*Les Cahiers de l'Analyse des Données*
*Vol. XXII - 1997 - n°1 - pp. 39-54.*

# SOURCES DE PROGRAMMES D'ANALYSE DE DONNÉES EN LANGAGE PASCAL: (I) : ANALYSE DES CORRESPONDANCES (IB) : LE PROGRAMME D'ANALYSE

## [SOURCES PASCAL (IB)]

*J.-P. & F. BENZÉCRI*

### 0 Introduction : structure et déroulement du programme 'qori'

```
begin benzecri;litaber;unloadseg(@litaber);
if (repc='O') then begin
  tabuler;carrer;unloadseg(@tabuler);
  diagu(@Phaj,@fjj,@dj,@zz,@pza,@rj,@lama,carj);
  renorm;unloadseg(@diagu);
  sortir;
  dispose(fi);dispose(sigli);dispose(siglt);dispose(slu);dispose(prla);
  for j:=0 to cartj do dispose(klji[j]);dispose(Fpa);
  for j:=1 to carj do dispose(fjj[j]);dispose(spgj);
  unloadseg(@sortir); readln(repa) end;end.
```

Avant d'expliquer le programme 'qori' en suivant le listage, lequel, après des déclarations, donne une hiérarchie complexe de procédures, réparties en segments, il vaut la peine, en guise d'introduction, de considérer le programme principal, qui est très bref.

D'abord, par la procédure 'litaber', l'utilisateur choisit le tableau à traiter ; et, sous réserve que le choix se fixe (par la réponse de confirmation repc='O') sur un tableau accessible sur un disque, ce tableau est mis en mémoire centrale, où est également réservé l'espace nécessaire pour choisir les éléments supplémentaires.

Ce dernier choix rentre dans la procédure 'tabuler', dont la fonction principale est, en bref, de disposer le tableau $I \times J$ suivant l'ordre $(Ip \cup Is) \times (Jp \cup Js)$, les éléments principaux et supplémentaires formant deux blocs consécutifs. La procédure 'carrer' calcule alors le tableau à diagonaliser par 'diagu'.

Du point de vue du calcul numérique, la procédure 'diagu' fait l'essentiel de toute analyse factorielle.

Il ne reste plus qu'à créer les fichiers de facteurs, gardés sur disque, sous forme binaire ou en texte: tel est l'objet de la procédure 'sortir'.

Les cahiers de l'analyse des données 0339-3097/97/01./$ 7.00/© Gauthier-Villars

```
program qori;uses memtypes,quickdraw,osintf,toolintf,sane,uver,utrir;
 var
  ve,w,w1,w2,d,memi,nivc,nivn,reste,long,amax,dir,i1,er1,u,iz1,
  al,s,smin,sp,cars,carm,carq,n,c,ia,ic,id,ip,is,ih,ijh,
  cn,na,nb,nz,la,lb,ina,li,q,r,ng,np,i,j,j1,j2,jj,ij,jp,jh,js,a,ap,b1,b,
  carti,cari,caris,cartj,carj,carjs,carf,care,carel,cares,carte:integer;
  ff,ff1,PDS,INR,QLT,CO2,CTR:longint;
  fin:file of integer;ffs:single;p1ww:pww;
  ft:text;
  norm,pod,pop,rdf,mtot,tr,lam,di,fijij,trace,dr,d2,fre,res: extended;
  k1ji,g1ji:kji;gi:pti;t1k:tk;pli,p2j:ptr;
  fjj,Phaj:sjj;
  carac,repa,resu,rep,reps,repg,repc,che,rup:char;
  titre,nomba,nomf:string;ttri:stringptr;
  lmt,adj:string[7];sgli:pzgl;
  spgj,sigli,sig1t:pzgi;sql,sqe:zigle;
  riri:ricla;ptrc1:ptrc;firi:file of integer;
  fi:ptdri; phj,dj,fj,rj,ffj,lama:rtj;
  tampi:array[1..titab]of integer;
  pza,zz:itj;
  Fpa:prtj;
  prla,s1u:ptni;
```

Le présent exposé renvoie fréquemment à l'article [SOURCES PASCAL (IA)], dont les § seront simplement cités IA§xx. Dans IA§1, sont les types particuliers figurant dans les déclarations du programme 'qori' et de ses procédures. Dans IA§2, est décrite la procédure 'litab', appelée, dans 'qori', par la procédure 'litaber', dont elle constitue l'essentiel. Dans IA§3, est décrite la procédure 'ensembler', appelée, dans 'qori', par 'tabuler', pour le choix des éléments supplémentaires. Dans IA§5 est schématiquement présentée la procédure de diagonalisation 'diagu' (laquelle appelle 'trire', procédure de tri décrite dans IA§4).

