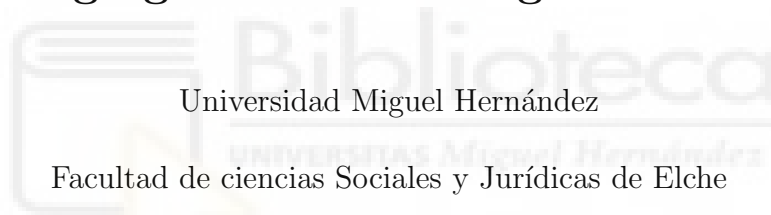




Obtención de un ranking parcial a partir de la agregación de rankings totales



Universidad Miguel Hernández

Facultad de ciencias Sociales y Jurídicas de Elche

Grado en Estadística Empresarial

Trabajo Fin de Grado

Autor: Jesús Carrión Salinas

Tutor: Mercedes Landete Ruiz

Índice general

Resumen	1
1. Preliminares	2
2. Introducción y objetivos	5
3. Agregación de rankings en cubos	7
3.1. Definición de problema	7
3.2. Ejemplo de prueba	8
3.3. El algoritmo GAP	12
3.3.1. Introducción al GAP	12
3.3.2. Etapa 1	13
3.3.3. Etapa 2	14
3.3.4. Etapa 3	20
3.3.5. Evaluación final	22
4. Aplicación a los Caballos del Vino	23
4.1. Los Caballos del Vino	23
4.1.1. Historia de los Caballos del Vino	23
4.1.2. La fiesta de los Caballos del Vino	24
4.2. Algoritmo de la agregación de cubos para los Caballos del Vino	25
4.2.1. Etapa 1	27
4.2.2. Etapa 2	28
4.2.3. Etapa 3	32
4.2.4. Nueva Etapa 3	37
5. Conclusiones	46

Resumen

El objetivo de este trabajo es doble. Por un lado, introducir una heurística que permita realizar agregaciones de rankings. Por otro lado, la aplicación de dicha heurística a un problema real, así como la programación en el lenguaje de programación R.

El trabajo comienza con una introducción a las técnicas de agrupación de datos, describiendo algunos tipos de algoritmos.

Tras describir brevemente el problema de agregación de rankings, introduciremos una heurística que permite resolver este tipo problema.

Para finalizar aplicaremos la heurística propuesta anteriormente en una base de datos real. En concreto, aplicaremos esta heurística a los caballos del vino.

La siguiente memoria se ha elaborado en \LaTeX , mediante RSweave. RSweave es un componente del lenguaje de programación R que permite la integración de código en documentos escritos en \LaTeX .

Capítulo 1

Preliminares

La minería de datos o exploración de datos es un campo de la estadística y las ciencias de la computación referido al proceso que trata de descubrir patrones en grandes volúmenes de conjunto de datos. Su objetivo general consiste en extraer información de un conjunto de datos y transformarla en una estructura comprensible para su uso posterior. La tarea de minería de datos es el análisis automático o semi-automático de grandes cantidades de datos para extraer patrones interesantes hasta ahora desconocidos, como los grupos de registro de datos (análisis clúster), registros poco usuales (la detección de anomalías) y dependencias (minería por reglas de asociación).

En este proyecto trataremos de abordar la forma de aplicar minería de datos cuando nuestro conjunto de datos son rankings. Nuestro objetivo es introducir una técnica que nos permita pasar de muchos rankings totales a un único ranking parcial. Imaginemos que tenemos n objetos y clasificamos estos n objetos de mejor a peor m veces, con esto obtendríamos m rankings totales. La idea es agrupar estos m rankings en grupos de mejores y peores clasificados.

La idea de este proyecto surge de la dificultad que conlleva cuando tenemos m clasificaciones de n objetos diferentes conocer cuales son los objetos mejor y peor clasificados.

K-medias

K-medias es un algoritmo de clasificación no supervisada que agrupa n objetos en k grupos. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o cluster. Normalmente, se suele usar la distancia cuadrática.

El algoritmo k-means resuelve un problema de optimización, siendo la función a optimizar, minimizar la suma de las distancias cuadráticas de cada objeto al centroide de su cluster.

Formalmente, dado un conjunto de observaciones (x_1, x_2, \dots, x_n) , donde cada observación es un vector real de d dimensiones. El algoritmo k-medias construye k grupos donde se minimiza la suma de distancias de los objetos de cada grupo $S = \{S_1, S_2, \dots, S_k\}$ a su centroide.

$$\min \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \quad (1.1)$$

donde μ_i es la media de los objetos (centroide) del grupo S_i .

El algoritmo consta de tres pasos:

1. **Inicialización:** una vez escogido el número de grupos k , se establecen k centroides. Por ejemplo, escogiéndolos aleatoriamente.
2. **Asignación de objetos a los centroides:** cada objeto de los datos es asignado a su centroide más cercano.

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\| \leq \|x_p - m_j^{(t)}\| \forall 1 \leq j \leq k\} \quad (1.2)$$

Donde cada x_p va exactamente dentro de un $S_i^{(t)}$, incluso aunque pudiera ir en dos de ellos.

3. **Actualización centroides:** se actualiza la posición del centroide de cada grupo, tomando como nuevo centroide la posición media de los objetos pertenecientes a dicho grupo.

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \quad (1.3)$$

Se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia determinada.

Las principales ventajas del método k-medias son que es un método sencillo y rápido. Las desventajas son que es necesario decidir el número de grupos k previamente y, como se trata de un algoritmo heurístico, no hay garantía de que converja a un óptimo global, puesto que el resultado final puede depender de la inicialización de los centroides.



Capítulo 2

Introducción y objetivos

Descubrir una agrupación ordenada de objetos B a partir de un conjunto de rankings totales es un problema fundamental que tiene muchas aplicaciones. Informalmente, un ranking parcial es una clasificación que permite agrupar objetos en grupos.

Dada una población $\{1, 2, \dots, n\}$, partimos de un conjunto de rankings totales $\{T_1, T_2, \dots, T_m\}$. Por ejemplo, $T_i = (2, 3, 1)$ para $n = 3$ significa que el individuo i opina que el mejor es el 2, el siguiente es el 3 y el último es el 1. Decimos que es total porque en $T_1 \dots T_m$ no hay empates. Es decir, cada objeto ocupa una posición del ranking y dos objetos no pueden ocupar la misma posición. Un ranking en cubos B es una ordenación de cubos, siendo los cubos grupos. Por ejemplo, si $n = 7$ y decimos que $B = \langle \{1, 3, 5\}, \{2, 4\}, \{6, 7\} \rangle$, significa que los objetos 1, 3 y 5 pertenecen al mismo grupo y están clasificados por delante de los otros dos grupos. De la misma forma, los objetos 2 y 4 pertenecen al mismo grupo y están clasificados por detrás de 1, 3 y 5 y por delante de 6 y 7. Los elementos 6 y 7 son los peor clasificados. Todos los objetos de un mismo grupo están empatados.

Existen algoritmos que tratan de realizar rankings parciales maximizando la similitud intra-grupo a la hora de realizar agrupaciones. Es decir, buscan hacer grupos en los que los objetos de un mismo grupo sean muy parecidos entre sí. La principal forma de maximizar la similitud intra-grupo, es minimizar la suma de las desviaciones de los rangos medios dentro de un grupo. Por tanto, objetos en un grupo grande b están forzados a tener diferentes rangos medios y, como resultado, b tendrá una suma de desviaciones alta. En consecuencia, minimizar el sumatorio de desviaciones

puede llevar a que la mayoría de los grupos grandes sean descompuestos en otros más pequeños.

En este proyecto proponemos una heurística que utiliza la disimilitud inter-grupo para una mejor agrupación. Esta heurística propone usar la “cercanía” en diferentes rangos cuantílicos para determinar si dos objetos deben ser puestos en el mismo grupo. Para esto usaremos una medida que se llama *Diferencia Anormal entre Rangos*, la cual nos dice si la diferencia que existe entre dos rangos es considerada grande. Como nuestra idea es realizar grupos en los cuales los objetos de diferentes grupos sean muy distintos entre sí, usaremos esta medida para determinar si dos objetos deben ser puestos en distinto grupo.

Parte de los conceptos que se estudian en esta memoria están relacionados con el contenido de la asignatura de Minería de Datos. Además, algunas habilidades que se han adquirido mediante la realización de esta memoria han sido: el uso de RSweave, escritura de textos en L^AT_EX y programación avanzada de R.

El capítulo 3 y las secciones 4.2.1, 4.2.2 y 4.2.3 del capítulo 4, son el resultado del estudio del problema de ordenación de cubos en la literatura. La sección 4.2.4 de este último capítulo, es nuestra aportación a la resolución de este tipo de problemas. De esta manera, la realización de esta memoria nos ha acercado al estudio de problemas en la literatura y a la investigación.

El objetivo de esta memoria es proponer una vía para resolver el problema de agregación de rankings y, al mismo tiempo, realizar un algoritmo que nos permita hacerlo de forma rápida y sencilla.

Capítulo 3

Agregación de rankings en cubos

La heurística desarrollada en este capítulo se presenta en los artículos [1] y [2].

3.1. Definición de problema

Formalmente, dado un conjunto I de n objetos, un orden de cubos B en el conjunto I es la clasificación total T definida sobre r grupos B_1, \dots, B_r , los cuales son agrupaciones sobre I .

Dado dos objetos t y u en I , no hay relación de precedencia entre t y u si están en el mismo grupo y se dice que están “empataados”. Si el objeto t pertenece al grupo B_p y el objeto u pertenece a B_q , decimos que t precede a u si y solo si B_p precede a B_q acorde con el orden total T . Un orden total en I puede ser visto como un caso especial de un orden de cubos tal que cada grupo contiene solo un objeto.

Dado un conjunto m de clasificaciones totales $\{T_1, T_2, \dots, T_m\}$ definidas en I , calculamos una matriz de precedencia a partir de las clasificaciones totales dadas. El valor de C_{tu} es definido como la fracción de las clasificaciones totales en las cuales el objeto t es clasificado por delante del objeto u . Llamamos C_{tu} a la probabilidad de precedencia de t con respecto de u . De forma similar, podemos usar la matriz de precedencia C^B para representar un orden de cubos B específico. A la entrada C_{tu}^B se le asigna un 0.5 si t y u están en el mismo grupo en B . Si t y u están en distintos grupos en B , y si t precede a u , a C_{tu}^B se le asigna un 1; en caso contrario, se le asigna un 0.

$$C_{tu}^B = \begin{cases} 0.5 & \text{si } t \text{ y } u \text{ pertenecen al mismo grupo en } B, \\ 1 & \text{si } t \text{ precede a } u \text{ en } B, \\ 0 & \text{si } u \text{ precede a } t \text{ en } B. \end{cases} \quad (3.1)$$

Formalmente, el problema *bucket ordering problem* (problema de ordenación de cubos) es definido como el siguiente problema de optimización.

