LES CAHIERS DE L'ANALYSE DES DONNÉES

J. JUAN

Le programme HIVOR de classification ascendante hiérarchique selon les voisins réciproques et le critère de la variance

Les cahiers de l'analyse des données, tome 7, n° 2 (1982), p. 173-184

<http://www.numdam.org/item?id=CAD_1982__7_2_173_0>

Les Cahiers de l'Analyse des Données
Vol. VII - 1982 - n° 2 - p. 173-184

LE PROGRAMME HIVOR
DE CLASSIFICATION ASCENDANTE HIÉRARCHIQUE
SELON LES VOISINS RÉCIPROQUES
ET LE CRITÈRE DE LA VARIANCE
[PROG. C.A.H. RECIP.]

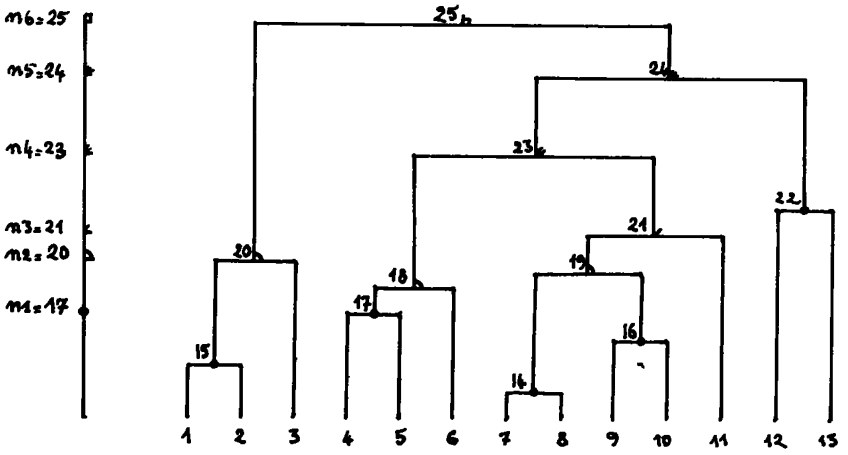
par J. Juan (1)

1 *Plan de l'article*

Après le rappel des principes de divers algorithmes (§ 2) de C.A.H. (classification ascendante hiérarchique), nous donnons le schéma de l'algorithme sur lequel repose HIVOR (§ 3), puis un aperçu des performances du programme (§ 4) et enfin le programme lui-même (§ 5) dont le listage est donné à la fin de l'article (§ 5.3) après la bibliographie (§ 5.2).

2 *Principe des algorithmes de classification ascendante hiérarchique*

Nous considérons trois algorithmes d'agrégation : l'algorithme de base (BASE), l'algorithme des graphes réductibles (REDUC) et l'algorithme des voisins réciproques (RECIP). Pour montrer comment REDUC et RECIP permettent d'accélérer BASE, nous prenons un exemple d'arbre d'une certaine complexité, construit sur un ensemble I de 13 individus.



Exemple de recherche des voisins réciproques : [PROG. CAH. RECIP.]

(1) *Boursier DGRST du laboratoire de statistique de l'université Pierre et Marie Curie (PARIS VI).*

Selon BASE, on calcule les niveaux d'agrégations (ou indices de distances) n(i,i') pour chacune des paires (i,i') d'individus de l'ensemble I = {1,..., 13} : le niveau le plus bas trouvé étant n(7 , 8) on agrège cette paire pour constituer le noeud 14. L'algorithme BASE reprend sur l'ensemble I' = {1,2,3,4,5,6,9,11,12,13,14} formé des 11 individus restants (tous sauf 7 et 8) et du noeud 14 : on trouve qu'il faut agréger 1 et 2 qui constituent le noeud 15. Et BASE recommence sur I" = {3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15} pour agréger 9 et 10 qui forment le noeud 16. Et ainsi de suite... On constate qu'à chaque étape de BASE, seul le plus petit de tous les indices de distances est utilisé.

Ceci aura deux lourdes conséquences : la place en mémoire centrale sera importante et les temps d'exécution seront lents.

Pour éviter de répéter le calcul de distances dont certaines étant très élevées ne peuvent constituer des candidats à la création d'un noeud que dans les étapes ultimes de la construction ascendante de l'arbre, REDUC garde en mémoire toutes les distances trouvées inférieures à un premier seuil nr1 et n'opère que sur elles jusqu'à avoir construit la partie de l'arbre inférieure à ce seuil. On prend alors un autre seuil nr2 au dessus de nr1 et de même on construit les noeuds dont le niveau est compris entre nr1 et nr2 ; et ainsi de suite par bandes successives on s'élève jusqu'au sommet de l'arbre. Il importe de voir que la mise en oeuvre de REDUC est difficile pour plusieurs raisons. On doit choisir le seuil (e.g. nr1) assez haut pour trouver au-dessous les niveaux de nombreux noeuds ; et assez bas pour ne pas avoir à garder une trop longue liste de "distances"; de plus chaque fois qu'on crée un noeud, cette liste doit être mise à jour : si on agrège i à i' (ou i à n ou n à n'...) on supprime de la liste les distances entre i et i' et d'autres éléments e, e', e",..., mais on y introduit les distances entre (i ∪ i') et e, e', e",..., si toutefois celles-ci sont inférieures au seuil.

