UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

MÈTODES I MODELS PER A L'ANÀLISI DE CLÚSTERS

Autor: Marc Morera de Frutos

Director: Dr. Josep Fortiana
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 24 de gener de 2021

Abstract

In this project we study clustering methods, focusing on the two most used classes of algorithms: partitional and hierarchical. We explore definition and performance of several methods, as well as some criteria to validate results. Then, we assess computational problems arising from adapting generic clustering methods to tackle large data sets, as well as algorithms designed with the goal of improving their performance in these cases. We include R code to demonstrate practical applications of clustering and to generate graphical visualizations.

Resum

En aquest treball estudiem els mètodes de clústering, centrant-nos en les dues classes d'algorismes més usats: els basats en particions i els jeràrquics. Analitzem la definició i rendiment de diversos mètodes, així com alguns criteris per validar resultats. Llavors, avaluem els problemes computacionals que sorgeixen d'adaptar mètodes generics de clústering per abordar problemes amb grans dades, així com alguns algorismes dissenyats per millorar el seu rendiment en aquests casos. Adjuntem codi en R per demostrar aplicacions pràctiques del clústering i generar visualitzacions gràfiques.

2010 Mathematics Subject Classification. 62H30, 62P99

Agraïments

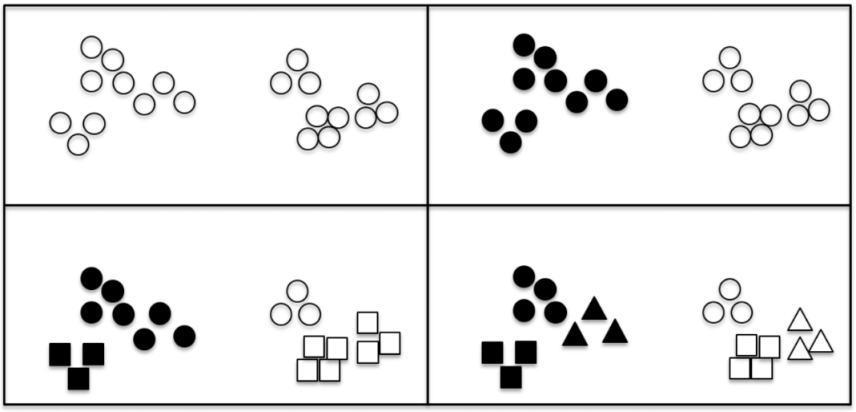
Vull agrair en primer lloc al Dr. Josep Fortiana per accedir a tutoritzar aquest treball, per guiar-me i aconsellar-me en la realització d'aquest, i per ajudar-me a polir els detalls fins a l'últim moment.

També vull expressar la meva gratitud a la meva família i amics, en especial als meus pares, que m'han recolzat al llarg de tota la carrera.

Índex

1 Introducció al clústering

L'anàlisi de dades és un conjunt de processos molt diversos àmpliament usats en una gran varietat de disciplines científiques i tècniques, com són diversos camps de la biologia, psicologia, medicina, màrqueting, informàtica i moltes d'altres, amb l'objectiu d'aconseguir informació útil de certs resultats o dades subjectes d'estudi. A l'hora d'analitzar dades, una de les preguntes més senzilles que ens podem fer és si existeix una classificació o agrupació dels objectes que volem estudiar. La pràctica de classificar objectes en funció de certes similituds entre ells és la base de bona part de la ciència. Organitzar un conjunt de dades en una sèrie de grups amb sentit és un dels mètodes fonamentals per entendre i aprendre del que tenim. D'aquesta necessitat sorgeix l'anàlisi de clústers, que podem definir com l'estudi formal dels mètodes i algorismes per agrupar i classificar objectes (o dades, en general). L'objectiu d'aquests algorismes és trobar algun tipus d'estructuració en les dades, ja que, tot i que podria semblar que agrupar objectes en grups és una tasca simple a primera vista, en realitat té certa complexitat, començant pel fet de que no és senzill saber quants grups, o clústers, com els anomenarem, hi ha en un conjunt de dades.



M. Embrechts et al, 2013, Fig. 2. Exemple d'un conjunt d'objectes (imatge superior esquerra) amb diferents possibles particions, en 2, 4 i 6 clústers respectivament.

Algunes motivacions de l'anàlisi de clústers inclouen l'exploració de dades, la divisió de conjunts grans de dades en subconjunts més manipulables, una bona visualització de les dades, detecció d'outliers o valors atípics, segmentació d'imatges, a més de moltes altres aplicacions específiques, com la classificació de gens en bioinformàtica o cerques de similituds en imatges mèdiques.

1.1 Clústers i clústerings

Un clúster és un subconjunt de les dades, format per elements similars entre ells d'alguna forma. Brian S. Everitt, al seu article de 1974, en dona diverses definicions, com: "Un clúster és un conjunt d'entitats considerades similars, i entitats de diferents clústers no són similars." o "Els clústers poden ser descrits com regions connectades d'un espai multidimensional que contenen una densitat relativament alta de punts, separades d'altres regions del mateix tipus per una regió amb una densitat relativament baixa de punts.".

Anomenarem clústering a un conjunt de clústers, i un clústering és el que obtindrem com a resultat d'aplicar un algorisme de classificació.

Podem formalitzar matemàticament els conceptes de clúster i clústering de la forma següent, seguint les idees del capítol de Cuadras:

Sigui $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$ un conjunt finit, amb n elements diferents, que indicarem de forma abreujada com $\Omega = \{1, 2, ..., n\}$. Per classificar els elements de Ω, podem definir una relació d'equivalència $\mathcal{R}$ sobre Ω. Aquesta relació ens defineix una partició sobre Ω en m classes d'equivalència, i tenim:

$$\Omega = c_1 + c_2 + ... + c_m$$

on + fa referència a la reunió disjunta de conjunts. Llavors anomenarem clústers a les classes d'equivalència $c_1, c_2, ..., c_m$; i clústering a la partició de Ω que hem definit.

1.2 Tipus de clústerings

Els algorismes de clústering, i els mateixos clústerings resultants, es poden classificar de forma general en dues classes. Diem que tenim un clústering "dur" (*hard*) quan cada objecte o dada pertany de forma exclusiva a un únic clúster. Per la contra, direm que un clústering és "tou" o difús ($soft/fuzzy$) si cada objecte pertany a cada clúster en una certa mesura. L'estudi de conjunts on els elements tenen una pertinença parcial a cada conjunt dóna lloc a l'anomenada lògica difusa, de gran interès en certs camps de la informàtica, particularment en sistemes de control i intel·ligència artificial, però nosaltres ens centrarem en les particions dures, que són les que ens interessen per a l'anàlisi de dades.

Per altra banda, podem classificar els clústerings en funció de l'enfocament que prenem per calcular-los, que pot donar lloc a algorismes molt diferents. Les dues formes més conegudes i usades són els clústerings amb particions, o particionals (de l'anglès *partitional clustering*), i els clústerings jeràrquics, però existeixen altres mètodes, basats en nuclis (*kernel-based*), en dades seqüencials o sistemes artificials. Centrarem el nostre estudi en els dos primers, els clàssics, ja que la gran majoria d'algorismes de classificació estan basats en aquests.

La idea dels algorismes amb particions és definir una quantitat K de clústers que volem trobar abans de començar la classificació, i aplicar un procés iteratiu, que va refinant el clústering, per dividir les dades en K particions adequades. Els algorismes jeràrquics, en canvi, no requereixen especificar el nombre de clústers que volem. El que fan es classificar ordenadament les observacions en agrupacions successives, de forma que obtenim una 'jerarquia'. Aquests algorismes tenen l'avantatge afegit que el resultat és un gràfic en forma d'arbre de les agrupacions d'observacions, que facilita la interpretació visual dels resultats. A aquest gràfic l'anomenem dendrograma. Veurem en profunditat els mètodes particionals i jeràrquics de clústering al següents capítols.

2 Clústering amb particions

Com hem començat definint anteriorment, un clústering amb particions és aquell en que dividim les dades en un nombre predefinit, al que usualment anomenem K, de clústers diferents i disjunts. L'estructura típica dels algorismes per a aquest tipus de clústerings és prendre una classificació inicial i anar-la refinant mitjançant un procés iteratiu, fins que aconseguim una classificació prou bona. El més conegut dels algorismes d'aquest tipus és l'algorisme k-mitjanes, que veiem tot seguit.

2.1 Plantejament del mètode K-mitjanes

Per plantejar el problema a resoldre amb el mètode de k-mitjanes i el conseqüent algosrisme ens basarem en les explicacions de G. James et al. (2013), pp.385-390.

Seguint l'estructura definida, l'algorisme K-mitjanes ens permet dividir un conjunt de dades en K clústers diferents i disjunts. Primer cal especificar la quantitat K desitjada de clústers, i llavors l'algorisme va assignant cada objecte a un (i només un) dels K clústers. Seguint la notació anterior, anomenem $C_1, ..., C_K$ als clústers resultants. Citant a James: 'La idea de l'algorisme K-mitjanes és que un clústering *bo* és aquell per al qual la variació interna ($within - cluster\ variation$) és tan petita com sigui possible'. La variació interna d'un clúster C_K és una mesura de com de diferents són les observacions dins d'aquest mateix clúster, i la podem indicar com $W(C_K)$. Llavors, el problema que vol resoldre l'algorisme K-mitjanes és el de minimitzar aquests valors, és a dir, volem trobar:

$$\min_{C_1,...,C_K} \{\sum_{k=1}^{K} W(C_k)\}$$

Per poder realitzar aquest càlcul, hem de definir la variació interna dels clústers. Aquesta es pot definir de moltes formes, en funció del problema a tractar, però la més típica és la distància euclidiana al quadrat. Definim:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

on $|C_k|$ denota el nombre d'observacions al clúster k, i p indica la dimensió de l'espai de les observacions. Així, tenim que la variació interna és la suma de les distàncies euclidianes al quadrat per a cada parella d'observacions al clúster k, dividit entre el total d'observacions dins de l'esmentat clúster. Combinant les dues darreres equacions, tenim que la resolució del clústering K-mitjanes ve donada pel següent problema d'optimització:

$$\min_{C_1,...,C_K} \{\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\}$$

2.2 Algorisme K-mitjanes

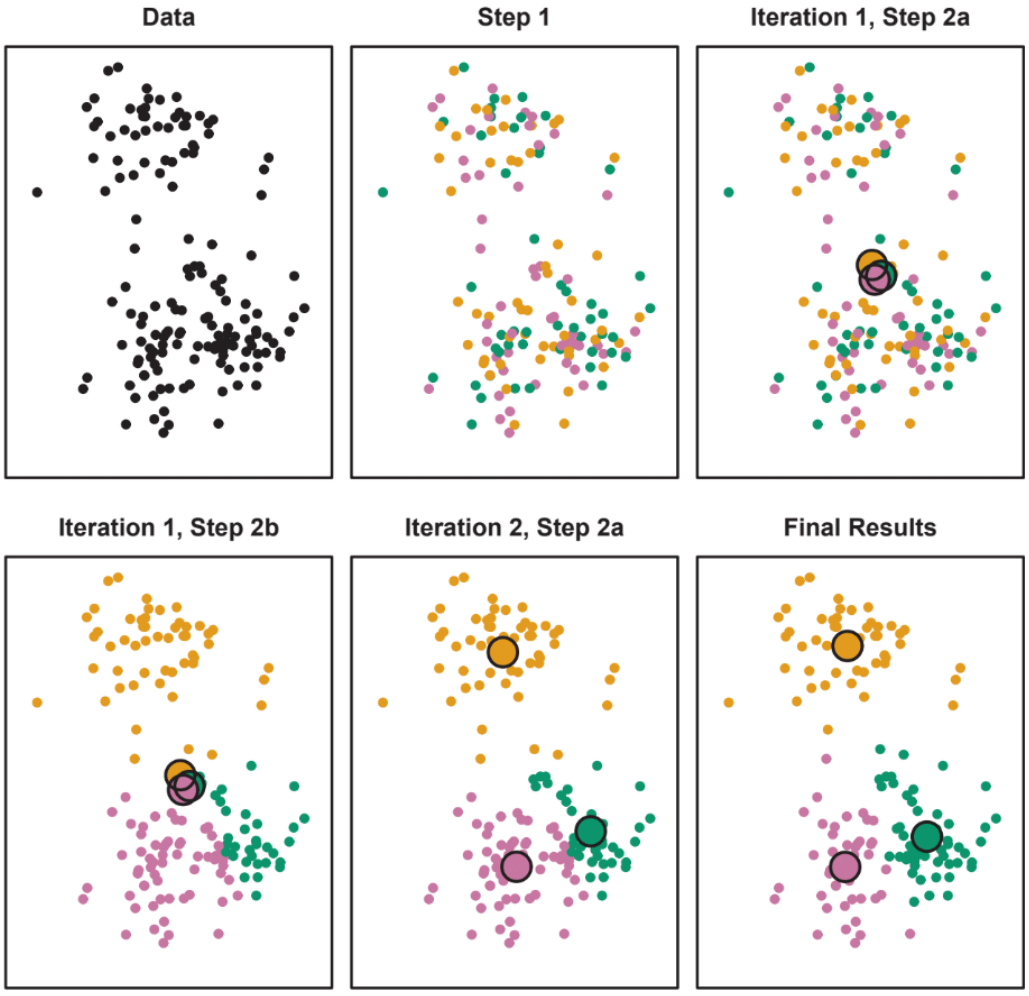
Trobar un algorisme per resoldre aquest problema pot semblar complicat, ja que hi ha prop de K^n formes de dividir n observacions en k clústers, però existeixen algorismes molt simples que funcionen molt bé localment, com el que s'exposa a continuació:

1. Per començar, assignem cada observació a un clúster aleatori, tot assignant un valor aleatori entre 1 i K.

2. Iterem fins convergència:

   (a) Computem el *centroide* corresponent a cadascun dels K clústers, tot calculant la mitjana entre totes les observacions del clúster, per a cadascuna de les p "coordenades" (el *centroide* és el vector format per aquestes mitjanes).

   (b) Assignem cada observació al clúster més proper, que serà aquell amb la menor distància entre el seu *centroide* i la observació en qüestió.



G. James et al, 2013, Fig. 10.6. Visualització gràfica dels passos de l'algorisme amb un exemple on prenem $k = 3$. Les dades inicials (primera imatge) són assignades a un clúster aleatori, indicat pel color (segona imatge). Es calculen els centroides de cada clúster i es reassignen les observacions al clúster amb centroide més proper (tercera i quarta imatge). Després de 10 iteracions, s'arriba al resultat final (sisena imatge).

Es pot demostrar que aquest algorisme convergeix, i que a més el valor a optimitzar decreix a cada iteració (es pot trobar al capítol de G. James et al., i en veurem una

petita demostració més endavant). Com hem dit, l'algorisme plantejat d'aquesta forma funciona de forma local, ja que els clústers resultants dependran de l'assignació inicial de clústers, que és aleatòria. Usualment el que es fa és aplicar l'algorisme diverses vegades, amb diferents clústers inicials, i llavors ens quedem amb el millor resultat, és a dir, aquell per al qual la suma de les variacions internes dels clústers és més petita.

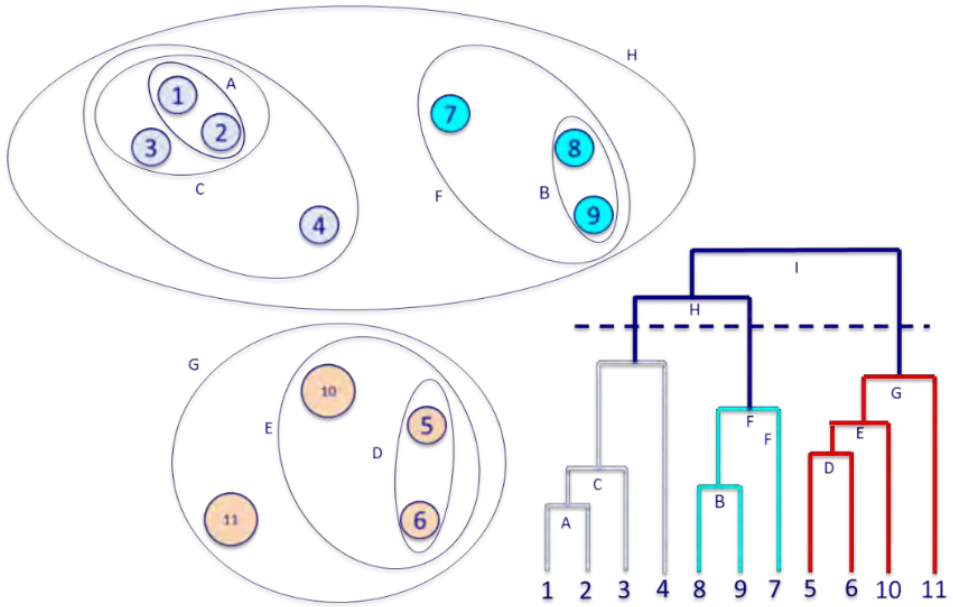
Els algorismes de K-mitjanes, tot i ser simples i efectius, presenten el problema de l'elecció inicial de K, ja que depén completament de les dades a tractar, i en alguns casos es tracta d'un problema prou complex.

