Herramientas del BIG DATA y analytics

Una introducción a los modelos de Clustering empleando R

Autores:
Julio César Alonso
Maria Fernanda Largo
Cristian Camilo Hoyos

Universidad Icesi | Editorial Universidad Icesi

Una introducción a los modelos de Clústering empleando R

Julio César Alonso C.[1] Cristian Camilo Hoyos B.[2]

María Fernanda Largo L.[3]

2025-03-26

[1]Departamento de Economía - Universidad Icesi, jcalonso@icesi.edu.co
[2]Departamento de Economía - Universidad Icesi
[3]Cienfi - Universidad Icesi

Julio César Alonso C. - Cristian Camilo Hoyos B. - María Fernanda Largo L.

Colección «Herramientas del Big Data y Analytics», vol. 6

Cali. Universidad Icesi, 2024.

193 páginas.

Incluye referencias bibliográficas.

ISBN: 978-628-7740-99-0 (eBook).

DOI: https://doi.org/10.18046/EUI/bda.h.6

Palabras Clave: 1. R | 2. Analítica | 3. Modelos de clústering | 4. Modelos de agrupamiento | 5. Big Data Analytics

Clasificación Dewey: 545 ddc 21

CIENFI - Centro de Investigación en Economía y Finanzas

www.icesi.edu.co/centros-academicos/cienfi

Rector: Esteban Piedrahita Uribe

Secretaria General: Olga Patricia Ramírez Restrepo

Director Académico: José Hernando Bahamón

Coordinador editorial: Adolfo A. Abadía

Corrección de estilo: Elizabeth Vejarano Soto

Diseño de portada: Sandra Moreno

Fotos tomadas por: Julio César Alonso

Editorial Universidad Icesi

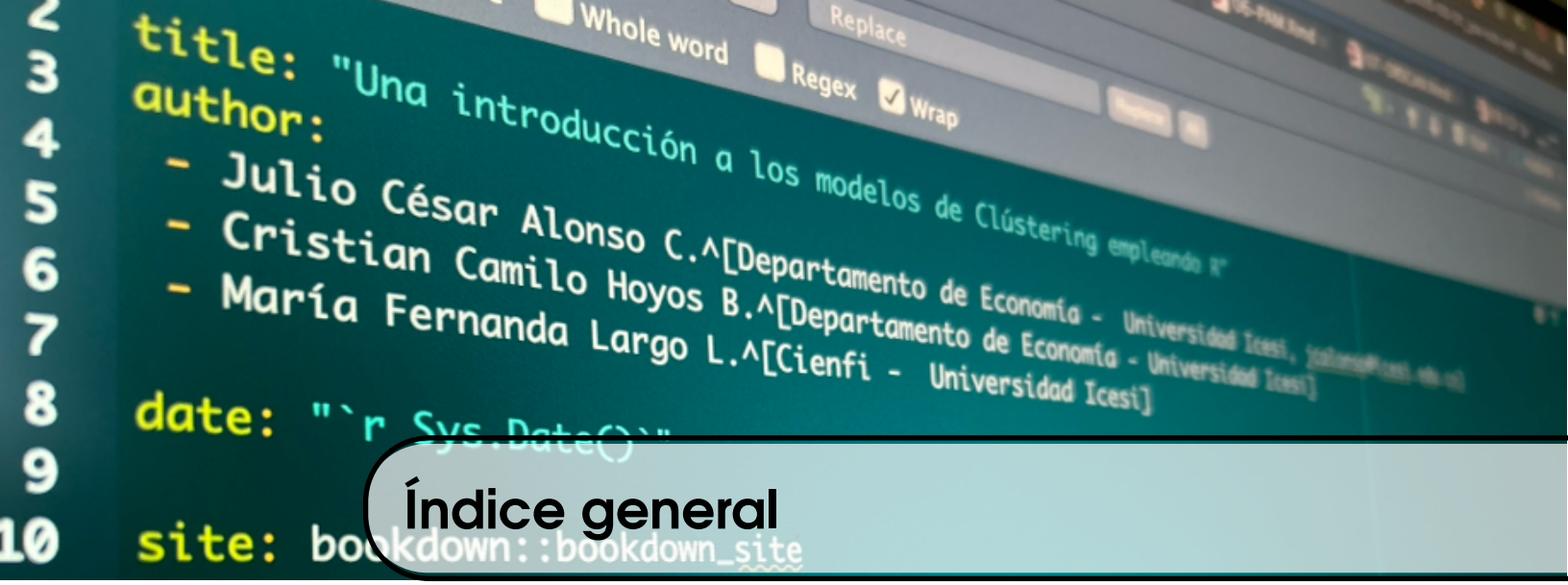
Calle 18 No. 122-135 (Pance), Cali – Colombia

Teléfono: +57 (2) 555 2334 | E-mail: editorial@icesi.edu.co

http://www.icesi.edu.co/editorial

Publicado en Colombia – *Published in Colombia*

La publicación de este libro se aprobó luego de superar un proceso de evaluación doble ciego.



Índice general

Prefacio **13**

I Conceptos fundamentales **15**

1 Introducción **17**

Objetivos del capítulo 17

1.1 Introducción 18

1.2 Comentarios Finales 23

2 Generalidades y métricas en el clústering **25**

Objetivos del capítulo 25

2.1 La intuición detrás de los algoritmos de clústering 26

2.2 Medidas de similitud 26

2.3 Algoritmos para la formación de clústeres 32

2.4 Criterio para determinar el número de clústeres 37

2.5 Comentarios finales 44

2.6 Anexo: Índices de validación 44

II Algoritmos jerárquicos **57**

3 Clústering Jerárquico **59**

Objetivos del capítulo 59

3.1 Introducción 60

3.2 La intuición de los métodos aglomerativos 60

3.3 Métodos de aglomeración 69

3.4 La intuición de los métodos de división 72

3.5 Comentarios finales 75

4 Implementando los algoritmos de clústering jerárquico en R . . **79**

Objetivos del capítulo 79

4.1 Introducción 80

4.2 Los datos y la pregunta de negocio 80

4.3 Exploración y preparación de los datos 81

4.4 Construcción de clústeres jerárquicos aglomerativos y dendrograma 84

4.5 Escogiendo el número óptimo de clústeres 87

4.6 Construcción de clústeres jerárquicos de división 94

4.7 Visualización y analisis de resultados 97

4.8 Comentarios finales 107

4.9 Anexos 107

III Algoritmos basados en centroides **113**

5 Modelo k-means **115**

Objetivos del capítulo 115

5.1 Introducción 116

5.2 La intuición 116

5.3 Detalle técnico del algoritmo k-means 120

5.4 k-means en R 120

5.5 k-means++ 127

5.6 Comentarios Finales 129

6 Modelos PAM y CLARA 131

Objetivos del capítulo 131

6.1 Introducción 132

6.2 El modelo PAM 132

6.3 El modelo CLARA 133

6.4 Implementación de PAM y CLARA en R 134

6.5 Calculando siluetas individuales y membrecías para los algoritmos PAM y CLARA 138

6.6 Comentarios Finales 141

IV Algoritmos basados en densidad 143

7 Modelo DBSCAN 145

Objetivos del capítulo 145

7.1 Introducción 146

7.2 El algoritmo DBSCAN 147

7.3 Implementación del algoritmo DBSCAN en R 152

7.4 Otro ejemplo 155

7.5 Comentarios finales 158

V Algoritmos basados en distribuciones 161

8 Modelo GMM 163

Objetivos del capítulo 163

8.1 Introducción 164

8.2 Formalmente 164

8.3 Implementación del modelo GMM 167

8.4 Otro ejemplo 171

8.5 Comentarios finales 172

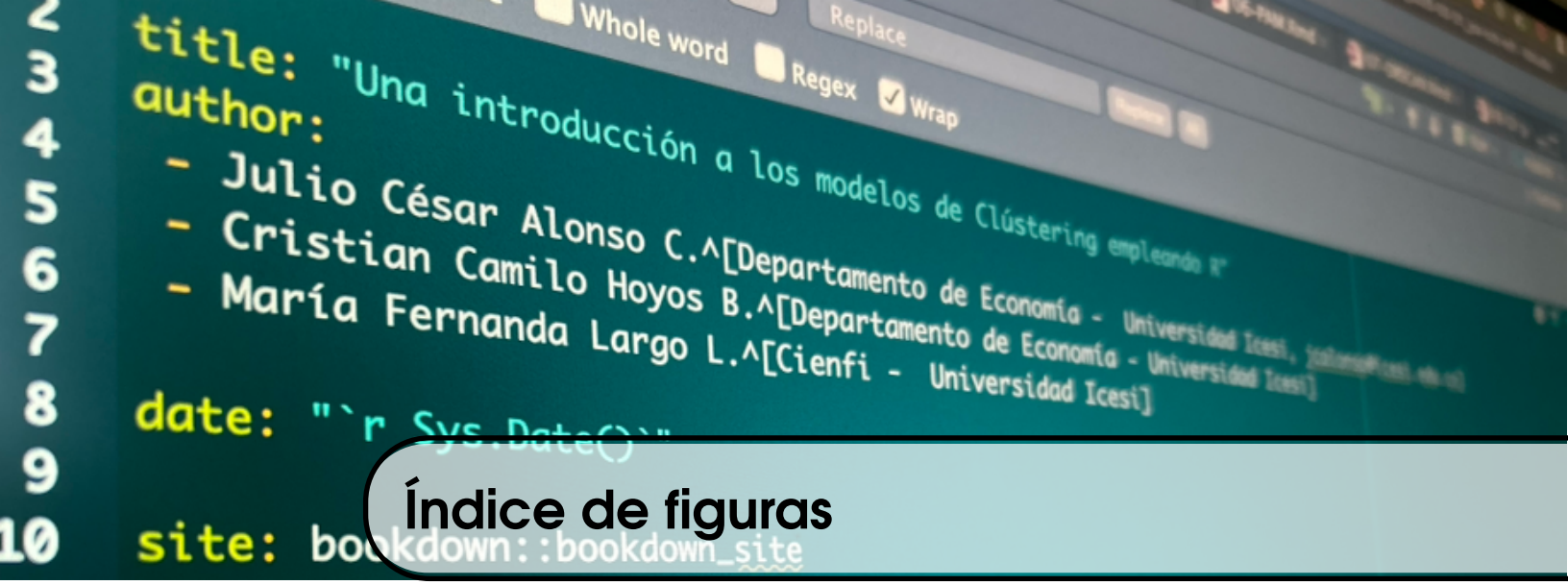
VI Algoritmos combinados 173

9 Modelo FANNY 175
Objetivos del capítulo 175
9.1 Introducción 176
9.2 El algoritmo FANNY 176
9.3 Implementación de FANNY en R 177
9.4 Comentarios finales 179

VII Referencias 181

Referencias 187

Índice alfabético 189



Índice de figuras

1.1 Tarea de clústering 19
1.2 Material multimedia: tarea de clústering 19
1.3 Material multimedia: tipos de analítica 21
1.4 Relación entre las tareas de analítica y los tipos de analítica 22

2.1 El problema de la construcción de clústeres 27
2.2 Representación de las distancias entre dos observaciones con dos variables 30
2.3 Clústeres jerárquicos 32
2.4 Clústeres particionados 33
2.5 Clústeres particionados basados en centroides 34
2.6 Clústeres particionados basados en densidad 35
2.7 Clústeres particionados basados en distribuciones 36
2.8 Representación de la selección del número de clústeres óptimo con el método del codo 39
2.9 Ejemplo de la selección del número de clústeres óptimo empleando el estadístico GAP 40
2.10 Representación de la distancia intraclúster para un individuo i 41
2.11 Representación de la distancia interclúster para un individuo i 43

3.1 Dendrograma para el clúster aglomerativo- Paso 1 63
3.2 Dendrograma para el clúster aglomerativo- Paso 2 65
3.3 Dendrograma para el clúster aglomerativo- Paso 3 66
3.4 Dendrograma para el clúster aglomerativo- Paso 4 68
3.5 Representación esquemática de los métodos de aglomeración 70
3.6 Pasos del ejemplo empleando el algoritmo aglomerativo (AGNES) 76
3.7 Pasos del ejemplo empleando el algoritmo de división (DIANA) 77

4.1 Relación entre todas las variables de la base de datos 82
4.2 Dedrograma para el HCA emplendo el método de aglomeración de centroide y distancia euclidiana 85

4.3 Otra versión del dedrograma para el HCA emplendo el método de aglomeración de centroide y distancia euclidiana 86
4.4 Votación de la batería de métricas por el número óptimo de clústeres para el HCA empleando el método de aglomeración de centroide y distancia euclidiana 90
4.5 Silueta para el HCA con dos clústeres emplendo el método de aglomeración de centroide y distancia euclidiana 93
4.6 Dedrograma para DIANA y distancia euclidiana 95
4.7 Dedrograma circular para DIANA, distancia euclidiana y cortando el árbol en 3 clústeres 96
4.8 Relación entre el límite de crédito promedio y el número de tarjetas de crédito por clúster 98
4.9 Distribución de las variables empleadas para construir las agrupaciones por clúster 99
4.10 Dendrograma método del centroide 100
4.11 Dendrograma método del centroide con el paquete ape 102
4.12 Dendrograma método del centroide con colores en las etiquetas 102
4.13 Dendrograma método del centroide tipo ucladogram 102
4.14 Dendrograma método del centroide tipo unrooted 103
4.15 Dendrograma método del centroide tipo fan 104
4.16 Dendrograma método del centroide horizontal 104
4.17 Dendrograma método del centroide 105
4.18 Dendrograma método del centroide 106
4.19 q-q plot 111

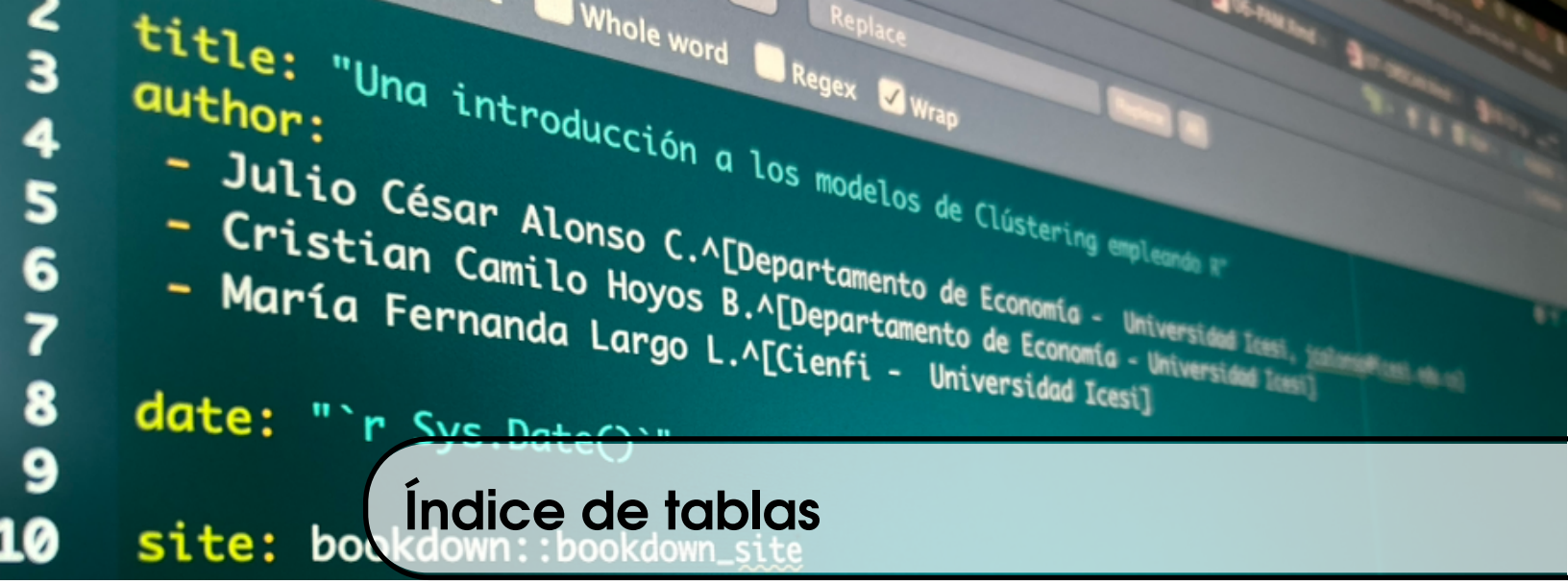
5.1 Representación de un algoritmo de clústering empleando k-means 117
5.2 Algoritmo k-means 119
5.3 Votación de la batería de métricas por el número óptimo de clústeres para el algoritmo k-means y distancia euclidiana 123
5.4 Silueta promedio para diferente número de clústeres para el algoritmo k-means y distancia euclidiana 124
5.5 Silueta para tres clústeres emplendo k-means y distancia euclidiana 126
5.6 Centroides inicializados incorrectamente (al azar) 128

6.1 Silueta promedio por número de clúster para el algoritmo PAM y distancia euclidiana 137
6.2 Silueta promedio por número de clúster para el algoritmo CLARA y distancia euclidiana 138
6.3 Silueta para tres clústeres emplendo PAM y distancia euclidiana 140

7.1 Clústeres particionados basados en densidad 146
7.2 Ejemplo 1 de los diferentes clústeres que pueden formar DBSCAN y k-means apartir de la misma muestra de individuos con dos características . . 148
7.3 Ejemplo 2 de los diferentes clústeres que pueden formar DBSCAN y k-means a partir de la misma muestra de individuos con dos características . . 149
7.4 Algoritmo DBSCAN detectanto outliers 150
7.5 Clasificación de individuos en el algoritmo DBSCAN con minPts = 6 151

7.6 Seleccionando el valor óptimo del parámetro eps por medio de la distancia kNN 154
7.7 Datos del ejemplo 2 156
7.8 Partición de los datos del ejemplo 2 empleando k-means 157
7.9 Partición de los datos del ejemplo 2 empleando DBSCAN 157

8.1 Representación de un algoritmo de clústering particionado basado en distribuciones 165
8.2 Clústeres encontrados con el algoritmo GMM 168
8.3 Distribuciones de probabilidad por clúster encontradas con el algoritmo GMM169
8.4 Partición de los datos del ejemplo 2 empleando GMM 171



Índice de tablas

3.1 Compras de los clientes . . . 61
3.2 Matriz de proximidad . . . 62
3.3 Compras de los clientes actualizadas - Paso 2 . . . 64
3.4 Matriz de proximidad - Paso 2 . . . 64
3.5 Compras de los clientes actualizadas - Paso 3 . . . 64
3.6 Matriz de proximidad - Paso 3 . . . 67
3.7 Compras de los clientes actualizadas - Paso 4 . . . 67
3.8 Matriz de proximidad - Paso 4 . . . 67
3.9 Paso O del algortimo DIANA . . . 73
3.10 Paso O del algortimo DIANA: encontrando más candidatos para hacer parte de C_1 . . . 73
3.11 Matriz de proximidad para el clúster C_1 . . . 74
3.12 Matriz de proximidad para el clúster C_2 . . . 74
3.13 Paso 1 del algortimo DIANA: escogiendo el cliente diferente . . . 74
3.14 Paso 1 del algortimo DIANA: encontrando más clientes para unirse al C_3 . . . 75

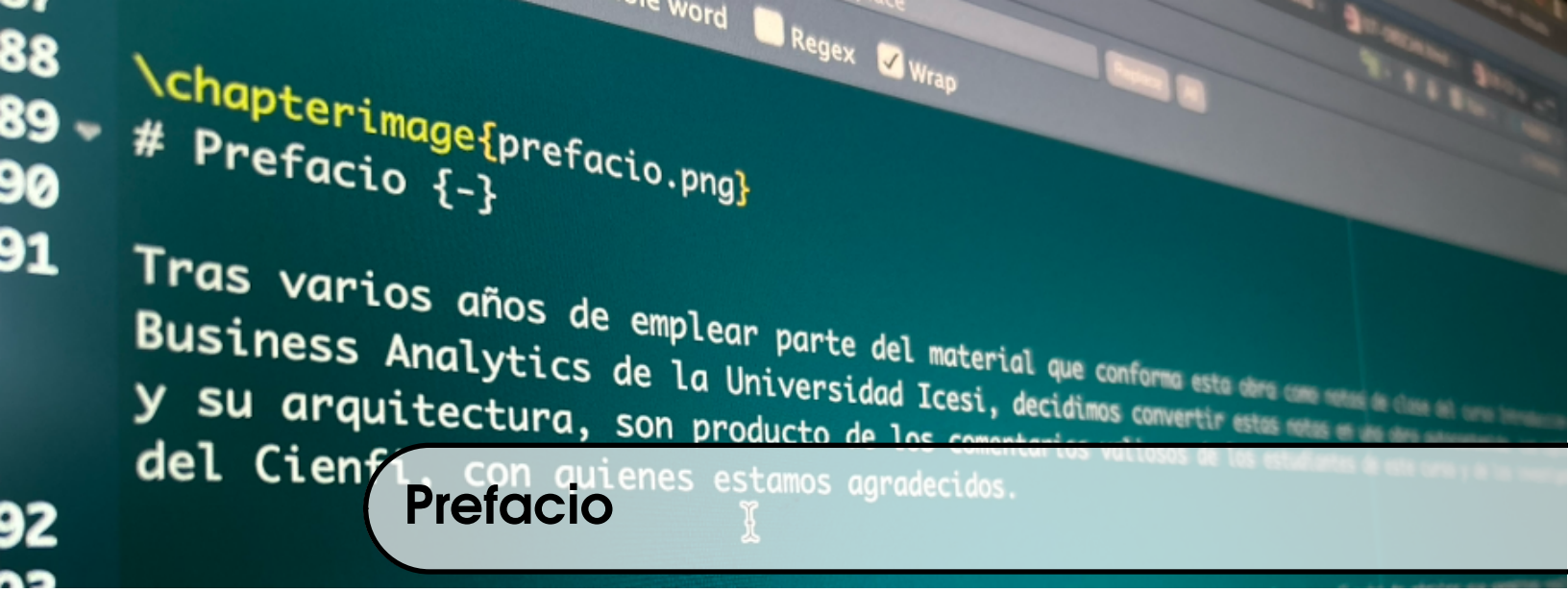
4.1 Siluetas y número de clústeres óptimos para HCA con diferentes métodos de aglomeración . . . 94

5.1 Siluetas y número de clústeres óptimos para diferentes aproximaciones de clústering (distancia euclidiana) . . . 124

6.1 Siluetas y número de clústeres óptimos para diferentes aproximaciones de clústering (distancia euclidiana) . . . 137

8.1 Siluetas y número de clústeres óptimos para diferentes aproximaciones de clústering (distancia euclidiana) . . . 170

9.1 Siluetas y número de clústeres óptimos para diferentes aproximaciones de clústering (distancia euclidiana) . . . 178



Prefacio

Tras varios años de emplear parte del material que conforma esta obra como notas de clase del curso Introducción al Business Analytics de la Universidad Icesi, decidimos convertir estas notas en una obra autocontenida. Los capítulos y su arquitectura, son producto de los comentarios valiosos de los estudiantes de este curso y de los investigadores del Cienfi, con quienes estamos agradecidos.

Este libro presenta una introducción a los modelos estadísticos y de aprendizaje de máquina que permiten realizar la tarea de clústering. La discusión de los diferentes capítulos está dirigida a personas que empiezan su formación como científico de datos.

El libro está dividido en tres partes. La primera parte corresponde a una discusión de los conceptos fundamentales para entender la tarea de clústering. En el Capítulo 1 se discute las generalidades de la tarea de clústering y cómo ésta está relacionada con los tipos de analítica. En el Capítulo 2 se discuten los conceptos de medidas de similitud, los diferentes tipos de algoritmos para la formación de clústeres y la métricas para la selección del número de clústeres.

La segunda parte del libro corresponde al estudio de los algoritmos jerárquicos para la construcción de clústeres. En el Capítulo 3 se discute la intuición detrás de los modelos de **clústering jerárquico aglomerativo** conocidos también como modelos HAC (del inglés *Hierarchical Agglomerative Clustering*), AGNES (del inglés *AGlomerative NESting*) o aproximación *bottom-up*. Ese capítulo también discute el **clústering jerárquico de división**, también conocidos como algoritmos *top-down* o DIANA (del inglés *DIvise ANAlysis*). En el Capítulo 4 ese discute como implementar estos algoritmos en R.

La tercera parte del libro discute los algoritmos **particionados** basados en centroides. En el Capítulo 5 se presenta el modelo **k-means** y una modificación de este conocido como **k-means++**. En el Capítulo 6) se estudian el algoritmo **k-medoids** también es conocido como **PAM** (por su sigla en inglés del término *Partitioning Around Medoids*) y el modelo **CLARA** (por su sigla en inglés del término *Clustering Large Applications*).

La cuarta parte del libro está destinada a los algoritmos basados en densidad. El Capítulo 7 explica el algoritmo **DBSCAN**. En la quinta parte se estudian los métodos **basados en la distribución**, también conocidos como **basados en modelos**. Para este tipo de algoritmos se presenta el Modelo de Mezcla Gausiano (**GMM**) en el Capítulo 8. Y finalmente en la sexta parte del libro se discute uno de los **métodos que combinan o modifican los anteriores** como el *fuzzy clustering*, también conocido como **FANNY**(Ver Capítulo 9). De esta manera este libro, si bien introductorio, te podrá brindar una mirada amplia de los modelos y filosofias diferentes para la construcción de clústeres con datos de corte transversal.

Este libro supone un uso intermedio de R (R Core Team, 2023). Si crees que necesitas algún refuerzo en R, te recomendamos tres libros. Alonso y Ocampo (2022) presenta una breve introducción para iniciar a usar R. Ese primer libro discute cómo instalar R y RStudio y paquetes, cómo cargar diferentes bases de datos y cómo realizar operaciones aritméticas y lógicas con objetos. En Alonso y Ocampo (2022) también se discuten las clases esenciales de objetos sencillos y compuestos. No dudes en consultar ese primer libro si aún no has iniciado tu camino por el universo de R.

El segundo libro de la serie, Alonso (2022), presenta una breve introducción al paquete *dplyr* (Wickham et al., 2022) que permite manipular objetos que contengan datos. En ese libro se discute cómo filtrar observaciones, crear nuevas variables y combinar objetos con datos. Es recomendable tener un conocimiento de ese paquete antes de leer esta obra. Consulta ese segundo libro si aún no has tenido alguna experiencia manipulando objetos con datos con *dplyr*.

Finalmente, recomendamos Alonso y Largo (2023) en el que se presenta una introducción a la creación de visualizaciones con el paquete *ggplot2* (Wickham, 2016). En esta obra emplearemos visualizaciones empleando este paquete. Así, este libro asume un manejo intermedio de R, y los paquetes *dplyr* y *ggplot2*.

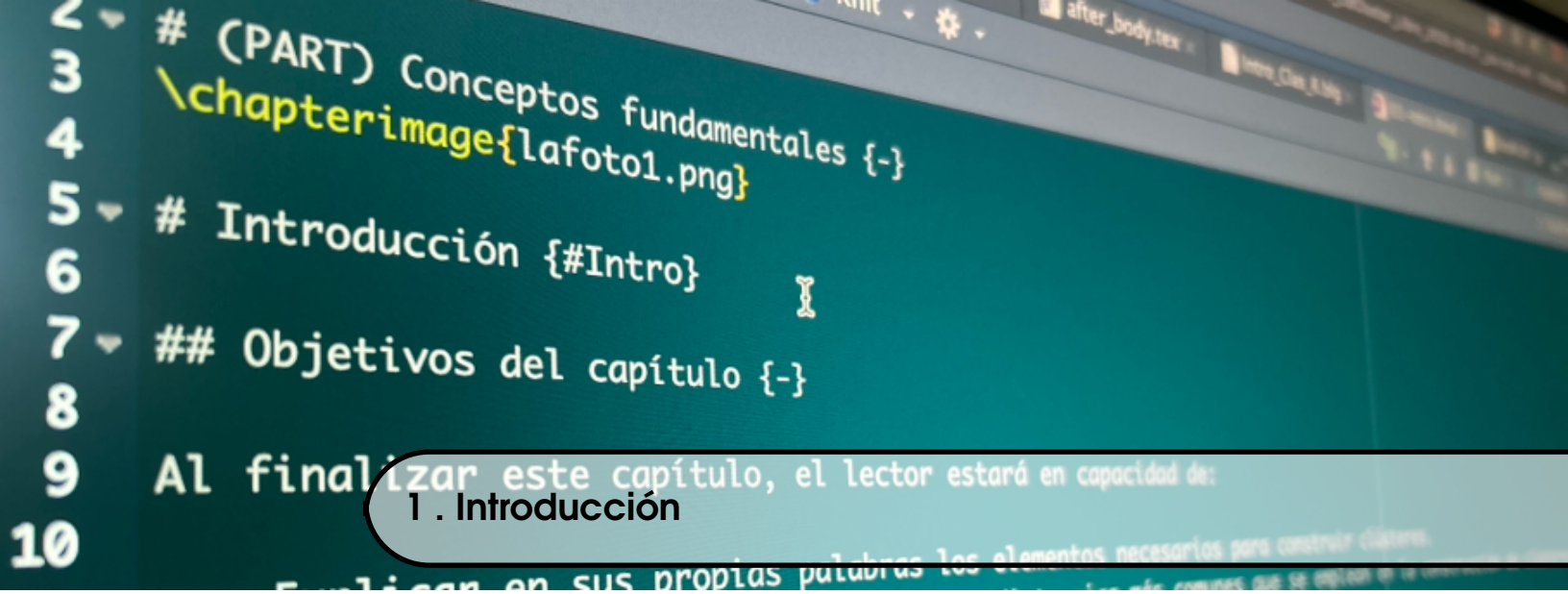
Por otro lado, un manejo del modelo de regresión múltiple conceptualmente y en R es deseable. Alonso (2024) te puede brindar una introducción a la fundamentación formal del modelo de regresión y cómo estimar estos modelos y chequear sus supuestos en R. Si te interesa aprender de otras tareas de analítica puedes encontrar útiles otros libros de la serie. Por ejemplo, Alonso y Arboleda (2025) presenta una introducción a la tarea de encontrar reglas de asociación entre productos (análisis de canastas) y Alonso y Hoyos (2025a) presenta una introducción a la tarea de clasificación.

La presente obra recoge nuestra experiencia trabajando con R y los modelos de clasificación para resolver problemas con datos desde el Cienfi (Centro de Investigación en Economía y Finanzas) de la Universidad Icesi. En el Cienfi empleamos R para la transformación de datos en conclusiones que faciliten la toma de decisiones en organizaciones privadas y públicas.

¡Esperamos encuentres esta obra útil y la compartas con otros! Si tienes alguna sugerencia o comentario, no dudes en escribirnos. Esta es una obra en constante construcción.

Parte I

Conceptos fundamentales



1 . Introducción

Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Explicar en sus propias palabras los elementos necesarios para construir clústeres.
- Explicar en sus propias palabras las distancias más comunes que se emplean en la construcción de clústeres.

1.1 Introducción

El *Business Analitycs* es el proceso científico de transformar datos en *insights* (conclusiones) con el propósito de tomar mejores decisiones. El *Business Analitycs* recoge un conjunto de herramientas de la estadística y de inteligencia artificial que nos permiten emplear datos para responder una pregunta de negocio planteada. Las técnicas se pueden clasificar de acuerdo con la finalidad de los cálculos que se realizan, en las siguientes tareas:

- *Clasificar*[1].
- *Estimar regresiones*[2].
- *Detectar anomalías*.
- **Clusterizar (Agrupar)**.
- *Encontrar reglas de asociación* (*o buscar coocurrencia de productos*)[3]
- *Pronosticar*
- *Resumir*
- *Visualizar*[4]

La tarea de *clusterizar* (clústering o agrupar) implica encontrar grupos de elementos, lo más similares posibles, al mismo tiempo que los grupos son lo más diferente posibles entre ellos. La Figura 1.1 muestra de manera esquemática esta tarea. Partiendo de una muestra de individuos que no tienen ninguna "marca" (**label** en inglés) que los distinga o los asigne a un grupo, el modelo de clústering generará grupos de tal manera que los individuos al interior del grupo son muy parecidos, pero entre grupos muy diferentes.

Para realizar las diferentes tareas de analítica empleamos modelos o algoritmos[5]. En el ejemplo de la Figura 1.1, el modelo de clústering encuentra dos grupos: los que tienen gafas y los que no. A los grupos o conjuntos de individuos se les puede denominar de manera intercambiable como: conglomerados, clases, grupos o clústeres.

La creación de grupos (clústering) es bastante útil en el mundo de los negocios. Por ejemplo, en el mercadeo es una herramienta vital para la construcción de la estrategia y las decisiones tácticas. Supongamos que se tienen mil clientes, lo que hace muy difícil gerenciar a cada cliente de manera diferente. Pero cuando se arman grupos de clientes (clústeres de clientes) esto permite concentrarse en ellos de manera diferente. Dicho de otra forma, esto permite entender cómo se comportan los miembros de la población empleando conjuntos de individuos similares, tarea que es mucho más sencilla que intentar entender el comportamiento individual. A esto se le conoce como **segmentación de clientes**.

[1] Para una discusión detallada de esta tarea y como implementarla en R se puede consular Alonso y Hoyos (2025b)

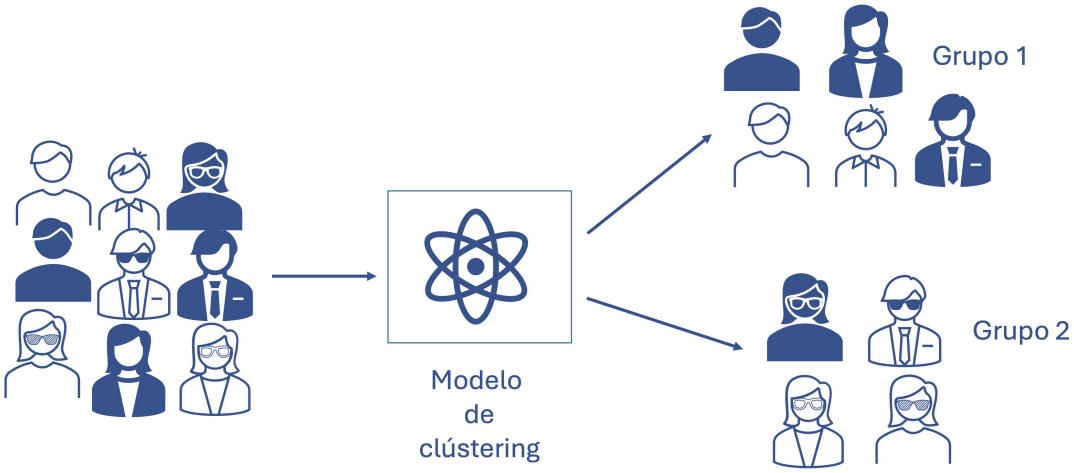
[2] Para una discusión detallada del modelo clásico de regresión y como implementarlo en R se puede consular Alonso (2024).

[3] Para una discusión detallada de esta tarea y como implementarla en R se puede consular Alonso y Arboleda (2025).

[4] Para una discusión detallada de esta tarea y como implementarla en R se puede consular Alonso (2022)

[5] Los modelos o algoritmos en algunos casos pueden ser útiles para hacer más de una tarea de analítica, como veremos más adelante. Por eso es importante distinguir entre la tarea de analítica y los modelos y algoritmos.

Figura 1.1. Tarea de clústering



Fuente: elaboración propia.

Figura 1.2. Material multimedia: tarea de clústering

Fuente: elaboración propia. https://youtu.be/z0LX3sBSuXg

A través del clústering, las organizaciones pueden optimizar la calidad de los mensajes que envían al público, como promociones de productos con mayor poder de adquisición o un servicio posventa acorde con la última compra. Esto refuerza la relación con los clientes y, en consecuencia, aumenta las ventas y la fidelidad con las marcas.

Otra aplicación del clústering la podemos encontrar en el sector de venta al por menor y en el comercio electrónico. En estos sectores un reto importante es ser asertivo en la distribución de los productos. El clústering puede ayudar a resolver este reto: empleando datos de los puntos de ventas o zonas[6] es posible perfilar qué tipo de productos se venderán mejor en cada agrupación de puntos de venta. Esto permite que la logística realice un despacho personalizado a cada tienda para maximizar las ventas.

El clústering también puede ser empleado para asignar de una manera eficiente un factor importante en un proceso productivo. Por ejemplo, en el sector salud, el clústering puede utilizarse para identificar grupos de pacientes con síntomas o afecciones similares. Esta información puede ayudar a los hospitales a asignar el personal médico, los equipos y los planes de tratamiento, garantizando que los recursos se utilicen al máximo de su potencial. Las aplicaciones son muchas.

Por otro lado, aunque suene raro, es importante notar que los modelos de clústering también pueden servir para realizar la tarea de detección de anomalías y en especial la detección de fraudes. Las técnicas de clústering pueden identificar patrones anómalos en grandes conjuntos de datos transaccionales. En este contexto, los algoritmos de clústering pueden identificar grupos de transacciones que se desvían del comportamiento "normal", ayudando a las instituciones financieras y a las agencias de seguridad a detectar posibles intentos de fraude y a tomar las medidas oportunas. Por ejemplo, una empresa de tarjetas de crédito puede utilizar el clústering para identificar grupos de transacciones con patrones sospechosos, como compras de alto valor en diferentes lugares dentro de un período corto. Esta información permite a la empresa señalar posibles actividades fraudulentas y proteger los intereses financieros de sus clientes.

Los modelos de clústering se estiman o entrenan empleando una muestra de individuos[7] para los cuales se cuenta con observaciones de algunas de sus características. Este tipo de tarea no tiene una variable a explicar e implica que el modelo o algoritmo aprenda sobre la estructura de los datos que caracterizan a los individuos. Así, estos modelos corresponden a lo que se conoce como aprendizaje no supervisado[8], pues la muestra no contiene una variable objetivo que se encuentra etiquetada con la

[6] Como por ejemplo de ingreso promedio de la región, el clima, los hábitos de consumo y la edad.

[7] Típicamente para los modelos de clústering se emplean muestras de corte transversal. Es decir, a muchos individuos se les observa sus características y la variable objetivo en el mismo período. En este libro concentraremos la atención en algoritmos de clústering que emplean datos de corte transversal. No obstante, es importante anotar que existen algoritmos de clústering que se emplean sobre series de tiempo de diferentes individuos; es decir, sobre datos de panel. Este tipo de clústering está por fuera del alcance de este libro.

[8] En el campo del aprendizaje de máquina se distinguen dos tipos de aprendizaje: supervisado y no supervisado. En el aprendizaje no supervisado los datos no contienen etiquetas o la "respuesta correcta". El aprendizaje no supervisado busca descubrir patrones o estructuras ocultas en los datos.

pertenencia o no de una observación a un grupo[9]. La misión del modelo o algoritmo es aprender cuál es el patrón que permitiría armar los grupos de individuos, dada las características de estos.

Regresando a las tareas del *business analytics*, una manera más común de clasificar los ejercicios de analítica es según su propósito. En ese caso, se tienen cuatro tipos de analítica (Ver Figura 1.3):

- **Analítica Descriptiva**: Esta analítica se enfoca en resumir y visualizar los datos para obtener información sobre lo que ha sucedido en el pasado. Ayuda a comprender patrones y tendencias.
- **Analítica Diagnóstica**: Esta analítica busca entender por qué algo ha sucedido. Examina los datos para identificar las causas raíz de los problemas o éxitos pasados.
- **Analítica Predictiva**: Esta analítica utiliza modelos estadísticos y de aprendizaje de máquina para hacer pronósticos y predecir eventos futuros.
- **Analítica Prescriptiva**: Esta analítica se centra en recomendar acciones y soluciones óptimas para lograr un objetivo.

Figura 1.3. Material multimedia: tipos de analítica

Fuente: elaboración propia. https://rb.gy/s7efwr

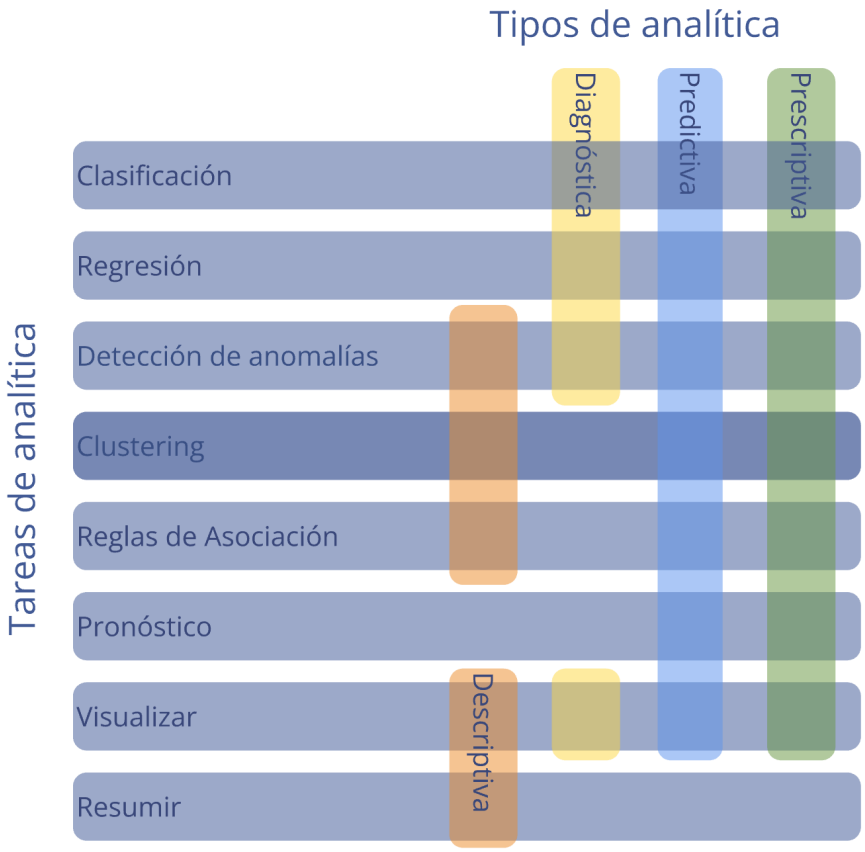
Estos cuatro tipos de analítica engloban las ocho tareas del *business analytics*. La tarea de clústering permite realizar analítica descriptiva y prescriptiva (Ver Figura 1.4).

La **analítica descriptiva** responde a la pregunta: **¿qué está pasando en el negocio?**[10]. El análisis de clústering permite describir los patrones (la estructura) de los datos identificando grupos de elementos similares entre sí. De hecho, los métodos o algoritmos que nos permiten construir los clústeres se consideran como métodos exploratorios

[9] Es decir, la respuesta "correcta" de pertenecer a un grupo no es conocida.

[10] Para ver una introducción rápida al tipo de datos que se emplean en el *business analytics* ver el video disponible en el siguiente enlace: https://youtu.be/2OxY2UTI_Bs.

Figura 1.4. Relación entre las tareas de analítica y los tipos de analítica



Fuente: elaboración propia.

de datos, pues nos permiten entender características de un conjunto de estos[11].

La **analítica predictiva** busca responder la pregunta: **¿qué es posible que ocurra?** Los modelos de clústering podrán ser empleados para determinar a qué grupo, probablemente, pertenecería un nuevo individuo, dadas sus características. Por ejemplo, una vez que se han identificado grupos de clientes con comportamientos similares, se puede aplicar un patrón identificado en los datos para predecir, usando las características del nuevo cliente y el modelo, el grupo al que pertenecería ese nuevo individuo.

La **analítica prescriptiva** busca responder la pregunta **¿qué necesito hacer?** Los modelos de clústering nos permiten hacer este tipo de analítica al sugerir qué deberíamos hacer en algunas situaciones. Por ejemplo, si se requiere agrupar los clientes para una segmentación de consumidores, este análisis nos podrá sugerir cuántos grupos emplear y cómo asignar a cada cliente a su respectivo grupo.

1.2 Comentarios Finales

Las técnicas de construcción de clústeres pueden responder, por sí solas, una pregunta de negocio o pueden hacer parte de la exploración de los datos antes de desarrollar modelos complejos y probar distintas hipótesis. Estos métodos de clústering no incluyen ninguna teoría detrás, al tiempo que permiten que las observaciones (o casos) sean agrupados según la estructura propia sin necesidad de establecer ninguna variable dependiente y variables explicativas.

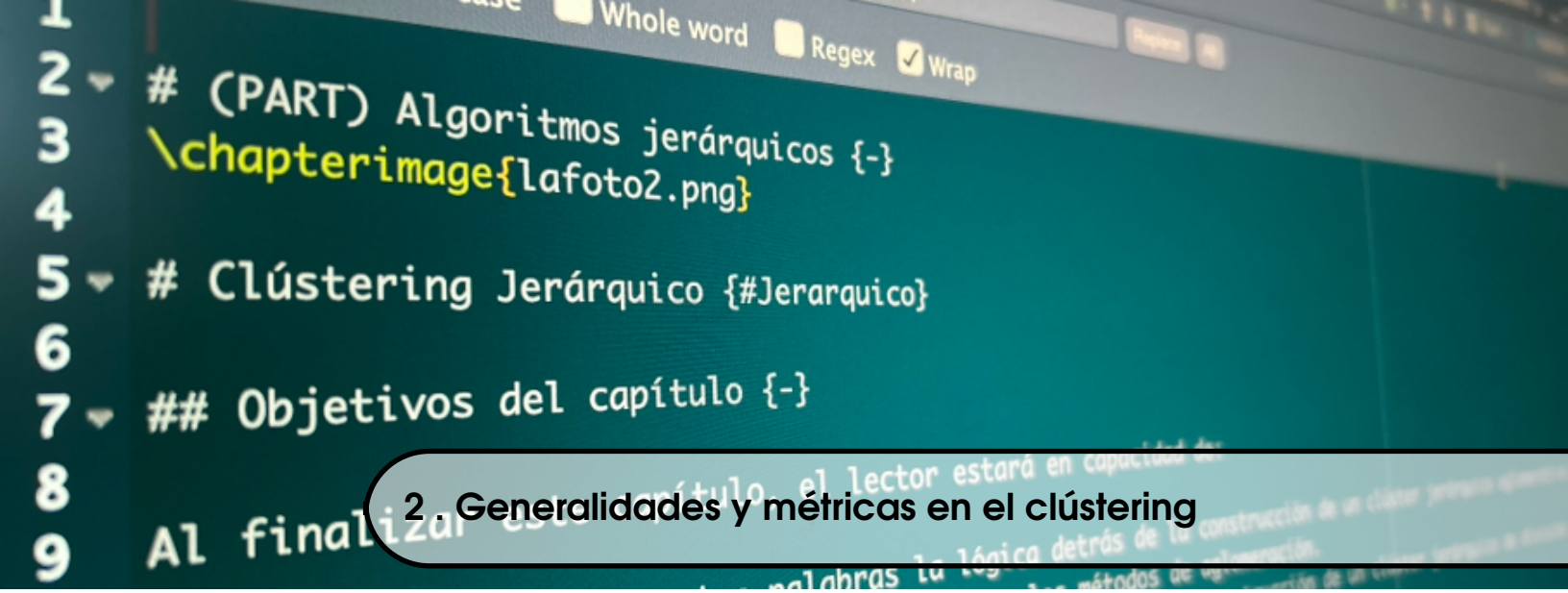
Precisamente, la característica de las técnicas de clústering de dejar que los datos se agrupen por si solos, presenta uno de los retos más importantes. Estamos frente a una tarea en la que queremos detectar cuáles son las agrupaciones "naturales" de las observaciones, pero no tenemos cómo observar cuáles serían los "verdaderos" grupos.

Para aclarar este punto, consideremos una tarea como la de pronosticar. Esa tarea implica estimar o entrenar un método para que se acerque a lo que ocurrió históricamente y este aprendizaje nos permite generar pronósticos. Pronósticos que podemos comparar con lo que realmente ocurra en el futuro y así establecer el error del pronóstico. Este tipo de tareas se conocen como aprendizaje supervisado.

Por otro lado, en la tarea de clústering tenemos un conjunto de datos para diferentes individuos y debemos construir los grupos sin tener un valor real que nos permita entrenar o estimar y que haga posible realizar una evaluación de los grupos formados frente a un valor "real" de pertenencia a un grupo. Este tipo de tareas se conoce como aprendizaje no supervisado. En este tipo de aprendizaje dejamos que la técnica determine los patrones de datos por sí misma. Las técnicas de clústering generan una variable (o característica) que no está en los datos originales observados que es la membresía a un grupo, mientras que los métodos de aprendizaje supervisado están diseñados para "generar" una variable que ya estaba en los datos originales.

[11] Las técnicas de construcción de clústeres también se pueden considerar como otra técnica de reducción de datos. Noten que en el caso de métodos de reducción de variables como el Análisis de Componentes Principales (**PCA** por su sigla en inglés) se generan grupos de variables, pero aquí los grupos se forman con los individuos. Por eso se puede considerar como una técnica de reducción de datos.

En el Capítulo 2 estudiaremos la intuición detrás de las técnicas que generan clústeres y qué medidas podemos emplear para determinar si las agrupaciones que construimos son razonables.



Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Explicar en sus propias palabras los elementos necesarios para construir clústeres.
- Explicar en sus propias palabras las distancias más comunes que se emplean en la construcción de conglomerados.
- Explicar en sus propias palabras la diferencia entre clústeres particionados y no particionados.
- Explicar en sus propias palabras cómo se puede seleccionar el número de clústeres.

En el Capítulo 1 discutimos cómo la tarea de clústering hace parte de la caja de herramientas de un científico de datos. También estudiamos la relación de esta tarea con los cuatro tipos de analítica. En este Capítulo nos concentraremos en estudiar la intuición detrás de esta tarea. Los elementos que son comunes a los algoritmos de clústering, y a cómo evaluar la bondad del resultado de un ejercicio de clústering dado que no se cuenta con un valor "real contra el cual comparar" (Ver discusión al final del Capítulo 1).

2.1 La intuición detrás de los algoritmos de clústering

Para hacernos una mejor idea de la tarea de clústering, veamos la Figura 2.1. Supongamos que tenemos dos características[1] para cada individuo (unidad de análisis que se desean agrupar), representadas en los dos ejes de la gráfica y cada punto es una observación o individuo. Por ejemplo, los puntos podrían estar representando clientes y las variables en los ejes ingreso de cada cliente y edad de cada cliente.

Claramente, podemos ver que hay tres grupos (Ver Figura 2.1) bien separados. La distancia entre los grupos es lo que se llama **distancia interclúster** y esta distancia es lo que se desea maximizar; por el contrario, la **distancia al interior de los clústeres** o intraclúster es lo que se quiere minimizar. En esta literatura, la distancia al interior de los grupos se conoce como la **coherencia** y la distancia entre grupos como **separación**.

Lo que queremos es tener clústeres relativamente separados entre sí, como se presenta en la Figura 2.1, y al mismo tiempo al interior de los conglomerados los individuos lo más cerca posible (lo más parecidos). Sin embargo, este ideal no se logra siempre. En algunas ocasiones se pueden sobreponer los clústeres.

Dado nuestro objetivo de construir grupos diferentes entre ellos, pero que a la vez sus miembros sean muy similares, cada vez que realicemos la tarea de clústering tendremos que definir tres elementos:

- Una medida de similitud para determinar qué tan "parecidos" (o "qué tan cerca") son/están los individuos al interior de los grupos y qué tan diferentes son entre grupos. En otras palabras, un criterio para determinar qué significa estar cerca o lejos.
- Un algoritmo para formar los clústeres.
- Un criterio para determinar el número de clústeres.

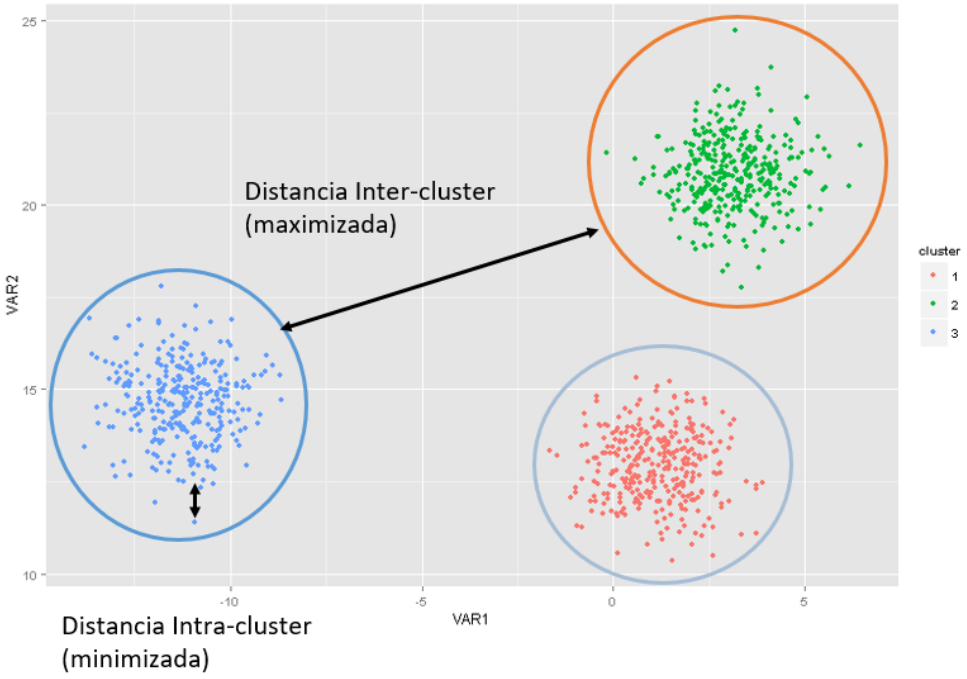
A continuación, discutiremos cada uno de estos tres elementos.