L'algorithme RECIP est d'une mise en oeuvre beaucoup plus simple : il repose sur la remarque que dès le premier calcul de la matrice des distances on pouvait conclure à l'agrégation non seulement de 7 et 8, mais aussi de (1 et 2) de (9 et 10) de (4 et 5) et même de (12 et 13). Dans quelle mesure cela est-il vrai ? Pour être assuré e.g. que 1 doit être agrégé à 2, il faut qu'à la fois 2 soit l'élément le plus proche de 1 et que 1 soit l'élément le plus proche de 2 : on dit alors que 1 et 2 sont des voisins réciproques . Il faut aussi que les agrégations effectuées entre d'autres éléments ne créent pas de nouveaux noeuds qui se trouvent à une distance de 1 ou de 2 inférieure à n(1 , 2). La même condition est d'ailleurs requise pour REDUC : elle ne met en jeu que trois éléments et s'écrit simplement :

$$\forall\ a,\ b,\ c\ :\ n(a;b) < \inf(n(a;c);n(b;c)) \Rightarrow$$
$$n(a \cup b;c) > \inf(n(a;c);n(b;c))\quad ;$$

formule où a, b, c désignent trois éléments (individus ou classes d'individus) et n(a,b),..., les niveaux d'agrégation de ces éléments .

Cette condition est opportunément vérifiée par les niveaux calculés suivant les critères usuels d'agrégation : saut minimum, diamètre, distance moyenne et variance.

Reprenons le déroulement de RECIP. On a donc agrégé les paires de voisins réciproques : {1 , 2}, {4 , 5}, {7 , 8}, {9 , 10}. La paire {12 , 13} n'a pas été agrégée dès la première étape car elle n'était pas composée de voisins réciproques. (En effet 12 avait 11 (par exemple) comme plus proche voisin mais pour 11 le plus proche élément était 9 et on a vu que 9 et 10 étaient voisins réciproques...). On recommence les agrégations des voisins réciproques sur I' = {15,17,6,14,16,11,12 ,13 } et ainsi de suite jusqu'à aboutir au sommet de l'arbre.

On utilise ainsi au maximum l'information apportée par les "distances" (ou ce qui en tient lieu comme critère d'agrégation) sur tous les éléments. Les ensembles I , I', I"... formés au cours des étapes successives de l'algorithme seront en nombre moins élevé que dans BASE. On peut s'attendre de plus à ce que la diminution du nombre d'éléments d'une étape à l'autre est d'autant plus forte que les éléments sont nombreux dans l'étape en cours. A titre d'exemple voici le comportement de l'algorithme sur un ensemble de 7520 individus définis par leurs 6 premières coordonnées factorielles :

étape	nombre d'agrégations	étape	nombre d'agrégations
1	2157	21	39
2	1154	22	31
3	778	23	26
4	601	24	22
5	472	25	13
6	381	26	13
7	311	27	12
8	257	28	9
9	220	29	11
10	177	30	8
11	151	31	5
12	119	32	4
13	104	33	3
14	84	34	3
15	69	35	4
16	61	36	1
17	61	37	1
18	56	38	2
19	47	39	1
20	50	40	1

On voit que les quatre premières étapes ont formé 4690 noeuds sur les 7519 que comporte la classification totalement terminée.

Il est d'autre part possible d'apprécier sur des cas modèles l'efficacité de l'algorithme; on trouve ainsi (c . [*PROP. RECIP.*]) que le nombre de noeuds formés à la 1-ère itération sur m individus est de l'ordre de (m/4) à (m/3) selon la dimension de l'espace ambiant . Ce qui s'accorde pleinement avec le cas traité ici : $(1/4) < (2154/7520) < (1/3)$. Une dernière remarque pour conclure les rappels généraux : le principe de RECIP peut fort bien être appliqué au sein des étapes successives de REDUC.

3 *L'algorithme des voisins réciproques*

3.1 *Structure générale des algorithmes* : L'algorithme de base part de la partition $P_0 = \{\{i\}/i \in I\}$. Il agrège les deux classes les plus voisines (au sens du critère n que l'on a choisi) de la partition en cours pour obtenir une partition plus grossière. On arrête les agrégations lorsqu'on a abouti à la partition $P_f = \{I\}$. La suite des partitions emboîtées permet de définir une hiérarchie totale H que l'on représente par un arbre.

Pour le critère de la variance on définit le niveau n par :

$$n(q,s) = \frac{f_q \cdot f_s}{f_q + f_s} \cdot d^2(q,s)$$

et la classe $q \cup s$ est (éventuellement) créée au niveau

$$\nu(q \cup s) = \eta(q,s)$$

Ici q et s sont des classes ou des éléments de I, f_q et f_s leurs poids et $d^2(q,s)$ la distance euclidienne usuelle. Une classe q est représentée par son centre de gravité q et par son poids f_q qui est la somme des poids de ses éléments . Cet indice de niveau ν est croissant i.e. si $q \neq r$ et $q \subset r$ alors $\nu(q) < \nu(r)$.