2.3 Algorismes de tipus K-medoides

Un altre tipus d'algorisme de clústering particional destacable són els K-medoides. Aquests funcionen de forma molt similar al K-mitjanes. La principal diferència entre els dos radica en el fet que en el K-medoides es fan servir com a centres observacions reals del conjunt de dades, en comptes d'un càlcul mitjà, com és el centroide, que no es correspon amb cap punt del clúster en qüestó. Un *medoide* és un dels elements del clúster que actúa com a representant d'aquest, fet que permet una interpretació més "real" del centre dels clústers. Com a conseqüència d'això, els algorismes de K-medoides són menys sensibles als valors atípics (*outliers*) que els K-mitjanes, éssent la contrapartida que els primers tenen un cost computacional major. Alguns dels algorismes més coneguts d'aquesta classe són l'anomenat PAM (de *Partitioning Around Medoids*), i el CLARA (de *Clustering Large Applications*), que és una adaptació del PAM per a conjunts grans de dades, tots dos proposats per Kaufman i Rousseeuw a 1990.

3 Clústering jeràrquic

En aquest capítol veurem amb detall el funcionament i la formalització dels algorismes de tipus jeràrquic, i així com diversos mètodes per aplicar-lo. Com hem mencionat anteriorment, un dels atractius principals d'aquest tipus de mètode de classificació és el fet que obtenim com a resultat un gràfic en forma d'arbre que classifica les observacions: un dendrograma.



M. Embrechts et al, 2013, Fig. 6. Il·lustració del clústering jeràrquic d'un conjunt de dades, i del dendrograma associat.

En aquests dendrogrames, les fulles de l'arbre representen les observacions a classificar. Conforme anem pujant a l'arbre, les fulles es comencen a fusionar en branques, que es corresponen amb conjunts d'observacions similars entre elles. Així, si seguim pujant, les branques es van fusionant amb altres branques fins que arribem a l'arrel. La forma d'interpretar el gràfic és que com abans es fusionin dues observacions (comptant desde la part inferior), major és la similitud entre elles, i dues observacions que s'agrupin tard, podrien ser molt diferents. La idea general és que l'alçada del punt on s'uneixen dues observacions és una mesura de la seva semblança.

Per identificar els clústers en un dendrograma, el que fem és realitzar un tall horitzontal al gràfic, i els sub-arbres que queden per sota del tall es poden interpretar com a clústers, on els elements continguts a cada clúster són les fulles corresponents a cada sub-arbre. Podem realitzar aquest tall a qualsevol alçada, en funció del nombre de clústers que volguem. Donada la naturalesa visual del dendrograma, en molts casos es pot identificar amb facilitat un tall que dona lloc a una classificació prou bona.

Un exemple històric important de classificació jeràrquica són les classificacions de Linneo, que en el seu llibre "Systema Naturae" (1735) va proposar una forma d'organitzar i classificar els èssers vius en funció de les similituds entre ells, creant una sèrie de nivells

d'agrupació (com són, en el cas dels animals: regne, classe, ordre, família, gènere i espècie, de general a específic), que a més alt corresponen a un nombre major d'espècies, però les similituds entre elles són més vagues. La representació gràfica d'aquestes classificacions ve donada per un dendrograma, on a les fulles i a les branques inferiors tenim els grups més concrets (espècie, gènere), i a les branques superiors els grups més generals (classe, regne), arribant a l'arrel, que correspondria al conjunt de tots els èssers vius. Tot i que la seva classificació distava de ser perfecta, i amb el temps s'han anat afegint més nivells a l'arbre de classificació per tal d'acomodar a totes les espècies descobertes, Linneo va ser el primer en fer una classificació d'aquest tipus, marcant el camí a seguir per a la classificació biològica.

Els algorismes jeràrquics de clústering poden ser divisius o aglomeratius. En el cas divisiu, l'algorisme inicia amb totes les observacions en un sol clúster, que es va dividint successivament, mentre que els aglomeratius funcionen a la inversa, comencem amb cada observació per separat (podem considerar que cadascuna té un clúster propi), i es van agrupant successivament fins que obtenim un sol clúster amb totes les dades. Ens centrarem en els algorismes de tipus aglomeratiu, ja que és una forma més intuïtiva d'enfocar la classificació, i a la pràctica els algorismes divisius són rarament usats. A més, els algorismes aglomeratius bàsics tenen un ampli marge de millora, que estudiarem més endavant, ja que els algorismes jeràrquics no escalen bé en temps de computació i usen molta memòria si treballem amb conjunts grans de dades.

Com a nota, en els capítols vinents farem servir el terme distància, per un tema de comoditat de notació, per referir-nos a qualsevol funció que prengui valors positius, compleixi la propietat simètrica, y tingui zeros a la diagonal (en la matriu associada), definició usualment associada al terme dissimilaritat. Quan la distància usada sigui una mètrica (que satisfà la desigualtat triangular), distància euclidiana o ultramètrica (que satisfà la propietat ultramètrica, explicada a la secció corresponent), ho indicarem explícitament. Per tant, usarem el terme matriu de distàncies per referir-nos a matrius de dissimilaritats en general. Direm que una matriu de distàncies $N \times N$ és euclidiana si existeix un espai euclidià d'una certa dimensió D i n punts en aquest espai tals que les distàncies quadrades entre tots els punts coincideixen amb els valors de la matriu.

3.1 Algorisme SAHN

L'algorisme de classificació jeràrquica en si mateix és prou simple. L'esquema de funcionament és el que indiquem a continuació, que diversos autors anomenen algorisme SAHN (de l'anglès, *Sequential Agglomerative Hierarchical Non-overlapping*). Per aplicar aquest tipus d'algorimes, necessitem una matriu de distàncies entre les observacions, i, suposant que tenim N observacions, precisa de $N - 1$ iteracions. Els passos que segueix són els següents:

1. Per començar, assignem cada observació per separat a un clúster unitari. Inicialitzem la matriu de distàncies.
2. Iterem fins que obtenim un únic clúster amb totes les observacions:
   (a) Busquem la parella d'observacions o clústers més semblants (usant la la matriu de distàncies).
   (b) Fusionem la parella de clústers trobada, i actualitzem la matriu (hem d'afegir la similitud o distància del nou clúster a la resta).

Com podem observar, les úniques dificultats que trobem en l'aplicació d'aquest algorisme s'on l'elecció i càlcul de les distàncies, ja que si tenim un conjunt gran de dades, estarem treballant amb una matriu de distàncies molt gran, i per tant prou difícil d'actualitzar a cada iteració.

Per analitzar els problemes i avantages d'aquest algorisme, caldrà formalitzar bé els mètodes que usem per mesurar les distàncies entre clústers, així com definir bé la idea de jerarquia que volem obtenir amb l'aplicació d'aquests algorismes. Per veure això últim, introduirem a continuació els conceptes de *jerarquia indexada* i *ultramètrica*, basantnos en una sèrie de definicions i teoremes proposats per Cuadras (2014), pp.187-200. Per un motiu de brevetat ometrem les demostracions dels teoremes proposats, que es poden trobar al capítol corresponent de l'esmentat autor.

3.2 Jerarquia indexada

Comencem formalitzant el que volem obtenir quan busquem ordenar de forma jeràrquica un conjunt d'objectes. Direm que una classificació jeràrquica és una successió de clústerings tal que cada un dels clusterings s'obté agrupant clústers del seu antecessor. Per exemple, si tenim 5 observacions, a, b, c, d i e, una classificació jeràrquica és:

$$\Omega = \{a\} + \{b\} + \{c\} + \{d\} + \{e\}$$

$$\Omega = \{a, b\} + \{c\} + \{d, e\}$$

$$\Omega = \{a, b\} + \{c, d, e\}$$

$$\Omega = \Omega$$

on cada fila correspon a un dels successius clústerings.

Definim de forma precisa una jerarquia indexada (Definició 10.2.1, Cuadras, 2014):

Definició 3.1. *Una jerarquia indexada (C, α) sobre Ω està formada per una col·lecció de clústers $C \subset \wp(\Omega)$ (particions de Ω) i un índex α tal que satisfà:*

- *Axioma de la intersecció: Si $c, c' \in C$ llavors $c \cap c' \in \{c, c', \emptyset\}$*
- *Axioma de la reunió: Si $c \in C$ llavors $c = \cup\{c' \mid c' \in C, c' \subset c\}$*
- *La reunió de tots els clústers és el conjunt total: $\Omega = \cup\{c \mid c \in C\}$*

L'índex α és una aplicació de C sobre els reals positius tal que:

- $\alpha(i) = 0 \ \forall i \in \Omega$
- $\alpha(c) \leq \alpha(c')$ *si* $c \subset c'$

Amb aquesta definició, tenim una classificació jeràrquica: el primer axioma diu que dos clústers són disjunts o bé un està contingut dins de l'altre, fet que ens assegura que cada element de Ω pertany a un únic clúster, mentre que el segon ens diu que cada clúster és la unió de tots els clústers que conté, i dona lloc a la ordenació jeràrquica dels clústers. L'índex α és una mesura de l'heterogeneïtat de cada clúster: a major valor, més diversos són els elements continguts (i per tant la semblança és menor).

Direm que una jerarquia és total si satisfà:

- $\forall i \in \Omega$, $\{i\} \in C$
- $\Omega \in C$

Veiem un teorema que ens dona una propietat interessant d'aquest índex α (Teorema 10.2.1, Cuadras, 2014):

Teorema 3.2. *Per a tot $x \geq 0$, la relació binària $\mathcal{R}_x$ sobre els elements de Ω donada per*

$$i\mathcal{R}_x j \text{ si } i, j \in c, \text{ éssent } \alpha(c) \leq x,$$

és d'equivalència.

Aquesta relació ens dona una partició de Ω per a cada $x \geq 0$, és a dir, obtenim un clústering a un cert nivell x. Així formalitzem el "tall horitzontal" del dendrograma que hem esmentat anteriorment, per a valors alts de x estaríem tallant per la part superior de l'arbre, i amb valors baixos tallaríem per la part inferior.

3.3 Geometria ultramètrica

En aquesta secció veurem el funcionament general de les ultramètriques, que després relacionarem amb les jerarquies indexades i ens permetran generalitzar els nostres algorismes jeràrquics. Comencem definint el que és un espai ultramètric (Definició 10.3.1, Cuadras, 2014):

Definició 3.3. *Un espai ultramètric (Ω, u) és una estructura formada per un conjunt finit Ω i una funció distància u sobre $\Omega \times \Omega$ verificant, per a tot i, j, k de Ω:*

- *No negativitat:* $u(i, j) \geq u(i, i) = 0$
- *Simetria:* $u(i, j) = u(j, i)$
- *Propietat ultramètrica:* $u(i, j) \leq \sup\{u(i, k), u(j, k)\}$

Observem en particular que una distància ultramètrica verifica la desigualtat triangular, i per tant, és una mètrica:

$$u(i, j) \leq \sup\{u(i, k), u(j, k)\} \leq u(i, k) + u(j, k)$$

Amb aquest tipus de distància, la geometria general d'aquests espais canvia considerablement. Un cas particularment interessant és el dels triangles (Definició i Teorema 10.3.2, Cuadras, 2014).

Definició 3.4. *Un triangle $\{i, j, k\}$ format per tres elements de Ω és ultramètric si és isòsceles i la seva base és el costat més petit. És a dir, si $u(i, j)$ és la base, llavors*

$$u(i, j) \leq u(i, k) = u(j, k)$$

Teorema 3.5. *En un espai ultramètric tots els triangles són ultramètrics.*

Podem definir un dendrograma de la forma següent (Definició 10.3.3, Cuadras, 2014):

Definició 3.6. *Un arbre ultramètric (també anomenat dendrograma) és un graf connex, sense cicles, amb un punt anomenat arrel i n punts extrems equidistants de l'arrel.*

Usant la propietat de que tots els triangles són ultramètrics a un espai ultramètric, es pot demostrar que tot espai ultramètric es pot representar gràficament usant un dendrograma, donant lloc al següent teorema (Teorema 10.3.3, Cuadras, 2014):

Teorema 3.7. *Sigui (Ω, u) un espai ultramètric. Llavors, podem representar-lo mitjançant un arbre ultramètric amb els elements de Ω per extrems.*

Un altre resultat important és la "traducció" del teorema de la relació, el 3.2, per a distàncies ultramètriques (Teorema 10.3.4, Cuadras, 2014).

Teorema 3.8. *Sigui (Ω, u) un espai mètric. Si u és una distància ultramètrica, llavors la relació binària $\mathcal{R}_x$ sobre els elements de Ω*

$$i\mathcal{R}_x j \text{ si } u(i,j) \leq x,$$

és d'equivalència per a tot $x \geq 0$. Recíprocament, si dita relació és d'equivalència per a tot $x \geq 0$, llavors u és distància ultramètrica.

Per acabar amb aquesta secció, veiem una propietat que diu que al fusionar elements propers de Ω la propietat ultramètrica es conserva, i això succeeix per a qualsevol clústering de Ω (Teorema 10.3.5, Cuadras, 2014).

Teorema 3.9. *Suposem que sobre els m clústers del clústering*

$$\Omega = c_1 + c_2 + ... + c_m$$

tenim definida una distància ultramètrica u. Siguin c_i, c_j els dos clústers més propers, és a dir, $u(c_i, c_j) = mínim$. Llavors, unint c_i amb c_j, es pot definir una distància ultramètrica u' sobre els $m-1$ clústers del clústering

$$\Omega = c_1 + ... + c_i \cup c_j + ... + c_m.$$

Aquesta nova distància u' ve donada per:

$$u'(c_k, c_i \cup c_j) = u(c_i, c_k) = u(c_j, c_k),\ k \neq i, j,$$

$$u'(c_a, c_b) = u(c_a, c_b),\ a, b \neq i, j,$$

Aquesta propietat serà important per raonar l'actualització de la mesura de distància a cada iteració dels algorismes de classificació, per tal de poder comparar els clústers recentment formats amb la resta d'elements.

3.4 Equivalència entre jerarquia indexada i ultramètrica

Usant un algorisme de classificació (que Cuadras anomena *algorisme fonamental de classificació*) en un espai ultramètric podem obtenir una jerarquia indexada com a resultat. L'algorisme queda ben definit gràcies a la propietat obtinguda del teorema 3.9, de la forma següent:

Sigui (Ω, u) un espai ultramètric. Considerem la notació abreujada $\Omega = \{1, ..., m\}$.

1. Comencem prenent la partició:

$$\Omega = \{1\} + ... + \{m\}.$$

2. Siguin i, j els dos elements més propers, és a dir, $u(c_i, c_j) = mínim$. Els unim en un sol clúster:

$$\{i\} \cup \{j\} = \{i, j\}$$

Llavors definim una nova distància ultramètrica u' de la següent forma:

$$u'(k, \{i, j\}) = u(i, k) = u(j, k),\ k \neq i, j,$$

3. Considerem la nova partició:

$$\Omega = \{1\} + ... + \{i, j\} + ... + \{m\}.$$

i repetim el pas 2 fins a obtenir un únic clúster, Ω. A cada iteració, quan unim c_i amb c_j de forma que $u(c_i, c_j) = mínim$, definim l'índex:

$$\alpha(c_i \cup c_j) = u(c_i, c_j)$$

El resultat de l'aplicació d'aquest algorisme és la jerarquia indexada (C, α), on C és la col·lecció dels clústerings obtinguts a cada iteració, i α és l'índex definit al pas 3.

De fet, tot i que una jerarquia indexada sigui una estructura cojuntista, i un espai ultramètric una de geomètrica, totes dues són equivalents, com indica el següent teorema (Teorema 10.5.1, Cuadras, 2014):

Teorema 3.10. *Sigui (C, α) una jerarquia indexada total sobre un conjunt Ω. Llavors podem definir una distància ultramètrica u sobre Ω. Recíprocament, tot espai ultramètric (Ω, u) defineix una jerarquia indexada (C, α).*

Aquesta equivalència, juntament amb la propietat de la distància ultramètrica que permet definir de manera inequívoca la distancia entre un clúster i la unió de dos clústers propers ens permeten definir de forma concreta els algorismes de classificació jeràrquica. Ara bé, en general no treballem amb espais ultramètrics, així que cal veure com podem aplicar aquestes propietats al cas general.

3.5 Algorismes de classificació jeràrquica

Suposem que tenim una sèrie de n observacions, pertanyents al conjunt Ω, i que coneixem les distàncies entre cadascuna d'elles, amb les quals construïm la matriu de distàncies $\Delta = (\delta(i, j))$, de mida $n \times n$:

$$\Delta = \begin{pmatrix} \delta_{11} & \delta_{12} & ... & \delta_{1n} \\ \delta_{21} & \delta_{22} & ... & \delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1} & \delta_{n2} & ... & \delta_{nn} \end{pmatrix} \quad \delta_{ij} = \delta_{ji} = \delta(i, j), \quad \delta_{ii} = 0.$$

Si δ és una ultramètrica, llavors podem obtenir la jerarquia indexada aplicant l'algorisme anterior. En cas contrari, podem aplicar l'algorisme adaptat següent.

Sigui (Ω, δ) un espai mètric.

1. Comencem prenent la partició:

$$\Omega = \{1\} + ... + \{m\}.$$

2. Siguin i, j els dos elements més propers, és a dir, $\delta(c_i, c_j) = mínim$. Els unim en un sol clúster:

$$\{i\} \cup \{j\} = \{i, j\}$$

   Llavors definim la distància d'un element k al clúster $\{i, j\}$ de la següent forma:

$$\delta'(k, \{i, j\}) = f(\delta(i, k), \delta(j, k)),\ k \neq i, j,$$

   on f és una funció adequada.