2.2 Medidas de similitud

Para determinar si los individuos a ser agrupados se encuentran "cercanos" o no, se emplean medidas de similitud. Las **medidas de similitud** se pueden clasificar en tres grandes grupos:

[1] En este contexto se emplean como sinónimos de características el término en inglés *features*. Si estás más acostumbrado al lenguaje que se emplea en estadística, un sinónimo para característica es variable. De hecho, recuerda que una variable se define como una característica de los individuos bajo estudio.

Figura 2.1. El problema de la construcción de clústeres



Fuente: elaboración propia.

- Grupo de métricas que emplean la **distancia**, como por ejemplo la **distancia euclidiana**, **Mahalanobis**, **Manhattan** o **Minkowski**. Este tipo de métricas solo funcionan para variables de intervalo o de razón (variables cuantitativas). Estas medidas se calculan para parejas de observaciones y se resumen en una matriz de distancias.
- Grupo de métricas que emplean la **correlación**. Si se trata de la correlación de Pearson, esta métrica solo funcionan para variables cuantitativas (**variables de intervalo o de razón**). No obstante, para variables ordinales puede emplearse otro tipo de correlaciones.
- Grupo de métricas que emplean coeficientes de **asociación**. Este tipo de métricas es apropiado solo para variables cualitativas (medidas en **escala nominal**). En este caso se genera una tabla de comparación y se calculan a partir del **Coeficiente de asociación simple**, **Coeficiente de Jacard**, **Regrers y Tanimoto**, **Sorensen o Dice**, **Sokal y Sneath**, o **Hannann**.

A priori, es difícil determinar cuál tipo de medida de distancia es la más adecuada para cada caso, más allá de descartar algunas medidas por el tipo de variables que se emplean. En últimas, la pregunta de negocio y los datos disponibles y pertinentes para resolver dicha pregunta serán la guía para escoger la medida adecuada de distancia entre las observaciones. Esto implica que, en la práctica, es común que se empleen diferentes medidas de similitud y se comparen los resultados para encontrar cuál medida de similitud (en combinación con el algoritmo de formación de clústeres) genera la agrupación más razonable.

Veamos en detalle la medidas de distancia que se emplean como métrica de similitud.

2.2.1 Medidas de distancia comunes para medir la similitud

Concentrémonos en las medidas de distancia por ser las más empleadas como medidas de similitud. De hecho, no es tan evidente cómo medir la distancia entre dos individuos; en especial si se cuentan con diferentes variables que caracterizan a cada uno de los individuos.

Supongamos que contamos con d variables (características, *features* o dimensiones) que se emplearán para realizar el proceso de agrupamiento de n individuos (observaciones). Definamos a x como un vector que contiene las d características (variables que se emplearán en la construcción de los conglomerados) para un individuo. Es decir, $x = [x_1 \quad x_2 \quad \cdots \quad x_d]$. De manera similar y es un vector que contiene las d características para otro individuo. En el panel a de la Figura 2.2 se presenta un ejemplo para el caso de dos variables.

La **distancia euclidiana** entre los dos individuos se define como la distancia al cuadrado entre los dos vectores[2]. En otras palabras, se calcula la distancia cuadrada entre los dos individuos para cada una de las d dimensiones, posteriormente se suman esas distancias y finalmente se regresan a la escala original calculando la raíz cuadrada[3].

[2] Esta es la norma vectorial de orden 2.

[3] Esta es la razón por la que los datos originales deben ser centrados, como se discutirá más adelante.

Formalmente,

$$d(x,y) = \sqrt{\left(\sum_{j=1}^{d} (x_j - y_j)^2\right)} \tag{2.1}$$

En el panel b de la Figura 2.2 se muestra un ejemplo para el caso de dos variables y dos observaciones.

La distancia euclidiana es la medida de distancia más común. Pero tiene un problema, a medida que aumenta la cantidad de variables involucradas en el cálculo de las distancias (dimensionalidad de los datos), la distancia euclidiana resulta menos útil[4].

Otro tipo de medida de distancia es la **distancia de Chebyshev**, **Distancia máxima** o también conocida como la norma vectorial máxima. La distancia de Chebyshev es la mayor diferencia entre dos vectores a lo largo de cualquier dimensión (variable). En otras palabras, es la distancia máxima entre todas las distancias de cada variable (Ver panel c de la Figura 2.2)[5]. En este caso:

$$d(x,y) = \max_{1 \leq j \leq d} |x_j - y_j| \tag{2.2}$$

Una natural desventaja de esta distancia es que solo tiene en cuenta una dimensión (variable) para medir la proximidad[6].

Por otro lado, tenemos la **distancia de Manhattan** también conocida como la distancia del taxi (en inglés, *Taxicab distance*) o distancia de la manzana (en inglés *City Block distance*). Esta distancia toma el nombre de Manhattan porque es la distancia que recorrería un carro en una ciudad (por ejemplo, Manhattan) en la que las manzanas tienen forma de cuadrícula y para ir de un punto a otro en carro solo se puede hacer rodeando las manzanas. Es decir, calcula la distancia entre dos vectores si sólo pudieran moverse en ángulo recto.

Formalmente, esta distancia corresponde a la distancia absoluta entre los dos vectores[7]. En otras palabras,

$$d(x,y) = \sum_{j=1}^{d} |x_j - y_j| \tag{2.3}$$

En el panel d de la Figura 2.2 se presenta un ejemplo para dos observaciones y dos variables (dimensiones). La distancia Manhattan funciona relativamente bien para

Es decir, se emplean variables centradas (se resta la media) y escaladas (divididas por su desviación estándar) para evitar que las escalas de medición de las variables afecten el resultado, a esto lo llamaremos estandarizar los datos.

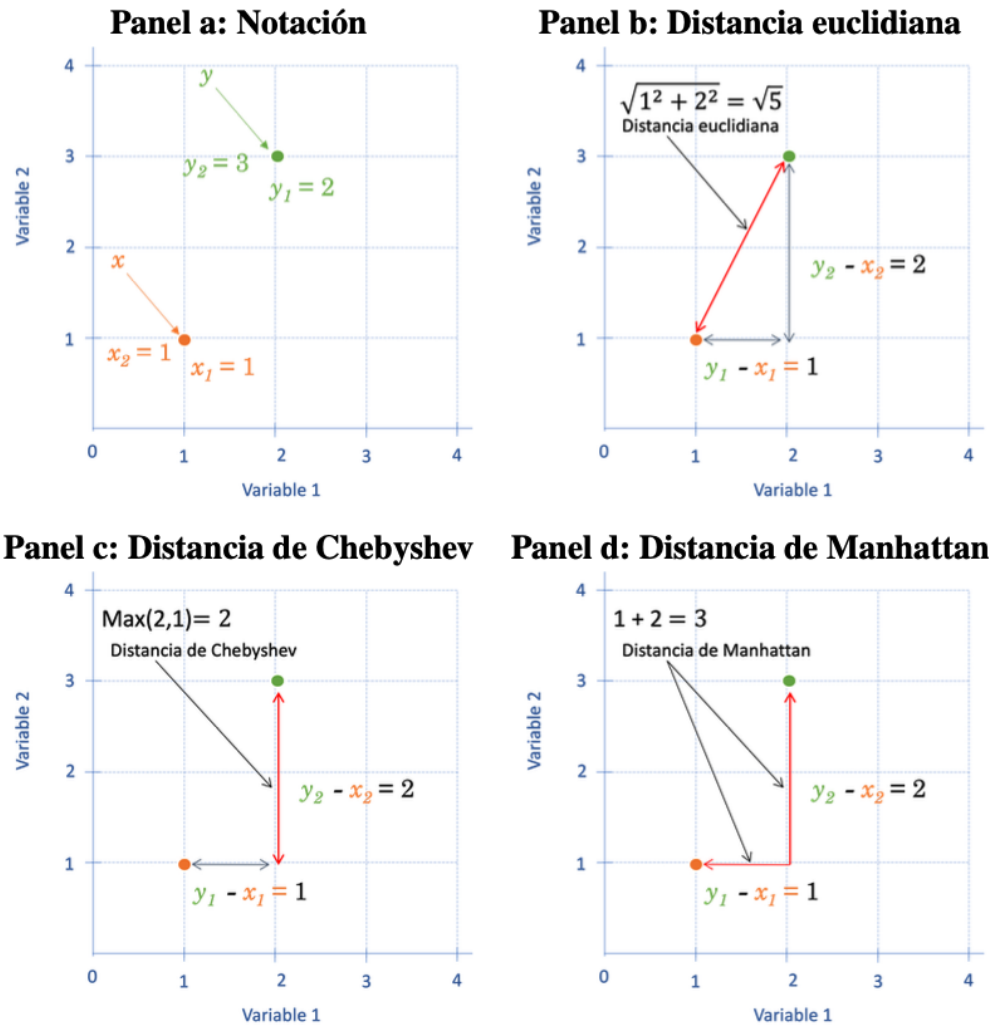
[4] Esto se conoce como la "maldición de la dimensionalidad".

[5] A esta distancia también se le denomina como la distancia del tablero de ajedrez, ya que el número mínimo de movimientos que necesita un rey para ir de una casilla a otra es igual a la distancia de Chebyshev.

[6] En la práctica, la distancia de Chebyshev se utiliza mucho en aplicaciones de logística en especial la de almacenes, ya que se asemeja mucho al tiempo que tarda una grúa en mover un objeto.

[7] Esta es la norma vectorial de orden 1.

Figura 2.2. Representación de las distancias entre dos observaciones con dos variables



Fuente: elaboración propia.

datos con muchas variables (alta dimensión), pero es una medida menos intuitiva que la distancia euclidiana, especialmente cuando se involucran más de dos dimensiones.

Por otro lado, como la distancia de Manhattan utiliza valores absolutos, en lugar de elevar a una potencia o el cálculo de una raíz, esta medida es más robusta frente a valores atípicos. Además, como el módulo es mucho más rápido de calcular que una potencia, esta medida puede ser más rápida en aplicaciones que empleen grandes volúmenes de datos.

La **distancia de Canberra** es una versión ponderada de la distancia Manhattan[8]. Esta distancia se define como:

$$d(x,y) = \sum_{j=1}^{d} \frac{|x_j - y_j|}{|x_j| + |y_j|} \tag{2.4}$$

Existen otras distancias menos intuitivas como la **distancia de Minkowski** . Esta distancia es una generalización de las tres primeras distancias estudiadas. Esta es la norma vectorial de orden p. Es decir, la distancia de Minkowski es la p-ésima raíz de la suma de las p-ésima potencia de las diferencias de los componentes. En otras palabras,

$$d(x,y) = \left(\sum_{j=1}^{d} (x_j - y_j)^p \right)^{\frac{1}{p}} \tag{2.5}$$

Noten que cuando $p = 1$, entonces esta distancia es exactamente igual a la de Manhattan. Con $p = 2$ esta distancia es igual a la euclidiana. Y finalmente, cuando $p = \infty$ entonces obtendremos la distancia de Chebyshev.

Finalmente, la **distancia binaria** implica reorganizar los vectores x y y como si fueran "bits" binarios. Los elementos que no son cero están "prendidos" (*on*) y los elementos iguales a cero están "apagados" (*off*). La distancia es la proporción de bits en los que solo uno está encendido, entre aquellos en los que al menos uno está encendido.

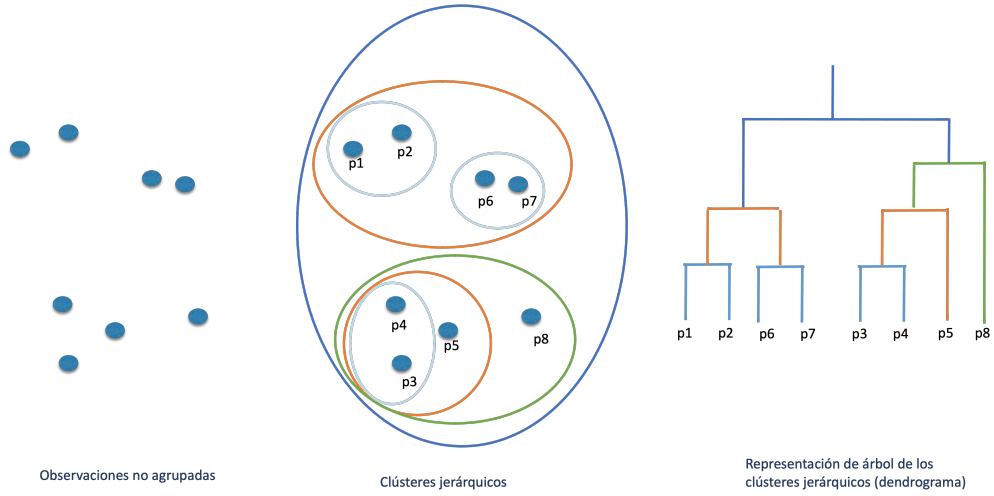
Antes de continuar, es importante anotar que todas estas medidas de distancia son muy sensibles a las unidades en las que se mida cada variable (a la escala). Comúnmente, para evitar este problema, se estandarizan los datos antes de utilizar una medida de distancia. Es decir, se emplean variables centradas (se resta la respectiva media) y con varianza igual (se divide cada una por la respectiva desviación estándar), para evitar que las escalas de medición de las variables y su volatilidad afecten el resultado. A esto lo llamaremos **estandarizar** los datos[9].

[8] Los creadores de esta distancia (Lance y Williams, 1967) le pusieron el nombre de su ciudad natal.
[9] Algunos autores, incorrectamente, llaman a esto normalizar.

2.3 Algoritmos para la formación de clústeres

Después de definir la medida de similitud, el segundo elemento necesario para la construcción de los clústeres es seleccionar un algoritmo para la formación de los grupos. Los **algoritmos para la formación de clústeres** se pueden clasificar en dos tipos: jerárquicos y particionados. Los algoritmos **jerárquicos** construyen subconjuntos de clústeres organizados jerárquicamente como un árbol (ver Figura 2.3). Estos clústeres por construcción no son mutuamente excluyentes. La intuición de este tipo de de algoritmos se discutirán en el Capítulo 3 y su implementación en R en el Capítulo 4.

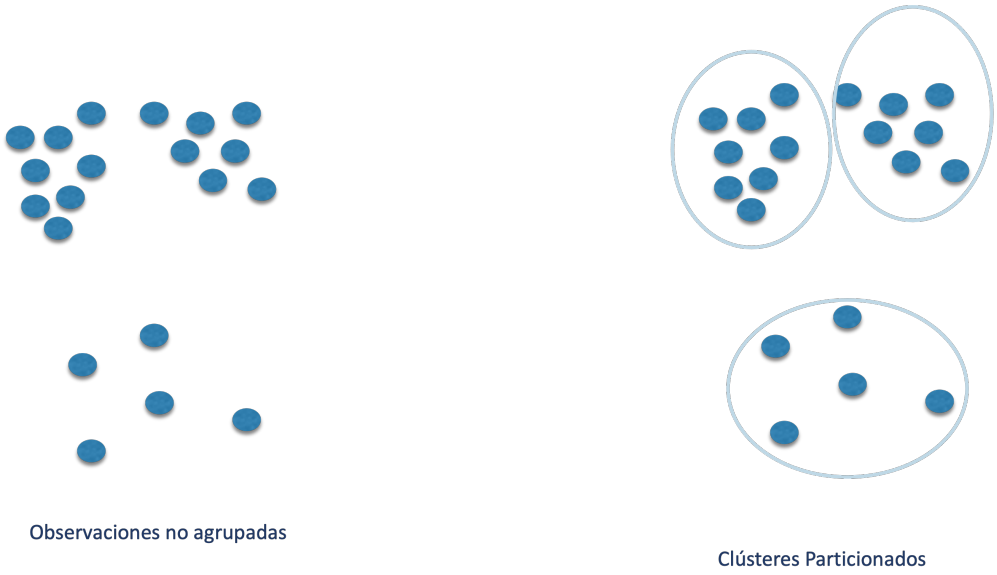
Figura 2.3. Clústeres jerárquicos



Fuente: elaboración propia.

Por otro lado, los algoritmos de clústeres **particionados** forman grupos que son mutuamente excluyentes; es decir, subconjuntos de individuos que no se sobreponen (ver Figura 2.4). Este tipo de algoritmos los discutiremos en los Capítulos 5 a 9.

Figura 2.4. Clústeres particionados



Fuente: elaboración propia.

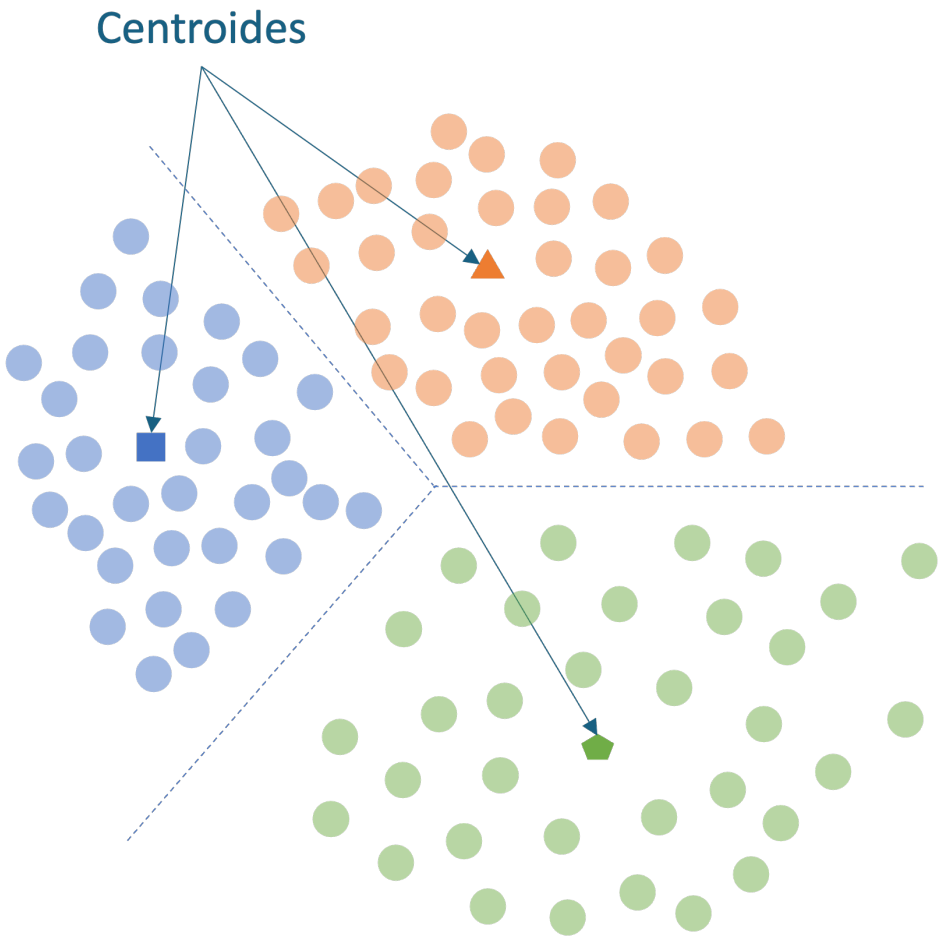
Los algoritmos de clústering **particionados** se pueden dividir a su vez en:

- **Basados en centroides**
- **Basados en densidad**
- **Basados en la distribución**
- **Métodos que combinan o modifican los anteriores**
- **Otros tipos de algoritmos de clústering**

Los algoritmos **basados en centroides** establecen un "centro de gravedad" para cada clúster (Ver Figura 2.5). Entre estos modelos tenemos el modelo **k-means** que divide los datos en k grupos, donde cada uno tiene un centroide. Los puntos se asignan al grupo con el centroide más cercano. Este algoritmo lo estudiaremos en el Capítulo 5. Otro método similar es **k-medoids** que emplea medoides (puntos centrales de un cluster) en lugar de centroides (Ver Capítulo 6). El algoritmo **k-medoids** también es conocido como **PAM** (por su sigla en inglés del término *Partitioning Around Medoids*). En esta categoría también se encuentra el modelo **CLARA** (por su sigla en inglés del término *Clustering Large Applications*) (Ver Capítulo 6).

Una característica de los algoritmos de clústeres particionados **basados en centroides** es que requieren que el usuario especifique de antemano el número de grupos que se van a crear, mientras que los algoritmos jerárquicos, divide a los individuos en todo los posibles grupos y el científico de datos deberá escoger el número de clústeres. Es decir, estos últimos algoritmos no requieren que se establezca *a priori* el número de conglomerados.

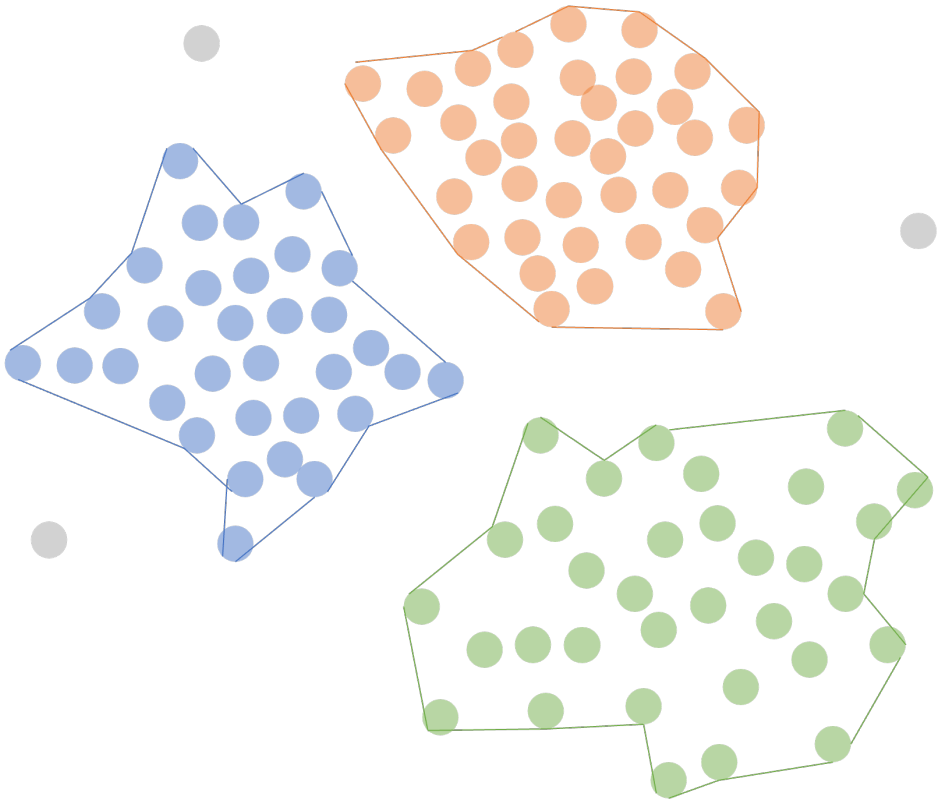
Figura 2.5. Clústeres particionados basados en centroides



Fuente: elaboración propia.

Por otro lado, los algoritmos **basados en densidad** convierten áreas de alta densidad de individuos en clústeres. Esto permite grupos de forma arbitraria siempre que se puedan conectar áreas densas. Un ejemplo de estos algoritmos es el **DBSCAN** (Ver Capítulo 7). El **DBSCAN** encuentra clústeres basados en la densidad de puntos en el espacio de características (*features*). No requiere especificar el número de grupos y es bueno para encontrar conglomerados de formas irregulares (Ver Figura 2.6).

Figura 2.6. Clústeres particionados basados en densidad

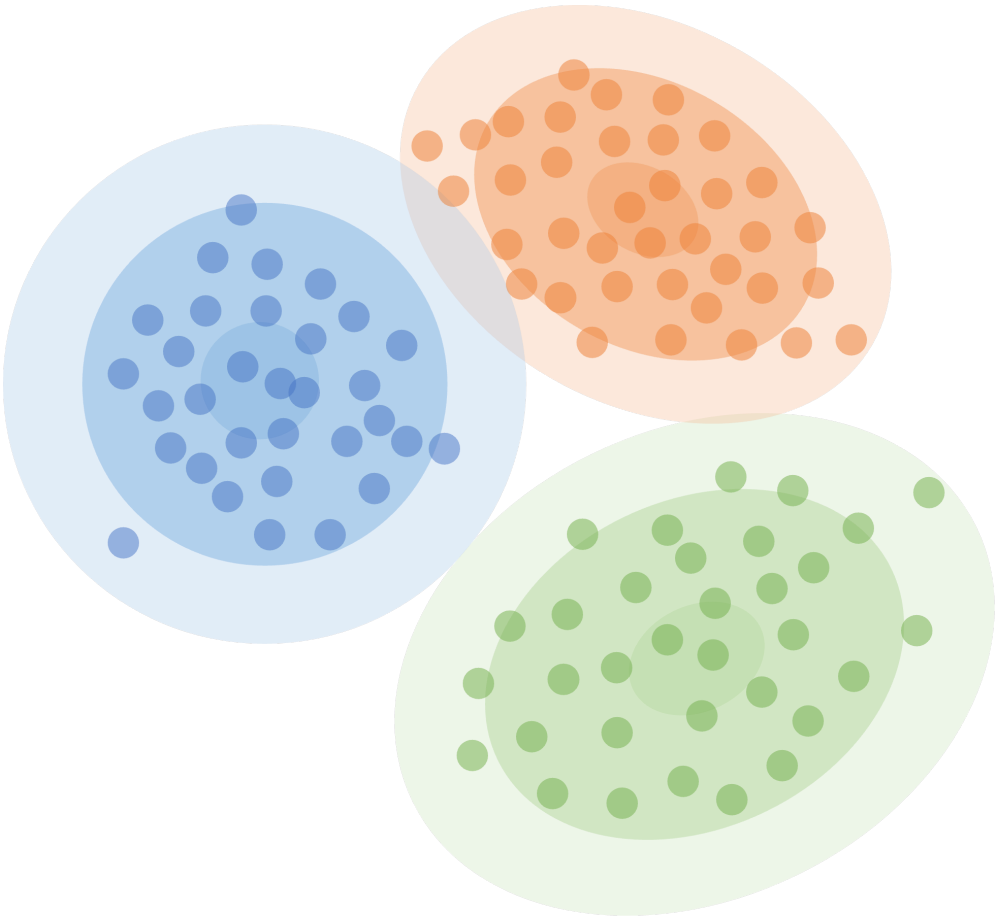


Fuente: elaboración propia.

Los métodos **basados en la distribución**, también conocidos como **basados en modelos**, suponen que los datos fueron generados por una distribución de probabilidad determinada. Un ejemplo de estos algoritmos es el Modelo de Mezcla Gausiano (**GMM** por su sigla en inglés del término *Gaussian Mixed Models*), que asume que los datos se generan a partir de una mezcla de distribuciones normales. En la Figura 2.7 se muestra una representación del algoritmo **GMM** que agrupa los individuos, empleando tres distribuciones normales. A medida que aumenta la distancia al centro de la distribución, disminuye la probabilidad de que un punto pertenezca a la distribución. Este modelo lo estudiaremos en el Capítulo 8.

Adicionalmente, contamos con **métodos que combinan o modifican los anteriores**

Figura 2.7. Clústeres particionados basados en distribuciones



Fuente: elaboración propia.

como por ejemplo *hierarchical K-means* y *fuzzy clustering* (también conocido como **FANNY** por el término en inglés *fuzzy analysis clustering*). El modelo **FANNY** lo estudiaremos en el Capítulo 9.

Finalmente, tenemos otros tipos de algoritmos de clústering para estructuras de datos especiales. Por ejemplo, para datos de redes tenemos los **algoritmos de identificación de comunidades**. Estos algoritmos encuentran grupos basados en la estructura de la red de conexiones entre los puntos de datos. Otro ejemplo son los algoritmos para hacer clústeres de series de tiempo. Estos algoritmos son útiles para analizar la evolución temporal de muchas variables y encontrar patrones recurrentes. Estos tipos de algoritmos no los estudiaremos.

En este libro nos concentraremos en algoritmos para datos de corte transversal; es decir, muchos individuos, cada uno con diferentes características (variables o *features*) cuantitativas. Independientemente del método que adoptemos, siempre deberemos responder a la pregunta ¿cuántos clústeres se deberían emplear para agrupar a los individuos? En la siguiente sección discutiremos técnicas para responder esta pregunta.

2.4 Criterio para determinar el número de clústeres

El tercer elemento necesario para la construcción de clústeres es un criterio para determinar el número de grupos razonables. Seleccionar el número de clústeres no es una tarea trivial. El número q es una variable no observable que reviste cierta dificultad para ser estimada. Es decir, no existe un valor "real" de q observable que se pueda emplear como referencia.

Por ejemplo, en los modelos de clasificación, regresión y pronósticos se puede guardar una parte de la muestra (muestra de evaluación), para determinar si el modelo que ha sido entrenado (estimado) en la muestra de entrenamiento (de estimación) tiene un buen comportamiento en esta. Es decir, se emplea una muestra de entrenamiento para estimar el modelo. Y posteriormente, los modelos candidatos son comparados en la muestra de evaluación. Entre más "cerca" estén las predicciones del modelo en la muestra de evaluación, mejor será el modelo. Esto siempre será posible con este tipo de modelos que implican un aprendizaje supervisado.

Lastimosamente, en la tarea de clústering esto no es posible. Dado que las observaciones no tienen un rótulo (*label*) que permita saber con certeza a qué grupo "realmente" pertenece cada individuo, no es posible comparar la "predicción" del grupo con un valor real. Es decir, dada la naturaleza de esta tarea, existe una variable latente (la membresía a un grupo) que queremos "adivinar" pero que no podemos observar para determinar si nos estamos equivocando mucho o poco.

Por eso es necesario tener una aproximación diferente para determinar cuál aproximación es mejor[10].

Existen cuatro maneras muy empleadas en la práctica para determinar el mejor número de grupos q: el método del "codo", el estadístico o índice de *Gap* (Tibshirani et al.,

[10]En este contexto una aproximación implica tanto el algoritmo seleccionado, la medida de distancia, como el número de clústeres q.

2001), la silueta promedio o una combinación de un número de métricas. Veamos a continuación cada una de estas aproximaciones.

2.4.1 Selección empleando el método del codo

Este método es tal vez el más conocido y parte de la idea fundamental de las tareas de armar agrupaciones. Recordemos que lo que se busca es encontrar clústeres, de forma que la distancia al interior de los grupos sea la más baja posible. Es decir, que la distancia intraclúster sea la menor posible (Ver Figura 2.1). Una medida de esa distancia intraclúster es la variación total intraclúster (WSS por su sigla en inglés de *Withincluster Sum of Square*) que se define como la suma de las distancias de cada una de las observaciones al respectivo centro[11] del clúster.

La suma de cuadrados de todas las observaciones que pertenecen a un conglomerado es una medida de la variabilidad de las observaciones dentro de cada clúster. En general, un conglomerado que tiene una suma de cuadrados pequeña es más compacto que uno que tiene una suma de cuadrados grande. Así, la suma (total) de todas estas medidas de variabilidad de los clústeres (WSS) es una medida de qué tanta variabilidad presentan las observaciones para un determinado número de clústeres k[12].

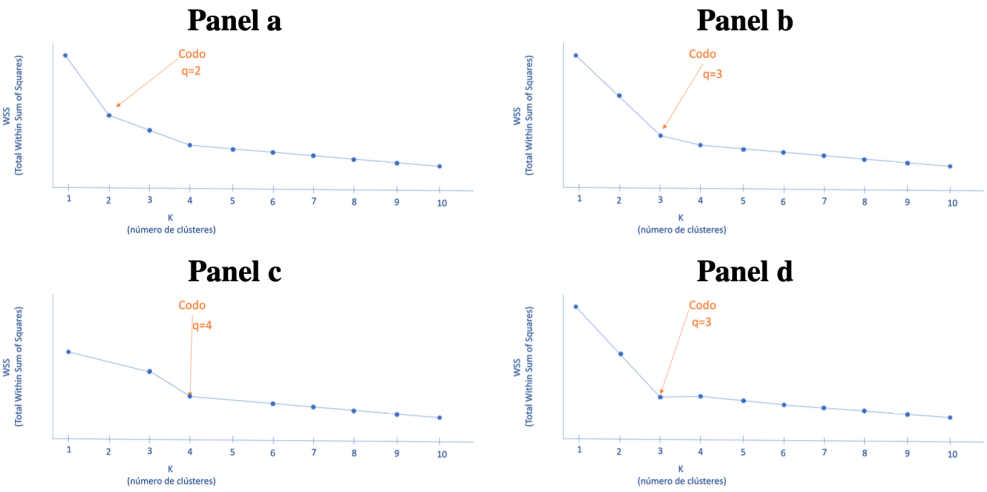
Entonces, para encontrar el número óptimo de k, se acostumbra graficar en el eje horizontal el número de clústeres y en el eje vertical el WSS (ver Figura 2.8). El número óptimo de clústeres (q) corresponderá al valor de k para el cuál se presente una caída repentina y más grande en el WSS, de tal manera que la gráfica parece tener un "codo" (de ahí el nombre del método).

En la Figura 2.8 se presentan diferentes representaciones de la curva WSS y la selección del número óptimo de clústeres empleando la técnica del codo. Noten que estamos buscando dónde se presente la mayor caída en el WSS de tal manera que se observa una forma de codo recogido. Por la manera como se selecciona el mejor clúster, este método de selección del número de grupos no permite comparar entre diferentes algoritmos de formación de clústeres.

[11] Recuerda que en este contexto el centro se conoce como el centroide.

[12] Sin embargo, al igual que las Sumas de Cuadrados y los Cuadrados medios en el modelo ANOVA o la tabla ANOVA de un modelo de regresión, la suma de cuadrados dentro del grupo está influida por el número de observaciones. A medida que aumenta el número de observaciones, la suma de cuadrados es mayor. Por lo tanto, la WSS no se puede emplear para comparar ejercicios de formación de clústeres con diferentes números de observaciones.

Figura 2.8. Representación de la selección del número de clústeres óptimo con el método del codo



Fuente: elaboración propia.

2.4.2 Selección empleando el estadístico de Gap

Otra manera muy empleada para seleccionar el número óptimo de clústeres es el estadístico *Gap*[13] propuesto por Tibshirani et al. (2001).

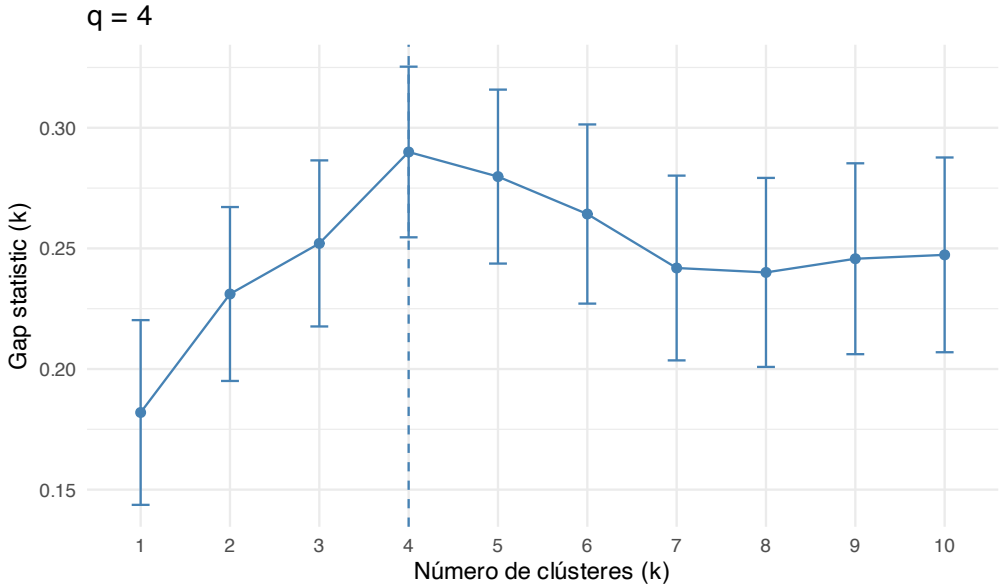
El estadístico de la brecha compara la variación total dentro del clúster (WSS) para diferentes valores de k con sus valores esperados bajo la distribución uniforme de referencia. Esto permite encontrar para cada k qué tanto se aleja el WSS del valor esperado, cuando todos los datos se distribuyen de manera aleatoria (y uniforme). En la Sección 2.4.2 se presenta el cálculo de este estadístico.

Intuitivamente, el número de grupos óptimo (q) será aquel para el cual la estructura de agrupación se aleja más de la distribución uniforme aleatoria de los puntos. Tibshirani et al. (2001) demuestran que esto ocurre cuando el número de clústeres corresponde al menor de los valores k para el que el estadístico *Gap* se aleja menos de una desviación estándar del valor *Gap* del siguiente ($k+1$). Afortunadamente, R hará ese cálculo por nosotros. Por la manera cómo se selecciona el mejor clúster, este método de selección del número de conglomerados no permite comparar entre diferentes algoritmos de formación de clústeres.

Al igual que con el método del codo, es común que este estadístico se calcule para un número de k y se grafique incluyendo el respectivo intervalo de confianza. Y R presentará con una línea punteada el k que garantiza la condición establecida por Tibshirani et al. (2001). La Figura 2.9 presenta un ejemplo en el que el número óptimo de conglomerados es 4.

[13] Una traducción sería brecha, pero en la literatura se conoce más como el estadístico *Gap*.

Figura 2.9. Ejemplo de la selección del número de clústeres óptimo empleando el estadístico GAP



Fuente: elaboración propia.

2.4.3 Selección empleando la silueta promedio

La silueta es una medida de qué tan similar es un objeto a su propio grupo (cohesión) en comparación con otros grupos (separación). Empecemos por definir un individuo i que pertenece al clúster C como C_i. Entonces, una medida para el individuo i de su cohesión con el grupo C_i es:

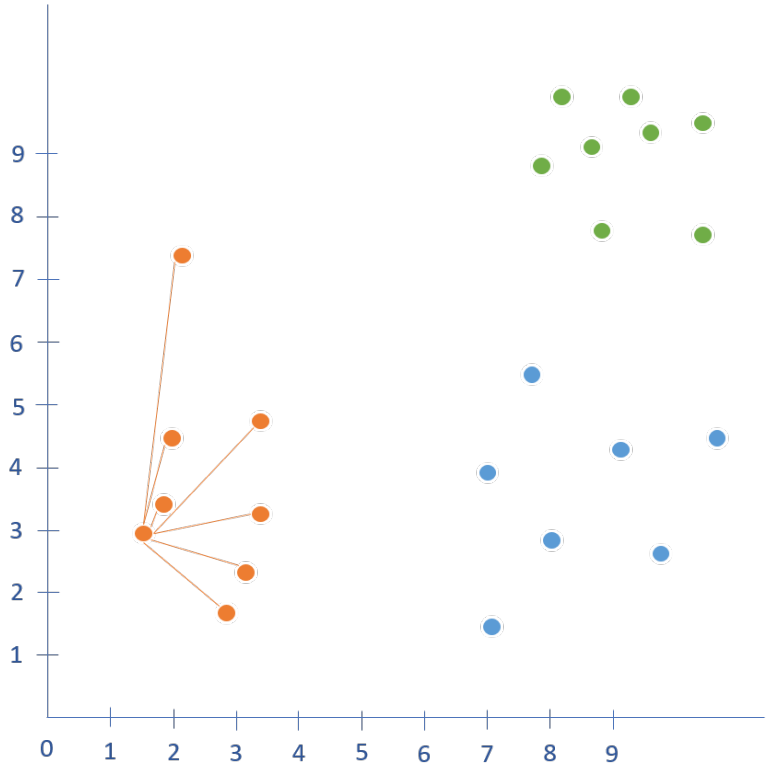
$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \tag{2.6}$$

donde $d(i,j)$ es la distancia[14] entre el individuo i y el j que también pertenece al clúster C_i. Es decir, $a(i)$ representa la distancia promedio del individuo i a todos los otros puntos del clúster C_i. Por eso también es conocida como la **distancia promedio intraclúster**.

[14]Cualquiera de las medidas de distancia estudiadas funciona en esta medida de cohesión.

La Figura 2.10 muestra una representación de todas las distancias (en rojo) a las que se le sacaría el promedio para calcular $a(i)$ para el individuo seleccionado. Entonces, $a(i)$ se puede interpretar como una medida de la asignación de i al clúster C_i. Entre más pequeño sea $a(i)$ mejor es la cohesión; i.e. mejor fue la decisión de asignar el individuo i al clúster C_i.

Figura 2.10. Representación de la distancia intraclúster para un individuo i



Fuente: elaboración propia.

Ahora podemos medir qué tan diferente es el individuo i de todas las observaciones de otro clúster (por ejemplo el clúster C_k) como

$$\frac{1}{|C_k|}\sum_{j\in C_k} d(i,j) \tag{2.7}$$

para todo $k \neq i$. Esta distancia también se conoce como la **distancia interclúster** para el individuo i. Noten que existirán tantas distancias interclústeres como números de conglomerados menos uno $(q-1)$ para cada observación[15]. Supongamos que contamos con tres clústeres: los naranjas, los azules y los verdes. Entonces existirán dos distancias interclúster para una observación del clúster naranja, como se puede observar en la

[15]No se necesita calcular la distancia con respecto al clúster en el que está el individuo i (C_i).

Figura 2.11. Noten que todas las líneas azules se promediarán, así como todas las verdes para producir dos distancias (promedio) interclúster.

Ahora calculemos para cada individuo i del clúster C_i el clúster más cercano (el vecino más cercano) de la siguiente manera:

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j) \tag{2.8}$$

Es decir, $b(i)$ es una medida de qué tan cerca se encuentra el clúster más cercano (diferente al que ya pertenecía (C_i)).

La silueta para una observación determinada i es una combinación de la cohesión $a(i)$ y $b(i)$. La silueta para el individuo i está definida como:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.9}$$

Esto es equivalente a

$$s(i) = \begin{cases} 1 - a(i)/b(i) & si \quad a(i) < b(i) \\ 0 & si \quad a(i) = b(i) \\ b(i)/a(i) - 1 & si \quad a(i) > b(i) \end{cases} \tag{2.10}$$

Esta idea fue propuesta por Rousseeuw (1987).

Es fácil demostrar que $s(i)$ se encuentra entre -1 y +1. Entre más cercano a uno indica que el individuo i está bien ubicado en su conglomerado. Es decir, la clasificación actual en C_i es mejor que con los grupos vecinos. Por otro lado, un valor cerca a cero o negativo implica que es posible que el individuo fue mal clasificado en su clúster C_i.

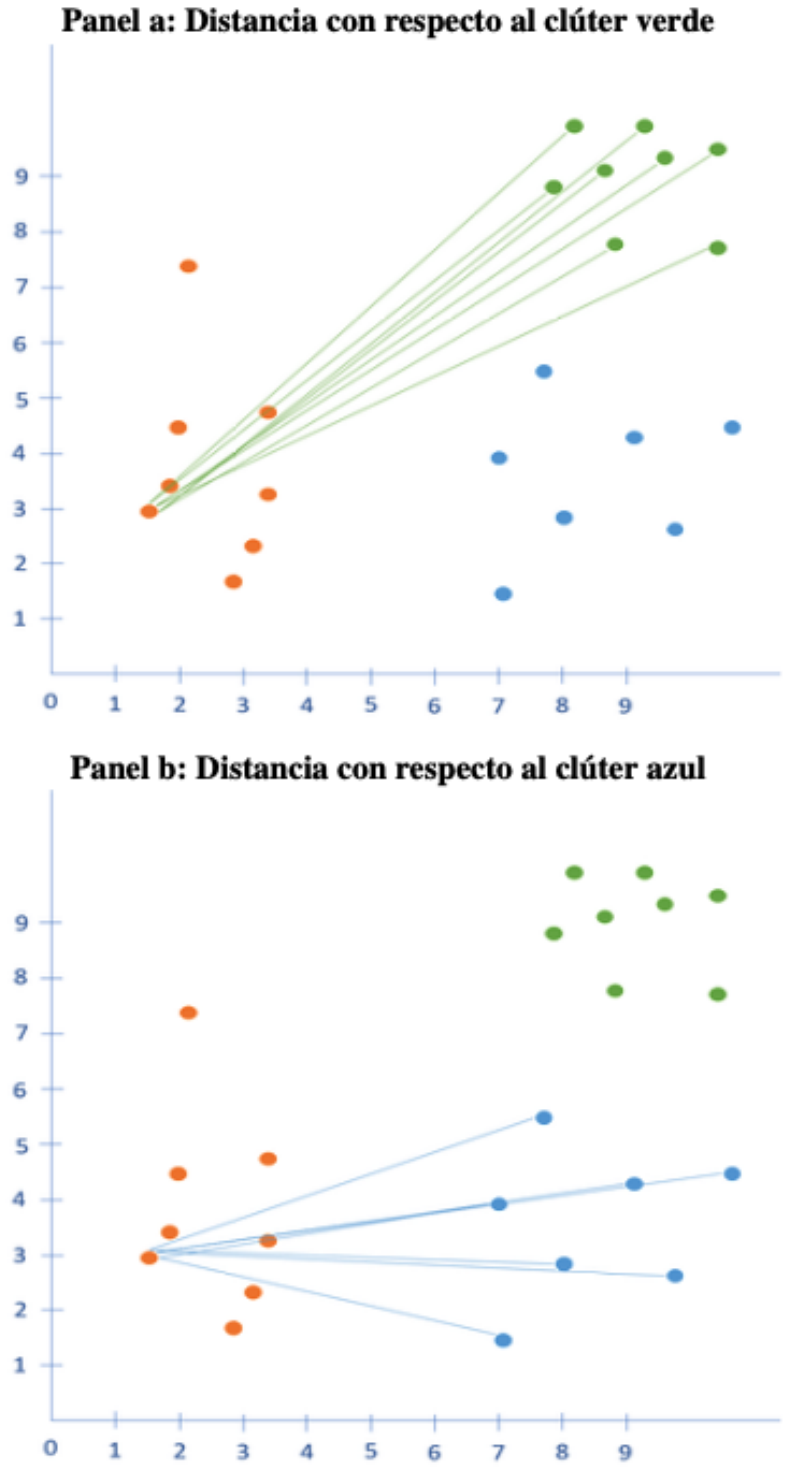
Si la mayoría de individuos tienen un valor alto de $s(i)$, entonces la estructura de agrupamiento es apropiada. Si muchos individuos tienen una silueta baja o negativa, entonces la estructura encontrada con los clústeres puede tener demasiados o muy pocos clústeres. Dado que en un ejercicio de clústering nos podemos contar con muchos individuos, es común que se calcule la silueta promedio para cada clúster y para todos los individuos.

En general una regla empírica comúnmente aceptada es :

- Si la silueta (promedio o individual) está entre **0.71 y 1.0** entonces la estructura de clasificación encontrada es **fuerte**.
- Si la silueta (promedio o individual) está entre **0.51 y 0.7** entonces la estructura de clasificación encontrada es **razonable** (aceptable).
- Si la silueta (promedio o individual) está entre **0.26 y 0.50** entonces la estructura de clasificación encontrada es **débil** y puede ser artificial.
- Si la silueta (promedio o individual) es **< 0.25** entonces no se ha encontrado una estructura de clasificación.

En la práctica, se emplea la silueta promedio para escoger el número óptimo de grupos. De igual manera que en el método del codo y el estadístico del *Gap*, la silueta promedio es calculada para un número relativamente grande de clústeres (k) y

Figura 2.11. Representación de la distancia interclúster para un individuo i



Fuente: elaboración propia.

se selecciona el número óptimo de conglomerados (q) donde la silueta promedio es maximizada. Por otro lado, también es común emplear las siluetas individuales para ver cuáles observaciones quedan mejor asignadas a un clúster. Finalmente, es importante aclarar que la silueta promedio permite comparar diferentes algoritmos para formar clústeres.

2.4.4 Selección empleando batería de métricas

En la literatura se han desarrollado una gran variedad de **índices de validación** que permitan comparar diferentes aproximaciones. Estos índices o métricas combinan de diferente manera información sobre la compacidad de la aproximación[16] de generar grupos parecidos (intragrupo) y grupos muy diferentes (el aislamiento entre grupos o intergrupos). En el Anexo de este Capítulo (Ver Sección 2.6) se presentan 30 métricas diferentes que comúnmente se emplean.

En la práctica, estos 30 índices se calculan para un número relativamente grande de posibles aproximaciones para determinar el algoritmo de construcción, medida de distancia y número de clústeres. El número de clústeres que sea seleccionado por la mayor cantidad de estos índices es considerado el mejor número de clústeres q. En los siguientes capítulos veremos ejemplos de esto.

2.5 Comentarios finales

Hasta el momento hemos discutido las generalidades de la tarea de clústering. Vimos que la construcción de clústeres implica seleccionar tanto un algoritmo, una métrica de distancia y una manera de seleccionar el número de clústeres. En los siguientes capítulos veremos diferentes algoritmos de construcción de clústering. Para seleccionar el número de clústeres (q) emplearemos los métodos discutidos en este capítulo.

2.6 Anexo: Índices de validación

Para describir cada uno de los 30 índices de validación más usados, empleemos, siguiendo a Charrad et al. (2014), la siguiente notación:

- n: Número de observaciones.
- d: Número de variables o características.
- q: Número de clústeres.
- X: Matriz de datos con n filas y d columnas (cada columna corresponde a una variable. Es decir, x_{ij}, i = 1,2,. . . , n y j= 1,2,. . . ,p.
- $\overline{X}$: Matriz q x d de métodos de agrupación (clúster).
- $\overline{x}$: Matriz centroide X.
- n_k: Número de objetos en el clúster C_k.
- c_k: Centroide del clúster C_k.
- x_i: Vector de d dimensiones de observaciones del i-ésimo objeto en el clúster C_k. En algunas ocasiones, para simplificar la notación se eliminará el subíndice.

[16] Como se mencionó anteriormente, en este contexto una aproximación implica: un algoritmo de construcción, una medida de distancia y un número de clústeres q.

- $\|x\|$: Corresponde a $(x^T x)^{1/2}$ (en este caso se omitieron los subíndices).
- W_q: Se define como $\sum_{k=1}^{q} \sum_{i \in C_k} (x_i - c_k)(x_i - c_k)^T$ Es la matriz de dispersión intraclúster, clusterizada en q clústeres.
- B_q: Se define como $\sum_{k=1}^{q} n_k (c_k - \overline{x})(c_k - \overline{x})^T$: Es la matriz de dispersión interclúster, agrupada en q clústeres.
- N_t: Número total de pares de observaciones en los datos: $N_t = \frac{n(n-1)}{2}$.
- N_w: Número de pares de observaciones que pertenecen al mismo clúster: $N_w = \sum_{k=1}^{q} \frac{n_k(n_k-1)}{2}$
- N_b: Número de pares de observaciones que pertenecen a diferentes clústeres. Es decir, $N_b = N_t - N_w$.
- S_w: Suma de las distancias intraclúster. Es decir, $S_w = \sum_{q}^{k=1} \sum_{i,j \in C_k} d(x_i, x_j)$.
- S_b: Suma de las distancias interclúster. Es decir,$S_b = \sum_{q-1}^{k=1} \sum_{l=k+1}^{q} \sum_{i \in C_k} d(x_i, x_j)$.

A continuación, se describen las 30 métricas más empleadas para la selección de q.

2.6.1 Índice CH

El **índice de Calinski y Harabasz (CH)** (Caliński y Harabasz, 1974) está definido como:

$$CH(q) = \frac{tr(B_q)/(q-1)}{tr(W_q)/(n-q)} \tag{2.11}$$

donde tr representa la traza de una matriz. El valor de q que maximiza la función $CH(q)$ sería el número de clústeres seleccionado con este método.