Dans les programmes, les éléments de H sont numérotés comme sur la figure du § 2 : les singletons (de 1 à Card I) dans l'ordre de lecture des éléments de I et les noeuds (de CARD I+ 1 à 2 Card I -1) dans l'ordre de leurs indices de niveau.

3.2 *Les voisins réciproques* : {i,j} est une paire de voisins réciproques si i a pour plus proche voisin (au sens de η) j et réciproquement.

On part de $P_0 = \{\{i\} | i \in I\}$. On agrège toutes les paires disjointes de voisins réciproques de la partition en cours pour obtenir une partition plus grossière. On arrête les agrégations lorsque la partition $P_f = \{I\}$ est formée. On peut montrer que l'on obtient des hiérarchies totales exactes (i.e. au sens de l'algorithme de base) pour les critères d'agrégations cités au § 2 .

3.3 *L'algorithme proposé ("algorithme des célibataires")* (*)

Posons $D(i) = \inf\{n(i,k) | k \in I\} \in R_+$

$V\emptyset(i) = \{j | j \in I \; ; \; D(i) = D(j) = n(i,j)\}$

On dira que q est *célibataire* à l'étape en cours si sa plus proche voisine r s'agrège avec $s \neq q$.

1 Lecture des poids et des coordonnées de $i \in I$
faire JU = 0, II = 0, N = $\emptyset$

2 Calculs de n(i,j) pour i < j et définition de
V$\emptyset$ (i) et D(i) pour $i \in I$

3 chercher les paires réciproques. A chaque paire trouvée (i,j),
faire $c = i \cup j$; II = II + 1 ; R(II) = c
$I = I - \{i\} - \{j\}$; JU = JU + 1 ; N = N +{c} ; DI(JU) = D(i)

4 Si JU ≥ NI-1 aller en 8

5 Ajouter tous les célibataires dans R et les enlever de I

6 Définition de V$\emptyset$(i) et D(i) pour $I \in R$ par calculs de n(i,j)
sur :
- $i \in R$, $j \in R$ et i < j
- $i \in I$ et $j \in R$

(*) *La notion de célibataire permet une économie importante dans les calculs de distance ; la recherche du plus proche voisin n'est à refaire que pour les célibataires; (outre la recherche à faire pour chaque noeud créé) tandis que les éléments dont le ... proche voisin ne s'est pas agrégé, garde celui-ci comme p.p.v. ;pour des complé- cf. [C.A.H. CHAINE RECIP.].*

7 Ajouter les éléments de R dans I ; faire II = 0 et aller en 3

8 Trier les noeuds n ∈ N selon l'ordre croissant de leurs indices de niveau (stockés dans DI) puis écrire la hiérarchie

8 Stop

Remarque : A l'étape 6, on ne calcule pas les niveaux n(i,i') entre éléments i et i' de I.

4 *Les performances*

On les compare à celles d'un programme de classification selon la méthode des graphes réductibles (REDUC cf. § 2). Cette méthode engendre aussi des hiérarchies totales exactes avec les critères cités § 2.

Sur la partition en cours, on sélectionne les éléments dont les proximités sont inférieures au seuil nr de l'étape. On agrège les deux éléments les plus voisins, la mise à jour des nouvelles proximités à calculer se fait en éliminant celles supérieures à nr. On passe à l'étape suivante (en calculant un nouveau seuil) s'il ne reste plus d'agrégations à faire et si la partition $P_f = \{I\}$ n'a pas été atteinte.

Ainsi qu'on l'a dit au § 2, le choix de la suite croissante des seuils est délicat. La comparaison a été effectuée sur un programme disponible au CIRCE qui utilise la définition suivante :

nr = inertie totale/nombre de proximités restantes ("seuil de l'inertie".

graphes réductibles "seuil inertie"		voisins réciproques	
Card I	temps d'exécution	Card I	temps d'exécution
43(7)	0,5	43(7)	0,3
188(7)	4,4	188(7)	1,8
371(6)	19,1	376(6)	3,0
728(6)	75,2	752(6)	9,9
1785(6)	816,9	1880(6)	54,4

Comparaisons faites sur IBM 370/168. Les temps sont en secondes et dans la colonne Card I on indique entre parenthèses, après le nombre d'individus, le nombre des coordonnées factorielles utilisées (6 ou 7).

Les temps observés (sur des tableaux provenant des mêmes données) montrent nettement la plus grande rapidité de l'algorithme des voisins réciproques. D'autre part le programme de classification selon les graphes réductibles occupe plus de place en mémoire car il doit garder les tableaux afférents aux proximités inférieures aux seuils.

5 *Le programme HIVOR*

5.1 *Remarques préalables* : Le programme effectue une classification sur des individus d'un tableau de facteurs I0P=1 ou bien d'un tableau de correspondance (I0P=2) ou encore d'un tableau de mesures (I0P=3).

On ne retient pour chaque élément qu'une seule plus proche voisine. Pour éviter les cycles qui pourraient se produire (e.g. cas du triangle équilatéral), on teste l'égalité de D(i) et D(j) pour les paires {i , j} susceptibles d'être réciproques.