Le listage du programme source, 'qori.p', commence par des déclarations de variables; où figurent nombre des types décrits dans IA§1. Nous ne donnerons pas, de cette longue liste, une explication préalable, nous bornant à y renvoyer à propos des procédures.

Avant le programme principal déjà présenté, les procédures appelées par celui-ci sont rangées en 4 segments (se succédant dans un ordre qui pourrait être modifié sans que le déroulement du programme s'en ressentît).

Le segment {$S sym}, qui appelle l'unité {$U udiaq} de diagonalisation, fait l'objet du §1.

Le segment {$S exit}, où, pour le tableau de données choisi, sont créés sur disque les fichiers de facteurs, à partir des vecteurs propres calculés dans {$S sym}, fera l'objet du §2.

Le segment {$S init}, qui appelle l'unité {$U ulire}, et le segment {$S tabl} qui applelle l'unité {$U uens}, seront considérés ensemble dans le §3; car, ensemble, ils créent, en mémoire centrale, le tableau de données mis en ordre (par blocs) et la matrice carrée soumise à 'diagu'.

### 1 Calcul des facteurs normalisés: le segment {$S sym}

```
{$S sym} {$U udiaq}
procedure diagu(phaj0,fjj0,dj0,zz0,pza0,rj0,lama0:ptr;
     carj:integer);external;
procedure renorm;begin
w:=aamax;if (carj-1<w) then w:=carj-1;
for a:=1 to w do for j:=1 to carj do Phaj[a]^[j]:=ffj[j]*Phaj[a]^[j];end;
```

Nous ne reprendrons pas l'explication minutieuse, donnée dans IA§5, des variables de la procédure 'diagu' et de leurs valeurs lors de l'appel qu'en fait 'qori', et après l'exécution de 'diagu'.

Il suffit de rappeler que la procédure 'diagu' reçoit, par le pointeur fjj0, la matrice carrée qu'elle traite; et elle renvoie à 'qori', par les pointeurs {Phaj0, lama0}, les résultats obtenus: vecteurs propres et valeurs propres. Tout autre effet de l'exécution de 'diagu' est sans conséquence pour la suite de 'qori': i.e., pour la création des fichiers de résultats, objet du §2 ci-après.

Plus précisément, à partir des valeurs Phaj[a]^[j] renvoyées par 'diagu', sont calculés, par 'renorm', les facteurs normalisés notés communément $\varphi_\alpha^j$.

En effet, notons $\pi_\alpha(j)$ les composantes des vecteurs propres normalisés de la matrice carrée symétrique, fjj(Jp × Jp):

$$fjj(j,j') = \Sigma\{k(i,j).k(i,j')/(k(i).\sqrt{(k(j).k(j'))} \mid i \in Ip\} ;$$

soumise à diagu (cf. *infra* §3.3). À partir des $\pi_\alpha(j)$, qui satisfont à la relation:

$$\forall\, \alpha : \Sigma\{(\pi_\alpha(j))^2 \mid j \in Jp\} = 1 \quad ,$$

on obtient les $\varphi_\alpha^j$ par la transformation:

$$\varphi_\alpha^j = \pi_\alpha(j) \,.\, (1 / \sqrt{(f_j)}) \ ;$$

d'où résulte la condition de normalisation propre aux facteurs $\varphi_\alpha$:

$$\forall\, \alpha : \Sigma\{(\varphi_\alpha^j)^2 \,.\, f_j \mid j \in Jp\} = \Sigma\{(\pi_\alpha(j))^2 \mid j \in Jp\} = 1 \quad ,$$

Or, ainsi qu'il est précisé au §3.3, la procédure 'carrer', appelée avant 'diagu' (cf. *supra*, §0: le programme principal), laisse dans ffj les inverses des racines carrées des masses fj des éléments principaux (masses dont la somme sur Jp est 1):

$$\forall\, j \in Jp : \ ffj[j] = 1 / \sqrt{(f_j)} \quad ; \quad \Sigma\{ f_j \mid j \in Jp\} = 1 \ ;$$

On a donc bien, après exécution de la procédure 'renorm':

$$\forall\, \alpha\,, \forall\, j \in Jp : \ Phaj[\alpha]^[j] = \varphi_\alpha^j \quad ;$$

Pour être complet, il resterait à dire comment, dans le calcul de la matrice fjj et sa diagonalisation, a été éliminé le facteur trivial constant et égal à 1, relatif à la valeur propre 1: cela est précisé au §3.3.

## 2 Création des fichiers de facteurs: le segment {$S exit}

Dans le programme principal, est appelée la procédure 'sortir'; celle-ci s'exécute en appelant diverses procédures qu'il faut considérer avant 'sortir'.

