external loops (loop1) ----
  it1 <- 0
  while((it1<loops1)&(abort==1)){
    it1 <- it1+1
    # First A's matrices and first assignments
    for (k in (1:K)) {
      # Center data matrix
      xc <- scale(x, center = m[, k], scale = FALSE)
      qi <- d[k]
      # Initialzing A's matrix by using LTS robust regression
      for (i in 1:n) {
        rr <- fastlts(matrix(t(b[[k]]), nrow = p, ncol = qi), xc[i, ],
        alpha = alpha.lts, nsamp = 10)
        A[[k]][, i] <- rr$coefficients
        # Use LTS residuals to compute "distances" to approximating subspaces
        r[i, k] <- sum(rr$residuals[rr$best] ^ 2)
      }
    }
    # First assignments from LTS
    cls <- apply(r,1,which.min)
    # Starting internal loops (loops2) ----
    #(improving m vector, B matrices and A matrices with fixed assignments)
```

```
    it2 <- 0
    while((it2<loops2)&(abort==1)){
      it2 <- it2+1
      # Matrix with updated weights
      # Estimated data matrix xhat
      for(k in 1:K){
        qi <- d[k]
        for(i in 1:n){
          xhat[i,,k] <- m[,k]+as.vector(matrix(t(b[[k]]),nrow=p,ncol=qi)%*%
          (matrix(A[[k]][,i],nrow=qi,ncol=1)))
          rijk[i,,k] <- (x[i,]-xhat[i,,k])^2
        }
      }
      # Creating weights
      n.cls <- table(cls)
      for(k in 1:K){
        for(j in 1:p){
          W[cls==k,j,k] <- as.numeric(
          rijk[cls==k,j,k] <=
          sort(rijk[cls==k,j,k])[floor(n.cls[k]*(1-alpha))]
          )
        }
      }
      # Reduced matrices of weights (W.reduc)
      for(k in 1:K){
        W.reduc[,,k] <- W[,,k]
        W.reduc[which(cls!=k),,k]<-rep(0,p)
      }
      # Updating m vectors and the B' and A's matrices
      for (k in 1:K) {
        ### Updating B's matrices
        qi <- d[k]
        rida <- apply((W.reduc[, , k]), 2, sum)
        ida <- which(rida > (qi + 1))
        for (j in sort(ida)) {
          b[[k]][, j] <-
          lm((x[, j] - m[j, k]) ~ matrix(t(A[[k]]), nrow = n, ncol = qi) - 1,
          weights = W.reduc[, j, k])$coefficients
        }
        # Normalization of b
        for (l in (1:d[k])){b[[k]][l, ] <- b[[k]][l,]/sqrt(sum(b[[k]][l, ]^2))}
      }
      # Updating mean vectors
      rm <- x
      for (k in 1:K) {
        qi <- d[k]
        for (i in 1:n) {
          rm[i, 1:p] <-
          x[i,] - as.vector(matrix(t(b[[k]]), nrow = p, ncol = qi) %*%
          (matrix(A[[k]][, i], nrow = qi, ncol = 1)))
        }
        m[, k] <- as.matrix((1/colSums(W.reduc[, , k])) *
        as.matrix(colSums(W.reduc[, , k]*rm)))
      }
      # Updating A's matrices
      for(k in 1:K){
        # Centred data
        xc<-scale(x,center = m[,k],scale = FALSE)

        qi <- d[k]
        iw<-apply(as.matrix(W.reduc[,,k]),1,sum)
        inx<-which(iw>(qi+1))
        for(i in inx) {
          A[[k]][,i] <- lm(xc[i, ]~
          matrix(t(b[[k]]),nrow=p,ncol=qi)-1,
          weights = W.reduc[i,1:p,k])$coefficients
        }
      }
      # Check if everything is ok
```

```
      check <- rep(0,K)
      for (k in (1:K)){
        check[k]<-sum(W[which(cls==k),,k])
      }
      # Compute target function or abort computing if troubles appear
      # (due to empty clusters)
      if (min(check) > 10) {
        shat <- 0
        for (k in 1:K) {
          if (is.na(sum(rijk[, , k])) == F)   {
            shat <- shat + sum(W.reduc[, , k] * rijk[, , k])
          }  else {
            shat <- shat + 10 ^ 10
          }
        }
        evo <- c(evo,shat)
      }
      else{
        abort <- 0
        print("Abort computing")
        shat <- 10 ^ 10
      }
    }  # Ending of loop2
    # Save the best iteration
    if(shat < best.shat){
      best.shat <- shat
      best.values <- list(
      A=A,b=b,m=m,xhat=xhat,
      cls=cls,W.reduc=W.reduc
      )
    }
  }  # Ending of loop1
  ## Ending of the iterative procedure