Le tri final se fait par la méthode de SHELL revue par KNUTH . Elle semble la plus indiquée car les effectifs sont peu nombreux (inférieurs à 10.000 en général) et surtout le tableau à trier est "presque rangé".

Le programme fonctionne en allocation dynamique grâce à l'utilisation du sous programme CESGET implanté au CIRCE dans la bibliothèque SYS1.BIBLI.NIV1. On donne le listage d'un CESGET de remplacement pour les non utilisateurs du CIRCE (l'allocation n'est plus dynamique).

Enfin la hiérarchie est stockée sous une forme compatible avec les programmes usuels de représentation d'arbres et d'aides à l'interprétation (format 1X,4I5,E20.10) correspondant au numéro du noeud, son nombre d'éléments, les numéros des deux éléments qui le composent et enfin son indice de niveau).

5.2 *Bibliographie*

J.P. Benzécri (1973) L'Analyse des Données (2 tomes)... DUNOD, Paris

H.H. Bock (1974) Automatische Klassifikation VANDENHOECK und RUPRECHT in Göttingen.

H. Brisse et G. Grandjouan (1977) Un procédé de classification par agrégations d'un effectif nombreux.Bulletin de l'A.S.U. 3. 1977 , (Paris).

M. Bruynooghe (1977) Méthodes nouvelles en classification automatique de données taxinomiques nombreuses. Bulletin de l'A.S.U., 3. 1977; Jussieu (Paris).

M. Bruynooghe (1978) Article [*CLASS. RAP.*] VOL III n° 1 *Cahiers* A.des D., DUNOD, Paris.

P.O. Flavigny (1974) Article d'Interface n° 12 du CIRCE, Orsay.

M. Jambu et M.O. Lebeaux (1978) Classification automatique (2 vol.) DUNOD, Paris.

D.E. Knuth (1973) Sorting and Searching (tome 3 de "the Art of computer programming"). ADDISON-WESLEY, Reading (Massachussets).

L.L. Mac Quitty(1966) Single and Multiple hierarchical classification by reciprocal pairs and rank types. EDUCATIONAL and PSYCHOLOGICAL MEASUREMENT vol 26, n° 2.

C. de Rham (1980) Article [*CAH. VOIS. RECIP.*] vol V n° 2 *Cahiers* A. des D. DUNOD, Paris.

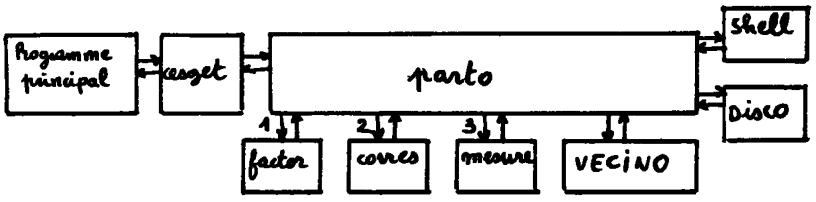
5.3 *Le listage de HIVOR* : Les trois cartes paramètres nécessaires à l'exécution de HIVOR sont expliquées dans les commentaires du programme principal. Celui-ci calcule la taille mémoire et appelle la sous-routine CESGET qui réserve l'espace mémoire et appelle PARTO. Les utilisateurs du CIRCE doivent assigner la bibliothèque SYS1.BIBLI.NIV1 dans leur flot de cartes contrôles :

```
//GO.SYSLIB DD
//          DD DISP=SHR,DSN=SYS1.BIBLI.NIV1
```

Les autres utilisateurs doivent ajouter au programme HIVOR la sous-routine CESGET de remplacement où selon la valeur de " MEMOIR " ils ajustent la dimension maximale des tableaux. On liste ici une CESGET qui réserve 10.000 mots en mémoire :

```
      SUBROUTINE CESGET(PARTO,LMOT,NBMOT,*)
C
C     CE SOUS PROGRAMME N'EST PAS A METTRE SI ON UTILISE LE CIRCE.
C    (IL SUFFIT ALORS DE METTRE LES CARTES DE CONTROLES SUIVANTES:
C            //GO.SYSLIB DD
C            //          DD DISP=SHR,DSN=SYS1.BIBLI.NIV1       )
C
C     SI ON UTILISE CE SOUS PROGRAMME ,L'ALLOCATION N'EST PAS DYNAMIQUE.
C
      DIMENSION S(10000)
      IF(NBMOT*LMOT*0.25.GT.10000) RETURN 1
      CALL PARTO(S,NNMOT)
      RETURN
      END
```

La structure du programme HIVOR est :



