Clasificación Jerárquica Ascendente con *Mathematica*

Carlos Arce, Javier Trejos
CIMPA-PIMAD
Escuela de Matemática
Universidad de Costa Rica

Resumen

Hemos desarrollado el algoritmo usual de clasificación jerárquica ascendente en el sistema MATHEMATICA. El usuario escoge la disimilitud según el tipo de datos que deba analizar: cuantitativos, cualitativos o binarios, así como el índice de agregación a utilizar. Se dispone de varias opciones para cada escogencia. Además, se ha implementado un gran número de manipulaciones sobre el árbol binario de clasificación, como el corte del árbol, la rotaciones, la dimensionalidad, el etiquetado, los colores, etc.

Abstract

We have developped the usual hierarchical agglomerative clustering algorithm on MATHEMATICA. The user chooses the dissimilarity index according to the type of data at hand: numerical, categorical or binary; he also has the choice of the aggregation criterion to be used. Many options for each case are available. Moreover, we have implemented a large number of manipulations on the binary tree of classification, such as the cut level, rotations, dimensionality, labeling, coloring, etc.

Résumé

Nous avons développé l'algorithme usuel de classification ascendante hiérarchique dans le système MATHEMATICA. L'utilisateur choisit le dissimilarité selon le type de données qu'il traite: quantitatives, qualitatives ou binarias, ainsi que l'indice d'agrégation à utiliser. Plusieurs options pour chaque choix sont disponibles. En plus, on a implémenté un grand nombre de manipulations sur l'arbre binaire de classification, tels que la coupure de l'arbre, les rotations, le dimensionnement, l'étiquetage, le coloriage, etc.

Matrices de disimilitudes

El primer paso para aplicar `HierarchicalTree`, el principal procedimiento en `HierarchicalCluster`, es disponer de una matriz de disimilitudes, que puede ser calculada con el comando `Dissimilarity[...]`, a partir de una tabla de datos -individuos $\times$ variables- con la posibilidad de escoger entre una gran diversidad de disimilitudes.

Disimilitudes

Definición 1 *Si Ω es un conjunto de objetos, una **disimilitud** es una función $d : \Omega \times \Omega \to \mathbb{R}^+$ tal que:*

$$d(i,j) = 0 \Leftrightarrow i = j,$$
$$\forall i,j \in \Omega : d(i,j) = d(j,i).$$

Dada una tabla de datos de n individuos y p variables, y una disimilitud d definida sobre los individuos, se determina una matriz de disimilitudes, notada también d, por:

$$d = \{\{d(1,2), d(1,3), d(1,4), \ldots, d(1,p)\}, \{d(2,3), d(2,4), \ldots, d(2,p)\}, \{d(3,4), \ldots, d(3,p)\}, \ddots \vdots \{d(n-1,p)\}\} \quad (1)$$

La biblioteca `Dissimilarity` tiene un gran número de funciones de disimilitud, que pueden ser aplicadas para obtener una tabla de disimilitudes. Estas funciones están definidas dependiendo de si las variables de la tabla de datos son binarias o cuantitativas, o si es una tabla de contingencia. Para esto conviene reconocer ciertos conceptos básicos para trabajar con matrices en MATHEMATICA.

Lectura de la tabla de datos

MATHEMATICA permite la lectura de varios tipos de formatos; sin embargo, los más fáciles para manipular son los de tipo texto. Así, convenimos en que un archivo tiene el **formato apropiado** de lectura para MATHEMATICA, cuando: a) es un archivo texto, b) los números de cada fila de la tabla están separados por espacios en blanco, y c) cada fila termina por el caracter Intro.

En esta situación, si `datos.txt` es el nombre de un archivo que contiene una tabla de n individuos y p variables, grabado en el directorio `C:\Ejemplo`, la orden:

```
X=ReadList[C:\Ejemplo\datos.txt,Record->True];
```

crea la matriz X, que contiene la tabla de datos leída, como matriz, bajo la forma:

$$X = \{\{x_{11}, x_{12}, \ldots, x_{1p}\}, \{x_{21}, x_{22}, \ldots, x_{2p}\}, \vdots \vdots \vdots \{x_{n1}, x_{n2}, \ldots, x_{np}\}\} \quad (2)$$

Podemos notar que en esta matriz los valores de las p variables medidas sobre un individuo i forman una lista $x_i = \{x_{i1}, x_{i2}, \ldots, x_{ip}\}$.

Disimilitudes sobre tablas de datos binarias

En este caso se supone que los valores que toman las variables de la tabla son solamente 0 ó 1 (ausencia, presencia, respectivamente), es decir $x_{ij} = 0$ ó 1. Dadas las filas i y j de la tabla, $x_i = \{x_{i1}, x_{i2}, \ldots, x_{ip}\}$ y $x_j = \{x_{j1}, x_{j2}, \ldots, x_{jp}\}$, para calcular $d(x_i, x_j)$ se calculan los valores:

$$
\begin{array}{cc} & j \\ i & \begin{array}{|c|cc|} \hline & 1 & 0 \\ \hline 1 & a_{ij} & b_{ij} \\ 0 & c_{ij} & d_{ij} \\ \hline \end{array} \end{array}
$$

donde a_{ij} (respectivamente d_{ij}) es el número de variables tales que i y j tienen al mismo tiempo el valor 1 (resp. 0), y b_{ij} (resp. c_{ij}) es tal que i vale 1 (resp. 0) y j vale 0 (resp. 1). Además, $n_i = a_{ij} + b_{ij}$ y $n_j = a_{ij} + c_{ij}$.

La tabla 1 muestra la definición de las funciones de disimilitud sobre tablas binarias disponibles en `Dissimilarity`; su nombre está formado por el nombre de uno o varios de los autores del índice.

Tabla 1: Indices de disimilitud para tablas que contienen variables binarias.

	Disimilitud	$d(x_i, x_j)$
1	`Jaccard`	$1 - \frac{a_{ij}}{a_{ij} + b_{ij} + c_{ij}}$
2	`Dice`	$1 - \frac{2a_{ij}}{2a_{ij} + b_{ij} + c_{ij}}$
3	`SokalMichener`	$1 - \frac{a_{ij} + d_{ij}}{p}$
4	`RusselRao`	$\begin{cases} 0 & \text{si } i = j \\ 1 - \frac{a_{ij}}{p} & \text{si } i \neq j \end{cases}$
5	`Kulczinski1`	$1 - \frac{a_{ij}(1/n_i + 1/n_j)}{2}$
6	`Ohiai`	$1 - \frac{a_{ij}}{\sqrt{n_i n_j}}$
7	`SokalSneath1`	$1 - \frac{2(a_{ij} + d_{ij})}{2(a_{ij} + d_{ij}) + b_{ij} + c_{ij}}$
8	`SokalSneath2`	$1 - \frac{a_{ij}}{a_{ij} + 2(b_{ij} + c_{ij})}$

9	Yule	$1-\frac{\lvert a_{ij}d_{ij}-b_{ij}c_{ij}\rvert}{a_{ij}d_{ij}+b_{ij}c_{ij}}$
10	AngularDistance	$1-\frac{\lvert a_{ij}d_{ij}-b_{ij}c_{ij}\rvert}{\sqrt{(a_{ij}+b_{ij})(c_{ij}+d_{ij})(a_{ij}+c_{ij})(b_{ij}+d_{ij})}}$
11	RogersTanimoto	$1-\frac{a_{ij}+d_{ij}}{a_{ij}+d_{ij}+2(b_{ij}+c_{ij})}$

Disimilitudes sobre tablas de datos con variables cuantitativas

Si todas las variables de la tabla de datos son cuantitativas se puede escoger entre las disimilitudes mostradas en la tabla 2, donde se usan las notaciones usuales para la varianza de la variable k (σ_k^2), la matriz de varianzas V, las ponderaciones p_k para las variables, y donde r es un número real positivo.