### 2.1 Les procédures propres à la procédure 'sortir'

#### 2.1.1 Calcul des facteurs pour une ligne: la procédure 'preligni'

```
{$S exit}
procedure preligni;begin
     res:=fre*mtot;if (res=0) then res:=1;d2:=-1;
     for j:=1 to carj do begin
         ffj[j]:=ffj[j]/res;
         if not(fj[j]=0) then d2:=d2 +(sqr(ffj[j])/fj[j]) end;
     if (d2<0) then d2:=0;
     for a:=1 to carf do begin Fpa^[a]:=0;
         for j:=1 to carj do Fpa^[a]:=Fpa^[a]+(ffj[j]*Phaj[a]^[j]) end;end;
```

Le nom de la procédure 'preligni' en évoque l'effet: préparer la ligne afférente à un individu i (principal ou supplémentaire), avant de l'écrire sur fichier de texte (listage) et fichier binaire. Avant tout appel de 'preligni', (cf. *infra* §2.2.3), on a dans 'mtot' le total du bloc principal, (Ip × Jp), du tableau des données, k1ji; on a dans 'fre' la fréquence, fi^[i], d'un individu i; et, par l'instruction composée:

for j:=1 to carj do ffj[j]:=k1ji[j]^[i] ;

on met dans les ffj[j] les carj valeurs de la ligne i, restreinte aux colonnes principales. Ces valeurs sont converties en un profil sur Jp (suite de ffj[j] ayant pour total 1). Simultanément, est calculé le carré, $d^2$, de la distance distributionnelle du profil de i au profil moyen du nuage. De façon précise, on part de la formule classique:

$$d^2(f_J^i, f_J) = \sum\{(f_j^i - f_j)^2 (1/f_j) \mid j \in Jp\} = (-1) + \sum\{(f_j^i)^2 (1/f_j) \mid j \in Jp\};$$

On calcule ensuite les facteurs pour i par la formule de transition:

$$F_\alpha(i) = \sum\{f_j^i \cdot \varphi_\alpha^j) \mid j \in Jp\}.$$

#### 2.1.2 Écriture, sur le listage, de la ligne afférente à un élément, i ou j, principal ou supplémentaire: procédure 'lignage'

Lors de l'appel de 'lignage', il y a, dans 'fre' la fréquence, $f_i$ ou $f_j$, de l'élément, i ou j, considéré (fréquence calculée avec, pour dénominateur, 'mtot', total du bloc principal; et pour numérateur le total des composantes k1ji[j]^[i], pris pour j dans Jp, s'il s'agit d'une ligne i; et pour i dans Ip, s'il s'agit d'une colonne j); d2 est le carré de la distance distributionnelle au centre du nuage (des éléments principaux). Et les Fpa^[a] sont les valeurs des facteurs déjà calculés pour l'élément considéré.

La qualité de la représentation, QLT (,exprimée en millièmes), est calculée avec pour dénominateur d2, et pour numérateur la somme, dr des carrés des facteurs, au nombre de 'care', écrits sur le listage.

```
procedure lignage;begin
     PDS:=round(1000*fre);INR:=round(1000*d2*fre/trace);
     if (d2=0) then  QLT:=0 else begin dr:=0;
         for a:=1 to care do dr:=dr+sqr(Fpa^[a]); QLT:=round(1000*dr/d2) end;
     write(ft,'|',sigler(sql):4,'|',QLT:4,PDS:4,INR:4,'|');
     write(sigler(sql):4);
     for a:=1 to care do begin ff:=round(1000*Fpa^[a]);
         CTR:=round(1000*fre*sqr(Fpa^[a])/lama[a]);
         if (d2=0) then CO2:=0 else CO2:=round(1000*sqr(Fpa^[a])/d2) ;
         write(ft,ff:5,CO2:4,CTR:4,chr(124));write(ff:7)  end;
     writeln(ft);writeln;
     end;
```

Sur le listage, ft, toutes les grandeurs numériques, afférentes aux i et aux j, sont en millièmes. On écrit successivement, avec pour séparateurs des barres verticales '|', le sigle, le bloc {QLT, PDS, INR}: qualité, poids, inertie; puis, pour chacun des facteurs retenus, du rang a=1 au rang a=care, le bloc {FAC, CO2, CTR}: facteur; cosinus carré de l'angle entre l'axe factoriel a et le rayon vecteur de l'individu (dans l'espace des profils sur Jp pour un élément i; ou sur Ip pour un j); et contribution CTR, de l'élément, relativement à l'inertie du nuage principal (les inerties étant calculées en projection sur l'axe a).

On rappelle que CO2 est encore appelé: contribution relative, du facteur a, à l'écart entre l'élément et l'origine; et noté COR (sigle qui suggère aussi qu'il s'agit, en un certain sens, du carré d'un coefficient de corrélation).