Structure du programme HIVOR.

```
C     PROGRAMME HIVOR.CLASSIFICATION ASCENDANTE BINAIRE,ALGORITHME
C     DES VOISINS RECIPROQUES PAR LA METHODE DES CELIBATAIRES.
C     STRATEGIE DE LA VARIANCE.
C     PAS DE FICHIER DE TRAVAIL.
C     LE FICHIER ILEC A HIERARCHISER EST UN FICHIER DE NI ELEMENTS; LE
C     PROGRAMME ADMET 3 TYPES DE FICHIER ILEC:
C     IOP-1 FICHIER DE FACTEURS
C     IOP=2 FICHIER DE CORRESPONDANCE
C     IOP=3 FICHIER DE MESURES CONTINUES.
C     SI IOP=3 ,LES NJ VARIABLES SONT CENTREES ET REDUITES.
C     POUR IOP=2 OU IOP=3 ON LIT SEULEMENT LES NJ VALEURS EN REELS.
C     POUR IOP-1 ON LIRA LE POIDS ET LES NJ COORDONNEES EN REELS. SI LE
C     FICHIER DES FACTEURS PROVIENT DE ADDAD , LE FORMAT A ECRIRE SERA
C     (40X,2E20.10/(4E20.10)).
C     LA HIERARCHIE EST ECRITE SUR LE FICHIER ISOR SELON LE FORMAT
C      (1X,4I5,E20.10) QUI CORRESPOND A:
C     NUM.DU NOEUD,NB ELEMENTS,NUM.AINE,NUM.BENJAMIN ET INDICE DU NOEUD
C
C     MEMOIRE : 11*NI+NI*NJ+2*NJ-5( VERSION HIERARCHIE USUELLE )
C
C     IL Y A 3 CARTES PARAMETRES:
C     1  TITRE( 20A4 )
C     2  NI,NJ,IOP,ILEC,ISOR( 5I5 )
C     3  FORMAT( 20A4 )
C
C     PROGRAMME DE JOSE JUAN,9 NOVEMBRE 1981.
C     REFERENCES : ARTICLE ''PROG.CAH.RECIP'' DES CAHIERS DE
C                  L'ANALYSE DES DONNEES.DUNOD.
C
```

```
      EXTERNAL PARTO
      INTEGER TITRE(20),FMT(20)
      COMMON /PAR/ NI,NJ,IOP,NTOT,NTOF
      COMMON /IO/ ILEC,ISOR,FMT
      READ 1,TITRE
1     FORMAT(20A4)
      PRINT 2,TITRE
2     FORMAT(1X,20A4/)
      READ 3,NI,NJ,IOP,ILEC,ISOR
3     FORMAT(5I5)
      PRINT 4,NI,NJ,IOP,ILEC,ISOR
4     FORMAT(1X,'     NI     NJ    IOP   ILEC   ISOR'/1X,5I6//)
      READ 1,FMT
      PRINT 2,FMT
      NTOF=NI-1
      NTOT=NTOF+NI
      MEMOIR=NTOT+5*NI+NI*NJ+2*NJ+4*NTOF
      CALL CESGET(PARTO,4,MEMOIR,85)
      STOP
5     CONTINUE
      PRINT 6,MEMOIR
6     FORMAT(1X,I7,'MOTS MINIMUMS A RESERVER.AUGMENTER ESPACE MEMOIRE')
      STOP
      END
      SUBROUTINE PARTO(V,MEMOIR)
      DIMENSION V(MEMOIR)
      COMMON /PAR/ NI,NJ,IOP,NTOT,NTOF
      LA=1+NI
      LB=LA+NI*NJ
      LC=LB+NTOT
      LD=LC+NI
      LE=LD+NI
      LF=LE+NI
      LG=LF+NTOF
      LN=LG+NJ
      LO=LN+NI
      LP=LO+NTOF
      LQ=LP+NTOF
      LR=LQ+NTOF
      IF(IOP.EQ.1) CALL FACTOR(V(1),V(LA))
      IF(IOP.EQ.2) CALL CORRES(V(1),V(LR),V(LA),V(LG))
      IF(IOP.EQ.3) CALL MESURE(V(1),V(LR),V(LA),V(LG))
      CALL VECINO(V(1),V(LA),V(LB),V(LC),V(LD),V(LE),V(LF),V(LG),V(LN),
     CV(LO),V(LP),V(LQ))
      CALL SHELL(V(LF),V(LQ))
      CALL DISCO(V(1),V(LB),V(LF),V(LO),V(LP),V(LQ))
      RETURN
      END
      SUBROUTINE VECINO(FI,FIJ,IP,SOM,VO,D,IVO,TAB,R,A,B,DI)
      INTEGER IP(NTOT),SOM(NI),VO(NI),IVO(NTOF),R(NI)
      INTEGER A(NTOF),B(NTOF)
      REAL FI(NI),FIJ(NI,NJ),D(NI),TAB(NJ),DI(NTOF)
      COMMON /PAR/ NI,NJ,IOP,NTOT,NTOF
C          TABLEAUX UTILISES :
C     FI    : POIDS DES SUJETS A HIERARCHISER
C     FIJ   : DONNEES
C     IP    : CARDINAUX DES CLASSES
C     SOM   : NUMERO DES SOMMETS
C     VO    : VOISINS RECIPROQUES
C     D     : DISTANCES MINIMALES
C     R     : INDIVIDUS RESTANTS
C     IVO   : INDIVIDUS INCHANGES