Tabla 2: Indices de disimilitud para tablas que contienen variables cuantitativas.

	Disimilitud	$d(x_i,x_j)$
1	EuclideanDistance	$\sqrt{\sum_{k=1}^{p}(x_{ik}-x_{jk})^2}$
2	VarianceInverse	$\sqrt{\sum_{k=1}^{p}(x_{ik}-x_{jk})^2/\sigma_k^2}$
3	Mahalanobis	$\sqrt{(x_i-x_j)^tV^{-1}(x_i-x_j)}$
4	Manhattan	$\sum_{k=1}^{p}p_k\lvert x_{ik}-x_{jk}\rvert$
5	Chebychev	$\text{máx}_k\lvert x_{ik}-x_{jk}\rvert$
6	Minkowski[r]	$(\sum_{k=1}^{p}\lvert x_{ik}-x_{jk}\rvert^r)^{1/r}$
7	Camberra	$\sum_{k=1}^{p}\frac{\lvert x_{ik}-x_{jk}\rvert}{x_{ik}+x_{jk}}$

Disimilitudes sobre tablas de contigencia

Si la tabla `X` es una tabla de contingencia o es asimilable a una de estas tablas, se puede emplear la distancia de Chi-cuadrado mostrada en la tabla 3 como índice de disimilitud, donde se usan las notaciones usuales para los márgenes de las filas $x_{i.}$ y las columnas $x_{.k}$.

Tabla 3: Indice de disimilitud para tablas de contingencia.

	Disimilitud	$d(x_i, x_j)$
1	`ChiSquare`	$\sqrt{\sum_{k=1}^{p}(\frac{x_{ik}}{x_{i.}} - \frac{x_{jk}}{x_{j.}})^2 / x_{.k}}$

Cálculo de la matriz de disimilitudes

Si `X` es una matriz de datos, que resulta de una orden como

```
X = ReadList[Datos.txt,Record->True]
```

o bien definida por la captura de datos en un editor de texto, la orden

```
d = Dissimilarity[X,Jaccard]
```

crea una matriz de disimilitudes `d`, como en (1), dada por el uso del índice de disimilitud de Jaccard sobre las filas de la tabla `X`. Naturalmente, esto supone que `X` es una tabla binaria.

Si `X` es una matriz de datos con variables continuas, la instrucción

```
d = Dissimilarity[X,Mahalanobis]
```

crea una matriz de disimilitudes `d`, usando la distancia de Mahalanobis.

Se debe hacer notar que el procedimiento `Dissimilarity` no verifica si la tabla de datos dada sea del tipo apropiado al índice.

Agregaciones

Además de la disimilitud, `HierarchicalTree` necesita que se especifique un índice de agregación con el fin de construir la jerarquía. El índice de agregación δ es una disimilitud definida sobre el conjunto de partes de Ω, sin que necesariamente $\delta(A, A) = 0$, para todo $A \subset \Omega$.

Tabla 4: Indices de agregación.

	Agregación	$\delta(h_1, h_2)$
1	`MinDistance`	$\min\{d(x_i, x_j) \mid x_i \in h_1, x_j \in h_2\}$
2	`MaxDistance`	$\max\{d(x_i, x_j) \mid x_i \in h_1, x_j \in h_2\}$
3	`MeanDistance`	$\frac{1}{\mu(h_1)\mu(h_2)} \sum_{x_i \in h_1, x_j \in h_2} p_i p_j d(x_i, x_j)$
4	`GravityCenter`	$d^2(g_{h_1}, g_{h_2})$
5	`Ward`	$\frac{\mu(h_1)\mu(h_2)}{\mu(h_1)+\mu(h_2)} d^2(g_{h_1}, g_{h_2})$
6	`IncreaseVariance`	$\mathrm{var}(h_1 \cup h_2) - \frac{\mu(h_1)}{\mu(h_1)+\mu(h_2)}\mathrm{var}(h_1) - \frac{\mu(h_2)}{\mu(h_1)+\mu(h_2)}\mathrm{var}(h_2)$
7	`JoinInertia`	$I(h_1 \cup h_2)$
8	`JoinVariance`	$\mathrm{var}(h_1 \cup h_2)$

Las agregaciones que se encuentran en `HierarchicalCluster` se muestran en la tabla 4. En estas definiciones, la inercia de una clase h es la inercia respecto a su centro de gravedad:

$$I(h) = \sum_{x_i \in h} p_i d^2(x_i, g_h),$$

donde p_i es el peso asociado al individuo x_i y $\mu(h)$ es el peso del conjunto $h \in \mathcal{P}(\Omega)$. Se define también $\mathrm{var}(h) = I(h)/\mu(h)$ como la varianza de h.

En la implementación de las agregaciones anteriores, se supone que $p_i = 1/n$ y $\mu(h) = \frac{\text{card}(h)}{n}$, y se usa la fórmula de Lance, Williams & Jambu:

$$\begin{aligned}\delta(h_1 \cup h_2, h) &= a_1\delta(h_1,h) + a_2\delta(h_2,h) + a_3\delta(h_1,h_2) \\ &\quad + a_4 f(h) + a_5 f(h_1) + a_6 f(h_2) \\ &\quad + a_7|\delta(h,h_1) - \delta(h,h_2)|.\end{aligned}$$

Cada función `MinDistance, MaxDistance, MeanDistance, GravityCenter, IncreaseVariance, Ward, JoinInertia`, y `JoinVariance` está definida como una función de nueve parámetros: las cardinalidades de h_1, h_2 y h, (denotadas `ca,cb,cc`), las agregaciones $\delta(h_1,h_2)$, $\delta(h_1,h)$ y $\delta(h_2,h)$ (denotadas `dab, dac, dbc`) y los índices $f(h_1)$, $f(h_2)$ y $f(h)$ (denotados `fa,fb,fc`). Por ejemplo, en el caso de la agregación de `Ward`, su definición es:

```
Ward[ca_,cb_,cc_,dab_,dac_,dbc_,fa_,fb_,fc_] :=
With[{abc = ca+cb+cc}, ((ca+cc)dac + (cb+cc)dbc - cc dab)/abc]
```

`HierarchicalTree` acepta como agregación cualquier función de los nueve parámetros mencionados arriba; así, el usuario puede especificar nuevos índices de agregación siempre que respete el orden de los parámetros y que la evaluación sea un número positivo.

Jerarquías indexadas

Se considera $\Omega = \{1,\ldots,n\}$ el conjunto de los n objetos a clasificar. Se denota $\mathcal{P}(\Omega)$ el conjunto de partes de Ω.

Definición 2 $H \subset \mathcal{P}(\Omega)$ *es una* ***jerarquía*** *de objetos sobre* Ω *si*

1. $\Omega \in H$ y $\emptyset \notin \Omega$.
2. $\forall\, i \in \Omega, \{i\} \in H$.
3. $\forall\, h_1, h_2 \in H$, se tiene $h_1 \cap h_2 = \emptyset$ ó $h_1 \subset h_2$ ó $h_2 \subset h_1$.