Definición 1: (El BOP Problem) Dado un conjunto de objetos I de n objetos y una matriz de preferencias C sobre I , el problema de ordenación de cubos consiste en encontrar un orden de cubos B de I , el cual minimice la distancia dada por la siguiente expresión:

$$d(C, C^B) = \sum_t^n \sum_u^n |C_{tu} - C_{tu}^B|, \quad (3.2)$$

Propiedades: Para todo elemento t y u en C y C^B se cumple que:

- $C_{tu} + C_{ut} = 1$
- $C_{tt} = 0.5$

3.2. Ejemplo de prueba

Supongamos que tenemos un conjunto de cinco clasificaciones totales de un conjunto de objetos I , el cual contiene seis objetos $\{a, b, c, d, e, f\}$ como se muestra en el Cuadro 3.1.

Dadas las clasificaciones de el Cuadro 3.1, calculamos la matriz de precedencias C mostrada en el Cuadro 3.2.

Cuadro 3.1: Ejemplo de una clasificación de 5 objetos

T_1 :	a	b	c	d	e	f
T_2 :	a	c	b	d	f	e
T_3 :	a	c	d	b	f	e
T_4 :	d	f	c	a	b	e
T_5 :	c	f	e	d	b	a

Cuadro 3.2: Matriz de precedencia C

	a	b	c	d	e	f
a	0.5	0.8	0.6	0.6	0.8	0.8
b	0.2	0.5	0.2	0.4	0.8	0.6
c	0.4	0.8	0.5	0.8	1	0.8
d	0.4	0.6	0.2	0.5	0.8	0.8
e	0.2	0.2	0	0.2	0.5	0.2
f	0.4	0.4	0.2	0.2	0.8	0.5

La forma de calcular la matriz es la siguiente:

- La diagonal deberíamos dejarla vacía puesto que no existe la probabilidad de que, por ejemplo, a preceda a a . Pero para que no sume nada cuando hagamos la fórmula de la distancia, la rellenamos con el valor 0.5.
- Para calcular el resto de probabilidades es tan sencillo como contar las veces que t precede a u en los distintos rankings totales y dividir este valor entre la cantidad de rankings totales que tenemos. Por ejemplo:
 - Para calcular la probabilidad C_{ab} , contamos en el Cuadro 3.1 la veces que a se encuentra clasificado por delante de b , que en este caso son 4 (T_1 , T_2 , T_3 , T_4), y lo dividimos entre el total de clasificaciones que es 5. Por tanto, $C_{ab} = 4/5 = 0.8$.
 - Para calcular la probabilidad C_{fc} , contamos en el Cuadro 3.1 la veces que f se encuentra clasificado por delante de c , que en este caso es 1 (T_4), y lo dividimos entre el total de clasificaciones que es 5. Por tanto, $C_{fc} = 1/5 = 0.2$.
 - Y así calculamos todas las probabilidades.

Teniendo la anterior matriz de probabilidades de precedencia, el siguiente paso es calcular la matriz C_{tu}^B . Para calcular dicha matriz tenemos que definir un orden

de cubos B e ir asignando los valores 0.5, 1 y 0 tal y como aparece en la fórmula 3.1. La idea es ir calculando dicha matriz para todos los posibles ordenes de cubos y mediante la fórmula 3.2 calculamos la distancia. Como queremos minimizar la distancia, nos quedaremos con el orden de cubos B que tenga la distancia más pequeña.

Vamos a realizar dos posibles agrupaciones para ejemplificar el problema.

■ **Agrupación 1.**

Una posible agrupación podría ser obtener los grupos mediante la posición media que ocupa cada elemento y tratar de hacer grupos a partir de la diferencia que haya entre las medias.

$$\bar{a} = \frac{1+1+1+4+6}{5} = 2.6$$

$$\bar{b} = \frac{2+3+4+5+5}{5} = 3.8$$

$$\bar{c} = \frac{3+2+2+3+1}{5} = 2.2$$

$$\bar{d} = \frac{4+4+3+1+4}{5} = 3.4$$

$$\bar{e} = \frac{5+6+6+6+3}{5} = 5.2$$

$$\bar{f} = \frac{5+4+4+2+2}{5} = 3.4$$

Al ver la diferencia que hay entre los distintos objetos, los grupos que podríamos hacer intuitivamente serían los siguientes $\{a, c\}$, $\{b, d, f\}$, $\{e\}$. A partir de esta agrupación formamos la matriz C_{tu}^B que quedaría de la siguiente forma:

Cuadro 3.3: Matriz de precedencia C^B

	a	b	c	d	e	f
a	0.5	1	0.5	1	1	1
b	0	0.5	0	0.5	1	0.5
c	0.5	1	0.5	1	1	1
d	0	0.5	0	0.5	1	0.5
e	0	0	0	0	0.5	0
f	0	0.5	0	0.5	1	0.5

Como en el caso de la matriz C_{tu} en la diagonal colocamos 0.5. Para calcular el resto de elementos de la matriz, es tan sencillo como ir asignando los valores correspondientes a la fórmula 3.1 Por ejemplo:

- Le asignamos 1 a la posición C_{ab}^B puesto que en la agrupación supuesta a precede a b .
- Le asignamos 0.5 a la posición C_{ac}^B puesto que en la agrupación supuesta a pertenece al mismo grupo que c .
- Le asignamos 0 al valor de la posición C_{bc}^B puesto que en la agrupación supuesta c precede a b .

Habiendo calculado tanto las matrices C y C^B , calculamos la distancia mediante la fórmula 3.2.

$$d(C, C^B) = |C_{aa} - C_{aa}^B| + |C_{ab} - C_{ab}^B| + \dots + |C_{fe} - C_{fe}^B| + |C_{ff} - C_{ff}^B| = \\ |0.5 - 0.5| + |0.8 - 1| + |0.6 - 0.5| + \dots + |0.8 - 1| + |0.5 - 0.5| = 6$$

■ Agrupación 2.

Otra posible agrupación podría ser $\{a, b, c, d\}, \{e, f\}$. Dando como resultado la siguiente matriz C^B :

Cuadro 3.4: Matriz de precedencia C^B

	a	b	c	d	e	f
a	0.5	0.5	0.5	0.5	1	1
b	0.5	0.5	0.5	0.5	1	1
c	0.5	0.5	0.5	0.5	1	1
d	0.5	0.5	0.5	0.5	1	1
e	0	0	0	0	0.5	0.5
f	0	0	0	0	0.5	0.5

Por tanto, la distancia sería:

$$d(C, C^B) = |0.5 - 0.5| + |0.8 - 0.5| + |0.6 - 0.5| + \dots + |0.8 - 0.5| + |0.5 - 0.5| = 6.6$$

El problema principal que tiene este cálculo, es que hay muchas agrupaciones posibles y, como no sabemos cual es la mejor, tenemos que probar todas. Esto conlleva a que el problema sea interminable. Para solucionar este problema introduciremos un algoritmo que permite obtener el orden de cubos óptimo.

3.3. El algoritmo GAP

En esta sección introducimos el algoritmo GAP [1], el cual trata de solucionar el problema BOP. Este algoritmo usa una heurística que se basa en la diferencia de los rangos para segmentar muchas clasificaciones hechas a partir de diferentes cuantiles y consiste en 2 fases de agregación de rankings.

3.3.1. Introducción al GAP

La idea básica del GAP es comprobar la similitud del rango de dos objetos con el fin de decidir si estos deben ser puestos en el mismo grupo. El GAP primero crea un orden de cubos inicial, correspondiente a cada rango cuantílico, para después agregar todos estos ordenes de cubos y producir un único orden de cubos. Formalmente, un rango cuantílico r es definido de la siguiente forma:

Definición 2: (Rango cuantílico) Dado un conjunto T de m clasificaciones totales $\{T_1, T_2, \dots, T_m\}$ sobre n objetos, y un cuantil q . El rango cuantílico r de un objeto i con respecto del cuantil q es la mínima posición en la cual al menos $[m \times q]$ de los rankings totales clasifican al elemento i .

Por ejemplo, consideramos los rankings totales dados en la tabla 3.1. El rango cuantílico del objeto a con respecto al cuantil 30% es 1, puesto que es en la primera ronda en la cual el objeto a aparece por primera vez al menos $[5 \times 30\%] = 1.5 \simeq 2$ veces. De forma similar, el rango cuantílico del objeto e con respecto al cuantil 30% es 5, puesto que es en la quinta ronda cuando el objeto e aparece por primera vez al menos 2 veces.

El algoritmo GAP consiste en la siguientes 3 etapas:

- Etapa 1: estimamos diferentes clasificaciones a partir de cuantiles y usamos el rango cuantílico para ordenar los objetos en orden ascendente. En esta etapa obtendremos un ranking total para cada cuantil seleccionado.
- Etapa 2: usamos una heurística denominada ARG para calcular el número de grupos a hacer en nuestro orden de cubos y segmentar las clasificaciones

obtenidas en la Etapa 1. En esta etapa pasaremos de los rankings totales de la etapa anterior a un orden de cubos por cada cuantil seleccionado. ARG define un criterio para formar grupos, el cual informalmente supone que, si dos objetos seguidos a lo largo de una clasificación dada por los distintos cuantiles tienen una diferencia “anormal” en sus rangos cuantílicos, estos dos objetos deberían ser puestos en dos grupos diferentes.

- Etapa 3: seleccionamos los ordenes de cubos generados en la Etapa 2 y obtenemos posibles ordenes de cubos B . Finalmente, calculamos la distancia con la fórmula 3.2 definida en la sección 3.1 y nos quedamos con el orden de cubos con la menor distancia.

Para cada etapa, primero haremos una introducción teórica de la etapa, para luego pasar a explicarla mediante un ejemplo.

3.3.2. Etapa 1

Esta etapa consiste en seleccionar un número k de cuantiles q mediante los cuales obtendremos k rankings totales $\{Q_1, Q_2, \dots, Q_k\}$. Cada clasificación Q es obtenida al ordenar los objetos de forma ascendente mediante el rango cuantílico (r). Para calcular r usamos la definición 2 del apartado 3.3.1.

