LES CAHIERS DE L'ANALYSE DES DONNÉES

J.-P. BENZÉCRI
F. BENZÉCRI

Sources de programmes d'analyse de données en langage PASCAL : (II) : élaboration des fichiers de facteurs (IIB) : classification ascendante hiérarchique

Les cahiers de l'analyse des données, tome 22, n° 2 (1997), p. 133-154

<http://www.numdam.org/item?id=CAD_1997__22_2_133_0>

Les Cahiers de l'Analyse des Données
Vol. XXII - 1997 - n°2 - pp.133-154.

SOURCES DE PROGRAMMES D'ANALYSE DE DONNÉES EN LANGAGE PASCAL: (II) : ÉLABORATION DES FICHIERS DE FACTEURS (IIB) : CLASSIFICATION ASCENDANTE HIÉRARCHIQUE

[SOURCES PASCAL (IIB)]

J.-P. & F. BENZÉCRI

0 Introduction : enchaînement des programmes de classification

Selon le format des données traitées et le critère d'agrégation adopté, on a conçu de multiples méthodes de Classification Ascendante Hiérarchique (CAH). Dans l'ensemble de programmes que nous avons entrepris de publier, il s'agit exclusivement du critère d'agrégation suivant la variance, appliqué à un ensemble E de points munis de masses et distances. L'ensemble E est toujours donné comme un fichier de facteurs, 'Fac.w', issu d'une analyse de correspondance (que E soit l'ensemble des lignes ou des colonnes, principales ou supplémentaires, du tableau de base; ou l'ensemble des lignes d'un tableau externe). Enfin, l'on utilise l'algorithme d'agrégation avec recherche en chaîne des paires de plus proches voisins réciproques. Car, d'une part, cet algorithme est simple; et, d'autre part, le temps de traitement croît seulement comme le carré du cardinal de l'ensemble E. Si, en appliquant cette méthode, on utilise toutes les ressources d'un Macintosh PowerPC (ce que ne fait pas le compilateur TML2 avec lequel ont été compilées les sources publiées ici), on peut, en quelques minutes, classer plusieurs milliers d'individus.

De la classification proprement dite, (i.e. de la construction d'une hiérarchie binaire sur E), on ne peut séparer le tracé de l'arbre; qu'il s'agisse de la hiérarchie complète, ou de l'arbre associé à une partition. De plus, la pratique nous a enseigné que la structure hiérarchique se comprend mieux si l'on dispose d'aides à l'interprétation; i.e., en bref, si l'on considère, dans l'espace, les centres des classes; l'orientation des dipôles; la contribution de chaque dipôle à l'inertie sur chaque axe... Mais, plus précisément, pour le programme FACOR, l'espace est rapporté au système des axes factoriels; tandis qu'avec VACOR, on remonte aux variables de base du tableau des données, pour considérer le système d'axes que constituent les composantes elles-mêmes de la ligne ou de la colonne associée à tout élément, e, de E.

De ce point de vue, il faut distinguer entre, d'une part, CAH, tracé d'arbre et listage FACOR, qui utilisent, comme seule donnée, le fichier de facteurs Fac.w, afférent à l'ensemble E (et cela, même si les résultats de la CAH sont transmis sous forme de fichiers à la procédure de tracé de l'arbre et à FACOR); et, d'autre part, VACOR, qui élabore conjointement les résultats de la CAH de

Les cahiers de l'analyse des données 0339-3097/97/02/$ 7.00/© Gauthier-Villars

E (issus de Fac.w) et le tableau des données, voire une CAH de l'ensemble, I ou J, des variables de base.

On peut (pourvu que la segmentation du code y suffise) intégrer, en un seul programme, les quatre procédures: classification, choix de la partition et tracé d'arbre, aides à l'interprétation par FACOR et par VACOR. Nous l'avons fait dans un programme CLS (compilé avec la première version de TML): l'utilisateur, ayant choisi un tableau de base, peut faire, pour I, successivement, les trois procédures CAH, arbre, FACOR; et de même pour J; puis VACOR sous ses diverses variantes (cf. IID); et le choix est alors offert de traiter des tableaux externes (ensembles Ea ou Eb, cf. ID) avec les quatre procédures.

Compte tenu de ce que VACOR se distingue par la complexité des informations qu'il traite, nous croyons préférable (même si le système et le compilateur ne limitent pas la complexité du programme; comme c'était le cas avec TML2), de faire de VACOR un programme séparé. Les trois autres procédures, CAH, arbre et FACOR, qui traitent un simple fichier de facteurs, peuvent être fondues en un seul programme.

Le programme CLH, objet du présent article, (compilé en TML2 sans appeler un coprocesseur arithmétique) fait la CAH et l'arbre. Les aides à l'interprétation seront données séparément ensuite: programme FAX, pour FACOR (cf. IIC); programme VAX, pour FACOR (cf. IID).

1 Demande de classification et de tracé d'arbre: structure du programme 'CLH'

1.1 Le programme principal

```
begin;benzecri;exri:=1;
while (exri=1) do begin
 ouvrir;unloadseg(@ouvrir);
 if (exri=1) then begin
  traiter;unloadseg(@traiter);exri:=0;
  write('faut-il faire une autre CAH oui(O) ou non(N) ');
  readln(rec);if not (rec='N') then exri:=1;
  end;end;
readln(rpv);end.
```

Le programme principal consiste simplement en une boucle, appelant deux procédures: 'ouvrir' et 'traiter'. Par 'ouvrir', l'utilisateur choisit le tableau de base et l'ensemble E qui doit être soumis à la CAH; et, si le fichier 'Fac.w' se trouve sur le disque qu'on a proposé, la variable d'existence 'exri' est mise à 1. En ce cas, s'exécute la procédure 'traiter' qui comprend CAH et tracé d'arbre. Éventuellement, l'utilisateur peut demander que soient traitées d'autres données. Nous considérons, ci-après, les procédures 'ouvrir' et 'traiter', telles qu'elles sont dans le listage 'CLH.p'; mais il faut prendre garde que, pour la CAH et l'arbre, 'traiter' appelle des procédures externes, 'cah' et 'arbt', qui renferment tout ce qui concerne la classification proprement dite (cf. *infra*, §§2, 3).

```
program CLH;uses memtypes,quickdraw,osintf,toolintf,uver;
var nombas,nomf,chat:str255;
rep,rec,rea,ref,rpv:char;
er1,resu:integer;
exri,excn:integer;
{$S segCAH} {$U utriz} {$U uCAHz}
procedure cah(nomen:str255);external;
{$S segarb} {$U uarbz}
procedure arbt (nomen:str255);external;
```

En dehors des noms de fichiers et du sigle de l'ensemble E, le programme CLH n'a pour variables, qui lui soient propres, que des caractères de réponse, et deux entiers {exri, excn} avertissant de l'existence d'un fichier de facteurs ou d'une classification.

1.2 Choix des données à traiter: la procédure 'ouvrir'

```
{$S segcom}
procedure ouvrir;begin er1:=0;rep:='N';
while not((rep='O') or (er1=6)) do begin
 rep:='O';exri:=0;
 write('le fichier de base est ');
 readln(nombas);
 if not(setvol(@nombas,10)=noerr) then begin rep:='N';er1:=er1+1;
    writeln('ERREUR le disque manque ') end
   else begin
    write('le sigle de l''ensemble à traiter est ');readln(chat);
    nomf:=concat(nombas,chat,'Fac.w');resu:=verif(@nomf) div 2;
    exri:=resu;
    if (exri=0) then writeln
       ('NB il n''y a pas de fichier de facteurs pour classer les ',chat);
    nomf:=concat(nombas,chat,'cnabd');resu:=verif(@nomf) div 2;
    excn:=resu;
    if (excn=0) then writeln
       ('NB il n''y a pas de classification déjà faite pour les ',chat);
    if (exri=0) then begin rep:='N';er1:=er1+1;writeln
     ('ERREUR les données manquent pour la classification') end;
   end;
 if (rep='O') then begin
   writeln('le nom du fichier de base, ',nombas);
   write('et de l''ensemble, ',chat,', sont-ils confirmés oui(O) ou non(N) ');
   readln(rep) end;
end;end;
```

A la différence de ce qui est fait par 'sigsuper' dans 'qorlsup' (ID§2) ou par 'principer' dans 'discri' (IIA§1.1), la procédure 'ouvrir' ne fait aucun cas de la forme du sigle, mais s'enquiert seulement de l'existence du fichier de facteur, nommé 'nomf', qui lui est seul nécessaire.

L'utilisateur est averti de l'absence éventuelle d'une CAH déjà faite sur les mêmes données. Sinon, dans 'traiter', le choix sera offert d'utiliser une telle CAH. En effet, l'utilisateur ayant étudié un listage, présentant une partition définie par les noeuds les plus hauts, a pu choisir les noeuds à spécifier pour une nouvelle partition (cf. §3.2.2). Il y a quelques années, il importait de ne pas faire deux fois de suite le long calcul d'une CAH; aujourd'hui, avec un PowerPC, cette économie importe peu.