3. Considerem la nova partició:

$$\Omega = \{1\} + ... + \{i, j\} + ... + \{m\}.$$

   i repetim el pas 2 fins a obtenir un únic clúster, Ω. A cada iteració, quan unim c_i amb c_j de forma que $u(c_i, c_j) = mínim$, definim l'índex:

$$\alpha(c_i \cup c_j) = \delta'(c_i, c_j)$$

La funció f del segon pas es defineix de forma que se satisfaci la propietat ultramètrica, ja que hem vist que aquesta ens assegura l'obtenció d'una jerarquia indexada. L'elecció d'aquesta funció f dona lloc als diversos mètodes de classificació jeràrquica.

Una observació interessant d'aquesta metodologia és que, al escollir f tal que tinguem la propietat ultramètrica, el que estem fent amb l'algorisme és deformar gradualment la matriu de dissimilaritats original per convertir-la en una de distàncies ultramètriques.

3.6 Mètode del mínim, o enllaç simple

Una primera opció, prou natural, per escollir la funció f consisteix en prendre el mínim de les distàncies possibles, és a dir:

$$\delta'(k, \{i, j\}) = \min\{\delta(i, k), \delta(j, k)\},\ k \neq i, j,$$

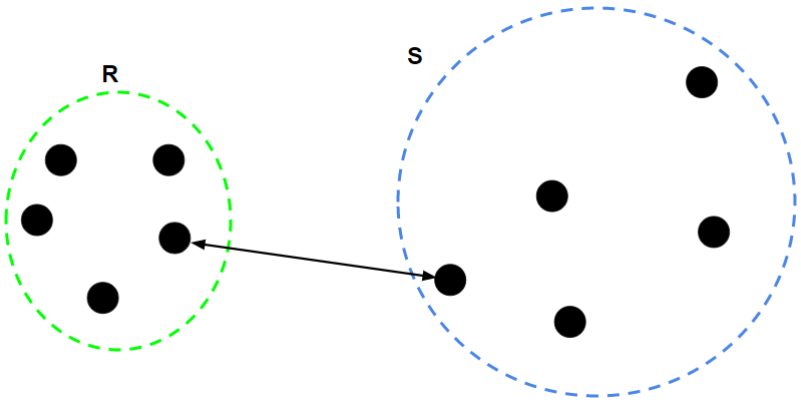
Dit d'altra forma, el que fem és prendre el valor més petit d'entre els costats $\{i, k\}$, $\{j, k\}$ del triangle $\{i, j, k\}$, amb base $\{i, j\}$. Amb la nova distància, el triangle, que abans satisfeia

$$\delta(i, j) \leq \delta(i, k) = a \leq \delta(j, k)$$

es converteix en ultramètric, ja que:

$$\delta'(i, j) \leq \delta'(i, k) = \delta'(j, k) = a$$

Aquest mètode s'anomena habitualment *single linkage*, que podríem traduir com enllaç simple, o mètode del mínim. De forma general, aplicant aquest mètode, la distància entre dos clústers qualssevol és la distància més curta entre totes les possibles prenent un punt del primer clúster i un altre del segon (R i S al gràfic).



Amb el mètode del mínim es tendeixen a obtenir clústers és "llargs", menys compactes, com a resultat. El principal desavantatge d'aquest mètode és que pot provocar la fusió prematura de clústers prou diferents, si aquests tenen parelles de punts molt propers.

Amb l'aplicació d'aquest mètode obtenim una distància ultramètrica $\underline{u}$ amb la següent propietat (Teorema 10.6.1, Cuadras, 2014):

Teorema 3.11. *Sigui*

$$\underline{U} = \{u \mid u \text{ és ultramètrica}, \ u(i,j) \leq \delta(i,j)\}$$

el conjunt de distàncies ultramètriques menors que δ. Llavors la distància ultramètrica $\underline{u}$ resultant d'aplicar el mètode del mínim és l'element màxim de $\underline{U}$:

$$\underline{u}(i,j) \geq u(i,j), \ u \in \underline{U}, \ \forall i,j \in \Omega.$$

Diem que $\underline{u}$ és la millor aproximació a δ per defecte.

3.7 Mètode del màxim, o enllaç complet

La segona opció per escollir la funció f, també bastant natural, consisteix en prendre el màxim de les distàncies possibles, és a dir:

$$\delta'(k, \{i,j\}) = \max\{\delta(i,k), \delta(j,k)\}, \ k \neq i,j,$$

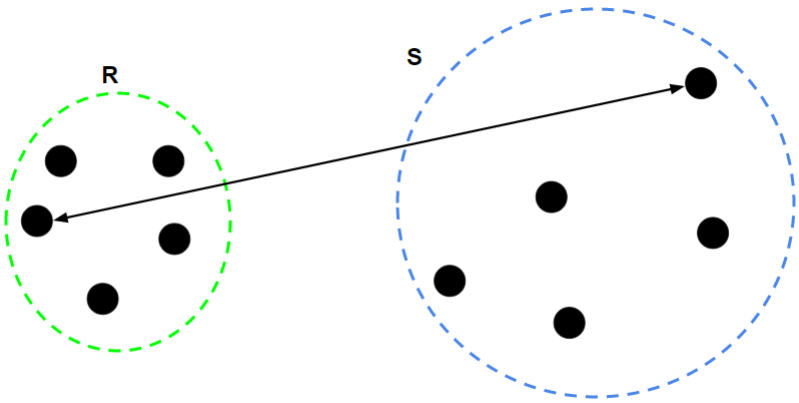
Podem fer la mateixa analogia que amb el mètode del mínim: el que estem fent és prendre el valor més gran d'entre els costats $\{i,k\}$, $\{j,k\}$ del triangle $\{i,j,k\}$, amb base $\{i,j\}$. Amb la nova distància, el triangle, que abans satisfeia

$$\delta(i,j) \leq \delta(i,k) \leq \delta(j,k) = b$$

es converteix en ultramètric, ja que:

$$\delta'(i,j) \leq \delta'(i,k) = \delta'(j,k) = b$$

Aquest mètode s'anomena habitualment *complete linkage*, que podríem traduir com enllaç complet, o mètode del màxim. De forma general, aplicant aquest mètode, la distància entre dos clústers qualssevol correspon a la distància entre els dos punts més llunyans, prenent-ne un de cada clúster (R i S a la imatge), i es tendeixen a obtenir clústers més compactes.



Un desavantatge d'aquest mètode és que els valors atípics o *outliers* poden provocar que grups d'observacions propers es fusionin més tard del que seria òptim.

Aplicant aquest mètode obtenim una distància ultramètrica $\overline{u}$ amb la següent propietat (Teorema 10.6.2, Cuadras, 2014):

Teorema 3.12. *Sigui*

$$\overline{U} = \{u \mid u \text{ és ultramètrica}, \ u(i,j) \geq \delta(i,j)\}$$

el conjunt de distàncies ultramètriques majors que δ. Llavors la distància ultramètrica $\overline{u}$ resultant d'aplicar el mètode del màxim és un element minimal de $\overline{U}$:

$$\overline{u}(i,j) \leq u(i,j), \ u \in \overline{U}, \ \forall i,j \in \Omega.$$

Diem que $\overline{u}$ és la millor aproximació a δ per excés. Les distàncies $\underline{u}$ i $\overline{u}$ verifiquen:

$$\underline{u}(i,j) \leq \delta(i,j) \leq \overline{u}(i,j).$$

A més, tindrem la igualtat $\underline{u} = \delta = \overline{u}$ si i només si δ és distància ultramètrica.

3.8 Altres mètodes

Existeixen molts altres mètodes per definir la distància entre clústers. Alguns dels més coneguts són l'*average linkage*, el mètode del centroide i el mètode de Ward.

El mètode de l'enllaç mitjà, o *average linkage*, funciona de forma molt similar als dos que ja hem vist, però computem la nova distància calculant la mitjana de les distàncies entre tots els punts del primer clúster i tots els punts del segon, és a dir, si tenim dos clústers c i c':

$$\delta'(c,c') = \frac{1}{n_c n_{c'}} \sum_{i=1}^{n_c} \sum_{j=1}^{n_{c'}} \delta(x_{ci}, x_{c'j})$$

on n_c i $n_{c'}$ corresponen al nombre d'elements de c i c', respectivament, i $x_{ci}, x_{c'j}$ indiquen els punts continguts a cada clúster.

Per al mètode del centroide es computen els centroides de cadascun dels clústers, de forma similar al procediment de l'algorisme K-mitjanes, i es pren com a nova distància la distància entre els centroides:

$$\delta'(c,c') = \delta(\overline{c}, \overline{c}')$$

on $\overline{c}, \overline{c}'$ indiquen els centroides de c, c'.

En el mètode de Ward, l'objectiu és minimitzar el total de les variàncies internes dels clústers. La idea és que a cada iteració es fusionen els dos clústers que fan que la suma total de les variàncies internes de cada clúster sigui mínima. Per funcionar, aquest mètdode requereix usar la distància euclidiana com a mesurar de dissimilaritat, i la distància entre clústers es pot calcular fent:

$$\delta'(c, c') = \frac{|c||c'|}{|c| + |c'|} ||\overline{c} - \overline{c}'||^2$$

on $|c|, |c'|$ indiquen el nombre d'elements a c i c', $\overline{c}, \overline{c}'$ els centres de dits clústers, i $||x||$ denota la distància euclidiana.

Els resultats del clústering jeràrquic poden variar bastant en funció del mètode que usem, així com de la mesura de distància inicial que prenem, i en general l'elecció de tots dos és situacional i depén fortament del problema a tractar.

4 Optimització global i validació de resultats

4.1 Problema d'optimització global

En l'estudi teòric dels problemes de cústering, ens trobem el que anomenem problema d'optimització global, això és, que volem trobar una funció global que depengui dels paràmetres del nostre problema (que obtenim a partir de les dades), per tal que el fet de trobar el millor clústering sigui equivalent a trobar els paràmetres que minimitzen aquesta funció, que anomenem funció objectiu.

En general, els algorismes de clústering de tipus particional s'han estudiat àmpliament de forma teòrica, i l'enfoc de dits algorismes ja consisteix en minimitzar una funció objectiu ben definida. En el cas particular de l'algorisme k-mitjanes, com hem exposat anteriorment, la funció objectiu és la suma de les variacions internes dels clústers, i es pot veure fàcilment que l'algorisme k-mitjanes bàsic convergeix a un mínim local d'aquesta funció. Per una banda, és obvi que en l'etapa d'assignació de clústers el valor de la funció es queda igual o bé decreix, ja que estem assignant cada observació al clúster del qual la distància al centroide és mínima. Per altra banda, en l'etapa del càlcul dels centroides tenim que, si considerem n punts $x_1, x_2, ..., x_n$, llavors, donat $a \in \mathbb{R}$ (que s'identifica amb el centroide), definim una funció f com la suma de les distàncies quadrades entre els n punts i a:

$$f(a) = \sum_{i=1}^{n} (x_i - a)^2$$

Derivant f, trobem que la mitjana aritmètica de $x_1, x_2, ..., x_n$ minimitza la funció:

$$f'(a) = \sum_{i=1}^{n} 2(x_i - a)(-1) = 0$$

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

De fet, enfocant el plantejament del k-mitjanes com un problema de descens de gradient, es pot demostrar que la velocitat de convergència de l'algorisme és prou bona. Ho estudien Bottou i Bengio, al seu article de 1995, en el qual descriuen l'algorisme k-mitjanes com un algorisme de descens de gradient, i llavors estableixen una equivalència entre l'algorisme definit i l'algorisme d'esperança-maximització, o EM (un conegut mètode per trobar estimadors de paràmetres en models probabilístics), demostrant que els espais sobre els que treballa cadascun comparteixen moltes propietats algebraiques importants, i descriuen una versió del k-mitjanes en l'estil de l'algorisme EM, que demostren que convergeix a la mateixa velocitat que el mètode d'optimització de Newton. De fet, és conegut que l'algorisme EM convergeix de forma similar al de Newton, pero Bottou i Bengio demostren que l'algorisme k-mitjanes definit amb l'estil de l'EM és encara millor i convergeix *exactament* igual que el mètode de Newton.

Comparat amb el clústering basat en particions, el clústering jeràrquic ha rebut molta menys atenció des d'una perspectiva teòrica. Existeixen molts estudis sobre clústerings jeràrquics, però la gran majoria s'enfoquen en mètodes i algorismes per usar a nivell pràctic, i no pas en l'extracció de conclusions per a la disciplina en general. Això es deu en part a que, tal i com estan definits els algorismes jeràrquics, no és gaire intuïtiu assignar una funció de cost per minimitzar. A més, trobem alguns problemes a l'intentar buscar

aquesta funció. Si intentem fer el mateix que en el k-mitjanes, i plantejar el problema de clústering com un algorisme iteratiu de descens del gradient, topem amb el problema de que hem de treballar amb una matriu de distàncies, i tot i que el concepte és prou simple i fàcil d'entendre, matemàticament, una matriu de distàncies és una estructura complicada de treballar. Si anomenem M a l'espai vectorial format per totes les matrius de mida $N \times N$, tenim que les matrius simètriques amb zeros a la diagonal formen un subespai S de M, però per tenir una matriu de distàncies necessitem complir dues condicions més: tots els elements de fora de la diagonal han de ser estrictament posititus, i els elements de la matriu han de satisfer la desigualtat triangular, és a dir, si $D = (d_{ij})$ amb $1 \leq i, j \leq N$ és la matriu en qüestió, se satisfà per tot i, j que:

$$d_{ij} \leq d_{ik} + d_{kj}, \quad \forall k, 1 \leq k \leq N$$

o la propietat ultramètrica, encara més restrictiva. Aquestes condicions ens impedeixen treballar amb el conjunt de matrius de distàncies com a subespai, i de fet, es pot demostrar que l'estructura que formen aquestes matrius és equivalent a un con convex. Degut a això, és fàcil que si apliquem un algorisme de tipus descens del gradient, en alguna iteració el resultat surti fora de la nostra regió admissible (el con convex). Podriem intentar forçar la continuació de l'algorisme, per exemple truncant el resultat per tal de que caigui dins la nostra regió, però llavors no compliriem els requeriments de l'algorisme de descens del gradient (com podria ser, per exemple, el mètode de Newton), ja que la operació de truncament és no diferenciable.

En els últims anys trobem alguns estudis, com els de Cohen-Addad et al. i Chatziafratis, que aborden el problema de definir una bona funció objectiu per al clústering jeràrquic, tots dos basant-se en la funció proposada per Dasgupta al 2016. La idea d'optimització de Dasgupta és que un arbre o dendrograma és bo si minimitza una certa funció objectiu. Es considera la matriu de distàncies com un graf ponderat $G = (V, E, w)$ (on V és el conjunt de nodes, E el conjunt d'arestes, i w la funció de pes de les arestes, que equival a l'alçada d'un punt al dendrograma). S'anomena T a l'arbre amb fulles corresponents als elements de V (és a dir, el dendrograma), i per qualsevol node u de T, es diu T_u al sub-arbre amb arrel u, (com si talléssim l'arbre per sobre d'aquest node). Llavors, per dues fulles $i, j \in V$, s'anomena T_{ij} al sub-arbre més baix que conté entre les seves fulles a i i j, i s'indica com $|T_{ij}|$ el conjunt de fulles d'aquest sub-arbre. Quan realitzem un clústering jeràrquic, volem que els punts s'uneixin a la part baixa del dendrograma, ja que llavors formen un clúster més compacte. Amb aquest fi, la funció de Dasgupta busca que els sub-arbres per a les parelles de fulles es formin el més abans possible, amb la funció de cost:

$$cost_G(T) = \sum_{i,j \in E} w(i, j)|T_{ij}|$$

Al seu article, Cohen-Addad et al. intenten definir el que és una "bona" funció objectiu per al clústering jeràrquic, tot estudiant del propietats de la funció de Dasgupta i proposant-ne algunes variants, i arriben a la conclusió de que per alguns problemes concrets, on existeix un resultat decididament òptim o "real" (com en alguns estudis de filogenètica), la funció objectiu és aplicable, mentre que en alguns altres casos es pot usar una aproximació heurística.

4.2 Índexs de validació

Quan utilitzem un algorisme de clústering, és molt difícil (o fins i tot impossible) saber a primera vista si el resultat obtingut és una bona classificació de les dades. Com que, en

general, no hi ha una "resposta correcta" al realitzar un clústering, hem de buscar alguna forma de validar els resultats, i per això existeixen multitud dels anomenats índexs de validació. Alguns exemples, que mecionen Embrechts et al. en el seu article, són l'índex de Davies-Bouldin, l'índex de Dunn, l'índex de validació de contorn (*Silhouette*) o l'estadístic GAP; i ho fan amb l'intenció de comparar-los entre ells, ja que afirmen que és difícil determinar un índex millor que la resta, i per això és una bona pràctica calcular-ne i comparar-ne més d'un a l'hora de validar un resultat, obtenint així una perspectiva més àmplia. Aquests índexs mencionats s'anomenen de tipus intern, ja que avaluen la qualitat del clústering usant únicament quantitats i atributs inherents a les dades. La contrapart d'aquests són els de tipus extern, que veurem a continuació, juntament amb l'explicació d'un parell d'exemples, com són els índexs de Davies-Bouldin i Dunn.

4.2.1 Índex de Davies-Bouldin

Originalment, aquest índex està orientat a obtenir la quantitat òptima de clústers k per a l'algorisme K-mitjanes, usant el valor del centroide dels clústers i les distàncies típiques d'aquest (euclidianes), però es pot adaptar per ser usat en algorismes de clústering jeràrquic. La idea general del mètode és valorar positivament una variació interna dels clústers petita, i una distància entre clústers el més gran possible.