Si además

4. $\forall\, h \in H$ no unitario, existen $h_1, h_2 \in H$ tales que $h_1 \cap h_2 = \emptyset$ y $h_1 \cup h_2 = h$,

se dice que H *es una* ***jerarquía binaria****.*

La pareja (H, f) *es una* ***jerarquía indexada*** *si* H *es una jerarquía y* $f : H \to \mathbb{R}^+$ *satisface:*

1. $\forall\, i \in \Omega, f(\{i\}) = 0$,
2. $\forall\, h_1, h_2 \in H, h_1 \subset h_2 \Rightarrow f(h_1) < f(h_2)$.

Los elementos de una jerarquía H *se llaman* ***clases***.

Representación de jerarquías binarias

Para construir y manipular jerarquías binarias con los procedimientos de `HierarchicalCluster` en MATHEMATICA se ha escogido la forma de representación siguiente. Una jerarquía binaria indexada (H, f) será representada por una lista estructurada como un árbol binario, denotado $N(H)$, y tal que cada clase $h \in H$ es asociada a un nodo, $N(h)$, definido como sigue.

Definición 3 *Se define el* ***nodo*** $N(h)$ *asociado a la clase* i *por:*

1. Para cada $i \in \Omega$, $N(\{i\}) = i$.
2. Si $h \in H$ es el resultado de unir las clases h_1 y h_2 de H, en el paso k del algoritmo de clasificación jerárquica ascendente (cf. 3.2 más abajo), entonces

$$N(h) = \{f(h) + kI, N(h_1), N(h_2)\}$$

donde I *es la unidad imaginaria en* MATHEMATICA.

Se puede notar que esta definición es recursiva y que:

- Los conjuntos unitarios de H, $\{i\}$, se representan por los símbolos o enteros i de Ω. Estos nodos se llaman las hojas del árbol $N(H)$.
- Cada nodo $N(h)$ del árbol, diferente de una hoja,

$$N(h) = \{f(h) + kI, N(h_1), N(h_2)\}$$

forma una rama cuyas hojas forman los elementos de la clase h y su raíz (o cabeza) es el número complejo $f(h) + kI$[3] cuya parte real es $f(h) = \delta(h_1, h_2)$ y la parte imaginaria es k, es decir el nivel en el cual se formó el nodo durante la unión de las clases h_1 y h_2.

- $N(\Omega) = N(H)$.

Algoritmo de clasificación jerárquica ascendente

La construcción de un árbol que representa a una jerarquía indexada, con el algoritmo de clasificación jerárquica ascendente (CJA), es un proceso recursivo que comienza con:

a)
la lista $\{1, 2, \ldots, n\}$ de enteros o símbolos con los cuales se representan los elementos de Ω, y la representación por lista de los nodos de la partición en los conjuntos unitarios de Ω;

b)
una matriz de disimilitudes d entre los objetos de Ω; y

c)
un criterio de agregación δ,

y produce en cada paso una nueva partición de Ω, por unión de las clases más cercanas de la partición anterior:

Paso 1
(Inicialización)

1. $k := 0$
2. $P_0 = \{1, 2, \ldots, n\}$ es la lista de los nodos que representan a las clases de la partición inicial {{1},{2},..., {n}}.

3. $$d_0 = \{\{d(1,2), d(1,3), d(1,4), \ldots, d(1,p)\}, \{d(2,3), d(2,4), \ldots, d(2,p)\}, \{d(3,4), \ldots, d(3,p)\}, \ddots \vdots \{d(n-1,p)\}\}$$

4. δ es una función de agregación.

Paso 2
Para $k = 1$ hasta $n - 1$:

1. Se determina (i_0, j_0) tal que $d(i_0, j_0) = \min\{d_{k-1}(i,j) | i = 1, \ldots, n-k, j = i+1, \ldots, n-k+1\}$
.

2. Se crea una nueva lista de los nodos P_k, eliminando de P_{k-1} los nodos $P_{k-1}[i_0]$ y $P_{k-1}[j_0]$ y

añadiendo el nodo $\{d(i_0, j_0) + kI, P_{k-1}[i_0], P_{k-1}[j_0]\}$.

3. Se crea una nueva matriz de disimilitudes d_k, eliminando de d_{k-1}, las disimilitudes que corresponden a las clases eliminadas y añadiendo las disimilitudes entre la clase que viene de ser creada y las clases restantes.

Niveles del árbol o jerarquía

El orden de creación de los nodos --del árbol que representa una jerarquía-- por aplicación del algoritmo CJA, define los niveles del árbol o jerarquía, de la manera siguiente:

Definición 4 *El **nivel** 0 de un árbol jerárquico está formado por las hojas (o conjuntos unitarios), que representan la partición inicial. El **nivel** k está formado por la lista de los nodos producidos por el algoritmo aplicando k veces los pasos 2.1, 2.2 y 2.3, es decir, P_k.*

Construcción del árbol asociado a una jerarquía

Si `d` es una matriz de disimilitudes entre n objetos, como en (1), y $P = \{1, 2, \ldots, n\}$ es la lista de las hojas que corresponden a los n objetos, la instrucción

```
A = HierarchicalTree[P,d,Ward]
```

construye el árbol `A`, que está asociado a la jerarquía que resulta de la aplicación del algoritmo CJA, utilizando la agregación de `Ward`. La función de agregación puede ser escogida entre la lista de las agregaciones mostradas en la tabla 4.

Manipulación de las jerarquías

La biblioteca `HierarchicalCluster` ofrece los procedimientos siguientes para la manipulación del árbol `A` asociado a une jerarquía.

- Clases y particiones

Clases y particiones

Para "cortar" un áárbol `A` obtenido con el procedimiento `HierarchicalTree`, o bien para escoger el nivel de corte y conocer las particiones y clases que se forman, se tienen los procedimientos presentados en la tabla 5.

Tabla 5: Procedimientos para cortar un árbol, escoger un nivel de corte y ver las particiones formadas.

Procedimiento	Resultado
`CutTree[A,k]`	Lista de los nodos del árbol `A` al nivel `k`.
`Particion[A,k]`	Lista de las clases que corresponden a los nodos del nivel `k`.
`TreeHeight[A]`	Determina la lista: $\{\{1,f_1\},\{2,f_2\},\ldots,\{n-1,f_{n-1}\}\}$, donde $f_k = f(h_k)$ y h_k es la clase formada al nivel k, por la aplicación del algoritmo CJA para obtener el árbol binario `A`.
`MaxJump[A]`	Si `A` es un áárbol y h_k como antes, con $S_k = f(h_{k+1}) - f(h_k)$, se calcula la lista de las parejas $\{k, S_k\}$ tales que $\forall t < k$, $S_t < S_k$.
`HeightPlot[A]`	Dibuja los puntos: $\{\{1,f_1\},\{2,f_2\},\ldots,\{n-1,f_{n-1}\}\}$, obtenidos por `Alturas` y señala los valores k que corresponden a los saltos maximales, obtenidos con `MaxJump`.

Gráficos de las jerarquías binarias

Dado el árbol `A` asociado a una jerarquía binaria indexada, el procedimiento `TreePlot[A]` construye la representación gráfica de `A`. La sintaxis general es:

```
TreePlot[A,opciones]
```

donde la palabra `opciones` puede representar ninguna, una o muchas especificaciones que modifican la forma del gráfico, y pueden ser escogidas entre las mostradas en lo que sigue. Estas opciones deben ser separadas por una coma.