C     A     : AINES
C     B     : BENJAMINS
C     DI    : INDICES DES NOEUDS
C     TAB   : COORDONNEES DU SUJET COURANT
C
      AFINI=1.E+50
      II=NI
      NR=NI
      ITER=0
      VO(1)=2
      DO 30 I=1,NI
      D(I)=AFINI
      IP(I)=1
      R(I)=I
30    SOM(I)=I
C     CALCULS DES PROXIMITES ENTRE SUJETS I ET K ( AVEC I<K ) POUR
C     I VARIANT DE 1 A NTOF (NTOF=NI-1).DIC EST LA PROXIMITE ENTRE I
C     ET SON PLUS PROCHE VOISIN IST .DDK EST LA PROXIMITE ENTRE I ET K.
C     DD EST LE MAX ENTRE DIC ET D(K).LE CALCUL DE DDK EST ABANDONNE SI
C     DD<DDK. A LA FIN DE L'ETAPE LES TABLEAUX D ET VO SONT REMPLIS.
```

```
      DO 1 I=1,NTOF
      PI=FI(I)
      IST=VO(I)
      DIC=D(I)
      DO 2 J=1,NJ
2     TAB(J)=FIJ(I,J)
      ISU=I+1
      DO 3 K=ISU,NI
      DK=D(K)
      DD=DIC
      IF(DD.LT.DK) DD=DK
      DDK=0.
      PON=PI*FI(K)/(PI+FI(K))
      DO 4 J=1,NJ
      RAP=TAB(J)-FIJ(K,J)
      DDK DDK+PON*RAP*RAP
      IF(DDK.GE.DD) GOTO 3
4     CONTINUE
      IF(DDK.GE.DIC) GOTO 5
      DIC=DDK
      IST=K
5     IF(DDK.GE.DK) GOTO 3
      D(K)=DDK
      VO(K)=I
3     CONTINUE
      D(I)=DIC
      VO(I)=IST
1     CONTINUE
      PRINT 50
50    FORMAT(1H1,1X,' CARACTERISTIQUES DES ITERATIONS:'///1X,
     C'ITER NOEUDS CELIB. INCHA.  TOTAL      INTRA'/)
      ANE=0.
      JU=0
C     ANE EST LA SOMME DES INDICES DE NIVEAUX . JU EST LE COMPTEUR DES
C     NOEUDS TROUVES ( SI JU=2*NI-1 ALORS LA HIERACHIE EST FINIE )
C    L'ETAPE SUIVANTE (ETIQUETTE 6) RECHERCHE LES VOISINS RECIPROQUES
C    POUR LES AGREGER.ON STOCKE LES NUMEROS DES NOEUDS CREES DANS R,LES
C    NUMEROS DES NON-AGREGES SONT DANS IVO.
C    UN ELEMENT I EST NON-AGREGE SI VO(I)<>0.
C    NC=NOMBRE DE NOEUDS CREES . JJ=NOMBRE DE SUJETS NON-AGREGES
C    ITER=COMPTEUR DES ITERATIONS
6     NC=0
      JJ=0
      ITER=ITER+1
      DO 7 IR=1,NR
      I=R(IR)
      K=VO(I)
      IF(K.EQ.0) GOTO 7
      IF(VO(K).EQ.0) GOTO 60
      IF(D(I).NE.D(K)) GOTO 60
      VO(I)=0
      VO(K)=0
      II=II+1
      LI=SOM(I)
      LK=SOM(K)
      IP(II)=IP(LI)+IP(LK)
      SOM(I)=II
      JU=JU+1
      A(JU)=LI
      B(JU)=LK
      ALI=D(I)
      ANE=ANE+ALI
      DI(JU)=ALI
C      REMPLACEMENT DE I PAR MOYENNE DE I ET K ( I=I U K )
      PI=FI(I)
      PK=FI(K)
      PA=PI+PK
      DO 8 J=1,NJ
8     FIJ(I,J)=(PI*FIJ(I,J)+PK*FIJ(K,J))/PA
      FI(I)=PA
      D(I)=AFINI
      NC=NC+1
      R(NC)=I
      GOTO 7
60    JJ=JJ+1
      IVO(JJ)=I
7     CONTINUE
      ICI=NC
C     ICI EST LE NOMBRE DE NOEUDS CREES.( NC VA DEVENIR LE NOMBRE DE
C     NON-INCHANGES APRES LE TEST D' ARRET.)
      IF(JU.GE.NTOF) GOTO 9
```

```
C     RECHERCHE DES CELIBATAIRES ET DES INCHANGES.
C     ETAPE SERVANT A DEFINIR DEUX SORTES DE SUJETS:
C        1 I EST DIT INCHANGE S'IL N'A PAS ETE AGREGE ET SON PLUS
C          PROCHE VOISIN VO(I) NON PLUS
C        2 TOUS LES AUTRES ELEMENTS (=CELIBATAIRES+NOEUDS)
C      ON MET LES NUMEROS DES INCHANGES DANS IVO ET LES NUMEROS DES
C      AUTRES DANS R.LL EST LE NOMBRE D'INCHANGES.
      LL=0
      IF(JJ.LE.0) GOTO 23
      DO 30 IR=1,JJ
      I=IVO(IR)
      IF(VO(VO(I)).NE.0) GOTO 31
      D(I)=AFINI