1.3 Appel des procédures de traitement: la procédure 'traiter'

```
{$S segtrai}
procedure traiter;
begin
 nomf:=concat(nombas,chat);rea:='N';rec:='N';
 if (excn=1) then begin
   writeln('faut-il utiliser une classification déjà faite pour ');
   write('l''ensemble des ',chat,' oui(O) ou non(N) ');readln(rec) end;
 if not (rec='O') then begin excn:=0;
  write('faut-il classer l''ensemble des ',chat,' oui(O) ou non(N) ');
  readln(rec);
  if not (rec='N') then begin excn:=1;
   cah(nomf);unloadseg(@cah);readln(rpv) end end;
 if (excn=1) then begin
  write('faut-il faire arbre et partition des ',chat,' oui(O) ou non(N) ');
  readln(rea);
  if not (rea='N') then begin
   arbt(nomf);unloadseg(@arbt);readln(rpv) end end;
end;
```

Telle qu'elle est publiée ici, la procédure 'traiter' n'est qu'un dialogue commandant l'appel des procédures de classification, 'cah'; et de partitition et tracé d'abre, 'arbt'. On remarquera que le seul argument transmis à ces procédures est la chaîne de caractères 'nomf'; avec laquelle, par addition de suffixes, 'Fac.w', 'cnabd'..., sont formés les noms de tous les fichiers afférents à l'ensemble E traité. Reste à étudier les procédures 'cah' et 'arbt'.

2 Classification Ascendante Hiérarchique: la procédure 'cah'

```
UNIT uCAHz;
INTERFACE uses memtypes,quickdraw,osintf,toolintf,uver;
procedure cah(nomen:str255);
implementation
procedure trier(parp,ppra,pVr,ptVr:ptr;cr:integer);external;
procedure cah;
 var  i,al,amax,cari,carf,s,smin,sp,cars,carm,n,c,id,ip,
  cn,li,q,r,ng,np,eps,ire,exp,dir:integer;
  dmin,pod,pop,rdf,res,rem:extended;
  nomfi: str255;rep:char;
  firi:file of integer;riri:ricla;ptrcl:ptrc;f:file of integer;
  An,Bn,Tn,ids,cs,tTn,anp,pna:ptni;
  poi,dms,dn:pti;pp:ptr;
  Fai:array[1..aamax] of pti;
```

La procédure 'cah' est contenue dans l'unité 'uCAHz.p'. En tête de l'implémentation, est déclarée une procédure externe, 'trier', elle-même contenue dans une autre unité, 'utriz.p', dont le nom est déclaré dans le programme principal 'CLH.p' (cf. §1.1), avec 'uCAHz.p', en tête du segment: {$S segCAH}.

Vient ensuite l'implémentation de la procédure 'cah' elle-même; qui débute par les déclarations de variables. On notera la mention 'extended', qui signale que, bien que les données de 'Fac.w' soient en format 'single' des calculs intermédiaires sont faits avec la précision maxima. On reconnaît, d'autre part, le fichier 'firi', lu en format entier mais compris sous le format 'ricla', pour acquérir les facteurs, vers lesquels pointent les Fai; etc...

2.1 Les procédures et fonctions appelées par 'cah'

2.1.1 Tri par fusion d'un ensemble de nombres entiers: la procédure 'trier' de l'unité 'utriz.p'

```
UNIT utriz5;
INTERFACE uses memtypes,quickdraw,osintf,toolintf,uver5;
procedure trier(parp,ppra,pVr,ptVr:ptr;cr:integer);
implementation procedure trier;
var arp1,pra1,Vr1,tVr1:ptni; ina,la,lb,n,na,nb,nz,x:integer;
begin
arp1:=ptni(parp);pra1:=ptni(ppra);Vr1:=ptni(pVr);tVr1:=ptni(ptVr);
for n:=1 to cr do arp1^[n]:=n;la:=1;ina:=1;
while (ina+la<=cr) do begin
    x:=cr-(ina+la-1);if(x<la)then lb:=x else lb:=la;
    na:=ina;nb:=ina+la;
    for n:=ina to (ina+la+lb)-1 do begin
       if(na<ina+la) and (nb<ina+la+lb) then
          if(Vr1^[na]<=Vr1^[nb]) then begin nz:=na;na:=na+1 end
            else begin nz:=nb;nb:=nb+1 end
       else if (ina+la=na) then begin nz:=nb;nb:=nb+1 end
            else begin nz:=na;na:=na+1 end ;
       tVr1^[n]:=Vr1^[nz] ; pra1^[n] :=arp1^[nz] end;
    for n:=ina to ina+la+lb-1 do begin
       Vr1^[n]:=tVr1^[n];arp1^[n]:=pra1^[n] end;
    ina:=ina+la+lb;
    if (cr<ina+la) then begin la:=2*la;ina:=1 end;end;
for n:= 1 to cr do pra1^[arp1^[n]]:=n;
end;end.
```

La procédure 'trier' ne diffère de 'trire', décrite dans IA§4, qu'en ce que celle-ci trie des réels et celle-là des entiers. Aussi, les pointeurs Vr1 et tVr1 sont-ils déclarés, ci-dessus, avec le type: 'ptni', comme pointant chacun vers une suite de valeurs entières (et non avec le type: ptdri, pointeur vers une suite de réels, comme dans 'trire'). Nous ne répéterons pas l'explication de l'algorithme de fusion; nous bornant à reprendre la description précise de l'effet de 'trier' sur le contenu des tableaux qui lui sont transmis par des pointeurs, avec le cardinal, cr, de l'ensemble à trier.

Expliquons le rôle des tableaux d'entiers arp1^ et pra1^. Initialement, arp1^ contient la suite des entiers de 1 à cr. Ces entiers sont considérés comme les adresses initiales des entiers donnés dans Vr1^. Au cours des itérations de la boucle générale 'while', on effectue, entre les deux tableaux d'entiers arp1^ et pra1^, les mêmes transferts qu'entre les deux tableaux d'entiers, Vr1^ et tVr1^; en sorte que, après exécution de la procédure 'trire', il y a, d'une part, dans Vr1^, (ainsi que dans tVr1^), l'ensemble des valeurs entières donnés initialement, mais rangés en une suite non décroissante; et, d'autre part, dans arp1^, un rappel des adresses qu'occupaient initialement ces nombres: e.g., si l'entier qui est finalement dans Vr1^[n] était, initialement, dans Vr1^[un], on a arp1^[n]=un. De plus, l'instruction:

```
for n:= 1 to cr do pra1^[arp1^[n]]:=n;
```

met, dans pra1^[un], l'adresse finale, n, de l'entier qui était, initialement, dans Vr1^[un].

2.1.2 Calcul du critère d'agrégation: la fonction 'dss'

Aprés les déclarations de variables, données ci-avant, en tête du §2, l'implémentation de la procédure 'cah' se poursuit avec une procédure et deux fonctions; dont la plus importante, 'dss', sert au calcul du critère.

```
function dss(s1,s2:integer):extended;
 var i1,i2,af:integer;dd,poire:extended;
 begin  i1:=ids^[s1];i2:=ids^[s2];dd:=0;af:=0;
 poire:=poi^[i1]*poi^[i2];
 if not(poire=0) then poire:=poire/(poi^[i1]+poi^[i2]);
 while (dd<dmin) and (af<carf) do begin af:=af+1;
       rdf:=Fai[af]^[i1]-Fai[af]^[i2];dd:=dd+(poire*sqr(rdf));end;
 dss:=dd;end;
```

On sait que l'algorithme de CAH part d'un ensemble, E, d'éléments, et procède en agrégeant les paires de plus proches voisins réciproques, pour créer des nœuds. À une étape donnée de la CAH, on appelle 'sommet' tout élément, ou nœud, qui n'a pas encore été agrégé; chacun des autres est le descendant d'un sommet. Initialement, il n'y a pas de nœuds, et les éléments de E sont tous des sommets; finalement, il n'y a plus qu'un sommet, dont descendent, directement ou indirectement, tous les autres nœuds et tous les éléments de E.

La fonction 'dss' donne, pour deux sommets désignés par des numéros d'ordre {s1, s2}, la distance au carré, ou plus exactement, la quantité critère d'agrégation, qui n'est autre que l'inertie du dipôle constitué par les deux sommets; suivant la formule crit = ((m1.m2)/(m1+m2)).d^2(s1, s2).

Dans la procédure dss, les données afférentes au sommet de rang s sont prises, sous l'indice ids^[s], dans des tableaux, initialement remplis pour les éléments de E, mais dont le contenu se modifie; quand les coordonnées (sur les axes factoriels) et le poids d'éléments qu'on agrège sont remplacés par ceux afférents aux sommets créés. Ainsi, la masse m du sommet s est dans: poi^[ids^[s]]; le facteur de rang a est dans: Fai[a]^[ids^[s]].

De plus, dss étant appelée pour chercher un minimum, le calcul du critère, comme somme de termes afférents aux facteurs, est arrêté, dès qu'est dépassé le minimum, dmin, précédemment trouvé. On a procédé de même en analyse discriminante: cf. IIA§2.1.