2.6.2 Índice Duda

Duda et al. (1973) propusieron un criterio usando la siguiente razón:

$$Je(2)/Je(1) \tag{2.12}$$

donde $Je(2)$ es la suma de las distancias al cuadrado intraclúster cuando los datos están divididos en dos clústeres, y $Je(1)$ es la distancia intraclúster al cuadrado con un solo clúster. Este índice se define como:

$$Duda = \frac{Je(2)}{Je(1)} = \frac{W_k + W_l}{W_m} \tag{2.13}$$

Se asume que los clústeres C_k y C_l se unen para formar C_m. Según Gordon (1999), el número óptimo de clústeres es el q más pequeño, tal que:

$$Duda \geq 1 - \frac{2}{\pi d} - z\sqrt{\frac{2(1 - \frac{8}{\pi^2 d})}{n_m d}} = Valor\ crítico\ Duda \tag{2.14}$$

2.6.3 Índice Pseudo t^2

Duda et al. (1973) propusieron otro índice denominado el Pseudo t^2, el cual solo es aplicable a métodos jerárquicos:

$$Pseudo\ t^2 = \frac{V_{kl}}{\frac{W_k+W_l}{n_k+n_l-2}} \tag{2.15}$$

donde V_{kl}=W_m-W_k-Wl, si $C_m = C_k \cup C_l$.

Gordon (1999) demostró que el número óptimo de clústeres es el mejor q tal que:

$$Pseudo\ t^2 \leq \frac{1 - Valor\ crítico\ Duda}{Valor\ crítico\ Duda} \cdot (n_k + n_l - 2) \tag{2.16}$$

2.6.4 Índice C

El **índice C** fue propuesto por Hubert y Levin (1976). Se calcula como:

$$Indice\ C = \frac{S_w - S_{min}}{S_{max} - S_{min}}, S_{min} \neq S_{max}, Indice\ C \in (0,1) \tag{2.17}$$

donde

- S_{min}: Es la suma de las N_w pequeñas distancias entre todos los pares de puntos de los datos (N_t es el número de pares).
- S_{max}: Es la suma de las N_w mayores distancias entre todos los pares de puntos de los datos totales.

Milligan y Cooper (1985) y Gordon (1999) sugieren que el número óptimo de clústeres se fije encontrando el número de clústeres (Q) para el cual este índice se minimiza.

2.6.5 Índice Gamma

El **Índice Gamma** es una adaptación del estadístico Gamma de Goodman y Kriskal para usar en clústeres (Baker y Hubert, 1976).

Se hacen comparaciones entre las diferencias de los individuos del mismo clúster (intraclúster) y entre los individuos de diferentes clústeres (interclúster) . Una comparación es **concordante** (**discordante**) si la diferencia intraclúster es estrictamente menor (estrictamente mayor) a la diferencia interclúster. Las igualdades entre un par de grupos diferentes son ignoradas en definición del índice (Gordon, 1999). El índice se define como:

$$Gamma = \frac{s(+) - s(-)}{s(+) + s(-)} \tag{2.18}$$

donde

- $s(+)$: Es el número de comparaciones **concordantes**.

- $s(-)$: Es el número de comparaciones **discordantes**.

El máximo valor del índice representa el número correcto de clústeres (Milligan y Cooper, 1985).

2.6.6 Índice de Beale

Beale (1969) propuso un estadístico F para probar la hipótesis de la existencia de q_1 contra q_2 clústeres en los datos, donde $q_2 > q_1$.

$$Beale_F \equiv \frac{\frac{V_{kl}}{W_k+W_l}}{(\frac{n_m-1}{n_m-2})2^{\frac{2}{d}}-1} \tag{2.19}$$

donde V_{kl} = W_m-W_k-W_l. Se asume que los clústeres C_k y C_l se unen para formar C_m. El número óptimo de clústeres se obtiene comparando F con una distribución $F_{d,(n_m-2)d}$. Para valores significativamente grandes de F, la hipótesis nula de un solo clúster es rechazada (Gordon, 1999).

2.6.7 Índice CCC

El **Criterio Cúbico de Clústering** (*Cubic Clustering Criterion* - CCC en inglés) es una prueba estadística, cuyo estadístico de prueba está dado por:

$$CCC = \ln\left[\frac{1-E(R^2)}{1-R^2}\right]\frac{\sqrt{\frac{nd^*}{2}}}{(0{,}001+E(R^2))^{1},2} \tag{2.20}$$

donde

$$R^2 = 1 - \frac{tr(X^TX - \overline{X}^TZ^TZ\overline{X})}{tr(X^TX)}$$

$\overline{X}$ corresponde al producto $(Z^TZ)^{-1}Z^TX$, donde Z es una matriz indicadora de clústeres (nxq) con z_{ik} = 1 si la observación i pertenece al clúster k y z_{ik}= 0 de lo contrario. Adicionalmente,

$$E(R^2) = 1 - \left[\frac{\sum_{j=1}^{d*}\frac{1}{n+u_j} + \sum_{j=d*+1}^{p}\frac{u_j^2}{n+u_j}}{\sum_{j=1}^{d}}\right]\left[\frac{(n-d)^2}{n}\right]\left[1+\frac{4}{n}\right]$$

donde u_j es $\frac{s_j}{c}$ y s_j es la raíz cuadrada del valor propio j de $\frac{X^TX}{(n-1)}$. Y $c = \left(\frac{v*}{q}\right)^{\frac{1}{d*}}$. Además, $v* = \Pi_{j=1}^{d*}s_j$ y $d*$ es el mayor entero escogido menor que q tal que u_{d*} no es menor que 1.

El máximo valor del índice indica el número óptimo de clústeres en los datos (Milligan y Cooper, 1985).

2.6.8 Índice Pt-biserial

Este índice, propuesto por Milligan (1980) y Kraemer (2004), es una correlación bi-serial entre la matriz de entrada de diferencias y una matriz correspondiente con entradas 0 y 1; 0 es asignado si los dos puntos correspondientes están clusterizados juntos por el algoritmo, y 1 es lo contrario (Milligan, 1980).

Dado que los valores positivos más grandes reflejan un mejor ajuste entre los datos y la división encontrada, el máximo valor del índice es usado para seleccionar el número óptimo de clústeres (Milligan, 1980).

El coeficiente de correlación bi-serial se calcula de la siguiente manera (Milligan, 1981).

$$Ptbiserial = \frac{[\overline{S}_b - \overline{S}_w][N_w N_b / N_t^2]^{1/2}}{s_d} \tag{2.21}$$

donde $\overline{S}_w$: S_w/N_w, $\overline{S}_b$: S_b/Nb y s_d es la desviación estándar de todas las distancias.

2.6.9 Índice Gplus

Este índice fue diseñado por Rohlf (1974) y reformulado por Milligan (1981). Se calcula de la siguiente manera:

$$Gplus = \frac{2s(-)}{N_t(N_t - 1)} \tag{2.22}$$

donde $s(-)$ es el número de comparaciones discordantes o el número de veces que dos puntos que estaban en el mismo clúster tenían una distancia mayor que dos puntos, que no se clusterizaron juntos (Milligan, 1981). Los valores mínimos del índice determinan el número óptimo de clústeres (Milligan y Cooper, 1985).

2.6.10 Índice DB

El índice de Davies y Bouldin (1979) es una función de la suma de ratios intraclúster dispersos a separación interclúster. Este índice se define como:

$$DB(q) = \frac{1}{q}\sum_{k=1}^{q} \max_{k \neq l}\left(\frac{\delta_k + \delta_l}{D_{kl}}\right) \tag{2.23}$$

donde $k, l = 1, ..., q$ es el número de clústeres. $D_{kl} = \sqrt[v]{\sum_{j=1}^{d} |c_{kj} - c_{lj}|}^{v}$ es la distancia de Minkowski entre los centroides de los clústeres C_k y C_l. δ_k es una medida de dispersión de un clúster C_k con respecto al centroide de ese clúster empleando la distancia Minkowski.

Según Milligan y Cooper (1985) y Davies y Bouldin (1979), el valor de q que minimiza DB(q) corresponde al número de clústeres óptimo.

2.6.11 Índice de Frey

El índice propuesto por Frey y Van Groenewoud (1972) solo puede ser aplicado a métodos jerárquicos. Es el ratio de la diferencia entre puntajes de dos niveles sucesivos en la jerarquía. El numerador es la diferencia entre la media de distancias interclúster, $\overline{D}_b$, de cada uno de los dos niveles jerárquicos (nivel j y $j+1$). Los autores proponen usar un puntaje (ratio) de 1.00, para identificar el nivel correcto del clúster. Los ratios suelen ser mayores o menores de 1.

Los mejores resultados se dieron cuando se continuó la clusterización hasta que el último ratio fue menor de 1.00. En este punto, el nivel del clúster anterior era tomado como partición óptima. Si el ratio nunca era menor de 1, se asumía que la solución era un solo clúster (Milligan y Cooper, 1985). Este indicador se define como:

$$Frey = \frac{\overline{S}_{b,j+1} - \overline{S}_{b,j}}{\overline{S}_{w,j+1} - \overline{S}_{w,j}} \tag{2.24}$$

donde $\overline{S}_b$=S_b/N_b es la media de la distancia interclúster y $\overline{S}_w$=S_w/N_w es la media de la distancia intraclúster.

2.6.12 Índice de Hartigan

Este índice se calcula de la siguiente manera (Hartigan, 1975):

$$Hartigan = \left(\frac{tr(W_q)}{tr(W_{q+1})} - 1\right)(n-q-1) \tag{2.25}$$

donde $q \in \{1, ..., n-2\}$. La máxima diferencia entre niveles jerárquicos es el indicativo del número correcto de clústeres (Milligan y Cooper, 1985).

2.6.13 Índice Tau

Este índice, propuesto por Rohlf (1974) y revisado por Milligan (1981), se calcula entre las entradas correspondientes de dos matrices. La primera contiene las distancias entre los objetos y la segunda es una matriz de 0 y 1 que indica si un par de puntos están dentro del mismo clúster o no.

$$Tau = \frac{s(+) - s(-)}{[(N_t(N_t-1)/2 - t)(N_t(N_t-1)/2)]^{1/2}} \tag{2.26}$$

El máximo valor del índice es tomado como indicativo del número correcto de clústeres (Milligan y Cooper, 1985).

2.6.14 Índice de Ratkowsky

Ratkowsky y Lance (1978) propusieron un criterio para determinar el número óptimo de clústeres basados en $\frac{\overline{S}}{q^{1/2}}$. El valor de $\overline{S}$ es el promedio de los ratios de ($\frac{BGSS_J}{TSS_j}$),

donde $BGSS$ es la suma de cuadrados interclúster para cada variable[17] y TSS es el total de la suma de cuadrados para cada variable[18] (Hill, 1980).

El número óptimo de clústeres es el valor q para el cual $\frac{\overline{S}}{q^{1/2}}$ alcanza su valor máximo (Milligan y Cooper, 1985). Si el valor de q se hace constante, este criterio se ve reducido de $\frac{\overline{S}}{q^{1/2}}$ a $\overline{S}$ (Hill, 1980).

$$Ratkowsky = \frac{\overline{S}}{q^{1/2}} \tag{2.27}$$

donde $\overline{S}^2 = \frac{1}{d}\sum_{j=1}^{d} \frac{BGSS_j}{TSS_j}$.

2.6.15 Índice de Scott

Scott y Symons (1971) introdujeron un índice basado en la siguiente ecuación:

$$Scott = n \log\left(\frac{det(T)}{det(S_q)}\right) \tag{2.28}$$

donde n es el número de elementos en los datos, T es la suma total de cuadrados y S_q es la suma de cuadrados entre los q clústeres.

La máxima diferencia entre los niveles jerárquicos se usa para sugerir el número correcto de particiones (Milligan y Cooper, 1985).

2.6.16 Índice de Marriott

Marriott (1971) propuso el siguiente índice:

$$Marriot = q^2 det(S_q) \tag{2.29}$$

La máxima diferencia entre niveles sucesivos se usa para determinar el mejor nivel de partición (Milligan y Cooper, 1985).

2.6.17 Índice de Ball

Ball y Hall (1965) propusieron un índice basado en la distancia promedio entre los objetos y su respectivo centroide del clúster. Se calcula:

$$Ball = \frac{S_q}{q} \tag{2.30}$$

La diferencia más grande entre niveles se usa para indicar la solución óptima (Milligan y Cooper, 1985). (Ver también Dimitriadou et al. (2002)).

[17] Es decir, $BGSS_j = \sum_{k=1}^{q} n_k (c_{kj} - \overline{x}_j)^2$.
[18] En otras palabras, $TSS_j = \sum_{i=1}^{n} (x_{ij} - \overline{x}_j)^2$.

2.6.18 Índice de Trcovw

Este índice, propuesto por Milligan y Cooper (1985), recoge la información en la matriz de covarianzas intraclústeres por medio de la traza (Ver también Dimitriadou et al. (2002)). Es decir,

$$Trcovw = tr(Cov(W_q)) \tag{2.31}$$

La máxima diferencia entre los puntajes de los niveles es usada para indicar la solución óptima (Milligan y Cooper, 1985).

2.6.19 Índice de Tracew

Este índice ha sido una de las sugerencias más populares en la literatura, por ejemplo ha sido propuesto por Milligan y Cooper (1985), Edwards y Cavalli-Sforza (1965), Friedman y Rubin (1967), Orlóci (1967) y Fukunaga y Koontz (1970). No emplea la covarianza de la matriz de distancias intraclústeres, sino la misma matriz de distancias. En otras palabras,

$$Tracew = tr(W_q) \tag{2.32}$$

Dado que este criterio aumenta monótonamente con soluciones que contienen menos clústeres, el máximo de las segundas diferencias de los puntajes es usado para determinar el número de clústeres en los datos (Milligan y Cooper, 1985).

2.6.20 Índice de Friedmdan

Este índice fue propuesto por Friedman y Rubin (1967) como una métrica para un método de clústering no jerárquico (para una mayor discusión ver Dimitriadou et al. (2002)). Formalmente, se define cómo:

$$Friedman = tr(W_q^{-1}B_q) \tag{2.33}$$

La diferencia máxima entre los valores de este criterio es usada para indicar el número óptimo de clústeres (Milligan y Cooper, 1985).

2.6.21 Índice de McClain

El índice de McClain y Rao (McClain y Rao, 1975) consiste en el ratio de entre el promedio de distancias intraclúster y el número de distancias intraclúster dividido por el número de distancias. Es decir,

$$McClain = \frac{\overline{S}_w}{\overline{S}_b} = \frac{S_w/N_w}{S_b/N_b} \tag{2.34}$$

El valor mínimo del índice es usado para indicar el número óptimo de clústeres.

2.6.22 Índice de Rubin

Friedman y Rubin (1967) propusieron otro criterio basado en el ratio del determinante de la matriz de la suma total de cuadrados y productos cruzados con el determinante de los agrupados en la matriz de conglomerados. Formalmente:

$$Rubin = \frac{det(T)}{det(W_q)} \tag{2.35}$$

Milligan y Cooper (1985) y Dimitriadou et al. (2002) sugieren emplear el valor mínimo de las segundas diferencias entre niveles para seleccionar el número óptimo de clústeres.

2.6.23 Índice de KL

Propuesto por Ratkowsky y Lance (1978), este índice se define como:

$$KL(q) = \left| \frac{DIFF_q}{DIFF_{q+1}} \right| \tag{2.36}$$

donde $DIFF_q = (q-1)^{\frac{2}{p}} tr(W_{q-1}) - q^{\frac{2}{p}} tr(W_q)$. El valor de q que maximiza $KL(q)$ sugiere el número óptimo de clústeres.

2.6.24 Índice de Silueta

En la Sección 2.4.3 discutimos la silueta promedio propuesta por Rousseeuw (1987). Éste se emplea como un índice más para comparar las diferentes aproximaciones. El valor máximo del índice es usado para determinar el número óptimo de clústeres (Kaufman y Rousseeuw, 2009).

2.6.25 Índice de Gap

El estadístico Gap fue propuesto por Tibshirani et al. (2001) y se calcula de la siguiente forma:

$$Gap(q) = \frac{1}{B} \sum_{b=1}^{B} log\left(W_{qb}\right) - log\left(W_q\right) \tag{2.37}$$

donde W_{qb} es la matriz intraclúster definida como en el índice de Hartigan . El número óptimo de clústeres se elige encontrando el menor q tal que:

$$Gap(q) \geq Gap(q+1) - s_{q+1} \tag{2.38}$$

para $q = 1, ..., n-2$.

2.6.26 Índice D

El índice D (Lebart et al., 1995) está basado en la ganancia del clúster de la inercia intraclúster. La inercia intraclúster mide el grado de homogeneidad entre los datos asociados con un clúster. Calcula las distancias comparadas con el punto de referencia

que representa el perfil del clúster (centroide). Se define como:

$$w(P^q) = \frac{1}{q}\sum_{k=1}^{q}\frac{1}{n_k}\sum_{x_i \epsilon C_k} d(x_i, c_k) \tag{2.39}$$

Dadas dos particiones, P^{k-1} compuesta por $k-1$ clústeres y P^k compuesta por k clústeres, la ganancia del clúster de la inercia intraclúster se define como:

$$Gain = w(P^{k-1}) - w(P^k) \tag{2.40}$$

Esta ganancia del clúster debe ser minimizada.

La configuración del clúster óptimo puede ser identificada por el "codo" que corresponde a la disminución significativa de las primeras diferencias de la ganancia del clúster contra el número de clústeres. Este "codo" puede ser identificada por un pico significante en las segundas diferencias de la ganancia del clúster.

2.6.27 Índice de Dunn

El índice de Dunn (Dunn, 1974) define un ratio entre la distancia mínima entre clústeres y la distancia máxima intraclúster. Es decir,

$$Dunn = \frac{\text{mín}_{1\leq i<j\leq q}\, d(C_i, C_j)}{\text{máx}_{1\leq k\leq q}\, diam(C_k)} \tag{2.41}$$

donde $d(C_i, C_j)$ es la función de la diferencia entre dos clústeres C_i y C_j, definida como $d(C_i, C_j) = \text{mín}_{x\epsilon C_i, y\epsilon C_j}$ y $diam(C)$ es el diámetro del clúster, que puede ser considerado una medida de dispersión del clúster. Se puede definir de la siguiente manera:

$$diam(C) = \max_{x,y\epsilon C} d(x, y)$$

Si tenemos clústeres compactos y bien separados, se espera que el diámetro de los clústeres sea pequeño y la distancia entre los clústeres sea grande. Por eso, el índice debe ser maximizado.

2.6.28 Estadístico de Hubert

El estadístico de Hubert (Hubert y Arabie, 1985) es el coeficiente de correlación entre cualquier par de matrices. Cuando dos matrices son simétricas, este índice puede ser escrito como:

$$\Gamma(P, Q) = \frac{1}{N_t}\sum_{i=1}^{n-1} P_{ij}Q_{ij} \tag{2.42}$$

para todo $i < j$ y donde P es la matriz de proximidad o distancias entre las observaciones. Q es una matriz $n \times n$ cuyo elemento (i, j) es igual a la distancia entre los puntos representativos (v_{ci}, v_{cj}) de los clústeres a los que pertenecen los objetos x_i y x_j.

Para $q = 1$ o $q = n$, el índice no está definido. El estadístico de Hubert normalizado es:

$$\overline{\Gamma} = \frac{\sum_{i=1}^{n-1}(P_{ij} - \mu_P)(Q_{ij} - \mu_Q)}{\sigma_P \sigma_Q} \tag{2.43}$$

donde μ_P,μ_Q,σ_P,σ_Q son las respectivas medias y varianzas de las matrices P y Q.

Este índice toma valores entre -1 y 1. Si P y Q no son simétricas, entonces todas las sumas se extienden sobre todas las entradas n^2 y $N_t = n^2$ (Bezdek y Pal, 1998).

Altos valores del Γ normalizado indican la existencia de clústeres compactos. En la gráfica del Γ normalizado contra k (k es el número de clústeres), se busca un "codo" que corresponda a un aumento grande del Γ normalizado al variar k de 2 a $qmax$, donde $qmax$ es el máximo número de clústeres posible. El número de clústeres al cual ocurre el "codo" es un indicador del número de clústeres óptimo (Haldiki et al., 2002).

2.6.29 Índice SD

El índice de validación SD está basado en los conceptos de dispersión promedio para clústeres y separación total entre clústeres. Se computa empleando la siguiente expresión:

$$SDindex(q) = \alpha Scat(q) + Dis(q) \tag{2.44}$$

donde $Scat(q)$ es el promedio de la distancia intraclúster. Un valor pequeño para este término indica clústeres compactos. $Dis(q)$ es la distancia entre clústeres. Y α es un factor de peso igual a $Dis(qmax)$ donde $qmax$ es el máximo número de clústeres de entrada. El número de clústeres, q, que minimiza el índice de arriba puede ser considerado como un valor óptimo para el número de clústeres presentes en la base de datos.

2.6.30 Índice SDbw

El índice de validación *SDbw* está basado en el criterio de compacto y separación entre clústeres.

$$SDbw(q) = Scat(q) + Density.bw(q) \tag{2.45}$$

donde $Density.bw(q)$ es la densidad entre clústeres, definida de la siguiente manera:

$$Density.bw(q) = \frac{1}{q(q-1)} \sum_{i=1}^{q} \left(\sum_{j=1, i \neq j}^{q} \frac{density(u_{ij})}{max(density(c_i), density(c_j))} \right) \tag{2.46}$$

donde u_{ij} es el punto de la mitad del segmento definido por los clústeres centroides c_i y c_j. Y

$$density(u_{ij}) = \sum_{l=1}^{n_{ij}} f(x_l, u_{ij}) \tag{2.47}$$

donde n_{ij} es el número de duplas que pertenecen a los clústeres C_i y C_j. Y $f(x_l, u_{ij})$ es igual a 0 si $d(x, u_i j) > Stdev$ y 1 en caso contrario. Finalmente, $Stdev$ es la desviación estándar promedio de los clústeres; es decir,

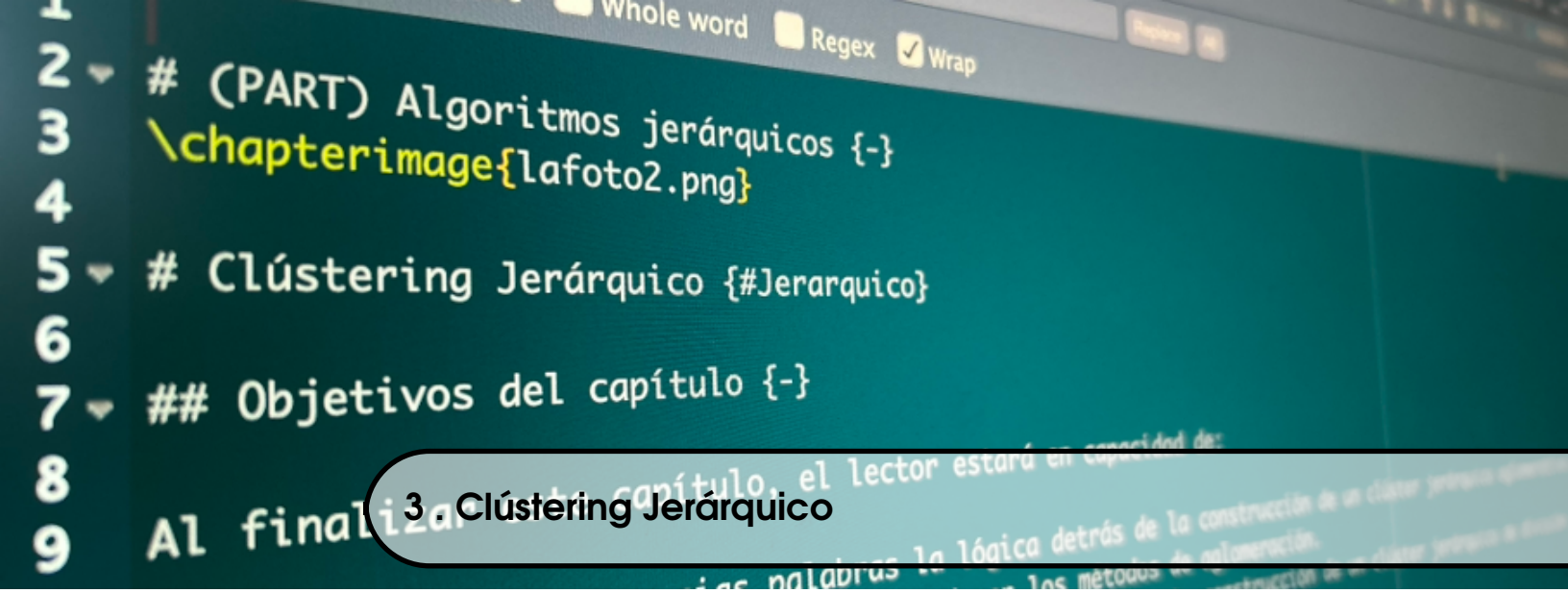
$$Stdev = \frac{1}{q}\sqrt{\sum_{k=1}^{q} \|\sigma^k\|} \tag{2.48}$$

El número de clústeres q que minimiza $SDbw$ es considerado el valor óptimo de clústeres para esa base de datos (Halkidi y Vazirgiannis, 2001).

Si la medida aumenta con el número de clústeres, como en el caso del índice D y el índice de Hubert, entonces encontrar el mínimo o máximo en una gráfica no es suficiente. Un cambio significativo en el valor de la medida, visto como el "codo" en la gráfica, indica los mejores parámetros para el clústering.

Parte II

Algoritmos jerárquicos



Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Explicar en sus propias palabras la lógica detrás de la construcción de un clúster jerárquico aglomerativo (AGNES).
- Explicar en sus propias palabras qué son los métodos de aglomeración.
- Explicar en sus propias palabras la lógica detrás de la construcción de un clúster jerárquico de división (DIANA).

3.1 Introducción

Los algoritmos jerárquicos para la construcción de clústeres tienen un origen en la estadística. Como se mencionó en el Capítulo 2, los algoritmos de clústering **jerárquicos** construyen subconjuntos de clústeres organizados jerárquicamente como un árbol (ver Figura 2.3), siguiendo un método o criterio de aglomeración[1]. Los grupos resultantes de un algoritmo jerárquico por construcción no son mutuamente excluyentes; es decir, se sobreponen.

Existen dos tipos de clústeres jerárquicos: aglomerativo y de división. El algoritmo de **clústering jerárquico aglomerativo** implica comenzar suponiendo que cada individuo es un clúster y se van formando (aglomerando) grupos según su proximidad. Estos algoritmos son conocidos por la sigla HAC (del inglés *Hierarchical Agglomerative Clustering*), la sigla AGNES (del inglés *AGlomerative NESting*) o también con el nombre de aproximación de abajo hacia arriba (*bottom-up*).

Por otro lado, el **clústering jerárquico de división** empieza suponiendo que todos los individuos hacen parte de un único grupo y el algoritmo empieza a dividir los clústeres según la disimilitud entre sus componentes. Estos últimos algoritmos son también conocidos como algoritmos *top-down* o por la sigla DIANA (del inglés *DIvise ANAlysis*). En la Sección 3.2 estudiaremos la intuición detrás del clústering jerárquico aglomerativo, que es de los más empleados en la práctica. Posteriormente, en la Sección 3.4 discutiremos la intuición del clústering jerárquico de división. En el Capítulo 4 estudiaremos cómo aplicar estos conceptos en R.

Antes de continuar, recordemos que en el capítulo pasado habíamos discutido que para agrupar individuos, el proceso de clústering implica:

- Seleccionar una medida de similitud para determinar qué tan parecidos son los individuos al interior de los grupos y qué tan diferentes entre grupos.
- Un algoritmo para formar los clústeres.
- Un criterio para determinar el número óptimo de clústeres.

En este Capítulo nos concentraremos en algoritmos de clústering jerárquico (aglomerativo y de división). Las medidas de similitud fueron discutidas en la Sección 2.2. Así mismo, emplearemos los diferentes criterios para determinar el número de clústeres fueron discutidos en la Sección 2.4.

3.2 La intuición de los métodos aglomerativos

Para los métodos de clústering aglomerativos, no solo es necesario seleccionar una medida de similitud y un criterio para determinar el número óptimo de clústeres. Adicionalmente, será necesario escoger un criterio o método de aglomeración. Esto se discutirá en la sección 3.3.

En general, después de seleccionar una medida de similitud y un método de aglomeración, los pasos para construir conglomerados jerárquicos son:

[1] Es decir, una forma de ir agrupando los individuos a los clústeres.

1. Calcular una matriz de proximidad (o distancias).
2. Iniciar con cada individuo siendo un clúster.
3. Unir los dos clústeres más cercanos.
4. Actualizar la matriz de proximidad.
5. Repetir 3 y 4 hasta que solo quede un clúster.

Para entender el funcionamiento del algoritmo de construcción de clústeres jerárquicos empleemos un ejemplo. Supongamos que contamos con cinco clientes y tenemos sus compras del año pasado en millones de dólares. La pregunta de negocio que queremos responder es: ¿cuántos tipos de clientes podemos identificar en los datos de tal manera que podamos segmentarlos y focalizar nuestras actividades de mercadeo? Es decir, nuestra tarea es construir grupos de clientes para hacer una segmentación. Para hacer el ejemplo más sencillo, supongamos que solo contamos con una variable (característica) de los consumidores: el valor de sus compras el año pasado. En el Cuadro 3.1 se presentan los datos.

Tabla 3.1. Compras de los clientes

Individuo (i)	Compras (Millones de pesos) (x_i)
1	10
2	7
3	28
4	20
5	35

Fuente: datos ficticios.

El primer paso para la construcción de este tipo de clústeres es calcular una matriz de proximidad (o distancias). Esta matriz contiene la distancia entre cada uno de los puntos de los casos (observaciones). Por tanto, es una matriz cuadrada (igual número de filas que de columnas) y simétrica, donde cada fila representa a cada una de las observaciones al igual que cada columna representa una observación. Para construir esta matriz de distancias es importante seleccionar una medida de distancia. Como se discutió en el Capítulo 2, existen diferentes medidas de similitud. Por simplicidad, supongamos que empleamos la distancia euclidiana. Recuerda que por simplicidad hemos supuesto que solo contamos con una variable para realizar la tarea de clusterización. Es decir,

$$x = \left[x_1\right] \tag{3.1}$$

y la distancia euclidiana entre los individuos i y j estará dada por:

$$d(x_i, x_j) = \sqrt{(x_{1,i} - x_{1,j})^2} \tag{3.2}$$

Ahora, procedamos con el primer paso. Calculemos la correspondiente matriz de

proximidad[2]. Por ejemplo, en la primera fila segunda columna tendremos la distancia entre las ventas del cliente 1 y el 2 que se calcula de la siguiente manera:

$$d(x_1, x_2) = \sqrt{(10-7)^2} = 3$$

De manera similar obtenemos la matriz de proximidad[3] reportada en el Cuadro 3.2.

Tabla 3.2. Matriz de proximidad

i	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Fuente: cálculos propios.

En el algoritmo, el segundo paso es iniciar con cada individuo como un clúster. El tercer paso es unir los puntos más "cercanos".

En este caso, según las distancias reportadas en el Cuadro 3.2, la mínima distancia corresponde a los individuos 1 y 2 (resaltado en rojo). Es decir, estos individuos corresponden al primer clúster. Esto lo podemos visualizar en un gráfico conocido como dendrograma (Ver Figura 3.1). Típicamente, en el eje horizontal de un dendrograma se muestran las etiquetas de las observaciones que se están agrupando y en el eje horizontal la distancia (disimilitud) entre las observaciones o grupos. Esta visualización tiene forma de árbol, dada la naturaleza jerárquica del algoritmo. En el árbol, cada nodo representa los "puntos" en que se unen dos grupos y las ramas muestran la relación entre los clústeres.

El cuarto paso es actualizar la matriz de proximidad. Para esto tenemos que considerar el clúster formado por el individuo 1 y 2 como si fuera una sola observación. En otras palabras, estos dos puntos representan un conglomerado. Para esto necesitamos un criterio de construcción de los conglomerados (Criterio de aglomeración), más explícito que el que habíamos usado hasta ahora. Es decir, cómo tratar los grupos que se van formando[4]. Por ejemplo, empleemos el valor mínimo de los individuos que componen el nuevo clúster como el valor del clúster en la matriz de valores originales[5]. La tabla actualizada para las compras de los clientes será como se reporta en el Cuadro 3.3.

La nueva matriz de proximidad se reporta en el Cuadro 3.4.

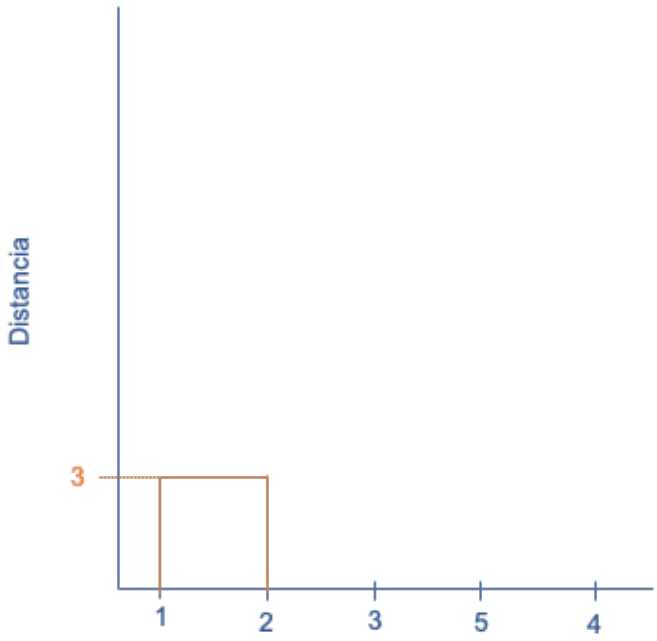
[2]Noten que no estamos estandarizando los datos, esto lo haremos porque solo contamos con una variable. Pero en la práctica es indispensable hacer la estandarización si se emplea más de una variable.

[3]La matriz de proximidad es una matriz cuadrada con tantas filas (y columnas) como observaciones se tengan. Cada celda de la matriz de proximidad tendrá el calculo de la correspondiente distancia de los individuos de la columna i y fila j ($d(x_i, x_j)$).

[4]En la sección 3.3 se discuten los diferentes métodos de aglomeración.

[5]Podríamos haber empleado otros criterios como el máximo, la mediana u otro. En la sección 3.3 se discuten los diferentes métodos de aglomeración.

Figura 3.1. Dendrograma para el clúster aglomerativo- Paso 1



Fuente: elaboración propia.

Tabla 3.3. Compras de los clientes actualizadas - Paso 2

Individuo (i)	Compras (Millones de pesos) (x_i)
1-2	7
3	28
4	20
5	35

Fuente: datos ficticios.

Tabla 3.4. Matriz de proximidad - Paso 2

i	1-2	3	4	5
1-2	3	0	21	13
3	21	0	8	7
4	13	8	0	15
5	28	7	15	0

Fuente: cálculos propios.

Los dos clústeres[6] más cercanos son los individuos 3 y 5. Esto lo podemos visualizar con un dendrograma (ver Figura 3.2).

Procedamos a actualizar las compras de los clientes (Cuadro 3.5) y la matriz de proximidad (Cuadro 3.6).

Tabla 3.5. Compras de los clientes actualizadas - Paso 3

Individuo (i)	Compras (Millones de pesos) (x_i)
1-2	7
3-5	28
4	20

Fuente: datos ficticios.

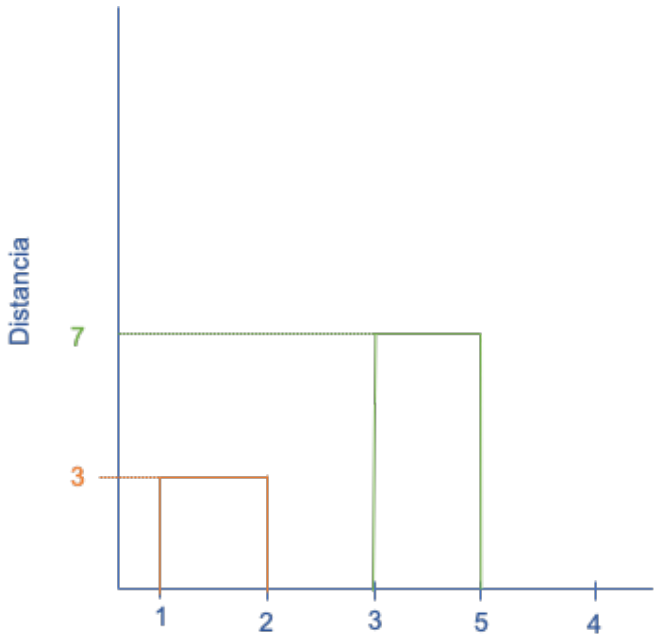
Ahora, los dos conglomerados más cercanos son el conformado por los individuos 3 - 5 y el individuo 4. Esto lo podemos representar en el dendrograma (ver Figura 3.3).

Actualicemos nuevamente las compras de los clientes (Cuadro 3.7) y la matriz de proximidad (Cuadro 3.8).

Ahora, los dos clústeres más cercanos conformarían el grupo padre con todas las

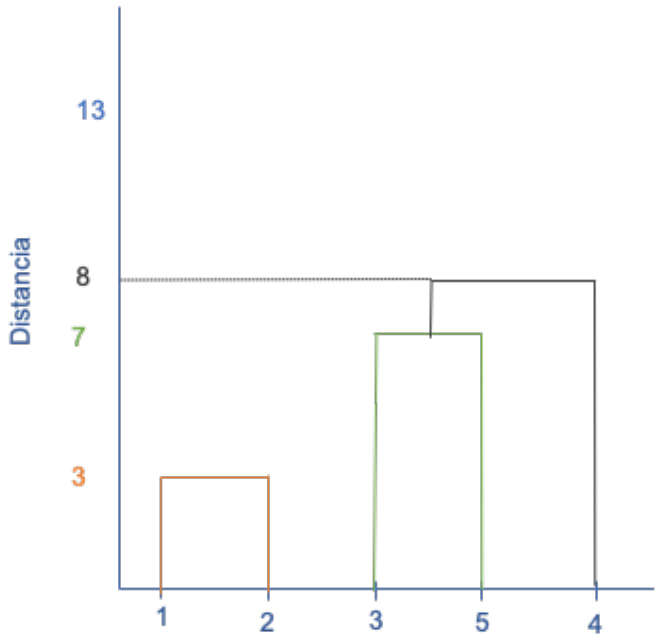
[6]Recuerden que en el primer paso establecimos que cada observación individual sería un clúster.

Figura 3.2. Dendrograma para el clúster aglomerativo- Paso 2



Fuente: elaboración propia.

Figura 3.3. Dendrograma para el clúster aglomerativo- Paso 3



Fuente: elaboración propia.

Tabla 3.6. Matriz de proximidad - Paso 3

i	1-2	3-5	4
1-2	0	21	13
3-5	21	0	8
4	13	8	0

Fuente: cálculos propios.

Tabla 3.7. Compras de los clientes actualizadas - Paso 4

Individuo (i)	Compras (Millones de pesos) (x_i)
1-2	7
3-5-4	20

Fuente: datos ficticios.

Tabla 3.8. Matriz de proximidad - Paso 4

i	1-2	3-5-4
1-2	0	13
3-5-4	13	0

Fuente: cálculos propios.

observaciones. Esto permite completar nuestro dendrograma (Ver Figura 3.4). Recuerda que la altura del nodo nos muestra la distancia entre los clústeres que se fusionan en ese punto y la longitud de la rama indica la similitud entre los conglomerados que se fusionan. De esta manera, el dendrograma (completo) es un buen resumen de los pasos de este algoritmo aglomerativo.

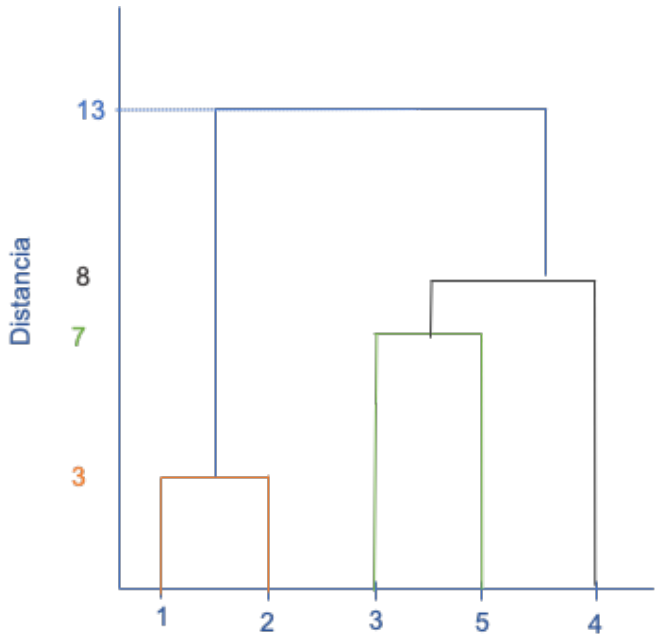
Finalmente, se debe emplear algún criterio para determinar el número de conglomerados. Normalmente, los dendrogramas sugieren el número de clústeres en cada paso, pero ¿dónde cortar?

Existen dos opciones. Emplear un procedimiento heurístico[7] que implica que los árboles son cortados por inspección subjetiva en diferentes niveles. Como el proceso es exploratorio, en algunos casos no existirá mucho problema si esta elección se hace de manera subjetiva. La otra opción es empleando métricas o siluetas como se discutió en el capítulo anterior.

En el Capítulo 4 veremos cómo realizar un proceso de clústering jerárquico aglomerativo en R (R Core Team, 2023). Antes de continuar, veamos las diferentes maneras en

[7] En la ciencia de datos el término heurístico se emplea en el mismo sentido que en la ingeniería. Es decir, una heurística es una aproximación que emplea la experiencia como ayuda para resolver un problema.

Figura 3.4. Dendrograma para el clúster aglomerativo- Paso 4



Fuente: elaboración propia.

que podemos aglomerar las observaciones en este tipo de algoritmos jerárquicos.

3.3 Métodos de aglomeración

Los métodos de agregación (o aglomeración) corresponden a las formas como el algoritmo jerárquico de agrupamiento va incluyendo a cada individuo en un grupo de manera iterativa.

Una idea natural (la que seguimos en nuestro ejemplo intuitivo) es emplear la distancia más pequeña entre las observaciones de los clústeres. Este criterio de aglomeración es conocido como **enlace único** (*Single linkage* en inglés). En este caso, la distancia[8] entre dos clústeres C_i y C_j (la denominaremos $D_{i,j}$) será la distancia más pequeña encontrada entre un individuo del grupo i y otro del clúster j. Es decir, si definimos a x como un vector que contiene las d características (variables que se incluyen para las construcciones de los conglomerados) de un individuo del clúster i ($x \in C_i$) y y como uno del clúster C_j ($y \in C_j$), entonces este criterio implica:

$$D_{i,j} = \min_{x \in C_i, y \in C_j} d(x, y) \tag{3.3}$$

En el panel a de la Figura 3.5 se presenta una representación de esta aproximación.

Un inconveniente de este método es que los grupos tienden a no diferenciarse mucho (a juntarse), debido a que los elementos individuales están cerca unos de otros, aunque muchos de los elementos de cada clúster pueden estar muy distantes entre sí. A este fenómeno se le conoce en la literatura como el **fenómeno de encadenamiento**.

Este problema lleva a pensar que una solución es emplear la distancia máxima y no la mínima. Este método se conoce como **enlace completo** (*Complete linkage* en inglés) (Ver panel b de la Figura 3.5). En otras palabras,

$$D_{i.j} = max_{x \in C_i, y \in C_j} d(x, y) \tag{3.4}$$

Una alternativa a los dos métodos anteriores es emplear la distancia promedio (método de **enlace promedio**) (Ver panel c de la Figura 3.5). Formalmente,

$$D_{i.j} = \frac{\sum_{\forall x \in C_i, \forall y \in C_j} d(x, y)}{n_i \cdot n_j} \tag{3.5}$$

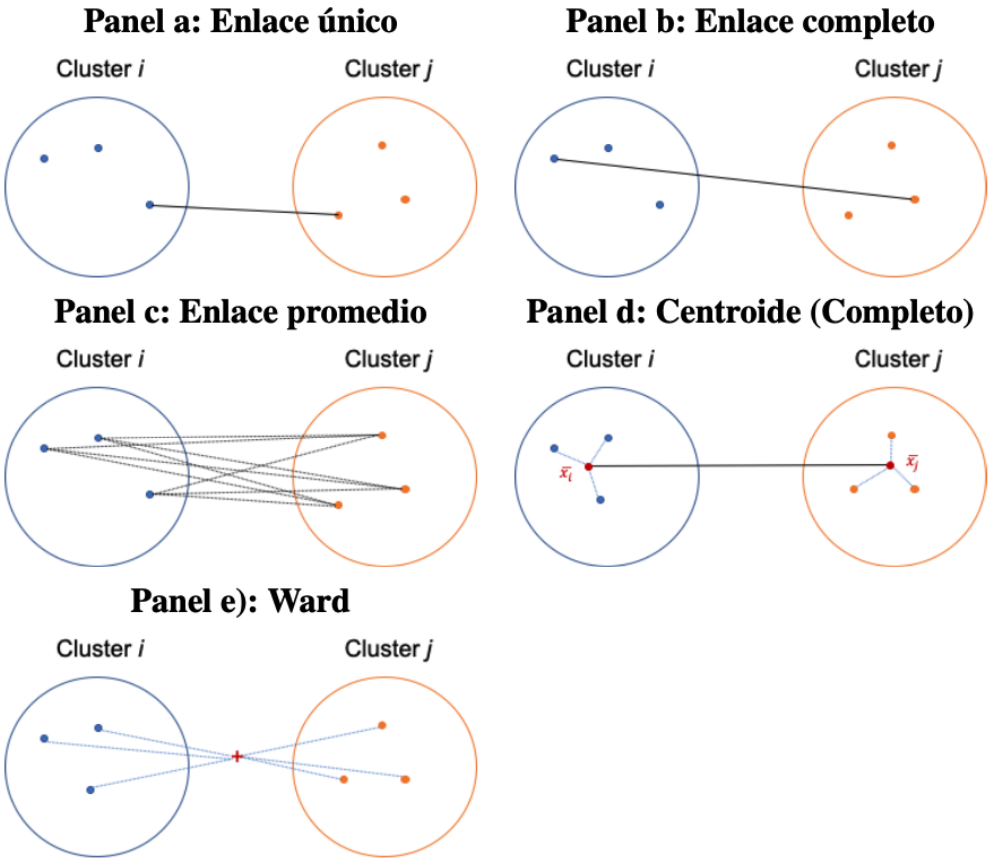
donde n_i y n_j representan el número de elementos en el clúster i (C_i) y clúster j (C_j).

Este método tiene la tendencia a formar grupos con la misma varianza y típicamente varianzas pequeñas. Otro método similar a este es emplear la mediana en vez de la media. Este es el método de **enlace mediano**.

El método del **centroide** define la distancia entre dos clústeres como el cuadrado de la distancia euclidiana entre los **centros de gravedad** de los dos grupos; es decir,

[8] Esta distancia puede ser cualquiera de las discutidas en la sección 2.2.

Figura 3.5. Representación esquemática de los métodos de aglomeración



Fuente: elaboración propia.

entre los vectores que contienen las medias de los dos grupos ($\bar{x_i}$ y $\bar{x_j}$) (Ver panel d de la Figura 3.5). En otras palabras,

$$D_{i,j} = \|\bar{x_i} - \bar{x_j}\|^2 \tag{3.6}$$

Una ventaja de este método es su robustez frente a la presencia de individuos (observaciones) atípicos.

Un criterio que se aparta un poco de los anteriores es el de Ward (Ward Jr, 1963) . Este criterio tiene como premisa minimizar la varianza total dentro del grupo. En cada paso se fusionan el par de clústeres con una distancia mínima de grupos (Ver panel e de la Figura 3.5).

Murtagh y Legendre (2014) proponen una modificación del criterio de Ward Jr (1963) que implica elevar al cuadrado la diferencia encontrada antes de la actualización del grupo. A este método lo llamaremos **Ward.D2** y al de Ward Jr (1963) **WarddD**.

Otro método es el conocido como **McQuitty**, que implica emplear una media ponderada de las diferencias al interior de los conglomerados (intraclúster). Es decir,

$$D_{i,j} = \frac{D_i, k + D_i, l}{2} \tag{3.7}$$

donde el clúster C_j se forma a partir de la agregación de los clústeres C_k y C_l.

En la práctica, es imposible saber cuál de estos cinco métodos de aglomeración generará grupos más plausibles para unos datos determinados. Por eso es común que se intenten diferentes métodos de aglomeración en combinación con diferentes distancias para determinar, por medio de métricas o de las siluetas, el mejor número de conglomerados.

3.4 La intuición de los métodos de división

A diferencia de los algoritmos aglomerativos, que inician considerando a cada uno de los individuos como un posible clúster, este algoritmo se inicia con un único conglomerado que contiene todas las observaciones. El algoritmo implica generar divisiones sucesivas hasta que cada observación forma un grupo independiente. En otras palabras, a diferencia de los métodos aglomerativos que construyen conglomerados de manera ascendente (bottom-up), DIANA sigue un enfoque descendente (top-down).

En el algoritmo DIANA, en cada iteración se selecciona el grupo con mayor diámetro. El diámetro de un clúster se define como la distancia más grande entre dos puntos dentro del clúster (distancia intraclúster). Nota que la distancia puede ser cualquiera de las definidas en la Sección 2.2. Una vez seleccionado el conglomerado de mayor diámetro, se identifica la observación más dispar, que es aquella con mayor distancia promedio respecto al resto de observaciones que forman el clúster. Esta observación inicia el nuevo grupo. Se reasignan las observaciones en función de si están más próximas al nuevo grupo o al resto de la partición, dividiendo así el clúster seleccionado en dos nuevos clústeres.

A diferencia del clústering aglomerativo, en el que hay que elegir un tipo de distancia y un método de aglomeración (*linkage*), en DIANA solo hay que elegir la distancia, no hay método de aglomeración .

Regresemos a nuestro ejemplo. Partamos de la matriz de proximidad que se presenta en el Cuadro 3.1. Recuerda que, en ese caso, tenemos solo una variable y estamos empleando la distancia euclidiana.

El primer paso del algoritmo DIANA es identificar la observación más dispar. En nuestro ejemplo, la disparidad de cada cliente corresponde a la distancia promedio entre las compras de dos clientes dentro del clúster inicial conformado por todas las observaciones (C_0). Noten que para cada cliente tenemos que la disparidad promedio es:

- Cliente 1: (3 + 18 + 10 + 25)/4 = 14
- Cliente 2: (3 + 21 + 13 + 28)/4 = 16.25
- Cliente 3: (18 + 21 + 8 + 7)/4 = 13.5
- Cliente 4: (10 + 13 + 8 + 15)/4 = 11.5
- Cliente 5: (25 + 28 + 7 + 15)/4 = 18.75

En este caso la máxima disparidad corresponde al cliente 5, quien será seleccionado para "liderar" el nuevo clúster. Hasta ahora tendríamos un clúster con el cliente 5 y otro con los clientes de 1 a 4. Para encontrar otro cliente que se unirá al cliente 5 en un grupo, calculamos para cada cliente su disparidad promedio al interior del grupo grande (el conformado por los clientes de 1 a 4) y lo comparamos con la distancia de cada uno de esos clientes con respecto al otro clúster (en este caso el cliente 5 únicamente).

Es decir, tendremos:

- Cliente 1: La disparidad promedio es igual a (3 + 18 + 10)/3 = 10.33. Y la diferencia de la disparidad promedio y la distancia al cliente del otro grupo es de: 10.33 - 25 = -14.67.

- Cliente 2: La disparidad promedio es igual a (3 + 21 + 13)/3 = 12.33. Y la diferencia de la disparidad promedio y la distancia al cliente del otro grupo es de: 12.33 - 28 = -15.67.
- Cliente 3: La disparidad promedio es igual a (18 + 21 + 8)/3 = 15.67. Y la diferencia de la disparidad promedio y la distancia al cliente del otro grupo es de: 15.67 - 7 = 8.67.
- Cliente 4: La disparidad promedio es igual a (10 + 13 + 8)/3 = 10.33. Y la diferencia de la disparidad promedio y la distancia al cliente del otro grupo es de: 10.33 - 15 = -4.67.

En promedio, el cliente 3 es el que más alejado está de los otros clientes de este grupo, al tener una diferencia en la disparidad promedio y la distancia al cliente del otro grupo positiva y más grande (8.67). Así crearemos en este paso inicial dos clústeres, C_1 y C_2. El C_1 formado por los clientes 5 y 3 y otro clúster C_2 con los clientes 1, 2 y 4.