- Descripción de las opciones:

Descripción de las opciones:

- `Leaf1Position->{x,y}`

  Coordenadas del punto que ocupará el nodo asociado a la primera hoja o elemento de Ω. El valor por defecto es `PosHoja1 ->{0,0}`.

- `LeafDistance->d`

  Distancia de separación entre dos hojas consecutivas. El valor por defecto es `LeafDistance -> 1`.

- `LeafLabelDistance->d`

  Distancia de separación entre una hoja y su etiqueta, bajo forma de porcentaje de la altura del gráfico , ó $f(\Omega)$. El valor por defecto es `LeafLabelDistance -> 0.05`.

- `LeafDirection->Vertical` u `Horizontal`.

  Los elementos de Ω son dados verticalmente u horizontalmente. El `LeafDirection->Horizontal` es el valor por omisión.

- `LevelCut->k`

  Dibuja el árbol partiendo de la raíz hasta los nodos del nivel `k`. Cada clase no unitaria es identificada con un círculo que rodea la cardinalidad de la clase. Éstas aparecen en el mismo orden que las listas `ClusterPartition[A,n-k]`. El valor por defecto es `LevelCut->0`.

- `ClustersNumber-> r`

  Para colorear los nodos que corresponden a la partición de Ω en `r` clases, es decir, los del nivel $n-r$. Con el valor por defecto 0, esta opción no tiene ningún efecto. Si en `ClusterStyle` se especifican menos colores que el número de clases a colorear, se usan estos colores cíclicamente hasta que todas las clases sean coloreadas.

- `ClusterStyle->{Green,Black,...}`

Permite escoger los colores utilizados por `ClustersNumber`. El valor `ClusterStyle->{Red,Blue}`, es el valor asignado por omisión.

- `LinesStyle->{Thickness[0.1], AbsolutePointSize[2] ,...}`

  Establece el estilo global para las líneas y puntos que conforman el gráfico.

- `AspectRatio -> GoldenRatio` ó `->Automatic` ó `->r`.

  El valor `r` es el cociente entre la altura y el ancho del gráfico. Si se usa `Automatic` el programa calcula una proporción `r` apropiada para que las unidades en los ejes de las abscisas y las ordenadas sean las mismas. Con `GoldenRatio` (valor por omisiòn) la proporción es `1/GoldenRatio` $\approx 0{,}618034$ si la orientación de las hojas es horizontal y `GoldenRatio` $\approx 1{,}61803$ si es vertical.

En `TreePlot` se puede también utilizar todas las opciones aplicables con el comando `Graphics`. Por ejemplo, `Axes, AxesLabel, Frame, FrameLabel, Ticks`, entre otras.

Ejemplo: Unidades Académicas según gasto del presupuesto

Con el propósito de ilustrar el uso de algunos recursos de la biblioteca `HirarchicalCluster` y las posibilidades que ofrecen, se construirá una jerarquía binaria indexada de unidades académicas (UA), de la Universidad de Costa Rica, a fin de clasificarlas según la forma en que distribuyen su presupuesto.

Para ello se dispone de una tabla de datos que detalla la cantidad de tiempos completos que cada Unidad Académica utilizó, durante el II ciclo lectivo del año 1993, en las actividades de Docencia, Investigación, Comisiones Institucionales, Administración y Acción Social. El tamaño de esta tabla es 50×5, que corresponde a 50 unidades académicas, cada una medida por las 5 variables: tiempos utilizados en 1) docencia, ..., 5) acción social.

Activar `HierarchicalCluster` y preparar de datos

Para activar los procedimientos en la biblioteca `HierarchicalCluster`, descritos en la sección anterior, se debe transmitir la siguiente orden, la cual activa también los procedimientos correspondientes a `Covariances, Inertias` y `Dissimilarity`:

```
Needs["DataAnalysis`HierarchicalCluster`"]
```

La lectura de datos del problema supone que PUAucr.dat es una archivo tipo texto, formado por 50 renglones y 5 columnas de números separados por espacios en blanco, que contiene la tabla de datos y está en la carpeta `C:\Clasif`.

```
SetDirectory["C:\Clasif"];
```

```
X = ReadList["PUAucr.dat", Number, RecordLists -> True];
```

El resultado de la lectura de `PUAucr.dat` es una matriz 50×5 denominada `X`. Para eliminar el efecto del volumen del presupuesto de cada UA en la clasificación, `X` se transformará en la tabla `Y` cuyas filas corresponden a los porcentajes de presupuesto aplicado a Docencia, Investigación, Comisiones Institucionales, Administración y Acción Social, de cada unidad académica. Además se elimina el primer renglón de la tabla `X` que corresponde a un encabezado sin interés para nosotros. Todo esto se realiza transmitiendo:

```
X = Drop[X, 1];
T = Apply[Plus, X, 1];
Y = X/T;
```

El arreglo de disimilitudes, correspondiente a las 50 UA, se determina a partir de esta tabla `Y` utilizando el índice de disimilitud demoninado `VarianceInverse`, o sea,

$$d_{ij} = \sqrt{\sum_{k=1}^{p} (y_{ik} - y_{jk})^2 / \sigma_k^2}.$$

La siguiente orden calcula dicho arreglo y lo denomina `d1`.

```
d1 = Dissimilarity[Y,VarianceInverse];
```

Construcción de la jerarquía

La construcción de la jerarquía mediante el algoritmo de clasificación jerárquica ascendente se realiza con el procedimiento `HierarchicalTree[]`, utilizando como datos de entrada: a) `Range[50]` que genera una lista de enteros `{1,2,...,50}` que se entiende como la partición inicial del conjunto de 50 objetos a clasificar (o la lista de sus etiquetas), b) el arreglo de disimilitudes `d1` obtenido anteriormente y c) una especificación para el índice de agregación a utilizar en la

construcción de la jerarquía, en este caso: `Ward`.

```
A = HierarchicalTree[Range[50],d1,Ward];
```

La jerarquía así obtenida se denomina `A`, la cual no se muestra en pantalla dado que la orden termina con `;` y porque de todas formas este resultado es poco legible. Debe recordarse que la representación de la jerarquía construída es de la forma:

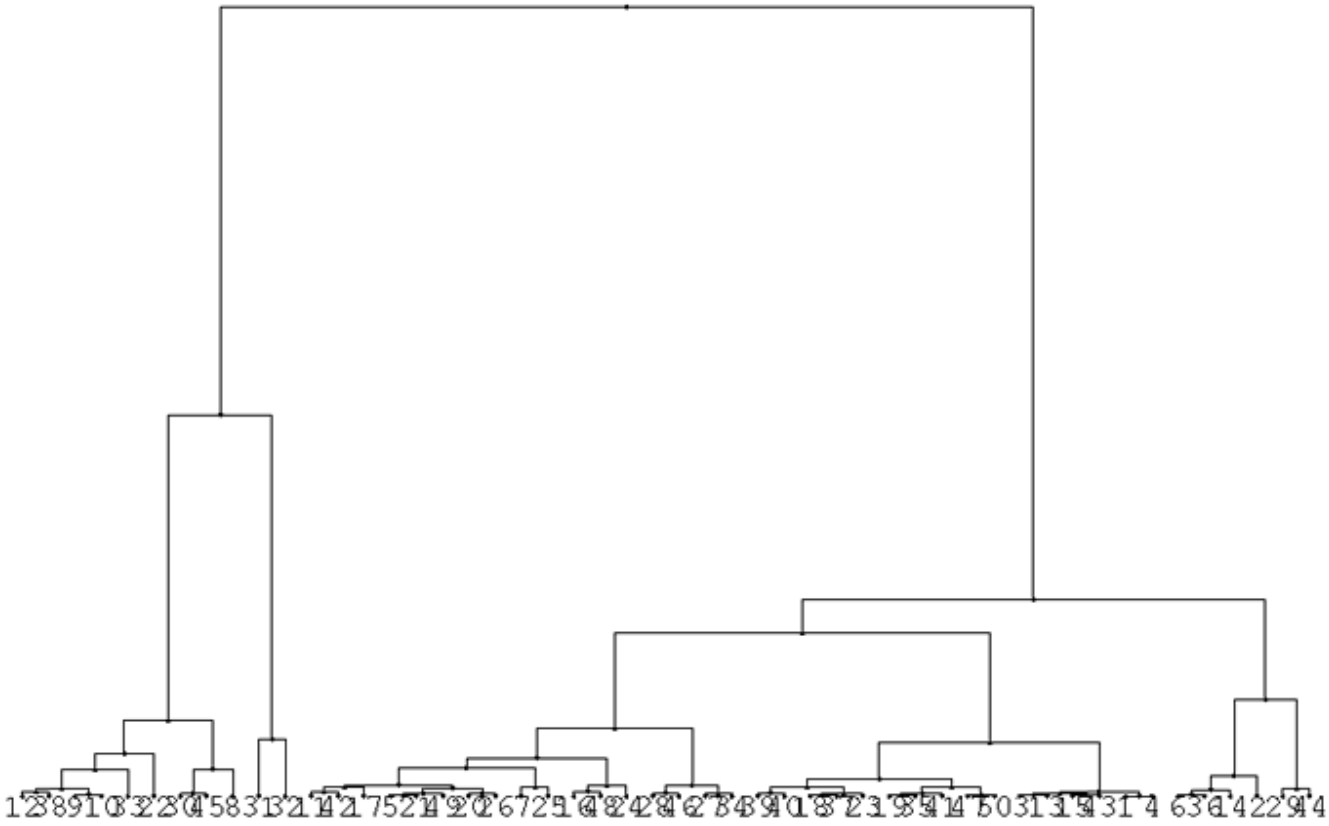
$$\{f(\Omega) + (n-1)I, \text{subárbol izquierdo}, \text{subárbol derecho}\}$$

donde $f(\Omega)$ es el valor de la función de agregación en el conjunto Ω de n objetos a clasificar y para los subárboles izquierdo y derecho se repite recursivamente esta estructura, hasta obtener conjuntos unitarios, representados simplemente por las etiquetas de los objetos.