2.1.3 Écriture d'un entier en virgule flottante: la procédure 'codire'

```
procedure codire;begin eps:=1;exp:=-4;rem:=10000*res;
 if (rem=0) then exp:=0;
 if (rem<0) then begin eps:=-1;rem:=-rem end;
 if (0<rem) then begin
    while (9999<rem) do begin exp:=exp+1;rem:=rem/10 end;
    while (rem<1000) do begin exp:=exp-1;rem:=rem*10 end end;
 ire:=eps*round(rem);end;
```

La procédure 'codire', appelée une fois, a pour donnée le nombre réel 'res': et met 'res' sous la forme d'un entier de 4 chiffres, ire, multiplié par une puissance de 10 dont l'exposant est exp.

2.1.4 Renumérotage de l'arbre suivant l'ordre des niveaux: la fonction 'pca'

```
function pca(ca:integer):integer;var k :integer;begin
    if(ca<=cn+1) then k:=ca else k:=cn+1+pna^[ca-(cn+1)];pca:=k;end;
```

Ainsi qu'on l'a dit au §2.1.1, l'algorithme ne crée pas, initialement, les nœuds dans l'ordre strict des niveaux croissants. Une fois la hiérarchie construite, les nœuds sont renumérotés (cf. §2.4). Il faut, d'autre part, prendre garde que, considérés séparément, les nœuds peuvent être numérotés de 1 à cn=cari-1 (nombre diminué de 1 des éléments de E); mais si l'on considère comme un ensemble unique les éléments de E (terminaux) et les nœuds, ceux-ci doivent être renumérotés de cari+1 à (2.cari)-1. D'où la complexité de la fonction 'pca', qui reçoit pour argument, ca, un numéro qui peut être celui d'un élément terminal ou d'un nœud; et n'applique que dans ce dernier cas le tableau de permutation des nœuds, pna, créé par la procédure 'trier'.

2.2 Zones réservées en mémoire centrale pour la procédure 'cah'

```
begin {cah} ptrc1:=ptrc(@riri);
 nomfi:=concat(nomen,'Fac.w');reset(firi,nomfi);
 with riri do begin
  for a1:=1 to titab-1 do read(firi,i);read(firi,cari);
  pp:=newptr(4*cari);poi:=pti(pp);
  pp:=newptr(4*cari);dms:=pti(pp);
  pp:=newptr(4*cari); dn:=pti(pp);
  pp:=newptr(2*cari); An:=ptni(pp);
  pp:=newptr(2*cari); Bn:=ptni(pp);
  pp:=newptr(2*cari); Tn:=ptni(pp);
  pp:=newptr(2*cari);ids:=ptni(pp);
  pp:=newptr(2*cari); cs:=ptni(pp);
  pp:=newptr(2*cari);tTn:=ptni(pp);
  pp:=newptr(2*cari);anp:=ptni(pp);
  pp:=newptr(2*cari);pna:=ptni(pp);
  writeln('cardinal =',cari:5);
```

La procédure 'cah' ouvre le fichier des facteurs et lit d'abord le cardinal, cari, de l'ensemble, E, à classer; afin de calculer, en fonction de cari, l'espace à réserver en mémoire centrale.

Par les entiers An^[n], Bn^[n] et Tn^[n], sera donnée la description de la hiérarchie: aîné, benjamin et taux d'inertie, pour le n-ème nœud.

Dans dms^[so] et dn^[n], seront les valeurs, du critère d'agrégation, respectivement, pour le sommet de rang so de la chaîne (de longueur variable: carm+1, cf. *infra*, §2.3); et pour le n-ème nœud créé.

On a déjà dit, au §2.1.2, que le tableau des entiers: ids^[s], relie le numéro, s, d'un sommet actuel, à celui de la ligne, i, du tableau où sont rangés, en l'état du calcul, les facteurs et poids afférents à s. Dans poi^[ids^[s]], est le poids (ou fréquence) du sommet s; et cs^[s], est le numéro de s, au sein de l'ensemble des éléments de E; plus précisément, dans E prolongé par la suite des nœuds déjà créés (cf. §2.1.4); on dira, en bref: dans E étendu, dans: Eé.

Les pointeurs tTn, anp, pna, sont des arguments transmis à 'trier' (cf. *supra*, §2.1.1) pour renuméroter les nœuds suivant l'ordre des niveaux.

```
read(firi,i);amax:=i-2;dir:=7+(2*amax);
for al:=1 to 3*(amax+2) do read(firi,i);
writeln('le nombre de facteurs disponibles est',amax:3);
write('nombre de facteurs à utiliser = '); readln(carf);
if (amax<carf) then carf:=amax; if (carf<1) then carf:=1;
for al:=1 to carf do
   begin pp:=newptr(4*cari);Fai[al]:=pti(pp) end;
for i:=1 to cari do begin
   for al:=1 to dir do read(firi,ptrc1^[al]);poi^[i]:=Fac[amax+1];
   for al:=1 to carf do Fai[al]^[i]:=Fac[al] end end;
close(firi);cars:=cari;n:=0;carm:=0;dms^[1]:=1e+19;
for s := 1 to cars do begin ids^[s]:=s;cs^[s]:=s;end;
```

On sait, (cf. IB§2.1.3), que le fichier 'Fac.w' contient un tableau dont les colonnes sont les amax facteurs retenus, suivis de 'poid' et 'dis2'. La procédure 'cah' n'utilise que les facteurs et le 'poid'; plus précisément, le nombre, carf, des facteurs à retenir est fixé par l'utilisateur.

La phase de lecture, commencée (au début du présent §) par la mention: with riri do begin..., s'achève. On ferme le fichier: firi, des données. Le nombre, cars, des sommets, est égal au nombre, cari, des éléments de E; et le numérotage ids^ est l'identité ainsi que cs^. Il n'y a pas encore de nœud (n=0); ni de chaîne (nombre des maillons, carm=0); le critère, dont on doit chercher le minimum, est mis arbitrairement haut: dms^[1]=1e+19.

2.3 L'algorithme de recherche en chaîne des voisins réciproques

Le nombre des nœuds doit être, finalement: cari-1; la recherche des nœuds se poursuit, dans la boucle 'while', tant que le numéro, n, n'a pas atteint cette valeur (on notera qu'il s'agit, ici, du numérotage des nœuds, à partir de 1, indépendamment des élements: cf. *supra*, §2.1.4).

Il faut se rappeler que, dans la recherche en chaîne des paires de voisins réciproques, à un instant donné du déroulement de l'algorithme, il y a une chaîne formée de carm maillons, donc de: carm+1 sommets, dont on a fait en sorte que les numéros aillent en croissant, de 1 à carm+1. Dans cette chaîne, chaque sommet (de rang 1 à carm) est placé immédiatement avant un autre sommet; lequel, au sein de l'ensemble des sommets (,considéré en son état actuel), est son plus proche voisin; la valeur du critère est notée: dms^[so], pour la paire {so-1, so} de sommets consécutifs de la chaîne.

Notons, simplement: s=carm+1, le numéro du dernier sommet de la chaîne. On sait déjà, en bref, par récurrence, que, dans la chaîne, c'est le pénultième, de rang carm, qui est le plus proche de s; avec, pour valeur du critère: dmin=dms^[s]. Par la boucle: for sp:=s+1 to cars do..., on cherche, parmi les autres sommets (numérotés au-delà de s) celui qui, avec s, réalise le minimum du critère; plus précisément, on s'intéresse exclusivement au cas où ce minimum est strictement inférieur à la valeur, dmin qu'on a avec le prédécesseur dans la chaîne.

Si un tel élément existe, est réalisée l'éventualité (s<smin); le sommet smin est ajouté à la chaîne, avec le rang s+1; tandis que l'ancien sommet s+1

```
while (n<cari-1) do begin
 s:=carm+1;dmin:=dms^[s];smin:=carm;
 for sp:=s+1 to cars do if (dss(s,sp)<dmin) then
        begin dmin:=dss(s,sp);smin:=sp end;
 if (s<smin) then  begin
   id:=ids^[smin];ids^[smin]:=ids^[s+1];ids^[s+1]:=id;
   c := cs^[smin]; cs^[smin]:= cs^[s+1]; cs^[s+1]:= c;
   carm:=carm+1;dms^[carm+1]:=dmin; end
 else begin
   id :=ids^[s-1];ip:=ids^[s];pod:=poi^[id];pop:=poi^[ip];
   res:=pod+pop;if not(res=0) then res:=1/res;
   for al := 1 to carf do
      Fai[al]^[id]:=(pod*Fai[al]^[id]+pop*Fai[al]^[ip])*res;
   poi^[id]:=pod+pop;n:=n+1;
   writeln('n=',n:3,' carm=',carm:3,'  niv=',dmin:3);
   An^[n]:=cs^[s-1];Bn^[n]:=cs^[s];dn^[n]:=dmin;
   if (Fai[1]^[id]<Fai[1]^[ip]) then begin
      An^[n]:=cs^[s];Bn^[n]:=cs^[s-1] end;
   ids^[s]:=ids^[cars];cs^[s]:=cs^[cars];cs^[s-1]:=n+cari;cars:=cars-1;
   if (carm=1) then carm:=0 else carm:=carm-2;end;end;
```

reçoit le rang smin; par le fait, les valeurs afférentes à s+1 et smin doivent être échangées dans les tableaux d'adresses; ids^, cs^.