Dados los cuantiles $q = (0.3 \ 0.5 \ 0.7 \ 0.9)$ y usando el ejemplo de prueba del Cuadro 3.1 del apartado 3.2, obtendremos las correspondientes clasificaciones Q_1 , Q_2 , Q_3 y Q_4 como se muestra en el Cuadro 3.5:

Cuadro 3.5: Etapa 1

Cuantil	Ranking total
0.3	$Q_1 : \langle a : 1, c : 2, f : 2, b : 3, d : 3, e : 5 \rangle$
0.5	$Q_2 : \langle a : 1, c : 2, b : 4, d : 4, f : 5, e : 6 \rangle$
0.7	$Q_3 : \langle c : 3, a : 4, d : 4, b : 5, f : 5, e : 6 \rangle$
0.9	$Q_4 : \langle c : 3, d : 4, b : 5, a : 6, e : 6, f : 6 \rangle$

Los números que aparecen en la tabla son los rangos cuantílicos de cada objeto para los diferentes cuantiles. Por ejemplo, la clasificación Q_1 es $\langle a : 1, c : 2, f : 2, b : 3, d : 3, e : 5 \rangle$, donde el objeto a tiene un rango cuantílico 1 y el objeto b tiene un

rango cuantílico 3. Aplicando la definición 2, calculamos los rangos cuantílicos de la siguiente forma:

- $q = 0.3$

Calculamos el número mínimo de veces en las que tiene que estar clasificado un objeto $[5 \times 0.3] = 1.5 \simeq 2$ veces. Observando el Cuadro 3.1 comprobamos que a aparece en la primera posición del ranking en tres de los cinco rankings totales $\{T_1, T_2, T_3\}$, y ,por consiguiente, como aparece clasificado en dos ocasiones o más en la primera posición, $r_a = 1$. Para calcular r_b hacemos el mismo procedimiento, comprobamos cual es la posición más pequeña en la que b ha aparecido clasificado al menos dos veces. Vemos que b aparece clasificado en segunda posición en T_1 y en tercera posición en T_2 , por tanto, $r_b = 3$. $r_c = 2$ ya que aparece en T_1 en primera posición y en $\{T_2, T_3\}$ en segunda posición. Y así sucesivamente.

Cuando hemos terminado de calcular los diferentes rangos los ordenamos en orden ascendente y da como resultado $Q_1 : \langle a : 1, c : 2, f : 2, b : 3, d : 3, e : 5 \rangle$.

- El siguiente paso es realizar lo mismo para el resto de cuantiles.
 - $q = 0.5 \rightarrow [5 \times 0.5] = 2.5 \simeq 3 \rightarrow Q_2 : \langle a : 1, c : 2, b : 4, d : 4, f : 5, e : 6 \rangle$
 - $q = 0.7 \rightarrow [5 \times 0.7] = 3.5 \simeq 4 \rightarrow Q_3 : \langle c : 3, a : 4, d : 4, b : 5, f : 5, e : 6 \rangle$
 - $q = 0.9 \rightarrow [5 \times 0.9] = 4.5 \simeq 5 \rightarrow Q_4 : \langle c : 3, d : 4, b : 5, a : 6, e : 6, f : 6 \rangle$

3.3.3. Etapa 2

Dado un conjunto de clasificaciones totales generadas en la etapa anterior, GAP segmenta cada clasificación Q en un orden de cubos B_Q . Intuitivamente, necesitamos seleccionar algún objeto como frontera para separar Q en una serie de subintervalos, y cada subintervalo es un grupo de B_Q .

A diferencia de otras heurísticas de segmentación, la heurística ARG no necesita que especifiquemos previamente el número de grupos que queremos hacer. Existen

ocasiones en las que hay expertos en la materia que nos indican un número estimado de grupos. La heurística ARG puede descubrir ordenes de cubos con o sin especificarle previamente el número de grupos.

Existen métodos de segmentación que tratan de realizar la agrupación maximizando la similitud de los objetos intra-grupo (dentro de un mismo grupo). En contraste, la heurística ARG realiza la agrupación maximizando la disimilitud de los objetos inter-grupo (entre grupos). Esta disimilitud inter-grupo es recogida mediante el concepto de “abnormal gap” (diferencia anormal) en los rangos cuantílicos. Objetos que pertenezcan al mismo grupo deberían tener rangos cuantílicos que sean tan parecidos como sea posible, y del mismo modo, objetos que pertenezcan a distintos grupos deberían tender a tener rangos cuantílicos que sean tan diferentes como sea posible. De forma más sencilla, si dos objetos clasificados consecutivamente en una dada clasificación cuantílica Q tienen una diferencia anormal entre sus rangos cuantílicos, estos dos objetos es muy probable que definan la frontera de dos grupos diferentes. Antes de definir la heurística ARG, necesitamos introducir el concepto de rank gap (diferencia entre rangos) y abnormal rank gap (diferencia anormal entre rangos)(ARG).

Definición 3: (Diferencia entre rangos) Dados dos objetos consecutivos $Q[i]$ y $Q[i + 1]$ en una dada clasificación cuantílica Q de n objetos, denotamos sus rangos cuantílicos como $R(i)$ y $R(i + 1)$ respectivamente. La diferencia entre rangos g_i es definida de la siguiente forma:

$$g_i = |R(i) - R(i + 1)|. \quad (3.3)$$

Para las clasificaciones cuantílicas Q dadas, obtenemos un vector de diferencia de rangos g_1, g_2, \dots, g_{n-1} , donde cada diferencia g_i es mayor o igual a 0. Entonces, calculamos la *diferencia media* y la *desviación estándar* del vector de la siguiente forma:

$$\mu = \frac{1}{n-1} \sum_{i=1}^{n-1} g_i, \quad (3.4)$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n-1} (g_i - \mu)^2}. \quad (3.5)$$

Definición 4: (Diferencia anormal entre rangos) Dado una diferencia entre rangos g_i , tenemos una diferencia anormal entre rangos (ARG) si:

$$g_i > \mu + \sigma. \quad (3.6)$$

Usando las definiciones 3 y 4, podemos entonces definir formalmente la heurística ARG, la cual consta de dos partes: la heurística ARG-A y la heurística ARG-K. Primero usamos la heurística ARG-A para obtener el número estimado de grupos en el cual hay que segmentar cada clasificación cuantílica y luego usamos la heurística ARG-K para obtener una clasificación parcial para cada clasificación cuantílica.

Definición 5: (Heurística ARG-A) Dado un conjunto de m clasificaciones cuantílicas $\{Q_1, Q_2, \dots, Q_m\}$, definimos N_i como las diferencias anormales entre rangos de cada clasificación cuantílica Q_i . Supón que N_1, N_2, \dots, N_m están ordenados de forma ascendente de tal forma $N_1 \leq N_2 \leq \dots \leq N_m$. Definimos la *mediana* (N_1, N_2, \dots, N_m) , denotada por N_a , de la siguiente forma:

- (1) N_a es $N_{\frac{(m+1)}{2}}$, cuando m es impar;
- (2) N_a es $N_{(\frac{m}{2})+1}$, cuando m es par.

Entonces, usamos $(N_a + 1)$ como el número estimado de grupos en los que dividir cada clasificación cuantílica Q_i .

Supón que g_i es un ARG de una clasificación cuantílica Q . Consideramos que el

objeto $Q[i]$ marca el final de un grupo y el objeto $Q[i+1]$ marca el inicio del siguiente grupo. A partir de identificar el número de ARGs, somos capaces de determinar el número de grupos. Entonces, al usar la mediana de todos los N_i , somos capaces de obtener una estimación robusta del número final de grupos ($N_i + 1$). Finalmente, segmentamos cada clasificación cuantílica Q_i usando la heurística ARG-K.

Definición 6: (Heurística ARG-K) Dado el número de grupos k , agrupamos las clasificaciones cuantílicas obtenidas en k grupos. Aquí k puede ser $(N_a + 1)$ o un número preestablecido proporcionado por algún experto. Para usar la heurística ARG-K, primero tenemos que ordenar la diferencia entre los rangos g_1, g_2, \dots, g_{n-1} , y entonces seleccionar $(k - 1)$ parejas de objetos $(Q[i], Q[i + 1])$ que tengan las $(k - 1)$ mayores diferencias y considerar esos objetos como la frontera entre dos grupos.

Cuadro 3.6: Etapa 2

Cuantiles	Oden de cubos
0.3	B1: $\langle \{a, b, c, d, f\}, \{e\} \rangle$
0.5	B2: $\langle \{a, c\}, \{b, d, e, f\} \rangle$
0.7	B3: $\langle \{a, b, c, d, f\}, \{e\} \rangle$
0.9	B4: $\langle \{b, c, d\}, \{a, e, f\} \rangle$

En el Cuadro 3.6 vemos el orden de cubos correspondiente a cada cuantil. La agrupación para el cuantil 0.3 es B1: $\langle \{a, b, c, d, f\}, \{e\} \rangle$, para el cuantil 0.5 es B2: $\langle \{a, c\}, \{b, d, e, f\} \rangle$, para 0.7 es B3: $\langle \{a, b, c, d, f\}, \{e\} \rangle$ y para 0.9 es B4: $\langle \{a, b, c, d\}, \{e, f\} \rangle$. Partiendo del Cuadro 3.5, como hemos explicado anteriormente, el procedimiento para calcular las diferentes clasificaciones es el siguiente: primero calculamos, mediante las fórmulas de la definición 3, la diferencia entre rangos, la media y la desviación; después, mediante la definición 4, comprobamos las diferencias anormales entre rangos; el siguiente paso es ordenar de menor a mayor las diferencias anormales entre rangos que tiene cada cuantil y, aplicando la definición 5, usamos la mediana para estimar el número de grupos; el último paso es segmentar las clasificaciones totales en la cantidad de grupos estimada, para esto usamos la definición 6 y seleccionamos los objetos que mayor diferencia entre rangos tengan para separar los grupos.