  # Retrieve the best iteration ----
  A <- best.values$A
  b <- best.values$b
  m <- best.values$m
  xhat <- best.values$xhat
  cls <- best.values$cls
  W.reduc <- best.values$W.reduc
  # Predictions in a nxp matrix ----
  xpred <- x
  for (i in 1:n){
    xpred[i,]<-xhat[i,,cls[i]]
  }
  # Get the residuals in a nxp matrix ----
  rij <- x - xpred
  rij.scaled <- matrix(NA, ncol = p, nrow = n)
  Wij <- matrix(NA, ncol = p, nrow = n)
  cluster.cell <- matrix(NA, ncol = p, nrow = n)
  for(k in 1:K){
    index <- which(cls == k)
    if(!is.null(dim(rij[index,]))){
      rjk <- apply(rij[index,], 2, mad, na.rm=T)
      rij.scaled[index,] <- sweep(rij[index,],2,rjk,"/")
    }else{
      rjk <- mad(rij[index,])
      rij.scaled[index,] <- rij[index,]/rjk
    }
    Wij[index,] <- W.reduc[index, ,k]
    cluster.cell[index,] <- ifelse(Wij[index,] == 1, k, 0)
  }

  ret <- list(init=a,
  cluster.raw=cls,
  A=A,
  b=b,
  m=m,
  xpred=xpred,
  cluster.cell.raw=cluster.cell,
  cluster.raw=cls,
```

```
  shat=best.shat,
  evo=evo)
  if(refinamiento){
    # Final improvement step ----
    # First improvement: recover incorrectly trimmed cells
    Wij <- Wij | abs(rij.scaled) < recover
    for(k in 1:K){
      index <- which(cls == k)
      cluster.cell[index,] <- ifelse(Wij[index,] == 1, k, 0)
    }
    # Second improvement: trim complete observations
    index.trim <- !apply(Wij,1,sum)/p > alpha.lts
    Wij[index.trim, ] <- 0
    cls[index.trim] <- 0
    cluster.cell[index.trim,] <- 0
    ret <- append(ret, list(cluster.cell=cluster.cell,
    cluster=cls))
    # Set NA the residuals of trimmed observations
    rij.scaled <- ifelse(matrix(rep(cls,each=p), ncol=p, byrow = T) == 0,
    NA,
    rij.scaled)
  }
  ret <- append(ret, list(residuals=rij.scaled))
  ret <- append(ret, list(functional=functional))
  ret <- append(ret, list(x=x))
  ret <- append(ret, list(d=d))
  # Returning the final output of the procedure
  class(ret) <- "cell.rlg"
  return(ret)
}
```

A.2. fastlts

```
#include <RcppArmadillo.h>
#include <Rcpp.h>
#include <iostream>
#include <algorithm>

using namespace Rcpp;
using namespace arma;
using namespace std;

//' @title Fast method to compute LTS Regression
//'
//' @param x matrix containing the explanatory variables
//' @param y vector with the response
//' @param alpha trimmed proportion of observations. Must be in
//' @param nsamp number of iterations of the external loop
//' @param n_best number of iterations of the internal loop
//'
//' @return NumericVector coefficients computed with LTS
// [[Rcpp::depends(RcppArmadillo)]]
// [[Rcpp::export]]
List fastlts(NumericMatrix x, NumericVector y,
        double alpha = 1 / 3, int nsamp = 20, int n_best = 5)
{
  int n = x.nrow();
  int p = x.ncol();
  mat x_arma = as<mat>(x);
  vec y_arma = as<vec>(y);
  mat b_l1 = mat(nsamp, p);
  vec r2_l1 = vec(nsamp);
  int h = (int)std::max(p, (int)ceil(n * (1 - alpha)));
  // Inicialization
  for (int i = 0; i < nsamp; i++)
```