      NC=NC+1
      R(NC)=I
      GOTO 30
31    LL=LL+1
      IVO(LL)=I
30    CONTINUE
C     ICEL=NOMBRE DES CELIBATAIRES
      ICEL=NC-ICI
C     MISE A JOUR DES TABLEAUX D ET VO SUR LES ELEMENTS NON-INCHANGES.
C     POUR CHAQUE NON-INCHANGE R(I),I VARIANT DE 1 A NC ,ON INITIALISE
C     TAB,IST,DIC,PI.PUIS LA RECHERCHE DU PLUS PROCHE VOISIN SE FAIT
C     EN 2 BOUCLES:
C     LA PREMIERE ( 15 ) FAIT CROISER LES NON-INCHANGES ENTRE EUX ET
C     LA SECONDE ( 19 ) CROISE INCHANGES ET NON-INCHANGES.
23    I=0
13    I=I+1
      I1=R(I)
      DIC=D(I1)
      IST=VO(I1)
      PI=FI(I1)
      DO 14 J=1,NJ
14    TAB(J)=FIJ(I1,J)
      IF(I.GE.NC) GOTO 16
C     BOUCLE 15 : NON-INCHANGES ENTRE EUX
      ISU=I+1
      DO 15 K=ISU,NC
      I2=R(K)
      DD=DIC
      DK=D(I2)
      IF(DD.LT.DK) DD=DK
      DDK=0.
      PON=PI*FI(I2)/(PI+FI(I2))
      DO 17 J=1,NJ
      RAP=TAB(J)-FIJ(I2,J)
      DDK=DDK+PON*RAP*RAP
      IF(DDK.GE.DD) GOTO 15
17    CONTINUE
      IF(DDK.GE.DIC) GOTO 18
      DIC=DDK
      IST=I2
18    IF(DDK.GE.DK) GOTO 15
      D(I2)=DDK
      VO(I2)=I1
15    CONTINUE
C     BOUCLE 19 : NON-INCHANGES ET INCHANGES.D(I1) ET VO(I1) SONT
C     COMPLETEMENT DEFINIS A L'ETIQUETTE 70 : ON PEUT DONC PASSER AU
C     NON-INCHANGE SUIVANT ( SI I<NC ).
16    CONTINUE
      IF(LL.LE.0) GOTO 70
      DO 19 K=1,LL
      I2=IVO(K)
      DDK=0.
      PON=PI*FI(I2)/(PI+FI(I2))
      DO 22 J=1,NJ
      RAP=TAB(J)-FIJ(I2,J)
      DDK=DDK+PON*RAP*RAP
      IF(DDK.GE.DIC) GOTO 19
22    CONTINUE
      DIC=DDK
      IST=I2
19    CONTINUE
70    D(I1)=DIC
      VO(I1)=IST
      IF(I.LT.NC) GOTO 13
C     LES TABLEAUX D ET VO SONT MAINTENANT A JOUR.IL RESTE A COMPLETER
C     R AVEC LES NUMEROS DES INCHANGES AVANT DE PASSER A L'ITERATION
C     SUIVANTE.( NR=NOMBRE DE SUJETS RESTANT A HIERARCHISER )
      IF(LL.LE.0) GOTO 62
      DO 61 IS=1,LL
      NC=NC+1
61    R(NC)=IVO(IS)
62    NR=NC
```

```
      PRINT 26,ITER,ICI,ICEL,LL,NR,ANE
26    FORMAT(1X,I4,4I7,1X,E10.5)
      GOTO 6
9     CONTINUE
      PRINT 27,ANE
27    FORMAT(1X///1X,'SOMME INDICES DE NIVEAUX: ',E10.5)
      RETURN
      END
      SUBROUTINE CORRES(FI,FJ,FIJ,TAB)
      REAL FI(NI),FJ(NJ),FIJ(NI,NJ),TAB(NJ)
      INTEGER FMT(20)
      COMMON /PAR/ NI,NJ,IOP,NTOT,NTOF
      COMMON /IO/ ILEC,ISOR,FMT
      DO 1 J=1,NJ
1     FJ(J)=0.
      T=0.
      DO 2 I=1,NI
      READ(ILEC,FMT) TAB
      FI(I)=0.
      DO 2 J=1,NJ
      A=TAB(J)
      FIJ(I,J)=A
      FI(I)=FI(I)+A
      FJ(J)=FJ(J)+A
2     T=T+A
      DO 3 J=1,NJ
3     FJ(J)=SQRT(FJ(J)/T)
      DO 4 I=1,NI
      FI(I)=FI(I)/T
      DO 4 J=1,NJ
4     FIJ(I,J)=FIJ(I,J)/(T*FI(I)*FJ(J))
      RETURN
      END
      SUBROUTINE MESURE(FI,FJ,FIJ,TAB)
      REAL FI(NI),FJ(NJ),FIJ(NI,NJ),TAB(NJ)
      INTEGER FMT(20)
      COMMON /PAR/ NI,NJ,IOP,NTOT,NTOF
      COMMON /IO/ ILEC,ISOR,FMT
      DO 1 J=1,NJ
      FJ(J)=0.
1     TAB(J)=0.
      DO 2 I=1,NI
      FI(I)=1.
      READ(ILEC,FMT) (FIJ(I,J),J=1,NJ)
      DO 2 J=1,NJ
      A FIJ(I,J)
      FJ(J)=FJ(J)+A
2     TAB(J)=TAB(J)+A*A
      AI=NI
      DO 3 J=1,NJ
      FJ(J)=FJ(J)/AI
      V=(TAB(J)-FJ(J)*FJ(J)*AI)/AI