Definim:

$$R_{ij} = \frac{S_i + S_j}{D_{ij}}$$

on S_i, S_j indiquen les variacions internes dels clústers i, j, i D_{ij} és una mesura de la distància entre els dos clústers. La idea d'aquesta R és una mesura de dispersió entre dos clústers, que serà baixa si els clústers són compactes i separats, i per tant els valors baixos seran millors. Llavors calculem:

$$\overline{R_i} = \max_{j \neq i} R_{ij},$$

que ens dóna la "pitjor" valoració entre les possibles parelles de clústers, i per últim, si tenim N clústers, definim l'índex de Davies-Bouldin com:

$$DB = \frac{1}{N} \sum_{i=1}^{N} \overline{R_i}$$

per al qual un valor més baix indica un clústering millor. En el cas del k-mitjanes, si calculem l'índex per a diversos valors de N, podem aconseguir el millor nombre de particions a fer. Les definicions de S_i i D_{ij}, com hem dit, dependran de l'algorisme que volem avaluar. Per al k-mitjanes usualment s'utilitza com a variació interna la suma ponderada de les distàncies euclidianes entre cada punt del clúster i el centroide, i per les distàncies entre clústers, es pot prendre la distància entre centroides. Una possible modificació per al cas jeràrquic, que proposa Embrechts, és usar la mitjana de les distàncies entre totes les parelles de punts, un de cada clúster, com a mesura de separació entre clústers, i substituir la variació interna per la distància mitjana entre totes les parelles de punts dins un mateix clúster.

4.2.2 Índex de Dunn

L'índex de Dunn funciona de forma similar al de Davies-Bouldin, en el sentit que considera un clústering bo aquell en el que els clústers són compactes i ben separats. L'índex de

Dunn es defineix com el ratio entre la mínima variació interna d'entre els clústers sobre la màxima distància entre clústers, és a dir, si tenim un clústering C:

$$Dunn = \frac{\min_{S \in C}(\Delta S)}{\max_{S,T \in C}(\delta(S,T))}$$

Per a la variació interna dels clústers, ΔS, es calcula la distància màxima entre dos punts dins el clúster, i per a la distància entre clústers, $\delta(S,T)$ es calcula la distància mínima entre dos punts, un de cada clúster:

$$\Delta S = \max_{x,y \in S}(d(x,y))$$

$$\delta(S,T) = \min_{x \in S, y \in T}(d(x,y))$$

on $d(x,y)$ és una mesura de distància adequada per al problema. Una diferència important d'aquest indicador respecte a l'anterior és que un valor alt de l'índex de Dunn indica un bon clústering, i un valor baix, un de poc fiable.

4.2.3 Validació externa: Índex de Rand

Hem vist els anomenats índexs de validació interns, però també existeixen els anomenats externs. Aquests s'utilitzen per validar clústerings quan treballem amb dades etiquetades, és a dir, en el nostre conjunt de dades existeix una classificació "per defecte", externa, que ens ve donada amb el propi conjunt. Els índexs externs comparen aquestes classes o etiquetes predefinides amb el clústering obtingut per realitzar la validació, i es basen en els valors donats per l'anomenada taula de contingències, que compta el nombre de parelles de punts que comparteixen classe i clúster, comparteixen només un dels dos, o no comparteixen res, de la següent forma:

	Mateix clúster	Diferent clúster
Mateixa classe	A	B
Diferent classe	C	D

Un exemple senzill d'aquest tipus d'indicador és l'anomenat índex de Rand, que es calcula sumant el nombre de parelles de punts que coicideixen o bé difereixen tant en classe i clúster, i dividint pel total de parelles possibles:

$$RI = \frac{A + D}{A + B + C + D}$$

Aquest càlcul resulta en un valor que oscil·la entre 0 i 1, i en general es considera que tenim un mal clústering per valors inferiors a 0.5, i un de plausible per valors superiors a aquesta xifra.

5 Clústering per a grans dades

5.1 Sobre el temps de computació i ús de memòria

Com hem dit anteriorment, un dels problemes principals dels algorismes jeràrquics és que escalen de forma bastant pobre en temps de computació i requeriments de memòria. Aquest problema es torna especialment greu si fem servir el mètode de l'enllaç mitjà. Això últim és rellevant degut a que els mètodes del mínim i el màxim, com s'exposava en les seves respectives seccions, poden presentar alguns problemes en certes circumstàncies, i tot i que siguin considerablement més ràpids que l'enllaç mitjà o average linkage, també són menys consistents.

Quan apliquem l'algorisme SAHN tenim que a cada iteració l'algorisme ha de buscar entre les $\frac{N(N-1)}{2}$ entrades de la matriu de distàncies per trobar els dos clústers més similars i fusionar-los. Tenint en compte que l'algorisme realitza un total de $N-1$ iteracions, tenim que el temps de computació referent a aquesta operació escala en l'ordre de $O(N^3)$. Veurem però, que aquesta estimació és bastant pobre, i a la pràctica es poden trobar mètodes més ràpids d'aplicar aquest mateix algorisme.

Pel que fa als requeriments de memòria, trobem que l'algorisme es necessita una matriu de distàncies que hem de mantenir a la memòria, i a més, es tracta de matrius denses, fet que ens elimina algunes possibilitats a l'hora d'optimitzar l'espai. Com que aquestes matrius són simètriques (i tenen la diagonal nul·la), l'espai necessari per emmagatzemar aquestes dades és $\frac{N(N-1)}{2}$, on N és la quantitat d'observacions que tenim, és a dir, escala de forma quadràtica.

Una observació que fan Embrechts et al. d'aquests resultats és que el temps total de computació per a magnituds grans de dades (fan proves amb 10000 i 100000 observacions), és de l'ordre d'hores, és a dir, que tot i que l'algorisme seria excessivament lent per a un nombre tan gran de dades, no és del tot impossible, i afirmen que el problema real el trobem en el requeriment de memòria. En general però, podem concluir que sense algún tipus d'algorisme millor, realitzar clústerings jeràrquics amb grans dades és pràcticament impossible.

5.2 Algorismes de millora d'eficiència

Existeixen molts estudis que proposen algorismes millors per al clústering jeràrquic, i en veurem alguns en aquesta secció. Tot i que dit algorisme SAHN escala en l'ordre de $O(N^3)$, una bona implementació de codi suposa una millora notable, i per això, en general, els algorismes és actualitzar les dissimilaritats entre clústers a cada iteració, en comptes de recalcular tota la matriu cada vegada.

De fet, es coneixen diversos algorismes jeràrquics que escalen de forma quadràtica en temps de computació en el pitjor dels casos. Alguns exemples coneguts són el NN-Chain (*Nearest-Neighbour*), que es basa en trobar els clústers candidats a fusionar-se seguint camins del graf de veïns més propers (*Nearest Neighbour Graph*), o els mètodes basats en parelles properes dinàmiques. Alguns d'aquests algorismes estan limitats a certs mètodes d'enllaç (*linkage*), com és el cas de l'algorisme de l'arbre generador minimal que es pot aplicar amb el mètode de l'enllaç simple. Aquest últim, a més, es pot accelerar mitjançant l'ús de processadors en paral·lel, com és el cas dels algorismes que proposen Li et al. i Hendrix et al.. Veurem el funcionament d'un d'ells més endavant. Una

observació interessant és que alguns d'aquests algorismes, a més d'escalar quadràticament en temps de computació, aconsegueixen evitar l'ús de la matriu de distàncies, utilitzant representants dels clústers (com centroides), o bé altres mètodes, per estalviar-se bona part de les dades que han de guardar a memòria.

Existeixen altres idees per accelerar el clústering jeràrquic, com la simplificació de les dades. Una forma senzilla de simplificar les dades és prendre una mostra aleatòria de les dades i classificar aquesta en comptes de tot el conjunt. Un inconvenient d'això és que un mostreig aleatori dona lloc a distorsions en els resultats del clústering. Aquestes distorsions es veuen afectades en gran mida pel mètode d'enllaç que utilitzem, i és per això que moltes tècniques sofisticades d'aquest tipus es veuen limitades a un tipus d'enllaç en concret. Embrechts et al. proposen una idea diferent per accelerar l'algorisme jeràrquic amb el mètode d'enllaç mitjà, que consisteix en inicialitzar el mètode jeràrquic amb un clústering obtingut de l'aplicació de l'algorisme k-mitjanes, per tal de minimitzar el temps de computació total. Anomena a aquest mètode *cluster seeding*. Una observació interessant és que es pot fer aquest *seeding* a la inversa, com proposen Kwon i Han, inicialitzant la classificació amb el clústering jeràrquic, per acabar aplicant el k-mitjanes. En aquest cas, però, l'obectiu no és accelerar el clústering, sinó arreglar certes deficiències particulars de l'algorisme jeràrquic en certs problemes d'anàlisi d'ADN.

Els algorismes que hem mencionat fins ara podrien ser anomenats "exactes", ja que realitzen tots els càlculs de les distàncies entre punts amb exactitud. Existeix un altre tipus de mètode per accelerar encara més el clústering que consisteix en usar aproximacions, els anomenats mètodes heurístics. Un exemple és l'algorisme LSH (*Locality Sensitive Hashing*), proposat per Koga et al., que té una complexitat de $O(nB)$, on B és pràcticament constant. Tot i que la velocitat de l'algorisme és molt prometedora, es veu limitat al cas de l'enllaç simple.

Kriege et al. proposen una variació heurística de l'algorisme SAHN, que anomenen HSAHN, que funciona per a qualsevol mètrica, amb la contrapartida de que només és aplicable amb els mètodes d'enllaç del centroide i de la mediana. L'algorisme es basa en una variació del SAHN clàssic usant una estructura d'índexs per la cerca del veí més proper d'un clúster (un altre clúster amb distància mínima al primer). Per determinar de forma heurística els veïns més propers dels clústers es combina una estructura d'arbre de pivots amb un mètode de cerca de millor frontera (*best frontier search*), que permeten realitzar de forma molt eficient els càlculs de les distàncies aproximades. El resultat és un algorisme amb una complexitat temporal $O(n \log n)$ en el millor cas, i gràcies a l'estructura de l'arbre de pivots, només requereix una quantitat lineal d'espai.

Veiem a continuació alguns dels mètodes esmentats.

5.2.1 Cluster Seeding amb K-mitjanes

Com hem dit, la idea d'aquest mètode és accelerar l'algorisme jeràrquic d'enllaç mitjà tot iniciant aquest últim amb un clústering obtingut mitjançant un altre algorisme més ràpid, com el K-mitjanes, o el propi SAHN usant els mètodes del mínim o el màxim. Cal mencionar que el cost computacional de l'algorisme K-mitjanes estàndard escala en l'ordre de $O(N^2)$, i existeixen implementacions més ràpides. Com que l'escalatge és millor que en l'algorsime SAHN, a l'hora de calcular el cost computacional de l'algorisme complet, només cal considerar la segona fase, la jeràrquica, ja que el temps de computació del k-mitjanes és negligible en comparació, fins i tot si l'apliquem diverses vegades, per tal

d'obtenir un resultat més fiable. La part negativa d'aquest mètode és que el resultat obtingut serà diferent del que obtindríem usant l'algorisme jeràrquic directament, però es pot comprovar, mitjançant diversos índexs de validació, que el clústering obtingut és acceptable.

Embrechts realitza dos experiments diferents de *seeding*, per comparar el funcionament del SAHN estàndard amb el el de l'algorisme híbrid proposat, sobre diversos conjunts de dades. En el primer inicia la fase jeràrquica amb $\frac{N}{2}$ clúster obtinguts amb el k-mitjanes (suposant N dades o observacions), i en l'altre amb $\frac{3N}{4}$ clústers. Degut a l'escalatge cúbic del temps de computació del SAHN, el clústering del primer experiment, amb la meitat de clústers inicials, hauria de ser 8 vegades més ràpid, mentre que en el segon, que tenim una quarta part del nombre total d'observacions, hauria de ser 64 vegades més ràpid. Per assegurar la fiabilitat de la fase del k-mitjanes, pren 100 inicialitzacions diferents, i selecciona la que té una menor dispersió interna dels clústers per a la segona fase. Embrechts avalua els resultats usant diversos índexs, i conclueix que, tot i que tant els clústerings obtinguts com la quantitat de clústers resultants al aplicar el *seeding* difereixen dels que obtindriem amb l'algorisme original, la signatura general dels índexs de validació (com el Davies-Bouldin o el de Dunn) és prou estable per la majoria de conjunts de dades, del que es pot deduir que la qualitat del clústering no varia gaire al aplicar aquest mètode.

5.2.2 Ús de memòria: Dividir i vèncer

Per una altra banda, tenim el problema de la memòria. Com hem vist, el fet de guardar a memòria una matriu de dissimilaritats provoca que l'escalat de memòria dels algorismes sigui de l'ordre de $O(N^2)$. Una possible solució que proposa Embrechts a aquest problema és prendre un model de dividir i vèncer. El que farem és dividir les dades inicials en una sèrie de subconjunts aleatoris, i llavors farem un clústering inicial d'aquests subconjunts. Pararem el clústering jeràrquic d'aquests subconjunts en un punt en que la matriu combinada de dissimilaritats per als diferents subconjunts hi càpiga a memòria, i llavors continuarem el clústering jeràrquic amb la matriu de dissimilaritats combinada, que serà considerablement més petita que l'original. No queda demostrat però, que la validació dels clústers sigui semblant amb l'aplicació d'aquest mètode.

5.2.3 Mètode de l'enllaç simple en paral·lel

Com hem mencionat anteriorment, en el cas de l'enllaç simple, es poden trobar algorismes bastant més ràpids que en el cas de l'enllaç mitjà. L'algorisme més conegut d'aquest tipus és el de l'arbre generador minimal (*minimum spanning tree*). Algunes de les versions d'aquest algorisme intenten explotar la idea de que és possible dividir-lo en diversos subprocessos independets, fet que permet usar múltiples processadors en paral·lel a l'hora de computar-lo. Per il·lustrar el funcionament i rendiment d'aquests processos, veiem com a exemple l'algorisme proposat per W. Hendrix et al., que anomenen PINK (de *parallel single linkage*). Aquest algorisme, es basa en el fet de que resoldre el problema del clústering jeràrquic amb enllaç simple és equivalent a calcular l'arbre generador minimal d'un graf complet ponderat, on el "pes" de les arestes correspon ve donat per la funció de distància corresponent aplicada als vèrtexs del graf.

L'estructura general del PINK per poder aplicar un algorisme en paral·lel consisteix en dividir el problema en múltiples subproblemes de dos tipus diferents, resoldre els subpro-

blemes de forma independent, i per acabar, combinar les solucions dels subproblemes en una solució pel problema principal. Per començar, es divideixen les dades en una sèrie de particions. Llavors, per al primer tipus de subproblema es pren el graf bipartit complet definit per cada parella de particions, mentre que pel segon s'utilitzen els grafs complets corresponents a cada partició, agrupats en parelles. A cadascun d'aquests subproblemes se li assigna un procés que es computarà en paral·lel a la resta. L'algorisme segueix el següent esquema:

1. Dividim les dades en k subconjunts de la mateixa mida, $D_1, D_2, ..., D_k$.
2. Formem $\binom{a}{b}$ subgrafs, prenent els grafs bipartits complets per a cada parella de subconjunts (D_i, D_j) .
3. Formem $k/2$ subgrafs prenent la unió dels grafs complets corresponents a D_i i D_{i-1}, per a cada valor parell de i. En cas de que k sigui imparell, formem un subgraf usant únicament el graf complet que correspon a D_K.
4. Cadascun dels processos (associats als grafs obtinguts) calcula l'arbre generador minimal del graf corresponent, mitjançant l'algorisme de Prim.
5. Ordenem segons el pes les arestes dels arbres obtinguts de forma independent.
6. Combinem els arbres generadors minimals afegint, de forma iterativa, l'aresta amb el mínim pes, i eliminem les arestes que formin un cicle.
7. Repetim el pas 6 fins que obtenim un únic arbre generador minimal.

Notem que en el pas 4 s'utilitza l'algorisme de Prim. El raonament és que aquest algorisme, a part de ser una forma eficient de calcular els arbres generadors minimals, es pot aplicar a grafs no complets (condició necessària per alguns dels subgrafs), i a més es pot implementar sense ocupar més espai del necessari per ficar l'arbre resultant. En els passos 6-7, es combinen els arbres de dos en dos, iterant sobre les arestes de tots dos, afegint cada vegada l'aresta de menor pes a l'arbre combinat (procés prou eficient, ja que hem ordenat les arestes al pas anterior), i evitant afegir arestes que formin cicles (ja que volem formar un nou arbre generador minimal). El resultat es tradueix fàcilment en un dendrograma, seguint el teorema següent, demostrat a l'article de Hendrix:

Teorema 5.1. *Sigui T el dendrograma associat a un conjunt de dades D, obtingut mitjançant el mètode de l'enllaç simple, i sigui M l'arbre generador minimal del graf complet ponderat induït per les distàncies entre totes les parelles de punts de D. Llavors, el conjunt de clústers formats tallant T a una alçada h són exactament les components de M formades per l'eliminació de totes les arestes amb un pes major que h.*

La fase més dura a nivell computacional del PINK és el càlcul corresponent a l'algorisme de Prim, que en el pitjor dels casos escala de forma quadràtica en temps de computació en el cas en sèrie, la divisió del problema en subproblemes paral·lels accelera encara més el procés. Pel que fa a l'ús de memòria, el PINK només requereix una quantitat lineal d'espai, que suposa una diferència molt gran respecte els algorismes que usen una matriu de distàncies.