- $q = 0.3 \rightarrow Q_1 : \langle a : 1, c : 2, f : 2, b : 3, d : 3, e : 5 \rangle$

- Calculamos la diferencia entre rangos g_i aplicando la fórmula 3.3:

$$g_1 = |R(1) - R(2)| = |1 - 2| = 1$$

$$g_2 = |R(2) - R(3)| = |2 - 2| = 0$$

$$g_3 = |R(3) - R(4)| = |2 - 3| = 1$$

$$g_4 = |R(4) - R(5)| = |3 - 3| = 0$$

$$g_5 = |R(5) - R(6)| = |3 - 5| = 2$$

Aplicando las fórmulas de la media (3.4) y la desviación (3.5) obtenemos:

$$\mu = \frac{1}{n-1} \sum_{i=1}^{n-1} g_i = \frac{1+0+1+0+2}{6-1} = 0.8$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n-1} (g_i - \mu)^2} = \sqrt{\frac{(1-0.8)^2 + (0-0.8)^2 + (1-0.8)^2 + (0-0.8)^2 + (2-0.8)^2}{6-1}} = 0.75$$

- Comprobamos la condición de la definición 4:

$$g_1 > \mu + \sigma \rightarrow 1 \not> 0.8 + 0.75$$

$$g_2 > \mu + \sigma \rightarrow 0 \not> 0.8 + 0.75$$

$$g_3 > \mu + \sigma \rightarrow 1 \not> 0.8 + 0.75$$

$$g_4 > \mu + \sigma \rightarrow 0 \not> 0.8 + 0.75$$

$$g_5 > \mu + \sigma \rightarrow 2 > 0.8 + 0.75$$

Existe una diferencia anormal entre rangos en g_5 .

- De la misma forma calculamos el resto de cuantiles:

- $q = 0.5 \rightarrow Q_2 : \langle a : 1, c : 2, b : 4, d : 4, f : 5, e : 6 \rangle$

$$g_i = (1, 2, 0, 1, 1), \mu = 1, \sigma = 0.63$$

Obtenemos una diferencia anormal (g_2).

- $q = 0.7 \rightarrow Q_3 : \langle c : 3, a : 4, d : 4, b : 5, f : 5, e : 6 \rangle$

$$g_i = (1, 0, 1, 0, 1), \mu = 0.6, \sigma = 0.49$$

No obtenemos ninguna diferencia anormal.

- $q = 0.9 \rightarrow Q_4 : \langle c : 3, d : 4, b : 5, a : 6, e : 6, f : 6 \rangle$

$$g_i = (1, 1, 1, 0, 0), \mu = 0.6, \sigma = 0.49$$

No obtenemos ninguna diferencia anormal.

- Aplicando la definición 5 ordenamos los cuantiles a partir de las diferencias anormales entre rangos que hay en cada uno. Los cuantiles 0.3 y 0.5 tienen

una diferencia anormal y los cuantiles 0.7 y 0.9 no tienen ninguna diferencia anormal. Por tanto:

- $q = 0.3 \rightarrow N_1 = 1$
- $q = 0.5 \rightarrow N_2 = 1$
- $q = 0.7 \rightarrow N_3 = 0$
- $q = 0.9 \rightarrow N_4 = 0$

Ordenamos de menor a mayor y nos queda $N_3 \leq N_4 \leq N_1 \leq N_2$.

Teniendo los cuantiles ordenados a partir del número de diferencias anormales que hay en cada uno, calculamos la mediana. Al ser par, usamos la fórmula $N_{(\frac{m}{2})+1} \rightarrow N_{(\frac{4}{2})+1} = 3$. Cogemos el N del cuantil que se encuentra en la tercera posición, el cual es $N_1 = 1$. Y usamos la fórmula $N_a + 1 = N_1 + 1 = 2$ para estimar el número de grupos a segmentar, dando como resultado 2 grupos.

- El último paso de esta etapa es utilizar la definición 6 para segmentar las clasificaciones totales obtenidas en la Etapa 1. Para hacer esta segmentación tenemos que mirar la diferencia entre rangos y , como solo tenemos que hacer dos grupos, los objetos que presenten la mayor diferencia entre rangos serán los que marquen la frontera de los dos grupos.

- $q_1 = 0.3 \rightarrow g_i = (1, 0, 1, 0, 2)$

La mayor diferencia es $g_5 = 2$. Como la clasificación a partir de los rangos obtenida en la Etapa 1 es $Q_1 : \langle a : 1, c : 2, f : 2, b : 3, d : 3, e : 5 \rangle$, g_5 corresponde a la diferencia entre d y e , por tanto, estos dos objetos marcan la frontera y la clasificación queda de la siguiente forma B1: $\langle \{a, b, c, d, f\}, \{e\} \rangle$.

- Del mismo modo segmentamos el resto de cuantiles.

- $q_2 = 0.5 \rightarrow g_i = (1, 2, 0, 1, 1)$

B2: $\langle \{a, c\}, \{b, d, e, f\} \rangle$

- $q_3 = 0.7 \rightarrow g_i = (1, 0, 1, 0, 1)$

Como tenemos tres diferencias anormales de rangos iguales podemos escoger cualquiera de esas tres diferencias como frontera, en

nuestro caso escogeremos la última de esas diferencias. En este caso

g_5 .

B3: $\langle \{a, b, c, d, f\}, \{e\} \rangle$

◦ $q_4 = 0.9 \rightarrow g_i = (1, 1, 1, 0, 0)$

Como en el caso anterior, escogemos como frontera la última de estas diferencias que son iguales. En este caso g_3 .

B4: $\langle \{b, c, d\}, \{a, e, f\} \rangle$

3.3.4. Etapa 3

Después de haber segmentado cada clasificación cuantílica en un orden de cubos inicial, empezamos la siguiente fase de la agregación para generar el orden de cubos final. En esta etapa generamos la agrupación final B y usamos la fórmula 3.2 para calcular la calidad de dicha agrupación. Para generar la agrupación final decimos que: si dos elementos están en el mismo grupo en al menos el 50% de los ordenes de cubos iniciales (los ordenes de cubos obtenidos en la etapa anterior), entonces estos deberían estar juntos en el orden de cubos final.

Siguiendo con el mismo ejemplo, a partir de las agrupaciones obtenidas en la etapa anterior presentes en el Cuadro 3.6, tenemos lo siguiente:

El objeto a y c están agrupados juntos en tres de los cuatro grupos (B_1, B_2, B_3) , es decir, en el 75% de las ocasiones, por tanto, estos dos objetos deben estar en el mismo grupo en la clasificación final. Además los objetos b , c y d , también están agrupados juntos en el 75% de las ocasiones (B_1, B_3, B_4) . Entonces estos tres objetos también deben estar en el mismo grupo. Como c debe estar en el mismo grupo que a , b y d , entonces a debe estar en el mismo grupo que b y d , dando como resultado el siguiente grupo $\{a, b, c, d\}$. Además el objeto e está clasificado en todas las ocasiones en el segundo grupo, y no coincide en el mismo grupo con ninguno de los objetos anteriores en al menos el 50% de las ocasiones, por tanto, en la agrupación final pertenece al segundo grupo $\{e\}$. Por último queda por clasificar el objeto f , que aunque parece que debe estar agrupado en el primer grupo debido a que está clasificado el 75% de las ocasiones con los objetos b y d , está clasificado con

e el 50% de las ocasiones, por tanto, vamos a escoger las siguientes agrupaciones como posibles $B_1 = \{a, b, c, d, f\}, \{e\}$ y $B_2 = \{a, b, c, d\}, \{e, f\}$ y comprobamos con la fórmula 3.2 cuál de estas debemos escoger.

Para calcular la distancia necesitamos la matriz C y C^B . Recordamos tanto la fórmula de la distancia como la matriz C :

$$d(C, C^B) = \sum_t^n \sum_u^n |C_{tu} - C_{tu}^B|$$

Cuadro 3.7: Matriz de precedencia C

	a	b	c	d	e	f
a	0.5	0.8	0.6	0.6	0.8	0.8
b	0.2	0.5	0.2	0.4	0.8	0.6
c	0.4	0.8	0.5	0.8	1	0.8
d	0.4	0.6	0.2	0.5	0.8	0.8
e	0.2	0.2	0	0.2	0.5	0.2
f	0.4	0.4	0.2	0.2	0.8	0.5

Calculamos la matriz C^B y la distancia:

- $B_1 = \{a, b, c, d, f\}, \{e\}$

Calculamos la matriz C^{B_1} :

Cuadro 3.8: Matriz de precedencia C^{B_1}

	a	b	c	d	e	f
a	0.5	0.5	0.5	0.5	1	0.5
b	0.5	0.5	0.5	0.5	1	0.5
c	0.5	0.5	0.5	0.5	1	0.5
d	0.5	0.5	0.5	0.5	1	0.5
e	0	0	0	0	0.5	0
f	0.5	0.5	0.5	0.5	1	0.5

$$d(C, C^{B_1}) = |0.5 - 0.5| + |0.8 - 0.5| + \dots + |0.8 - 1| + |0.5 - 0.5| = 5.6$$

- $B_2 = \{a, b, c, d\}, \{e, f\}$

Calculada en la agrupación 2 del apartado 4.2:

$$d(C, C^{B_2}) = 6.6$$

Al ser la distancia para la agrupación B_1 menor que la distancia para B_2 , nuestra agrupación final es $B = \{a, b, c, d, f\}, \{e\}$.

3.3.5. Evaluación final

En este apartado vamos a realizar una evaluación de los distintos grupos obtenidos usando diferentes medidas.

Vamos a introducir un nuevo concepto que se llama porcentaje de cumplimiento, el cual hace referencia al total de clasificaciones totales que cumplen con el orden de cubos seleccionado. Es decir, con esta medida queremos conocer, a partir de nuestra muestra, cuantas clasificaciones totales realmente cumplen o no con el orden de cubos obtenido.

Cuadro 3.9: Evaluación Final

Agrupaciones	Distancia	Porcentaje de cumplimiento
$B_1 : \langle \{a, c\}, \{b, d, f\}, \{e\} \rangle$	6	40 %
$B_2 : \langle \{a, b, c, d\}, \{e, f\} \rangle$	6.6	60 %
$B_3 : \langle \{a, b, c, d, f\}, \{e\} \rangle$	5.6	60 %

Para calcular el porcentaje de cumplimiento tenemos que mirar en nuestra muestra cuantas clasificaciones totales siguen el orden de cubos B_i y este valor lo dividimos entre el total de clasificaciones totales. Vemos que la agrupación B_1 sólo representa bien el 40 % de la muestra, puesto que únicamente clasifica bien dos clasificaciones totales (T_1 y T_2). Dividiendo $2/5$, cinco es el total de clasificaciones totales que tenemos, obtenemos 0.4 y pasándolo a porcentaje obtenemos el 40 %. La agrupación B_2 clasifica bien las muestras T_1 , T_2 y T_3 , por tanto, el porcentaje de cumplimiento es 60 %. Por último, la agrupación B_3 clasifica bien las muestras T_2 , T_3 y T_4 dando un 60 % de cumplimiento.