3     TAB(J)=SQRT(V)
      DO 4 I=1,NI
      DO 4 J=1,NJ
4     FIJ(I,J)=(FIJ(I,J)-FJ(J))/TAB(J)
      RETURN
      END
      SUBROUTINE FACTOR(FI,FIJ)
      REAL FI(NI),FIJ(NI,NJ)
      INTEGER FMT(20)
      COMMON /PAR/ NI,NJ,IOP,NTOT,NTOF
      COMMON /IO/ ILEC,ISOR,FMT
      T=0.
      DO 1 I=1,NI
      READ(ILEC,FMT) FI(I),(FIJ(I,J),J=1,NJ)
1     T=T+FI(I)
      DO 2 I=1,NI
2     FI(I) FI(I)/T
      RETURN
      END
      SUBROUTINE SHELL(IVO,DI)
      INTEGER IVO(NTOF)
      REAL DI(NTOF)
      COMMON /PAR/ NI,NJ,IOP,NTOT,NTOF
C     TRI DU TABLEAU DI (INDICES DE NIVEAU ) PAR LA METHODE DE SHELL.
C     REFERENCES : KNUTH , SORTING AND MERGING (1973).
C
      DO 70 I=1,NTOF
70    IVO(I)=I
C     LE TABLEAU DES POINTEURS EST IVO.
      I=1
      NT=NTOF/3
```

```
 71     M=3*M+1
        IF(M.LT.NT) GOTO 71
 72     M=M/3
        IF(M.LE.0) GOTO 75
        K=NTOF-M
        DO 74 I=1,K
        J=I
 73     L J+M
        IF(DI(IVO(L)).GE.DI(IVO(J))) GOTO 74
        KP=IVO(J)
        IVO(J)=IVO(L)
        IVO(L)=KP
        J=J-M
        IF(J.GT.0) GOTO 73
 74     CONTINUE
        GOTO 72
 75     RETURN
        END
        SUBROUTINE DISCO(NOM,IP,IVO,A,B,DI)
        INTEGER A(NTOF),B(NTOF),IP(NTOT),IVO(NTOF),NOM(NTOT),FMT(20)
        REAL DI(NTOF)
        COMMON /PAR/ NI,NJ,IOP,NTOT,NTOF
        COMMON /IO/ ILEC,ISOR,FMT
C       LES 2*NI-1 NUMEROS DES ELEMENTS DE LA HIERARCHIE TOTALE SONT
C       STOCKES DANS LE TABLEAU NOM.
        DO 2 I=1,NTOF
        NOM(I)=I
 2      NOM(IVO(I)+NI)=NI+I
        NOM(NI)=NI
        DO 1 IR=1,NTOF
        I=IVO(IR)
        NOEU=NI+IR
        ICAR=IP(I+NI)
        IAIN=NOM(A(I))
        IBEN=NOM(B(I))
        DNIV=DI(I)
        WRITE(ISOR,100) NOEU,ICAR,IAIN,IBEN,DNIV
 1      CONTINUE
 100    FORMAT(1X,4I5,E20.10)
        RETURN
        END
```

```
        SORTIES DU PROGRAMME HIVOR
        HIERARCHIE DES SUJETS
          NI     NJ    IOP   ILEC   ISOR
         176      7      1     10     11
     (40X,2F20.10/(4E20.10))
```

```
  CARACTERISTIQUES DES ITERATIONS:
ITER NOEUDS CELIB.  INCHA.   TOTAL      INTRA
   1     47     57     25     129  .14947E-01
   2     21     40     47     108  .25049E-01
   3     17     33     41      91  .33253E-01
   4     17     25     32      74  .45959E-01
   5     12     23     27      62  .57017E-01
   6     12     22     16      50  .73270E-01
   7     10     15     15      40  .99410E-01
   8      9     17      5      31  .13457E+00
   9      6      8     11      25  .19534E+00
  10      4      5     12      21  .22122E+00
  11      3      7      3      18  .24195E+00
  12      3      8      4      15  .27628E+00
  13      2      8      3      13  .31919E+00
  14      4      3      2       9  .49431E+00
  15      2      4      1       7  .61220E+00
  16      2      3      0       5  .85013E+00
  17      2      1      0       3  .12183E+01
  18      1      1      0       2  .13796E+01
SOMME INDICES DE NIVEAUX: .10790E+01
```

Commentaire : La première itération a créé 47 noeuds et dénombré 57 célibataires et 25 inchangés. La partition obtenue comporte 129 classes et son inertie intra-classes est 0,014947.