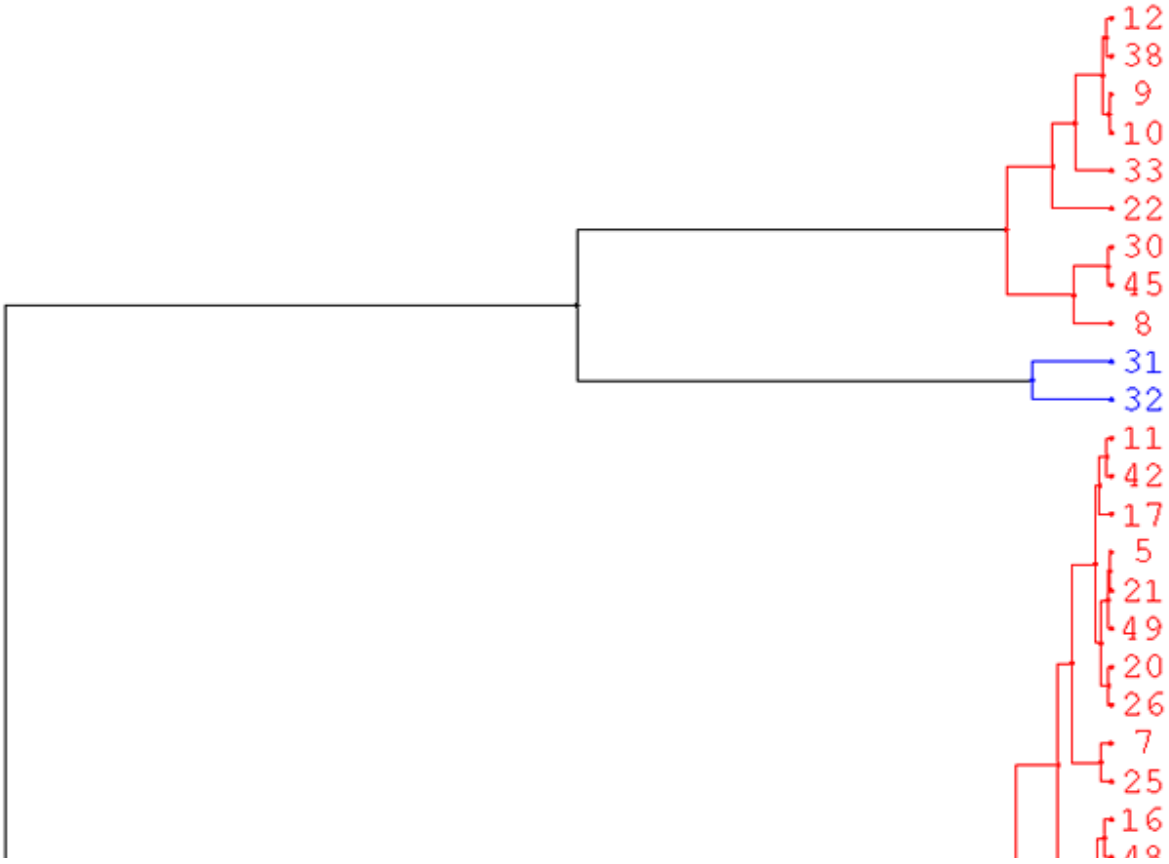
Representación gráfica de la jerarquía

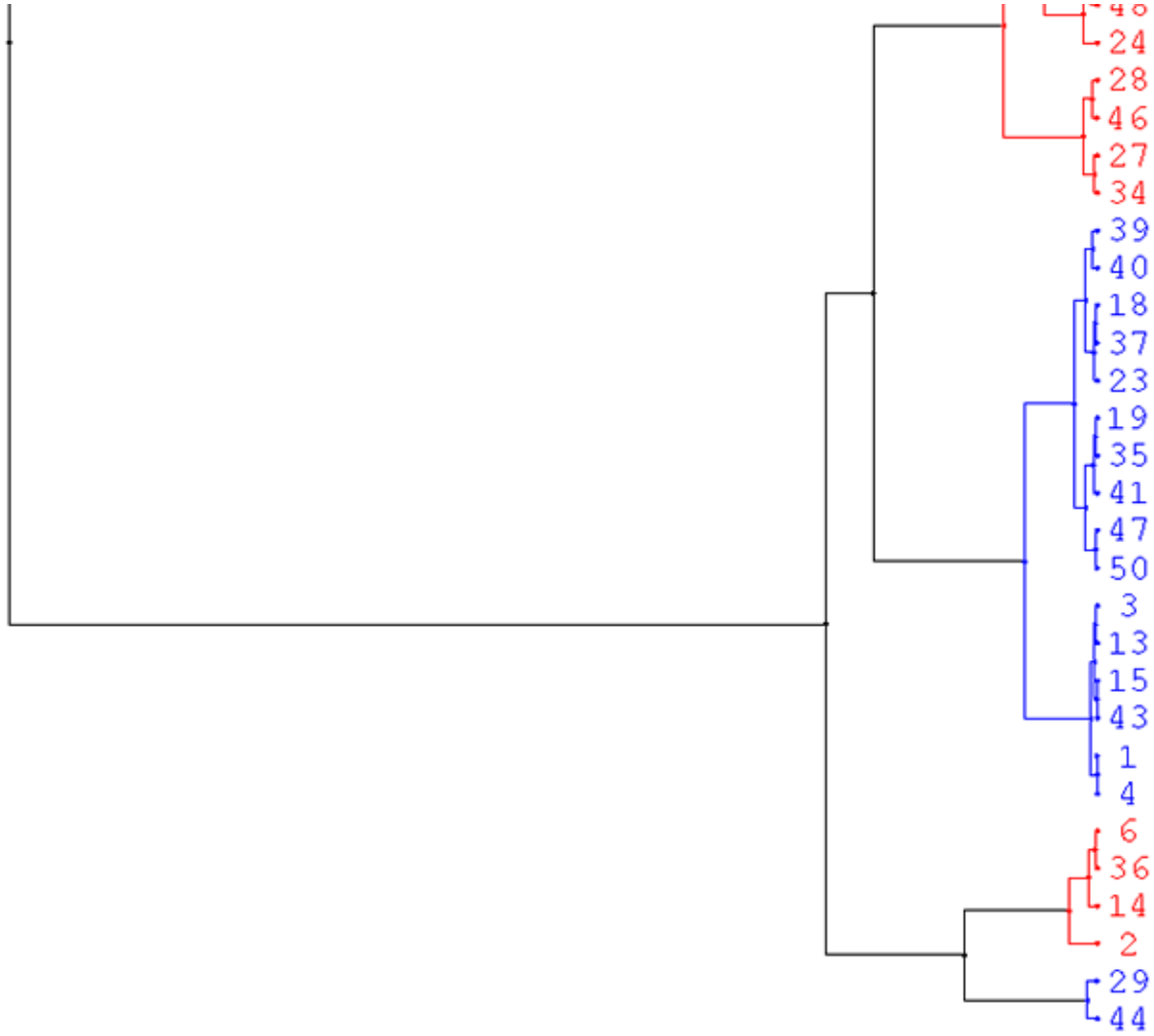
Para obtener la representación gráfica de la jerarquía, se utiliza el procedimiento `TreePlot`:

```
TreePlot[A];
```



```
TreePlot[A, LeafDirection -> Vertical, LeafLabelDistance -> 0.05,
ClustersNumber -> 6];
```



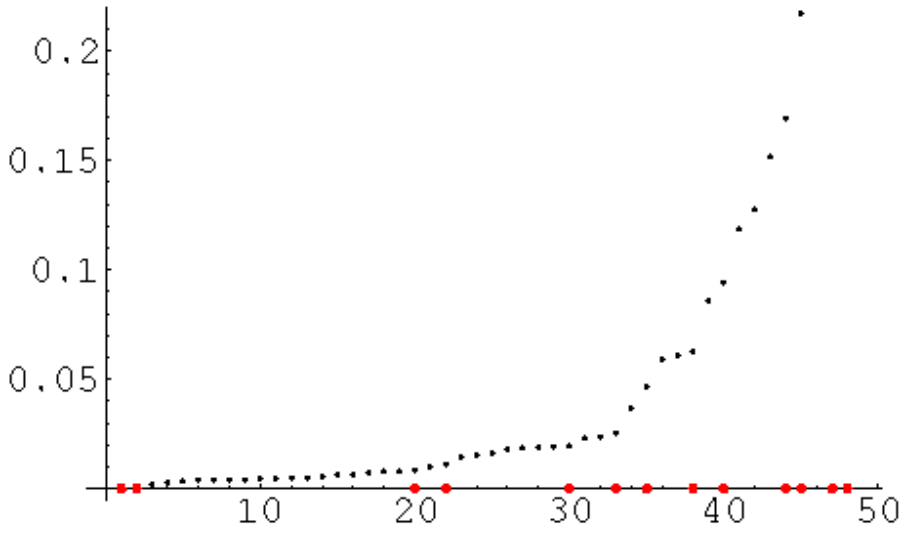


Con el anterior comando se vuelve a dibujar el árbol correspondiente a la jerarquía `A`, pero se cambia la dirección en que se dibujan las hojas, también se modifica la distancia entre nodos y hojas y se distinguen, utilizando dos colores, la partición correspondiente a 6 clases.

Corte del árbol y definición de las clases

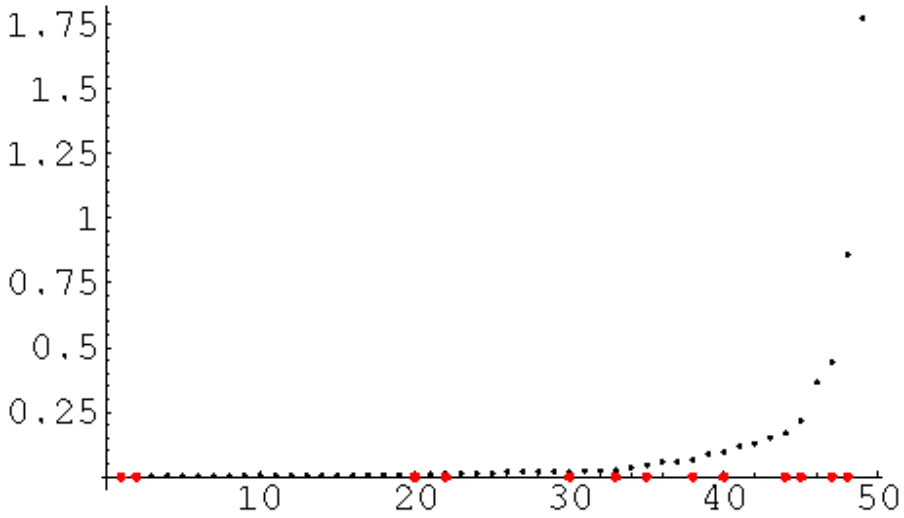
Una primera vista al gráfico parece definir cuatro clases, una de las cuales es unitaria. Sin embargo, para definir este número de clases se puede obtener más información del gráfico que produce `HeightPlot`. En él se muestran los valores que asumió la función de agregación `Ward`, en cada paso del algoritmo de clasificación jerárquica ascendente, al unir las dos clases más cercanas.

```
HeightPlot[A];
```



En este gráfico los últimos 7 puntos quedaron fuera de la región de graficación, lo cual es un efecto definido por omisión que procura un mejor detalle del comportamiento inicial del índice de agregación. Este efecto se puede eliminar utilizando la opción `PlotRange->All`.

```
HeightPlot[A,PlotRange->All];
```



En el eje X, se marcan con puntos rojos, los pasos del algoritmo en los que el valor de la función de agregación produjo un incremento mayor que todos los anteriores. Estos valores pueden ser explícitamente obtenidos con la siguiente orden:

```
MaxJump[A]
```

```
{{1,0,000145181},{2,0,000147613},{3,0,000273929},{5,0,000499001},
 {15,0,00179897},{23,0,00249425},{31,0,00317906},{32,0,00462032},
```