Noten que ahora podemos repetir los cálculos realizados anteriormente para los clientes del grupo C_2 para ver si podemos mover más clientes al grupo C_1. Los resultados se presentan en el Cuadro 3.9.

Tabla 3.9. Paso O del algortimo DIANA

i	Disparidad promedio en el clúster	Disparidad promedio al otro clúster	Diferencia
1	(3 + 10)/2 = 6.5	(18 + 25)/2 = 21.5	-15
2	(3 + 13)/2 = 8	(21 + 28)/2 = 24.5	-16.5
4	(18 + 21)/2 = 19.5	(8 + 15)/2 = 11.5	8

Fuente: cálculos propios.

En este caso aún tenemos un cliente (el Cliente 4) que es muy diferente a su grupo (diferencia positiva y grande). Así que podemos mover al Cliente 4 al clúster C_1. Y repetimos el procedimiento con los clientes que quedan en el grupo C_2.

Tabla 3.10. Paso O del algortimo DIANA: encontrando más candidatos para hacer parte de C_1

i	Disparidad promedio en el clúster	Disparidad promedio al otro clúster	Diferencia
1	3	(18 + 10 + 25)/3 = 17.67	-14.67
2	3	(21 + 13 + 28)/3 = 20.67	-17.67

Fuente: cálculos propios.

Ahora, todos los clientes del C_2 tienen una diferencia negativa con el clúster C_1. Es decir, se parecen más al conglomerado 2 que al 1. Por tanto, no necesitamos mover a más clientes entre grupos. Ahora podemos pasar a la siguiente iteración.

Trabajemos ahora con el clúster $C1$ que es el del diámetro más grande (diámetro de 15 frente a 3 del C_2). En los Cuadros 3.11 y 3.12 se presentan las matrices de proximidad para los dos clústeres que tenemos hasta ahora.

Tabla 3.11. Matriz de proximidad para el clúster C_1

i	3	4	5
3	0	8	7
4	8	0	15
5	7	15	0

Fuente: cálculos propios.

Tabla 3.12. Matriz de proximidad para el clúster C_2

i	1	2
1	0	3
2	3	0

Fuente: cálculos propios.

Ahora repliquemos el procedimiento anterior al C_1. Primero busquemos el cliente más diferente al interior de este clúster empleando la disparidad promedio más grande (Ver Cuadro 3.13).

Tabla 3.13. Paso 1 del algortimo DIANA: escogiendo el cliente diferente

Cliente	Disparidad promedio en el clúster
3	(8 + 7)/2 =7.5
4	(8 + 15)/2 =11.5
5	(7 + 15)/2 =11

Fuente: cálculos propios.

De acuerdo con estos resultados, el cliente 4 puede conformar el nuevo clúster (C_3). Ahora miremos cuál otro cliente se le puede unir en el nuevo clúster. En el Cuadro 3.14 vemos que todas las diferencias son negativas.

Tabla 3.14. Paso 1 del algortimo DIANA: encontrando más clientes para unirse al C_3

Cliente	Disparidad promedio en el clúster	Disparidad promedio al otro clúster	Diferencia
3	7	8	-1
5	7	15	-8

Fuente: cálculos propios.

Así, podemos dar por terminado este paso. Es decir, por ahora tenemos tres clústeres:

- C_1: Clientes 3 y 5.
- C_2: Clientes 1 y 2.
- C_3 Cliente 4.

En la siguiente iteración buscaremos el grupo con el diámetro más grande (el C_1) y en la tercera iteración estaríamos trabajando con el C_2, hasta que terminamos con clústeres de un solo elemento.

Este ejemplo te permite entender la intuición detrás del algoritmo jerárquico de división. Si deseas conocer el detalle técnico detrás de DIANA puedes consultar el capítulo 6 de Kaufman y Rousseeuw (2009).

3.5 Comentarios finales

En este Capítulo, hemos discutido dos tipos de algoritmos para construir grupos de carácter jerárquicos. Los algoritmos aglomerativos que parten de suponer que cada uno de los individuos es un clúster y empleando un método aglomerativo[9] y una medida de distancia[10], se empiezan a conformar conglomerados de individuos.

Para ilustrar los algoritmos jerárquicos aglomerativos (AGNES), empleamos un ejemplo en el que usamos el método de aglomeración de **enlace único** (*Single linkage*) y la distancia **euclidiana**. En la Figura 3.6 se presentan los pasos que realizamos en nuestro ejemplo para construir los clústeres.

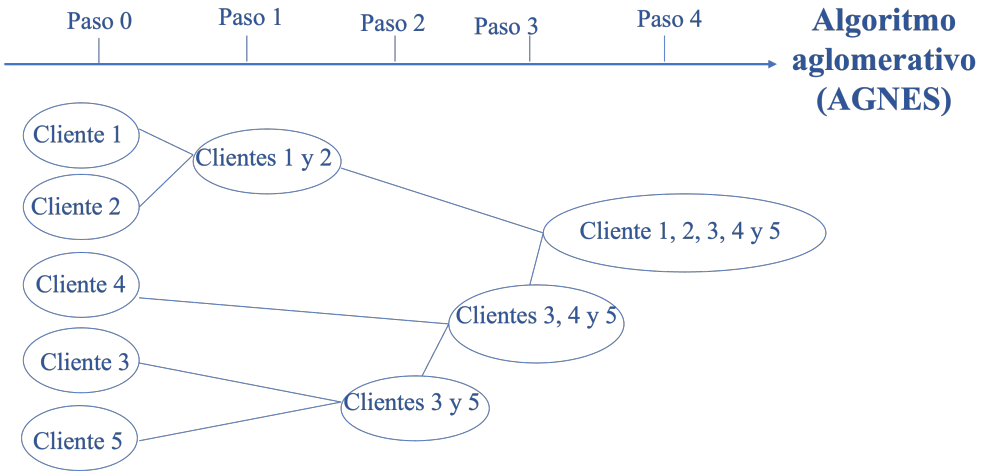
En general, es importante recordar que en los AGNES debemos escoger uno de los siguientes 8 métodos de aglomeración:

- **Enlace único** (*Single linkage*),
- **Enlace completo** (*Complete linkage*),
- **Enlace promedio**,
- **Enlace mediano**,
- **Centroide**,

[9] Los métodos considerados fueron: **enlace único** (*Single linkage*), **enlace completo** (*Complete linkage*), **enlace promedio**, **enlace mediano**, **centroide**, **Ward**, y **McQuitty**.

[10] Las medidas de distancia que estudiamos fueron: **distancia euclidiana**, **Mahalanobis**, **Manhattan**, y **Minkowski**.

Figura 3.6. Pasos del ejemplo empleando el algoritmo aglomerativo (AGNES)



Fuente: elaboración propia.

- **Ward.D**,
- **Ward.D2**, y
- **McQuitty**.

Adicionalmente, es importante escoger una medida de distancia de las siguientes:

- **Distancia euclidiana**
- **Mahalanobis**
- **Manhattan**
- **Minkowski**

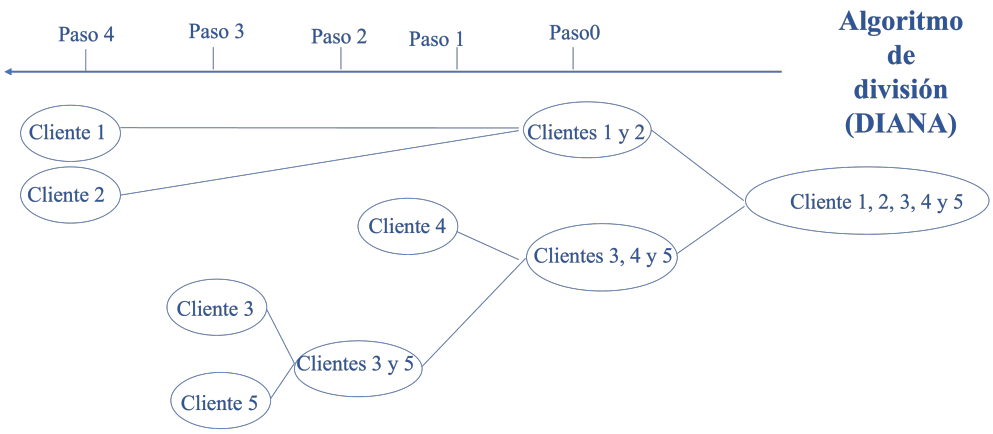
Es decir, en la práctica podremos comparar diferentes AGNES que empleen diferentes algoritmos de aglomeración y medidas de similitud.

En este Capítulo también consideramos los algoritmos jerárquicos de división (DIANA). En este caso, no es necesario escoger un método de aglomeración pero si una medida de distancia. Los DIANA parten de un clúster con todos los individuos y van dividiendo a los individuos en grupos, hasta llegar al punto en que todos los individuos representan un clúster aparte.

En nuestro ejemplo, empleamos la distancia euclidiana para construir los clústeres de manera jerárquica, como se representa en la Figura 3.7.

En nuestros dos ejemplos (AGNES y DIANA) construimos todos los clústeres posibles. Recuerden que en la práctica debemos escoger el número óptimo de grupos (como lo discutimos en el Capítulo 2). En este Capítulo no nos encargamos de ese paso esencial, pues enfatizamos en cómo funciona cada una de estas filosofías de métodos jerárquicos de clústering. En el Capítulo 4 estudiaremos la forma de implementar estos algoritmos en R (R Core Team, 2023) y cómo escoger el número óptimo de conglome-

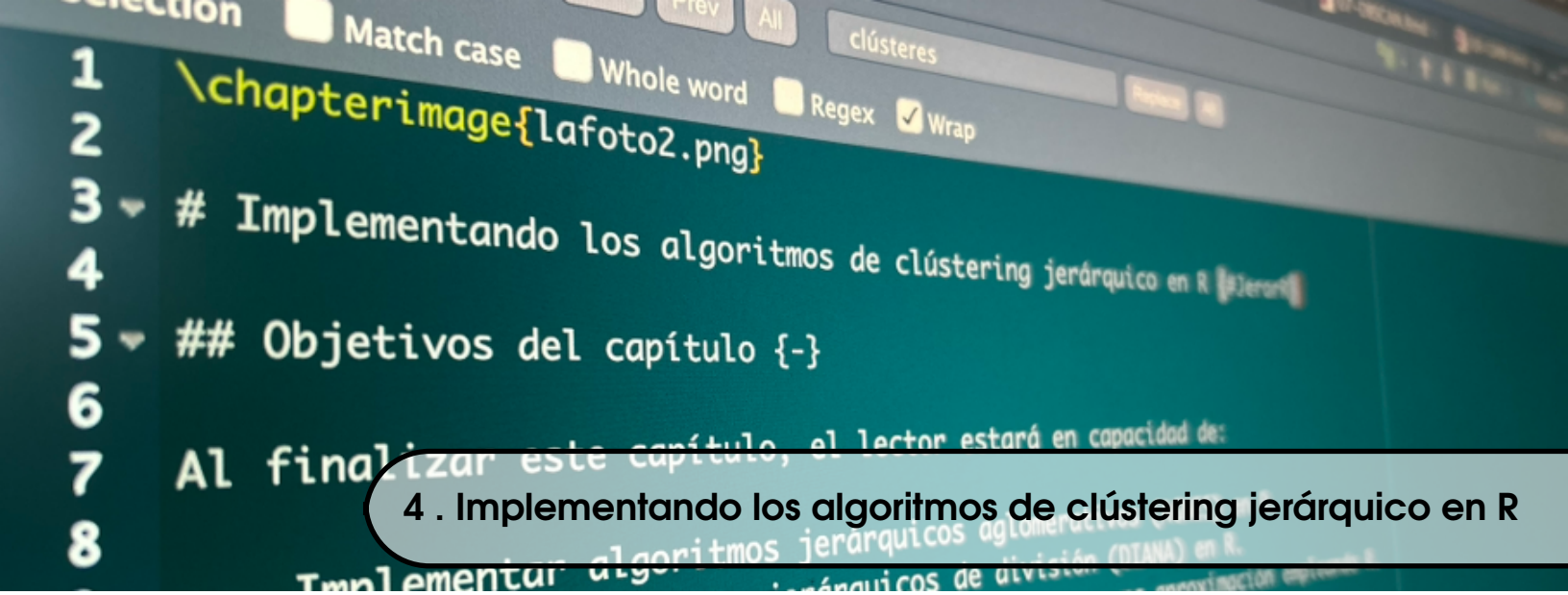
Figura 3.7. Pasos del ejemplo empleando el algoritmo de división (DIANA)



Fuente: elaboración propia.

rados.

Antes de continuar al siguiente capítulo es importante mencionar que el análisis de clúster empleando métodos jerárquicos también es conocido con la sigla **HCA** del inglés *Hierarchical Cluster Analysis*.



4 . Implementando los algoritmos de clústering jerárquico en R

Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Implementar algoritmos jerárquicos aglomerativos (AGNES) en R.
- Implementar algoritmos jerárquicos de división (DIANA) en R.
- Seleccionar el número óptimo de clústeres para una aproximación empleando R.
- Construir dendrogramas en R.

4.1 Introducción

Como se mencionó en el Capítulo 3, los algoritmos jerárquicos para la construcción de clústeres tienen un origen en la estadística. En dicho capítulo discutimos como el **HCA** (por sus siglas en inglés: *Hierarchical Cluster Analysis*) construye una jerarquía de grupos a partir de los elementos individuales. Existen dos estrategias principales para realizar HCA:

- **HCA Aglomerativo**: También conocida como AGNES o método "bottom-up", comienza con cada observación en su propio clúster y va fusionando iterativamente clústeres más similares, hasta llegar a un único grupo con todos los individuos en la muestra.
- **HCA de división**: Conocido como método DIANA o "top-down", parte de un clúster que contiene todas las observaciones y lo va dividiendo de forma recursiva en conglomerados más pequeños en cada paso.

En este Capítulo nos concentraremos en cómo implementar en R un HCA. Primero nos concentraremos en la aproximación aglomerativa, cómo construir dendrogramas y cómo seleccionar el número óptimo de agrupaciones empleando los métodos estudiados en la Sección 2.4. Posteriormente, concentraremos la atención al **HCA** de división (DIANA).

4.2 Los datos y la pregunta de negocio

Para desarrollar este ejemplo emplearemos datos provistos por Shah (2021)[1]. Los datos corresponden a 660 clientes de una tarjeta de crédito. Los datos están disponibles en el archivo `Credit_Card_Customer_Data.csv`[2]. Esta base de datos contiene los siguientes 7 *features* (variables o características):

- `Sl_No`: Número de cliente (sirve como índice de cada cliente en la base de datos).
- `Customer.Key`: Número de identificación del cliente.
- `Avg_Credit_Limit`: Límite de crédito promedio (no se especifica la moneda, supondremos que está en dólares).
- `Total_Credit_Cards`: Número total de tarjetas de crédito que tiene el cliente.
- `Total_visits_bank`: Total de visitas del cliente a la sucursal del banco (físicamente) emisor de la tarjeta de crédito.
- `Total_visits_online`: Total de visitas a la sucursal virtual (la plataforma electrónica) del banco emisor de la tarjeta de crédito.
- `Total_calls_made`: Total de llamadas realizadas por el cliente a la banca telefónica del banco emisor de la tarjeta de crédito.

Nuestro objetivo es emplear los datos para responder la pregunta de negocio: *"Con el fin de mejorar la satisfacción del cliente, brindar una atención más personalizada a*

[1] La base de datos fue descargada del siguiente enlace: https://www.kaggle.com/datasets/aryashah2k/credit-card-customer-data?select=Credit+Card+Customer+Data.csv.

[2] Los datos se pueden descargar de la página web del libro: http://www.icesi.edu.co/editorial/intro-clustering/.

los clientes y optimizar el gasto en mercadeo, ¿cómo segmentamos a nuestros clientes?" Nuestra tarea, entonces, es crear clústeres de clientes para focalizar las acciones de mercadeo de la sucursal bancaria.

4.3 Exploración y preparación de los datos

Carguemos los datos y exploremos su clase.

```
# carga de archivo
datos_originales <- read.csv("Credit_Card_Customer_Data.csv")

# estructura de los datos cargados
str(datos_originales)
```

Noten que las dos primeras variables no son útiles para nuestro análisis: `Sl_No` (número de cliente) y `Customer.Key` (número de identificación del cliente). Removamos estas variables para iniciar nuestro análisis.

```
# eliminando las dos primeras filas
datos <- datos_originales[, -c(1, 2)]
```

Recuerda que siempre es necesario realizar una exploración de los datos. Puedes constatar que no tenemos datos perdidos. Exploremos rápidamente los datos empleando los paquetes *ggplot2*[3] (Wickham, 2016) y *GGally* (Schloerke et al., 2023) para tener una visualización más agradable como la que se presenta en la Figura 4.1.

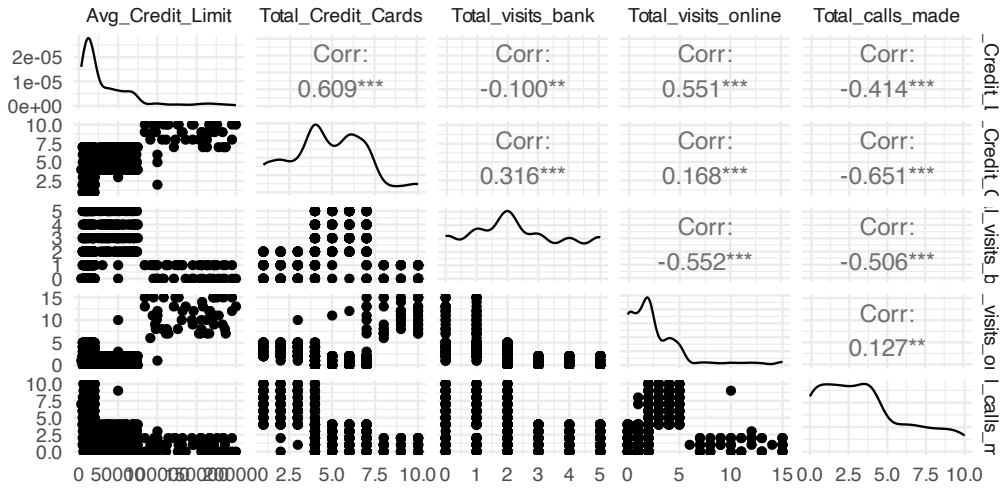
```
# Cargar paquetes
library(ggplot2)

library(GGally)

# Graficar todas las variables de la base
ggpairs(datos) + theme_minimal()
```

La Figura 4.1 no revela mucho; los datos no presentan una forma natural de organizarse en grupos. Esto es lo más común en la práctica, al ojo no es fácil encontrar los grupos de individuos. Podríamos trabajar un poco más con las visualizaciones pero no llegaremos muy lejos. Procedamos a construir los clústeres jerárquicos, pero antes tendremos que estandarizar los datos.

Noten que las variables tienen rangos diferentes. Por eso es importante centrarlas y, además, unificar la volatilidad de los datos. Es decir, debemos estandarizar todas las variables[4] quitándoles la respectiva media (la media de la respectiva columna) y dividirla por la desviación estándar.

[3] Para una introducción a este paquete se puede consultar Alonso y Largo (2023).
[4] Algunos autores incorrectamente llaman a esto normalizar.

Figura 4.1. Relación entre todas las variables de la base de datos



Fuente: elaboración propia.

Para estandarizar nuestro objeto de clase **data.frame** podemos emplear la función **scale()** que está en el paquete base de R. Esta función típicamente incluye los siguientes argumentos:

scale(x, center = TRUE, scale = TRUE)

donde:

- **x**: Es un objeto de clase **matrix** o **data.frame**.
- **center**: Un valor lógico que establece si las columnas de la matriz o **data.frame** deben centrarse o no. Es decir, si se debe restar la media o no. El valor por defecto es **center = TRUE**. En otras palabras, la función centra las columnas a menos que se le indique lo contrario.
- **scale**: Un valor lógico que establece si las columnas de la matriz deben "escalarse" o no. Es decir, si se debe dividir cada valor centrado por la desviación estándar de la correspondiente columna. El valor por defecto es **scale = TRUE**. Así, la función escala las columnas a menos que se le indique lo contrario.

Esta función produce los datos estandarizados en clase **matrix**.

Entonces, estandaricemos el objeto **data.frame** `datos` con la función **scale()** y convirtamos de nuevo el resultado a un objeto de clase **data.frame**.

```
# Estandarizar los datos
datos_est <- as.data.frame(scale(datos))

# Chequear clase de las variables
library(dplyr)
glimpse(datos_est)
```

```
## Rows: 660
## Columns: 5
## $ Avg_Credit_Limit    <dbl> 1.7388680, 0.4099816, 0.4099816, -0.1215730, 1.738~
## $ Total_Credit_Cards  <dbl> -1.2482780, -0.7869883, 1.0581707, 0.1355912, 0.59~
## $ Total_visits_bank   <dbl> -0.8597985, -1.4726139, -0.8597985, -0.8597985, -1~
## $ Total_visits_online <dbl> -0.5470748, 2.5186084, 0.1341882, -0.5470748, 3.19~
## $ Total_calls_made    <dbl> -1.2505889, 1.8904250, 0.1454173, 0.1454173, -0.20~
```

El siguiente paso, antes de emplear los algoritmos jerárquicos de clústering, es calcular la matriz de proximidad como lo realizamos en el Capítulo 3. Esto implicaría crear un objeto con las distancias.

Esto se puede hacer empleando la función **dist()** que está en el paquete central de R. El argumento **method** determina el tipo de medida de distancia que se quiere emplear (Ver Sección 2.2). Las opciones para este argumento son: "euclidean", "maximum", "manhattan", "canberra", "binary" o "minkowski". Por defecto, **method = "euclidean"**. Así mismo, está el argumento **p** para establecer la potencia que se desea emplear para la distancia de **Minkowski**, por defecto **p = 2**.

Esta función creará una matriz que tiene 660 filas y columnas que contiene todas las distancias entre los individuos teniendo en cuenta las 5 variables, Es decir, empleando la notación que usamos en los capítulos anteriores tenemos que:

$$x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} \tag{4.1}$$

y cada celda de la matriz de proximidad tendrá el cálculo de la correspondiente distancia de los individuos de la columna i y fila j ($d(x_i, x_j)$)[5].

Noten que la matriz de distancias (proximidad) puede llegar a ser muy grande si tenemos muchas observaciones, lo que puede implicar un uso no eficiente de la memoria RAM, al almacenar un objeto tan grande. Una opción más común en la práctica de los científicos de datos, es emplear la posibilidad de calcular la matriz de distancias desde la misma función que se emplee para hacer el algoritmo de clústering. Esto puede hacer más eficiente nuestro uso de la memoria, en especial cuando estemos trabajando con millones de individuos. En este libro adoptaremos esta segunda aproximación siempre que sea posible; es decir, siempre que la función lo permita.

Empleemos por ahora la función **dist()** para calcular la matriz de proximidad de nuestros datos empleando la distancia euclidiana.

```
# Calcular de la matriz de distancias empleando la distancia euclidiana
datos_dist <- dist(datos_est)

# noten que esto es equivalente a datos_dist <- dist(datos_est, method =
# 'euclidean')
```

[5]En la Sección 2.2 se discutieron todas las posibles formas que tomará la función $d(x_i, x_j)$.

4.4 Construcción de clústeres jerárquicos aglomerativos y dendrograma

El siguiente paso implica emplear el algoritmo para la creación de los clústeres jerárquicos aglomerativos y visualizarlos con un dendrograma. Recordemos que esto implica seleccionar un método de aglomeración y una medida de similitud (tipo de distancia), para la construcción de los conglomerados.

La función **hclust()** de la base de R permite emplear los diferentes métodos o criterios de aglomeración discutidos en la sección 3.3. Esta función típicamente incluye los siguientes argumentos:

hclust(d, method = "complete")

donde:

- **d**: Es una matriz de distancias de clase *dist*.
- **method**: El método de aglomeración que se desea emplear (Ver Sección 3.3). Las opciones son: "ward.D" (Ward Jr, 1963) , "ward.D2" (Murtagh y Legendre, 2014), "single" (enlace único), "complete" (enlace completo), "average" (enlace medio), "mcquitty", "median" (enlace mediano) o "centroid". Por defecto, **method = "complete"**.

Construyamos el clúster jerárquico empleando el **método de agregación del centroide** (y la distancia euclidiana). Y guardemos los resultados en el objeto `HCA_centroide`.

```
# construcción de clústeres jerárquicos con método de aglomeración centroide y
# distancia euclidiana

HCA_centroide <- hclust(datos_dist, method = "centroid")
# Imprimir el objeto
HCA_centroide
```

```
##
## Call:
## hclust(d = datos_dist, method = "centroid")
##
## Cluster method   : centroid
## Distance         : euclidean
## Number of objects: 660
```

```
# Constatar la clase del objeto
class(HCA_centroide)
```

```
## [1] "hclust"
```

Antes de discutir la visualización de los resultados, recuerda que podemos emplear otros métodos de aglomeración cambiando el argumento **method** de la función **hclust()** . Por ejemplo, para emplear el método aglomeración del **enlace promedio**

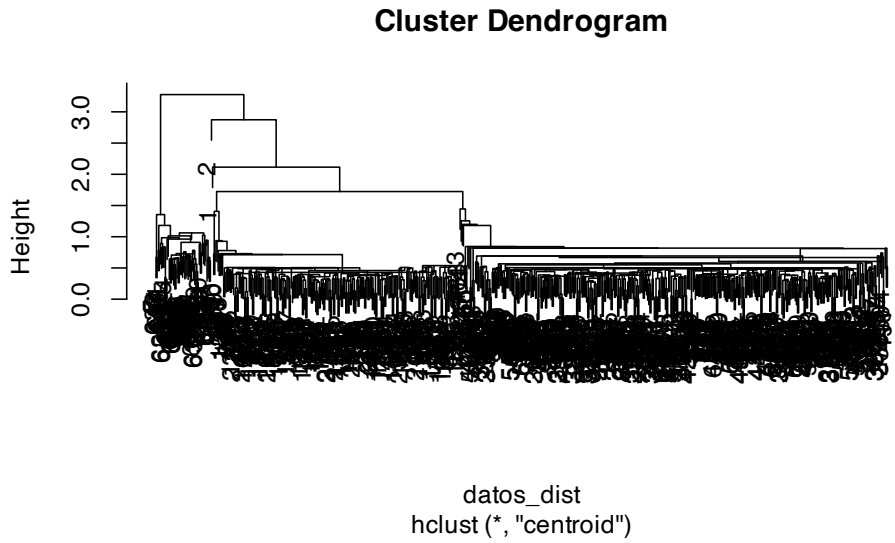
podemos emplear el siguiente código.

```
# construcción de clústeres jerárquicos con método de aglomeración enlace
# promedio y distancia euclidiana
HCA_promedio <- hclust(datos_dist, method = "average")
```

Regresando a nuestra HAC con el método de aglomeración centroide, hemos creado un objeto de clase **HCA_centroide** que tiene toda la información necesaria para construir el dendrograma. Podemos visualizar los pasos del algoritmo jerárquico empleando la función **plot()** de la base de R.

```
# Construir del dendrograma
plot(HCA_centroide)
```

Figura 4.2. Dedrograma para el HCA emplendo el método de aglomeración de centroide y distancia euclidiana



Fuente: cálculos propios.

Como lo estudiamos en el Capítulo 3, un dendrograma es una representación gráfica en forma de árbol que muestra cómo el algoritmo jerárquico (en este caso aglomerativo) fue agrupando las observaciones paso a paso. La altura de las ramas representa la medida de similitud entre los elementos o grupos; ramas más largas indican mayor diferencia o distancia, mientras que ramas más cortas indican mayor similitud o proximidad.

La visualización que reportamos en la Figura 4.2 no es muy estética. Podemos jugar un poco con esta poniendo todos los nombres de las observaciones al mismo nivel empleando el argumento **hang** de la función **plot()**.

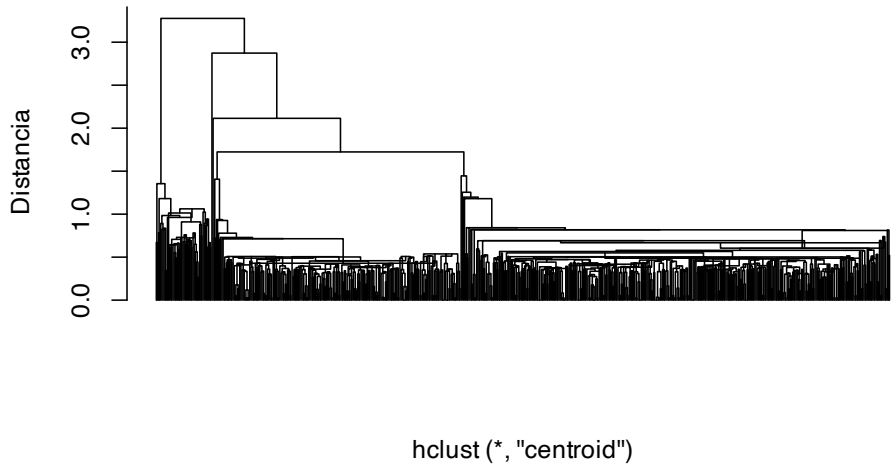
```
plot(HCA_centroide, hang = -1)
```

Y sigamos jugando con esta visualización cambiando el tamaño de las etiquetas de los individuos (argumento **cex**) y poniendo títulos (argumento **main**).

```
plot(HCA_centroide, main = "Dendrograma método centroide", hang = -1, cex =
↪  0.2,
    ylab = "Distancia")
```

De pronto tenemos muchas observaciones como para ponerle las etiquetas a todas. Podemos omitir las etiquetas con el argumento **labels = FALSE** (Ver Figura 4.3)[6].

Figura 4.3. Otra versión del dedrograma para el HCA emplendo el método de aglomeración de centroide y distancia euclidiana



Fuente: cálculos propios.

También podríamos visualizar el dendrograma horizontalmente con el argumento **horiz = TRUE**

```
plot(as.dendrogram(HCA_centroide), main = "Dendrograma método centroide", type
↪  = "rectangle",
    horiz = TRUE, cex = 0.2, xlab = "Distancia")
```

Por ahora no exploraremos más visualizaciones. Más adelante lo haremos, si te sientes más cómodo con visualizaciones siguiendo la lógica del paquete *ggplot2* (Wickham, 2016) , puedes explorar el paquete *ggdendro*, (de Vries y Ripley, 2024) que agrega una geometría para poder emplear toda la estructura de *ggplot2* para construir visualizaciones por capas.

[6]No se provee el código para esta visualización, ¡intenta reproducirla!

4.5 Escogiendo el número óptimo de clústeres

Hasta ahora hemos agrupado de manera jerárquica los clientes del banco. El AGNES, empleando el centroide como método de aglomeración, nos ha permitido ir agrupando a los clientes de manera iterativa. Ahora la pregunta es: ¿en cuántos clústeres sería conveniente agrupar los individuos? Es decir, ¿cuál es el número de conglomerados que permiten crear los grupos que sean más diferentes entre grupos y que al interior del grupo los individuos sean lo más parecidos (de acuerdo con los 5 *features* que tenemos)? O visto de otra manera, ¿por dónde deberíamos podar el dendrograma?.

En la Sección 2.4 discutimos cómo responder esta pregunta empleando una batería de indicadores o solo la silueta promedio. A continuación, veremos cómo implementar en R estos métodos para escoger el número óptimo de clústeres.

4.5.1 Empleando batería de métricas

Como se mencionó en la sección 2.4, una forma de determinar el número de clústeres es emplear los 30 indicadores de uso más común. Estos se encuentran disponibles en el paquete *NbClust*, (Charrad et al., 2014) empleando la función **NbClust()**. Esta función típicamente incluye los siguientes argumentos:

NbClust(data = NULL, diss = NULL, distance = "euclidean", min.nc = 2, max.nc = 15, method = NULL, index = "all")

donde:

- **data**: Es un objeto de clase matriz o **data.frame**.
- **diss**: El tipo de medida de distancia que se quiere emplear (ver Sección 2.2). Las opciones son: "euclidean", "maximum", "manhattan", "canberra", "binary" o "minkowski". Por defecto, **distance = "euclidean"**.
- **min.nc**: El número mínimo de clústeres que se desea evaluar. Por defecto, **min.nc = 2**.
- **max.nc**: El número máximo de clústeres que se desea evaluar. Por defecto, **max.nc = 15**.
- **method**: Los métodos de aglomeración que se desea emplear (ver Sección 3.3). Las opciones son: "ward.D" (Ward Jr, 1963)). , "ward.D2" (Murtagh y Legendre, 2014), "single" (enlace único), "complete" (enlace completo), "average" (enlace medio), "mcquitty", "median" (enlace mediano) o "centroid".
- **index**: El índice que se desea calcular. Las opciones son: "kl", "ch", "hartigan", "ccc", "scott", "marriot", "trcovw", "tracew", "friedman", "rubin", "cindex", "db", "silhouette", "duda", "pseudot2", "beale", "ratkowsky", "ball", "ptbiserial", "gap", "frey", "mcclain", "gamma", "gplus", "tau", "dunn", "hubert", "sdindex", "dindex", "sdbw", "all" (todos los índices excepto GAP, Gamma, Gplus y Tau), "alllong" (todos los índices incluido Gap, Gamma, Gplus y Tau) (Ver Sección 2.4.4). Por defecto, **index = "all"**.

Empleemos esta función para evaluar el número de clústeres para nuestro proble-

ma usando 26 indicadores[7]. Recuerden que en este caso hemos escogido emplear la distancia euclidiana y como método de aglomeración el centroide y el enlace promedio. En este caso emplearemos los datos estandarizados y no la matriz de proximidad; es decir, emplearemos el objeto `datos_est`.

```
# Instalar el paquete si no se tiene

# install.packages('NbClust')

# Cargar el paquete
library(NbClust)
# calculamos las 30 métricas
res_centroid <- NbClust(datos_est, distance = "euclidean", min.nc = 2, max.nc =
↪  10,
    method = "centroid", index = "all")

# Ver todos los compartimientos del objeto construido
attributes(res_centroid)
```

En el compartimiento `All.index` del objeto `res_centroid` puedes encontrar el valor de los 26 índices para los números de clústeres de 2 a 10. En el compartimiento `Best.nc` se encuentra el número de grupos óptimo según cada uno de los indicadores. Para nuestro caso:

```
# Ver todos los indicadores por número de clúster
res_centroid$All.index
# Ver el número de clústeres óptimo según cada uno de los indicadores
res_centroid$Best.nc
```

Además, podemos graficar los resultados empleando la función **fviz_nbclust()** del paquete *factoextra* (Kassambara y Mundt, 2020) . Esta función emplea el paquete *ggplot2* (Wickham, 2016) para realizar las gráficas[8].

Procedamos a visualizar los resultados.

```
# Instalar el paquete si se necesita

# install.packages('factoextra')

# cargar el paquete
library(factoextra)
# visualizando las métricas (sigue leyendo si te sale un mensaje de error)
fviz_nbclust(res_centroid) + theme_minimal()
```

[7] En este ejemplo no emplearemos los indices GAP, Gamma, Gplus y Tau por demandar mucho tiempo su cálculo. Puedes probar que el resultado cambian si incluyes estos 4 indicadores.

[8] En Alonso y Largo (2023) se puede encontrar una breve introducción a la creación de visualizaciones con el paquete *ggplot2*.

Si estás empleando una versión de R superior a 4.2, tendrás un mensaje de error. La función **fviz_nbclust()** tiene un problema que está bien documentado en la comunidad de R[9] y a la fecha de la publicación de este libro aún no ha sido solucionado. Para resolver el problema puedes emplear la siguiente función[10]:

```
nueva_fviz_nbclust <- function(x, print.summary = TRUE, barfill = "steelblue",
↪  barcolor = "steelblue") {
    best_nc <- x$Best.nc
    best_nc <- as.data.frame(t(best_nc), stringsAsFactors = TRUE)
    best_nc$Number_clusters <- as.factor(best_nc$Number_clusters)

    ss <- summary(best_nc$Number_clusters)
    cat("Entre todos los índices: \n====================\n")
    for (i in 1:length(ss)) {
        cat("*", ss[i], "proponen ", names(ss)[i], "como el mejor número de
        ↪  clústeres\n")
    }
    cat("\nConclusión\n=========================\n")
    cat("* De acuerdo con la mayoría, el mejor número de clústeres es is ",
    ↪  names(which.max(ss)),
        ".\n\n")

    df <- data.frame(Number_clusters = names(ss), freq = ss, stringsAsFactors =
↪  TRUE)
    p <- ggpubr::ggbarplot(df, x = "Number_clusters", y = "freq", fill =
↪  "steelblue",
        color = "steelblue") + ggplot2::labs(x = "Número de clústeres k", y =
        ↪  "Frecuencia entre todos los índices",
        title = paste0("Número óptimo de clústeres - k = ",
        ↪  names(which.max(ss))))
    p
}
```

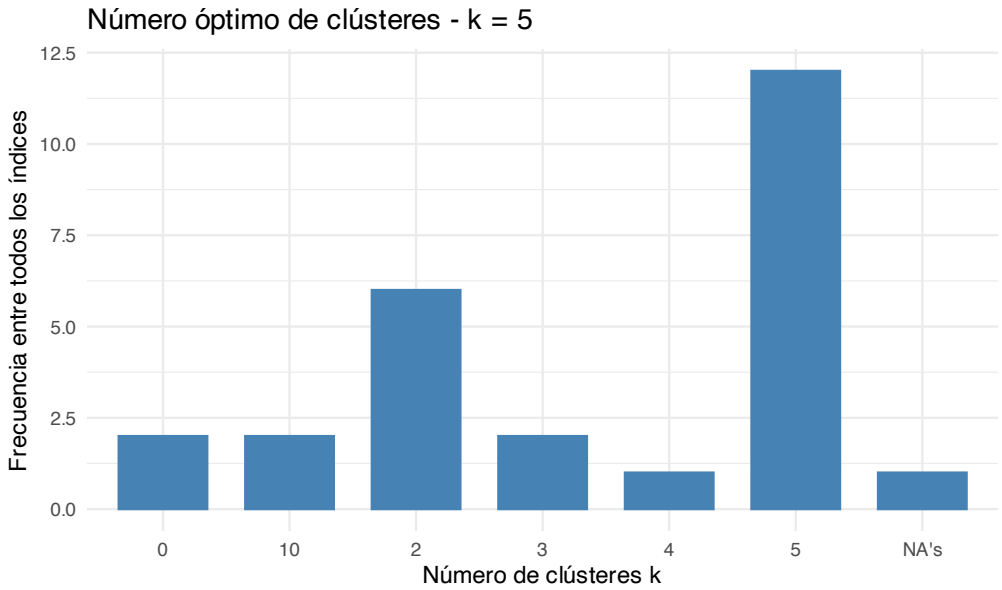
```
## Entre todos los índices:
## ====================
## * 2 proponen  0 como el mejor número de clústeres
## * 6 proponen  2 como el mejor número de clústeres
## * 2 proponen  3 como el mejor número de clústeres
## * 1 proponen  4 como el mejor número de clústeres
## * 12 proponen  5 como el mejor número de clústeres
## * 2 proponen  10 como el mejor número de clústeres
## * 1 proponen  NA's como el mejor número de clústeres
##
```

[9] Por ejemplo, se puede ver la discusión en el siguiente enlace: https://community.rstudio.com/t/rstudio-error-with-fviz-nbclust-for-nbclust-results/176083/2.

[10] Esta función tiene una pequeña modificación en el condicional para resolver el problema.

```
## Conclusión
## ========================
## * De acuerdo con la mayoría, el mejor número de clústeres es is  5 .
```

Figura 4.4. Votación de la batería de métricas por el número óptimo de clústeres para el HCA empleando el método de aglomeración de centroide y distancia euclidiana



Fuente: cálculos propios.

De acuerdo con los resultados reportados en la Figura 4.4, 12 de los 30 métodos sugieren 5 clústeres, mientras que 6 índices sugieren 2 conglomerados.

Ustedes pueden replicar el resultado para los otros métodos de aglomeración. Por ejemplo, para el método del enlace promedio el código sería:

```
# calculamos las 30 métricas
res_media <- NbClust(datos_est, distance = "euclidean", min.nc = 2, max.nc =
↪  10,
    method = "average", index = "all")
```

Si inspeccionas los resultados, encontrarás que, para el método del enlace promedio, el resultado es algo diferente. En este caso, 16 indicadores sugieren 3 clústeres y otras 4 métricas sugieren 2 clústeres. Esta contradicción en las votaciones de los indicadores, hace que, en la práctica, algunos científicos de datos prefieran emplear solo un criterio, como por ejemplo la silueta promedio por su fácil interpretación.

4.5.2 Empleando la silueta promedio

Otra aproximación complementaria para determinar el número de clústeres es emplear la silueta promedio (Ver Sección 2.4.3 para una definición de la silueta y cómo se

interpreta.) . Esto se puede hacer rápidamente empleando la función **NbClust()** que empleamos en la sección anterior. Solo necesitamos fijar el argumento **index** igual a **"silhouette"**. Es decir:

```
# detectar el número de clúster óptimo empleando la silueta promedio
res_centroid_Sil <- NbClust(datos_est, distance = "euclidean", min.nc = 2,
↪   max.nc = 10,
    method = "centroid", index = "silhouette")

# mirar el resultado óptimo
res_centroid_Sil$Best.nc
```

Noten que el número de clústeres que maximiza la silueta es 2[11]. La silueta promedio para 2 clústeres es 0.57. Esta silueta implica que la estructura de clasificación encontrada es aceptable, pues está entre 0.51 y 0.7 (Ver Capitulo @ref{Metricas}).

También puede ser interesante ver las siluetas individuales por medio de un gráfico de siluetas (*Silhoutte Plots* en inglés). Este gráfico muestra qué tan bien se ajusta cada observación al clúster que ha sido asignado comparando qué tan cerca se encuentra de las demás en su conglomerado, dado un número determinado de conglomerados. Como se discutió en la Sección 2.4.3, una silueta cercana a uno significa que la observación está bien ubicada en su grupo, mientras que cerca a cero o menor implica que es posible que el individuo fue mal clasificado en su clúster.

La función **silhouette()** del paquete *cluster* (Maechler et al., 2022) permite calcular las siluetas para todas las observaciones. Esta función típicamente incluye los siguientes argumentos:

silhouette(x, dist, dmatrix)

donde:

- **x**: Es un objeto de clase matriz o **data.frame**.
- **dist**: Una matriz de distancias de clase **dist**.
- **dmatrixc**: Una matriz simétrica de distancias.

Para aplicar esta función, además tendremos que especificar el número de clústeres que deseamos evaluar. Para esto podemos emplear la función **cutree()** del paquete base de R. Esta función corta un árbol (dendrograma) resultado de aplicar la función **hclust()** al nivel (número de clústeres) que se desee. Es decir, para incluir 2 clústeres ($q = 2$) o 5 ($q = 5$). Esta función típicamente incluye los siguientes argumentos:

cutree(tree, k = NULL)

donde:

- **tree**: Un dendrograma (árbol) de clase **hclust**.
- **k**: El número de clústeres[12].

[11] Este resultado lo podríamos también extraer del objeto `res_centroid` en el compartimiento `Best.nc`.
[12] En nuestra notación q.

Para el **HCA** con método de aglomeración centroide, distancia euclidiana y para $q = 2$ (de acuerdo con los resultados encontrados anteriormente), podemos emplear las siguientes líneas de código.

```
# instalamos el paquete si se

# necesita

# install.packages('cluster')

# cargamos el paquete
library(cluster)

# comprobamos la clase del objeto
class(HCA_centroide)
# graficamos las siluetas
plot(silhouette(cutree(HCA_centroide, 2), datos_dist), border = NA)
```

En la Figura 4.5 podemos observar que 50 clientes fueron clasificados en el clúster 2 y los restantes 610 en el grupo 1. Los clientes del conglomerado 2 tienen una silueta promedio de 0.62, más alta que el promedio del clúster 1 (0.57). En el clúster 2 no encontramos clientes con silueta individual por debajo de 0.5; mientras que en el clúster 1 sí encontramos clientes con siluetas inferiores a 0.5, pero superiores a 0.4. Recuerda que cuando la silueta por debajo de 0.5 (pero superiores a 0.25) indica que la estructura de clasificación encontrada es débil y puede ser artificial.

Ahora empleemos el criterio de la silueta promedio para escoger el número de clústeres óptimo para los 5 métodos de aglomeración que nos faltaban. De tal manera que construiremos los siguientes objetos:

- **Enlace único**: `res_unico_Sil`, (*Single linkage*)
- **Enlace completo**: `res_completo_Sil`, (*Complete linkage*)
- **Enlace promedio**: `res_promedio_Sil`,
- **Enlace mediano**: `res_mediano_Sil`,
- **Centroide**: `res_centroide_Sil`,
- **Ward.D**: `res_ward_D_Sil`,
- **Ward.D2**: `res_ward_D2_Sil`, y
- **McQuitty**: `res_Mcquitty_Sil`.

Tras realizar la búsqueda del número de clústeres óptimo con el criterio de la silueta, obtenemos los resultados que se presentan en el Cuadro 4.1. Intencionalmente no se presenta el código para producir este cuadro. ¡Intenta reproducirla empleando la función **NbClust()**!

En el Cuadro 4.1 podemos ver que los métodos de aglomeración de **enlace completo**, **enlace promedio**, **centroide** y **McQuitty** generan el mismo resultado, con la mayor silueta. Así, continuemos empleando los resultados del método de **centroide** que ya habíamos analizado anteriormente.

Figura 4.5. Silueta para el HCA con dos clústeres emplendo el método de aglomeración de centroide y distancia euclidiana

Fuente: cálculos propios.

Tabla 4.1. Siluetas y número de clústeres óptimos para HCA con diferentes métodos de aglomeración

Método de aglomeración	Silueta promedio	Número óptimo de clústeres
Enlace único	0.3758	3
Enlace completo	0.5703	2
Enlace promedio	0.5703	2
Enlace mediano	0.3022	6
Centroide	0.5703	2
Ward.D	0.5157	3
Ward.D2	0.5148	3
McQuitty	0.5703	2

Fuente: cálculos propios.

Miremos en más detalle los resultados que tenemos con el **HCA**, el método de aglomeración **centroide**, distancia euclidiana y dos clústeres.

Antes de continuar con el análisis de los resultados, veamos cómo realizar el **HCA** división (DIANA).

4.6 Construcción de clústeres jerárquicos de división

El algoritmo jerárquico de división (DIANA) puede implementarse fácilmente en R empleando la función **diana()** del paquete *cluster* (Maechler et al., 2022). Esta función típicamente incluye los siguientes argumentos:

diana(x, metric, stand, keep.diss)

donde:

- **x**: Es un objeto de clase **matrix** o **data.frame** que contiene los datos. También es posible entregar los datos a la función con una matriz de proximidad.
- **metric**: Especifica el tipo de distancia que se desea emplear para calcular la matriz de proximidad (o de disparidad). Solo hay dos opciones disponibles **"euclidean"** y **"manhattan"**. Si se quisiera emplear una medida de proximidad diferente, entonces se puede emplear en el argumento una matriz de proximidad calculada previamente, por ejemplo con la función **dist()** como lo vimos anteriormente.
- **stand**: Si este parámetro se establece igual a **TRUE**, los datos serán estandarizados. Nota que si **x** es una matriz de distancias, entonces este argumento será ignorado. El valor por defecto de este argumento es **FALSE**.
- **keep.diss**: Operador lógico que indica si la matriz de proximidad debe mantenerse en el resultado. Si se establece en **FALSE**, se pueden obtener objetos con menor tamaño y, por tanto, ahorrar tiempo de asignación de memoria RAM. Pero en algunos casos, como lo veremos pronto, será necesario guardar esta matriz.

Implementemos el algoritmo DIANA empleando la distancia euclidiana. Adicional-

mente, empleemos los datos estandarizados (`datos_est`); por tanto, no es necesario volver a estandarizar los datos. Finalmente, guardemos la matriz de proximidad. El correspondiente código será:

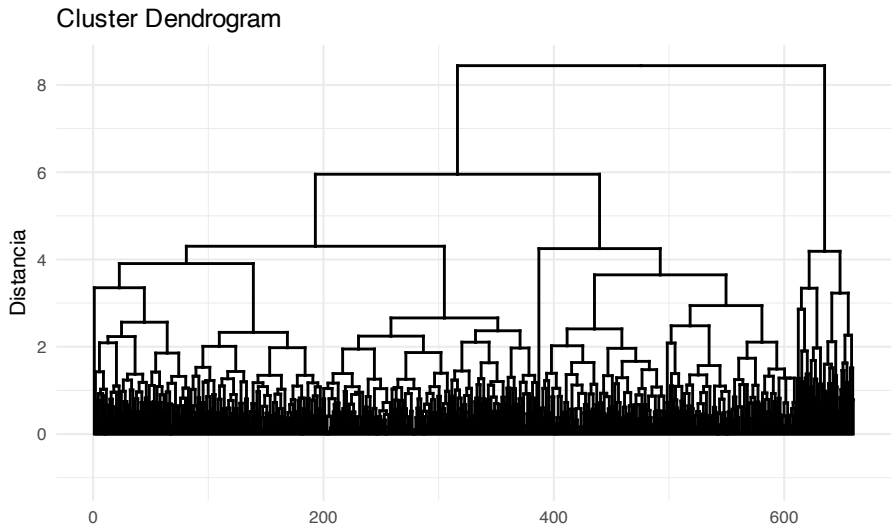
```
HCA_DIANA <- diana(datos_est, metric = "euclidean", keep.diss = TRUE)
```

Puedes visualizar el objeto `HCA_DIANA` como lo hicimos anteriormente con el **HCA** aglomerativo. Nota que en el compartimiento `diss` del objeto `HCA_DIANA` encontrarás la matriz de disparidad.

Ahora podemos visualizar las agrupaciones jerárquicas realizadas por el algoritmo DIANA con un dendrograma. Esto lo podríamos hacer con la función **plot()** como lo hicimos en el caso del algoritmo AGNES. Pero intentemos otra función para aprender algo diferente. Podemos emplear la función **fviz_dend()** del paquete *factoextra* (Kassambara y Mundt, 2020). Esta función permite visualizar dendrogramas en formato **ggplot2**.

```
fviz_dend(HCA_DIANA, show_labels = FALSE) + labs(y = "Distancia") +
↪  theme_minimal()
```

Figura 4.6. Dedrograma para DIANA y distancia euclidiana



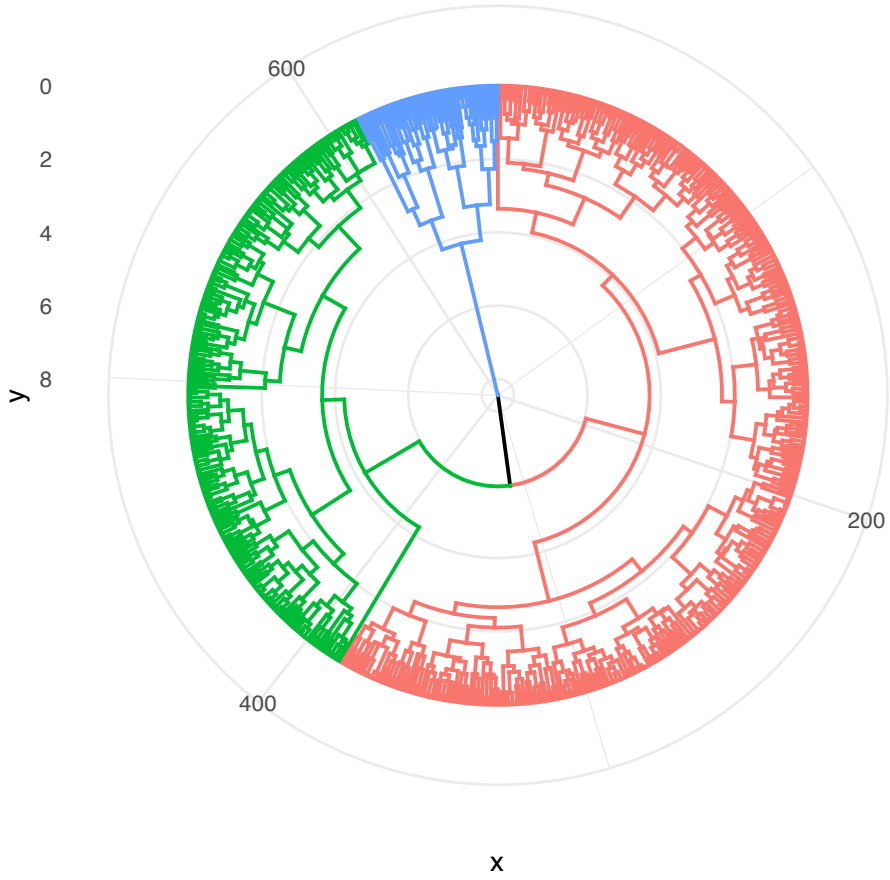
Fuente: cálculos propios.