```
33    {
34      IntegerVector h_i = sample(n, p) - 1;
35      uvec hi = as<uvec>(h_i);
36      uvec index;
37      mat x_h1 = x_arma.rows(hi);
38      vec y_h1 = y_arma.rows(hi);
39      vec r2;
40      vec b = arma::solve(x_h1, y_h1, solve_opts::fast);
41      // 2 C-steps
42      for (int j = 0; j < 3; j++)
43      {
44        vec r = y_arma - x_arma * b;
45        r2 = pow(r, 2);
46        vec r2_sorted = arma::sort(r2);
47        index = arma::find(r2 <= r2_sorted(h - 1));
48        x_h1 = x_arma.rows(index);
49        y_h1 = y_arma.rows(index);
50        b = arma::solve(x_h1, y_h1, solve_opts::fast);
51      }
52      float Qi = arma::sum(r2.elem(index)); // Q_3
53
54      b_l1.row(i) = b.t();
55      r2_l1(i) = Qi;
56    }
57    // Get the n_best smaller residuals
58    uvec order = arma::sort_index(r2_l1);
59    uvec best_index;
60    vec best_b;
61    float best_r2 = INFINITY;
62    vec best_resid;
63    bool terminar;
64    vec r, r2;
65    // C-steps until converge
66    for (int i = 0; i < n_best; i++)
67    {
68      vec b1 = b_l1.row(order(i)).t();
69      uvec index;
70      mat x_h1;
71      mat y_h1;
72      vec b0;
73      do
74      {
75        b0 = b1;
76        r = y_arma - x_arma * b1;
77        r2 = pow(r, 2);
78        vec r2_sorted = arma::sort(r2);
79        try
80        {
81          index = arma::find(r2 <= r2_sorted(h));
82        }
83        catch (const std::out_of_range)
84        {
85          index = arma::find(r2 <= r2_sorted(h - 1));
86        }
87        x_h1 = x_arma.rows(index);
88        y_h1 = y_arma.rows(index);
89        b1 = arma::solve(x_h1, y_h1, solve_opts::fast);
90        terminar = true;
91        for (int j = 0; j < p; j++)
92        {
93          if (std::abs(b0(j) - b1(j)) > (1e-8 + 1e-5 + std::abs(b1(j))))
94          {
95            terminar = false;
96            break;
97          }
98        }
99      } while (!terminar);
100     float Qi = arma::sum(r2.elem(index));
101     // Save best iteration according to objective function
102     if (best_r2 > Qi)
```

```
    {
      best_b = b1;
      best_index = index;
      best_r2 = Qi;
      best_resid = r;
    }
  }
  NumericVector b_ret = NumericVector(best_b.begin(), best_b.end());
  NumericVector r_ret = NumericVector(best_resid.begin(), best_resid.end());
  NumericVector index_ret = NumericVector(best_index.begin(), best_index.end()) + 1;
  List ret = List::create(Named("coefficients") =
        b_ret,_["crit"] = best_r2,
        _["best"] = index_ret,
        _["residuals"] = r_ret);

  return ret;
}
```

A.3. plot.cell.rlg

```
#' @description Four plots (selectable by \code{which}) are currently provided:
#' \enumerate{
#'      \item a plot with the cellwise standarized residuals
#'      \item a plot with the assignation to clusters
#'      \item G plots with the scores
#'      \item \eqn{G \times q_g} plots with the loadings
#' }
#' Moreover, if the data is \code{functional}, there are three more plots
#' provided:
#' \enumerate{
#'      \item a plot with the assignation of each curve to a group
#'      \item G plots with the assignation of each curve to a group
#'      \item a plot with the mean curve of each group
#' }
#' @param obj a 'cell.rlg' object, result of \code{cell.rlg}
#' @param which string indicating which plot to show
#' @param sort whether the observations should be sorted according to the group
#' @details Possible options to \code{which} argument:
#'
#' \describe{
#'      \item{\code{all}:}{all plots that could be done}
#'      \item{\code{residuals}:}{a plot with the cellwise standarized residuals}
#'      \item{\code{clusters}:}{a plot with the assignation to clusters}
#'      \item{\code{scores}:}{G plots with the scores}
#'      \item{\code{loadings}:}{\eqn{G \times q_g} plots with the loadings}
#'      \item{\code{curves}:}{a plot with the assignation of each curve to a group}
#'      \item{\code{curves.splitted}:}
#'      {G plots with the assignation of each curve to a group}
#'      \item{\code{mean.curves}:}{a plot with the mean curve of each group}
#' }
plot.cell.rlg <- function(obj, which="all", sort = TRUE){
  if(!inherits(obj, "cell.rlg"))
  stop("Use only with 'cell.rlg' objects")
  par(mfrow=c(1,1))
  if(obj$functional){
    .plot.cell.rlg.functional(obj, which=which, sort=sort)
  }else{
    .plot.cell.rlg(obj, which=which, sort=sort)
  }
  par(mfrow=c(1,1))
}