Si, au contraire, le sommet le plus proche de s, au sens du critère, n'est autre que s-1, c'est que {s-1, s} est, au sein de l'ensemble actuel des sommets, une paire de plus proches voisins réciproques: cette paire doit être agrégée pour créer un nouveau nœud: tel est l'objet de l'instruction composée: else begin...

Dans les tableaux Fai[a]^ et poi^ (facteurs et poids de l'ensemble des sommets en l'état), les informations afférentes à ce nouveau sommet remplaceront celles afférentes à s-1 (avec l'indice: i=ids^[s-1]). Le poids du nouveau nœud n'est autre que la somme des poids des nœuds agrégés. Les valeurs des facteurs se calculent par la formule barycentrique usuelle.

On affiche à l'écran, après l'ordre de création du nœud: n, le nombre de maillons de la chaîne: carm, et le niveau d'agrégation: dmin.

Dans les tableaux An et Bn, sont inscrits les numéros (dans: Eé = E étendu) des nœuds agrégés; soit: cs^[s] et cs^[s-1]. Et, dans dn^[n] est mis le niveau du nouveau nœud.

Pour la suite, dans le dessin de l'arbre, il est bon que la qualité d'aîné soit attribuée, systématiquement, à celui des deux nœuds agrégés qui est situé le plus à droite sur l'axe 1: ainsi, autant que faire se peut, les terminaux sont rangés (à la marge gauche de l'arbre) dans l'ordre de F1 décroissant.

De la suite des cars sommets, ceux de rang {s-1, s} ont été fondus en un nouveau nœud, auquel a été donné le rang s-1; au rang s, resté vide, on place le dernier des nœuds préexistants, i.e., celui dont le rang était cars. Quant au nouveau nœud, il a, dans Eé (i.e.: E étendu), le rang: cs^[s-1]=n+cari. Le nombre des sommets: cars, est diminué d'une unité. Si la chaîne ne comptait qu'un maillon: carm=1, elle est maintenant vide: carm=0; en général, par la disparition de s et s-1, qui ont été fondus, sa longueur est diminuée de 2.

2.4 Tri des niveaux d'agrégation des nœuds; calcul et écriture du fichier des résultats définitifs

```
cn:=cari-1;res:=0;for n:=1 to cn do res:=res+dn^[n];codire;res:=1/res;
for n:=1 to cn do Tn^[n]:=round(10000*res*dn^[n]);
trier(@anp^,@pna^,@Tn^,@tTn^,cn);
for n:= 1 to cn do tTn^[n]:=An^[n];
for n:=1 to cn do  An^[n]:= pca(tTn^[anp^[n]]);
for n:= 1 to cn do tTn^[n]:=Bn^[n];
for n:=1 to cn do  Bn^[n]:= pca(tTn^[anp^[n]]);
nomfi:=concat(nomen,'cnabd');rewrite(f,nomfi) ;
write(f,cn+10000);write(f,ire+10000);
exp:=(carf*256)+128+exp-32768;write(f,exp);
for n:=1 to cn do begin
     write(f,An^[n]);write(f,Bn^[n]);write(f,Tn^[n]) end;
close(f);
```

On calcule dans res la somme des indices de niveaux des cn nœuds de la hiérarchie (cette somme n'est autre que l'inertie totale du nuage N(E) dans l'espace rapporté aux carf axes factoriels retenus pour la CAH; dans les cas: E=Ip, et: E=Jp, cette inertie totale n'est autre que la somme des carf premières valeurs propres issues de l'analyse du tableau de base). Par 'codire' (cf. §2.1.3), le nombre réel: res, est codé suivant un entier de 4 chiffres, ire, multiplié par une puissance de 10 dont l'exposant est exp. On calcule ensuite, dans Tn^[n], le taux d'inertie afférent au nœud n.

Après la procédure 'trier', (cf. §2.1.1), les nœuds (considérés selon le numérotage de 1 à n, par ordre de création) sont renumérotés suivant l'ordre des niveaux croissants (plus exactement, des taux: Tn^[n], croissants). Bien que les Tn^[n] soient calculés à 1 dix-millième près, la procédure de tri est telle qu'il est exclu qu'un nœud reçoive un rang inférieur à celui d'un de ses descendants (car, en bref, entre des nœuds de même niveau, est respecté l'ordre initial, qui est celui de la création des nœuds). Le tableau: anp^, donne l'ancien numéro du nœud, en fonction du nouveau; et, réciproquement: pna^ , donne le nouveau uméro en fonction de l'ancien.

Les taux Tn^[n] sont bien rangés. Reste à recalculer les {An^[n], Bn^[n]} dans le nouveau numérotage de l'ensemble Eé (i.e.: E étendu). Ceci est fait grâce à la fonction de renumérotage: 'pca' (cf. §2.1.4).

En tête du fichier d'entiers, 'cnabd', sont écrits sur 3 entiers, les quatre nombres: cn (i.e.: cari-1); {ire, exp} (inertie totale écrite en virgule flottante); carf (nombre des facteurs utilisés). Le calcul du troisième entier écrit est complexe: mais cette formule a servi à assurer la compatibilité avec des fichiers 'cnabd' créés par des versions antérieures du logiciel; en sorte que tous les formats sont acceptés par la procédure 'arbt' (cf. §3).

Après les trois entiers de l'en-tête, le fichier 'cnabd' contient les triplets {An^[n]), Bn^[n]), Tn^[n]}, rangés suivant l'ordre croissant du nouveau numérotage des nœuds.

2.5 Affichage à l'écran de résultats partiels

```
q:=1+((cn-1) div 15); ng:=cn+15;if(5<q) then q:=5;
for li :=1 to q do begin
  ng:=ng-15;np:=ng-14;if(np<1) then np:=1;
  write('c  ');for n:=ng downto np do write(n+cn+1:5);
  writeln;
  write('T  ');for n:=ng downto np do write(Tn^[n]:5);
  writeln;
  write('A  ');for n:=ng downto np do write(An^[n]:5);
  writeln;
  write('B  ');for n:=ng downto np do write(Bn^[n]:5);
  writeln  end;
 dispose(An);dispose(Bn);dispose(Tn);dispose(ids);
 dispose(cs);dispose(tTn);dispose(anp);dispose(pna);
 dispose(poi);dispose(dms);dispose(dn);
 for al:=1 to carf do dispose(Fai[al]);end;{cah}end.
```

Dans l'exécution du programme 'CLH', la procédure 'cah' précède immédiatement le choix de la partition retenue pour le listage de la clasification. Ce choix, au premier abord, repose sur des données formelles simples: principalement, la suite des niveaux des nœuds. C'est pourquoi 'cah' affiche à l'écran un tableau partiel des résultats. (Ultérieurement, si, cf. §1.3 l'utilisateur de 'CLH' reprend une classification déjà faite, 'arbt' est appelée sans affichage de résultats par 'cah'; mais, en ce cas, la partition peut avoir été choisie d'après un premier listage; non seulement suivant des critères formels; mais d'après l'interprétation du contenu des classes).

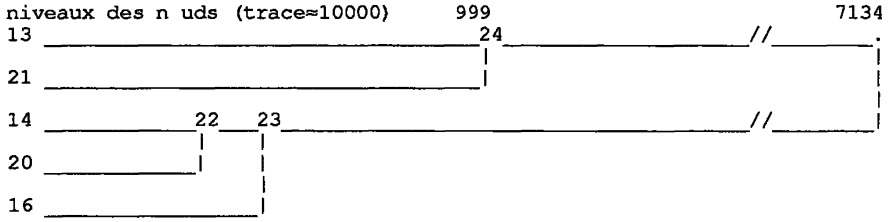
La procédure 'cah' s'achève en libérant les zones affectés en mémoire centrale à divers pointeurs.

3 Choix d'une partition et tracé d'arbre: la procédure 'arbt'

```
UNIT uarbz;
INTERFACE uses memtypes,quickdraw,osintf,toolintf,uver;
procedure arbt(nomen:str255);
implementation procedure arbt;
 const qmax=100;lmax=136;
 var
  cari,i,j,c,cn,li,n,ng,np,ve,w,d,q,den,nivc,nivn,amax,dir,u,vin,carf,
  reste,carq,long,larg,ir,ex,cA,cB,m,noir:integer;
  res:extended;req,ret,rhs,cht,rec:char;
  An,Bn,Tn,Carn,nin,ilil,ren:ptni;pp:ptr;
  PER,nlic,qdc:ptni;clq,cdm:array[1..qmax] of integer;
  vert:array[0..255] of integer;
  pnmi:pzgi;nomf,lin1,lin2:str255;
  riri:ricla;ptrcl:ptrc;
  fin:file of integer;firi:file of integer;ft:text;
 function carc(cl:integer):integer;var cc:integer;begin
  if cl<=cari then cc:=1 else cc:=Carn^[cl-cari];carc:=cc end;
```

L'unité 'uarbz.p' contient uniquement l'implémentation de la procédure 'arbt' elle-même. D'abord les constantes: qmax, nombre maximum des classes de la partition retenue; lmax, largeur maxima du listage à créer. Puis les variables, où l'on reconnaît des notations de 'cah'; et qu'on expliquera ensuite. La fonction 'carc' donne le cardinal de tout élément cl de Ee: individu (cl≤cari), ou nœud (dont le cardinal est lu sur le tableau Carn^). Vient ensuite la procédure 'traceli'; de tracé du listage d'un arbre, ligne par ligne.