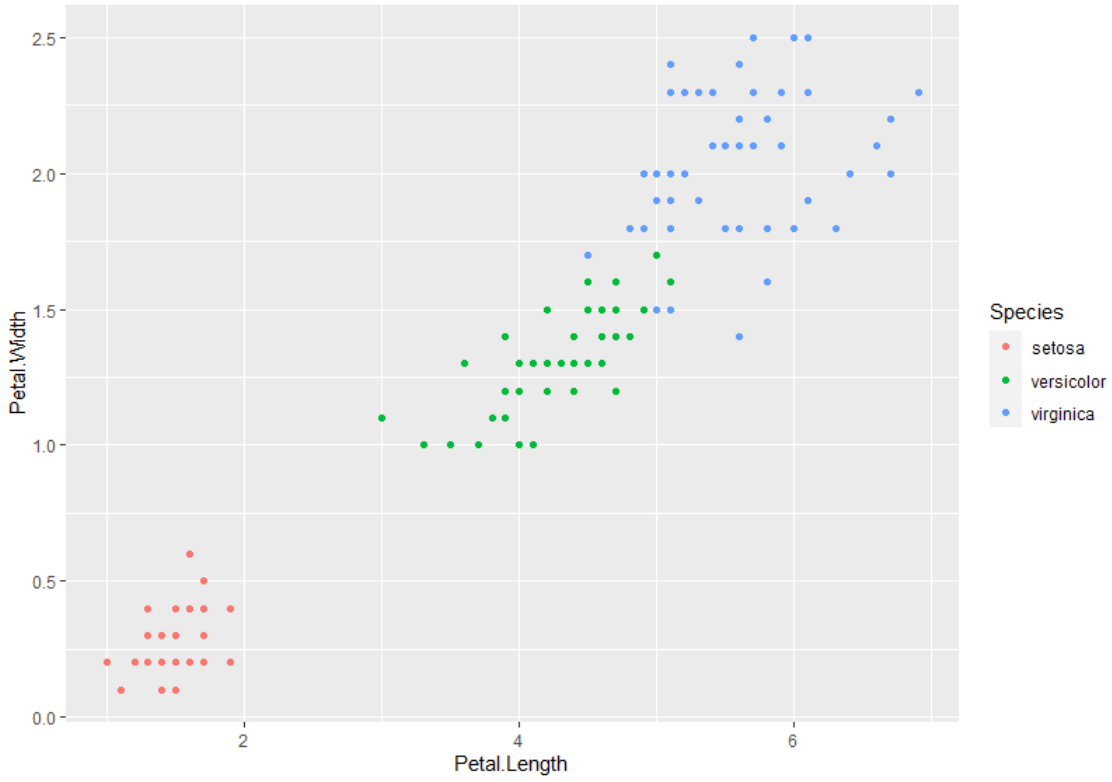
6 Càlculs i presentacions gràfiques

En aquest darrer capítol veurem alguns exemples dels càlculs i visualitzacions de dades referents a mètodes de clústering que podem realitzar amb R. Les funcions bàsiques per realitzar clústerings particionals i jeràrquics són *kmeans* i *hclust*, respectivament, totes dues del paquet *stats*. En veurem exemples de l'aplicació de totes dues, així com algunes funcionalitats més avançades que podem obtenir incorporant altres paquets.

6.1 Clústering amb k-mitjanes: kmeans

Per l'exemple de k-mitjanes, usarem el conjunt de dades "iris" de Fisher, que conté tres espècies de plantes (setosa, virginica i versicolor). Aquest conjunt de dades té 150 entrades amb 5 atributs per cadascuna, corresponents al nom de l'espècie, l'alçada i l'amplada del pètal, i l'alçada i amplada del sèpal.

Observant una mica les dades, es pot arribar a la conclusió de que les mides dels pètals són molt semblants per mostres de la mateixa espècie, però varien de forma apreciable entre espècies diferents. Si representem gràficament, usant la funció *ggplot* (del paquet "ggplot2") les dades segons les mides del pètals, observem que les espècies queden ben classificades:



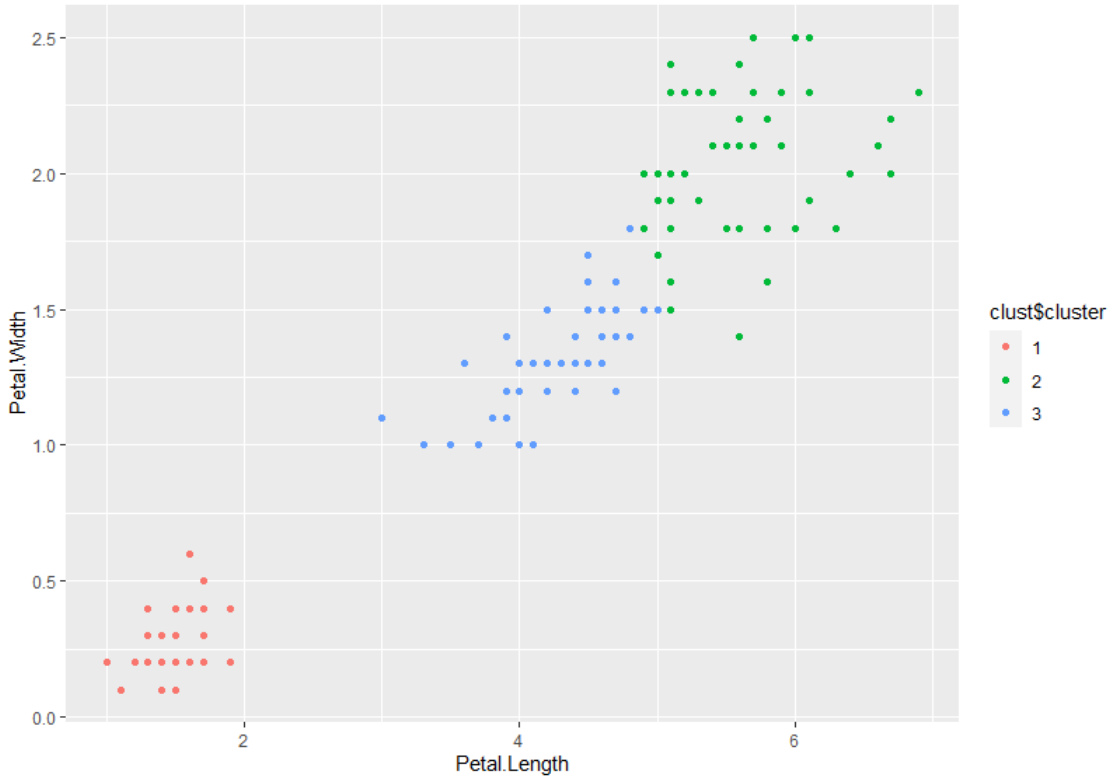
Probarem d'aplicar l'algorisme k-mitjanes usant les mides dels pètals per veure si s'aconsegueix classificar correctament les diferents espècies. Per fer-ho, usem la funció *kmeans*, passant com a paràmetres les columnes 3 i 4 de les dades (corresponents a les alçades i amplades dels pètals), el nombre de clústers a obtenir, que serà $k = 3$,

per tal de poder comparar el resultat amb la classificació original, i per últim passem $nstart = 20$, que indica a la funció que realitzi 20 classificacions diferents variant el clústering inicial, que és aleatori, i es quedi amb la millor (aquella que té la mínima suma de variacions internes als seus clústers). Donada la naturalesa aleatòria de la primera part de l'algorisme, farem *set.seed* abans d'aplicar-lo, per poder reproduir els resultats.

Comparem els clústers obtinguts amb les espècies de plantes mitjançant una taula:

	setosa	versicolor	virginica
Clúster 1	50	0	0
Clúster 2	0	2	46
Clúster 3	0	48	4

Observem que les mostres de l'espècie setosa han acabat al clúster 1, les de l'espècie versicolor al 2, i les de tipus virginica al 3. A més, veiem que 4 mostres de tipus virginica i dues de tipus versicolor han acabat al clúster equivocat, però en general, la classificació és prou bona. Ho podem veure també amb la representació gràfica del clústering obtingut:

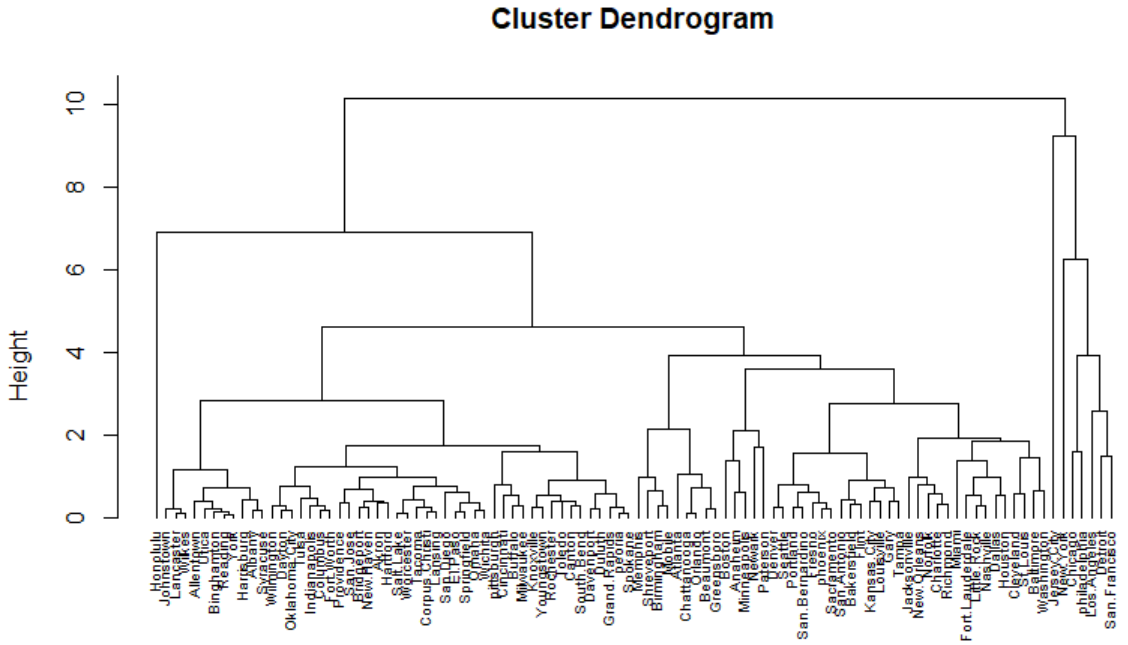


6.2 Clústering jeràrquic: hclust i agnes

Per aquest exemple de clústering jeràrquic, usarem el conjunt de dades de Freedman, del paquet "car". Aquest conté observacions corresponents a les àrees metropolitanes dels Estats Units amb poblacions superiors a 250000 habitants en l'any 1968, amb les variables següents per cadascuna: població total (en milers), percentatge de població "no-blanca", densitat de població, i ratio de crim per 100000 habitants. Aquestes dades

contenen algunes entrades incompletes, així que abans de començar convé netejar-les usant la funció *na.omit*, que elimina les entrades amb algún atribut amb valor NA. Per calcular les distàncies entre observacions és convenient normalitzar les dades, per tal d'evitar que l'algorisme depengui fortament d'una única variable, si la proporció entre els valors de variables és molt diferent. Ho fem usant la funció *scale*, que corregeix les variables per tal de que cadascuna tingui mitjana 0 i desviació típica 1.

La funció principal per realitzar clústerings jeràrquics a R és *hclust*. Aquesta rep com a paràmetres una matriu de distàncies condensada i un string corresponent al mètode d'enllaç que volem usar. Els mètodes que accepta la funció són: "single", "complete", "average", "mcquitty", "centroid", "median", "ward.D" i "ward.D2". Podem computar la matriu de distàncies que correspon a les dades mitjançant la funció *dist*. En aquest cas, usarem la distància euclidiana, i aplicarem el mètode de l'enllaç mitjà. La funció *hclust* retorna un objecte de tipus clústering jeràrquic, que representa un dendrograma. El podem visualitzar usant *plot*:

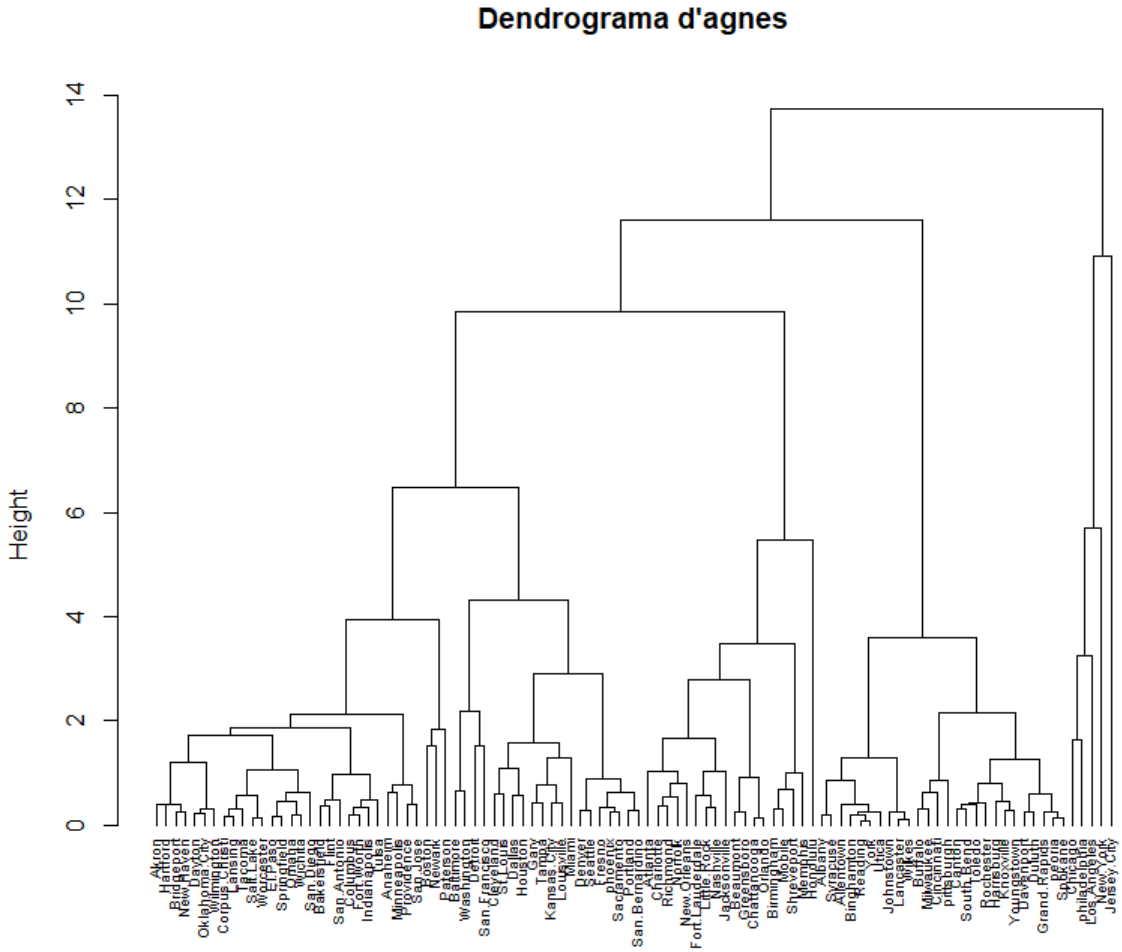


Una altra opció prou coneguda per realizar clústerings jeràrquics és la funció *agnes*, del paquet "cluster". L'avantatge d'aquesta funció és a part del resultat del clústering dóna un valor que anomena coeficient aglomeratiu, una mesura orientativa de la qualitat d'estructura de clústering trobada (un valor proper a 1 representa una estructura millor). Per exemple, si apliquem *agnes* amb el mètode de l'enllaç complet, usant el sufix $*ac* per veure l'esmentat coeficient, obtenim un valor de 0.9317.