Este paquete provee gran versatilidad de dendrogramas. Por ejemplo, intenta jugar un poco con el siguiente código que genera la Figura 4.7.

```
fviz_dend(HCA_DIANA, show_labels = FALSE, type = "circular", k = 3) +
↪  theme_minimal()
```

Continuemos con nuestro **HCA**. Recuerda que si bien hemos construido jerárquicamente todos los posibles clústeres, aún no sabemos cuál es el número de grupos óptimo

Figura 4.7. Dedrograma circular para DIANA, distancia euclidiana y cortando el árbol en 3 clústeres



Fuente: cálculos propios.

(por donde cortar el árbol). De manera análoga a lo realizado en la sección anterior, podemos emplear la función **NbClust()**. Solo necesitamos hacer un truco para incluir DIANA a las posibilidades que nos brinda esta función, pues recuerda que DIANA no es una de las opciones del argumento **method**. El truco es: no alimentar la función con datos estandarizados, sino con la matriz de disparidad y en el argumento **method** emplear la opción "single". Nota que el método de aglomeración simple es similar a los pasos que realizamos en el algoritmo DIANA[13]. Así, el código para encontrar el número óptimo de clústeres por medio de la silueta promedio es el siguiente:

```
library(NbClust)
res_DIANA_Sil <- NbClust(data = NULL, diss = HCA_DIANA$diss, distance = NULL,
↪  min.nc = 2,
    max.nc = 10, method = "single", index = "silhouette")
```

```
res_DIANA_Sil$Best.nc
```

```
## Number_clusters     Value_Index
##          3.0000          0.3758
```

Nota que, en este caso, el número óptimo de clústeres sería de 3 y la silueta promedio es de 0.3758. Esta silueta es sustancialmente menor que la que encontramos con la aproximación aglomerativa (Ver Cuadro 4.1). Así, en este caso en particular, es mejor el método jerárquico aglomerativo con método de aglomeración **centroide** y 2 clústeres. Continuemos el análisis de los resultados.

4.7 Visualización y analisis de resultados

Ya hemos detectado el mejor método jerárquico para agrupar los datos, empleando la distancia euclidiana,[14] y un algoritmo aglomerativo con el método de **centroide** y 2 clústeres.

Ahora, concentrémonos en los resultados del ejercicio de clústering. Con la misma función **cutree()** del paquete base de R que ya habíamos empleado, podemos obtener los resultados del algoritmo de clústering para el número de conglomerados deseados. En este contexto, por resultados del algoritmo de clustering se entiende como el respectivo clúster al que se asigna cada observación, también conocida como la membrecía de cada individuo a un conglomerado.

```
# calculamos la membrecía para cada cliente
grupos_HCA_aglo <- cutree(HCA_centroide, 2)
```

Veamos la membrecía de los primeros 10 clientes.

```
# vemos
head(grupos_HCA_aglo, 10)
```

[13] Para una mayor detalle sobre este tema ver el capítulo 6 de Kaufman y Rousseeuw (2009).

[14] Un análisis completo debería incluir también un análisis para otras distancias y comparar las siluetas obtenidas.

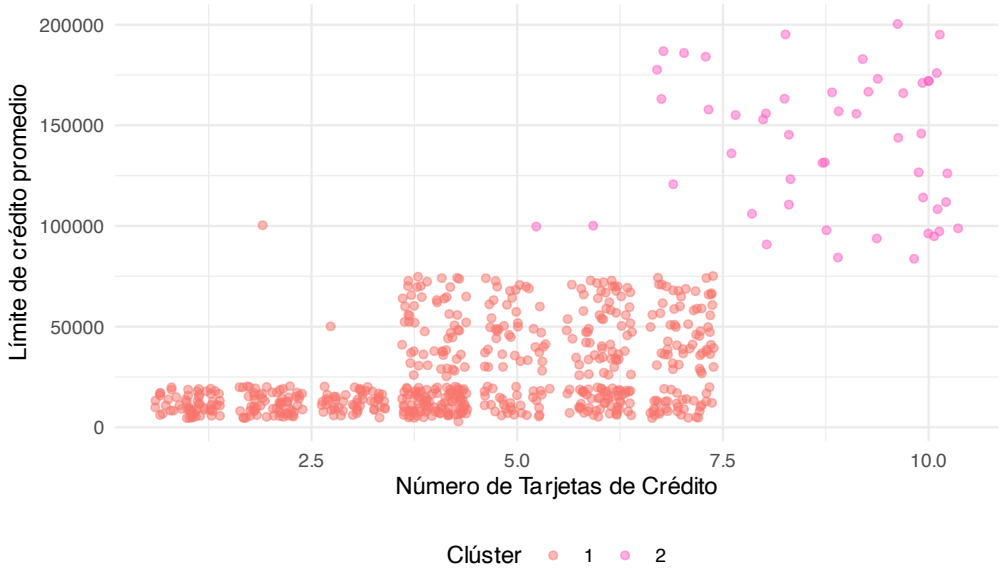
```
##  [1] 1 1 1 1 2 1 2 1 1 1
```

Esto implica que los primeros cuatro clientes pertenecen al clúster 1 y el quinto cliente y el séptimo hacen parte del clúster 2. Ahora podemos incluir esta variable de membrecía (variable cualitativa) a los datos originales.

```
datos_con_segmentacion <- cbind(datos_originales, (as.factor(grupos_HCA_aglo)))
names(datos_con_segmentacion)[8] <- "cluster"
# str(datos_con_segmentacion)
```

Una tarea que, típicamente, se realiza tras construir los clústeres es examinar el comportamiento de diferentes variables en cada uno de los grupos. Exploremos un poco cómo se agrupan los datos. Por ejemplo, la Figura 4.8 muestra cómo se separan de manera relativamente clara los datos en los dos clústeres si consideramos el límite de crédito promedio y el número de tarjetas de crédito (el código que genera esta visualización se presenta en el Anexo 1 de este Capítulo (sección 4.9.1)).

Figura 4.8. Relación entre el límite de crédito promedio y el número de tarjetas de crédito por clúster

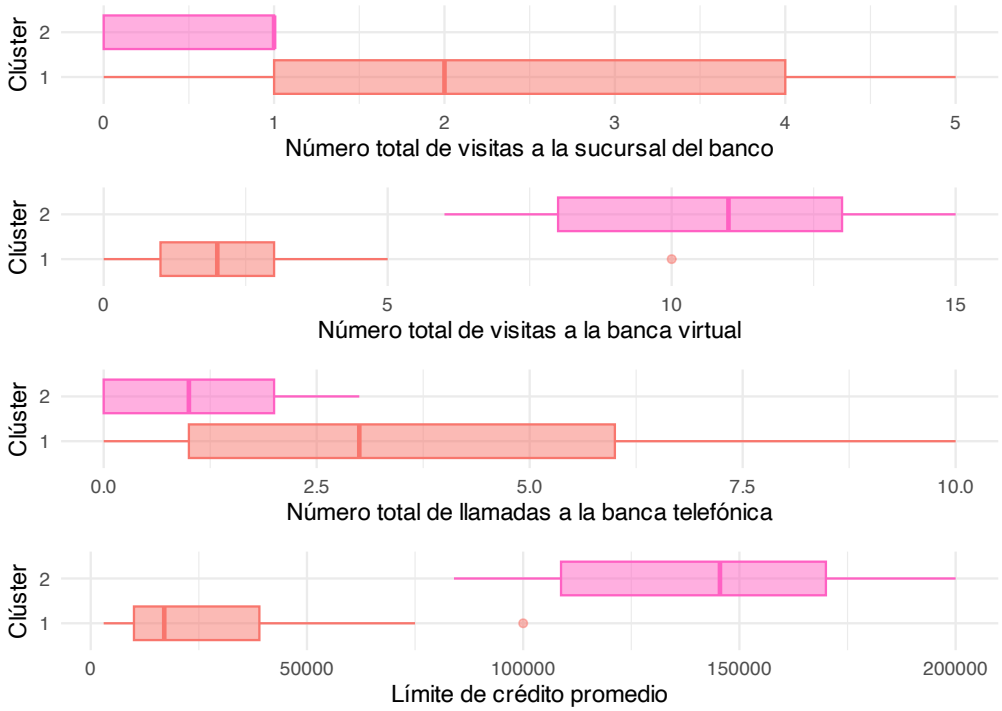


Fuente: cálculos propios.

Otra forma muy común de visualizar cada variable para los diferentes clústeres es emplear *boxplots*. En la Figura 4.9 se observa la distribución por clúster de todas las variables empleadas para construir las agrupaciones (el código que genera esta visualización se presenta en el Anexo 1 de este Capítulo (Sección 4.9.1)).

La Figura 4.9 permite ver que el clúster 1 está conformado por los clientes con un monto de límite de crédito aprobado relativamente bajo, que visitan poco la banca virtual y emplean con frecuencia relativamente alta el Banco y la Banca telefónica.

Figura 4.9. Distribución de las variables empleadas para construir las agrupaciones por clúster



Fuente: cálculos propios.

Por otro lado, en el clúster dos se encuentran los clientes con un mayor cupo promedio de crédito aprobado. Este segundo clúster se caracteriza porque sus miembros usan relativamente mucho más la banca virtual (en comparación al clúster uno) y no usan con tanta frecuencia las visitas al banco o las llamadas a la banca telefónica.

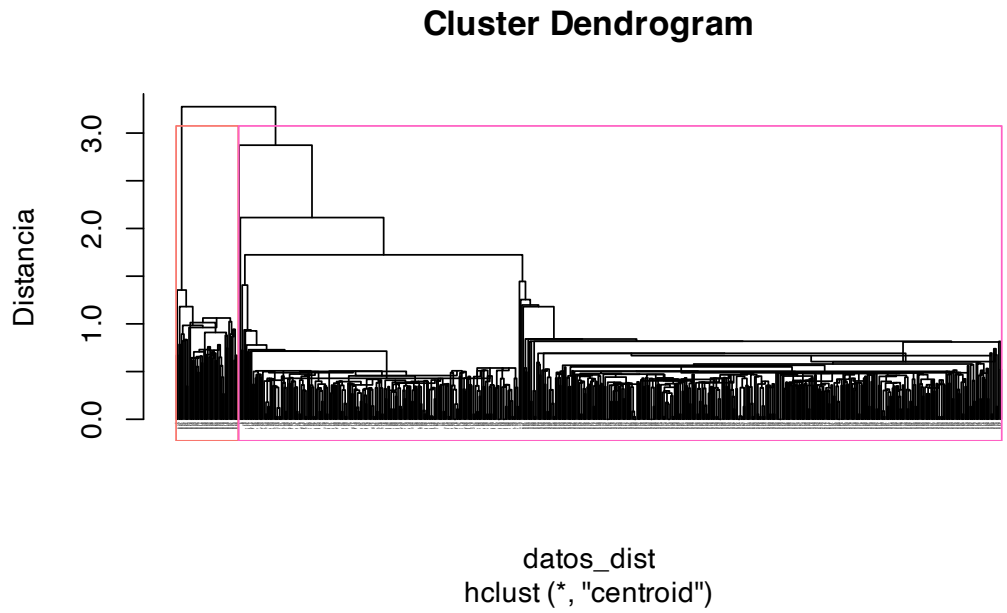
El segundo clúster lo podríamos llamar: el clúster de clientes premium. Noten que esto ya le permitiría a la sucursal bancaria generar diferentes estrategias de mercadeo y de servicio al cliente para los dos clústeres.

En el Anexo 2 de este Capítulo (Sección 4.9.2) se presenta un análisis estadístico complementario para determinar si los clústeres presentan medias diferentes. Ese tipo de análisis puede ser muy útil para entender las características de los conglomerados construidos. Pero no se debe emplear para validar si los clústeres quedaron bien conformados o no, pues los algoritmos de construcción no necesariamente tienen como lógica hacer que las medias sean diferentes entre los grupos.

Ahora visualicemos los resultados de diferentes maneras. Por ejemplo, en algunas ocasiones puede ser deseable resaltar los conglomerados seleccionados directamente en el dendrograma (Ver Figura 4.10).

```
plot(HCA_centroide, main = "Dendrograma método del centroide", hang = -1, cex =
↪  0.1,
    ylab = "Distancia")
rect.hclust(HCA_centroide, k = 2, border = 2:5)
```

Figura 4.10. Dendrograma método del centroide



Fuente: cálculos propios.

Esta forma de visualización permite acentuar los resultados y muestra los dos grupos seleccionados.

Si queremos variar la estética, podemos emplear el paquete *ape* (R Core Team, 2020). Ese paquete tiene la función **plot.phylo()** que grafica dendrogramas. Esta función incluye los siguientes argumentos:

plot.phylo(x, type = "phylogram", show.tip.label = TRUE, edge.color = "black", edge.width = 1, edge.lty = 1, tip.color = "black")

donde:

- **x**: Es un objeto de clase **phylo** que contiene el dendrograma. Se puede emplear la función **as.phylo()** para convertir un objeto de clase **hclust** a clase **phylo**.
- **type**: Determina el tipo de dendrograma. Las opciones son: "phylogram", "cladogram", "fan", "unrooted" y "radial". Más adelante ilustraremos cada uno de esos tipos con un ejemplo. Por defecto **type = "phylogram"**.
- **show.tip.label**: Permite mostrar o no las etiquetas de cada individuo. Por defecto se muestran las etiquetas (**show.tip.label = TRUE**).
- **edge.color**: Permite escoger los colores de las ramas. Por defecto se emplea el negro (**edge.color = "black"**).
- **edge.width**: Un número que especifica el ancho de las ramas. Por defecto **edge.width = 1**.
- **tip.color = "black"**: El color de la etiqueta. El valor por defecto es **tip.color = "black"**.

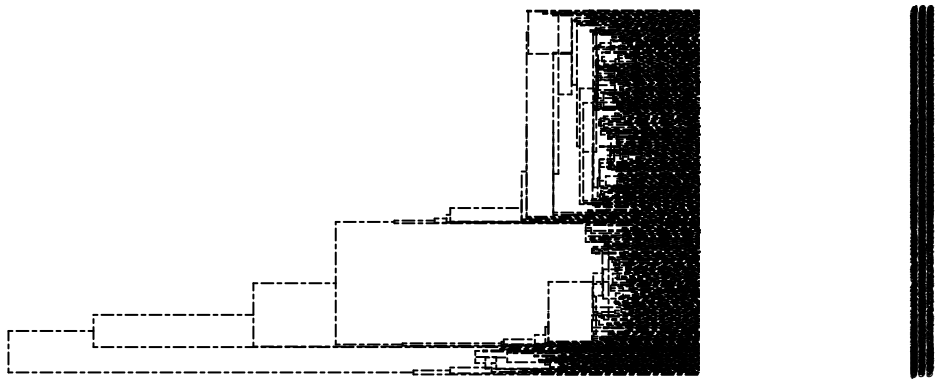
A continuación, se presentan diferentes visualizaciones empleando esta función.

```
# instalamos el paquete si se

# necesita

# install.packages('ape')

# cargamos el paquete
library(ape)

plot.phylo(as.phylo(HCA_centroide), cex = 0.4, label.offset = 0.5, edge.lty =
↪  6)
```

```
colores <- c("#F8766D", "#FF61C3")
plot(as.phylo(HCA_centroide), cex = 0.2, label.offset = 0.5, tip.color =
↪  colores[grupos_HCA_aglo])
```

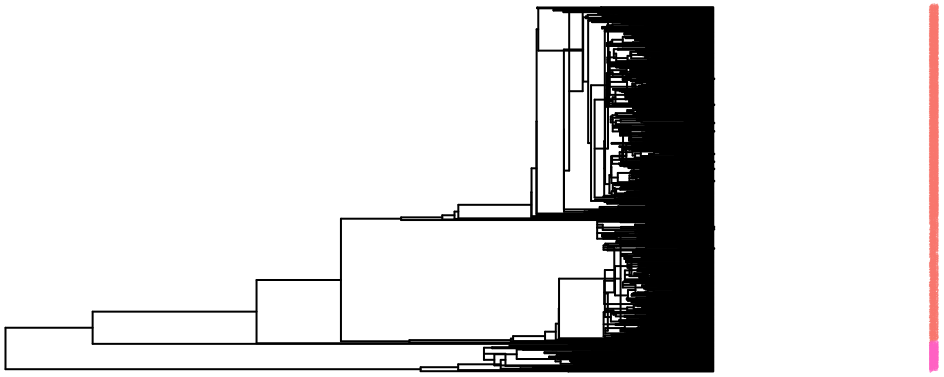
```
plot(as.phylo(HCA_centroide), cex = 0.2, label.offset = 0.5, tip.color =
↪  colores[grupos_HCA_aglo],
    type = "cladogram")
```

Figura 4.11. Dendrograma método del centroide con el paquete ape



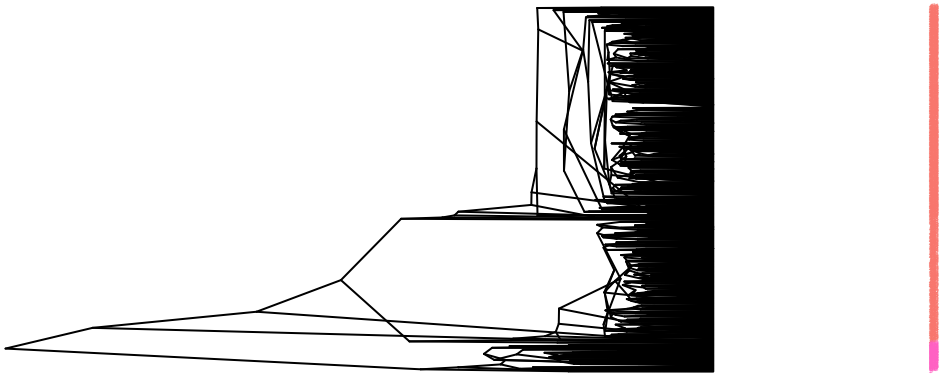
Fuente: cálculos propios.

Figura 4.12. Dendrograma método del centroide con colores en las etiquetas



Fuente: cálculos propios.

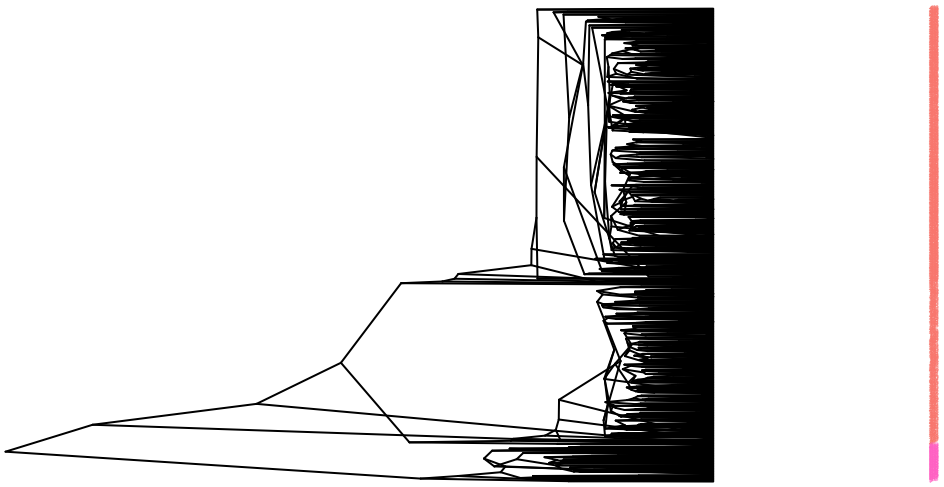
Figura 4.13. Dendrograma método del centroide tipo ucladogram



Fuente: cálculos propios.

```
plot(as.phylo(HCA_centroide), cex = 0.2, label.offset = 0.5, tip.color =
↪  colores[grupos_HCA_aglo],
    type = "unrooted")
```

Figura 4.14. Dendrograma método del centroide tipo unrooted



Fuente: cálculos propios.

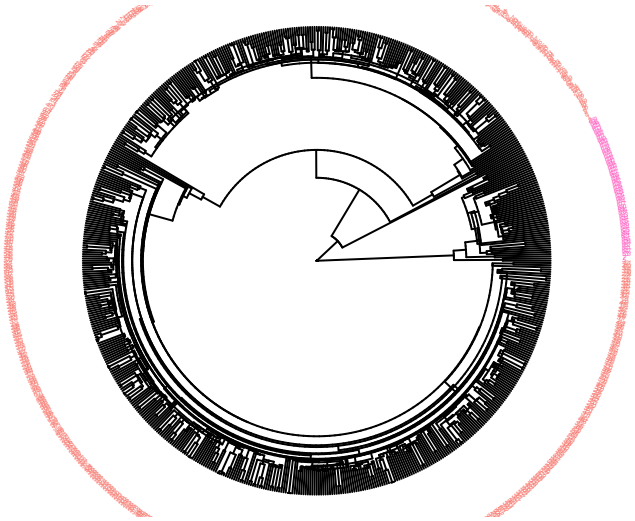
```
plot(as.phylo(HCA_centroide), cex = 0.2, label.offset = 0.5, tip.color =
↪  colores[grupos_HCA_aglo],
    type = "fan")
```

```
plot(as.phylo(HCA_centroide), cex = 0.2, label.offset = 0.5, tip.color =
↪  colores[grupos_HCA_aglo],
    edge.color = "steelblue", edge.width = 2, edge.lty = 2)
```

Otra opción es emplear el paquete *ggdendro* (de Vries y Ripley, 2024), que permite hacer dendrogramas con la lógica del paquete *ggplot2*. En este caso, la función que emplearemos es **ggdendrogram()**. A continuación se presentan unos ejemplos.

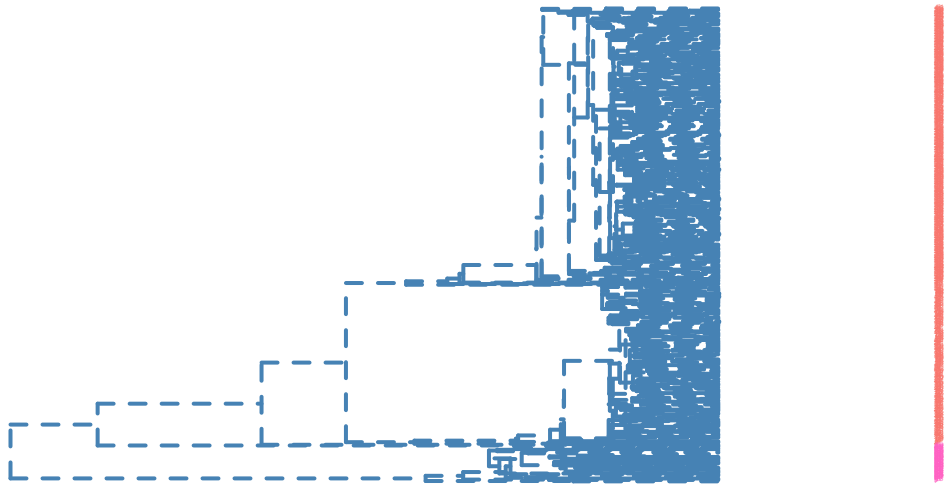
```
# instalamos el paquete si se necesita install.packages('ggdendro') cargamos
↪  el
# paquete
library(ggdendro)

ggdendrogram(HCA_centroide)
ggdendrogram(HCA_centroide, rotate = TRUE)
ggdendrogram(HCA_centroide, rotate = TRUE, theme_dendro = FALSE) + labs(title =
↪  "Dendrograma con ggplot2") +
    theme_minimal()
```

Figura 4.15. Dendrograma método del centroide tipo fan



Fuente: cálculos propios.

Figura 4.16. Dendrograma método del centroide horizontal



Fuente: cálculos propios.

Finalmente, regresemos al paquete *factoextra* (Kassambara y Mundt, 2020) que usamos anteriormente. Como se discutió antes, este paquete emplea *ggplot2* para realizar las visualizaciones. La función **fviz_dend()** de este paquete tiene los siguientes argumentos:

viz_dend(x, k = NULL, cex = 0.8)

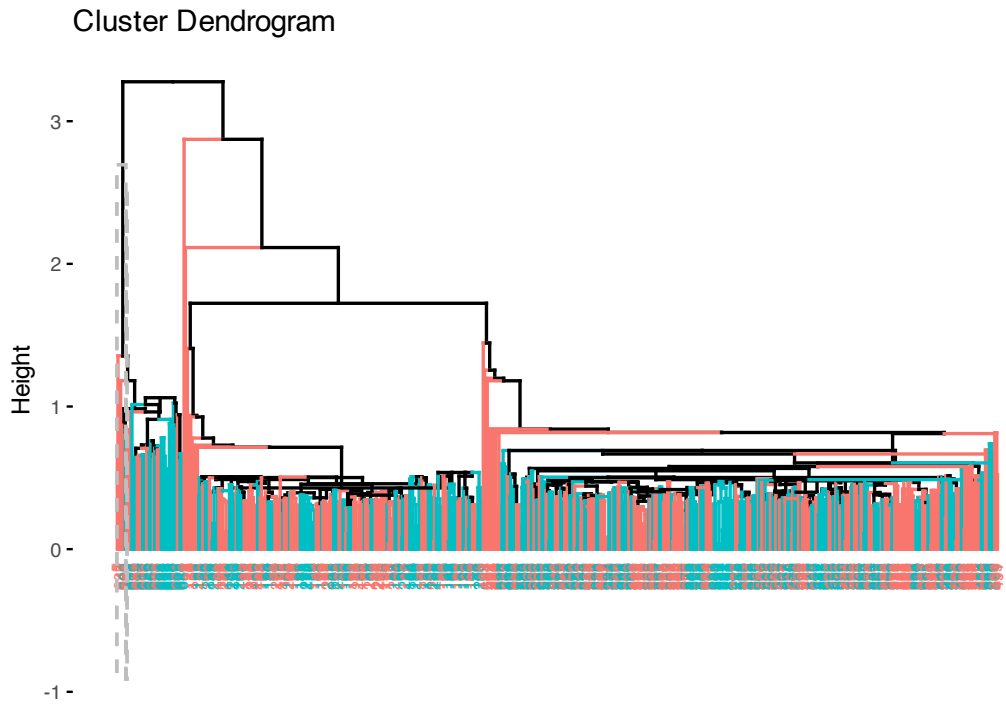
donde:

- **x**: Es un objeto de clase **hclust**, **agnes**, **diana**, **hcut**, **hkmeans** o **HCPC**.
- **k**: El número de clústeres[15].
- **cex**: El tamaño de la etiquetas. Por defecto **cex = 0.8** de distancias.
- **color_labels_by_k**: Si se desea que se coloree los grupos o no.
- **rect**: Si se desea adicionar un rectángulo para acentuar los conglomerados. Por defecto, **rect = FALSE**.

Veamos un ejemplo.

```
fviz_dend(HCA_centroide, k = 2, cex = 0.5, color_labels_by_k = TRUE, rect =
↪  TRUE)
```

Figura 4.17. Dendrograma método del centroide



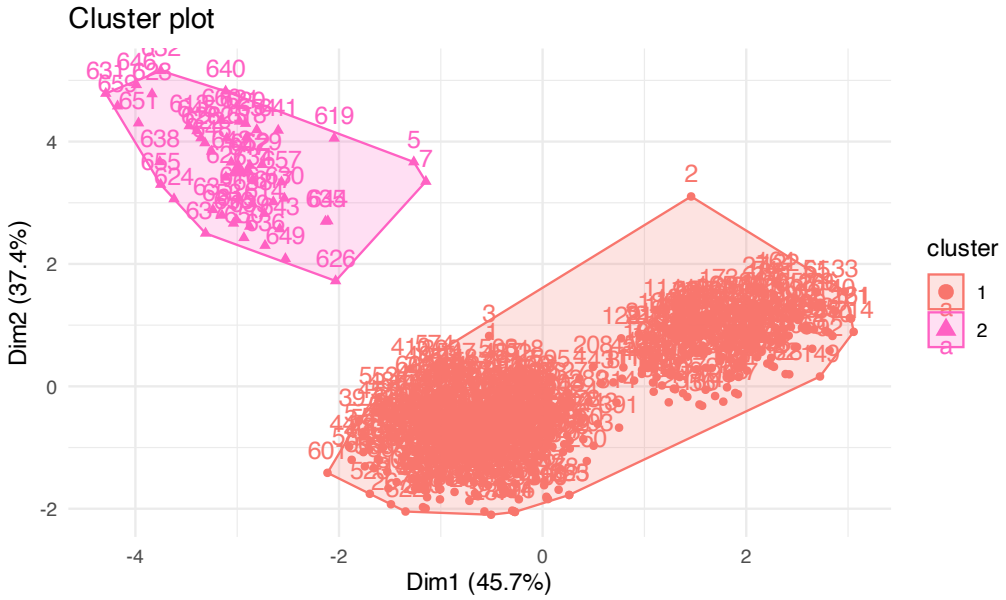
Fuente: cálculos propios.

[15]En nuestra notación q.

En el mismo paquete está una función que permite encontrar los primeros dos Componentes Principales (para una discusión de esta técnica ver Alonso C. (2020)) y. en ese espacio, visualizar los clústeres construidos. Esa función es **fviz_cluster()**. Esta función requiere que la alimentemos con los datos (estandarizados) para calcular los dos primeros componentes principales y un objeto que tenga la membrecía de cada uno de los individuos. Es decir,

```
fviz_cluster(list(data = datos_est, cluster = grupos_HCA_aglo)) +
↪  theme_minimal() +
   scale_colour_manual(values = c("#F8766D", "#FF61C3")) +
   ↪  scale_fill_manual(values = c("#F8766D",
   "#FF61C3"))
```

Figura 4.18. Dendrograma método del centroide



Fuente: cálculos propios.

La Figura 4.18 nos muestra que los dos componentes principales que resumen las variables empleadas en el clústering explican el 83.1 % de la variación de estas. Esto es relativamente alto. Por otro lado, y más importante para nuestro estudio, cuando mapeamos a ese espacio (espacio que representa el 83.1 % de la variación de las variables originales) los individuos, observamos cómo el algoritmo de clasificación pudo separar los datos de una manera aparentemente acertada.

Por último, guarda el espacio de trabajo para ser empleado en los siguientes capítulos.

4.8 Comentarios finales

Finalmente, es importante anotar que una tarea prioritaria es tratar de caracterizar cada uno de los clústeres. En la práctica es común que las organizaciones quieran ponerles nombres a los conglomerados. Por ejemplo, si estamos hablando de clústeres de clientes, el área de mercadeo deseará "bautizar" los segmentos encontrados. El grupo de clientes "premium", los "gold", etc. Esta tarea no es fácil, pero se puede realizar entendiendo de manera muy detallada las características de los individuos al interior de los conglomerados. En el siguiente capítulo estudiaremos otro tipo de algoritmo de clústering.

4.9 Anexos

4.9.1 Anexo 1. Código para generar visualizaciones de las variables en los clústeres

A continuación se presenta el código que generó la Figura 4.8.

```
library(ggplot2)
ggplot(datos_con_segmentacion, aes(x = Total_Credit_Cards, y =
↪  Avg_Credit_Limit,
    col = cluster)) + geom_jitter(size = 1.5, alpha = 0.5) +
    ↪  scale_color_manual(values = c(`1` = "#F8766D",
    `2` = "#FF61C3")) + labs(x = "Número de Tarjetas de Crédito", y = "Límite
    ↪  de crédito promedio",
    color = "Clúster") + theme_minimal() + theme(legend.position = "bottom")
```

El siguiente código generó la Figura 4.9.

```
p2 <- ggplot(datos_con_segmentacion, aes(y = cluster, x = Total_visits_bank,
↪  col = cluster,
    fill = cluster)) + geom_boxplot(alpha = 0.5) +
    ↪  scale_color_manual(aesthetics = c("fill",
    "colour"), values = c(`1` = "#F8766D", `2` = "#FF61C3")) + labs(x = "Número
    ↪  total de visitas a la sucursal del banco",
    y = "Clúster") + theme_minimal() + theme(legend.position = "none")



p3 <- ggplot(datos_con_segmentacion, aes(y = cluster, x = Total_visits_online,
↪  col = cluster,
    fill = cluster)) + geom_boxplot(alpha = 0.5) +
    ↪  scale_color_manual(aesthetics = c("fill",
    "colour"), values = c(`1` = "#F8766D", `2` = "#FF61C3")) + labs(x = "Número
    ↪  total de visitas a la banca virtual",
    y = "Clúster") + theme_minimal() + theme(legend.position = "none")

p4 <- ggplot(datos_con_segmentacion, aes(y = cluster, x = Total_calls_made, col
↪  = cluster,
```

```
    fill = cluster)) + geom_boxplot(alpha = 0.5) +
     ↪  scale_color_manual(aesthetics = c("fill",
    "colour"), values = c(`1` = "#F8766D", `2` = "#FF61C3")) + labs(x = "Número
     ↪  total de llamadas a la banca telefónica",
    y = "Clúster") + theme_minimal() + theme(legend.position = "none")

p5 <- ggplot(datos_con_segmentacion, aes(y = cluster, x = Avg_Credit_Limit, col
 ↪  = cluster,
    fill = cluster)) + geom_boxplot(alpha = 0.5) +
     ↪  scale_color_manual(aesthetics = c("fill",
    "colour"), values = c(`1` = "#F8766D", `2` = "#FF61C3")) + labs(x = "Límite
     ↪  de crédito promedio",
    y = "Clúster") + theme_minimal() + theme(legend.position = "none")


library(gridExtra)
grid.arrange(p2, p3, p4, p5, nrow = 5)
```

4.9.2 Anexo 2. Empleando modelos ANOVA para comparar las medias en los clústeres construidos

Podemos hacer un análisis estadístico complementario para determinar si los grupos que se formaron tienen un comportamiento diferente en su media. Para eso compararemos las medias de los grupos.

```
aggregate(datos_con_segmentacion[, 3:7], list(datos_con_segmentacion$cluster),
 ↪  mean)
```

```
##   Group.1 Avg_Credit_Limit Total_Credit_Cards Total_visits_bank
## 1       1         25847.54            4.37541           2.55082
## 2       2        141040.00            8.74000           0.60000
##   Total_visits_online Total_calls_made
## 1             1.92623         3.788525
## 2            10.90000         1.080000
```

Veamos si realmente existe una diferencia en las medias entre los grupos empleando un modelo ANOVA. Esto se debe hacer para cada una de las variables. A manera de ejemplo, lo haremos solo para la variables `Avg_Credit_Limit`. Para esto empleemos la función **aov()** que se encuentra en el paquete base de R.

Esta función típicamente incluye los siguientes argumentos:

aov(formula, data = NULL)

donde:

- **formula**: Es una fórmula que especifica el modelo a estimar. A mano derecha se expresa la variable continua dependiente, en vez del signo igual se emplea "$\sim$" y a la izquierda del signo se expresan las variables categóricas. En caso de existir

más de una variable categórica se separan con un singo de "$+$". Por ejemplo, si se quiere probar si la media de la variable cuantitativa y es diferente para las diferentes categorías de la variable cualitativa x, entonces la fórmula será $y \sim x$.
- **data**: Corresponde al **data.frame** que se empelará para estimar el modelo ANOVA.

Entonces las siguientes líneas presentan el cálculo del modelo ANOVA.

```
fit_credit_limit <- aov(Avg_Credit_Limit ~ cluster, data =
↪  datos_con_segmentacion)
summary(fit_credit_limit)
```

```
##              Df    Sum Sq   Mean Sq F value Pr(>F)
## cluster       1 6.132e+11 6.132e+11    1262 <2e-16 ***
## Residuals   658 3.197e+11 4.859e+08
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Un mirada precipitada a estos resultados nos podrían llevar a concluir, con un 99% de confianza, que se puede rechazar la hipótesis nula de que las medias de los cupos promedios aprobados por cliente sean iguales entre los dos grupos. Pero recuerda que es necesario comprobar los supuestos del modelo para poder sacar conclusiones.

Entonces, procedamos a constatar el cumplimiento de los supuestos. Recuerden que los supuestos de un modelo ANOVA son: varianza constante, independencia entre las observaciones y normalidad de los residuales. La independencia entre las observaciones parece más fácil de argumentar que se cumple, pues no necesariamente el cupo aprobado de un cliente afecta al de otro cliente. Este supuesto lo dejaremos de lado.

Para constatar la varianza homogénea podemos emplear la prueba de Levene. Esta se puede implementar con la función **leveneTest()** del paquete *car* (Fox y Weisberg, 2019). Esta función solo requiere como argumento el objeto en el que está el modelo a ser evaluado.

```
library(car)
leveneTest(fit_credit_limit)
```

En este caso, el valor p permite rechazar la nula de una varianza homogénea. Es decir, no se cumple el supuesto de varianza homogénea según esta prueba. Ahora constatemos este resultado con otras dos pruebas de varianza constante: la prueba de Bartlett y la de Figner-Killeen. Las dos pruebas están implementadas en el paquete base de R, y se puede acceder por medio de las funciones **bartlett.test()** y **fligner.test()**, respectivamente. Estas dos funciones emplean como argumentos los mismos que empleamos para la función **aov()**.

```
bartlett.test(Avg_Credit_Limit ~ cluster, data = datos_con_segmentacion)
```

```
##
##  Bartlett test of homogeneity of variances
```

```
##
## data:  Avg_Credit_Limit by cluster
## Bartlett's K-squared = 31.198, df = 1, p-value = 2.33e-08
```

```
fligner.test(Avg_Credit_Limit ~ cluster, data = datos_con_segmentacion)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Avg_Credit_Limit by cluster
## Fligner-Killeen:med chi-squared = 33.817, df = 1, p-value = 6.055e-09
```

Las dos pruebas ratifican la conclusión de del *test* de Levene: el supuesto de varianza homogénea es violado.

De hecho, existe una alternativa para cuando hay una varianza no homogénea: la prueba de Welch de una vía (Welch one-way test). Esta prueba supone normalidad en los diferentes grupos.

```
oneway.test(Avg_Credit_Limit ~ cluster, data = datos_con_segmentacion)
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  Avg_Credit_Limit and cluster
## F = 542.69, num df = 1.000, denom df = 51.945, p-value < 2.2e-16
```

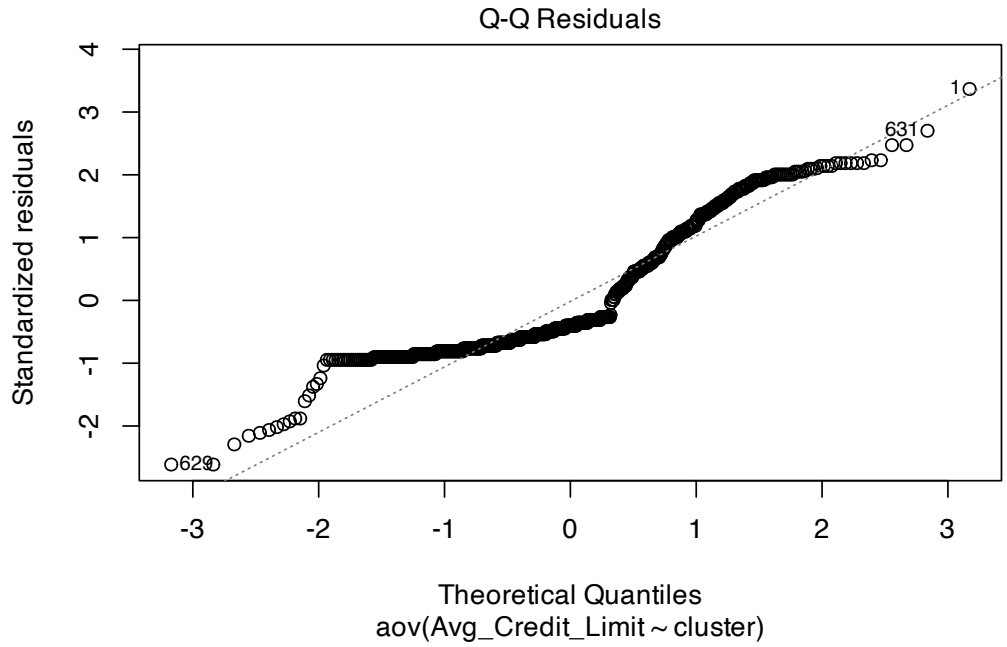
De acuerdo con esta prueba, es posible rechazar la hipótesis nula de que las medias son iguales. Pero de nuevo, es necesario comprobar si el supuesto de la prueba se cumple o no. Esta prueba supone normalidad, el cumplimiento o no de este supuesto se puede chequear por medio de un gráfico conocido como el gráfico Q-Q. Este grafica los cuantiles de los residuos, versus los cuantiles de la distribución normal. Si estos son iguales, deberían estar sobre la línea de 45 grados que se incluye en el gráfico.

```
plot(fit_credit_limit, 2)
```

La Figura 4.19 muestra con puntos, para cada observación, el cuantil acumulado observado y el teórico si los datos provienen de una distribución normal. La observaciones están relativamente cerca de la línea punteada, lo cual es indicio de un comportamiento similar al de una distribución normal. Este gráfico muestra que los residuos no siguen una distribución normal. Las colas de la distribución de los residuos se alejan de la distribución normal. Pero la decisión final la deberíamos tomar con pruebas estadísticas. Empleemos la prueba de normalidad de Shapiro que se puede implementar con la función **shapiro.test()** del paquete base de R. Esta función requiere como único argumento los residuales del modelo.

```
fit_residuals <- residuals(fit_credit_limit)

shapiro.test(x = fit_residuals)
```

Figura 4.19. q-q plot



Fuente: cálculos propios.

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_residuals
## W = 0.88844, p-value < 2.2e-16
```

Esta prueba permite rechazar la hipótesis nula de normalidad. Así, las conclusiones no son válidas.

En estos casos existe una alternativa de prueba que no supone normalidad: la prueba de Kruskal-Wallis de suma de rangos (*Kruskal-Wallis Rank Sum Test*). La prueba de Kruskal-Wallis es un método no paramétrico que permite comprobar la hipótesis nula de que los datos de la muestra provienen de la misma población. Es decir, que no hay diferencia entre los grupos. Estrictamente hablando no es exactamente una comparación de medias, sino también una comparación de la distribución de origen. Esta prueba la podemos realizar con la función **kruskal.test()** de la base de R.

```
kruskal.test(Avg_Credit_Limit ~ cluster, data = datos_con_segmentacion)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Avg_Credit_Limit by cluster
```

```
## Kruskal-Wallis chi-squared = 138.37, df = 1, p-value < 2.2e-16
```

Esta prueba permite rechazar la hipótesis nula de que los datos de los dos clústeres vienen de la misma distribución.

Finalmente, otra opción es emplear una segunda prueba que no tiene supuestos. Se conoce como la prueba no paramétrica de Tukey. La prueba también es conocida como el método de Tukey (Tukey's honest significance test, Tukey's HSD (*honest significant difference* en inglés) test, o *the Tukey-Kramer method*). Esta prueba la podemos realizar con la función **TukeyHSD()** de la base de R.

```
TukeyHSD(fit_credit_limit)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Avg_Credit_Limit ~ cluster, data = datos_con_segmentacion)
##
## $cluster
##         diff      lwr      upr p adj
## 2-1 115192.5 108825.3 121559.6     0
```

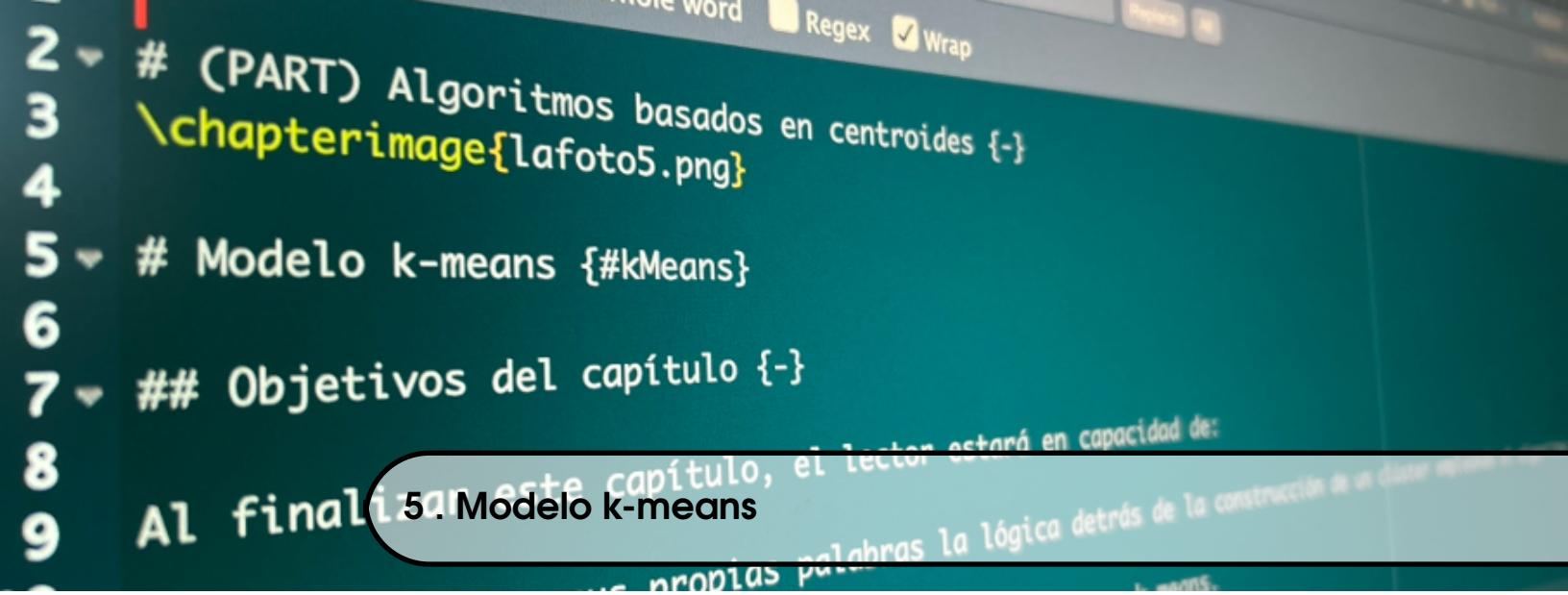
Esta prueba permite rechazar la hipótesis de que las medias son iguales.

En resumen, podemos concluir que el comportamiento del crédito aprobado para los dos clústeres es diferente. Las varianzas no son iguales, pero el uso de pruebas no paramétricas como la de Kruskal-Wallis permite concluir que las distribuciones no son iguales. La prueba no paramétrica de Tukey concluye que las medias son diferentes. Así, poniendo todos los resultados juntos, encontramos que las medias no son iguales.

Ustedes pueden realizar un ejercicio similar para las otras variables y encontrarán que, en efecto, el comportamiento de los dos clústeres construidos es diferente para cada una de las variables consideradas.

Parte III

Algoritmos basados en centroides



5 . Modelo k-means

Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Explicar en sus propias palabras la lógica detrás de la construcción de un clúster empleando el algoritmo k-means.
- Construir en R clústeres empleando el algoritmo k-means.

5.1 Introducción

Como se discutió en el Capítulo 1, los algoritmos de clústering particionado construyen grupos en los que cada individuo pertenece a solo uno de ellos y el número de clústeres es solo uno. Esto difiere de los algoritmos jerárquicos que estudiamos en el Capítulo 3. Sin importar si se trata de un **HCA** AGNES o DIANA, cada individuo puede pertenecer a múltiples grupos, dependiendo del punto de la construcción del algoritmo; en este caso los clústeres se sobreponen. En los algoritmos jerárquicos, el número de conglomerados cambia a medida que se construye el árbol jerárquico y se debe seleccionar el número de clústeres óptimo.

En este capítulo nos concentraremos en desarrollar un tipo de clústering **particionado**, que es intuitivo y computacionalmente no muy costoso. Como se mencionó en el Capítulo 1, el clústering particionado es una forma de conformar los grupos que son mutuamente excluyentes; es decir, subconjuntos de individuos que no se sobreponen, como se presenta en la Figura 5.1.

En este capítulo estudiaremos el algoritmo **k-means**, que es un algoritmo de aprendizaje de máquina (*machine learning*)[1]. Con seguridad, este es uno de los algoritmos más populares y, a su vez, sencillo.

5.2 La intuición

Como cualquier otra técnica de clústering, la idea principal detrás del algoritmo de **k-means** es dividir los individuos en k grupos de tal manera que los puntos dentro de cada grupo sean lo más similares posible entre sí y lo más diferentes posible de los puntos de otros grupos. La peculiaridad de esta aproximación es que va construyendo iterativamente los grupos al minimizar la suma de las distancias al cuadrado de cada punto al centroide más cercano. Intuitivamente, el centroide es un punto medio de todos los individuos que pertenecen a un grupo (Ver Figura 5.1).

Para entender la intuición de este algoritmo, veamos los pasos para realizar un clústering particionado, usando el algoritmo **k-means** para un determinado número de clústeres k:

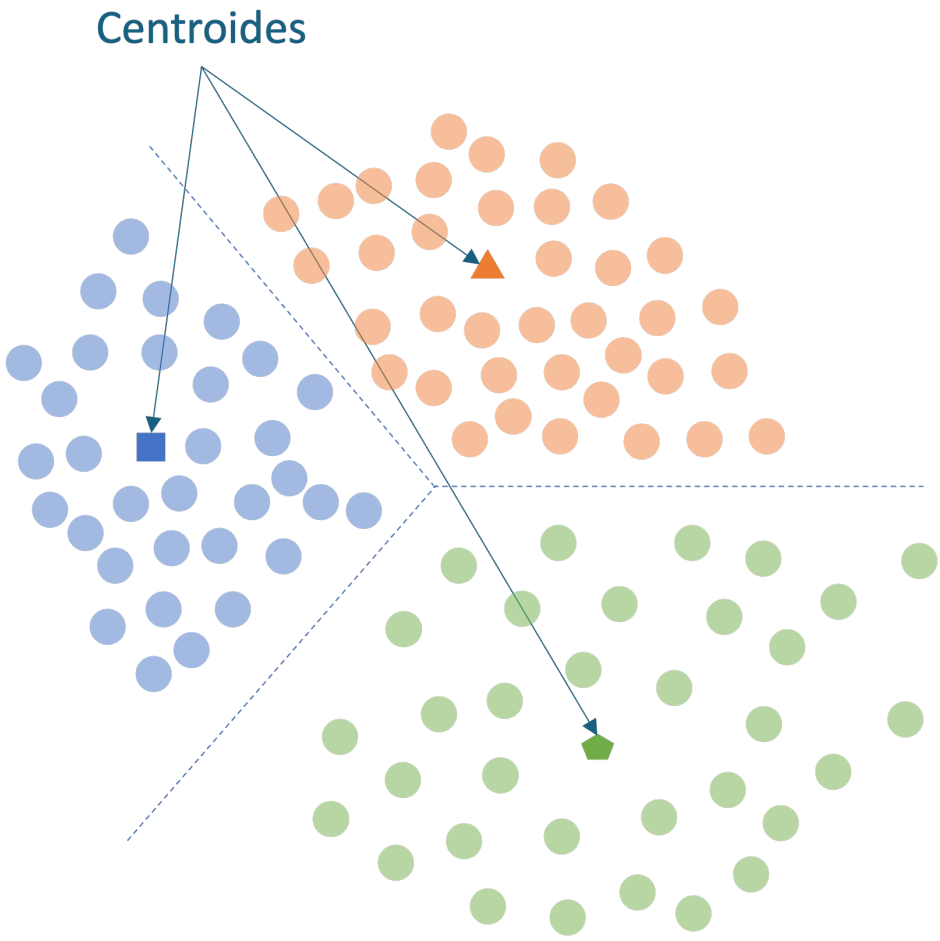
1. Elegir aleatoriamente k centroides a partir de los datos disponibles.
2. Calcular las distancias de cada observación a los centroides.
3. Agrupar individuos al centroide más cercano.
4. Recalcular el centroide usando la media aritmética[2].
5. Agrupar de nuevo al centroide más cercano.
6. Repetir los pasos 4 y 5 hasta que no se produzca ningún cambio en la asignación de las individuos a los clústeres o se alcance un número máximo de iteraciones.

El algoritmo **k-means**, al igual que los jerárquicos estudiados hasta ahora, solo funciona para variables (*features*) cuantitativas. Y cómo ya lo vimos en el Capítulo 3, para

[1] Más específicamente, un algoritmo de aprendizaje no supervisado.

[2] El nuevo centroide de un clúster se calcula como el promedio de las distancias de todos los individuos que pertenecen a ese clúster (distancias intra-clúster).

Figura 5.1. Representación de un algoritmo de clústering empleando k-means



Fuente: elaboración propia.

no ver afectado el análisis de clústering por la escala de las variables, se emplean variables estandarizadas.