Mediante esta evaluación final, podemos concluir que la mejor agrupación es B_3 puesto que tiene un porcentaje de cumplimiento mayor y una distancia menor.

Capítulo 4

Aplicación a los Caballos del Vino

4.1. Los Caballos del Vino

Se trata de un festejo único en el mundo que se celebra los primeros días de mayo en Caravaca de la Cruz. Una fiesta que combina el arte, la competición, la tradición, la convivencia, la historia y el amor por los caballos.

4.1.1. Historia de los Caballos del Vino

Según cuenta la leyenda, en el siglo XIII ocurrió un hecho histórico en el Castillo de Caravaca. Cuando la población cristiana se encontraba dentro de la fortaleza debido a las constantes batallas que sufría la villa de Caravaca, mientras los alrededores eran tomados por las tropas musulmanas, los alimentos empezaron a ser escasos, debido a los saqueos y los daños que se producían en los huertos de la ciudad. Los cristianos empezaron a enfermar y las reservas de los alimentos a desaparecer.

Decidieron ir en busca de alimento, porque incluso las aguas de los aljibes se habían infectado. Tras largas caminatas pudieron encontrar una casa con bodegas en su interior, al no encontrar agua, decidieron cargar el vino en los caballos. Volvieron velozmente pero, a la llegada al santuario, encontraron un gran cerco musulmán que impedía el paso. Con una espectacular carrera burlaron el cerco enemigo para llevar

el líquido a los defensores del Castillo.

Al llegar fueron recibidos con alborozo. Las mujeres ofrecieron a los mozos y a los caballos mantos bordados y ramilletes de flores, considerándolos héroes y salvadores de la situación. Posteriormente el vino fue bendecido por la Stma. Cruz de Caravaca, el cual fue dado a beber a los enfermos, que milagrosamente sanaron al beberlo.

4.1.2. La fiesta de los Caballos del Vino

En la actualidad consiste en un triple concurso repleto de fuerza, belleza y emoción: el de caballo a pelo, donde se valorara la figura y el porte del animal; el de enjaezamiento, que premia la belleza y calidad de las piezas y su adecuación al caballo que lo porta; y el de carrera, donde destreza y velocidad se enfrentan al implacable veredicto de cronómetro.

La fiesta comienza el 1 de mayo con el concurso a pelo. Las peñas, con sus correspondientes caballos, recorren las calles de la ciudad, donde se premia la morfología del equino en su estado puro.

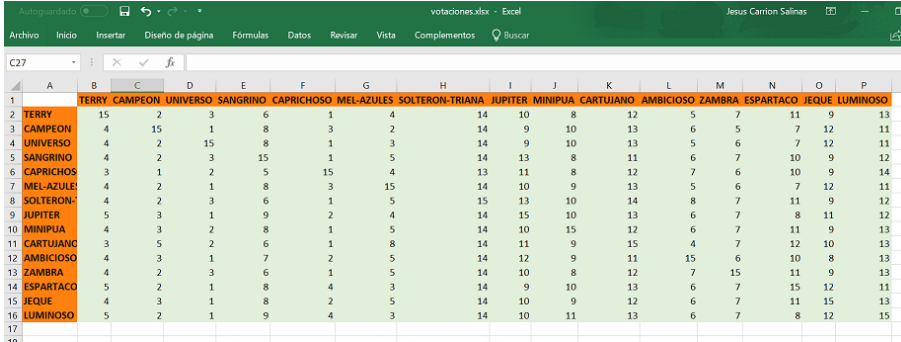
El 2 de mayo es el día grande de la fiesta de los Caballos del Vino. La actividad comienza en plena madrugada con el ritual de “vestir al caballo”. Tras varios pasacalles y desfiles, a medio día, en la subida al castillo, tiene lugar la legendaria carrera de los Caballos del Vino.

Se trata de una prueba cronometrada, en la cual un caballo recorre 80 metros acompañado por cuatro mozos que corren a su lado, agarrados a él, dos delante y dos detrás, produciéndose la eliminación si alguno de ellos se suelta antes de cruzar la meta.

Para cerrar las fiestas, el día 3 de mayo tiene lugar la participación de los más pequeños, quienes realizan un pasacalles y una carrera.

4.2. Algoritmo de la agregación de cubos para los Caballos del Vino

Para la aplicación de la heurística GAP a los Caballos del Vino, disponemos de la base de datos denominada votaciones.xlsx, la cual ha sido rescatada de hace unos años y la hemos modificado un poco para mantener el anonimato. Dicha base de datos presenta la siguiente forma:



	TERRY	CAMPEON	UNIVERSO	SANGRINO	CAPRICHOSO	MEL-AZULES	SOLTERON-TRIANA	JUPITER	MINPIUA	CARTUJANO	AMBICIOSO	ZAMBRA	ESPARTACO	JEQUE	LUMINOSO
TERRY	15	2	3	6	1	4	14	10	8	12	5	7	11	9	13
CAMPEON	4	15	1	8	3	2	14	9	10	13	6	5	7	12	11
UNIVERSO	4	2	15	8	1	3	14	9	10	13	5	6	7	12	11
SANGRINO	4	2	3	15	1	5	14	13	8	11	6	7	10	9	12
CAPRICHOSO	3	1	2	5	15	4	13	11	8	12	7	6	10	9	14
MEL-AZULES	4	2	1	8	3	15	14	10	9	13	5	6	7	12	11
SOLTERON-TRIANA	4	2	3	6	1	5	15	13	10	14	8	7	11	9	12
JUPITER	5	3	1	9	2	4	14	15	10	13	6	7	8	11	12
MINPIUA	4	3	2	8	1	5	14	10	15	12	6	7	11	9	13
CARTUJANO	3	5	2	6	1	8	14	11	9	15	4	7	12	10	13
AMBICIOSO	4	3	1	7	2	5	14	12	9	11	15	6	10	8	13
ZAMBRA	4	2	3	6	1	5	14	10	8	12	7	15	11	9	13
ESPARTACO	5	2	1	8	4	3	14	9	10	13	6	7	15	12	11
JEQUE	4	3	1	8	2	5	14	10	9	12	6	7	11	15	13
LUMINOSO	5	2	1	9	4	3	14	10	11	13	6	7	8	12	15

Figura 4.1: Votaciones

Como vemos en la figura 4.1 disponemos de una tabla de doble entrada, en la que 15 peñas distintas realizan una clasificación del resto de las peñas. De tal forma que, por ejemplo, la peña Terry ha clasificado en primer lugar a Caprichoso, en segundo lugar a Campeón, en tercer lugar a Universo y así sucesivamente. Por tanto, decimos que cada fila corresponde a una clasificación total.

Además, podemos observar que cada peña se ha clasificado a ella misma en último lugar. En nuestro caso, no vamos a realizar ninguna modificación, puesto que suponemos que si todas las peñas se han clasificado a ellos mismos en último lugar, esto no va a influir en nuestros resultados. En caso de querer modificar la base de datos, una posible corrección de este valor sería obtener la posición media de cada peña y en su propia clasificación, colocar dicha peña en esa posición. De tal forma que tendríamos que mover una posición las peñas clasificadas en posiciones posteriores.

Con el fin de resolver este y futuros problemas de agregación de rankings de manera más rápida y sencilla, hemos realizado un algoritmo mediante el lenguaje de programación R que nos permite resolver este tipo de problemas.

Lo primero que hacemos en R es cargar los datos y, como en la primera columna tenemos los nombres de las distintas peñas, eliminamos esta columna.

```
> library(readxl)
> datos <- read_excel(
+   "C:/Users/Jesus Carrion/Desktop/TFG/votaciones.xlsx")
> datos <- datos[-1]
> datos<-data.frame(datos)
```

Calculamos el número de peñas y de clasificaciones totales que tenemos.

```
> #número de objetos
> n<-dim(datos)[2]
> #número de clasificaciones totales
> m<-dim(datos)[1]
```

Montamos la matriz C . Mediante el bucle, lo que hacemos es ir contando en cuantas clasificaciones la peña i precede a la peña j , para después dividir este número entre el total de clasificaciones y, así, obtener la probabilidad. Mediante la última línea de código le insertamos a la diagonal el valor 0.5.

```
> #Montamos la matriz C
> C<-matrix(0,nrow = n, ncol = n)
> colnames(C)<-colnames(datos)
> rownames(C)<-colnames(datos)
> for (i in 1:n){
+   for (j in 1:n){
+     cuenta=0
+     for (t in 1:m){
+       if (datos[t,i]<datos[t,j]){
+         cuenta=cuenta+1
+       }
+       C[i,j]=cuenta
+     }
+   }
+ }
```

```
+ }
> C=C/m
> diag(C)=0.5
```

4.2.1. Etapa 1

Recordamos que en la Etapa 1 tenemos que calcular los rangos cuantílicos.

El script se divide en los siguientes pasos:

- Empezamos introduciendo los cuantiles q y calculando el número de cuantiles k , en nuestro caso tenemos 4 cuantiles.
- Creamos una matriz vacía llamada *rangos*, en la cuál vamos a almacenar los rangos cuantílicos y le ponemos el nombre de las distintas peñas a las columnas.
- Mediante el primer bucle vamos añadiendo los rangos cuantílicos. Como queremos obtener la mínima posición en la que cada peña aparece clasificada al menos $[m * q]$ veces, con la variable auxiliar *cabeza* lo que hacemos es, para cada peña (columna), almacenar los $[m * q]$ primeros valores ordenados de forma ascendente. Mediante la variable auxiliar *rango*, obtenemos el máximo de la variable *cabeza*, el cual es el rango de esa peña, y almacenamos el *rango* en la matriz de *rangos*.
- Mediante el último bucle lo que hacemos es crear una variable Q_i para cada cuantil y le asignamos los rangos cuantílicos ordenados.

```
> #Etapa 1
> q=c(0.3,0.5,0.7,0.9)#cuantiles
> k=length(q)#número de cuantiles
> rangos<-matrix(0, nrow = k, ncol = n)
> colnames(rangos)<-colnames(datos)
> for (i in 1:n) {
+   for (j in 1:k){
+     #obtenemos los [m*q] primeros numeros
```

```

+   #ordenados de forma ascendente
+   cabeza<-head(datos[order(datos[,i], decreasing = FALSE), ]
+               , ceiling(m*q[j]))
+   #seleccionamos el maximo (de esta forma calculamos el rango)
+   rango<-max(cabeza[i])
+   #añadimos el rango a la matriz de rangos cuantílicos
+   rangos[j,i]<-rango
+ }
+ }
> for (i in 1:k){
+   assign(paste("Q",i,sep=""), sort(rangos[i,]))
+ }

```

4.2.2. Etapa 2

Recordemos que en la Etapa 1 tenemos que calcular la diferencia entre rangos, la media y desviación típica de esta diferencia entre rangos, la mediana, que es la que nos dirá el número de grupos a hacer; y obtener un orden de cubos inicial para cada cuantil.