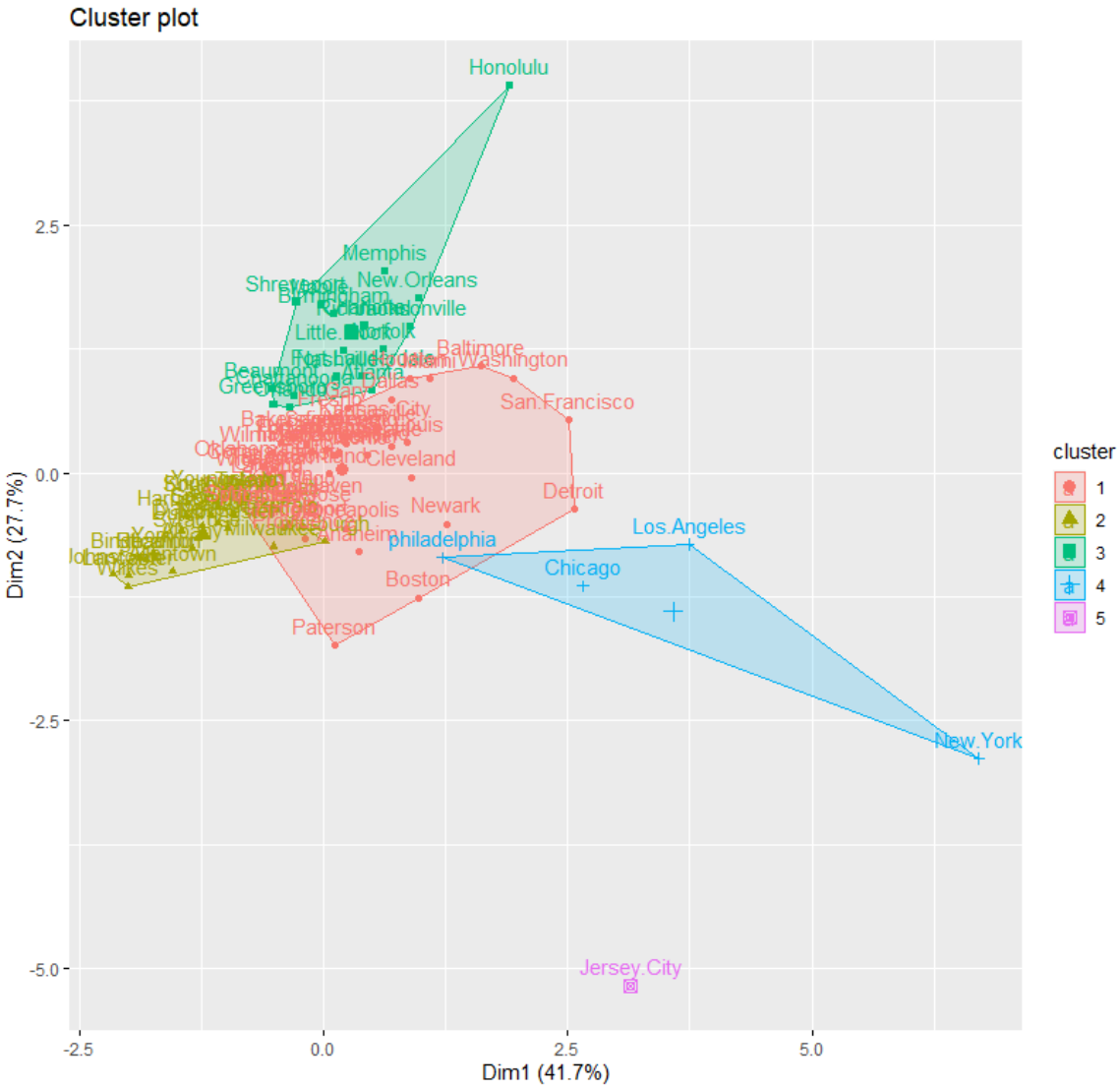
Un experiment interessant és calcular aquest coeficient per diferents mètodes d'enllaç. Ho provem amb els mètodes d'enllaç simple, complet, mitjà, i el mètode de Ward, i obtenim els següents resultats:

average	single	complete	ward
0.9241325	0.9215283	0.9317012	0.9493598

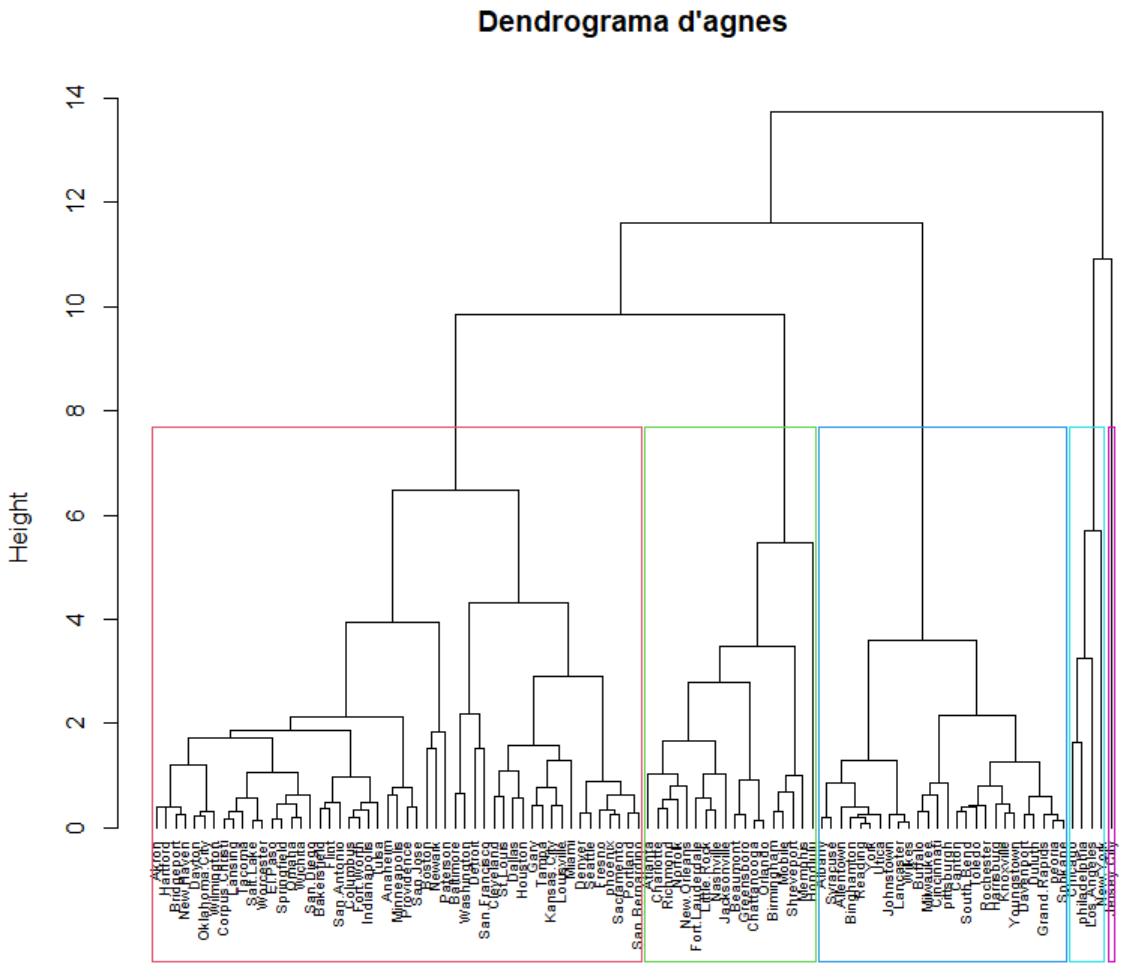
Veiem que el mètode de Ward ens dona el coeficient més alt. Usem aquest resultat per veure un exemple de dendrograma d'*agnes*, que podem obtenir amb la funció *pltree* (l'estructura retornada per *agnes* és diferent de la de *hclust*, i no pot ser visualitzada mijançant *plot*), del paquet "cluster".



Per visualitzar els clústerings resultants, com sabem, hem de tallar el dendrograma. Ho podem fer amb la funció *cutree*, que talla un dendrograma en la quantitat de clústers k o bé l'alçada h que desitgem. Provem a tallar el darrer arbre obtingut amb el mètode de Ward de forma que obtinguem 5 clústers. Per visualitzar els resultats, podem crear un gràfic de dispersió amb la funció $fviz\_cluster$, del paquet "factoextra" (basada en "ggplot2").



Una altra opció disponible és usar *rect.hclust* per visualitzar els clústers dins el propi dendrograma, obteninint un resultat com el següent (del mateix dendrograma del mètode de Ward):



6.3 Acceleració de hclust: flashClust i fastcluster

Existeixen dos paquets a R, el "flashClust" i el "fastcluster", que implementen versions més ràpides de la funció *hclust*. A continuació fem una sèrie de proves amb les implementacions que ens ofereixen aquests paquets, per comparar els resultats amb els de la versió original de *hclust*, del paquet "stats". Cal tenir en compte que aquests paquets sobreescriuen la funció *hclust* original, i per tant caldrà cridar-la fent *stats* :: *hclust* per fer les comparacions. A més, per la prova amb "flashClust", que realitzarem primer, cal desactivar el paquet "fastcluster", ja que aquest també sobreescriu la versió de "flashClust".

Per comparar la implementació de "flashClust" amb l'original, generarem dues matrius de distàncies aleatòries, usant la funció $runif$, que genera desviacions aleatòries. La primera matriu tindrà les 'distàncies' entre 2000 observacions, i la segona entre 10000, i realitzarem 5 proves de computzació de cada mètode per a cadascuna, per assegurar-nos de que obtenim un resultat consistent. Mesurem el temps de computació amb *system.time*. Obtenim els següents resultats, en temps total (*elapsed*):

	stats 2000	flashClust 2000	stats 10k	flashClust 10k
1	0.13	0.18	4.56	5.96
2	0.13	0.17	4.53	6.00
3	0.14	0.19	4.54	5.95
4	0.15	0.21	4.62	5.99
5	0.16	0.18	4.46	6.06

Els resultats obtinguts amb la implementació de "flashClust" no són gaire conclusius respecte l'acceleració de *hclust*. Realitzem les mateixes proves amb la implementació de "fastcluster" per veure si obtenim un resultat millor.

	fastcluster 2000	fastcluster 10k
1	0.08	3.22
2	0.08	3.18
3	0.08	3.23
4	0.07	3.20
5	0.08	3.23

Comparant aquests resultats amb els de la versió de "stats", podem concluir que "fascluster" ens proporciona una millora considerable de rendiment.

Una observació interessant que aplica tant a la versió de "flashClust" com a la de "fastcluster" és que l'objecte que retorna la funció és exactament igual al que s'obté amb la versió de "stats", a excepció de la component *call*. En particular, els dendrogrames resultants coincideixen.

El paquet "fastcluster", a més, ofereix una funció anomenada *hclust.vector*, que millora els requeriments de memòria en el cas de que treballem amb dades de tipus vector. Aquesta funció requereix un espai de mida $O(ND)$ per a N punts de $\mathbb{R}^D$, una millora considerable respecte al requeriment $O(N^2)$ de *hclust*. Un parell d'observacions sobre el funcionament de *hclust.vector* és que requereix com a paràmetre una matriu $N \times D$ de les dades (no de distàncies), i que es veu limitada als mètodes del centroide, de la mediana, de Ward, i l'enllaç simple.

Per exemple, si intentem generar una matriu de distàncies com les anteriors amb 50000 nodes, obtenim un error de memòria (demana un vector de 18.6 Gb). En canvi, guardar les dades de 50000 vectors de $\mathbb{R}^3$ és relativament senzill (una matriu 50000×3). Podem provar l'execució de *hclust.vector* amb una matriu d'aquesta mida, i veiem que funciona adequadament, trigant poc més de 21 segons en acabar l'execució.

Totes les proves d'aquesta secció s'han realizat sobre una CPU Intel Core I7-9750H (2.6GHz), amb un sistema operatiu Windows 10, i una memòria RAM de 16 Gb, usant l'entorn RStudio.

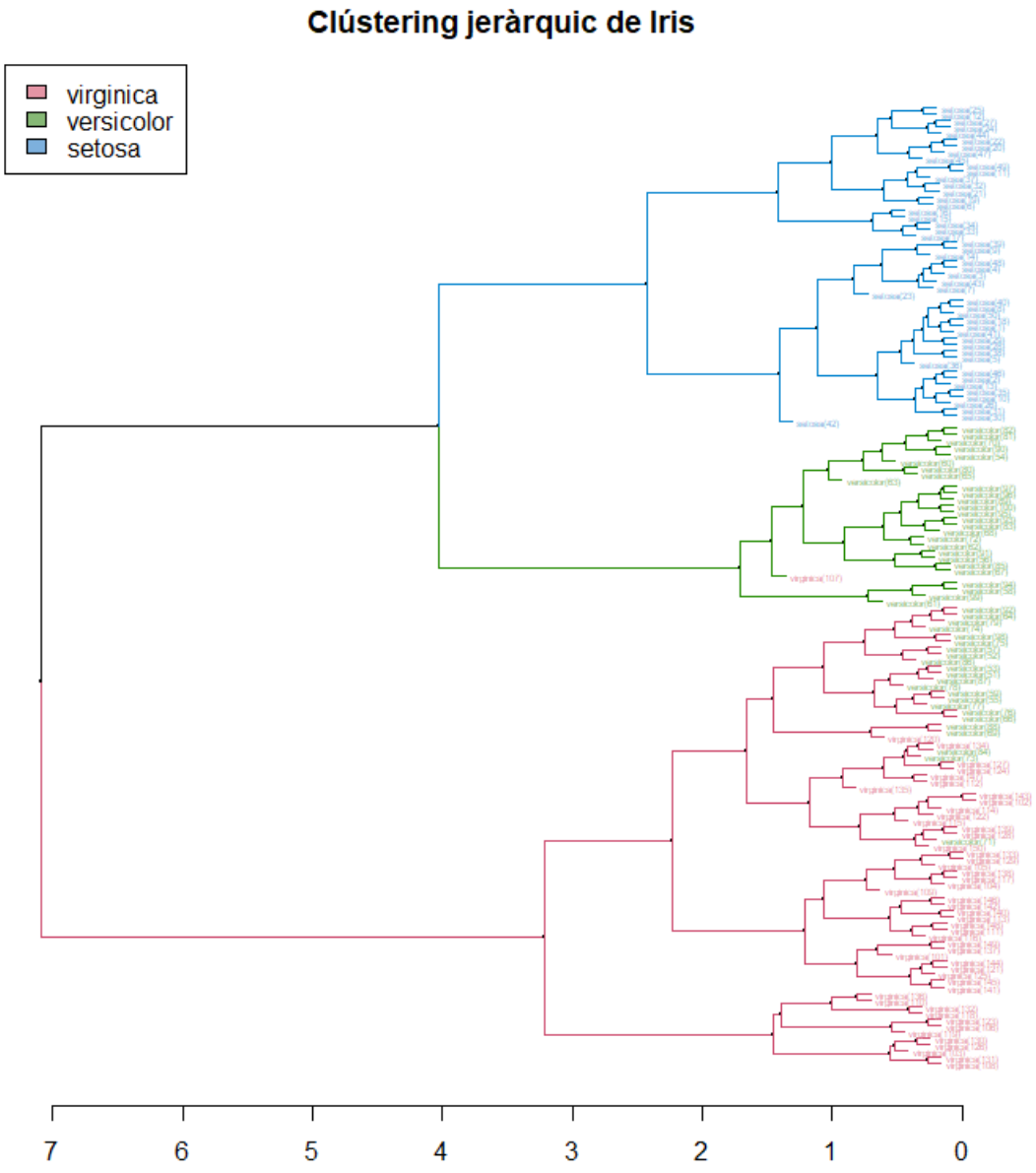
6.4 Eines per al clústering jeràrquic: dendextend

Un altre paquet interessant és el "dendextend", que ens ofereix eines més avançades per al clústering jeràrquic, mitjançant una sèrie d'opcions gràfiques per visualitzar i comparar dendrogrames, així com realitzar comparacions entre els diferents mètodes d'enllaç.

Farem una sèrie de proves tornant a usar les dades "iris". Hem vist abans un exemple molt simple, considerant només unes certes mesures, en que es podia realitzar una classi-

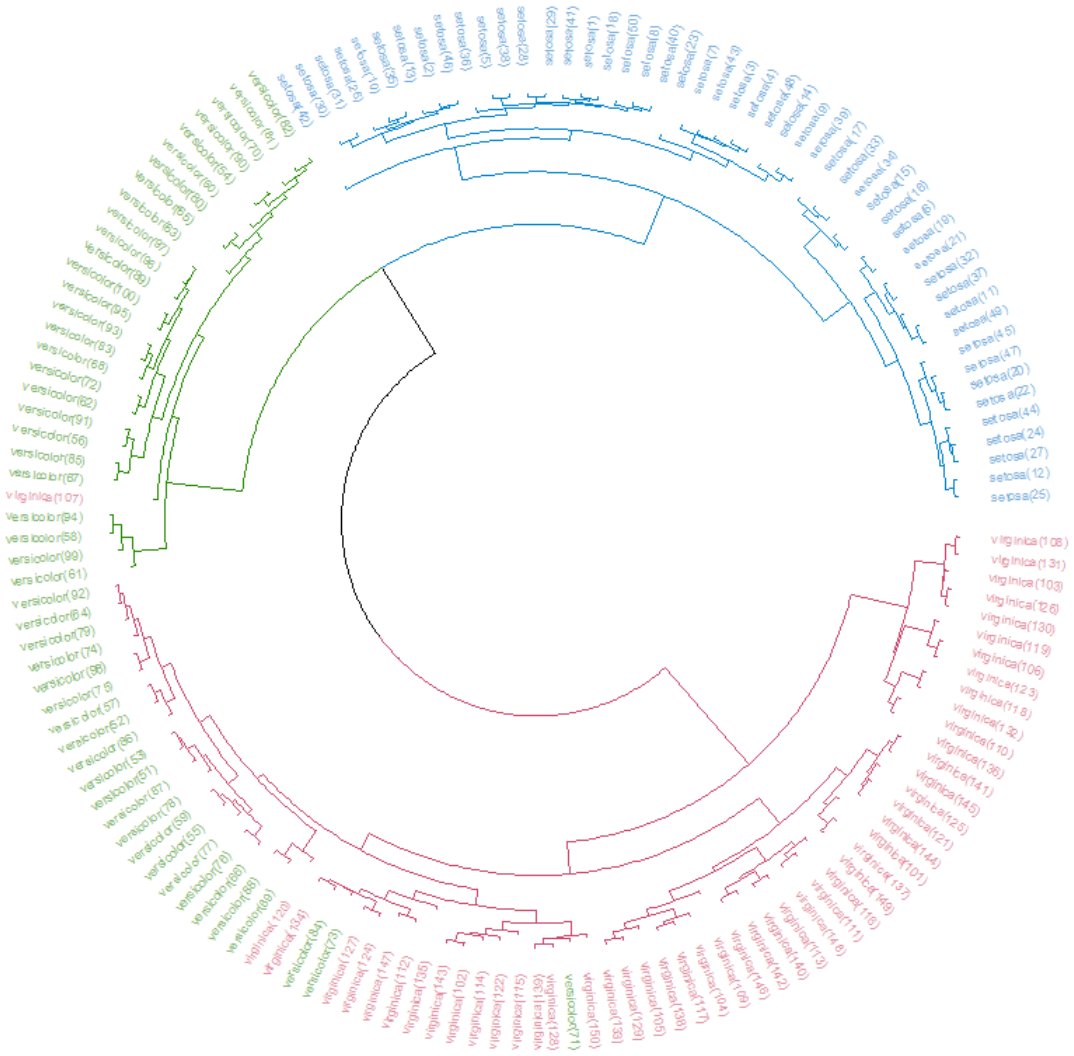
ficació prou bona, però si estudiem les dades en la seva totalitat, la classificació no és tan senzilla, especialment entre les espècies versicolor i virginica.