El algoritmo de **k-means** inicia seleccionando k centroides al azar, donde k [3] es el número de grupos que queremos conformar, algo como el panel a de la Figura 5.2. Cada color es un centroide elegido al azar.

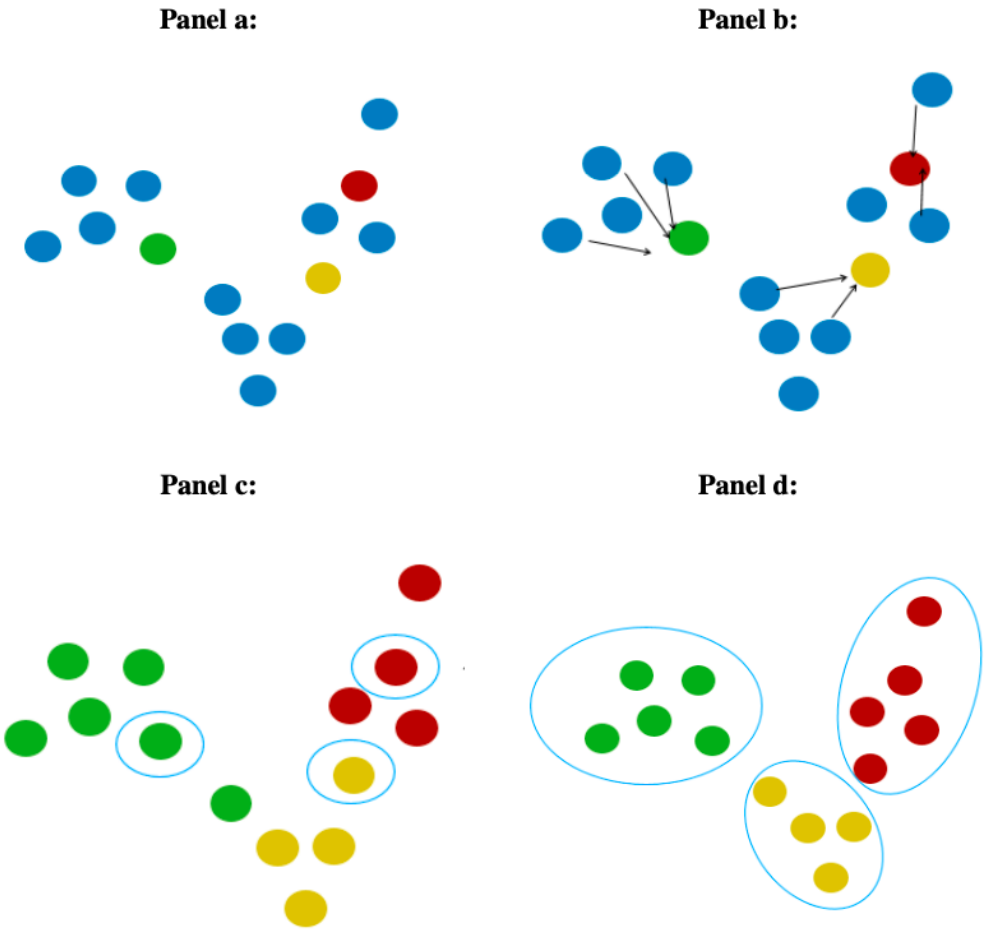
Luego se calcula la distancia de cada individuo a los centroides (panel b de la Figura 5.2) y se asigna al centroide más cercano, conformando los clústeres (panel c de la Figura 5.2).

A continuación, se calcula la media aritmética al interior de cada clúster, siendo esta media el nuevo centroide. Se repite el respectivo cálculo de distancia al nuevo centroide, así como la asignación de cada observación al grupo.

Esto se desarrolla de manera iterativa hasta que ya no se cambien de grupo los individuos o hasta cierto número de iteraciones que se definan arbitrariamente. Al final, el objetivo es llegar a algo como lo que se presenta en el panel d de la Figura 5.2.

[3] En los capítulos anteriores hemos denominado el número de clústeres óptimo con q y cualquier número de grupos con k. Este algoritmo se conoce como **k-means**, dado que k es el número de clústeres.

Figura 5.2. Algoritmo k-means



Fuente: elaboración propia.

5.3 Detalle técnico del algoritmo k-means

Matemáticamente, para asignar cada individuo a su centroide más cercano se quiere minimizar la suma de las distancias cuadráticas a las medias. Formalmente,

$$\min_{s} \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \bar{x}_i||^2 \tag{5.1}$$

donde $(x_1, x_2, x_3 ... x_n)$ son las observaciones de d dimensiones, k es el número de grupos a formar y $S = S_1, S_2, .., S_k$ es la suma de los cuadrados dentro de cada grupo. Para actualizar el centroide en la siguiente iteración $(t+1)$ se emplea la siguiente expresión:

$$\bar{x}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{5.2}$$

El algoritmo termina cuando los individuos no cambian de clúster entre iteraciones. En la siguiente sección, desarrollaremos un ejemplo de clústering particionado usando **k-means** en R.

5.4 k-means en R

Para implementar el algoritmo de **k-means** en R, emplearemos los mismos paquetes que hemos empleado hasta ahora. Es decir,

- factoextra (Kassambara y Mundt, 2020)
- cluster (Maechler et al., 2022)
- NbClust (Charrad et al., 2014)

Carguemos nuevamente los paquetes.

```
# cargamos los paquetes
library(factoextra)
library(NbClust)
library(cluster)
```

Para realizar el ejemplo práctico, continuaremos con la base de datos de clientes de tarjeta de crédito que empleamos en el Capítulo 4. (Ver detalles de cada uno de los pasos para construir la base de datos en el Capítulo 4). Carga el `working space` que guardaste en el Capítulo 4.

Nuestro objetivo continua siendo emplear los datos para responder la pregunta de negocio: *Con el fin de mejorar la satisfacción del cliente, brindar una atención más personalizada a los clientes y optimizar el gasto en mercadeo, ¿cómo segmentamos a nuestros clientes?*

De pronto te estarás preguntando ¿por qué tendremos que construir clústeres con el algoritmo **k-means** si ya lo hicimos con el algoritmo jerárquico? La razón es muy sencilla.

No sabemos si este algoritmo puede proveer "mejores" agrupaciones. Así, el científico de datos siempre está en la obligación de probar la mayor cantidad de aproximaciones y usar métricas para determinar cuál aproximación se ajusta a los datos empleados. En otras palabras, hasta no crear conglomerados con este método no podremos saber si era "mejor" emplear la aproximación jerárquica.

En la práctica, el algoritmo **k-means** se emplea para diferentes valores de k y después se selecciona el número óptimo de clústeres q comparando el comportamiento de las agrupaciones para cada uno de los posibles k considerados. Es decir, en este caso corremos el algoritmo para diferentes valores de k y luego seleccionamos entre las diferentes corridas del modelo (uno para cada k). En el caso de los modelos jerárquicos existe una diferencia grande que puede parecer sutil. En los modelos jerárquicos, el algoritmo se corre una sola vez (una sola vez para DIANA o para AGNES con el respectivo método aglomerativo) y al interior del árbol que se generó escogemos dónde cortar el árbol. En la práctica, tanto en el **HCA** como en el algoritmo **k-means** tendremos que seleccionar el "mejor" número de grupos, aún cuando la naturaleza de la selección es algo diferente.

Es decir, la tarea de seleccionar el número óptimo de clústeres q implica correr el algoritmo de **k-means** para diferentes k (número de clústeres) y comparar esas aproximaciones. Como lo discutimos en la Sección 2.4, podemos tener varias aplicaciones. De manera similar al Capítulo 4, emplearemos dos aproximaciones: una batería de métricas y la silueta promedio. A continuación, miraremos ambas aproximaciones.

5.4.1 Empleando batería de métricas

Evaluaremos entre 2 y 10 clústeres formados por el algoritmo **k-means**, usando 26 indicadores[4] que empleamos en el Capítulo 4. De manera similar a lo realizado en el Capítulo anterior, esto lo podemos realizar con el paquete *NbClust* (Charrad et al., 2014) empleando la función **NbClust()**.

La diferencia, en este caso, con respecto a los algoritmos de clústering jerárquicos que empleamos en el Capítulo 4, es que el argumento **method** debe establecerse como **"kmeans"**. La función calcula los pasos mencionados en la sección anterior, es decir, asigna al azar los centroides, calcula distancias, agrupa y repite iterativamente hasta conformar los grupos finales.

Adicionalmente, empleemos la **distancia euclidiana**, pero recuerden que existen otras distancias. Empleando los datos estandarizados (objeto `datos_est`), el siguiente código permitirá escoger el número óptimo de clústeres con los 26 indicadores[5]:

```
# Crear una semilla para que los centroides aleatorios siempre sean los mismos
# intenten diferentes semillas
```

[4] En este ejemplo no emplearemos los indices GAP, Gamma, Gplus y Tau por demandar mucho tiempo su cálculo. Puedes probar que el resultado no cambia si incluyes estos cuatro indicadores.

[5] Es importante recordar que el algoritmo **k-means** tiene, en su primera iteración, un componente aleatorio. Es decir, se parte de unos centroides aleatorios. Para evitar que cada vez que se corra el algoritmo cambien los resultados, es importante usar la función **set.seed()** . En otras palabras, si queremos obtener los mismos resultados es importante fijar una semilla aleatoria.

```
set.seed(12345)

res_kmeans <- NbClust(datos_est, distance = "euclidean", min.nc = 2, max.nc =
↪  10,
    method = "kmeans", index = "all")
```

Veamos cuál es el mejor número de clústeres según las 26 métricas empleando la función **fviz_nbclust()** del paquete *factoextra*(Kassambara y Mundt, 2020) que empleamos en el capítulo anterior. Recuerda que si obtienes un mensaje de error, puedes emplear la función **nueva_fviz_nbclust()**. Procedamos a visualizar los resultados con el siguiente código (Ver Figura 5.3):

```
# cargamos el paquete
library(factoextra)
# visualizando las métricas

fviz_nbclust(res_kmeans) + theme_minimal()

# nueva_fviz_nbclust(res_kmeans) + theme_minimal()
```

```
## Entre todos los índices:
## ===================
## * 2 proponen  0 como el mejor número de clústeres
## * 1 proponen  1 como el mejor número de clústeres
## * 1 proponen  2 como el mejor número de clústeres
## * 20 proponen  3 como el mejor número de clústeres
## * 1 proponen  6 como el mejor número de clústeres
## * 1 proponen  10 como el mejor número de clústeres
##
## Conclusión
## =========================
## * De acuerdo con la mayoría, el mejor número de clústeres es is  3 .
```

De acuerdo con los resultados de la Figura 5.3, 20 de los 26 métodos sugieren 3 clústeres, mientras que 2 índices sugieren no usar grupos.

5.4.2 Empleando Silueta promedio

Ahora usemos el coeficiente de silueta promedio para comparar los resultados del clústering. Al igual que en el Capítulo 4, podemos emplear la función **NbClust()** con el argumento **index = "silhouette"** (recuerda emplear la misma semilla nuevamente para asegurarte de tener el mismo resultado). También podrías extraer el resultado del número de clústeres óptimo que ya está en el objeto `res_kmeans`.

Otra opción, que no habíamos explorado es emplear la función **fviz_nbclust()** del paquete *factoextra* para visualizar la silueta promedio para diferentes k que ya se encuentran en el objeto `res_kmeans`. Por ejemplo, el siguiente código genera la Figura 5.4:

Figura 5.3. Votación de la batería de métricas por el número óptimo de clústeres para el algoritmo k-means y distancia euclidiana

Fuente: calculos propios.

```
set.seed(12345)
fviz_nbclust(datos_est, kmeans, method = "silhouette", k.max = 10) +
↪  theme_minimal() +
    ggtitle("Coeficiente de silueta")
```

```
## Entre todos los índices:
## ====================
## * 2 proponen  0 como el mejor número de clústeres
## * 1 proponen  1 como el mejor número de clústeres
## * 1 proponen  2 como el mejor número de clústeres
## * 20 proponen  3 como el mejor número de clústeres
## * 1 proponen  6 como el mejor número de clústeres
## * 1 proponen  10 como el mejor número de clústeres
##
## Conclusión
## ==========================
## * De acuerdo con la mayoría, el mejor número de clústeres es is  3 .
```

No importa la aproximación que sigamos, encontramos que el número de grupos sugeridos por la silueta promedio es 3 con una silueta promedio de 0.5157.

Noten que la silueta promedio del método de **k-means** no es la más grande (Ver Cuadro 5.1). El mejor modelo sigue siendo el de **centroide**, cuyos resultados son iguales

Figura 5.4. Silueta promedio para diferente número de clústeres para el algoritmo k-means y distancia euclidiana

Fuente: calculos propios.

Tabla 5.1. Siluetas y número de clústeres óptimos para diferentes aproximaciones de clústering (distancia euclidiana)

Método de aglomeración	Silueta promedio	Número óptimo de clústeres
Enlace único	0.3758	3
Enlace completo	0.5703	2
Enlace promedio	0.5703	2
Enlace mediano	0.3022	6
Centroide	0.5703	2
Ward.D	0.5157	3
Ward.D2	0.5148	3
McQuitty	0.5703	2
k-means	0.5157	3

Fuente: cálculos propios.

a los de **enlace completo**, **enlace promedio** y **McQuitty**. No obstante este resultado, por razones pedagógicas, continuaremos ilustrando cómo se pueden extraer las membrecías a los grupos si se emplea el algoritmo de **k-means**.

Antes de continuar, guarda tu espacio de trabajo. Lo emplearemos en los siguientes capítulos.

5.4.3 Siluetas individuales y membrecía para el algoritmo k-means

Después de conocer el número óptimo de clústeres, podemos aplicar el algoritmo **k-means** para continuar con nuestro análisis, similar a lo que presentamos en la Sección 4.7. La función que usaremos para implementar el algoritmo es **kmeans()** de la base de R. Esta función requiere, en su forma más simple, un objeto de clase **data.frame** con los datos (**x**) y el número de clústeres que queremos formar (**centers**). Es decir, el código será

```
# fijamos la semilla aleatoria
set.seed(12345)
# calculamos los clústeres para k = 3
cluster_kmeans <- kmeans(x = datos_est, centers = 3)
# Mirar todos los comportamientos del objeto creado
attributes(cluster_kmeans)
```

```
## $names
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
##
## $class
## [1] "kmeans"
```

El objeto que creamos con el algoritmo **k-means** (en este caso `cluster_kmeans`) tiene varios compartimientos (*slots*) entre los cuales podemos destacar:

- **cluster**: Un vector con números enteros (de 1 a k) que corresponde a la membrecía de cada observación. Es decir, el clúster al que se asigna cada dato.
- **centers**: Una matriz con los centros de cada clúster.
- **size**: El número de observaciones en cada clúster.

Por ejemplo, podemos saber el número de clientes en cada uno de los dos clústeres de la siguiente manera:

```
cluster_kmeans$size
```

```
## [1]  50 224 386
```

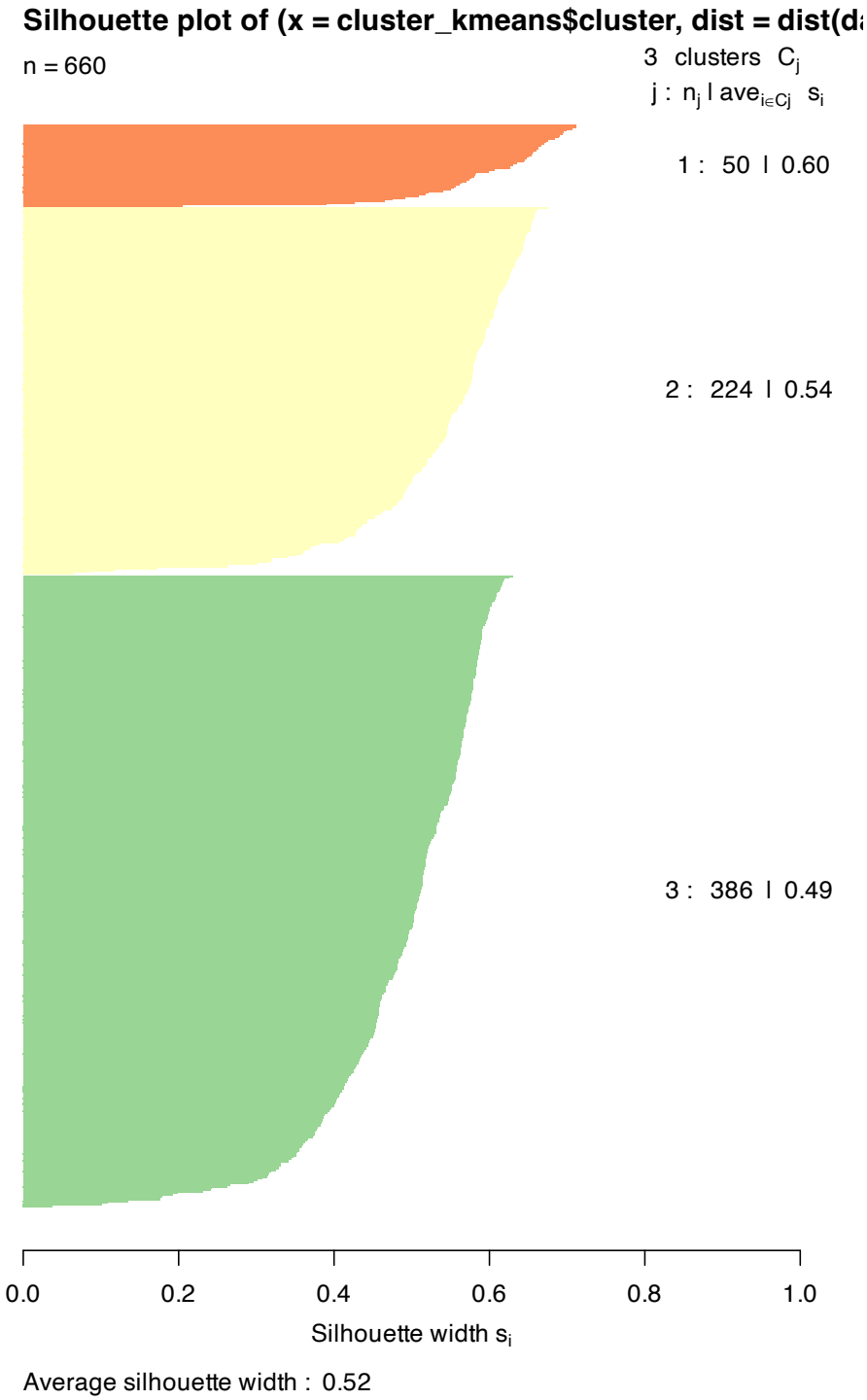
Es decir, en el primer clúster tenemos 50 clientes. Las membrecías de todos los clientes las podemos extraer fácilmente de la siguiente manera:

```
cluster_kmeans$cluster
```

Empleemos este resultado para construir las siluetas individuales. Recuerda que en la Figura 5.4 se presenta la silueta promedio para cada k. Si queremos visualizar el gráfico de siluetas individuales para tres clústeres conformados por el algoritmo de **k-means**, podemos emplear las membrecías que encontramos anteriormente y la función **silhouette()** del paquete *cluster* (Maechler et al., 2022):

```
plot(silhouette(cluster_kmeans$cluster, dist(datos_est)), border = NA)
```

Figura 5.5. Silueta para tres clústeres emplendo k-means y distancia euclidiana



Fuente: calculos propios.

Continuando con el análisis de resultados, como en el Capítulo 4, guardemos las membrecías de todos los clientes en el objeto `grupos_kmeans` con el siguiente código:

```
grupos_kmeans <- cluster_kmeans$cluster
```

Rápidamente, comparemos las agrupaciones que produce el algoritmo **k-means** con la obtenida en el **HCA** con el método del centroide.

```
table(grupos_kmeans, grupos_HCA_aglo)
```

```
##              grupos_HCA_aglo
## grupos_kmeans   1   2
##             1   0  50
##             2 224   0
##             3 386   0
```

Nota que este método está manteniendo a los mismos clientes en un clúster (clúster 2 en la aproximación jerárquica y clúster 3 en el de **k-means**), la diferencia se presenta en el clúster 1 del método jerárquico que es dividido en dos clústeres (clúster 1 y clúster 2) en el método de **k-means**. Ya discutimos que parece más adecuada la clasificación encontrada en por el **HCA**. Recuerda que esto cambiará dependiendo de los datos que se empleen.

5.5 k-means++

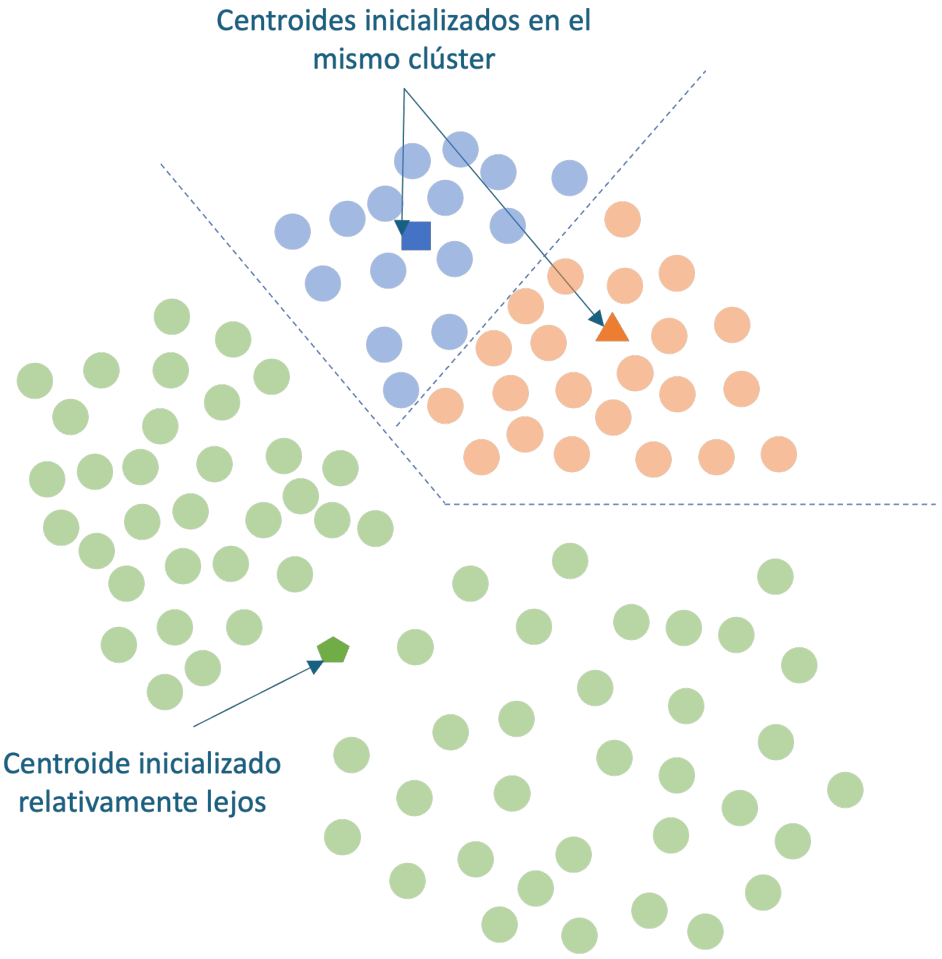
Antes de estudiar estos otros métodos, es importante reconocer que una desventaja del algoritmo **k-means** es que es sensible a la "inicialización" de los centroides. Es decir, de la elección aleatoria original de los centroides. Si aletaoriamente se selecciona como centroide un punto relativamente "lejano", puede acabar sin puntos asociados a él y, al mismo tiempo, más de un clúster puede acabar vinculado a un único centroide. Del mismo modo, más de un centroide podría inicializarse en el mismo clúster, lo que daría lugar a una agrupación deficiente (Ver grupos Figura 5.6).

De esta manera, por una mala inicialización de los centroides (al azar), obtenemos como resultado una partición inadecuada de los datos (compara la mala partición de las Figuras 5.6 con la presentada en la Figura 5.1).

Arthur et al. (2007) proponen una solución para este problema que se conoce con el nombre de **k-means++**. La propuesta es inicializar de manera "más inteligente" el algoritmo **k-means**.

Este algoritmo inicia, como el **k-means**, seleccionando aleatoriamente k centroides de los datos. Posteriormente, para cada individuo se calcula su distancia al centroide más cercano. En el siguiente paso se encuentra la diferencia entre los dos algoritmos. El siguiente paso será seleccionar el siguiente centroide, de tal forma que la probabilidad de elegir un punto como centroide sea directamente proporcional a su distancia al centroide más cercano elegido anteriormente. Es decir, el punto que se encuentra a la máxima distancia del centroide más cercano tiene más probabilidades de ser seleccionado como centroide. Una vez se establece este nuevo conjunto de centroides,

Figura 5.6. Centroides inicializados incorrectamente (al azar)



Fuente: elaboración propia.

se asignan de nuevo los individuos a cada centroide y se recalculan otros centroides siguiendo la metodología para el paso anterior. Y se continúa el algoritmo hasta que los grupos se estabilicen.

Noten que en esta aproximación se escogen centroides que están lejos unos de otros. Esto aumenta las posibilidades de arrancar con centroides que se encuentran en diferentes clústeres.

En últimas, el algoritmo **k-means++** es el mismo algoritmo **k-means** combinado con una inicialización más inteligente de los centroides. Este algoritmo es más lento que el **k-means** y es más sensible a datos con una separación poco clara[6].

Este algoritmo se puede emplear en R empleando la función **kmeanspp()** del paquete *maotai* (You, 2023). Esta función solo necesita los datos estandarizados (argumento **data**) y establecer el número de clústeres (argumento **k**).

Puedes explorar esta función. Desacertadamente, no podemos emplear este algoritmo con la función **NbClust()** del paquete *NbClust* (Charrad et al., 2014) para automatizar la búsqueda del número óptimo de clústeres. Si decides emplear este algoritmo, tendremos que calcular la silueta promedio para cada número de clústeres a "mano". En este libro no emplearemos esta aproximación.

5.6 Comentarios Finales

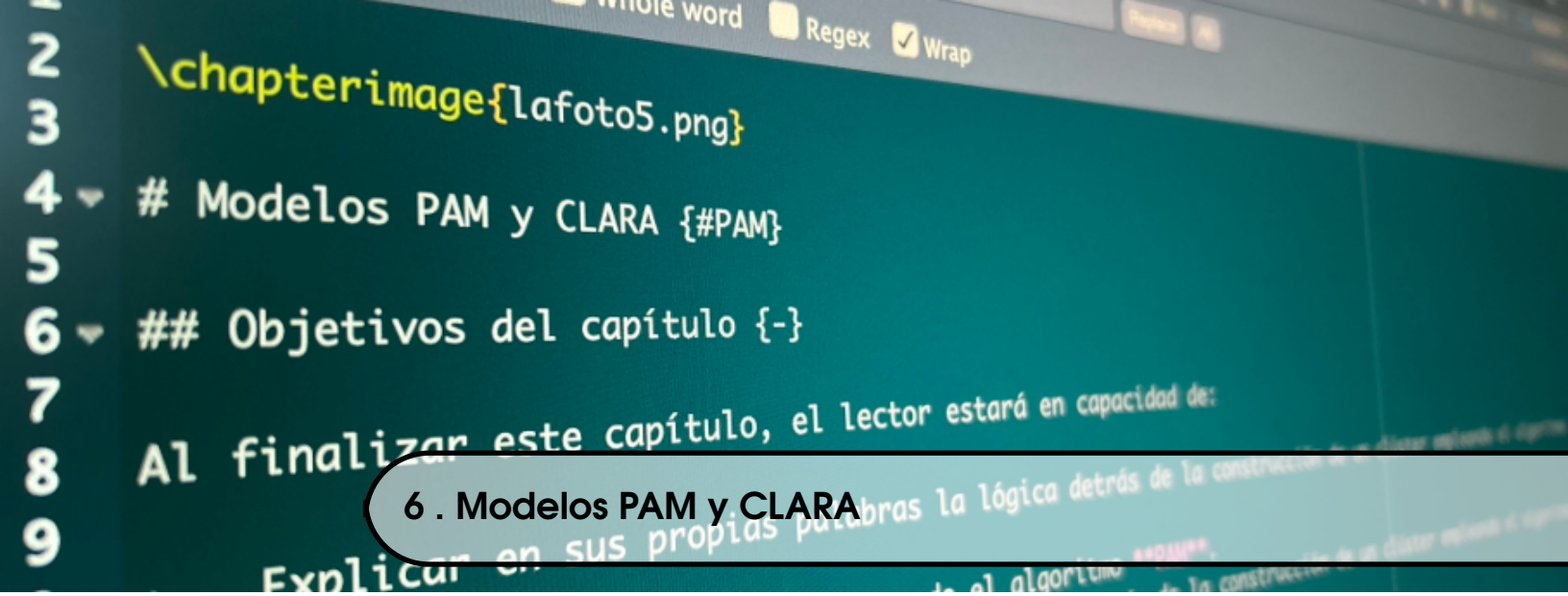
Por último, es importante mencionar que en este caso los dos algoritmos (el de este capítulo y el del capítulo anterior) no se pusieron de acuerdo en cuanto al número de clústeres. En algunos casos, los algoritmos se pueden poner de acuerdo con el número de clústeres, pero no la conformación de cada grupo. Una buena práctica es comparar por medio de métricas los resultados de uno y de otro tipo de algoritmos para elegir la mejor partición de los datos.

Adicionalmente, solo hemos empleado una medida de similitud (la distancia euclidiana). En la práctica es importante realizar el ejercicio con las diferentes distancias (que tengan sentido para la pregunta de negocio) y comparar los resultados empleando, por ejemplo, la silueta o las métricas estudiadas en la sección 2.4.

Por otro lado, no olvidemos que estas aplicaciones toman sentido para el negocio, en la medida en que obtenemos conclusiones de este tipo de análisis. Siempre podemos intentar asignarle nombres a cada clúster, que permiten un mejor entendimiento para luego tomar las decisiones necesarias con esta información.

En este caso, estudiamos una forma de clústering que es el particionado, y usando **k-means**. Sin embargo, hay otros algoritmos particionados que se aproximan de diferente manera a la construcción de los clústeres. En los próximos capítulos estudiaremos otros métodos de clústering particionado.

[6] En la jerga de los científicos de datos se le denomina a esta situación como datos con mucho ruido.



6 . Modelos PAM y CLARA

Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Explicar en sus propias palabras la lógica detrás de la construcción de un clúster empleando el algoritmo **PAM**.
- Construir en R clústeres empleando el algoritmo **PAM**.
- Explicar en sus propias palabras la lógica detrás de la construcción de un clúster empleando el algoritmo **CLARA**.
- Construir en R clústeres empleando el algoritmo **CLARA**.

6.1 Introducción

Los algoritmos de clústering se pueden clasificar en dos grandes tipos: los jerárquicos y los particionados (Ver Sección 2.3). Adicionalmente, los modelos de clústering particionados se pueden dividir en cinco categorías:

- **Basados en centroides**
- **Basados en densidad**
- **Basados en la distribución**
- **Métodos que combinan o modifican los anteriores**
- **Otros tipos de algoritmos de clústering**

En el Capítulo 5 estudiamos un algoritmo de clústering basado en centroides. Este modelo emplea el promedio de las distancias entre las observaciones del clúster como centroide (punto de gravedad del clúster). El algoritmo inicia eligiendo, al azar, los centroides y luego se recalculan empleando la media; de ahí el nombre de **k-means**.

Una de las desventajas de este algoritmo, al emplear la media, es que es sensible a datos extremos. Una mejora natural a este algoritmo es calcular de una manera diferente el centroide.

El centroide en un algoritmo de clústering actúa como un centro de gravedad alrededor del cuál se agrupan los individuos de un grupo. En el algoritmo **k-means** se emplea la distancia promedio de los individuos del clúster como centroide. Otra aproximación, para evitar el problema que pueden generar los datos atípicos en una medida de tendencia central como la media, es emplear la mediana. A este método se le conoce como **k-medians**[1].

El modelo de **k-medians**[2] ha sido generalizado con otros algoritmos particionados. En este Capítulo estudiaremos dos de estos algoritmos de clústering particionado basado en centroides: el modelo **PAM** (por su sigla en inglés del término *Partitioning Around Medoids*) y **CLARA** (*Clustering Large Applications*).

6.2 El modelo PAM

El modelo **PAM** (*Partitioning Around Medoids*) emplea como centro de gravedad para agrupar a los individuos un medoide. Por medoide se entiende un individuo (observación) dentro de un clúster cuya distancia (diferencia) promedio entre él, y todos los otros del mismo grupo, es la menor posible. En otras palabras, el medoide es el individuo "más central" del clúster y por lo tanto puede considerarse como el más representativo.

La intuición para preferir **PAM** frente al **k-means** (y **k-medians**) es que emplear medoids en lugar de centroides hace que **PAM** sea más robusto que **k-means**, viéndose menos afectado por datos atípicos o ruido.

[1] Es muy común que el algoritmo **k-medians** se emplee con la distancia de Manhattan y no la euclidiana.
[2] El modelo **k-medians** se puede implementar en R por medio la función **Kmedians()** del paquete *Kmedians* (Godichon-Baggioni y Surendran, 2023).

De esta menera, en **PAM** cada clúster está "representado" por una observación presente en el clúster (el medoide), mientras que en **k-means** cada conglomerado está representado por el promedio de las distancias de todos los individuos de todas las observaciones del clúster (el centroide).

Para entender la intuición de este algoritmo, veamos los pasos para realizar un clústering particionado usando el algoritmo **PAM**, para un determinado número de clústeres k:

1 Elegir aleatoriamente k individuos para ser el medoide de cada clúster.

2 Calcular las distancias de cada observación a los medoides.

3 Agrupar individuos al medoide más cercano.

4 Actualizar el medoide de cada clúster[3].

5 Repetir los pasos 3 y 4 hasta que no se produzca ningún cambio en la asignación de los individuos a los clústeres o se alcance un número máximo de iteraciones.

6.3 El modelo CLARA

Como vimos, el modelo **PAM** emplea como centro de gravedad para cada clúster una observación. Uno de los problemas de **PAM** es que se vuelve muy lento cuando la muestra es relativamente grande (miles de observaciones). El algoritmo **CLARA** (*Clustering Large Applications*) fue diseñado por Rdusseeun y Kaufman (1987) para reducir el tiempo de cálculo y solucionar el problema de almacenamiento en la memoria RAM que implica desarrollar **PAM**. Para lograr esto, **CLARA** tiene una aproximación diferente para escoger los medoides iniciales que implica emplear muestreo.

CLARA selecciona, al azar, una submuestra de los datos originales y aplica el algoritmo **PAM** en esa submuestra. Como resultado, se obtiene un conjunto de medoides basado en dicha submuestra. Este proceso se repite varias veces, con el fin de minimizar cualquier sesgo inherente al muestreo.

Nuevamente, veamos los pasos del algoritmo para entenderlo mejor. Los pasos para realizar un clústering particionado usando el algoritmo **CLARA**, para un determinado número de clústeres k, son:

1 Seleccionar una submuestra aleatoria, típicamente el tamaño de la submuestra es de $40 + 2 * k$ individuos.

2 Ejecutar **PAM** sobre la submuestra y encontrar k medoides.

3 Asignar todos los individuos (no solo los de la submuestra) al medoide más cercano.

4 Calcular la calidad del clústering empleando la suma de las distancias cuadradas entre los individuos y sus medoides (distancias intra-clúster).

5 Repetir los pasos 1 a 4 un número de veces predeterminada. Típicamente, 5.

[3]El medoide se calcula para cada clúster como el individuo del grupo con la menor distancia total a las demás observaciones del mismo clúster.

6 Seleccionar el mejor clústering (medoides y respectiva partición de los individuos) como aquel que tenga la mejor calidad (menor suma cuadrada de las distancias intra-clúster).

Noten que este algoritmo hace que **CLARA** sea menos sensible que el modelo **PAM** a la selección inicial de los medoides. Por otro lado, **PAM** puede ser más "preciso" para muestras pequeñas, pero **CLARA** es generalmente más rápido y eficiente en muestras grandes.

Habitualmente, la elección entre **PAM** y **CLARA** dependerá del tamaño de la muestra y la disponibilidad de recursos computacionales. Si bien es común que **PAM** sea recomendable para muestras pequeñas y **CLARA** para muestras grandes, en la práctica, cuando sea posible, emplearemos ambos algoritmos y los compararemos con alguna de las métricas que estudiamos en la Sección 2.2.

6.4 Implementación de PAM y CLARA en R

PAM y **CLARA** se pueden implementar en R empleando el paquete *cluster* (Maechler et al., 2022) con las funciones **pam()** y **clara()**, respectivamente.

Por otro lado, para estos dos algoritmos no podemos realizar la búsqueda automática del mejor número de clústeres empleando la función **NbClust()** del paquete *NbClust*. (Charrad et al., 2014) Existen otros dos paquetes que pueden hacer una búsqueda del número de grupos óptimo similar a lo realizado en los capítulos anteriores: *clValid* (Brock et al., 2008) y *parameters* (Lüdecke et al., 2020).

En este Capítulo, y en lo que resta de este libro, estaremos empleando el paquete *parameters*. Este paquete provee funciones que permiten integrar diferentes paquetes con algoritmos de clústering (como la base de R y el paquete *cluster*) y paquetes con métricas (como *NbClust*). Este paquete tiene la función **n_clusters()**, que permite, rápidamente, identificar el número de clúster óptimo empleando una métrica o una batería de métricas. Esta función tiene los siguientes argumentos:

n_clusters(x, standardize = TRUE, include_factors = FALSE, package = c("easystats", "NbClust", "mclust"), fast = TRUE, nbclust_method = "kmeans", n_max = 10, distance_method = "euclidean")

donde:

- **x**: Objeto con datos de clase **data.frame**.
- **standardize**: Si es **TRUE**, se estandarizarán los datos. Este es el valor por defecto.
- **include_factors**: Si es **TRUE**, las variables de clase **factor** se convierten en objetos numéricos para ser incluidos en los datos para determinar el número de clústeres. Por defecto, es igual a **FALSE**; es decir, se eliminan las variables que sean de clase **factor**. Esto se hace porque la mayoría de los métodos que determinan el número de conglomerados solo funciona con variables cuantitativas.
- **package**: Paquete desde el que se llamarán las métricas para determinar el número óptimo de conglomerados. En nuestro caso, emplearemos el paquete *NbClust*; es decir, **package = NbClust** . Otras opciones para este argumento son:

"all", **"easystats"**, **"mclust"** y **"M3C"**.
- **fast**: Si es igual a **FALSE**, se calcularán los 30 índices del paquete *NbClust*. Es equivalente a **index = "allong"** en el paquete *NbClust*. Por defecto, **fast = TRUE**
- **nbclust_method**: El método de clústering. Puede ser, por ejemplo, **cluster::diana** para el algoritmo **DIANA** del paquete *cluster*. Este argumento permite decir el paquete de origen y el nombre de la función que queremos emplear.
- **n_max**: Número máximo de clústeres a evaluar.
- **distance_method**: El tipo de distancia a calcular. Este elemento es pasado a la función **dist()**. Por defecto, el método es **"euclidean"**; los otros posibles valores son "maximum", "manhattan", "canberra", "binary" y "minkowski".
- **hclust_method**: El método de aglomeración para el clústering jerárquico. Este argumento solo se emplea cuando **nbclust_method = "hclust"** (Ver Sección 3.3). Las opciones son: "ward.D" (Ward Jr, 1963) , "ward.D2" (Murtagh y Legendre, 2014), "single" (enlace único), "complete" (enlace completo), "average" (enlace medio), "mcquitty", "median" (enlace mediano) o "centroid". Por defecto, **method = "complete"**.

Este paquete *parameters* también tiene un función que no calcula la batería de indicadores, sino solo la silueta promedio: **n_clusters_silhouette()**. Los argumentos de esta función son los mismos de la función **n_clusters()**. La única excepción es el argumento **nbclust_method** de la función **n_clusters()**, que cambia su nombre a **clustering_function** en la función **n_clusters_silhouette()**.

Tambien existen las funciones **n_clusters_elbow()** y **n_clusters_gap()** si se desea solo emplear el método del codo o el índice GAP (Ver Sección 2.4 para una discusión de estos métodos), respectivamente.

Ahora, procedamos a emplear esta "infraestructura" que nos brinda el paquete *parameters* para emplear **PAM** y **CLARA** en los datos que hemos venido trabajando en los capítulos anteriores. Carga el espacio de trabajo que guardaste en el Capítulo 5.

Empleemos la función **n_clusters_silhouette()** para determinar el número óptimo de clústeres de acuerdo con la silueta promedio, empleando el algoritmo **PAM** (**clustering_function= cluster::pam**), la distancia euclidiana (**distance_method = "euclidean"**) y el objeto de datos que ya estaban estandarizados (`datos_est`). El código será:

```
# Instalar el paquete si es necesario

# install.packages('parameters')

# Cargar el paquete
library(parameters)

# Crear una semilla para que los medoides

# aleatorios siempre sean los mismos
```

```
# intenten diferentes semillas
set.seed(12345)

#
res_PAM_Sil <- n_clusters_silhouette(datos_est, standardize = FALSE,
↪  distance_method = "euclidean",
    clustering_function = cluster::pam, n_max = 20)
```

Todos los resultados al emplear el algoritmo **PAM** para un número de clústeres de 1 a 20 ($k = 1, 2.., 20$) y evaluar la silueta promedio para cada uno de esos casos, han quedado en el objeto `res_PAM_Sil`. Imprimamos ese objeto:

```
res_PAM_Sil
```

```
## The Silhouette method, based on the average quality of clustering, suggests that the optimal nu
```

El mensaje muestra que el número óptimo de clústeres es 3. En los compartimientos **Silhouette** y **n_Clusters**, podemos encontrar la silueta promedio y el número de clústeres, respectivamente. En este caso, podemos encontrar fácilmente que la silueta promedio (máxima) fue de 0.5158. El código es el siguiente:

```
# Encontrar la silueta máxima para el número de clústeres óptimo
max(res_PAM_Sil$Silhouette)
```

```
## [1] 0.5157992
```

```
# encontrar el número de clústeres óptimo
res_PAM_Sil$n_Clusters[which.max(res_PAM_Sil$Silhouette)]
```

```
## [1] 3
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

También podemos graficar las siluetas para cada número de clústeres evaluados. El código que generó la Figura 6.1 es el siguiente:

```
plot(res_PAM_Sil) + theme_minimal()
```

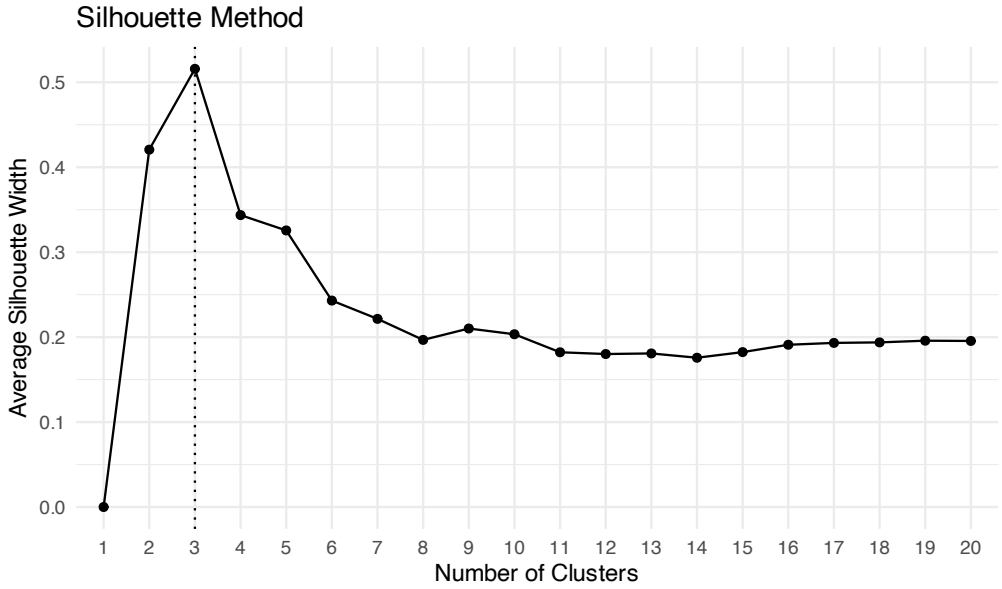
Ahora, hagamos lo mismo para el algoritmo **CLARA**. El código para seleccionar el número de clústeres óptimo es:

```
res_CLARA_Sil <- n_clusters_silhouette(datos_est, standardize = FALSE,
↪  distance_method = "euclidean",
    clustering_function = cluster::clara, n_max = 20)

res_CLARA_Sil
```

```
## The Silhouette method, based on the average quality of clustering, suggests that the optimal nu
```

En la Figura 6.2 se presentan las siluetas promedio para cada número de clústeres. En este caso, el número de clústeres óptimo es 3, que corresponde a una silueta promedio

Figura 6.1. Silueta promedio por número de clúster para el algoritmo PAM y distancia euclidiana



Fuente: calculos propios.

de 0.5139.

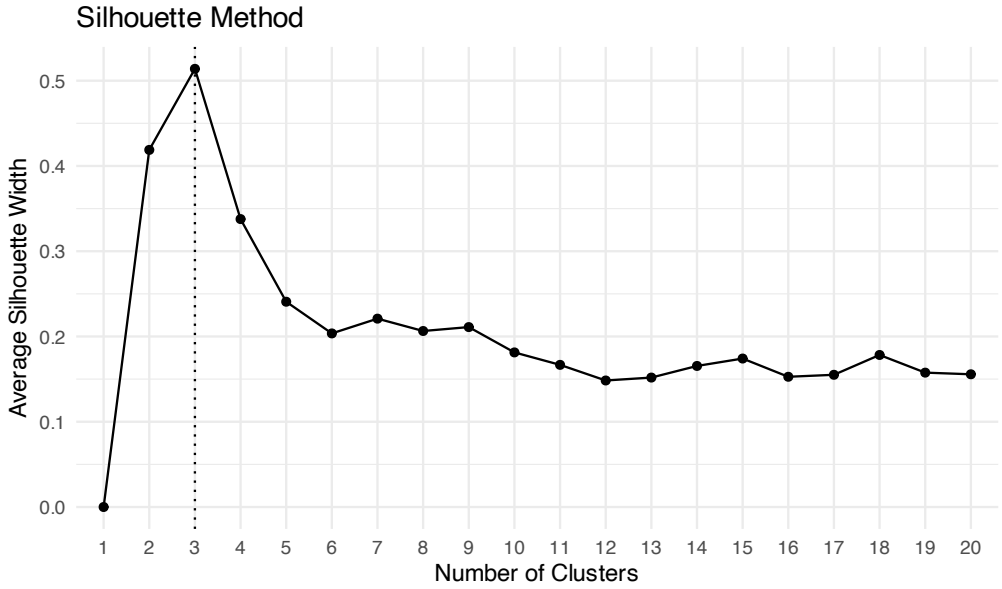
Los resultados hasta aquí los podemos resumir en el Cuadro 6.1.

Tabla 6.1. Siluetas y número de clústeres óptimos para diferentes aproximaciones de clústering (distancia euclidiana)

Método de aglomeración	Silueta promedio	Número óptimo de clústeres
Enlace único	0.3758	3
Enlace completo	0.5703	2
Enlace promedio	0.5703	2
Enlace mediano	0.3022	6
Centroide	0.5703	2
Ward.D	0.5157	3
Ward.D2	0.5148	3
McQuitty	0.5703	2
k-means	0.5157	3
PAM	0.5158	3
CLARA	0.5139	3

Fuente: cálculos propios.

Figura 6.2. Silueta promedio por número de clúster para el algoritmo CLARA y distancia euclidiana



Fuente: calculos propios.

6.5 Calculando siluetas individuales y membrecías para los algoritmos PAM y CLARA

Después de conocer el número óptimo de clústeres, podemos aplicar el respectivo algoritmo para continuar con nuestro análisis similar a lo que presentamos en las Secciones 4.7 y 5.4.3. Es importante recordar que esto solo se haría si alguno de estos métodos fueran la mejor forma de particionar los datos.

Por razones pedagógicas, en esta sección mostramos cómo calcular la silueta promedio y las membrecías para **PAM** y **CLARA**. No obstante, en nuestro caso específico no tendría sentido este análisis.

Para el caso de **PAM**, la función que usaremos para implementar el algoritmo es **pam()** del paquete *cluster* (Maechler et al., 2022). Esta función requiere en su forma más simple, un objeto de clase **data.frame** con los datos (**x**), el número de clústeres que queremos formar (**k**) y la medida de distancia (**metric**). Es decir, el código será

```
# fijamos la semilla aleatoria
set.seed(12345)
# calculamos los clústeres para k = 3
cluster_PAM <- pam(x = datos_est, k = 3, stand = FALSE, metric = "euclidean")
# Mirar todos los comportamientos del objeto creado
attributes(cluster_PAM)
```

```
## $names
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
##
## $class
## [1] "pam"       "partition"
```

El objeto que creamos con el algoritmo **PAM** (`cluster_PAM`) tiene varios compartimientos (*slots*) entre los cuáles podemos destacar **clustering** que contiene las membrecías de cada uno de los clientes. Por ejemplo, las membrecías de todos los clientes las podemos extraer fácilmente de la siguiente manera:

```
# Calcular membrecias
cluster_PAM$clustering
```

Algo similar podemos hacer para el algoritmo **CLARA** empleando la función **clara()** del paquete *cluster* (Maechler et al., 2022). Esta función tiene argumentos similares a la función **pam()**. Sin embargo, hay dos argumentos que son diferentes:

- **sampsize**: El tamaño de las submuestras que se emplearán. Por defecto **sampsize** = $40 + 2 * k$. (Paso 1 del algoritmo **CLARA**)
- **samples**: El número de veces que se repetirá el algoritmo. Es decir, el número de veces que se repetirá el algoritmo **PAM**, lo cuál implica el número de veces que se sacarán las muestras aleatorias (paso 5 del algoritmo **CLARA**). El valor por defecto es **samples = 5**.

En este caso, las membrecías se pueden calcular con el siguiente código:

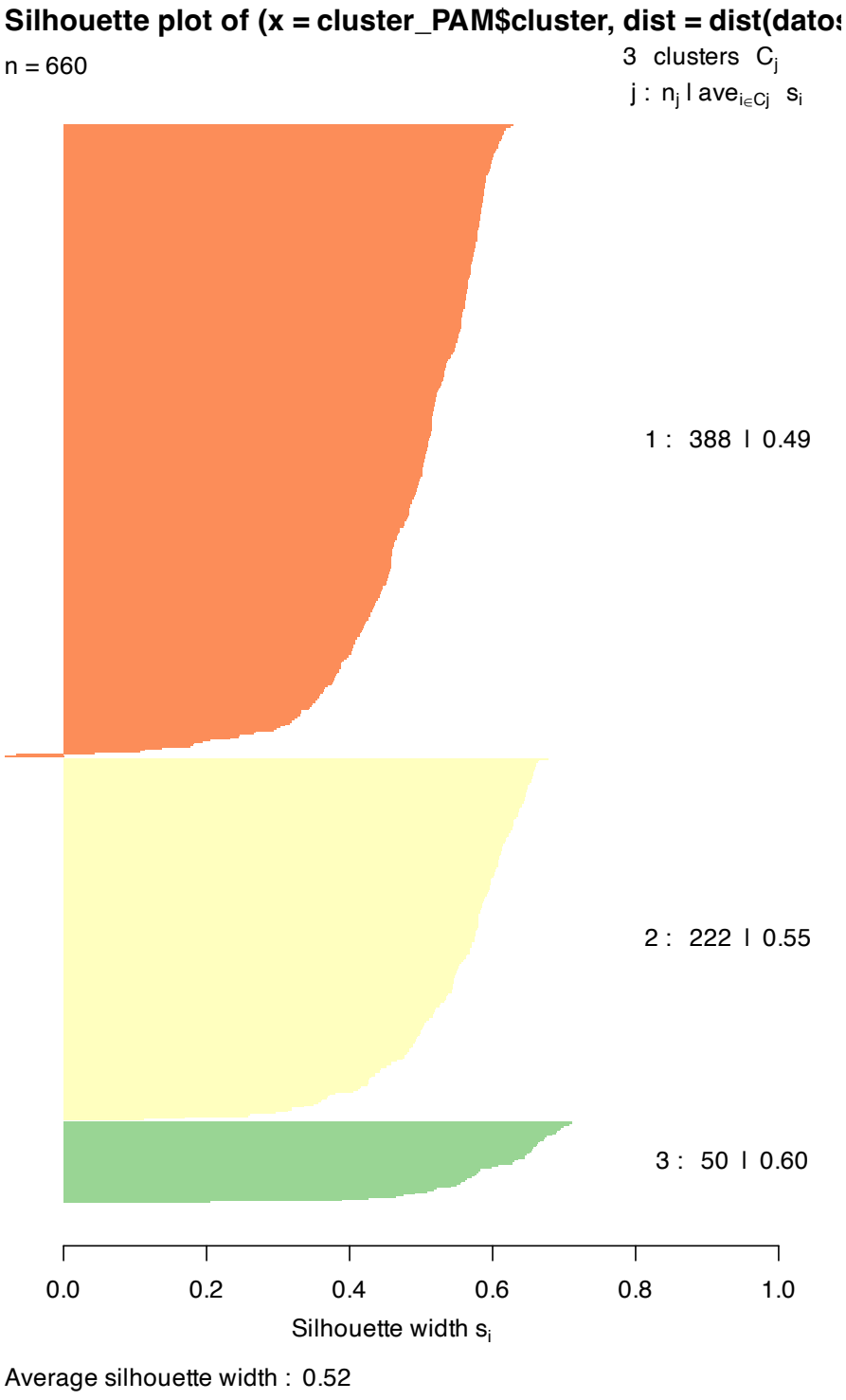
```
# Fijar la semilla aleatoria
set.seed(12345)
# Calcular los clústeres para k = 3
cluster_CLARA <- clara(x = datos_est, k = 3, stand = FALSE, metric =
↪  "euclidean")
# Mirar todos los comportamientos del objeto creado
attributes(cluster_CLARA)
```