.plot.cell.rlg <- function(obj, which, sort){
  if(which == "all" || which == "residuals")
```

```
45    .plot.residuals(obj, sort=sort)
46    if(which == "all" || which == "clusters")
47    .plot.clusters(obj, sort=sort)
48    if(which == "all" || which == "scores")
49    .plot.scores(obj)
50    if(which == "all" || which == "loadings")
51    .plot.loadings(obj)
52  }
53
54  .plot.cell.rlg.functional <- function(obj, which, sort){
55    if(which == "all" || which == "curves")
56    .plot.curves(obj)
57    if(which == "all" || which == "mean.curves")
58    .plot.mean.curves(obj)
59    if(which == "all" || which == "curves.splitted")
60    .plot.curves.splitted(obj)
61    if(which == "all" || which == "residuals")
62    .plot.residuals(obj, sort=sort)
63    if(which == "all" || which == "clusters")
64    .plot.clusters(obj, sort=sort)
65    if(which == "all" || which == "scores")
66    .plot.scores(obj)
67    if(which == "all" || which == "loadings")
68    .plot.loadings(obj)
69  }
70
71  .plot.residuals <- function(obj, sort){
72    pal <- c('#3B9AB2','#3B9AB2','#3C9AB2','#3D9BB2','#3E9BB3','#3F9CB3','#409CB3','#419DB4','#4
73
74    index <- .sort(obj, sort)
75
76    rownames(obj$residuals) <- .get.rownames(obj)
77    colnames(obj$residuals) <- .get.colnames(obj)
78
79    obj$residuals[obj$residuals > 10] <- 10
80    obj$residuals[obj$residuals < -10] <- -10
81
82    heatmap(obj$residuals[index,], Rowv = NA, Colv = NA, scale = "none", col=pal)
83  }
84
85  .plot.clusters <- function(obj, sort){
86    index <- .sort(obj, sort)
87
88    rownames(obj$cluster.cell) <- .get.rownames(obj)
89    colnames(obj$cluster.cell) <- .get.colnames(obj)
90    if(sum(obj$cluster.cell == 0) == 0)
91    colors <- 1:length(obj$d)+1
92    else
93    colors <- 1:(length(obj$d)+1)
94
95    heatmap(obj$cluster.cell[index,], Rowv = NA, Colv = NA, scale = "none",
96    col = colors)
97  }
98
99  .plot.scores <- function(obj){
100   par(mfrow=rev(n2mfrow(length(obj$d))))
101
102   for(k in seq_len(length(obj$d))){
103     colnames(obj$A[[k]]) <- .get.rownames(obj)
104     if(obj$d[k] >= 2){
105       plot(obj$A[[k]][1, obj$cluster==k], obj$A[[k]][2, obj$cluster==k],
106       main=paste("Cluster", k), xlab="dim 1", ylab="dim 2", col="white")
107       text(obj$A[[k]][1, obj$cluster==k], obj$A[[k]][2, obj$cluster==k],
108       as.character(colnames(obj$A[[k]])[obj$cluster==k]),cex=0.6)
109     }
110
111   }
112 }
113
114 .plot.loadings <- function(obj){
115   par(mar=c(2,1,2,1))
116   dim <- rev(n2mfrow(length(obj$d)))
```