Prendrem les dades "iris", excloent la darrera columna, que conté les etiquetes, i aplicarem *hclust* amb el mètode 'complete'. Podem convertir el resultat del clústering en un objecte de tipus dendrograma, usant *as.dendrogram*. Dendextend ens ofereix diverses funcions per ajustar aquest objecte, modificant-ne els atributs. Per exemple, podem reordenar les branques del dendrograma amb *rotate*, podem canviar el color de les branques de l'arbre en funció d'un cert tall (per a l'exemple, ho provarem amb $k = 3$). També podem modificar els colors de les etiquetes per tal de que es corresponguin amb les espècies reals, i afegir els noms de les espècies a aquestes etiquetes. Ajustant l'alçada de les fulles i les mides generals, obtenim un gràfic com el següent:



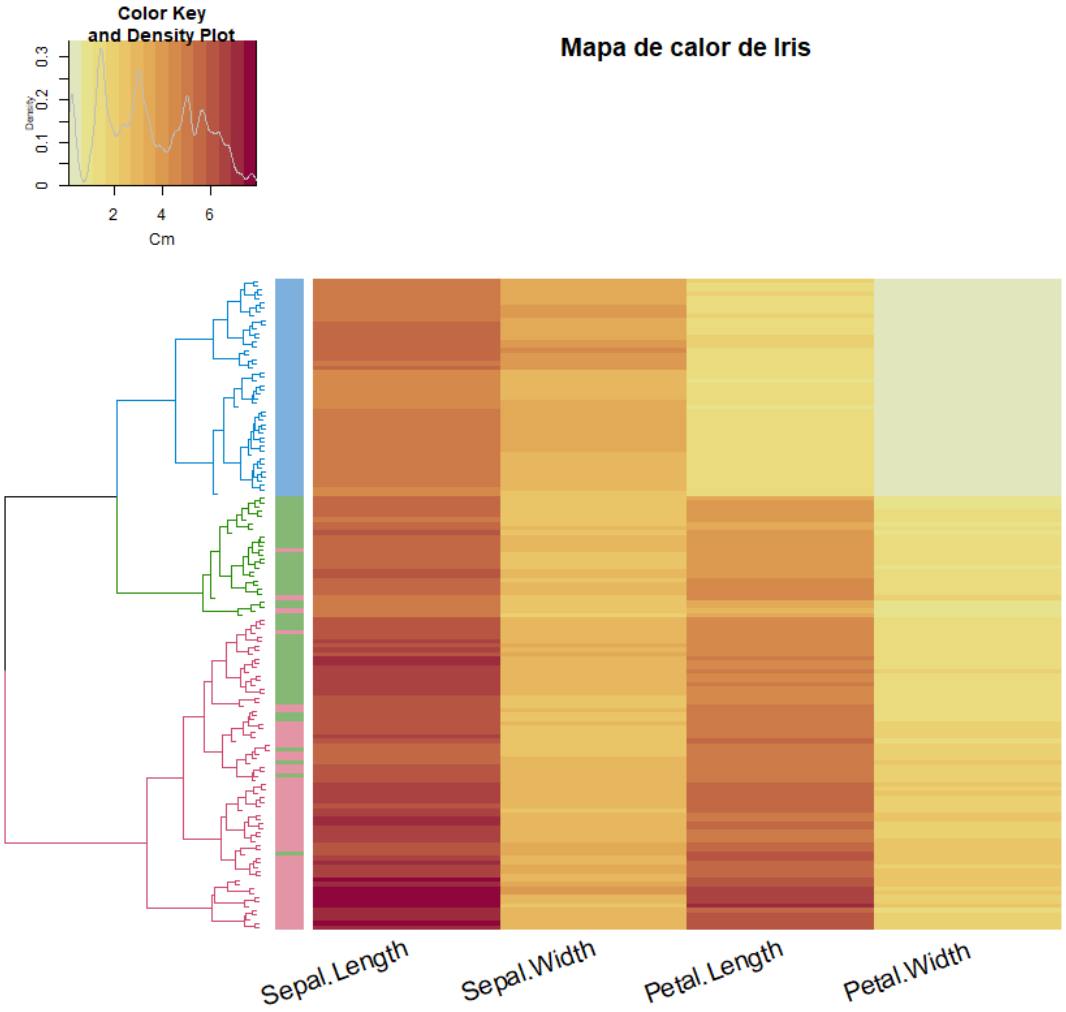
Dendextend també ens permet veure el mateix dendrograma en forma circular, amb la

funció *circlize_dendrogram*. Aquesta funció requereix adicionalment el paquet "circlize".



En aquestes visualitzacions observem que el clústering jeràrquic classifica molt bé les observacions de l'espècie setosa, però s'equivoca bastant amb les altres dues: veiem que bona part de les orbservacions versicolor acaben al clúster de virginica. Amb l'escalat de les alçades de les fulles podem trobar fàcilment casos extrems, com l'observació "virginica (107)", que no és gaire semblant a les oservacions versicolor, i tot i així acaba classificada entre elles. També podem observar que "versicolor (71)" acaba classificada al mig del grup de virginica, bastant allunyada de la resta de la seva espècie.

Una altra forma de visualitzar les dades és mitjançant un mapa de calor, que podem generar amb la funció *heatmap*.2. Les columnes d'aquest gràfic estan ordenades seguint l'ordre del dendrograma. A la barra esquerra (amb 3 colors) veiem l'espècie a la que pertany cada observació, i els colors del propi mapa de calor indiquen la mesura de cada atribut (vermell per valors alts, groc per valors baixos):



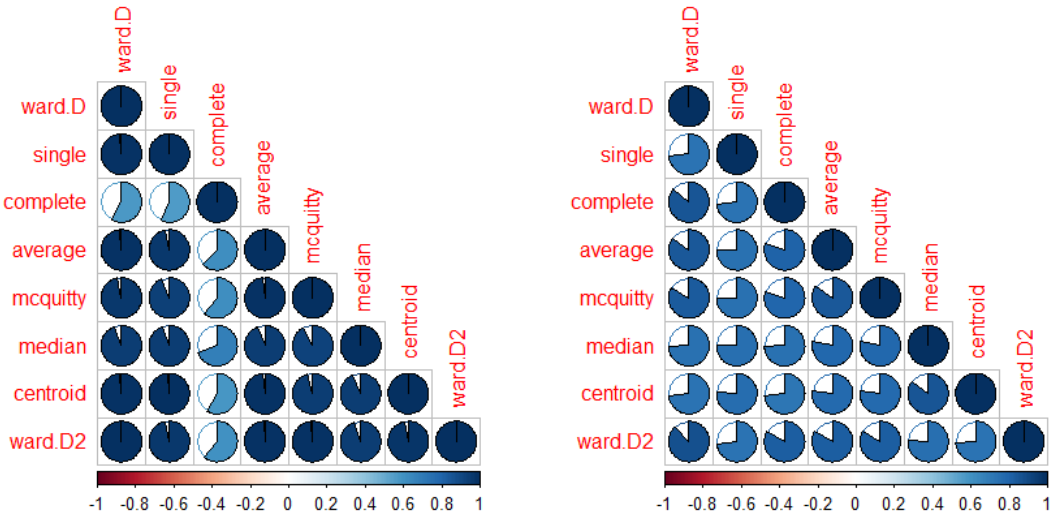
Podem observar, per exemple, com les observacions de l'espècie setosa té valors baixos per a les mides dels pètals, i distingim clarament la frontera amb les altres dues espècies.

Una altra funció interessant que trobem a "dendextend" és *cor.dendlist*, que ens permet calcular un coeficient de correlació entre diferents clústerings, per comparar com de similars són els resultats obtinguts. Realizem un experiment comparant els resultats que podem obtenir amb els 8 possibles mètodes d'enllaç disponibles per a *hclust*. Per fer-ho, podem guardar tots els resultats en un objecte de tipus *dendlist*. Obtenim els següents resultats:

	ward.D	single	complete	average
ward.D	1.0000000	0.9836838	0.5774013	0.9841333
single	0.9836838	1.0000000	0.5665529	0.9681156
complete	0.5774013	0.5665529	1.0000000	0.6195121
average	0.9841333	0.9681156	0.6195121	1.0000000
mcquitty	0.9641103	0.9329029	0.6107473	0.9828015
median	0.9451815	0.9444723	0.6889092	0.9449422
centroid	0.9809088	0.9903934	0.5870062	0.9801444
ward.D2	0.9911648	0.9682507	0.6096286	0.9895131

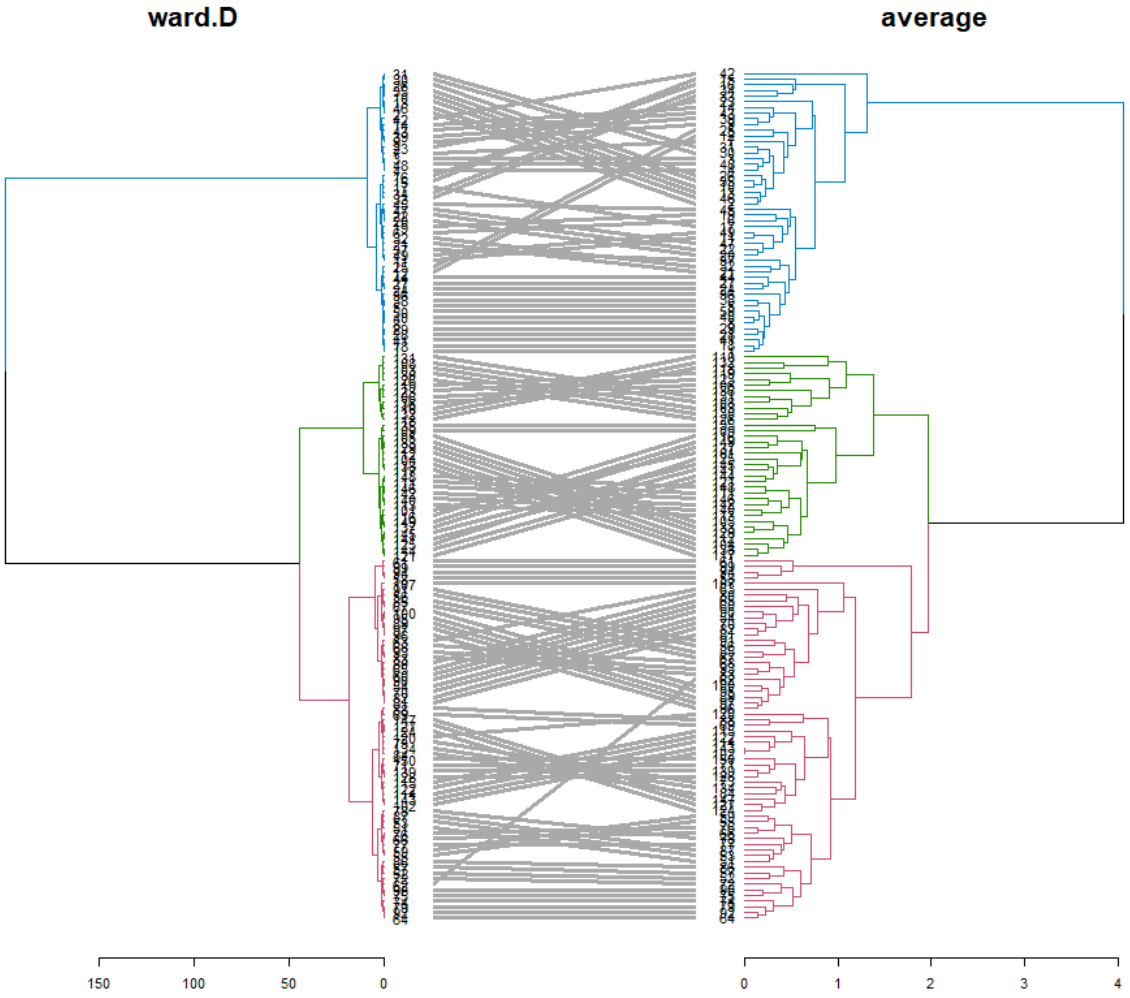
	mcquitty	median	centroid	ward.D2
ward.D	0.9641103	0.9451815	0.9809088	0.9911648
single	0.9329029	0.9444723	0.9903934	0.9682507
complete	0.6107473	0.6889092	0.5870062	0.6096286
average	0.9828015	0.9449422	0.9801444	0.9895131
mcquitty	1.0000000	0.9203374	0.9499123	0.9829977
median	0.9203374	1.0000000	0.9403569	0.9445832
centroid	0.9499123	0.9403569	1.0000000	0.9737886
ward.D2	0.9829977	0.9445832	0.9737886	1.0000000

Aquesta funció calcula per defecte la correlació de Pearson, però també pot calcular altres valors. Podem, per exemple, comparar els nodes comuns a cada clústering, cridant al mètode *common_nodes*. Podem representar gràficament de forma bastant clara aquestes matrius de coeficients usant la funció *corrplot* del paquet amb al mateix nom. Veiem a l'esquerra el gràfic corresponent al coeficient de Pearson, i a la dreta la comparativa de nodes comuns (en percentatge):



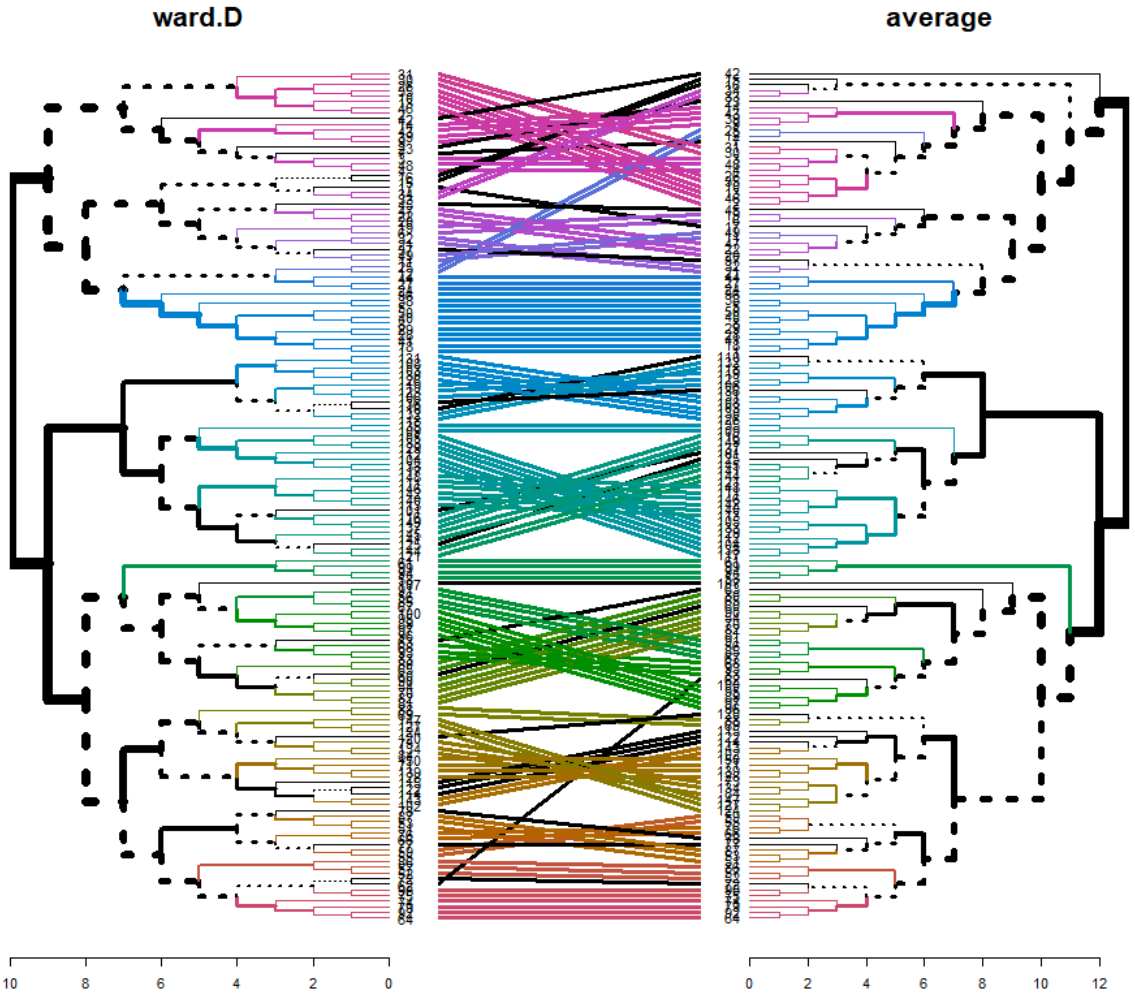
Observant tant la taula com el diagrama, podem concluir que segons el coeficient de Pearson els resultats obtinguts són molt similars entre la majoria de mètodes, amb correlacions superiors a 0.9. L'excepció és l'enllaç complet, les correlacions del qual oscil·len al voltant de 0.6. Si observem el gràfic dels nodes comuns, en canvi, veiem que la majoria de mètodes comparteixen al voltant del 75% dels nodes, amb alguns mètodes concrets superant el 85%.