3.1 Tracé d'une ligne du listage d'un arbre: la procédure 'traceli'



Le lecteur doit avoir étudié des listages d'arbre publiés dans *CAD*. L'exemple succint donné ici, ne sert qu'à soutenir le commentaire de l'algorithme par une description minutieuse; où sont introduites les notations sur lesquelles on reviendra dans la suite.

Il s'agit de la partition en 5 classes d'un ensemble E de 13 individus: ce qu'on voit d'après le numérotage de Eé (ensemble E, des cari individus, étendu par les cari-1 nœuds), qui s'arrête à (2.cari)-1=25; car le sommet de l'arbre, marqué seulement d'un point, a pour aîné: '24' et pour benjamin: '23'.

Sur la 1-ère ligne, on a, à gauche, le numéro (dans Eé) d'une classe (en fait, réduite à un élément: i=13). Puis un trait, qui va jusqu'au sommet de l'arbre; interrompu par le numéro: 24, rang, dans Eé, du nœud dont 13 est le descendant immédiat; i.e. de ce qu'on appelle: le père de 13. Dans la suite des nœuds, numérotés de 1 à cari-1, ce nœud a pour numéro: n=(24–13)=11. On note: 13=A^[11]; et, de plus: PER^[13]=11. La 1-ère ligne s'arrête au sommet.

En général, on a sur le trait qui part d'un élément cé de Eé, inscrit à gauche, les numéros de nœuds qui en sont les ancêtres: PER^[cé], PER^[PER^[cé]], ...; mais en s'arrêtant sur un trait vertical quand on parvient à un ancêtre aé dont cé descend par son benjamin: B^[aé]. Ce dont la classe 14 offre un exemple, avec la lignée: 14 = A^[9] ; 22 = A^[10] ; 23 = B^[12]. Formules où il faut prendre garde que les nœuds de rang {9, 10, 12} sont renumérotés, dans Eé: 22, 23; 25.

Quand, sur un trait horizontal, s'inscrit le numéro, aé, d'un nœud, descend de ce numéro, un trait vertical; lequel, après un coude vers la gauche, rejoint le benjamin de aé. C'est ce qu'on voit, sur l'exemple, pour le sommet; et les autres nœuds de l'arbre: numérotés {22, 23, 24}, dans Eé; mais {9, 10, 11} dans la seule suite des 12 nœuds de la CAH générale de E. Dans le présent graphique, les lignes de rang impair commencent par un sigle; mais chacune est suivie d'une ligne de rang pair, réduite à des traits verticaux. Cet interposition n'est pas obligatoire: nous dirons, si elle est faite, qu'on a passé une ligne.

Reste à signaler qu'avec un grand nombre de classes, il se peut que le réseau des traits verticaux soit si dense, qu'on n'ait pas la place d'écrire les numéros des nœuds; où, même, que plusieurs traits verticaux se superposent. On peut éviter cela en composant l'arbre sur une plus grande largeur qu'il n'est finalement écrit: cette troncature est ici rappelée par des traits obliques; et par la mention des véritables niveaux du sommet et de son aîné.

```
procedure traceli;var dd,vv,ww:integer;begin
  ve:=6;n:=PER^[c];nivn:=nin^[n];
  den:=nivn;ve:=ve+den;lin1:='';lin2:='';
  while c=An^[n] do begin vert[ve]:=vert[ve]+1;
    for dd:=1 to den do lin1:=concat(lin1,'_');
    if (n=cari-1) then begin lin1:=concat(lin1,'.') ;n:=cari end;
    if (n<cn) then begin c:=n+cari;n:= PER^[c];nivc:=nivn;nivn:=nin^[n];
       den:=nivn-nivc;ve:=ve+den;reste:=den-1;
       if (c> 9) then reste:=reste-1;
       if (c>99) then reste:=reste-1;
       if (reste>=0) then  begin
           if (c>99) then lin1:=concat(lin1,chr(48+(c div 100)));
           if (c> 9) then lin1:=concat(lin1,chr(48+((c div 10) mod 10)));
           lin1:=concat(lin1,chr(48+(c mod 10)));
          den := reste;end;end;end;
  if c=Bn^[n] then begin  for dd := 1 to den do lin1:=concat(lin1,'_');
    while (vert[noir]=0) do noir:=noir-1;
    vert[ve]:=vert[ve]-1;lin1:=concat(lin1,'|');
    for ww:=ve+1 to noir do
     if vert[ww]>=1 then lin1:=concat(lin1,'|') else
        lin1:=concat(lin1,' ') end;
  if (larg-5<length(lin1)) then lin1:=copy(lin1,1,larg-5);
  if (req='O') then begin
    while (vert[noir]=0) and (0<noir) do noir:=noir-1;
    for vv:=1 to noir do
      if vert[vv]>0 then lin2:=concat(lin2,'|')
       else lin2:=concat(lin2,' ');
    if (larg<length(lin2)) then lin2:=copy(lin2,1,larg);end;end;
```

La procédure 'traceli' est appelée par 'arbt' pour composer la ligne afférente à la classe numérotée 'c' (dans Eé). Lors du 1-er appel, sont faites les initialisations (cf. *infra*, §3.4.3): for ve:=1 to long do vert[ve]:=0;noir:=long. Ceci veut dire qu'il n'y a aucun trait vertical dont la tracé est en cours. En général, la valeur de vert[ve] est le nombre (éventuellement >1, cf. *supra*) des traits verticaux passant au niveau du caractère de rang ve. On sait que la 1-ère ligne, qui va jusqu'au sommet prendra toute la largeur fixée *a priori*: noir=long. En général, noir est l'espace couvert par une ligne.

On voit, d'après l'exemple, qu'on doit suivre la lignée des ancêtres (de la classe inscrite en marge à gauche de la ligne) jusqu'à rencontrer le sommet, ou un benjamin, ß, dont le père a déjà pris place sur une ligne précédente. D'un ancêtre, on tente d'inscrire le numéro entre son propre niveau: nivc, et celui de son père: nivn (si cela est possible, on a: reste≥0). À son niveau, ve, vert[ve] est augmenté de 1 pour créer la verticale descendant de cet ancêtre vers son benjamin. Au contraire, au-delà de ß, la verticale descendant de son père se termine; et vert[ve] est diminué de 1, pour les tracés ultérieurs.

Quand s'achève la composition de lin1, qui n' est pas toute la ligne afférente à une classe, mais, plus précisément, ce qu'en commentant l'exemple, on a appelé le 'trait', il faut vérifier que, la place du sigle étant comptée, on ne sort pas de la largeur: larg, assignée au listage; sinon lin1 est restreinte à ses larg-5 premiers caractères. La mention: req='O', veut dire: 'passer une ligne': il faut alors créer lin2, que 'traceli' a initialement mise à vide. Pour cette deuxième ligne, on explore toute la longueur (noir) de lin1, mais pour ne retenir, en non-blanc, que les tirets spécifiés dans vert.

3.2 Corps de la procédure 'arbt': lectures des données et paramètres de la CAH

```
begin {arbt} ptrc1:=ptrc(@riri);
nomf:=concat(nomen,'cnabd');reset(fin,nomf);read(fin,cn);cht:='N';
if (9999<cn) then begin
   cn:=cn-10000;read(fin,ir);ir:=ir-10000;
   read(fin,ex);
   if (0<ex) then begin ex:=ex-10000;cht:='O' end
     else begin cht:='F';
        ex:=ex+32768;carf:=ex div 256;ex:=(ex mod 256)-128 end end;
cari:=cn+1;rhs:='z';
```

La procédure 'arbt' commence par ouvrir le fichier 'cnabd' créé par 'cah'.Sont lus dans ce fichier: le nombre des nœuds, cn=cari-1; le nombre, carf, des facteurs utilisés; l'inertie totale, écrite en virgule flottante, {ir, exp}. Plus précisément, comme on l'a dit au §2.4, cette lecture se fait sous un format complexe, qui assure la compatibilité avec des versions antérieures de 'cah'. Mais avec la version de 'cah' publiée au §2, sont disponibles les 4 nombres {cn, ire, exp, carf}; et le caractère 'cht' n'est autre que la lettre 'F'.