```
  d.max <- min(max(obj$d),3)
  dim[1] <- dim[1]*d.max
  mat <- matrix(1:(dim[1]*dim[2]), nrow = dim[1], ncol = dim[2])
  layout(mat)
  ylims <- c(min(unlist(obj$b)), max(unlist(obj$b)))
  for(k in 1:length(obj$d)){
    n2 <- min(obj$d[k], 3)
    for(i2 in 1:d.max){
      if(i2 <= n2){
        title <- ifelse(i2 == 1, paste("Cluster", k, "\nDim", i2) , paste("Dim", i2))
        bar <- barplot(obj$b[[k]][i2,], main=title, ylim = ylims)
        axis(1, at = bar, labels=.get.colnames(obj), cex.axis=0.7)
      }else{
        plot(0,type='n',axes=FALSE,ann=FALSE)
      }
    }
  }
  par(mfrow=c(1,1))
}

.plot.curves <- function(obj){
  p <- ncol(obj$x)
  n <- nrow(obj$x)
  s <- (1:p)/p
  plot(s,obj$x[1,],type="n",col=obj$cluster[1]+1,
  ylim=c(min(obj$x),max(obj$x)),xlab="",ylab="", xaxt = "n")
  axis(1, at=s, labels=.get.colnames(obj))
  for(i in 1:n){
    lines(s,obj$x[i,],col=obj$cluster[i]+1)
    points(s[which(obj$cluster.cell[i,]==0)],
    obj$x[i,obj$cluster.cell[i,]==0], cex=0.6)
  }
}

.plot.mean.curves <- function(obj){
  p <- ncol(obj$x)
  n <- nrow(obj$x)
  s <- (1:p)/p
  plot(s,obj$x[1,],type="n",col=obj$cluster[1]+1,
  ylim=c(range(obj$x)[1],range(obj$x)[2]),xlab="",ylab="", xaxt = "n")
  axis(1, at=s, labels=.get.colnames(obj))
  for(k in 1:length(obj$d)){
    mean.curve <- apply(obj$x[obj$cluster == k,],2,mean)
    lines(s,mean.curve,col=k+1)
  }
}

.plot.curves.splitted <- function(obj){
  p <- ncol(obj$x)
  n <- nrow(obj$x)
  s <- (1:p)/p
  K <- which(obj$d >= 2)
  par(mfrow=rev(n2mfrow(length(K))))
  for(k in 1:length(obj$d)){
    plot(s,obj$x[1,],type="n",col=obj$cluster[1]+1,
    ylim=c(min(obj$x),max(obj$x)),xlab="",ylab="",
    main=paste("Cluster", k), xaxt = "n")
    axis(1, at=s, labels=.get.colnames(obj))
    for(i in 1:n){
      if(obj$cluster[i] == k){
        lines(s,obj$x[i,],col=obj$cluster[i]+1)
        points(s[which(obj$cluster.cell[i,]==0)],
        obj$x[i,obj$cluster.cell[i,]==0], cex=0.6)
      }
    }
  }
}
```

```
184
185 .sort <- function(obj, sort){
186   if(sort)
187   index <- order(obj$cluster)
188   else
189   index <- 1:length(obj$cluster)
190
191   return(index)
192 }
193
194 .get.rownames <- function(obj){
195   if(!is.null(rownames(obj$x)))
196   return(rownames(obj$x))
197   else
198   return(1:nrow(obj$x))
199 }
200
201 .get.colnames <- function(obj){
202   if(!is.null(colnames(obj$x)))
203   return(colnames(obj$x))
204   else
205   return(1:ncol(obj$x))
206 }
```

A.4. plot.rlg

```
1  plot.rlg <- function(obj, which="all", sort = TRUE){
2    if(!inherits(obj, "rlg"))
3    stop("Use only with 'rlg' objects")
4    if(which == "all" || which == "clusters")
5    .plot.clusters(obj, sort=sort)
6    if(which == "all" || which == "scores")
7    .plot.scores(obj)
8    if(which == "all" || which == "loadings")
9    .plot.loadings(obj)
10   if(which == "all" || which == "eigenvalues")
11   .plot.eigenvalues(obj)
12 }
13
14 .plot.clusters <- function(obj, sort){
15   index <- .sort(obj, sort)
16
17   p <- ncol(obj$x)
18   mat <- matrix(rep(obj$cluster, each=p), ncol=p, byrow=T)
19
20   rownames(mat) <- .get.rownames(obj)
21   colnames(mat) <- .get.colnames(obj)
22
23   if(sum(obj$cluster == 0) == 0)
24   colors <- 1:length(obj$d)+1
25   else
26   colors <- 1:(length(obj$d)+1)
27
28   heatmap(mat[index,], Rowv = NA, Colv = NA, scale = "none", col = colors)
29 }
30
31 .plot.scores <- function(obj){
32   par(mfrow=rev(n2mfrow(length(obj$d))))
33   for(k in seq_len(length(obj$d))){
34     scores <- princomp(obj$x[obj$cluster == k, ],cor = TRUE)$scores
35     plot(scores[,1], scores[,2],
36     main=paste("Cluster", k), xlab="dim 1", ylab="dim 2", col="white")
37     text(scores[,1], scores[,2],
38     as.character(.get.rownames(obj)[obj$cluster==k]), cex=0.6)
39   }
40 }
41
42 .plot.loadings <- function(obj){
```