```
## $names
##  [1] "sample"     "medoids"    "i.med"      "clustering" "objective"
##  [6] "clusinfo"   "diss"       "call"       "silinfo"    "data"
##
## $class
## [1] "clara"     "partition"
```

```
# Calcular membrecias
cluster_CLARA$clustering
```

Las membrecías se pueden emplear para construir las siluetas individuales. Recuerden que esto lo podemos hacer empleando la función **silhouette()** del paquete *cluster* (Maechler et al., 2022). Intenta reproducir la Figura 6.3

Figura 6.3. Silueta para tres clústeres emplendo PAM y distancia euclidiana



Fuente: calculos propios.

6.6 Comentarios Finales

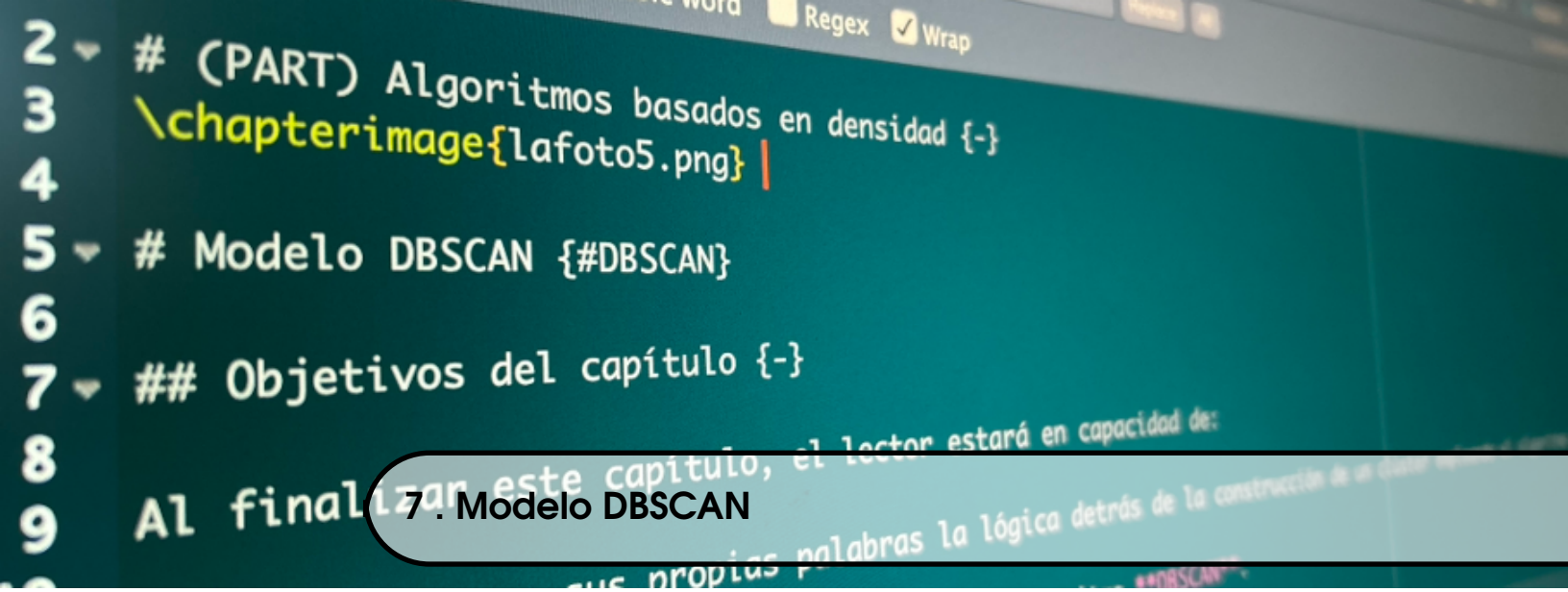
En este Capítulo estudiamos dos métodos de clústering particionado basados en centroides: **PAM** y **CLARA**. Estos dos algoritmos hacen parte de la caja de herramientas "estándar" para la tarea de hacer clústeres. Son relativamente comunes y se emplean como complemento al **k-means**.

En el método **PAM** se seleccionan observaciones como medoides iniciales y se actualizan iterativamente. Por otro lado, **CLARA** utiliza submuestras para manejar conjuntos de datos grandes y mejorar la eficiencia. Ambos métodos son útiles para agrupar datos en conjuntos homogéneos. Al combinarlos con **k-means** y los algoritmos jerárquicos (**AGNES** y **DIANA**), se pueden obtener un conjunto de algoritmos robustos para realizar el análisis de clústeres.

En los siguientes capítulos estudiaremos más métodos de clústering que te permitirán armar una caja de herramientas aún más robusta para responder a diversos problemas que impliquen armar conglomerados.

Parte IV

Algoritmos basados en densidad



Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Explicar en sus propias palabras la lógica detrás de la construcción de un clúster empleando el algoritmo **DBSCAN**.
- Construir en R clústeres empleando el algoritmo **DBSCAN**.

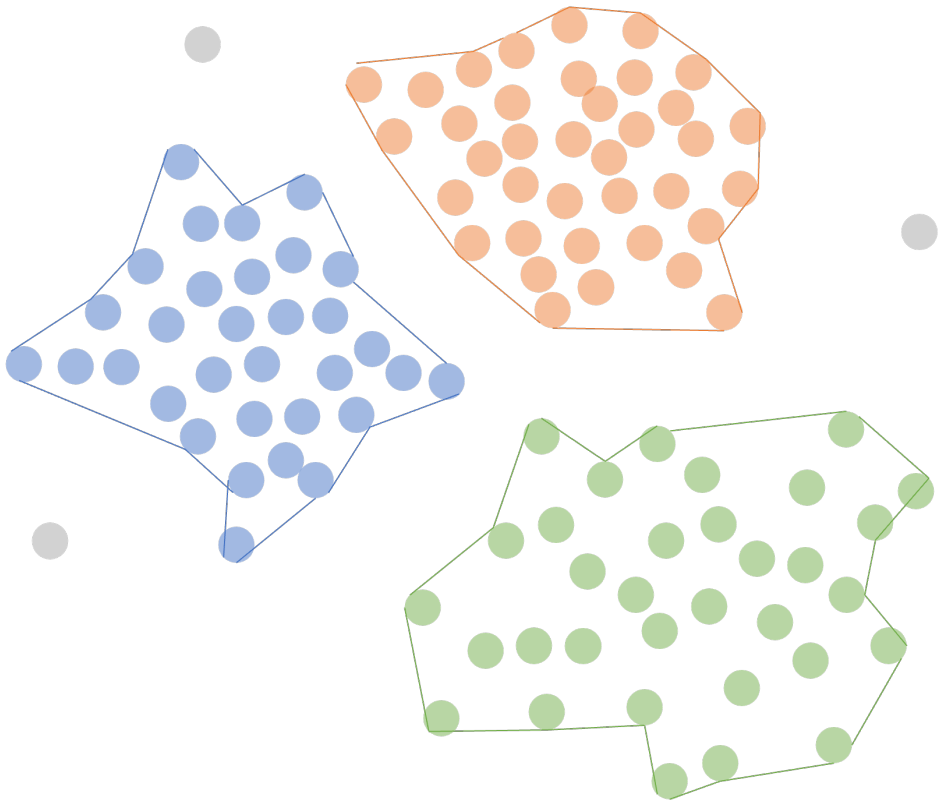
7.1 Introducción

En el Capítulo anterior continuamos con el estudio de algoritmos de clústering particionado basado en centroides. Con el estudio de los algoritmos basados en medoides del Capítulo 6 ya tenemos una lista grande algoritmos. Recordemos que hemos estudiado los siguientes modelos:

- **AGNES** (algoritmo jerárquico aglomerativo)
- **DIANA** (algoritmo jerárquico de división)
- **k-means** (algoritmo particionado basado en centroides)
- **PAM** (algoritmo particionado basado en medoides)
- **CLARA** (algoritmo particionado basado en medoides)

En este Capítulo estudiaremos un algoritmo de clústering particionado basado en densidad. En general, este tipo de algoritmos basado en densidad se caracteriza por identificar regiones de alta densidad de individuos separadas por regiones de baja densidad en el espacio de datos (Ver Figura 7.1).

Figura 7.1. Clústeres particionados basados en densidad



Fuente: elaboración propia.

El algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise) es

uno de los enfoques más populares en esta categoría y se destaca por su capacidad para identificar grupos de formas arbitrarias y su robustez ante ruido en los datos[1] y datos atípicos.

7.2 El algoritmo DBSCAN

DBSCAN fue propuesto por Ester et al. (1996) como una solución a la dificultad de encontrar conglomerados de forma arbitraria en conjuntos de datos con diferentes densidades y formas. La primera diferencia de este con los algoritmos de clústering particionado estudiados hasta ahora (**k-means**, **PAM** y **CLARA**), es que **DBSCAN** no necesita especificar el número de conglomerados. **DBSCAN** detecta automáticamente el número de conglomerados basándose en los datos y parámetros de entrada. Otra diferencia con los algoritmos estudiados hasta ahora (particionados y jerárquicos), es que **DBSCAN** puede encontrar clústeres de formas arbitrarias. Por ejemplo, en la Figura 7.2 (primera fila) podemos observar cómo **DBSCAN** puede detectar un clúster rodeado por otro clúster, mientras que un algoritmo como el **k-means** divide el espacio de una manera lineal. También puede encontrar formas no lineales, como en la segunda y tercera fila de la Figura 7.2.

No obstante, aunque el algoritmo **DBSCAN** puede encontrar formas no tradicionales en algunas ocasiones puede detectar los mismos clústeres que detectarían otros algoritmos como el **k-means** (Ver línea uno de la Figura 7.3). Por otro lado, cuando los individuos se encuentran relativamente concentrados (Ver línea uno de la Figura 7.3) es poco probable que **DBSCAN** pueda construir grupos, como sí lo hacía el algoritmo **k-means**.

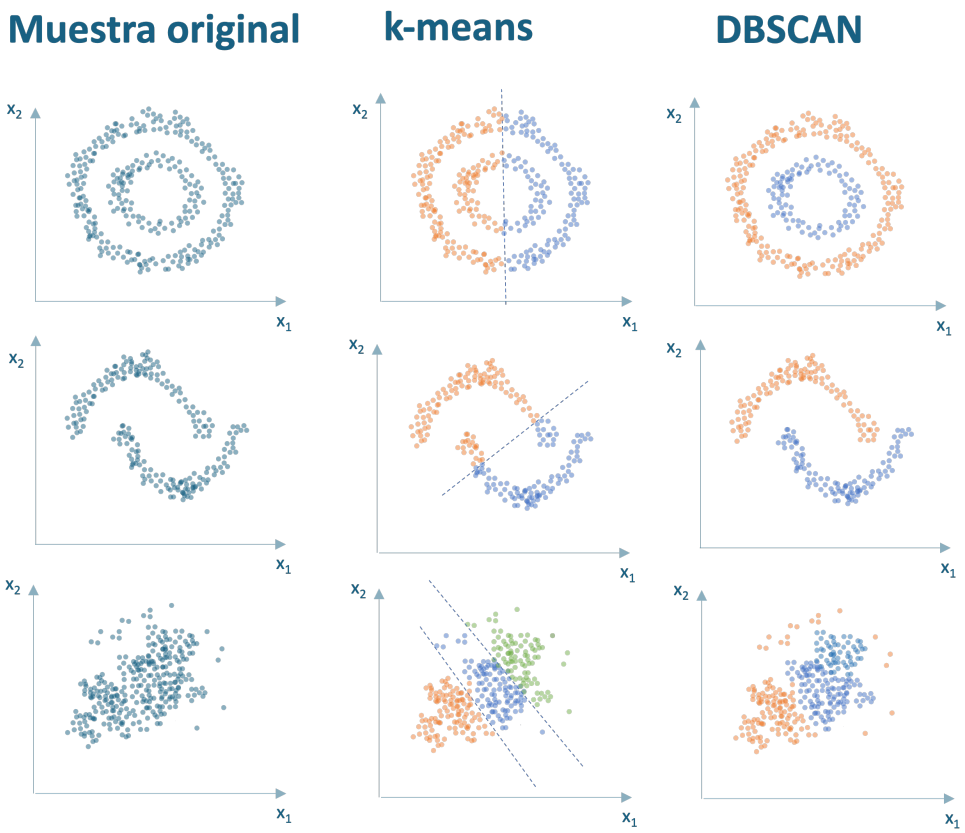
Por otro lado, **DBSCAN** puede tratar el ruido y los valores atípicos. Todos los valores atípicos serán identificados y marcados sin ser clasificados en ningún clúster (Ver línea uno de la Figura 7.4). Por lo tanto, DBSCAN también puede utilizarse para la detección de anomalías (detección de valores atípicos) (Kassambara, 2017).

Antes de describir los pasos del algoritmo **DBSCAN**, tenemos que definir dos parámetros: **eps** (ϵ) y **minPts**. **eps** (ϵ) que representan el radio máximo del vecindario; en otras palabras, es la distancia máxima entre dos puntos para que se consideren vecinos y por tanto pertenezcan al mismo grupo. Esto corresponde al vecindario ϵ (ϵ-**neighborhhood**) de un determinado punto central (*core point*). Por otro lado, **minPts** es el número mínimo de puntos que deben estar dentro de la vecindad de un punto para que este sea considerado un punto central y, por tanto, se configure un clúster (incluyendo el mismo punto).

Intuitivamente, cada individuo en la muestra que tenga un número de vecinos (en el vecindario ϵ) igual o mayor **minPts** será marcado como un punto central (Ver Figura 7.5). Por otro lado, si un individuo w tiene un número de vecinos inferior a **minPts**, pero

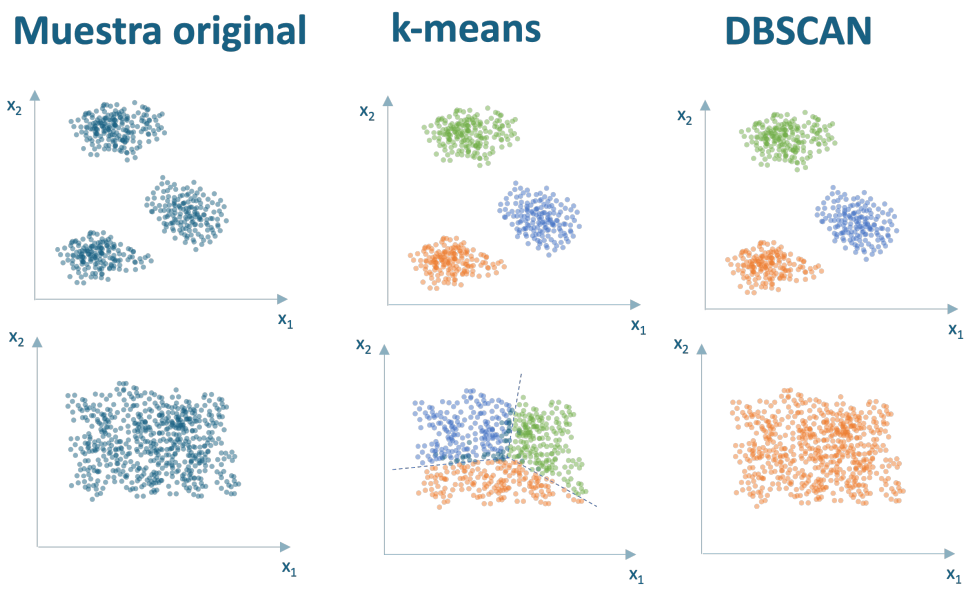
[1] En la jerga del científico de datos, el término "ruido" se refiere a los datos que no siguen el patrón general o la estructura subyacente de los datos. En el contexto de la tarea de clústering, el ruido se refiere a los individuos que no pertenecen claramente a ningún clúster o que pueden interferir con la identificación de clústeres con mayor definición.

Figura 7.2. Ejemplo 1 de los diferentes clústeres que pueden formar DBSCAN y k-means apartir de la misma muestra de individuos con dos características



Fuente: elaboración propia.

Figura 7.3. Ejemplo 2 de los diferentes clústeres que pueden formar DBSCAN y k-means a partir de la misma muestra de individuos con dos características



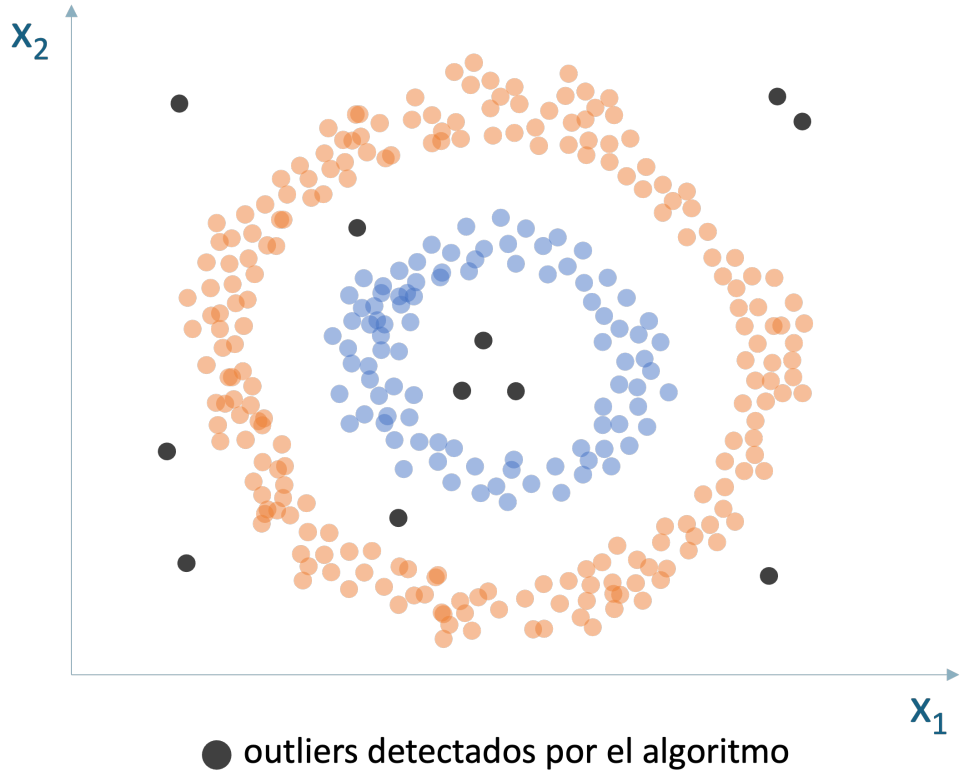
Fuente: elaboración propia.

pertenece al vecindario ϵ (ϵ-**neighborhood**) de otro punto, entonces w será denominado un punto de frontera (*border point*) (Ver Figura 7.5). Y si un punto no es punto central (*core point*) ni uno en la frontera (*border point*), entonces será denominado ruido (*noise point*) o *outlier* (Ver Figura 7.5).

Como lo hemos hecho en los capítulos anteriores, veamos los pasos del algoritmo para entenderlo mejor. Los pasos para realizar un clústering particionado usando el algoritmo **DBSCAN**, son:

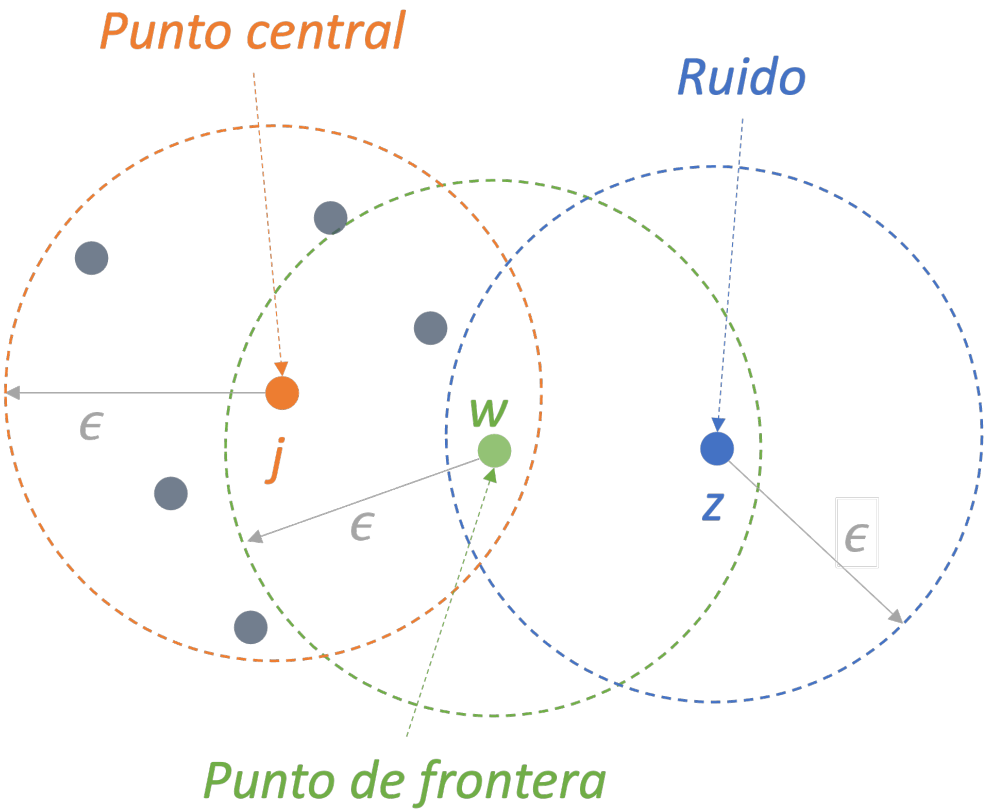
1. Marcar todos los individuos como "no visitados".
2. Seleccionar un individuo al azar entre los puntos no visitados anteriormente y calcular la distancia de este con respecto a los demás individuos.
3. Si se encuentra que el individuo tiene un número igual o superior de vecinos a **minPts** que se encuentren a una distancia menor o igual a **eps** (ϵ) (el individuo es un punto central), crear un nuevo clúster y marcar al individuo como visitado.
4. Agregar al clúster todos los individuos vecinos del individuo inicial que no hayan sido visitados y que se encuentren dentro de la distancia **eps** (ϵ).
5. Repetir los pasos 3 y 4 para todos los vecinos del individuo inicial, hasta que no hayan más individuos que agregar al clúster.
6. Marcar el individuo inicial como "visitado".
7. Repetir pasos 2 a 5 para todos los individuos no visitados, hasta que no queden individuos sin visitar.

Figura 7.4. Algoritmo DBSCAN detectanto outliers



Fuente: elaboración propia.

Figura 7.5. Clasificación de individuos en el algoritmo DBSCAN con minPts = 6



Fuente: adaptación de Kassambara (2017).

8. Asignar a cada punto la membrecía al clúster en el que fue clasificado. Para los individuos que no se encuentran dentro de ningún clúster se marcan como ruido o *outliers* y se les asigna una etiqueta especial.

El algoritmo **DBSCAN** no requiere la definición previa del número de clústeres, dado que el número óptimo de clústeres q es un resultado del este algoritmo. Antes de emplear la función, es importante reconocer que, en este caso, el número de clústeres óptimo (q) es seleccionado en el mismo proceso de maximización.

En este orden de ideas, no tiene mucho sentido emplear la silueta promedio para escoger el número óptimo de clústeres, procedimiento que sí tenía sentido para todos los modelos de clústering estudiados hasta el momento[2].

Por otro lado, sí es posible encontrar el valor óptimo del parámetro **eps** (ϵ). Chen et al. (2020) sugieren emplear la media de las distancias de cada individuo a sus k vecinos más cercanos (distancia **kNN**). Esto implica calcular la distancia **kNN** para cada individuo y diferentes valores del parámetro **eps** y posteriormente calcular la suma cuadrada de las distancias **kNN**. Empleando ese resultado podemos graficar, para los diferentes valores de **eps**, el número mínimo de observaciones que se encontraron en el ejercicio de clusterización. Posteriormente, se puede identificar por medio del método del "codo" el valor óptimo de **eps**. En otras palabras, buscar el punto en la gráfica donde la pendiente cambia significativamente. En la siguiente sección estudiaremos cómo implementar este algoritmo en R.

7.3 Implementación del algoritmo DBSCAN en R

Este algoritmo se puede implementar en R por medio de la función **dbscan()** del paquete *dbscan* (Hahsler et al., 2019). Pero antes de emplear esta función es necesario conocer cuál es el valor óptimo del parámetro **eps**. La propuesta de Chen et al. (2020) se puede implementar en R empleando la función **n_clusters_dbscan()** del paquete *parameters*. Esta función es similar a la función **n_clusters_silhouette()** que empleamos en el Capítulo 6.

Esta función tiene los siguientes argumentos:

n_clusters_dbscan(x, standardize = TRUE, include_factors = FALSE, eps_range = c(0.1, 3), distance_method = "euclidean")

donde:

- **x**: Objeto con datos de clase **data.frame**.
- **standardize**: Si es **TRUE** se estandarizarán los datos, este es el valor por defecto.
- **include_factors**: Si es **TRUE**, las variables de clase **factor** se convierten en objetos numéricos para ser incluidos en los datos, para determinar el número de clústeres. Por defecto, es igual a **FALSE**; es decir, se eliminan las variables que sean de clase

[2]Recuerda que en el caso de los algoritmos particionados, empleamos el algoritmo para diferentes números de clústeres y se selecciona el número de clústeres óptimo (q) con métricas o con la silueta promedio. Para el caso de los algoritmos jerárquicos, éstos encuentran todos los posibles clústeres. Con las métricas o con la silueta decidimos por dónde podar el árbol jerárquico (q).

factor. Esto se hace porque la mayoría de los métodos que determinan el número de conglomerados solo funciona con variables cuantitativas.
- **min_size**: El número mínimo de individuos (incluyendo al mismo individuo) requeridos en el vecindario ϵ (ϵ-**neighborhood**) para ser considerado como un punto central (*core point*). Por defecto, el valor es el 10% de la muestra (**min_size = 0.1**). Si se emplea un entero, este será el número de observaciones. Equivale al parámetro **minPts**.
- **eps_range**: El rango sobre el cuál se evaluarán los posibles valores del parámetro **eps**.
- **method**: El método para escoger el parámetro **eps** óptimo. En nuestro caso empleemos **method = "SS"** para que se calcule la suma cuadrada.
- **distance_method**: El tipo de distancia a calcular. Este elemento es pasado a la función **dist()**. Por defecto el método es **"euclidean"**; los otros posibles valores son "maximum", "manhattan", "canberra", "binary" y "minkowski". No obstante, para el caso del algoritmo **DBSCAN**, no se encuentra habilitada la opción para otra distancia.

El valor óptimo del parámetro **eps** (ϵ) para el algoritmo **DBSCAN** y el método de la distancia **kNN** lo podemos encontrar con el siguiente código, empleando nuestros datos ya estandarizados (`datos_est`) y la distancia euclidiana.

```
res_dbscan_knn <- n_clusters_dbscan(datos_est, standardize = FALSE, eps_range =
↪  c(0.001,
    3), min_size = 5, distance_method = "euclidean", method = "SS")
```

Los resultados del **eps** óptimo lo podemos visualizar imprimiendo el resultado en la consola o por medio de la Figura 7.6

```
res_dbscan_knn
```

```
## The DBSCAN method, based on the total clusters sum of squares, suggests that the optimal eps =
```

```
plot(res_dbscan_knn)
```

El resultado que minimiza la suma de las distancias **kNN** corresponde a un 0.5518367 Noten que en este caso esto implica 16 clústeres.

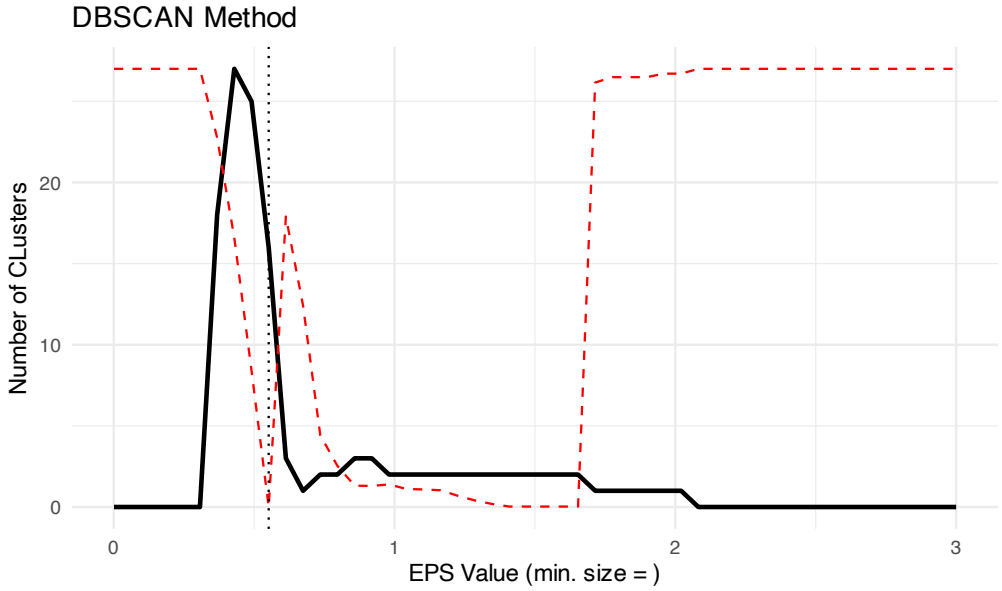
El valor óptimo del parámetro **eps** (ϵ) se puede extraer de la siguiente manera:

```
# Extraer el valor optimo de eps

eps_opt <- attributes(res_dbscan_knn)$eps

eps_opt
```

```
## [1] 0.5518367
```

Si bien, en este caso no tiene mucho sentido calcular la membrecía de cada individuo, pues no tenemos clústeres, continuaremos el ejercicio por razones pedagógicas.

Figura 7.6. Seleccionando el valor óptimo del parámetro eps por medio de la distancia kNN



Fuente: calculos propios

La partición de los datos se puede realizar empleando la función **dbscan()** del paquete *dbscan* (Hahsler et al., 2019). Esta función solo requiere como argumentos:

- **x**: Los datos estandarizados o una matriz de proximidad[3].
- **eps**: El parámetro (ϵ) que representa el radio máximo del vecindario.
- **minPts**: El número mínimo de individuos (incluyendo al mismo individuo), requeridos en el vecindario ϵ (ϵ-**neighborhood**) para ser considerado como un punto central (*core point*).

En nuestro caso:

```
# Instalar el paquete si no se tiene

# install.packages('dbscan')

# Cargar el paquete

library("dbscan")


# DBSCAN
```

[3]Si se quisiera emplear otro tipo de distancia diferente a la euclidiana, esto se puede realizar empleando una matriz de proximidad con la función **dist()**.

```
res_DBSCAN <- dbscan::dbscan(x = datos_est, eps = eps_opt, minPts = 5)
```

Las membrecías se pueden encontrar fácilmente con el siguiente código:

```
# Resultados de la asignación
head(res_DBSCAN$cluster)
```

```
## [1] 0 0 0 0 0 1
```

Por otro lado, si bien emplear la silueta para escoger el número óptimo de grupos para el algoritmo **DBSCAN** no tiene sentido, es posible calcular la silueta promedio. Y si es el caso, puede emplearse para comparar esta aproximación con otros algoritmos (realizar la validación del clúster).

En todo caso, será importante recordar que la silueta promedio casi siempre favorecerá modelos como los jerárquicos y particionados basados en centroides (**k-means**, **PAM** y **CLARA**) que a modelos como el **DBSCAN**. Esto se debe a que la aproximación de **DBSCAN** y **GMM** permiten crear conglomerados "largos y delgados" y hasta "envolver otros conglomerados"; es decir, por construcción no intentan optimizar las distancias intra e inter conglomerados, que es lo que mide la silueta. Y por ejemplo, **k-means** sí intenta, más o menos, optimizar estas dos cantidades.

En nuestro caso, cada silueta se podría calcular con el siguiente código:

```
# Calcular las siluetas individuales
sil_DBSCAN <- silhouette(res_DBSCAN$cluster, dist(datos_est))
# Calcular la silueta promedio
mean(sil_DBSCAN[, 3])
```

```
## [1] -0.09057502
```

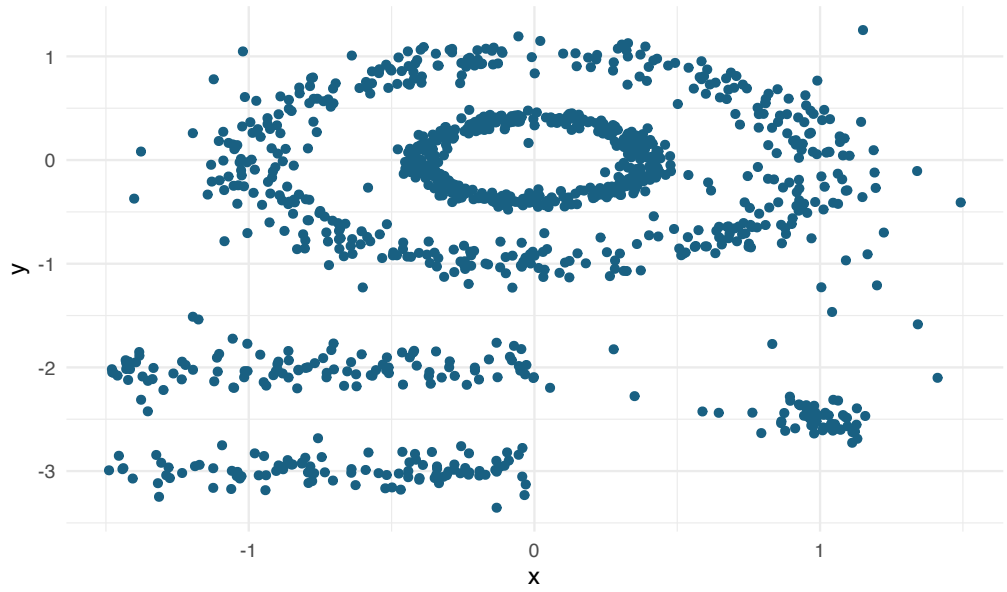
7.4 Otro ejemplo

Para mostrar las bondades de este algoritmo, empleemos un ejemplo con datos del paquete *factoextra* (Kassambara y Mundt, 2020). Estos datos tienen una distribución no lineal que permite entender mejor las bondades de **DBSCAN**. Carguemos los datos en el objeto `multishapes`y solo empleemos las dos primeras variables. Estos datos ya están estandarizados.

```
# Cargar el paquete
library(factoextra)
# Cargar los datos
data("multishapes")
# Seleccionar las dos primeras variables
multishapes <- multishapes[, 1:2]
```

La Figura 7.7 nos permite observar la no linealidad de los datos; algo similar a lo presentado en la Figura 7.2 (primera fila).

Figura 7.7. Datos del ejemplo 2



Fuente: paquete factoextra

Para poder comparar los resultados de **DBSCAN**, empleemos el algoritmo **k-means**. Puedes encontrar que, empleando la distancia euclidiana y el criterio de la silueta, el número óptimo de clústeres será 2 (el código no se presenta intencionalmente). Los datos y sus respectivas membrecías se reportan en la Figura 7.8. Con el algoritmo **k-means** encontramos 2 clústeres.

Ahora empleemos el algoritmo **DBSCAN**. Primero optimicemos el parámetro **eps**.

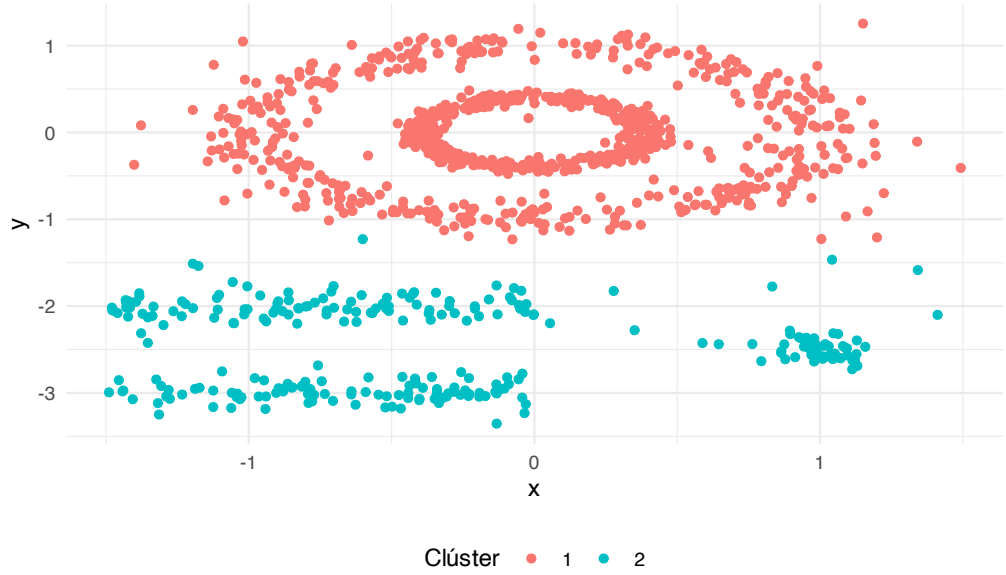
```
res_eje2_dbscan <- n_clusters_dbscan(multishapes, standardize = FALSE,
↪  eps_range = c(0.001,
    3), distance_method = "euclidean", min_size = 5, method = "SS")
```

El valor óptimo de **eps** es 0.1234082 lo que implicará 7 clústeres.

En la Figura 7.9 se presenta la partición generada por el algoritmo **DBSCAN**. El clúster cero corresponde a los outliers marcado por el algoritmo.

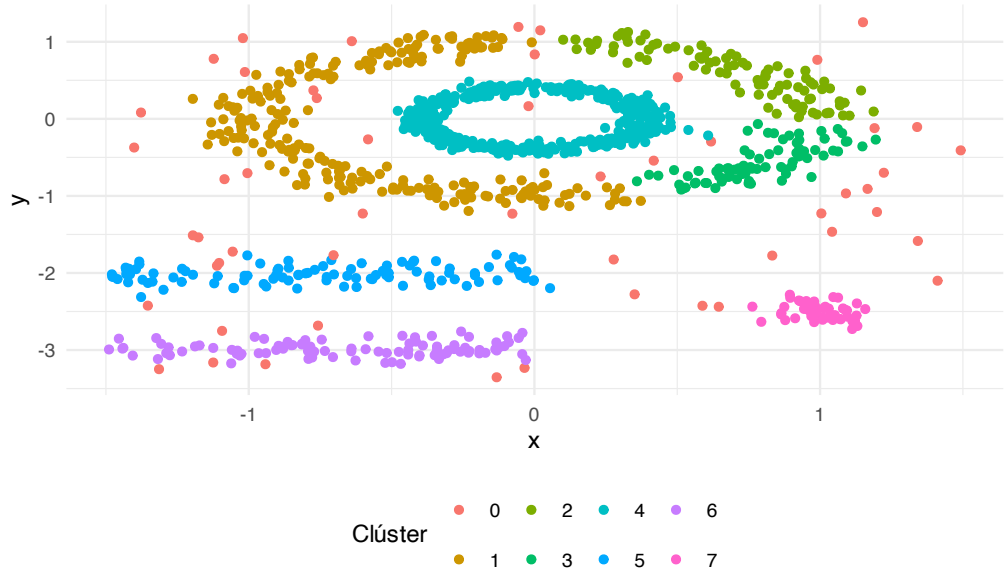
Observando los resultados, parece ser mucho mejor emplear **DBSCAN** que el algoritmo **k-means** para generar la partición de los datos. No obstante, al emplear la silueta para realizar la comparación de los dos algoritmos, encontraremos (¡Intenta reproducir estos resultados!) que la silueta promedio para la partición empleando **k-means** es de 0.418 y para el caso del agrupamiento generado por **DBSCAN** la silueta promedio es de 0.307. Esto nos lleva, de nuevo, a la observación que hacíamos más arriba. La silueta

Figura 7.8. Partición de los datos del ejemplo 2 empleando k-means



Fuente: elaboración propia.

Figura 7.9. Partición de los datos del ejemplo 2 empleando DBSCAN



Fuente: elaboración propia.

promedio, por su construcción, tiende a favorecer particiones lineales como **k-means** y casi siempre una aproximación como **DBSCAN** será penalizada fuertemente por esta métrica. Por eso, es importante la intervención del científico de datos en el momento de construcción de los clústeres. La selección automática de la mejor aproximación para particionar los datos puede llevarnos a estas contradicciones. Es importante entender en qué contexto nos encontramos, y si tiene o no sentido emplear determinada métrica para tomar esta decisión. En este ejemplo, es evidente que la silueta promedio no es la mejor opción para seleccionar la "mejor" aproximación para particionar los datos.

7.5 Comentarios finales

En este capítulo hemos estudiado un tipo de clústering particionado basado en la densidad de los datos: **DBSCAN**. El algoritmo **DBSCAN** busca, de manera iterativa, aquellos individuos que se encuentran en un radio inferior a una distancia previamente establecida (parámetro **eps**). Con esta búsqueda, arma los conglomerados de acuerdo a dónde exista mayor densidad de individuos. De hecho este algoritmo depende de dos parámetros: el radio (distancia) para establecer los vecinos (**eps**) y el número mínimo de individuos que deben estar dentro de la vecindad, para que el individuo se configure en un clúster (**minPts**).

Los resultados de este algoritmo son muy sensibles a la escogencia de estos parámetros, en especial del parámetro **eps**. Por esta razón, también estudiamos una estrategia para fijar, a partir de los datos, este parámetro.

Finalmente, es importante recordar que el algoritmo **DBSCAN** es una herramienta más para realizar la tarea de hacer clústeres, que debes tener en tu caja de herramientas. Como las otras herramientas que hemos estudiado, **DBSCAN** no es una solución mágica que funcione en todos los casos, pero tiene algunas ventajas significativas sobre otros algoritmos de agrupamiento.

Por ejemplo, **DBSCAN** no requiere que especifiques el número de clústeres por adelantado. Esto puede ser una gran ventaja, ya que a menudo es difícil saber cuántos clústeres hay en un conjunto de datos. DBSCAN encontrará automáticamente el número de clústeres que mejor se ajuste a los datos.

DBSCAN es capaz de encontrar clústeres de formas arbitrarias. Esto es importante, ya que los clústeres en el mundo real no siempre son bonitos y redondos. **DBSCAN** puede encontrar clústeres que sean alargados, ramificados o incluso con forma de anillo. **DBSCAN** también es relativamente eficiente. Esto significa que se puede ejecutar en conjuntos de datos grandes sin tomar demasiado tiempo.

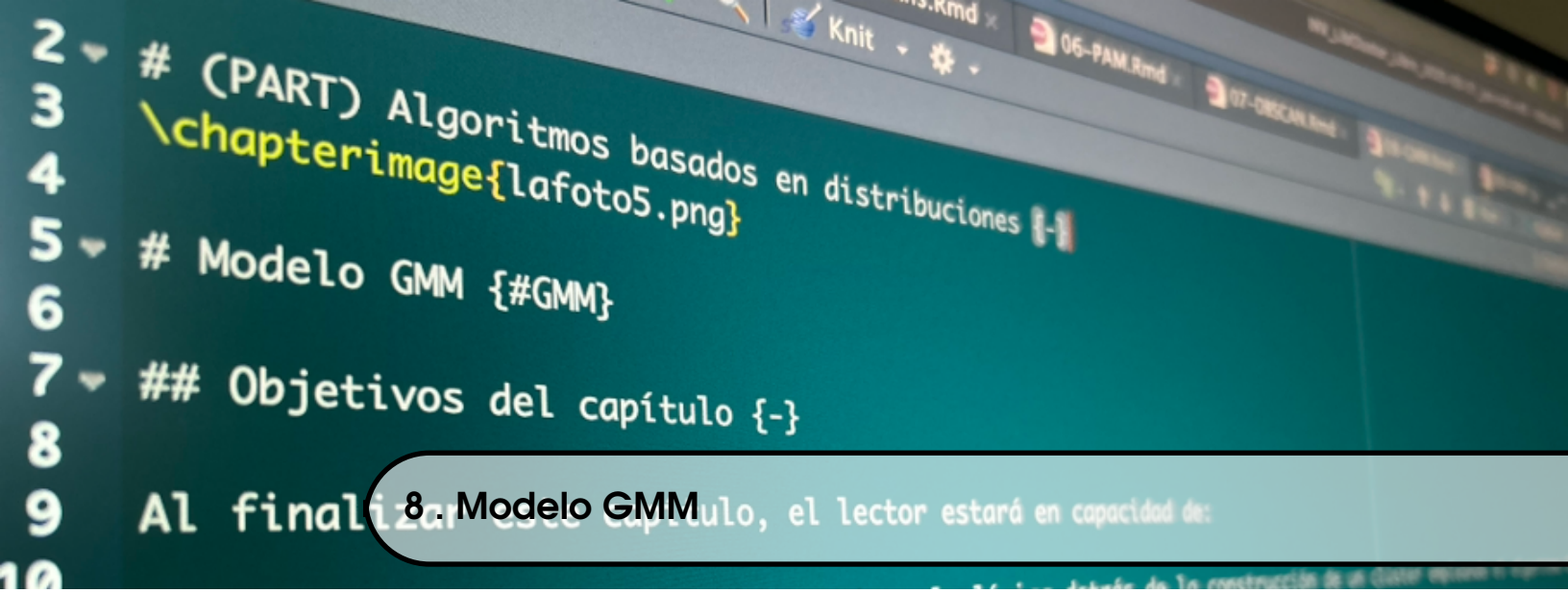
Sin embargo, **DBSCAN** también tiene algunas desventajas. Como lo acabamos de mencionar, **DBSCAN** puede ser sensible a la elección de los parámetros. Por construcción, **DBSCAN** puede tener dificultades para encontrar clústeres de baja densidad.

En general, **DBSCAN** es una herramienta poderosa para el agrupamiento de datos. Es importante ser consciente de sus ventajas y desventajas para poder utilizarlo de forma eficaz. En el siguiente capítulo continuaremos nuestro análisis de algoritmos para

construir clústeres. Estudiaremos una filosofía totalmente diferente a la construcción de agrupaciones.

Parte V

Algoritmos basados en distribuciones



8. Modelo GMM

Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Explicar en sus propias palabras la lógica detrás de la construcción de un clúster empleando el algoritmo GMM.
- Construir en R clústeres empleando el algoritmo GMM.

8.1 Introducción

La tarea de hacer clústeres se puede responder con diferentes modelos. Ya hemos estudiado los algoritmos jerárquicos como **AGNES** y **DIANA** (Ver Capítulos 3 y 4 para una discusión de estos modelos). En el Capítulo 2 vimos que existe otra gran categoría de modelos de clústering: los algoritmos particionados. A su vez, hemos estudiado algoritmos de clústering particionados **basados en centroides**, como por ejemplo los modelos **k-means** (Ver Capítulo 5), **k-means++** (Ver Capítulo 5), **PAM** (Ver Capítulo 6) y **CLARA** (Ver Capítulo 6). También hemos estudiado el algoritmo **DBSCAN**, que es un algoritmo particionado **basado en densidad** (Ver Capítulo 7).

Tanto los modelos de clústering jerárquico como los particionados estudiados hasta ahora emplean un enfoque heurístico[1] para construir clústeres; es decir, no se basan en un modelo formal.

En este capítulo nos concentraremos en los modelos particionados **basados en modelos** (también conocidos como **basados en la distribución**). Los algoritmos de clústering basados en modelos asumen un modelo generador de datos que implica estimar unos parámetros, entre los cuales se encuentra el número de grupos.

Estos modelos parten de los datos y tienen como objetivo encontrar la mezcla de distribuciones normales que pudieron generar los datos (ver Figura 8.1). La tarea de este algoritmo es encontrar el número de clústeres y la media, varianza y covarianzas de cada distribución multivariada normal (gaussiana) que pudo generar los datos. Por ejemplo, en la Figura 8.1 cada uno de los conglomerados representados corresponden a una distribución normal bivariada (con una media para cada variable, una varianza para cada variable y las respectivas covarianzas); esto es lo que se representa con las elipses y círculos concéntricos. A medida que nos acercamos al centro es mucho más probable (colores más fuertes) que los individuos pertenezcan al clúster y a medida que se alejan del centro (la media) la probabilidad disminuye (color más tenue).

En la siguiente sección estudiaremos el modelo que se encuentra por detrás de este algoritmo y, posteriormente, estudiaremos cómo implementar este método en R.

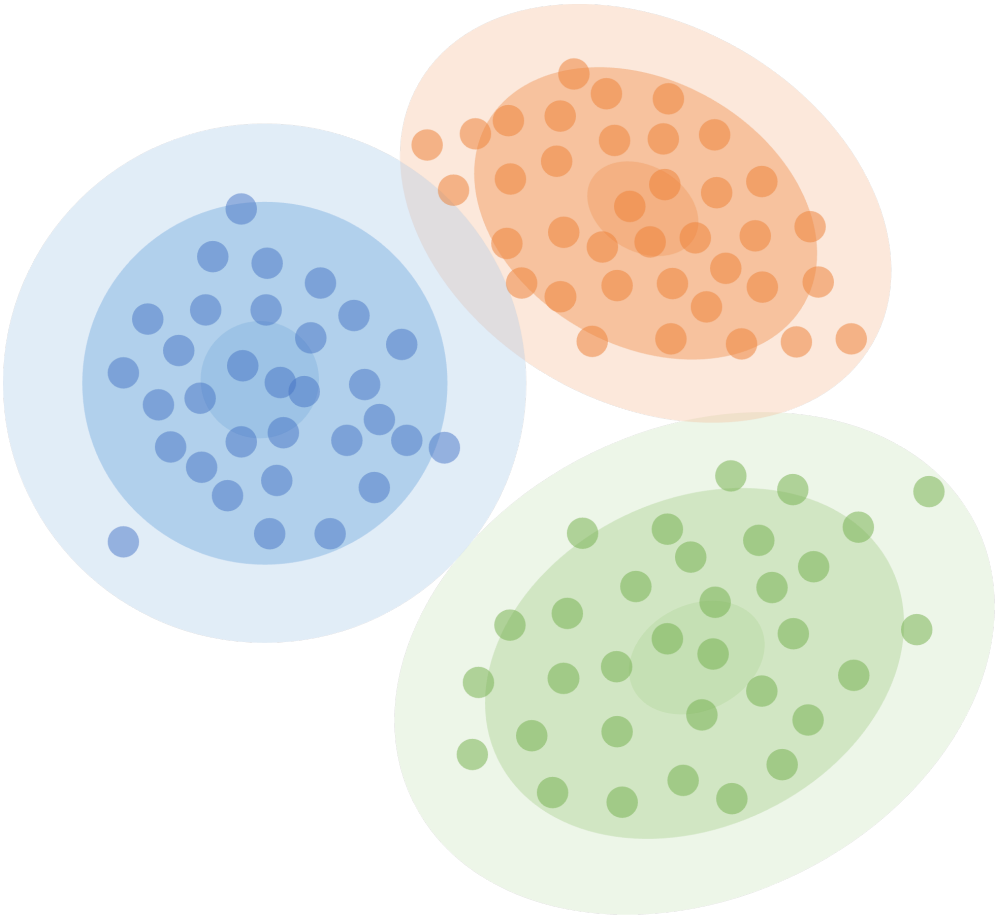
8.2 Formalmente

Los modelos de mezclas gaussianas (**GMM**, por sus siglas en inglés) son una técnica de clústering que se basa en la idea de que los datos observados debieron ser fruto de una muestra aleatoria que proviene de una población, cuya función de probabilidad es una mezcla de varias distribuciones gaussianas. En otras palabras, supone que el proceso generador de los datos es una mezcla de distribuciones normales, donde cada una de estas distribuciones representa un clúster.

Formalmente, reconozcamos que los valores observados para cada unas de las d variables (*feature*) cuantitativas, observadas para cada uno de los n individuos ($x = [x_1 \quad x_2 \quad \cdots \quad x_d]$) son realizaciones de d variables aleatorias $\mathbf{x} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_d]$ (por

[1] En la ciencia de datos, el término heurístico se emplea en el mismo sentido de la ingeniería. Es decir, una heurística es una aproximación que emplea la experiencia como ayuda para resolver un problema.

Figura 8.1. Representación de un algoritmo de clústering particionado basado en distribuciones



Fuente: elaboración propia.

ahora, omitiremos el subíndice i para evitar confusiones). El modelo **GMM** supone que este vector de *features* provienen de una mezcla de distribuciones normales. Es decir, para cada individuo i tendremos:

$$p(\mathbf{x}_i) = \sum_{k=1}^{q} \pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{8.1}$$

donde q es le número (óptimo en este caso) de conglomerados o componentes gaussianos, π_k es el peso del clúster k. Además, estos pesos deben garantizar que $\sum_{k=1}^{K} \pi_k = 1$. μ_k es el vector de medias para el conglomerado k, Σ_k es la matriz de varianzas y covarianza del clúster k y $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ es la función de densidad de probabilidad gaussiana (normal) multivariada para el conglomerado k.

El objetivo del **GMM** es encontrar los parámetros $\theta_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$ para $k = 1, 2, ...q$ y q que maximice la probabilidad de que los datos provengan de dicha mezcla de distribuciones normales. En otras palabras, este es un problema de estimación por el método de máxima verosimilitud en el que queremos resolver el siguiente problema:

$$L(\theta, q) = p(\mathbf{x}_1) \cdot p(\mathbf{x}_2) \cdots p(\mathbf{x}_n) = \prod_{i=1}^{n} p(\mathbf{x}_i) \tag{8.2}$$

donde $p(\mathbf{x}_i)$ está definida por (8.1) y $\theta = \{\theta_1, \theta_2, \ldots \theta_q\}$. De esta manera el problema será

$$\left(\hat{\theta}, \hat{q}\right) = arg \max_{\theta, q} L(\theta, q) \tag{8.3}$$

sujeto a las siguientes restricciones:

- $\sum_{k=1}^{K} \pi_k = 1$
- $q > 0$

En otras palabras, el objetivo del **GMM** es encontrar los parámetros θ y q que maximizan la verosimilitud de los datos observados. Este problema se puede resolver empleando el algoritmo **EM** (*Expectation-Maximization*)[2]. En el paso de *Expectation* se calculan las probabilidades de pertenencia de cada individuo a cada grupo. Estas probabilidades se conocen como responsabilidades. Y en el paso *Maximization* se actualizan los parámetros del modelo, utilizando las responsabilidades calculadas en el paso *Expectation*. Estos pasos se repiten hasta que la verosimilitud converja; es decir, no cambie mucho de una iteración a otra. Al final, cada individuo se asigna al clúster con la mayor responsabilidad.

Los **GMM** tienen la ventaja de poder modelar conglomerados con diferentes formas y tamaños, así como proporcionar una asignación de clúster suave (*Soft Clustering*)[3].

[2]Para una discusión del algoritmo **EM** puedes consultar Alonso (2002).

[3]El *Soft Clustering* se refiere a que este algoritmo permite asignar probabilidades a que los individuos pertenezcan a cada uno de los conglomerados, en lugar de asignarlos de manera estricta a un grupo, como lo hacen los algoritmos hasta aquí estudiados.

Sin embargo, son sensibles a la inicialización y pueden ser computacionalmente costosos en conjuntos de datos grandes debido al ajuste de múltiples distribuciones gaussianas.

Es importante anotar que está bien documentado que el algoritmo **k-means** puede expresarse como un caso especial del modelo **GMM**. Al igual que con **k-means**, la estimación de un **GMM** por medio del algoritmo **EM** puede ser sensible a las condiciones iniciales.

8.3 Implementación del modelo GMM

El algoritmo **GMM** se puede implementar con la función **Mclust()** del paquete *mclust* (Scrucca et al., 2023). Antes de emplear la función, es importante reconocer que, en este caso, el número de clústeres óptimo (q) es seleccionado en el mismo proceso de maximización. En este orden de ideas, no tiene mucho sentido emplear la silueta promedio para escoger el número óptimo de grupos, procedimiento que sí tenía sentido para todos los modelos de clústering estudiados hasta el momento (excepto **DBSCAN**). Por otro lado, la silueta promedio sí es posible calcularla y puede emplearse para comparar esta aproximación con otros (realizar la validación del clúster).

En todo caso, será importante recordar que la silueta promedio casi siempre favorecerá modelos como los jerárquicos (como **k-means**, **PAM** y **CLARA**), que a modelos como el **DBSCAN** o el **GMM**. Esto se debe a que la aproximación de **DBSCAN** y **GMM** permiten crear conglomerados "largos y delgados" y hasta "envolver otros conglomerados"; es decir, por construcción no intentan optimizar las distancias intra e inter conglomerados, que es lo que mide la silueta. Y, por ejemplo, **k-means** sí intenta más o menos optimizar estas dos cantidades.

Regresando a la función **Mclust()**, esta solo necesita como argumento los datos estandarizados. Por ejemplo:

```
# Cargar el paquete
library(mclust)

res_GGM <- Mclust(datos_est)
```

Los resultados de la estimación del **GMM** por medio del algoritmo **EM** están en el objeto `res_GGM`. El número óptimo de clústeres lo podemos encontrar en el compartimiento **G**.

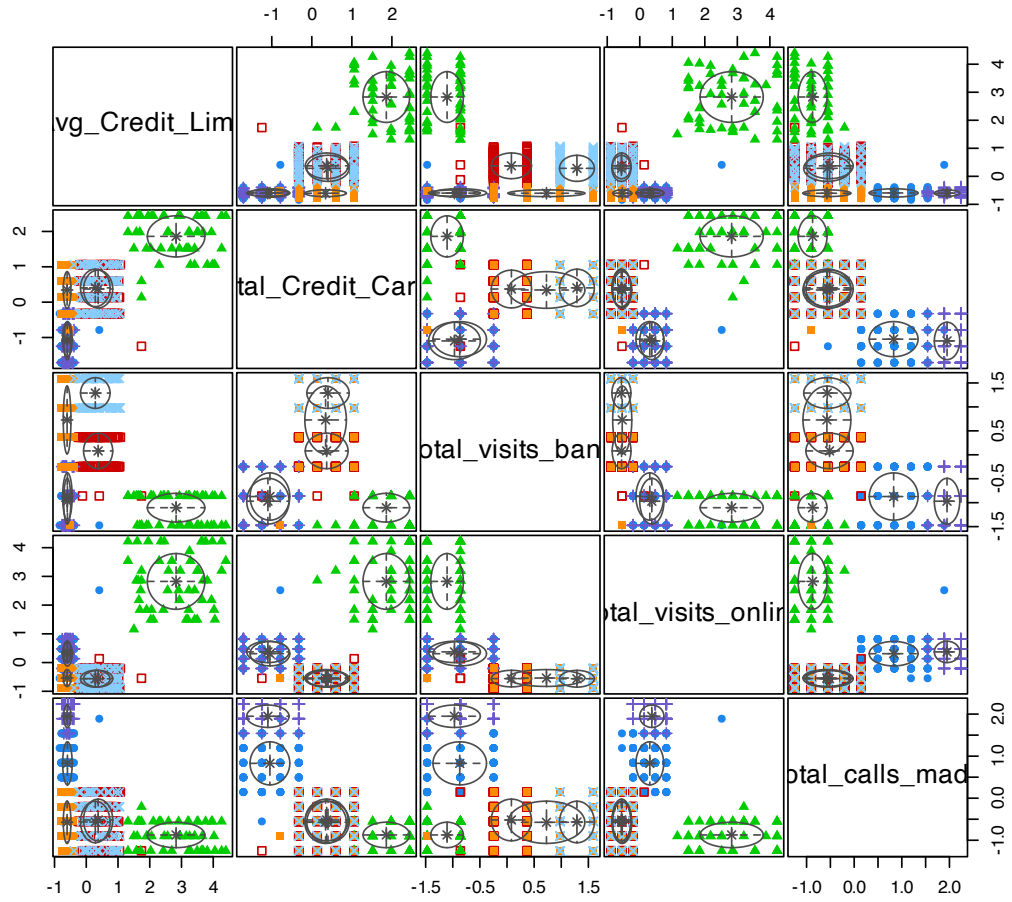
```
# Número de clústeres óptimo
res_GGM$G
```

```
## [1] 6
```

Así, según este algoritmo, el número óptimo de clústeres es 6. Así mismo, podemos visualizar los clústeres encontrados con el siguiente código (Ver Figura 8.2).

```
library(mclust)
plot(res_GGM, what = c("classification"))
```

Figura 8.2. Clústeres encontrados con el algoritmo GMM



Fuente: calculos propios

```
plot(res_GGM, what = c("density"))
```

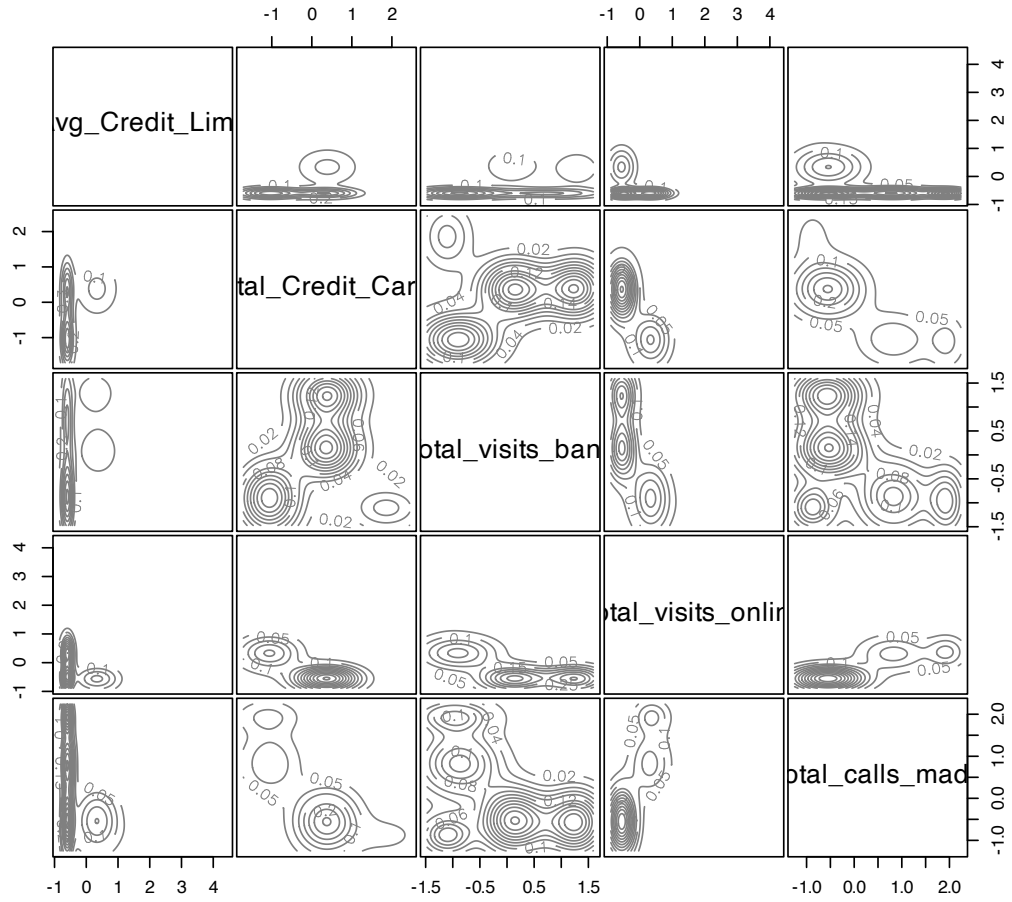
También podemos visualizar las distribuciones estimadas por medio del siguiente código (Ver Figura 8.3):

Y las membrecías de cada individuo se pueden encontrar en el compartimiento **classification** del objeto `res_GGM`.

```
# Membrecías
res_GGM$classification
```

Y la siluetas se pueden calcular, para efectos de comparación con otras aproxima-

Figura 8.3. Distribuciones de probabilidad por clúster encontradas con el algoritmo GMM



Fuente: calculos propios

ciones, con la función **silhouette()** del paquete *cluster* (Maechler et al., 2022).

```
# Calcular las siluetas individuales
sil_GMM <- silhouette(res_GGM$classification, dist(datos_est))
# Calcular la silueta promedio
mean(sil_GMM[, 3])
```

```
## [1] 0.2173319
```

Noten que la silueta promedio es relativamente baja (0.2173), como se esperaba (Ver Cuadro 8.1).

Tabla 8.1. Siluetas y número de clústeres óptimos para diferentes aproximaciones de clústering (distancia euclidiana)

Método de aglomeración	Silueta promedio	Número óptimo de clústeres
Enlace único	0.3758	3
Enlace completo	0.5703	2
Enlace promedio	0.5703	2
Enlace mediano	0.3022	6
Centroide	0.5703	2
Ward.D	0.5157	3
Ward.D2	0.5148	3
McQuitty	0.5703	2
k-means	0.5157	3
PAM	0.5158	3
CLARA	0.5139	3
DBSCAN	-0.0906	16
GMM	0.2173	6

Fuente: cálculos propios.

Ya hemos discutido cómo la silueta promedio no favorece los algoritmos que crean conglomerados "largos y delgados", y "envuelven otros conglomerados"; como por ejemplo, el algoritmo **DBSCAN** y el **GMM**. Estos resultados confirman que la silueta promedio no es una métrica adecuada para evaluar la calidad de los clústeres creados por estos algoritmos.

Existen otras métricas que son más adecuadas para evaluar la calidad de los grupos creados por **DBSCAN** y **GMM**, como por ejemplo, el índice de Dunn (Ver Sección 2.6.27), el índice de Calinski-Harabasz y el índice de Davies-Bouldin. Es importante elegir la métrica adecuada para evaluar la calidad de los conglomerados en función de la aplicación específica. Por ejemplo, si es importante que los clústeres sean compactos y estén bien separados, entonces el índice de Dunn sería una buena opción.

8.4 Otro ejemplo

Siguiendo la misma lógica del Capítulo 7, para mostrar las bondades de este algoritmo continuemos el ejemplo con datos del paquete *factoextra* (Kassambara y Mundt, 2020).

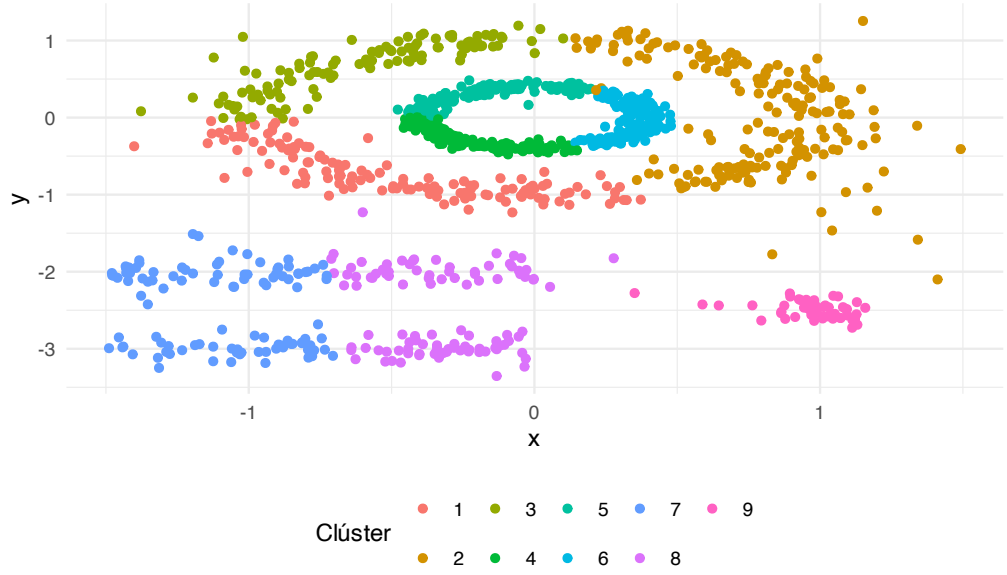
Empecemos por estimar el **GMM** empleando el objeto donde tenemos los datos (`multishapes`) (Ver Sección 7.4):

```
res_eje2_GGM <- Mclust(multishapes)
res_eje2_GGM$G
```

```
## [1] 9
```

En este caso encontramos 9 clústeres. En la Figura 8.4 se muestra cómo el algoritmo **GMM** particiona los datos.

Figura 8.4. Partición de los datos del ejemplo 2 empleando GMM



Fuente: elaboración propia.

Nota que los datos parecen estar mejor particionados con el algoritmo **GMM** que con el algoritmo **k-means** (Ver Sección 7.4). Es discutible si el **GMM** produce una mejor partición que el modelo **DBSCAN**.

Recordemos que, al emplear la silueta para realizar la comparación de los algoritmos, encontraremos algo que no cuadra con lo que estamos observando. En este caso, la silueta promedio para la partición, empleando **k-means**, es de 0.418. Para el caso del agrupamiento generado por **DBSCAN**, la silueta promedio es de 0.307 y para la agrupación que genera **GMM** es de 0.307 (no se presenta el código intencionalmente, realiza este cálculo). Estos resultados ratifican la importancia de escoger una

métrica adecuada para el problema de armar los conglomerados que enfrentemos.

8.5 Comentarios finales

En este capítulo estudiamos un método de clústering, que a diferencia de todos los estudiados hasta ahora, implica estimar el proceso generador de los datos. El modelo **GMM** asume que los datos son generados a partir de una mezcla de distribuciones normales multinomiales, para las cuales debemos estimar sus correspondientes medias, varianzas, covarianzas y el número de clústeres.

Como lo hemos venido mencionando, el **GMM** es otra de las herramientas que deberíamos tener en nuestra caja de herramientas para enfrentar las tareas de hacer clústeres. Y como toda herramienta, es importante reconocer en qué situación funcionará bien y en cuál no.

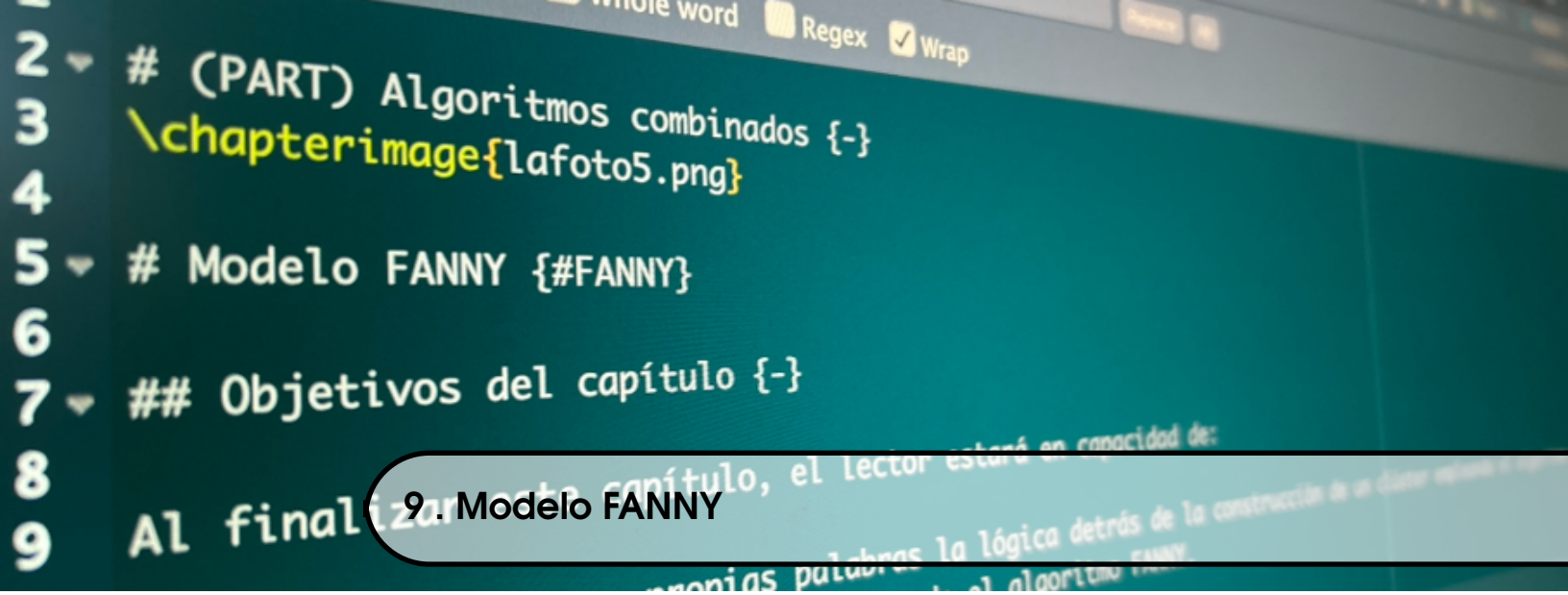
El **GMM** es especialmente útil cuando los datos en cada clúster se distribuyen aproximadamente de manera normal. Funciona bien en datos con clústeres de forma elipsoidales o con forma de "gota", pero puede tener dificultades con clústeres de formas irregulares. Además, el GMM asume que los datos están distribuidos de manera similar en todas las dimensiones, por lo que puede no ser adecuado para datos con varianzas muy diferentes en cada dimensión.

Los **GMM**, a diferencia de todos los modelos estudiados hasta ahora, generan asignaciones de clúster suave (*Soft Clustering*) para los individuos. Este algoritmo permite asignar probabilidades a que los individuos pertenezcan a cada uno de los conglomerados, en lugar de asignarlos de manera estricta a un clúster. Sin embargo, son sensibles a la inicialización y pueden ser computacionalmente costosos en conjuntos de datos grandes, debido al ajuste de múltiples distribuciones gaussianas.

En el siguiente capítulo estudiaremos otra herramienta para realizar la tarea de armar clústeres que sigue una filosofía totalmente diferente.

Parte VI

Algoritmos combinados



9. Modelo FANNY

Objetivos del capítulo

Al finalizar este capítulo, el lector estará en capacidad de:

- Explicar en sus propias palabras la lógica detrás de la construcción de un clúster empleando el algoritmo FANNY.
- Construir en R clústeres empleando el algoritmo FANNY.

9.1 Introducción

En los capítulos anteriores hemos estudiado diferentes filosofías para armar clústeres. Ya son siete métodos los que hemos estudiado:

- **Dos algoritmos jerárquicos**: **AGNES** y **DIANA**
- **Tres algoritmos particionados basados en centroides**: **k-means**, **PAM** y **CLARA**
- **Un algoritmo particionado basado en densidad**: **DBSCAN**
- **Un algoritmo particionado basado en distribución** o modelos: **GMM**

En este capítulo estudiaremos un método de clústering que es una combinación de filosofías: **FANNY** (por el termino en inglés *fuzzy*[1] *analysis*).

Los modelos *fuzzy* de clústering son una extensión de los métodos de clústering tradicionales, que permiten que los individuos pertenezcan a múltiples grupos con diferentes grados de membrecía. A diferencia de los métodos de clústering convencionales, FANNY genera asignaciones de clúster suave (*Soft Clustering*) como lo hace el **GMM**.

Los modelos **fuzzy** de clústering se basan en la teoría de conjuntos difusos propuesta por Zadeh (1965), donde un elemento puede pertenecer a un conjunto con diferentes grados de pertenencia. En el contexto del clústering, esto significa que un punto de datos puede pertenecer a varios conglomerados simultáneamente, cada uno con un grado de pertenencia específico.

FANNY modifica los algoritmos basados en centroides incluyendo la posibilidad de un clústering suave. Es decir, en **FANNY** cada clúster se caracteriza por un centroide y cada individuo tiene un grado de pertenencia a cada clúster. En **FANNY** los individuos cercanos al centro de un grupo tienen un mayor grado de membrecía que aquellos en el borde de un clúster.

FANNY se emplea en una gran gama de áreas que van, desde la segmentación de imágenes en la que se agrupan píxeles con características similares, hasta las finanzas donde se emplean para agrupar activos financieros con características similares; pasando por aplicaciones tan variadas como en la medicina para agrupar pacientes con enfermedades similares y en la ingeniería para agrupar piezas o componentes con características similares.

9.2 El algoritmo FANNY

Uno de los algoritmos más comunes para el **FANNY** es el *Fuzzy C-Means* (Kaufman y Rousseeuw, 2009) al cual nos seguiremos refiriendo como **FANNY**. Este algoritmo asigna

[1] La traducción del termino *fuzzy* es difuso. En este contexto el término se refiere al análisis difuso o lógica difusa. La lógica difusa es un enfoque que permite manejar la incertidumbre y la imprecisión en los procesos de toma de decisiones. Se basa en el concepto de que las cosas pueden ser verdaderas solo en cierto grado, en lugar de simplemente verdaderas o falsas. Este enfoque ha encontrado aplicaciones en campos como la inteligencia artificial, el control de sistemas y la toma de decisiones en situaciones complejas donde la precisión absoluta puede ser difícil de lograr. Y más recientemente en los modelos de inteligencia artificial generativa como los Large Lenguage Models (LLM) detrás de herramientas como ChatGPT. Esta idea fue inicialmente propuesta por Zadeh (1965).

a cada individuo un vector de pertenencia, donde cada elemento del vector representa la membrecía del punto a un grupo específico. El objetivo de **FANNY** es minimizar la distancia entre los puntos y los centroides de los clústeres ponderados por las membrecías.

Como lo hecho anteriormente, veamos los pasos del algoritmo **FANNY** para entender cómo funciona. Para un número de clústeres previamente establecido (k), los pasos de **FANNY** son los siguientes:

1. Seleccionar aleatoriamente k centroides iniciales de los clústeres y asignar a cada individuo una membresía inicial.
2. Calcular las nuevas membrecías de los individuos basadas en la distancia a los centroides y los grados de pertenencia actuales.
3. Calcular los nuevos centroides de los clústeres basados en las membrecías de los individuos.
4. Repetir los pasos 2 y 3 hasta que se alcance un criterio de convergencia, como la estabilidad de los centroides o un número máximo de iteraciones.

En la siguiente sección estudiaremos cómo implementar este algoritmo en R.

9.3 Implementación de FANNY en R

FANNY se puede implementar en R empleando el paquete *cluster* (Maechler et al., 2022) con la función **fanny()** . El parámetro fundamental para implementar el algoritmo **FANNY** es el número de conglomerados (k). Para encontrar el número de clústeres óptimo (q) podemos emplear las estrategias discutidas en la Sección 2.4. Si queremos emplear la batería de 30 índices podemos emplear la función **n_clusters()** del paquete *parameters* (Lüdecke et al., 2020).

Como lo discutimos en el Capítulo 6, podríamos emplear las funciones **n_clusters_elbow()** y **n_clusters_gap()** si se desea solo emplear el método del codo o el índice GAP (Ver Sección 2.4 para una discusión de estos métodos), respectivamente.

Como lo hemos realizado a lo largo del libro[2], empleemos la función **n_clusters_silhouette()** para encontrar el número óptimo de clústeres empleando la silueta promedio.

Procedamos a emplear esta "infraestructura" que nos brinda el paquete *parameters* para implementar **FANNY** en los datos que hemos venido trabajando en los capítulos anteriores. Carga el espacio de trabajo que guardaste en el Capítulo 8.

El código para encontrar el número óptimo de conglomerados empleando los datos estandarizados (`datos_est`), la distancia euclidiana (**distance_method = "euclidean"**) y **FANNY** (**clustering_function= cluster::fanny**) es:

```
# Fijar la semilla aleatoria
set.seed(12345)
```

[2]La excepción son los algoritmos **DBSCAN** y **GMM** en los cuáles no es necesario fijar k.

```
res_FANNY_Sil <- n_clusters_silhouette(datos_est, standardize = FALSE,
↪  distance_method = "euclidean",
    clustering_function = cluster::fanny, n_max = 20)
```

Puedes ver rápidamente que el número de clústeres óptimo en este caso es 7. La silueta promedio corresponde a 0.5159. Silueta que no es superior a la alcanzada con otros métodos ya evaluados. En el Cuadro 9.1 se reportan el número de clústeres y la silueta promedio encontrada para cada método estudiado.

Tabla 9.1. Siluetas y número de clústeres óptimos para diferentes aproximaciones de clústering (distancia euclidiana)

Método de aglomeración	Silueta promedio	Número óptimo de clústeres
Enlace único	0.3758	3
Enlace completo	0.5703	2
Enlace promedio	0.5703	2
Enlace mediano	0.3022	6
Centroide	0.5703	2
Ward.D	0.5157	3
Ward.D2	0.5148	3
McQuitty	0.5703	2
k-means	0.5157	3
PAM	0.5158	3
CLARA	0.5139	3
DBSCAN	-0.0906	16
GMM	0.2173	6
FANNY	0.5159	7

Fuente: cálculos propios.

Si bien en este caso **FANNY** no parece una buena aproximación para particionar los datos, veamos cómo se puede implementar el algoritmo para el número óptimo de clústeres encontrado.

En general, después de conocer el número óptimo de clústeres, podemos aplicar el respectivo algoritmo para continuar con nuestro análisis similar a lo que presentamos en las Secciones 4.7 y 5.4.3. Es importante recordar que esto solo se haría si esta aproximación fuese la mejor forma de particionar los datos.

Para implementar el algoritmo **FANNY**, la función que usaremos es **fanny()** del paquete *cluster* (Maechler et al., 2022). Esta función requiere en su forma más simple, un objeto de clase **data.frame** con los datos (**x**), el número de clústeres que queremos formar (**k**) y la medida de distancia (**metric**). Es decir, el código será:

```
# Fijar la semilla aleatoria
set.seed(12345)
```

```
# calculamos los clústeres para k = 7
cluster_FANNY <- fanny(x = datos_est, k = 7, stand = FALSE, metric =
↪  "euclidean")
# Mirar todos los comportamientos del objeto creado
attributes(cluster_FANNY)
```

```
## $names
##  [1] "membership"  "coeff"       "memb.exp"    "clustering"  "k.crisp"
##  [6] "objective"   "convergence" "diss"        "call"        "silinfo"
## [11] "data"
##
## $class
## [1] "fanny"     "partition"
```

El objeto que creamos (`cluster_FANNY`) tiene varios compartimientos (*slots*) entre los cuales podemos destacar **clustering** que contiene las membrecías de cada uno de los clientes (la que tiene la mayor probabilidad) y en el *slot* **membership** se encontrará la probabilidad de que cada individuo (fila) pertenezca a cada uno de los clústeres (columna). Por ejemplo, las membrecías de todos los clientes las podemos extraer fácilmente de la siguiente manera:

```
# Calcular membrecias
cluster_FANNY$clustering
```

9.4 Comentarios finales

En este capítulo estudiamos el modelo **FANNY** que es una extensión de los métodos tradicionales de clústering particionados basados en centroides. **FANNY** permite una representación más flexible de la estructura de los datos al permitir un clústering suave. Esto permite una mayor flexibilidad en la representación de la estructura de los datos.

FANNY también se caracteriza por tener una robustez mayor frente al ruido en los datos que los métodos basados en centroides.

Por otro lado, similar a **k-means** y **PAM**, **FANNY** es sensible a la inicialización de los centroides y a la elección del número de clústeres. Este método implica una mayor complejidad computacional debido a la actualización de las membrecías y los centroides en cada iteración.

Con el estudio de **FANNY** terminamos la introducción a los principales modelos de clústering disponibles. La lista de modelos disponibles no para allí. Existen muchos más modelos, como por ejemplo **hk-means** (*Hierarchical k-means*), **HDBSCAN** (*Hierarchical* **DBSCAN**), clústering de afinidad y propagación (*Affinity Propagation Clustering*), **SOM** (*Self Organizing Map*) y muchos más.

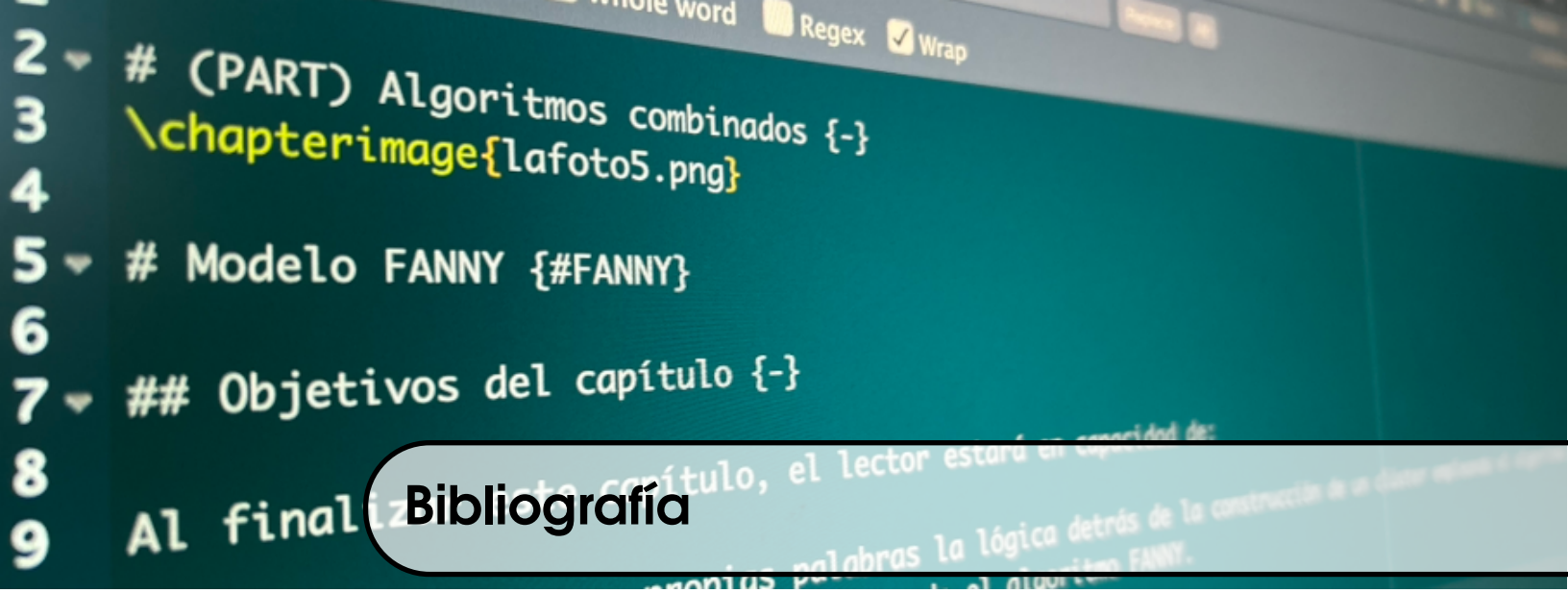
También existen técnicas para combinar los resultados de diferentes metodologías llamado clústering por consenso (*Consensus clustering* en inglés). Esta es una técnica empleada para combinar los resultados de múltiples algoritmos de agrupamiento o

incluso múltiples ejecuciones del mismo algoritmo para crear una solución de agrupamiento única y más consistente.

La investigación en el campo de métodos de clústering es grande y muy dinámica. La diversidad de enfoques y algoritmos utilizados en el clústering refleja la complejidad y amplitud de este campo de estudio. No obstante, las bases para los desarrollos actuales en esta área son los modelos que hemos estudiado en este libro. Esperamos que este libro te permita iniciar en el estudio de esta área y empezar a llenar tu caja de herramientas para realizar esta tarea del *Business Analytics*. Y recuerda, "**en el mundo del *Business Analytics* la imaginación es el límite**".

Parte VII

Referencias



Bibliografía

Alonso, J. C. (2002). A new accelerator for the em algorithm. Master's thesis, Iowa State University.

Alonso, J. C. (2022). *Empezando a transformar bases de datos con R y dplyr*. Universidad Icesi.

Alonso, J. C. (2024). *Introducción al Modelo Clásico de Regresión para Científico de Datos en R*. Universidad Icesi.

Alonso, J. C. y Arboleda, A. M. (2025). *Introducción al Análisis de Canastas de Compra para analytics translators y científicos de datos (empleando R)*. Universidad Icesi.

Alonso, J. C. y Hoyos, C. C. (2025a). *Una introducción a los modelos de Clasificación empleando R.* Universidad Icesi.

Alonso, J. C. y Hoyos, C. C. (2025b). *Una introducción a los modelos de clasificación empleando R*. Universidad Icesi.

Alonso, J. C. y Largo, M. F. (2023). *Empezando a visualizar datos con R y ggplot2.* Universidad Icesi, 2. edition.

Alonso, J. C. y Ocampo, M. P. (2022). *Empezando a usaR: Una guía paso a paso.* Universidad Icesi.

Alonso C., J. C. (2020). Herramientas del Business Analitycs en R : Análisis de Componentes Principales para resumir variables. *Economics Lecture Notes*, (10):1–32.

Arthur, D., Vassilvitskii, S., et al. (2007). k-means++: The advantages of careful seeding. In *Soda*, volume 7, pages 1027–1035.

Baker, F. B. y Hubert, L. J. (1976). A graph-theoretic approach to goodness-of-fit in complete-link hierarchical clustering. *Journal of the American Statistical Association*, 71(356):870–878.

Ball, G. H. y Hall, D. J. (1965). Isodata, a novel method of data analysis and pattern classification. Technical report, Stanford research inst Menlo Park CA.

Beale, E. (1969). *Euclidean cluster analysis*. Scientific Control Systems Limited.

Bezdek, J. C. y Pal, N. R. (1998). Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* (*Cybernetics*), 28(3):301–315.

Brock, G., Pihur, V., Datta, S., y Datta, S. (2008). clValid: An R package for cluster validation. *Journal of Statistical Software*, 25(4):1–22.

Caliński, T. y Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27.

Charrad, M., Ghazzali, N., Boiteau, V., y Niknafs, A. (2014). NbClust: An R package for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, 61(6):1–36.

Chen, Y., Ruys, W., y Biros, G. (2020). Knn-dbscan: a dbscan in high dimensions. *arXiv preprint arXiv:2009.04552*.

Davies, D. L. y Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227.

de Vries, A. y Ripley, B. D. (2024). *ggdendro: Create Dendrograms and Tree Diagrams Using 'ggplot2'*. R package version 0.2.0.

Dimitriadou, E., Dolničar, S., y Weingessel, A. (2002). An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1):137–159.

Duda, R. O., Hart, P. E., et al. (1973). *Pattern classification and scene analysis*, volume 3. Wiley New York.

Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104.

Edwards, A. W. y Cavalli-Sforza, L. L. (1965). A method for cluster analysis. *Biometrics*, pages 362–375.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Fox, J. y Weisberg, S. (2019). *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, third edition.

Frey, T. y Van Groenewoud, H. (1972). A cluster analysis of the d2 matrix of white spruce stands in saskatchewan based on the maximum-minimum principle. *The Journal of Ecology*, pages 873–886.

Friedman, H. P. y Rubin, J. (1967). On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62(320):1159–1178.

Fukunaga, K. y Koontz, W. L. (1970). A criterion and an algorithm for grouping data. *IEEE Transactions on Computers*, 100(10):917–923.

Godichon-Baggioni, A. y Surendran, S. (2023). *Kmedians: K-Medians*. R package version 2.2.0.

Gordon, A. (1999). Cluster description. *University of St. Andrews Scotland*.

Hahsler, M., Piekenbrock, M., y Doran, D. (2019). dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, 91(1):1–30.

Haldiki, M., Batistakis, Y., y Vazirgiannis, M. (2002). Cluster validity methods. In *SIGMOD*, volume 31, pages 40–45.

Halkidi, M. y Vazirgiannis, M. (2001). Clustering validity assessment: Finding the optimal partitioning of a data set. In *Proceedings 2001 IEEE international conference on data mining*, pages 187–194. IEEE.

Hartigan, J. A. (1975). *Clustering algorithms*. John Wiley & Sons, Inc.

Hill, R. S. (1980). A stopping rule for partitioning dendrograms. *Botanical Gazette*, 141(3):321–324.

Hubert, L. y Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.

Hubert, L. J. y Levin, J. R. (1976). A general statistical framework for assessing categorical clustering in free recall. *Psychological bulletin*, 83(6):1072.

Kassambara, A. (2017). *Practical guide to cluster analysis in R: Unsupervised machine learning*, volume 1. Sthda.

Kassambara, A. y Mundt, F. (2020). *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. R package version 1.0.7.

Kaufman, L. y Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.

Kraemer, H. C. (2004). Biserial correlation. *Encyclopedia of statistical sciences*, 1.

Lance, G. N. y Williams, W. T. (1967). Mixed-data classificatory programs i - agglomerative systems. *Australian Computer Journal*, 1(1):15–20.

Lebart, L., Morineau, A., y Piron, M. (1995). *Statistique exploratoire multidimensionnelle*, volume 3. Dunod Paris.

Lüdecke, D., Ben-Shachar, M. S., Patil, I., y Makowski, D. (2020). Extracting, computing and exploring the parameters of statistical models using R. *Journal of Open Source Software*, 5(53):2445.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., y Hornik, K. (2022). *cluster: Cluster Analysis Basics and Extensions*. R package version 2.1.4 — For new features, see the 'Changelog' file (in the package source).

Marriott, F. (1971). Practical problems in a method of cluster analysis. *Biometrics*, pages 501–514.

McClain, J. O. y Rao, V. R. (1975). Clustisz: A program to test for the quality of clustering of a set of objects. *Journal of Marketing Research*, pages 456–460.

Milligan, G. W. (1980). An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *psychometrika*, 45(3):325–342.

Milligan, G. W. (1981). A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46(2):187–199.

Milligan, G. W. y Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179.

Murtagh, F. y Legendre, P. (2014). Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion? *Journal of classification*, 31(3):274–295.

Orlóci, L. (1967). An agglomerative method for classification of plant communities. *The Journal of Ecology*, pages 193–206.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ratkowsky, D. y Lance, G. (1978). Criterion for determining the number of groups in a classification.

Rdusseeun, L. y Kaufman, P. (1987). Clustering by means of medoids. In *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*, volume 31.

Rohlf, F. J. (1974). Methods of comparing classifications. *Annual Review of Ecology and Systematics*, 5(1):101–113.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Schloerke, B., Cook, D., Larmarange, J., Briatte, F., Marbach, M., Thoen, E., Elberg, A., y Crowley, J. (2023). *GGally: Extension to 'ggplot2'*. R package version 2.2.0.

Scott, A. J. y Symons, M. J. (1971). Clustering methods based on likelihood ratio criteria. *Biometrics*, pages 387–397.

Scrucca, L., Fraley, C., Murphy, T. B., y Raftery, A. E. (2023). *Model-Based Clustering, Classification, and Density Estimation Using mclust in R*. Chapman and Hall/CRC.

Shah, A. (2021). Credit card customer data. Technical report.

Tibshirani, R., Walther, G., y Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423.

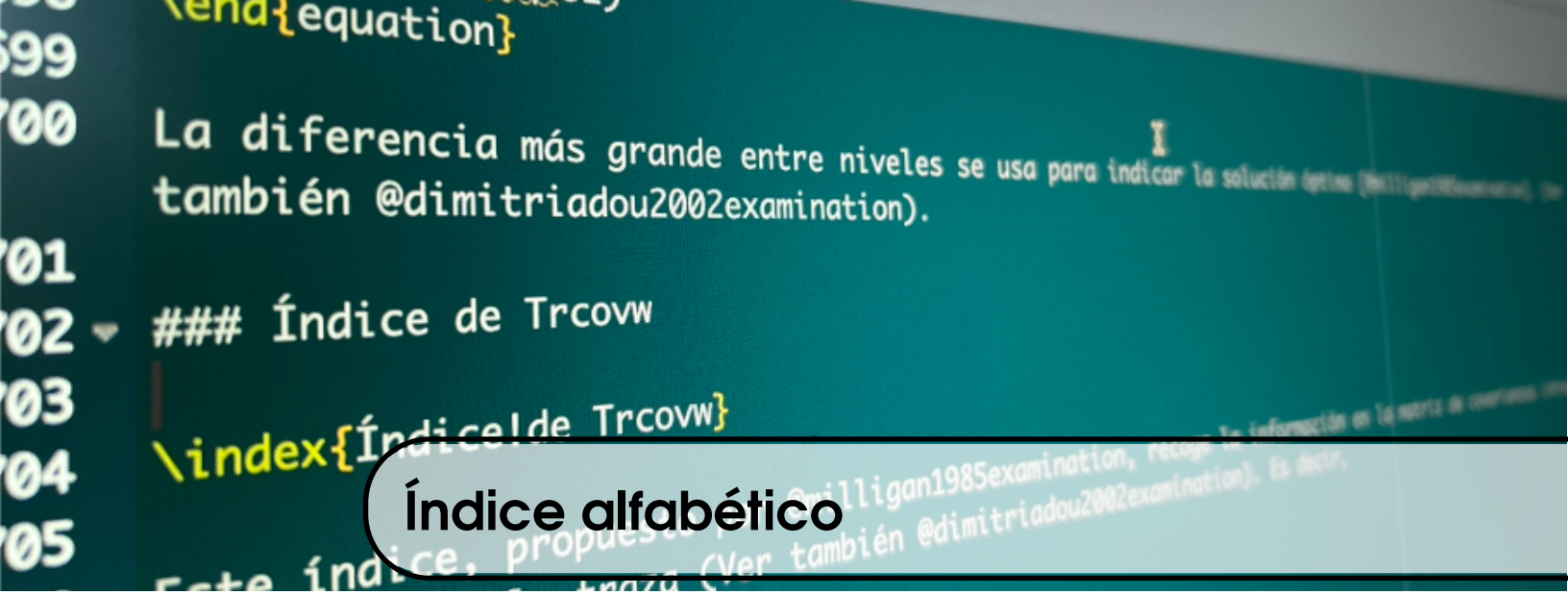
Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H., François, R., Henry, L., y Müller, K. (2022). *dplyr: A Grammar of Data Manipulation*. R package version 1.0.8.

You, K. (2023). *maotai: Tools for Matrix Algebra, Optimization and Inference*. R package version 0.2.5.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.



Índice alfabético

AGNES, 60
Algoritmo
    basados en centroides, 33
    basados en densidad, 35
    basados en distribución, 35
    basados en modelos, 35
    bottom-up, 60
    CLARA, 33
    DBSCAN, 35
    EM, 166
    Gaussian Mixed Models, 35
    GMM, 35
    k-means, 33
    k-means++, 127
    k-medians, 132
    k-medoids, 33
    PAM, 33
    top-down, 60
Algortimo
    CLARA, 133
    DBSCAN, 149
    k-means, 116
    PAM, 133
Analítica
    descriptiva, 21
    diagnóstica, 21
    predictiva, 21, 23
    prescriptiva, 21, 23
Aprendizaje no supervisado, 21
Aprendizaje supervisado, 21

Border point, 149
Business Analitycs, 18

Centroide, 38, 69, 75, 92
científico de datos, 13
CLARA, 133
Clases, 18
clúster
    particionado, 25
Clústeres, 18
    jerárquicos, 32, 60
    particionados, 32
Clústering, 18, 26
    jerárquico aglomerativo, 60
    jerárquico de división, 60
Coeficiente
    de asociación simple, 28
    de Jacard, 28
    Hannann, 28
    Regrers y Tanimoto, 28
    Sokal y Sneath, 28
    Sorensen o Dice, 28
Coherencia, 26
Complete linkage, 69
Concordancia, 46
Conglomerados, 18
Core point, 147, 153, 154
Correlación

de Pearson, 28

DBCSAN, 149
DBSCAN, 35
Dendrograma, 62, 64, 67, 85, 95
DIANA, 60, 72
Disconcordancia, 46
Distancia
binaria, 31
de Canberra, 31
de Chebyshev, 29
de Manhattan, 29
de Minkowski, 31, 83
euclidiana, 28, 75, 76, 121
interclúster, 26, 41, 46, 49
intraclúster, 26, 46, 49
Mahalanobis, 28, 75, 76
Manhattan, 28, 75, 76
Minkowski, 28, 75, 76
máxima, 29
promedio intraclúster, 40
Diámetro, 72

eps, 147
Estadístico
de Hubert, 53
Estandarizar, 31

Fenómeno de encadenamiento, 69
Función
aov(), 108, 109
as.phylo(), 101
bartlett.test(), 109
clara(), 139
cutree(), 91, 97
dbscan(), 152
dianay(), 94
dist(), 83, 135, 153, 154
fanny(), 177, 178
fligner.test(), 109
fviz_cluster(), 106
fviz_dend(), 95, 105
fviz_nbclust(), 88, 122
ggdendrogram(), 103
hclust(), 84, 91
kmeans(), 125
kmeanspp(), 129
kruskal.test(), 111
leveneTest(), 109
library(), 132
Mclust(), 167
n_clusters(), 134, 177
n_clusters_dbscan(), 152
n_clusters_silhouette(), 135, 152, 177
NbClust(), 87, 91, 97, 121, 122, 129, 134
nueva_fviz_nbclust, 89
nueva_fviz_nbclust(), 122
pam(), 134, 138
plot(), 85
plot.phylo(), 101
scale(), 82
set.seed(), 121
shapiro.test(), 110
silhouette(), 91, 125, 139, 170
TukeyHSD(), 112

Gráfico
de siluetas, 91

HAC, 60
HCA, 77
Heurístico, 67
Hierarchical Cluster Analysis, 77

k-means, 116
k-means++, 127
k-medians, 132

maldición de la dimensionalidad, 29
Matriz
de proximidad, 62, 83, 88
Medidas de similitud, 26
Medoide, 132
Membrecía, 97, 127
minPts, 147
Muestra de corte transveral, 20
Muestra de panel, 20
Máxima Verosimilitud, 166
Método
del Codo, 38
Método Aglomeración
enlace completo, 69, 75, 92
enlace mediano, 69, 75, 92

enlace promedio, 69, 75, 85, 92
enlace único, 69, 75, 92
McQuitty, 71, 75, 76, 92
Ward, 75
Ward.D, 71, 76, 92
Ward.D2, 71, 76, 92

Noise point, 149

outlier, 149

PAM, 133
Paquete
ape, 101
car, 109
cluster, 91, 94, 120, 134, 138, 139, 170, 177, 178
clValid, 134
dbscan, 152, 154
dplyr, 132
factoextra, 88, 95, 105, 120, 122, 155, 171
GGally, 81
ggdendro, 86, 103
ggplot2, 81, 86, 88, 103, 105
maotai, 129
mclust, 167
NbClust, 87, 120, 121, 129, 134
parameters, 134, 135, 152, 177
PCA, 23
Punto central, 147, 153, 154
Punto de frontera, 149

Ruido, 149
Ruido en los datos, 147

Segmentación de clientes, 18
Separación, 26
Silueta promedio, 91
Siluetas, 40, 91
Single linkage, 69

Tareas en la analítica, 18

Variable
Definición, 26

Índice
C, 46
CCC), 47
Criterio Cúbico, 47
D, 52
DB, 48
de Ball, 50
de Beale, 47
de Calinski y Harabasz (CH), 45
de Clústering, 47
de Dunn, 53
de Frey, 49
de Friedman, 51
de Gap, 39, 52
de Hartigan, 49, 52
de KL, 52
de Marriott, 50
de McClain, 51
de Ratkowsky, 49
de Rubin, 52
de Scott, 50
de Silueta, 52
de Tracew, 51
de Trcovw, 51
Duda, 45
Gamma, 46
Gplus, 48
Pseudo t^2, 46
Pt-biserial, 48
SD, 54
SDbw, 54
Tau, 49

Universidad Icesi | Editorial Universidad Icesi

Este libro es una introducción clara y accesible a los modelos estadísticos y de aprendizaje de máquina aplicados al *clustering*. Dirigido a quienes inician su formación como científicos de datos, ofrece una guía esencial para entender cómo identificar y agrupar elementos similares, manteniendo la mayor diferencia posible entre los grupos, también llamados conglomerados, clases o clústeres. Estas técnicas no solo pueden responder preguntas de negocio por sí solas, sino que también son clave en la exploración de datos antes de desarrollar modelos complejos y probar hipótesis. A lo largo del libro, descubrirás cómo construir y analizar estos grupos con aplicaciones prácticas y enfoques fundamentales.

Herramientas del BIG DATA y analytics