3.2.1 Zones réservées en mémoire centrale pour la procédure 'arbt'

```
if (cari<ibmax) then begin pp:=newptr(6*cari);pnmi:=pzgi(pp) end
   else new(pnmi);
pp:=newptr(4*cari); PER:=ptni(pp);
pp:=newptr(4*cari);nlic:=ptni(pp);
pp:=newptr(4*cari); qdc:=ptni(pp);
pp:=newptr(2*cari);  An:=ptni(pp);
pp:=newptr(2*cari);  Bn:=ptni(pp);
pp:=newptr(2*cari);  Tn:=ptni(pp);
pp:=newptr(2*cari);Carn:=ptni(pp);
pp:=newptr(2*cari); nin:=ptni(pp);
pp:=newptr(2*cari);ilil:=ptni(pp);
pp:=newptr(2*cari); ren:=ptni(pp);
for n:=1 to cn do ren^[n]:=0;rec:='N';
```

En donnant les fonctions des divers tableaux créés en mémoire centrale, nous préciserons les notations esquissées, au §3.1, dans le commentaire de la procédure 'traceli'.

La procédure réserve des zones, d'une part, pour les sigles des cari éléments de l'ensemble E (soumis à 'cah'); d'autre part, pour des tableaux d'entiers. Trois tableaux, avec 4.cari octets, sont destinés à contenir 2.cari nombres entiers; i.e., une fonction sur l'ensemble E étendu, Eé, des éléments et des nœuds. Les sept autres tableaux, avec 2.cari octets, contiennent cari entiers: il peut s'agir d'une fonction sur E; ou d'une fonction sur l'ensemble des nœuds (numérotés de 1 à cn=cari-1). En commentant le contenu de tous ces tableaux, nous avons l'occasion d'anticiper sur le déroulement de l'algorithme.

De façon précise, dans {An, Bn, Tn} sont les informations, relatives aux nœuds de la CAH, lues sur 'cnabd'. Ces informations sont conservées telles quelles (à ceci près que si, par une erreur que nous croyons exclue: cf. §2.4, il y avait une inversion dans la suite des niveaux des nœuds, celle-ci serait

corrigée: cf. *infra*, §3.3.1, *in fine*). Dans Carn^[n], est le cardinal de la classe des individus de E descendant du nœud n. Dans nin^[n], on trouve le niveau, Tn^[n], remis à l'échelle, en fonction du nombre de caractères choisi comme largeur pour un tracé d'arbre.

L'entier PER^[é] est le numéro (de 1 à cn) du nœud, ci-dessus appelé: père, dont é (élément de Eé) est l'aîné ou le benjamin (; ce numéro n'est donc pas défini pour le sommet: é=cari+cn). On a dans nlic^[é] le numéro de la ligne où doit se placer é dans le tracé de l'arbre de la CAH générale (arbre de E tout entier; et non: arbre d'une partition, comme dans l'exemple du §3.1). Il faut comprendre, en particulier, que sur la 1-ère ligne se trouve (à droite) le sommet (é=(2.cari)+1) ainsi que son aîné, l'aîné de son aîné; etc...; jusqu'à un élément de E dont le sigle est à l'extrémité gauche de la 1-ère ligne. Réciproquement, l'arbre s'écrivant sur cari lignes, il y a, à l'extrémité gauche de toute ligne, i, le sigle d'un élément i de E, dont le numéro est: ilil^[i].

À la différence des précédents tableaux, ren dépend du choix de la partition à représenter: les nœuds qui figurent comme tels (et non, éventuellement, comme classes terminales) dans l'arbre de cette partition sont marqués: ren^[n]=1; pour les autres nœuds: ren^[n]=0.

Pour expliquer le contenu du tableau qdc, il faut, également, présupposer le choix d'une partition. Soit: carq, le nombre des classes de celle-ci. Sur l'arbre de la CAH générale, les individus de chaque classe constituent un bloc de lignes consécutives: il est naturel d'attribuer à toute classe le rang, de 1 à carq, du bloc qui lui correspond.

Cependant, dans Eé, cette classe (qui peut être réduite à un élément unique de E; ou être formée des descendants d'un nœud) a déjà un numéro; lequel est mis dans clq[q]; case d'un tableau déclaré en tête de 'arbt', et dont la dimension est restreinte à: qmax=100, maximum arbitrairement imposé au nombre carq des classes de la partition demandée par l'utilisateur.

Considérons maintenant l'arbre de la partition retenue, en carq classes. Les terminaux, qui ne sont autres que ces classes, sont déjà numérotées de 1 à carq (en plus de leur numéro clq[q], dans Eé). Les nœuds, peuvent être numérotés de m=carq+1 à m=(2.carq)-1. Ce numérotage, noté dans un tableau, cdm, analogue à clq, est choisi pour s'accorder avec celui de la CAH générale. De façon précise, si cdm[m] est, dans cette CAH, le numéro du nœud m de la hiérarchie restreinte, cdm est une fonction croissante de m.

On peut maintenant dire, en bref, que qdc est la fonction inverse (non partout définie sur Eé) de clq et de cdm: si c=clq[q], on pose: qdc^[c]:=q; si c=cdm[m], on pose: qdc^[c]:=m. Ces valeurs suffisent pours les appels que fait 'arbt' au tableau qdc^ (cf. *infra*, §3.3.3).

3.2.2 Dialogue du choix de la partition

```
while not (rec='O') do begin rhs:='*';
 while not ((rhs='H')or(rhs='S')) do begin
  writeln('la partition à écrire est-elle définie par');
  write('les noeuds les plus hauts(H) ou par des noeuds spécifiés(S) ');
  readln(rhs) end;
 if (rhs='H') then begin rec:='N';
  while not (rec='O') do begin
   write('le nombre de classes (de 2 à 100) de la partition à écrire est ');
   readln(carq);
   if (100<carq) then carq:=100;if (cari<carq) then carq:=cari;
   if (carq<2) then carq:=2;
   writeln('le nombre des classes sera ',carq:3);
   write('ce nombre est-il confirmé O ou N ');readln(rec);end;
  for n:=(1+cari)-carq to cn do ren^[n]:=1 end;
 if (rhs='S') then begin rec:='N';
  while not(rec='O') do begin
   write('le nombre des noeuds spécifiés (de 1 à 50) est ');readln(j);
   if (50<j) then j:=50; if (j<1) then j:=1;
   writeln('le nombre des noeuds à spécifier est',j:3);
   write('ce nombre est-il confirmé oui(O) ou non(N) ');readln(rec);end;
  for n:=1 to j do begin
   write('le noeud spécifié en place',n:3,' a pour no de classe c = ');
   readln(q);q:=q-cari;if (q<1)or(cn<q) then q:=cn;ren^[q]:=1 end end;
 write('le choix de la partition est-il confirmé oui(O) ou non(N) ');
 readln(rec) end;
```

On distingue deux éventualités: rhs='H', et: rhs='S', selon que la partition est définie par les noeuds les plus hauts (H) ou par des noeuds spécifiés (S). Dans le premier cas, l'utilisateur n'a qu'à donner le nombre des classes, carq, de la partition souhaitée. Le programme, après avoir limité ce nombre à 100, écrit immédiatement, dans le tableau ren^ (cf. *supra*, §3.2.1) la valeur 1 pour les carq–1 nœuds les plus hauts, seuls destinés à figurer, comme, tels dans l'arbre de cette partition.

Le cas rhs='S' est plus complexe. L'utilisateur, considérant un premier listage de la CAH, voit des classes interprétables qu'il veut conserver dans la partition définitive. Au contraire les subdivisions de certaines branches lui paraissent dépourvues de sens. Pour obtenir qu'une classe figure dans la partition définitive, il suffit de spécifier le nœud qui en est le père.

Mais si l'on spécifie des nœuds au nombre de j, seront, *ipso facto*, retenus tous ceux qui en sont les ancêtres dans la hiérarchie en sorte que, finalement, le nombre, carq, des classes de la partition retenue dépasse j+1; et il peut même dépasser 100. En ce cas, ultérieurement, une fois acquise la structure de la hiérarchie générale, lue dans 'cnabd' (cf., *infra*: §3.3.2), le programme, par la boucle:

```
for n:=cn downto 1 do if (q<qmax) then q:=q+ren^[n] else ren^[n]:=0;
```

élimine, comme surnuméraires, les (carq-99) nœuds les plus bas.

3.2.3 Formats pour le listage de la partition retenue

```
rec:='N';
while not (rec='O') do begin
  write('la largeur (en caractères) choisie pour le listage est ');
  readln(larg);if (lmax<larg) then larg:=lmax;if (larg<76) then larg:=76;
  write('la largeur (en caractères) choisie pour tracer l''arbre est  ');
  readln(long);if (255<long) then long:=255;if (long<40) then long:=40;
  if (larg<long) then
    writeln('ATTENTION: l''arbre sera tronqué à droite');
  write('faut-il passer une ligne entre 2 classes oui(O) ou non(N) ');
  readln(req);
  write('le choix de l''arbre de la partition est-il confirmé O ou N ');
  readln(rec) end;
```

L'utilisateur fixe d'abord, comme un nombre de caractères, larg, la largeur du listage de la CAH; puis la largeur, long, choisie pour tracer l'arbre de la partition retenue. Nous retrouvons ici des notions déjà introduites à propos de l'exemple du §3.1. Si long excède larg, le programme accepte ce choix mais avertit le lecteur de ce que l'arbre sera tronqué à droite. Cette éventualité est utile lorsque le niveau du sommet de la hiérarchie dépasse de beaucoup celui de ses descendants immédiats: en augmentant l'échelle horizontale, on peut conserver, sur le graphique, le détail des subdivisions inférieures; tandis que la perte d'une longue fourche à droite importe peu. D'autre part: passer une ligne entre 2 classes, permet d'introduire aisément, dans le tracé l'arbre, des commentaires (ou aides à l'interpration: cf. IIC, IID).