```
  dim <- rev(n2mfrow(length(obj$dimensions)))
  d.max <- min(max(obj$dimensions),3)
  dim[1] <- dim[1]*d.max
  mat <- matrix(1:(dim[1]*dim[2]), nrow = dim[1], ncol = dim[2])
  layout(mat)
  for(k in 1:length(obj$dimensions)){
    n2 <- min(obj$dimensions[k], 3)
    for(i2 in 1:d.max){
      if(i2 <= n2){
        title <- ifelse(i2 == 1, paste("Cluster", k, "\nDim", i2) , paste("Dim", i2))
        barplot(obj$U[[k]][,i2], main=title)
      }else{
        plot(0,type='n',axes=FALSE,ann=FALSE)
      }
    }
  }
  par(mfrow=c(1,1))
}

.plot.eigenvalues <- function(obj){
  p <- ncol(obj$x)
  eigenvalues <- matrix(NA,nrow=length(obj$dimensions),ncol=p)
  for (k in 1:length(obj$dimensions)){
    eigenvalues[k,] <- princomp(obj$x[obj$cluster == k, ],cor = TRUE)$sdev^2
  }
  lim.sup <- max(eigenvalues)
  plot(eigenvalues[1,],ylim=c(0,lim.sup),type="n",xlab="intrisic dimensions",
  ylab="eigenvalues")
  for (k in 1:length(obj$dimensions)){
    lines(eigenvalues[k,],col=k+1,type="b",pch=19)
  }
  legend("topright",
  legend=paste("Cluster",as.character(1:length(obj$dimensions))),
  col=1:length(obj$dimensions)+1, lty=rep(1,length(obj$dimensions)),
  pch=rep(19,length(obj$dimensions)))
}

.sort <- function(obj, sort){
  if(sort)
  index <- order(obj$cluster)
  else
  index <- 1:length(obj$cluster)

  return(index)
}

.get.rownames <- function(obj){
  if(!is.null(rownames(obj$x)))
  return(rownames(obj$x))
  else
  return(1:nrow(obj$x))
}