Para explicar el script, vamos a explicar primero la forma de obtener la diferencia entre rangos, la media y la desviación; después explicaremos como obtener la diferencia anormal entre rangos y la mediana; y terminaremos obteniendo el orden de cubos inicial.

- Diferencia entre rangos, media y desviación:

Con la primera parte creamos para cada cuantil un vector g_i en el que almacenaremos la diferencia entre rangos. Posteriormente mediante el bucle almacenamos la diferencia entre rangos. En la segunda parte calculamos para cada cuantil la media de la diferencia entre rangos. Y en la tercera parte la desviación típica.

```

> #Calculamos la diferencia entre rangos
> g1<-c()

```

4.2. ALGORITMO DE LA AGREGACIÓN DE CUBOS PARA LOS CABALLOS DEL VINO29

```
> g2<-c()
> g3<-c()
> g4<-c()
> for (i in 1:n-1) {
+   g1<-c(g1,abs(Q1[i]-Q1[i+1]))
+   g2<-c(g2,abs(Q2[i]-Q2[i+1]))
+   g3<-c(g3,abs(Q3[i]-Q3[i+1]))
+   g4<-c(g4,abs(Q4[i]-Q4[i+1]))
+ }
> #Calculamos la media
> mu1=mean(g1)
> mu2=mean(g2)
> mu3=mean(g3)
> mu4=mean(g4)
> #Calculamos la desviacion tipica
> dt1<-sd(g1)
> dt2<-sd(g2)
> dt3<-sd(g3)
> dt4<-sd(g4)
```

■ Diferencia anormal entre rangos y mediana:

Primero calculamos la suma de la media y la desviación para cada cuantil. Mediante el bucle comprobamos para cada diferencia entre rangos g_i , si esta es mayor que la suma de la media y la desviación y, por tanto, es considerada una diferencia anormal. En caso de ser una diferencia anormal, al vector N , que contiene el total de diferencias anormales que tiene cada cuantil, le sumamos un 1. Es decir, con el bucle lo que vamos haciendo es contar y almacenar el total de diferencias anormales de cada cuantil. Cuando tenemos el vector N , lo ordenamos de menor a mayor con la función *sort*. Mediante el bucle final, comprobamos si el número de cuantiles es par o impar y calculamos la mediana. Para finalizar obtenemos el número de grupos a realizar.

```
> #Calculamos la diferencia anormal entre rangos
> dif1<-mu1+dt1
```

```

> dif2<-mu2+dt2
> dif3<-mu3+dt3
> dif4<-mu4+dt4
> N=c(rep(0,k))
> for (i in 1:length(g1)) {
+   if (g1[i]>dif1){
+     N[1]=N[1]+1
+   }
+   if (g2[i]>dif2){
+     N[2]=N[2]+1
+   }
+   if (g3[i]>dif3){
+     N[3]=N[3]+1
+   }
+   if (g4[i]>dif4){
+     N[4]=N[4]+1
+   }
+ }
> names(N)<-c("g1", "g2", "g3", "g4")
> N<-sort(N)
> #Calculamos la mediana
> if(length(N)%2!=0) {
+   Na=N[(k+1)/2]
+ }else {
+   Na=N[(k/2)+1]
+ }
> names(Na)<-c()
> num_grupos<-Na+1

```

- Orden de cubos inicial:

Primero, calculamos la frontera. Para determinar la frontera, ordenamos de manera ascendente la diferencia entre rangos, y guardamos la posición de los $num_grupos - 1$ últimos valores. Con esto lo que estamos consiguiendo es obtener la posición de las diferencia entre rangos que marcan la frontera. Con

4.2. ALGORITMO DE LA AGREGACIÓN DE CUBOS PARA LOS CABALLOS DEL VINO31

el primer bucle, vamos guardando, en las variables creadas justo encima, la posición de las peñas que marcan el final de un grupo. Creamos una variable auxiliar para cada cuantil que nos dice la posición actual por la que vamos agrupando (empezamos a agrupar por la posición 1). Mediante el bucle vamos a calcular los diferentes grupos para cada cuantil. La forma de calcularlos es por ejemplo en *B1_grupo1* (cuantil 1, grupo 1), almacenamos las peñas que, en la clasificación por rangos, ocupan desde la posición 1 hasta la posición de la peña que marca el final del grupo. Posteriormente, a *pos_actual* le asignamos la posición de la primera peña que va a formar el segundo grupo. Cuando hemos calculado los distintos grupos, mediante el bucle if, calculamos el grupo final.

```
> #calculamos la frontera
> frontera1<-tail(g1[order(g1, decreasing = FALSE)], num_grupos-1)
> frontera2<-tail(g2[order(g2, decreasing = FALSE)], num_grupos-1)
> frontera3<-tail(g3[order(g3, decreasing = FALSE)], num_grupos-1)
> frontera4<-tail(g4[order(g4, decreasing = FALSE)], num_grupos-1)
> #calculo la posicion en Q de los elementos que forman la frontera
> posicion1=c()
> posicion2=c()
> posicion3=c()
> posicion4=c()
> for (i in 1:length(Q1)){
+   for (j in 1:length(frontera1)){
+     if (names(frontera1[j])==names(Q1[i])){
+       posicion1=c(posicion1,i)
+     }
+     if (names(frontera2[j])==names(Q2[i])){
+       posicion2=c(posicion2,i)
+     }
+     if (names(frontera3[j])==names(Q3[i])){
+       posicion3=c(posicion3,i)
+     }
+     if (names(frontera4[j])==names(Q4[i])){
+       posicion4=c(posicion4,i)
+     }
+   }
+ }
```

```

+     }
+   }
+ }
> #agrupa Qi en clasificaciones parciales con los grupos calculados
> pos_actual1=1
> pos_actual2=1
> pos_actual3=1
> pos_actual4=1
> for (i in 1:length(posicion1)){
+   assign(paste("B1_grupo",i,sep=""), c(Q1[pos_actual1:posicion1[i]]))
+   assign(paste("B2_grupo",i,sep=""), c(Q2[pos_actual2:posicion2[i]]))
+   assign(paste("B3_grupo",i,sep=""), c(Q3[pos_actual3:posicion3[i]]))
+   assign(paste("B4_grupo",i,sep=""), c(Q4[pos_actual4:posicion4[i]]))
+   pos_actual1=posicion1[i]+1
+   pos_actual2=posicion2[i]+1
+   pos_actual3=posicion3[i]+1
+   pos_actual4=posicion4[i]+1
+   if (i==length(posicion1)){
+     assign(paste("B1_grupo",i+1,sep=""), c(Q1[pos_actual1:length(Q1)]))
+     assign(paste("B2_grupo",i+1,sep=""), c(Q2[pos_actual2:length(Q2)]))
+     assign(paste("B3_grupo",i+1,sep=""), c(Q3[pos_actual3:length(Q3)]))
+     assign(paste("B4_grupo",i+1,sep=""), c(Q4[pos_actual4:length(Q4)]))
+   }
+ }

```

4.2.3. Etapa 3

Esta tercera etapa recordemos que consiste en comprobar en los ordenes de cubos que obtenemos en la Etapa 2, en cuantos cuantiles una peña i está clasificada con cada una del resto de peñas. Si una peña i coincide en el mismo grupo con una peña j en al menos el 50% de las clasificaciones, la peña i y la peña j deben estar en el mismo grupo en la clasificación final.

Ante la dificultad que conlleva programar esta etapa, vamos a tratar de obtener

4.2. ALGORITMO DE LA AGREGACIÓN DE CUBOS PARA LOS CABALLOS DEL VINO33

el orden de cubos final de la misma forma que lo hicimos en el ejemplo de prueba.

Para hacerlo de manera más sencilla, vamos a realizar una recodificación de peñas. Vamos a sustituir el nombre de cada peña por un número. Mediante el Cuadro 4.1 podemos observar la recodificación.

Cuadro 4.1: Reasignación nombre de peñas

TERRY	1
CAMPEON	2
UNIVERSO	3
SANGRINO	4
CAPRICHOSSO	5
MEL-AZULES	6
SOLTERON-TRIANA	7
JUPITER	8
MINIPUA	9
CARTUJANO	10
AMBICIOSO	11
ZAMBRA	12
ESPARTACO	13
JEQUE	14
LUMINOSO	15

Habiendo recodificado los nombres, mediante el Cuadro 4.2, podemos observar las clasificaciones cuantílicas obtenidas en la Etapa 2.

Cuadro 4.2: Ordenes de cubos iniciales

q=0.3	Grupo1:	{1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 13, 14}
	Grupo2:	{10, 15}
	Grupo3:	{7}
q=0.5	Grupo1:	{2, 3, 5}
	Grupo2:	{1, 4, 6, 8, 9, 11, 12, 13, 14}
	Grupo3:	{7, 10, 15}
q=0.7	Grupo1:	{1, 2, 3, 4, 5, 6, 11, 12}
	Grupo2:	{8, 9, 10, 13, 14, 15}
	Grupo3:	{7}
q=0.9	Grupo1:	{1, 2, 3, 5}
	Grupo2:	{4, 6, 11, 12}
	Grupo3:	{7, 8, 9, 10, 13, 14, 15}

Mediante el Cuadro 4.2, podemos comprobar que las peñas 2, 3 y 5 están clasificadas en los cuatro cuantiles en el mismo grupo, por tanto, estas tres peñas deberán pertenecer al mismo grupo en la clasificación final. Además, las peñas {4, 6, 11, 12},

$\{8, 9, 13, 14\}$ y $\{10, 15\}$, también están clasificadas en todos los cuantiles en el mismo grupo, por tanto, también deberán estar juntas en la clasificación final.

Entonces, tenemos que las siguientes peñas deben estar en el mismo grupo: $\{2, 3, 5\}$, $\{4, 6, 11, 12\}$, $\{8, 9, 13, 14\}$ y $\{10, 15\}$. Y nos quedan las peñas 1 y 7 sueltas. La peña 7 está clasificada el 50% de los cuantiles sola y el 50% con las peñas 10 y 15, por tanto, la peña 7 la metemos en el grupo de las peñas 10 y 15. Por otro lado, la peña 1, coincide, en el mismo grupo, con las peñas $\{2, 3, 5\}$ y $\{4, 6, 11, 12\}$ el 75% de los cuantiles, por tanto, podría pertenecer a cualquiera de estos dos grupos. Por consiguiente, tenemos que comprobar estas dos opciones para ver cuál es la clasificación con menor distancia.