3.2.4 Lecture des données afférentes à la CAH et à l'ensemble E

```
for n:=1 to cn do read(fin,An^[n],Bn^[n],Tn^[n]) ;
close(fin);An^[cari]:=0;Bn^[cari]:=0;
nomf:=concat(nomen,'Fac.w');reset(firi,nomf);
with riri do begin
 for u:=1 to titab do read(firi,n);read(firi,n);amax:=n-2;
 dir:=7+(2*amax);for u:=1 to 3*(amax+2) do read(firi,n);
 for i := 1 to cari do begin
      for u:=1 to dir do read(firi,ptrcl^[u]);
      pnmi^[i]:=sis;write(sigler(sis):5);
      if (i mod 15=0) then writeln end end;
close(firi);writeln;
```

Le fichier 'cnabd' est ouvert; on y a lu seulement 3 nombres; dont celui, cari, des éléments de l'ensemble E. Le programme lit maintenant la description de la CAH, suivant les triplets: {An^[n], Bn^[n], Tn^[n]}.

La valeur 0 est attribuée à An^[cari] et Bn^[cari], en vue de la condition c=An^[n], ou c=Bn^[n] qui est dans 'traceli'. La ligne est composée à partir de la base; on doit s'arrêter en arrivant au sommet; pour lequel 'traceli' pose n=cari; instruction que nous avions omis de commenter ci-dessus.

De plus, sur le fichier des facteurs, 'Fac;w', sont lus les sigles des éléments de E. En ffet, ces sigles doivent figurer dans le tableau du contenu des classes de la partition retenue; ainsi qu'à l'extrémité gauche des lignes de l'arbre de la CAH générale. Au fur et à mesure de la lecture, les sigles sont affichés à l'écran.

3.3 Calculs effectués dans la structure de la CAH générale

3.3.1 Calcul des tableaux: PER^, nlic^ et ilil^, afférents à la CAH générale

```
for n :=1 to cn do begin PER^[An^[n]]:=n;PER^[Bn^[n]]:=n;
   Carn^[n]:=carc(An^[n])+carc(Bn^[n]) end;
nlic^[(2*cari)-1]:=1;
for n:=cn downto 1 do begin
   nlic^[An^[n]]:=nlic^[n+cari];nlic^[Bn^[n]]:=nlic^[n+cari]+carc(An^[n]) en
for n:=1 to cari-2 do
  if (Tn^[n]>Tn^[PER^[n+cari]]) then Tn^[PER^[n+cari]]:=Tn^[n];
for i:=1 to cari do ilil^[nlic^[i]]:=i;
```

Ces tableaux ont déjà été définis, au §3.2.1, à propos de l'allocation de zones de mémoire centrale aux pointeurs. Ainsi, le tableau PER donne, pour tout élément é de Eé, le numéro PER^[é] du nœud qui en est le père. On notera que Carn^[n], cardinal de la classe des individus de E descendants du nœud n, se calcule en appelant la fonction 'carc'; laquelle, si son argument est ≤cari, vaut 1; et qui, pour un nœud, est une valeur déjà calculée de Carn^.

L'entier: nlic^[é], est le numéro de la ligne où doit se placer é dans le tracé de l'arbre de la CAH générale. Ce numéro est 1 pour le sommet: é=(2.cari)-1. Il est, pour l'aîné: An^[n], le même que pour le père, nœud n, considéré comme l'élément de rang cari+n de Eé. Mais pour le benjamin: Bn^[n], il faut descendre d'autant de lignes que la classe An^[n] compte d'éléments. Le tableau ilil^ est comme la fonction inverse de nlic^ (restreint aux seuls éléments de E).

Enfin, est corrigée (par récurrence à partir du bas de la hiérarchie) toute inversion dans la suite des indices de niveaux. Éventualité que nous croyons exlue après 'cah', mais qui pourrait se rencontrer dans une CAH réalisée autrement.

3.3.2 Ensemble des nœuds au-dessus de la partition retenue: le tableau ren

```
ren^[cn]:=1;
for n:=cn-2 downto 1 do if (ren^[n]=1) then begin np:=n;
   while ((np<cn-1) and (ren^[PER^[np+cari]]=0)) do begin
      np:=PER^[np+cari];ren^[np]:=1 end end;       q:=1;
for n:=cn downto 1 do if (q<qmax) then q:=q+ren^[n] else ren^[n]:=0;
carq:=q;i:=1;
for q:=1 to carq do begin c:=ilil^[i];
   while (ren^[PER^[c]]=0) do c:=PER^[c]+cari;clq[q]:=c;i:=i+carc(c) end;
```

Si la partition est définie par les nœuds les plus hauts, le tableau ren^ enregistre, d'emblée, les nœuds à retenir. Mais on sait que la donnée de nœuds spécifiés implique que soient retenus d'autres nœuds. Le sommet (n=cn) est toujours retenu dans la hiérarchie restreinte. Pour tout nœud retenu, il faut considérer son père et ses ascendants successifs np, afin de les marquer par 1.

Toutefois, si l'on parcourt en descendant la suite des nœuds spécifiés, on est assuré que, d'un nœud dont le père est retenu, tous les ancêtres le sont, *ipso facto*, ce qui abrège la recherche itérative. Le tableau ren étant rempli, il reste à s'assurer que le nombre des nœuds retenus n'excède pas qmax=100. À cette fin, on calcule le total: q, des nœuds retenus qu'on rencontre en descendant la suite des nœuds; et dès que q=qmax, les nœuds marqués que l'in rencontre sont mis à zéro dans ren.

On peut alors calculer, pour toute classe q de la partition retenue, son numéro: clq[q], dans Eé. À cette fin, on se réfère à l'arbre de la CAH générale. L'élément ilil^[1], dont le sigle est à gauche de la première ligne de cet arbre, appartient à la 1-ère classe, q=1 de la partition retenue; et cette classe, en tant que nœud, n'est autre que l'aîné du premier élément retenu qu'on rencontre en remontant la hiérarchie à partir de: c=ilil^[1]. Une fois connu c=clq[1], on a le numéro, 1+carc(c), de la première ligne du bloc que constitue, dans la CAH générale, la classe q=2: on procède donc, à partir de ilil^[1+carc(c)] comme on l'a déjà fait à partir de ilil^[1]; etc.

3.3.3 Création de fichiers décrivant la hiérarchie restreinte associée à la partition retenue

```
nomf:=concat(nomen,'qcqabc');rewrite(fin,nomf);write(fin,10000+carq);
for q:=1 to carq do begin c:=clq[q];cA:=0;cB:=0;qdc^[c]:=q;
   if (cari<c) then begin cA:=An^[c-cari];cB:=Bn^[c-cari] end;
   write(fin,cA,cB,c) end;close(fin);
m:=carq;n:=cari;
while (1<m) do begin n:=n-1;if (ren^[n]=1) then begin
   m:=m-1;cdm[m]:=n+cari;qdc^[n+cari]:=m+carq end end;
nomf:=concat(nomen,'qcmabc');rewrite(fin,nomf);write(fin,9999+carq);
for m:=carq-1 downto 1 do begin c:=cdm[m];
  cA:=qdc^[An^[c-cari]];cB:=qdc^[Bn^[c-cari]];write(fin,cA,cB,c) end;
close(fin);
```

Le fichier 'qcqabc' contient, après la valeur carq+10000, une suite de carq triplets {cA, cB, c}, afférents aux classes la partition retenue: c, est le numéro de la classe dans Eé; et, s'il s'agit d'un nœud, cA et cB sont, dans Eé, les numéros de son aîné et de son benjamin; sinon: cA=cB=0.

On a dit que cdm[m] est, dans Eé, le numéro du nœud m de la hiérarchie restreinte; et que si c=cdm[m], on pose: qdc^[c]:=m. Ceci fait, est créé le fichier 'qcmabc' qui décrit l'arborescence restreinte (afférente à la partition retenue). Après la valeur 10000+carq-1, on a une suite de triplets {cA, cB, c}, pour les carq-1 nœuds (rangés de haut en bas): c, numéro du nœud dans Eé; cA et cB, numéros de ses deux descendants, considérés comme éléments de la hiérarchie restreinte (numéros de 1 à (2.carq)-1).

Les fichiers: 'qcqabc' et 'qcmabc', serviront aux programmes d'aide à l'interprétation, FACOR et VACOR.