.get.colnames <- function(obj){
  if(!is.null(colnames(obj$x)))
  return(colnames(obj$x))
  else
  return(1:ncol(obj$x))
}
```

Bibliografía

[1] L. Á. García-Escudero, D. Rivera-García, A. Mayo-Iscar y J. Ortega, «Cluster analysis with cellwise trimming and applications for the robust clustering of curves,» *Information Sciences*, vol. 573, págs. 100-124, 2021. DOI: 10.1016/j.ins.2021.05.004.

[2] B. S. Everitt, S. Landau, M. Leese y D. Stahl, *Cluster Analysis* (Wiley Series in Probability and Statistics), eng, 5th ed., S. Landau, M. Leese y D. Stahl, eds. Chicester: Wiley, 2010, pág. 330, ISBN: 1-280-76795-2.

[3] R. A. Maronna, R. D. Martin, V. J. Yohai y M. Salibián-Barrera, *Robust statistics : theory and methods (with R)* (Wiley series in probability and statistics), 2nd ed. Hoboken, New Jersey: WIley, 2019, ISBN: 9781119214656.

[4] F. Alqallaf, S. Van Aelst, V. J. Yohai y R. H. Zamar, «Propagation of outliers in multivariate data,» *The Annals of Statistics*, vol. 37, n.º 1, págs. 311-331, 2009. DOI: 10.1214/07-AOS588.

[5] L. Á. García-Escudero y A. Gordaliza, «Robustness Properties of k Means and Trimmed k Means,» *Journal of the American Statistical Association*, vol. 94, n.º 447, págs. 956-969, 1999. DOI: 10.1080/01621459.1999.10474200.

[6] K. Pearson, «On lines and planes of closest fit to systems of points in space,» *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, n.º 11, págs. 559-572, 1901. DOI: 10.1080/14786440109462720.

[7] H. Hotelling, «Analysis of a complex of statistical variables into principal components.,» *Journal of Educational Psychology*, vol. 24, n.º 6, págs. 417-441, 1933. DOI: 10.1037/h0071325.

[8] E. J. Candès, X. Li, Y. Ma y J. Wright, «Robust Principal Component Analysis?» *Journal of the ACM*, vol. 58, n.º 3, 2011. DOI: 10.1145/1970392.1970395.

[9] J. Wright, A. Ganesh, S. Rao, Y. Peng e Y. Ma, *Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices via Convex Optimization*. dirección: https://papers.nips.cc/paper/2009/file/c45147dee729311ef5b5c3003946c48f-Paper.pdf.

[10] B. Wohlberg, R. Chartrand y J. Theiler, «Local principal component pursuit for nonlinear datasets,» en *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, págs. 3925-3928. DOI: 10.1109/ICASSP.2012.6288776.

[11] P. J. Rousseeuw, A. M. Leroy y J. Wiley, *Robust Regression and Outlier Detection* (Wiley Series in Probability and Statistics). Hoboken, NJ, USA: John Wiley & Sons, Inc., 1987, ISBN: 9780471852339. DOI: 10.1002/0471725382.

[12] A. Gordaliza, «Best approximations to random variables based on trimming procedures,» *Journal of Approximation Theory*, vol. 64, n.º 2, págs. 162-180, 1991. DOI: 10.1016/0021-9045(91)90072-I.

[13] R. Maronna, «Principal Components and Orthogonal Regression Based on Robust Scales,» *Technometrics*, vol. 47, n.º 3, págs. 264-273, 2005. dirección: http://www.jstor.org/stable/25471020.

[14] J. A. Cuesta-Albertos, A. Gordaliza y C. Matrán, «Trimmed k-means: an attempt to robustify quantizers,» *The Annals of Statistics*, vol. 25, n.º 2, págs. 553-576, 1997. DOI: 10.1214/AOS/1031833664.

[15] L. Á. García-Escudero, A. Gordaliza y C. Matrán, «Trimming Tools in Exploratory Data Analysis,» *Journal of Computational and Graphical Statistics*, vol. 12, n.º 2, págs. 434-449, 2003. DOI: 10.1198/1061860031806.

[16] L. Á. García-Escudero, A. Gordaliza, C. Matrán y A. Mayo-Iscar, «A general trimming approach to robust cluster Analysis,» *The Annals of Statistics*, vol. 36, n.º 3, págs. 1324-1345, 2008. DOI: 10.1214/07-AOS515.

[17] H. Fritz, L. Á. García-Escudero y A. Mayo-Iscar, «tclust: An R Package for a Trimming Approach to Cluster Analysis,» *Journal of Statistical Software*, vol. 47, n.º 12 SE - Articles, págs. 1-26, 2012. DOI: 10.18637/jss.v047.i12.

[18] M. T. Gallegos, «Maximum Likelihood Clustering with Outliers BT - Classification, Clustering, and Data Analysis,» K. Jajuga, A. Sokołowski y H.-H. Bock, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, págs. 247-255, ISBN: 978-3-642-56181-8. DOI: 10.1007/978-3-642-56181-8_27.