■ Opción 1:

Clasificamos la peña 1 en el grupo con las peñas 2, 3 y 5, y tenemos los siguientes grupos $\{1, 2, 3, 5\}$, $\{4, 6, 11, 12\}$, $\{8, 9, 13, 14\}$ y $\{7, 10, 15\}$. Como tenemos que realizar tres grupos, tenemos que juntar dos de estos grupos. Y tenemos que el grupo $\{1, 2, 3, 5\}$ coincide en el 50% de los cuantiles con el grupo $\{4, 6, 11, 12\}$. El grupo $\{4, 6, 11, 12\}$ coincide el 50% de los cuantiles con el grupo $\{8, 9, 13, 14\}$. Y el grupo $\{8, 9, 13, 14\}$ coincide el 50% de los cuantiles con las peñas 10 y 15. Por lo que tenemos las siguientes posibles clasificaciones:

- $B_1 : \langle \{1, 2, 3, 4, 5, 6, 11, 12\}, \{8, 9, 13, 14\}, \{7, 10, 15\} \rangle$
- $B_2 : \langle \{1, 2, 3, 5\}, \{4, 6, 8, 9, 11, 12, 13, 14\}, \{7, 10, 15\} \rangle$
- $B_3 : \langle \{1, 2, 3, 5\}, \{4, 6, 11, 12\}, \{7, 8, 9, 10, 13, 14, 15\} \rangle$

■ Opción 2:

Clasificamos la peña 1 en el grupo con las peñas 4, 6, 11 y 12, y tenemos los siguientes grupos $\{2, 3, 5\}$, $\{1, 4, 6, 11, 12\}$, $\{8, 9, 13, 14\}$ y $\{7, 10, 15\}$. Como en el caso anterior, al tener que realizar tres grupos, tenemos que juntar dos de estos grupos. Y tenemos que el grupo $\{2, 3, 5\}$ coincide en el 50% de los cuantiles con el grupo $\{1, 4, 6, 11, 12\}$. El grupo $\{1, 4, 6, 11, 12\}$ coincide el 50% de los cuantiles con el grupo $\{8, 9, 13, 14\}$. Y el grupo $\{8, 9, 13, 14\}$ coincide el 50% de los cuantiles con las peñas 10 y 15. Por lo que tenemos las siguientes posibles clasificaciones:

- La primera agrupación es la misma agrupación que B_1 .
- $B_4 : \langle \{2, 3, 5\}, \{1, 4, 6, 8, 9, 11, 12, 13, 14\}, \{7, 10, 15\} \rangle$
- $B_5 : \langle \{2, 3, 5\}, \{1, 4, 6, 11, 12\}, \{7, 8, 9, 10, 13, 14, 15\} \rangle$

Habiendo obtenido los distintos ordenes de cubos posibles, hemos programado la siguiente función para calcular la distancia.

La función recibe un parámetro denominado *OrdenacionFinal*, el cual es el vector que indica en que grupo está clasificada cada peña. A partir de ese vector con el bucle lo que hacemos es calcular la matriz C^B y ,usando la fórmula 3.2, calcula la distancia.

```
> distancia<-function(OrdenacionFinal){
+   CB<-matrix(c(NA), nrow = n,ncol = n,byrow = TRUE)
+   for (i in 1:n){
+     for (j in 1:n){
+       if (OrdenacionFinal[i]==OrdenacionFinal[j]){
+         CB[i,j]=0.5
+       }else{
+         if(OrdenacionFinal[i]<=OrdenacionFinal[j]){
+           CB[i,j]=1
+         }else{
+           CB[i,j]=0
+         }
+       }
+     }
+   }
+   colnames(CB)<-colnames(datos)
+   rownames(CB)<-colnames(datos)
+   d<-sum(abs(C-CB))
+   return(distancia=d)
+ }
```

El siguiente paso es crear los vectores con los diferentes grupos y llamar a la función. Mediante el siguiente script lo que hacemos es crear el vector del orden de

cubos B_1 , llamamos a la función distancia pasándole el vector y lo guardamos en la variable $d1$.

```
> #B_1
> Orden1<-c(1,1,1,1,1,1,3,2,2,3,1,1,2,2,3)
> names(Orden1)<-colnames(datos)
> d1<-distancia(Orden1)
> d1
```

```
[1] 33.53333
```

Realizamos lo mismo para el resto de ordenes de cubos.

```
> #B_2
> Orden2<-c(1,1,1,2,1,2,3,2,2,3,2,2,2,2,3)
> names(Orden2)<-colnames(datos)
> d2<-distancia(Orden2)
> d2
```

```
[1] 34.6
```

```
> #B_3
> Orden3<-c(1,1,1,2,1,2,3,3,3,3,2,2,3,3,3)
> names(Orden3)<-colnames(datos)
> d3<-distancia(Orden3)
> d3
```

```
[1] 30.86667
```

```
> #B_4
> Orden4<-c(2,1,1,2,1,2,3,2,2,3,2,2,2,2,3)
> names(Orden4)<-colnames(datos)
> d4<-distancia(Orden4)
> d4
```

[1] 37.2

```
> #B_5
> Orden5<-c(2,1,1,2,1,2,3,3,3,3,2,2,3,3,3)
> names(Orden5)<-colnames(datos)
> d5<-distancia(Orden5)
> d5
```

[1] 30.53333

Cuadro 4.3: Resultados finales

Orden de cubos	Distancia
$B_1 : \langle \{1, 2, 3, 4, 5, 6, 11, 12\}, \{8, 9, 13, 14\}, \{7, 10, 15\} \rangle$	33.53
$B_2 : \langle \{1, 2, 3, 5\}, \{4, 6, 8, 9, 11, 12, 13, 14\}, \{7, 10, 15\} \rangle$	34.60
$B_3 : \langle \{1, 2, 3, 5\}, \{4, 6, 11, 12\}, \{7, 8, 9, 10, 13, 14, 15\} \rangle$	30.87
$B_4 : \langle \{2, 3, 5\}, \{1, 4, 6, 8, 9, 11, 12, 13, 14\}, \{7, 10, 15\} \rangle$	37.20
$B_5 : \langle \{2, 3, 5\}, \{1, 4, 6, 11, 12\}, \{7, 8, 9, 10, 13, 14, 15\} \rangle$	30.53

Observando el Cuadro 4.3 comprobamos que la mejor agrupación es B_5 , ya que tiene la menor distancia. Aunque, la agrupación B_3 es prácticamente igual de válida, puesto que la distancia es muy similar.

4.2.4. Nueva Etapa 3

En la Etapa 3 hemos comprobado lo molesto que puede llegar a ser realizar dicha etapa “a ojo”. Ante la dificultad que conlleva, en una base de datos con únicamente 15 peñas, realizar todas las posibles combinaciones de ordenes de cubos y, de esta manera, poder comparar las distintas clasificaciones mediante la fórmula de la distancia, en este apartado vamos a realizar una modificación de la Etapa 3 original que vamos a denominar Nueva Etapa 3. El fin de esta Nueva Etapa 3 es conseguir un orden de cubos final de manera más rápida y sencilla, y que además nos permita usar dicho algoritmo en bases de datos más grandes.

Esta Nueva Etapa 3 consiste en, a partir de los grupos obtenidos en la Etapa 2, conseguir el orden de cubos final mediante el algoritmo de las k-medias.

Como hemos visto en la introducción el algoritmo de la k-medias se basa en minimizar la diferencia entre el centroide y los elementos que forman el grupo y maximizar la distancia con elementos de otros grupos. Esto es muy similar al algoritmo que hemos propuesto, puesto que, en nuestro algoritmo, decimos que si un elemento i precede a un elemento j , entonces la probabilidad de precedencia del elemento i con respecto al elemento j tiene que ser muy parecida a 1 (ya que queremos minimizar la distancia). En caso de pertenecer al mismo grupo, la probabilidad de precedencia tiene que ser parecida a 0.5 y en caso de que el elemento j preceda al i , parecida a 0. Estos valores pueden verse en nuestro algoritmo como una especie de “centroides”. Entonces, mediante las distintas agrupaciones que probamos, lo que vamos haciendo es probar distintos “centroides” y quedarnos con los “centroides” que minimizan la distancia. Para minimizar la distancia, lo que hacemos es minimizar la diferencia entre el “centroide” y los elementos del grupo.

El problema del algoritmo de las k-medias, es que da como resultado los distintos grupos a hacer pero no nos indica el orden de precedencia de estos grupos. Por tanto, a partir de los grupos obtenidos mediante la k-medias, lo que vamos a hacer es ordenar estos grupos mediante sus rangos medios y, así, obtener el orden de cubos.

Mediante la función `kmeans` implementada en R, conseguimos los distintos grupos. En nuestro caso, la sentencia R para que realice el algoritmo es la siguiente:

```
> res<-kmeans(t(datos),num_grupos)
```

Realizamos la traspuesta de nuestra base de datos porque el algoritmo realiza los grupos por observaciones (filas) y nosotros queremos realizar los grupos por columnas.

Habiendo guardado en la variable `res` el resultado del algoritmo de las k-medias, tenemos que ordenar estos grupos. Para ordenar estos grupos calculamos el rango medio de cada peña. Mediante el primer bucle lo que vamos haciendo es: para cada grupo i , guardamos en la variable `minRangosGrupos` la peña con el menor rango medio que pertenezca a dicho grupo. Es decir, obtenemos la peña con el rango más bajo de cada grupo. Ordenamos estos rangos y, mediante el último bucle, comprobamos si la peña i está en el mismo grupo que la peña con el rango más pequeño del

grupo. En caso de estar en el mismo grupo, le asignamos la posición de esa peña. Como la peñas con el rango más bajo están ordenadas de forma ascendente, la posición de cada una de esas peñas marca el grupo en el que están en el orden de cubos final.

```
> rangosMedio<-apply(rangos, 2, mean)
> minRangosGrupos<-c()
> OrdenacionFinal<-c(rep(NA,n))
> for (i in 1:num_grupos){
+   minRangosGrupos<-c(minRangosGrupos,rangosMedio[rangosMedio==min(
+     rangosMedio[names(res$cluster[res$cluster==i])]))
+ }
> minRangosGrupos<-sort(minRangosGrupos)
> for(i in 1:n){
+   for (j in 1:num_grupos){
+     if(res$cluster[i]==res$cluster[names(minRangosGrupos[j])]){
+       OrdenacionFinal[i]=j
+     }
+   }
+ }
```

Teniendo el orden de cubos final calculamos la distancia mediante la función *distancia*.