Per seguir estudiant les similituds i diferències entre els diferents mètodes de clústering podem usar la funció *tanglegram*, que compara directament dos dendrogrames de forma gràfica. Veiem-ho tot comparant per exemple els arbres de "ward.D" i "average", que hem vist que tenen un coeficient de correlació molt alt:



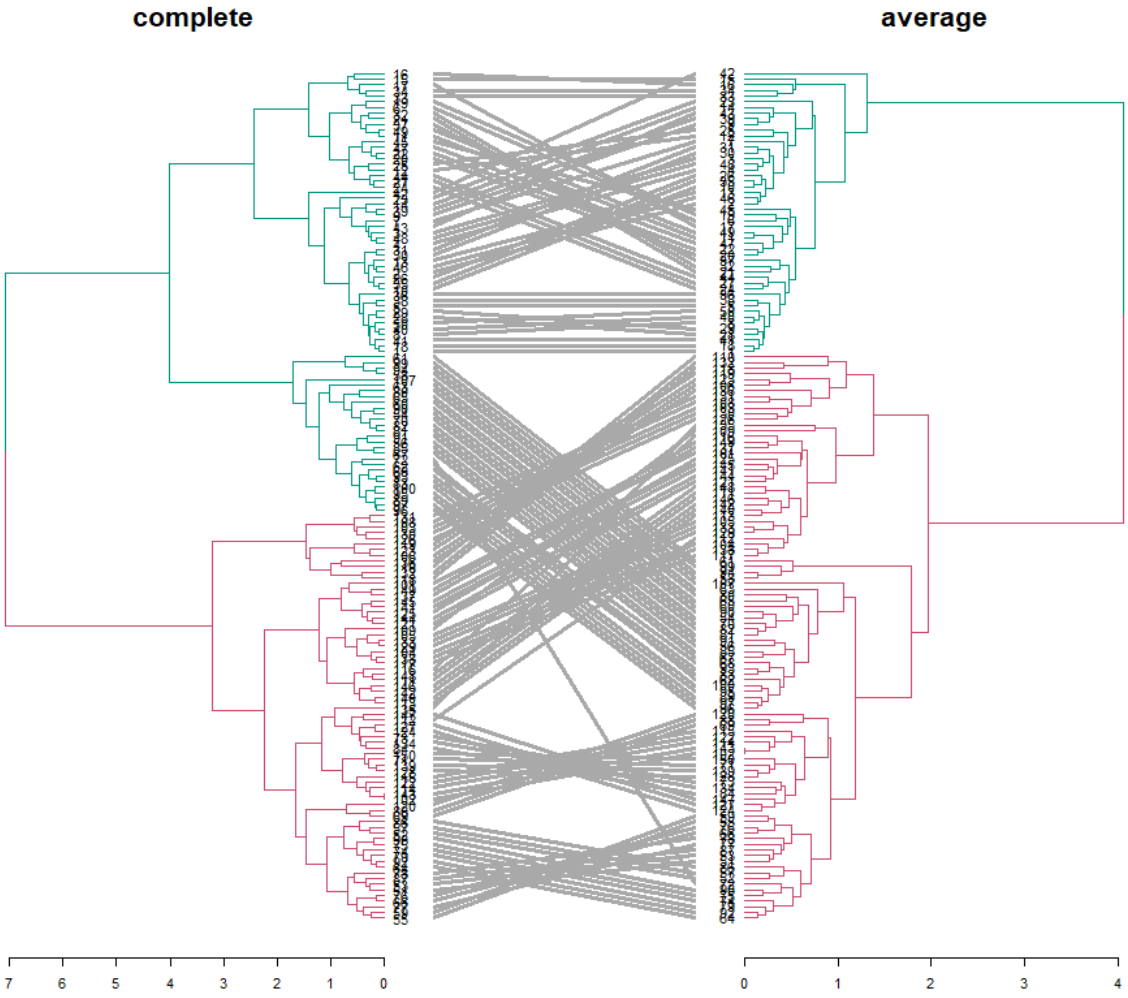
Les línies centrals del gràfic uneixen les posicions de cada observació a cadascun dels dos dendrogrames. Veiem que els clústerings no són idèntics, ja que hi ha creuaments. Podem observar clarament que els dos algorismes produeixen un resultat similar pel que fa als clústers majors: les dues branques principals coincideixen, i el tall en 3 clústers també és el mateix. Comparar els resultats comparar els resultats en els clústers més petits és més complicat, ja que el dendrograma de Ward és molt més pla que el de l'enllaç mitjà.

Per realitzar una comparació més profunda, podem indicar amb diferents colors les branques comunes entre els dos dendrogrames, i amb línies puntejades les arestes (unions entre nodes de l'arbre) que no coincideixen. A més, podem escalar l'alçada de les branques per veure millor l'estructura dels clústers més petits.

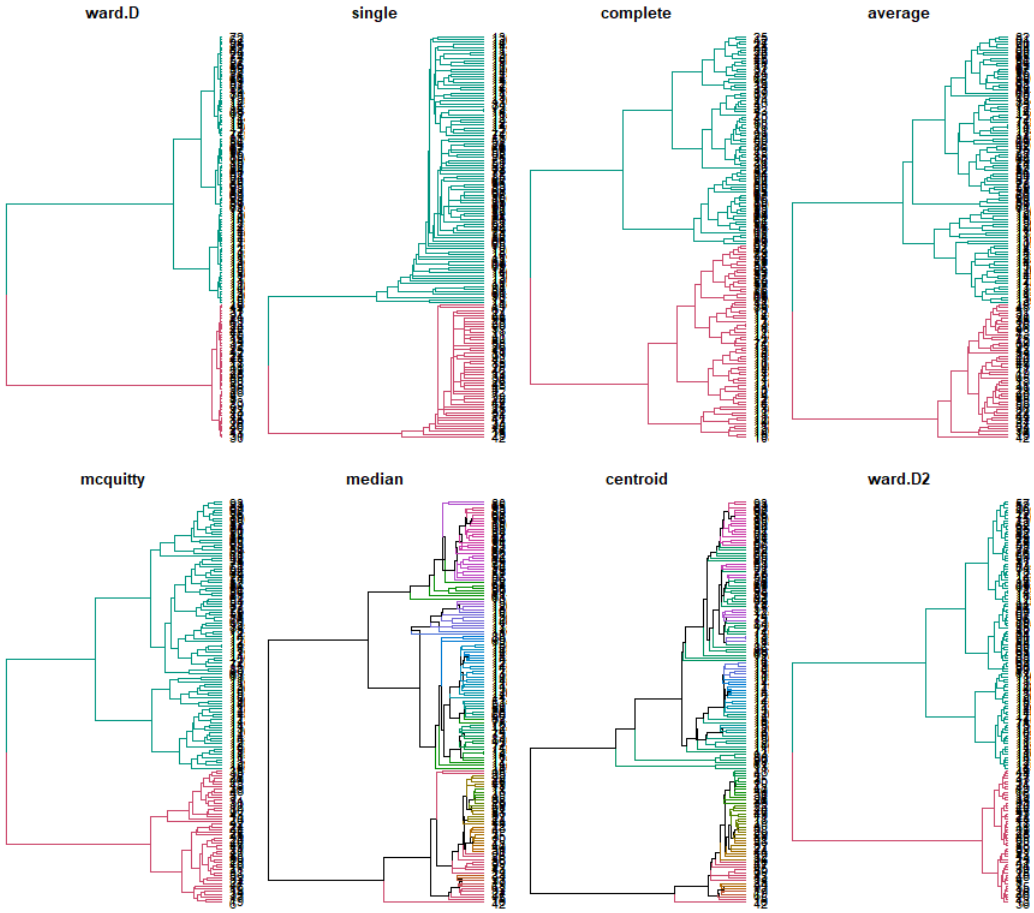


Trobem un total de 39 sub-arbres idèntics entre els dos dendrogrames, indicats cadascun amb un color diferent. Si ens fixem en les línies puntejades, veiem que els dos algorismes semblen tenir resultats prou diferents en les branques més altes, però com tots dos aconsegueixen dividir els dos clústers majors (el que correspon a l'espècie setosa i el que correspon a les altres dues), el coeficient de correlació els considera molt similars.

En canvi, hem vist que el mètode de l'enllaç complet té un coeficient de correlació menor amb tota la resta de mètodes. Comparem el seu dendrograma amb el de l'enllaç mitjà, per exemple, per veure si en podem determinar la causa:

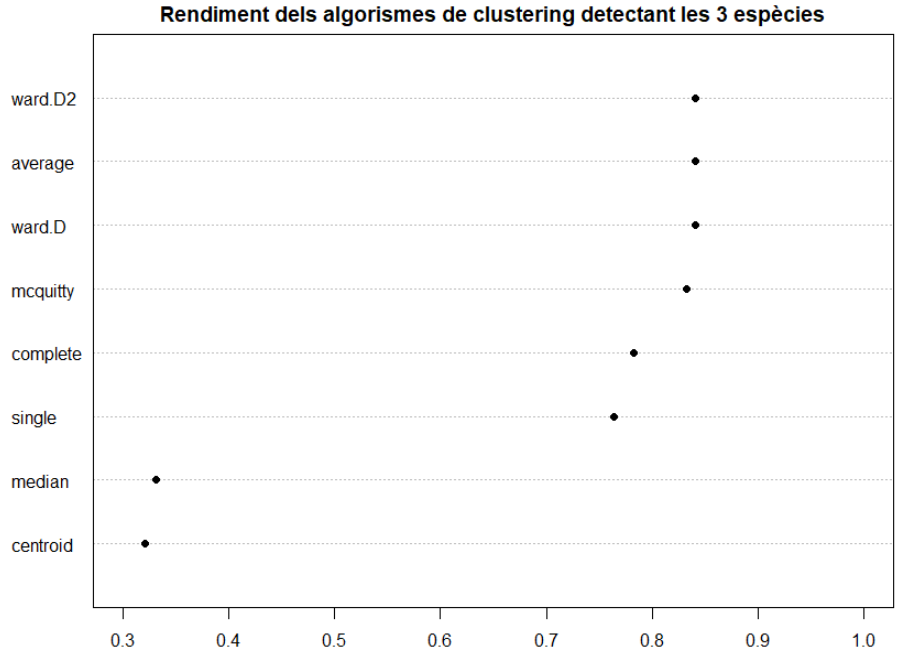


Hem vist en la comparació anterior que de les dues branques principals dels dendrogrames, la més petita (que podem associar a les observacions de setosa) era molt similar en tots dos dendrogrames. En aquest cas, però, veiem que l'enllaç complet dóna lloc a una branca considerablement més gran, és a dir, el tall del dendrograma en 2 clústers difereix de la resta. Si representem els 8 dendrogrames, podem comprovar que aquesta diferència és única del mètode de l'enllaç complet, en la resta de mètodes aquesta classificació coincideix:



Amb aquests resultats, sembla que en el coeficient de correlació influeix fortament la classificació dels clústers principals, els més grans del dendrograma.

Per acabar, podem intentar veure quin dels mètodes de clústering jeràrquic s'ha apropat més a classificar correctament les 3 espècies de plantes, usant un tall del dendrograma de $k = 3$. Per fer-ho, podem comparar els clústerings resultants de cada algorisme amb els clústers reals, usant un índex de validació extern (ja que en aquest cas tenim dades etiquetades). Per exemple, "dendextend" ens permet calcular l'índex de Fowlkes-Mallows, una mesura molt similar a l'índex de Rand, que dóna valors entre 0 i 1, on un valor proper a 1 indica que el clústering es correspon amb l'etiquetació real. Comparem els resultats en un gràfic:



Observem que el mètode de Ward "ward.D2" obté el millor resultat, tot i que "ward.D" i l'enllaç mitjà obtenen puntuacions similars. Els mètodes de la mediana i el centroide, en canvi, mostren un resultat bastant pobre.

7 Conclusions

En aquesta memòria hem estudiat el funcionament i les característiques dels principals mètodes de clústering, amb els avantatges i inconvenients de cadascun. Hem vist com els algorismes de particions es defineixen de forma senzilla a partir d'un problema d'optimització, i hem definit els algorismes jeràrquics veient que podem establir una equivalència entre un espai ultramètric i una estructura de jerarquia de forma prou natural. Hem parlat de la dificultat de plantejar els algorismes jeràrquics a partir d'una funció global, i de com aquesta és probablement la causa de que els estudis de caire teòric d'aquest tipus d'algorisme hagin quedat endarrerits en comparació als particionals. Tot i així, hem trobat alguns articles dels darrers anys que intenten acurtar aquesta diferència proposant possibles funcions globals per al cas jeràrquic.

Hem analitzat els problemes computacionals del clústering jeràrquic aplicat a conjunts grans de dades, i hem vist que l'ús de l'algorisme base en aquests casos és gairebé impossible, tant en temps d'execució com en requeriments de memòria. Hem vist, però, que s'han proposat algorismes que busquen resoldre aquest problema, aconseguint millores notables de rendiment. Hem trobat que alguns d'aquests algorismes, tot i tenir un rang limitat d'aplicació, serveixen per realitzar estudis concrets, com és el cas d'algunes aplicacions en genòmica, una de les disciplines on hi ha un interès major pel cústering jeràrquic amb grans dades.

Per acabar, hem vist alguns exemples d'aplicacions del clústering, a través d'una sèrie d'estudis senzills sobre dades en R, que il·lustren bé l'avantatge principal dels mètodes jeràrquics de classificació: podem extreure informació de forma simple i intuïtiva d'una bona visualització de dades.

Referències

[1] L. Bottou, Y. Bengio, 1995: *Convergence Properties of the K-Means Algorithms*, Conference: Advances in Neural Information Processing Systems 7, Denver.

[2] E. Chatziafratis, 2020: *Clustering with global objectives: approximation algorithms and hardness results*, Stanford University

[3] V. Cohen-Addad et al., 2019: *Hierarchical Clustering: Objective Functions and Algorithms*, Journal of the ACM, Volume 66, 4, article 26

[4] C.M. Cuadras, 2014: *Nuevos Métodos de Análisis Multivariante*, CMC Editions, Barcelona.

[5] S. Dasgupta, 2016: *A Cost Function for Similarity-Based Hierarchical Clustering*. In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC'16)*, ACM, New York, pp.118-127

[6] D.L. Davies, D.W. Bouldin, 1979: *A Cluster Separation Measure*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume: PAMI-1, pp.224-227.

[7] J.C. Dunn, 1974: *Well-Separated Clusters and Optimal Fuzzy Partitions*, Journal of Cybernetics, Volume 4, pp.95-104.

[8] M. Embrechts et al., 2013: *Hierarchichal Clustering for Large Data Sets*. In: P. Georgieva et al.: *Advances in Intelligent Signal Processing and Data Mining: Theory and Applications*, Springer, Berlin Heidelberg, pp.197-233.

[9] T. Galili, 2015: *dendextend: an R package for visualizing, adjusting, and comparing trees of hierarchical clustering*, https://academic.oup.com/bioinformatics/article/31/22/3718/240978/dendextend-an-R-package-for-visualizing-adjusting

[10] W. Hendrix et al., 2013: *A Scalable Algorithm for Single-Linkage Hierarchical Clustering on Distributed-Memory Architectures*, IEEE Symposium on Large Data Analysis and Visualization 2013, Atlanta.

[11] A.K. Jain, R.C. Dubes, 1988: *Algorithms for clustering data*, Prentice-Hall Inc, New Jersey.

[12] G. James et al., 2013: *An Introduction to Statistical Learning: with applications in R*, Springer, New York.

[13] L. Kaufman, P.J. Rousseeuw, 1990: *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley Sons, New Jersey.

[14] T. Kodali 2015: *K Means Clustering in R*, https://www.r-bloggers.com/2015/12/k-means-clustering-in-r/, 15/1/2020

[15] H. Koga et al. 2007: *Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing*, Knowledge and Information Systems, 12, pp.25-53.

[16] N. Kriege et al., 2014: *Practical SAHN Clustering for Very Large Data Sets and Expensive Distance Metrics*, Journal of Graph Algorithms and Applications, Volume 18, pp.577-602.

[17] U. Kutbay et al., 2018: *Partitional Clustering*. In: H. Pirim: *Recent Applications in Data Clustering*, IntechOpen, pp.19-33.

[18] S. Kwon, C. Han, 2002: *Hybrid Clustering Method for DNA Microarray Data Analysis*, Genome Informatics, Volume 13, pp.258-259.

[19] Z. Li et al., 2007: *An Adaptive Parallel Hierarchical Clustering Algorithm*. In: R. Perrott et al.: *High Performance Computing and Communications*, Springer, Berlin Heidelberg, pp.97-107.

[20] H.-S. Park, C.-H. Jun, 2009: *A simple and fast algorithm for K-medoids clustering*, Expert Systems with Applications, Volume 36, pp.3336-3341.

[21] Perception Analytics (A. Singh et al.), 2017: *How to Perform Hierarchical Clustering using R*, https://r-posts.com/how-to-perform-hierarchical-clustering-using-r/, 11/1/2020

[22] W.M. Rand, 1971: *Objective Criteria for the Evaluation of Clustering Methods*, Journal of the American Statistical Association, Volume 66, pp.846-850.

[23] R Core Team, 2020: *R: A Language and Environment for Statistical Computing*, Viena, https://www.R-project.org