[19] M. T. Gallegos y G. Ritter, «A robust method for cluster analysis,» *The Annals of Statistics*, vol. 33, n.º 1, págs. 347-380, 2005. DOI: 10.1214/009053604000000940.

[20] A. P. Dempster, N. M. Laird y D. B. Rubin, «Maximum likelihood from incomplete data via the EM algorithm,» *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 39, n.º 1, págs. 1-38, 1977. dirección: https://www.jstor.org/stable/2984875.

[21] L. Kaufman y P. J. Rousseeuw, *Finding Groups in Data* (Wiley Series in Probability and Statistics), L. Kaufman y P. J. Rousseeuw, eds. Hoboken, NJ, USA: John Wiley & Sons, Inc., 1990, ISBN: 9780470316801. DOI: 10.1002/9780470316801.

[22] L. Á. García-Escudero, A. Gordaliza, R. San Martín, S. Van Aelst y R. Zamar, «Robust linear clustering,» *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, n.º 1, págs. 301-318, 2009. DOI: 10.1111/J.1467-9868.2008.00682.X.

[23] J. Fernández Iglesias, «Implementación de técnicas para análisis cluster robusto en torno a subespacios afines,» 2021. dirección: https://uvadoc.uva.es/handle/10324/50485.

[24] R. A. Maronna y V. J. Yohai, «Robust Low-Rank Approximation of Data Matrices with Elementwise Contamination,» *Technometrics*, vol. 50, n.º 3, págs. 295-304, 2008. dirección: http://www.jstor.org/stable/25471491.

[25] H. Cevallos Valdiviezo, «On methods for prediction based on complex data with missing values and robust principal component analysis,» Tesis doct., Ghent University, 2016, págs. XV, 157. dirección: http://hdl.handle.net/1854/LU-8502386.

[26] J. O. Ramsay y B. W. Silverman, «Functional Data Analysis,» Springer Series in Statistics, 2005. DOI: 10.1007/B98888.

[27] P. Kokoszka y M. Reimherr, *Introduction to functional data analysis* (Texts in Statistical Science), eng, M. Reimherr, ed. Boca Raton: CRC Press, 2017, pág. 290, ISBN: 978-1-498-74634-2.

[28] J. Jacques y C. Preda, «Functional data clustering: A survey,» *Advances in Data Analysis and Classification*, vol. 8, n.º 3, págs. 231-255, 2014. DOI: 10.1007/S11634-013-0158-Y/TABLES/5.

[29] D. B. Hitchcock y M. C. Greenwood, «Clustering Functional Data,» en *Handbook of Cluster Analysis*, 1st ed., Chapman y Hall/CRC, 2015, págs. 286-309, ISBN: 9780429185472. DOI: 10.1201/B19706-19.

[30] C. Yassouridis y F. Leisch, «Benchmarking different clustering algorithms on functional data,» *Advances in Data Analysis and Classification*, vol. 11, n.º 3, págs. 467-492, 2017. DOI: 10.1007/S11634-016-0261-Y.

[31] W. S. Cleveland, «Robust locally weighted regression and smoothing scatterplots,» *Journal of the American Statistical Association*, vol. 74, n.º 368, págs. 829-836, 1979. DOI: 10.1080/01621459.1979.10481038.

[32] K. You, *CRAN - Package T4cluster*. dirección: https://cran.r-project.org/web/packages/T4cluster/index.html (visitado 08-06-2022).

[33] P. J. Rousseeuw y W. V. D. Bossche, «Detecting Deviating Data Cells,» *Technometrics*, vol. 60, n.º 2, págs. 135-145, 2017. DOI: 10.1080/00401706.2017.1340909. arXiv: 1601.07251.

[34] A. Alfons, «robustHD: An R package for robust regression with high-dimensional data,» *Journal of Open Source Software*, vol. 6, n.º 67, pág. 3786, 2021. DOI: 10.21105/joss.03786.

[35] M. Maechler, P. Rousseeuw, C. Croux, V. Todorov, A. Ruckstuhl, M. Salibian-Barrera, T. Verbeke, M. Koller, E. L. T. Conceicao y M. Anna di Palma, *robustbase: Basic Robust Statistics*, 2022. dirección: http://robustbase.r-forge.r-project.org/.

[36] K. Ram, *CRAN - Package wesanderson*. dirección: https://cran.r-project.org/web/packages/wesanderson/ (visitado 08-06-2022).

[37] P. J. Rousseeuw y K. Van Driessen, «Computing LTS regression for large data sets,» *Data Mining and Knowledge Discovery*, vol. 12, n.º 1, págs. 29-45, 2006. DOI: 10.1007/S10618-005-0024-4.

[38] R. J. Hyndman, *CRAN - Package demography*. dirección: https://cran.r-project.org/web/packages/demography/index.html (visitado 15-06-2022).

[39] D. Hernangómez, *mapSpain: Administrative Boundaries of Spain*, 2022. DOI: 10.5281/zenodo.5366622. dirección: https://ropenspain.github.io/mapSpain/.