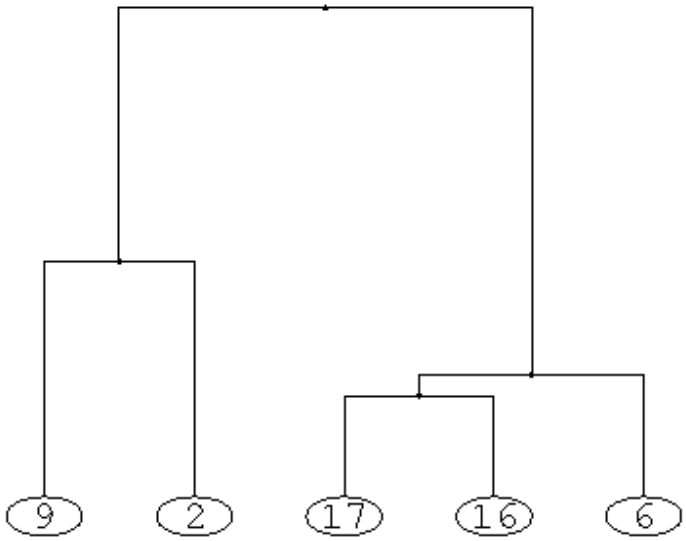
{36,0,00476865},{37,0,00487145},{38,0,01218},{39,0,0288899},

{43,0,0501833},{46,0,317675},{47,0,392263},{48,0,943263}}

El comando `MaxJump[A]//TableForm` despliega estos datos como una tabla con dos columnas.

Con la información de esta tabla y el gráfico anterior podría elegirse, por ejemplo, cortar el árbol en el nivel 45 definiendo 50 - 45 = 5 clases. El siguiente gráfico, muestra la parte superior de la jerarquía, si se corta al nivel 45. El número en el círculo de los nodos terminales corresponde a la cardinalidad de la clase que se define con el corte. Si el número no aparece encerrado por un círculo se trata de la etiqueta de un objeto e indica que la clase es unitaria compuesta sólo por ese objeto.

```
TreePlot[A, LevelCut -> 45, LeafLabelDistance -> 0.07];
```



La siguiente orden muestra la composición de cada una de las clases de la partición resultante al cortar el árbol en el nivel 45 y define los símbolos `P5` para identificar la partición completa y `C1,C2,C3,C4` y `C5` para cada una de sus clases:

```
P5 = {C1, C2, C3, C4, C5} = ClusterPartition[A, 45]
```

{{12,38,9,10,33,22,30,45,8},{31,32}, {11,42,17,5,21,49,20,26,7,25,16,48,24,28,46, 27,34},

{39,40,18,37,23,19,35,41,47,50,3,13,15,43,1,4},{6,36,14,2,29,44}}

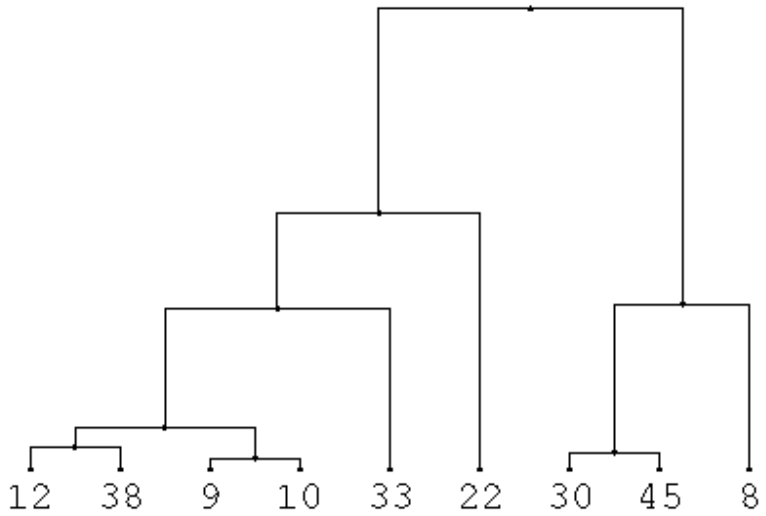
Representación gráfica de las clases

El comando siguiente define la variable `SubArbol` como una lista de los 5 nodos o subárboles que resultan de cortar el árbol `A` al nivel 45.

```
SubArbol = CutTree[A, 45];
```

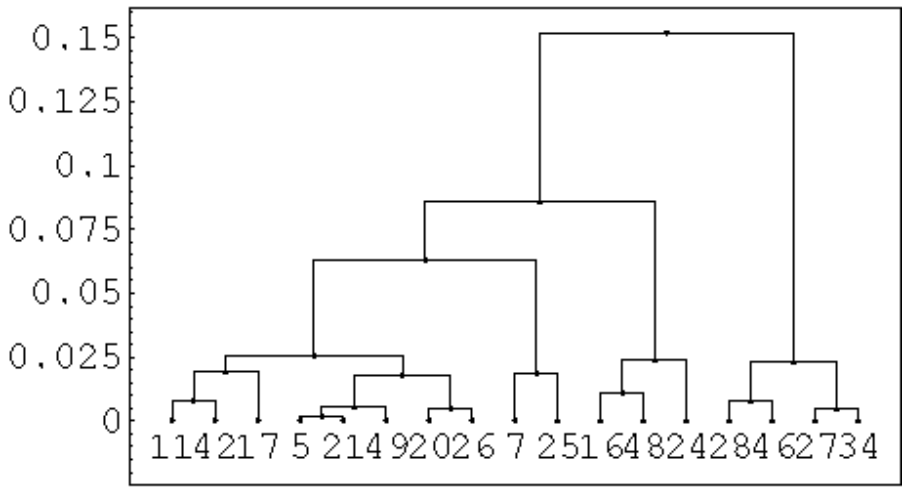
Nuevamente se termina la orden con ; para no desplegar el resultado. En este caso sólo interesa definir la variable `SubArbol` para el propósito de trazar el gráfico de cada uno de los subárboles correspondientes a la partición en 5 clases obtenida con el corte especificado. Como en el caso de la jerarquía total, el comando `TreePlot` permite construir los gráficos del subárbol de cada clase:

```
TreePlot[SubArbol[[1]], LeafLabelDistance -> 0.01];
```

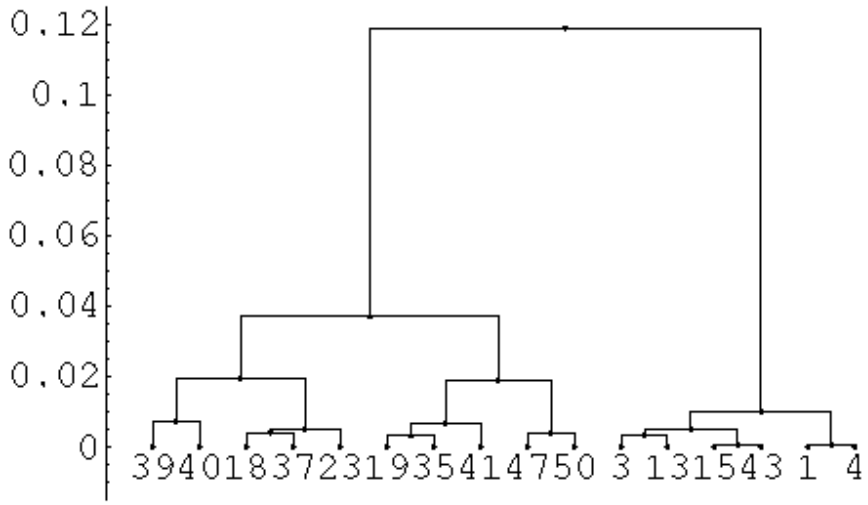


Además de las opciones propias de `TreePlot`, este procedimiento acepta todas aquellas que reconoce la primitiva `Graphics` de MATHEMATICA. Por ejemplo `PlotRange, Axes, Frame`, etc. Con la construcción de los gráficos asociados a las clases 3 y 4 se ilustran algunas posibilidades.

```
TreePlot[SubArbol[[3]], LeafLabelDistance -> 0.01, Frame -> True,
FrameTicks -> {False, True, False, False}];
```