3.4 Le listage de la CAH

3.4.1 En-tête du listage et tableau de la hiérarchie

```
nomf:=concat(nomen,'arbx');rewrite(ft,nomf);
writeln(ft,nomf); if not(cht='N') then begin
   writeln(ft,'   les inerties et indices de niveau sont calculés dans');
   write(ft,'l''espace engendré par les');if (cht='F') then write(ft,carf:3);
   writeln(ft,' axes utilisés pour la CAH');
   writeln(ft,'   la somme des indices de niveau est',ir:6,' e ',ex:3) end;
writeln(ft,'   NB : les taux T sont comptés en e-4');
vin:=(larg-5) div 5;
w:=1+((cn-1) div vin); ng:=cn+vin;
if(100 div vin<w) then w:=1+(100 div vin);
for li :=1 to w do begin
  ng:=ng-vin;np:=1+ng-vin;if(np<1) then np:=1;writeln(ft);
  write(ft,'c  ');for n:=ng downto np do write(ft,n+cn+1:5);writeln(ft);
  write(ft,'car');for n:=ng downto np do write(ft,Carn^[n]:5);writeln(ft);
  write(ft,'T  ');for n:=ng downto np do write(ft,Tn^[n]:5);writeln(ft);
  write(ft,'A  ');for n:=ng downto np do if (cari<An^[n]) then
     write(ft,An^[n]:5) else write(ft,sigler(pnmi^[An^[n]]):5);writeln(ft);
  write(ft,'B  ');for n:=ng downto np do if (cari<Bn^[n]) then
     write(ft,Bn^[n]:5)
    else write(ft,sigler(pnmi^[Bn^[n]]):5);writeln(ft) end;
writeln(ft);
```

Ainsi qu'on l'a dit au début du §3.2, avec la procédure 'cah' publiée ici, le caractère 'cht' n'est autre que la lettre 'F': le nombre des axes utilisés et l'inertie totale (ou somme des niveaux des nœuds de la hiérarchie complète) sont donc effectivement écrits en tête du listage.

Vient ensuite un tableau, analogue à celui qu'a affiché à l'écran le procédure 'cah' (cf. §2.5); mais composé par blocs de 5 lignes (et non de 4); avec autant de blocs qu'il en faut pour écrire les informations afférentes à tous les nœuds; ou, au moins, pour écrire 100 nœuds. Sont donnés: sur la ligne 1 du bloc, les numéro succesifs (dans Eé), c, de classes (prises en descendant à partir du sommet); puis, sur les lignes suivantes, avec une colonne par classe: le cardinal, le taux, le numéro (dans Eé) de l' aîné et du benjamin.

3.4.2 Tableau du contenu des classes de la partition retenue

En même temps que le tableau du contenu des classes est écrit sur le listage (i.e., sur le fichier de texte 'arbx'), on crée un fichier numérique, de contenu équivalent, avec le suffixe: 'cqki'.

En tête du fichier 'cqki', figure le nombre des classes de la partition retenue. Puis, pour chacune des classes (prises dans l'ordre de: q= 1 à carq), on a son numéro, dans Eé, augmenté de 20000; son cardinal, augmenté de 10000; et la liste des numéros, dans E, des éléments qui constituent la classe.

Sur le listage, le tableau décrit successivement les classes (de q=1 à carq) en donnant, en marge gauche, le numéro de la classes dans Eé; puis, les sigles des individus qui constituent la classe; sur autant de lignes qu'il le faut,

```
den:=0;
for q:=1 to carq do if (den<carc(clq[q])) then den:=carc(clq[q]);
den:=5*(den+1);if (larg<den) then den:=larg;
if (den<76) then den:=76;
for ve:=1 to den do write(ft,'=');writeln(ft);
nomf:=concat(nomen,'cqki');rewrite(fin,nomf);write(fin,carq);
write(ft,'  c | Partition en',carq:4,' classes : ');
writeln(ft,'Sigles des individus de la classe numéro c');
for q:=1 to carq do begin
    for ve:=1 to den do write(ft,'-');writeln(ft);
    c:=clq[q];ve:=nlic^[c]-1;w:=1+((carc(c)-1) div vin);ng:=0;
    j:=20000+c;write(fin,j);j:=10000+carc(c);write(fin,j);
    for li:=1 to w do begin
        if (li=1) then write(ft,c:4,'|') else write(ft,'    |');
        np:=ng+1;ng:=np+vin-1;if (carc(c)<ng) then ng:=carc(c);
        for i:=np to ng do begin
            write(ft,sigler(pnmi^[ilil^[ve+i]]):5);
            write(fin,ilil^[ve+i]);end;
        writeln(ft) end end;
close(fin);
for ve:=1 to den do write(ft,'=');writeln(ft);writeln(ft);
```

compte tenu de la largeur demandée pour le listage.

3.4.3 Tracé de l'arbre de la partition retenue

```
res:=(long-6)/Tn^[cari-1];
for n:=1 to cn do nin^[n]:=round(res*Tn^[n]);
for ve:=1 to long do vert[ve]:=0;noir:=long;q:=0;
for q:=1 to carq do begin
     c:=clq[q];writeln(c:5);
     write(ft,c:4,' ');traceli;
     writeln(ft,lin1);if (req='O') then writeln(ft,lin2) end;
writeln(ft);
```

Compte tenu de la place prise par le numéro de la classe, écrit en marge gauche, la largeur de l'arbre proprement dit est: long-6. [Plus précisément, (long-6) est la longueur du trait horizontal écrit sur la 1-ère ligne; non compris le point par lequel celle-ci se termine; et à partir duquel descend une verticale: cf, *supra*, §3.1].

Ainsi qu'on l'a dit au §3.2.1, on a dans: nin^[n], le niveau, Tn^[n], remis à l'échelle en fonction de cette largeur: (long-6).

La mise à zéro du tableau vert a déjà été expliquée au §3.1: avant le premier appel de 'traceli', il n'y a aucune ligne verticale en cours de tracé. L'instruction: noir:=long, donne à la 1-ère ligne, qui va jusqu'au sommet, toute la largeur fixée pour l'arbre.

Le nombre des lignes consécutives qu'occupe le graphique de l'arbre n'est autre que le nombre: carq, des classes de la partition retenue. La procédure 'traceli' est appelée pour chacune de ces classes, q, après que le numéro de celle-ci: clq[q], a été écrit sur le listage. Le trait 'lin1', composé par 'traceli', est ensuite écrit pour terminer la ligne. Si l'utilisateur l'a demandé (req='O'), on place, après lin1, une ligne lin2, ne contenant que des barres: '|'.

3.4.4 Dialogue pour le tracé de l'arbre de la CAH générale

```
writeln(ft,'ci-dessus l''arbre de la partition en',carq:4,' classes');
write('faut-il tracer l'' arbre de la CAH générale oui(O) ou non(N) ');
readln(req);
if (req='O')then begin rec:='N';
   while not (rec='O') do begin
    writeln('on donnera ci-après les dimensions de l''arbre de la CAH');
    write('la largeur (en caractères) choisie pour tracer l''arbre est ');
    readln(long);if (255<long) then long:=255;
    if (long<40) then long:=40;
    if (larg<long) then
      writeln('ATTENTION: l''arbre sera tronqué à droite');
    write('faut il passer une ligne entre 2 individus oui(O) ou non(N) ');
    readln(req);
    write('le choix du tracé de l''arbre de la CAH est-il confirmé O ou N ')
    readln(rec) end;
```

On choisit les formats de l'arbre de la CAH générale comme ceux de l'arbre afférent à la partition retenue (cf. *supra*, §3.2.3)

3.4.5 Tracé de l'arbre de la CAH générale

```
writeln(ft,'ci-dessous l''arbre de la CAH générale');
res:=(long-6)/Tn^[cari-1];
for n:=1 to cari-1 do nin^[n]:=round(res*Tn^[n]);
for ve:=1 to long do  vert[ve]:=0;noir:=long;writeln(ft);
for i:=1 to cari do  begin
   c:=ilil^[i];write(ft,sigler(pnmi^[c]):4, ' ');
    traceli;writeln(ft,lin1);
   if (req='O') then writeln(ft,lin2) end end;
```

Il n'y a, avec le tracé de l'arbre de la partition, qu'une seule différence: en marge, à gauche de la ligne afférente à un inividu, est écrit le sigle de celui-ci.

La procédure 'traceli' se termine en libérant les zones affectées, en mémoire centrale, à divers tableaux (cf supra, §3.2.1).

```
close(ft);dispose(pnmi);dispose(PER);dispose(nlic);dispose(qdc);
dispose(An);dispose(Bn);dispose(Tn);dispose(Carn);
dispose(nin);dispose(ilil);dispose(ren);
end;end.
```

Références bibliographiques

J.-P. & F. BENZÉCRI : "Programmes d'analyse des correspondances et de classification ascendante hiérarchique: notice d'utilisation"; [NOT. CORR. CAH]; in *CAD*, Vol.XIV, n°1; pp. 7-34; (1989).

J.-P. BENZÉCRI : "Construction d'une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques"; [CAH CHAÎNE RÉCIP.]; in *CAD*, Vol.VII, n°2; pp. 209-218; (1982). Et, dans le présent cahier: [RAPPEL CAH CHAÎNE RÉCIP.].