Como hemos comentado en los preliminares, dependiendo de la agrupación inicial por la que empiece el algoritmo de las k-medias, pueden darse diferentes soluciones. Para solventar este problema, ejecutamos varias veces el algoritmo y nos quedamos con la agrupación de menor distancia.

Calculando varias veces las k-medias, tenemos que nos da dos posibles ordenes de cubos. En el Cuadro 4.4 vemos estos dos ordenes de cubos.

Cuadro 4.4: Resultados k-medias

Orden de cubos	Distancia
$B_1 : \langle \{2, 3, 5\}, \{1, 4, 6, 11, 12\}, \{7, 8, 9, 10, 13, 14, 15\} \rangle$	30.53
$B_2 : \langle \{1, 2, 3, 5, 6\}, \{4, 9, 11, 12, 13\}, \{7, 8, 10, 14, 15\} \rangle$	30.67

Podemos observar que el primer orden de cubos es el mismo que hemos seleccionado en la Etapa 3 y que el otro orden de cubos que nos da tiene una distancia muy similar.

Como conclusión esta forma de agregación de rankings nos da unos resultados muy buenos y nos permite usar este método para grandes base de datos.

Con el fin de simplificar el algoritmo, hemos creado una función que recibe únicamente, como parámetro de entrada, los datos y da como salida el orden de cubos final y la distancia.

```

> GAP<-function(datos){
+   #Montamos la matriz C
+   n<-dim(datos)[2] #número de objetos
+   m<-dim(datos)[1] #número de clasificaciones totales
+
+   C<-matrix(0,nrow = m, ncol = n)
+   colnames(C)<-colnames(datos)
+   rownames(C)<-colnames(datos)
+   for (i in 1:n){
+     for (j in 1:n){
+       cuenta=0
+       for (t in 1:m){
+         if (datos[t,i]<datos[t,j]){
+           cuenta=cuenta+1
+         }
+         C[i,j]=cuenta
+       }
+     }
+   }
+   C=C/m
+   diag(C)=0.5
+
+   #Etapa 1
+   q=c(0.3,0.5,0.7,0.9)

```

4.2. ALGORITMO DE LA AGREGACIÓN DE CUBOS PARA LOS CABALLOS DEL VINO41

```
+ k=length(q)
+ rangos<-matrix(0, nrow = k, ncol = n)
+ colnames(rangos)<-colnames(datos)
+ for (i in 1:n) {
+   for (j in 1:k){
+     #obtenemos los [m*q] primeros numeros ordenados de forma ascendente
+     cabeza<-head(datos[order(datos[,i], decreasing = FALSE), ],
+                 ceiling(m*q[j]))
+     #seleccionamos el maximo (de esta forma calculamos el rango)
+     rango<-max(cabeza[i])
+     #añadimos el rango a la matriz de rangos cuantílicos
+     rangos[j,i]<-rango
+   }
+ }
+ for (i in 1:k){
+   assign(paste("Q",i,sep=""), sort(rangos[i,]))
+ }
+
+ #Etapa 2
+
+ #Calculamos la diferencia entre rangos
+ g1<-c()
+ g2<-c()
+ g3<-c()
+ g4<-c()
+ for (i in 1:n-1) {
+   g1<-c(g1,abs(Q1[i]-Q1[i+1]))
+   g2<-c(g2,abs(Q2[i]-Q2[i+1]))
+   g3<-c(g3,abs(Q3[i]-Q3[i+1]))
+   g4<-c(g4,abs(Q4[i]-Q4[i+1]))
+ }
+
+ #Calculamos la media
+ mu1=sum(g1)/(n-1)
```

```

+ mu2=sum(g2)/(n-1)
+ mu3=sum(g3)/(n-1)
+ mu4=sum(g4)/(n-1)
+
+ #Calculamos la desviacion tipica
+ dt1<-sqrt(sum((g1-mu1)^2)/(n-1))
+ dt2<-sqrt(sum((g2-mu2)^2)/(n-1))
+ dt3<-sqrt(sum((g3-mu3)^2)/(n-1))
+ dt4<-sqrt(sum((g4-mu4)^2)/(n-1))
+
+ #Calculamos la diferencia anormal entre rangos
+ N=c(rep(0,k))
+ dif1<-mu1+dt1
+ dif2<-mu2+dt2
+ dif3<-mu3+dt3
+ dif4<-mu4+dt4
+ for (i in 1:length(g1)) {
+   if (g1[i]>dif1){
+     N[1]=N[1]+1
+   }
+   if (g2[i]>dif2){
+     N[2]=N[2]+1
+   }
+   if (g3[i]>dif3){
+     N[3]=N[3]+1
+   }
+   if (g4[i]>dif4){
+     N[4]=N[4]+1
+   }
+ }
+ names(N)<-c("g1", "g2", "g3", "g4")
+ N<-sort(N)
+
+ #Calculamos la mediana

```

4.2. ALGORITMO DE LA AGREGACIÓN DE CUBOS PARA LOS CABALLOS DEL VINO43

```
+ if(length(N)%2!=0) {
+   Na=N[(k+1)/2]
+ }else {
+   Na=N[(k/2)+1]
+ }
+ names(Na)<-c()
+ num_grupos<-Na+1
+
+ #Nueva Etapa 3
+ nombre<-colnames(datos)
+ res<-kmeans(t(datos),num_grupos)
+ rangosMedio<-apply(rangos, 2, mean)
+ minRangosGrupos<-c()
+ OrdenacionFinal<-c(rep(NA,n))
+ for (i in 1:num_grupos){
+   minRangosGrupos<-c(minRangosGrupos,rangosMedio[rangosMedio==min(
+     rangosMedio[names(res$cluster[res$cluster==i])])])
+ }
+ minRangosGrupos<-sort(minRangosGrupos)
+ for(i in 1:n){
+   for (j in 1:num_grupos){
+     if(res$cluster[i]==res$cluster[names(minRangosGrupos[j])]){
+       OrdenacionFinal[i]=j
+     }
+   }
+ }
+ names(OrdenacionFinal)<-nombre
+ CB<-matrix(c(NA), nrow = n,ncol = n,byrow = TRUE)
+ for (i in 1:n){
+   for (j in 1:n){
+     if (OrdenacionFinal[i]==OrdenacionFinal[j]){
+       CB[i,j]=0.5
+     }else{
+       if(OrdenacionFinal[i]<=OrdenacionFinal[j]){
```

```

+         CB[i,j]=1
+     }else{
+         CB[i,j]=0
+     }
+ }
+ }
+ }
+ }
+   colnames(CB)<-nombre
+   rownames(CB)<-nombre
+   OrdenacionFinal<-sort(OrdenacionFinal)
+   d<-sum(abs(C-CB))
+   return(list(agrupacion=OrdenacionFinal,distancia=d))
+ }

```

Con el siguiente código hacemos la llamada a la función y guardamos los resultados en *grupos*.

```
> grupos<-GAP(datos)
```

Como hemos dicho anteriormente, el resultado depende de los grupos iniciales que realice el algoritmo de la k-medias. Para solucionar este problema, ejecutamos varias veces la función y nos quedamos con la agrupación con la distancia más pequeña. Para hacer esto de forma sencilla podemos utilizar el siguiente código, el cual ejecuta diez veces la función GAP y guarda en la variable *grupos* el orden de cubos que tiene la distancia más pequeña.

```

> grupos<-GAP(datos)
> for (i in 1:10){
+   grupos1<-GAP(datos)
+   if (grupos1$distancia<grupos$distancia){
+     grupos<-grupos1
+   }
+ }

```

Para finalizar este capítulo observamos el resultado obtenido.

4.2. ALGORITMO DE LA AGREGACIÓN DE CUBOS PARA LOS CABALLOS DEL VINO45

> *grupos*

\$agrupacion

CAMPEON	UNIVERSO	CAPRICHOSO	TERRY	SANGRINO
1	1	1	2	2
MEL.AZULES	AMBICIOSO	ZAMBRA SOLTERON.TRIANA		JUPITER
2	2	2	3	3
MINIPUA	CARTUJANO	ESPARTACO	JEQUE	LUMINOSO
3	3	3	3	3

\$distancia

[1] 30.53333



Capítulo 5

Conclusiones

Mediante esta memoria, hemos introducido una heurística que nos permite realizar agregaciones de rankings. Además, hemos creado una función en el lenguaje de programación R, que nos permite resolver el problema de agregación de rankings de manera rápida y sencilla.

En el trabajo hemos comenzado introduciendo una técnica de agrupación de datos, la heurística de las k -medias. Esta heurística nos permite hacer agrupaciones de objetos en k grupos predefinidos inicialmente.

Tras realizar una pequeña introducción, hemos explicado la heurística GAP, la cual nos permite realizar agrupaciones usando la *diferencia anormal entre rangos* como medida para separar los distintos grupos.

Para finalizar, hemos desarrollado un algoritmo en el lenguaje de programación R, que nos permite resolver el problema de agregación de rankings. Para realizar este algoritmo hemos utilizado las dos primeras etapas de la heurística GAP y, cuando hemos obtenido el número de grupos, una Nueva Etapa 3, en la que hemos aplicado la heurística de las k -medias.

Este algoritmo nos permite resolver el problema de agregación de rankings para cualquier tamaño de base de datos de forma rápida.

Bibliografía

- [1] FENG, J., FANG, Q. y NG, W., *Discovering Bucket Orders from Full Rankings*, en Proceedings of the 2008 ACM SIGMOD international conference on Management of data (2008) 55-66.
- [2] GARCÍA-NOVÉ, E.M., ALCARAZ, J., LANDETE, M. y PMONGE, J.F., *The Generalized Linear Ordering Problem: A new partial ranking* (2018)
- [3] MINERÍA DE DATOS
https://es.wikipedia.org/wiki/Minería_de_datos
- [4] K-MEDIAS
https://www.unioviado.es/compnun/laboratorios_py/kmeans/kmeans.html
- [5] CABALLOS DEL VINO
<http://caballosdelvino.org>
- [6] L^AT_EX
<http://metodos.fam.cie.uva.es/~latex/curso-2015/apuntes3.pdf>
<https://es.sharelatex.com/learn/>
- [7] RSWEAVE
https://cran.r-project.org/doc/contrib/Rivera-Tutorial_Sweave.pdf