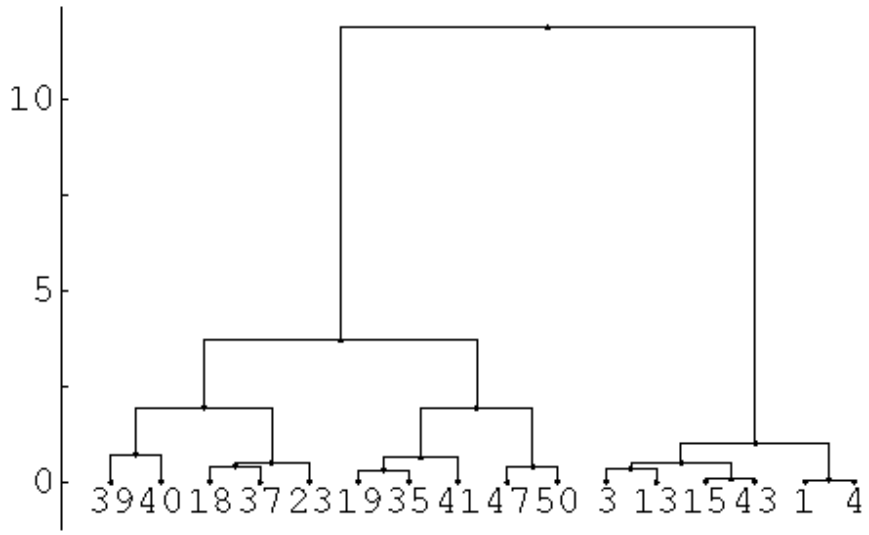

```
TreePlot[SubArbol[[4]], Axes -> False, True, LeafLabelDistance -> 0.006];
```



La variable `MisTicks` define las marcas y etiquetas deseados para el eje Y, a emplear en los siguientes gráficos:

```
MisTicks = {None, {{0, 0}, {0.025, " "}, {0.05, 5}, {0.075, " "}, {0.1,10},
{0.125, " "}, {0.15, 15}, {0.175, " "}, {0.2, 20}, {0.225," "}}};
```

```
TreePlot[SubArbol[[4]], Axes -> {False, True}, LeafLabelDistance -> 0.005,
Ticks ->MisTicks];
```



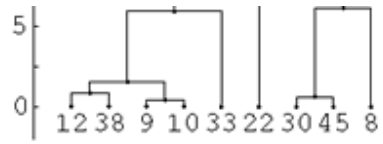
La función `Graf` recibe un nodo representante de una clase y produce el gráfico del árbol correspondiente, estableciendo explícitamente la misma área de gráfico, `PlotRange->{{-0.2,13.4},{-0.02,0.11}}`, para todo gráfico que trace. Además agrega otras características deseadas para el gráfico.

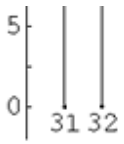
```
Graf[arb_] := TreePlot[arb, LeafLabelDistance -> 0.01,
PlotRange -> {{-0.2, 18.0}, {-0.02,0.24}},
Axes -> {False, True},Ticks -> MisTicks];
```

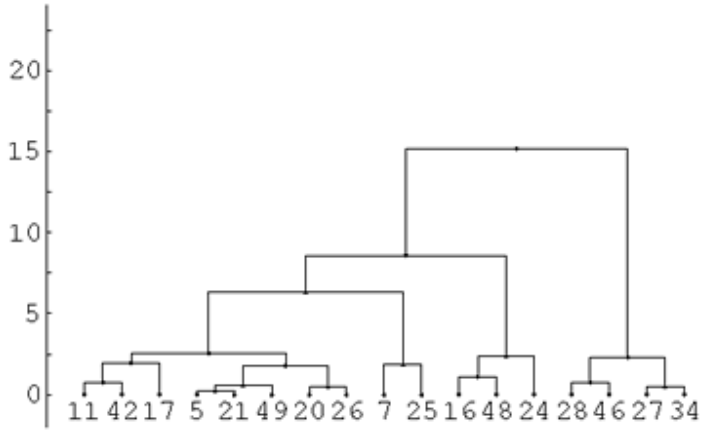
Con esta función es posible construir los gráficos asociados a las clases, en todos los casos utilizando las mismas escalas en los ejes X y Y, para definir con ellos un solo arreglo utilizando la primitiva `GraphicsArray`.

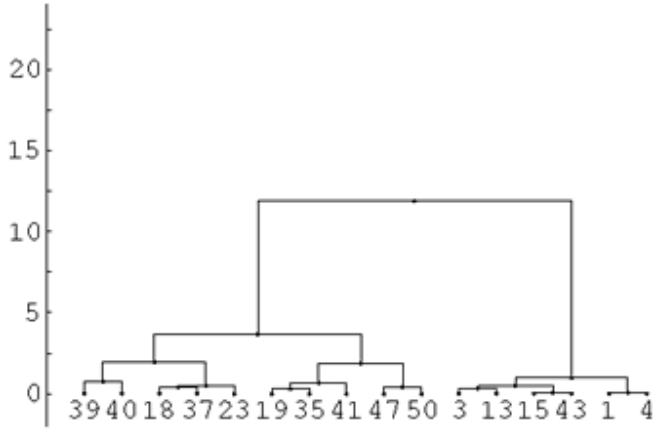
```
{g1,g2,g3,g4,g5} = Map[Graf,SubArbol];
```

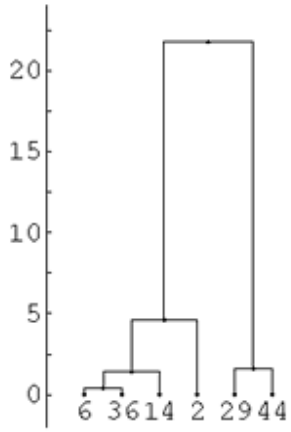
```
Show[GraphicsArray[{{g1, g2}, {g3, Graphics[{}]}, {g4, g5}}]];
```











Bibliografía

1. DIDAY, E.; LEMAIRE, J.; POUGET, J.; TESTU, F. (1982) *Eléments d'Analyse des Données*. Dunod, Paris.

2. GRAY, J. (1994) *Mastering Mathematica: Programming Methods and Applications*. AP Professional, New York.

3. JAMBU, M. (1978) *Classification Automatique pour l'Analyse des Données. Méthodes et Algorithmes*. Dunod, Paris.
4. JAMBU, M. (1999) *Méthodes de Base de l'Analyse des Données*. Eyrolles, Paris.
5. TREJOS, J. (2003) *Clasificación Automáática*. Notas de clase, Universidad de Costa Rica, San José.