On notera que 'lignage' affiche à l'écran le sigle de l'élément traité et les valeurs des facteurs.

### 2.1.3 Création d'un fichier binaire de facteurs et écriture de l'en-tête: la procédure 'ecriclun'

Le programme 'qori', crée, dans tous les cas, deux fichiers de facteurs pour, les ensembles principaux Ip et Jp, avec, pour suffixes respectifs, ajoutés au nom de base, 'nomba', du tableau des données:iFac.w, et: jFac.w. De plus, si l'ensemble Is, des lignes supplémentaires, est non vide, est créé un fichier avec pour suffixe isFac.w; et, de même, pour Js, jsFac;w. Nous dirons qu'aux ensembles {Ip, Jp, Is, Js} sont affectés les sigles: {'i', 'j', 'is', 'js'}.

Avant l'appel de 'ecriclun', le sigle de l'ensemble à traiter se trouve, dans 'sqe', (considéré comme zigle, i.e., packed array[0..5]of byte, cf. IA§1.2; les deux lectures, sigle et zigle, étant liées par les procédures 'sigler' et 'zigler', inverses l'une de l'autre, cf. IA§1.3.4). Avec sqe, est créé, par concaténation, le nom du fichier de facteurs, ouvert par 'ecriclun' comme un fichier d'entiers (ou de mots de deux octets).

La lecture d'un tableau en format binaire, expliquée en IA§2.5.1, n'est autre que l'opération inverse de l'écriture considérée ici. Dans l'un et l'autre sens, intervient, comme tampon, un tableau d'entiers, 'tampi', array[1..titab]of integer; avec des pointeurs, accédant au contenu de 'tampi' sous divers fomats: non seulement comme entier, mais comme réel, ou comme chaîne de caractères.

```
procedure ecriclun;begin
nomf:=concat(nomba,sigler(sqe),'Fac.w');rewrite(firi,nomf);
ttri^:=concat(sigler(sqe),':',titre);
if (127<length(ttri^)) then ttri^:= copy(ttri^,1,127);
for a:=1 to carf do begin
     plww:=pww(@tampi[63+(2*a)]);plww^:=lama[a] end;
plww:=pww(@tampi[65+(2*carf)]);plww^:=trace;
plww:=pww(@tampi[67+(2*carf)]);plww^:=mtot;
tampi[titab]:=carel;
for u:=1 to titab do write(firi,tampi[u]);write(firi,2+carf);
for a:=1 to carf do begin
  if (a<10) then
     sgli^:=zigler(concat('axe',chr(48+a))) else
     sgli^:=zigler(concat('ax',chr(48+(a div 10)),chr(48+(a mod 10))));
  for u:=1 to 3 do write(firi,tampi[u]) end;
sgli^:=zigler('poid');for u:=1 to 3 do write(firi,tampi[u]);
sgli^:=zigler('dis2');for u:=1 to 3 do write(firi,tampi[u]);
end;
```

La procédure 'ecriclun' prépare d'abord, dans 'tampi', une suite de 256 entiers (titab=256) à écrire en tête du fichier de facteurs (nous dirons, en bref: écrire en tête de 'fin'). D'abord, sur au plus 128 octets, soit 64 entiers, le titre du tableau analysé, précédé du sigle 'sqe' de l'ensemble. Ensuite, les valeurs propres afférentes aux facteurs, de rang 1 à carf, à écrire sur 'fin'; puis la trace et la masse totale, 'mtot'. Le cardinal, 'carel' de l'ensemble d'éléments considéré (Ip, J p, Is ou Js), est dans la dernière position, tampi[titab].

Si le titre comprend moins de 127 caractères, il y a dans tampi, entre la fin du titre et le début de la suite des valeurs propres, des entiers qui n'apportent aucune information utile. De même, si le nombre, carf, des facteurs à garder est inférieur à aamx (93), il y a un vide entre les deux entiers (de rang 67+(2.carf) et 68+(2.carf)), où s'inscrit 'mtot', et tampi[titab], où est 'carel'. (Plus précisément, avec 64 entiers réservés au titre; 2×(93+2)=190 entiers, réservés aux nombres réels, lama[a], 'trace' et 'mtot', il reste toujours un entier tampi[255] sans signification).

Une fois 'tampi' ainsi rempli, 'ecriclun' l'écrit dans 'fin'. Puis, après le nombre des colonnes, care+2, les sigles attribués à celles-ci: i.e., pour les care facteurs, 'axe1', 'axe2', ..., 'axe9', 'ax10',...; puis: 'poid' et 'dis2'.

L'écriture de l'en-tête est achevée: commence celle des lignes afférentes aux éléments: tel est l'objet de la procédure 'riclage'.

### 2.1.4 Écriture des facteurs pour un élément, sur fichier binaire: la procédure 'riclage'

```
procedure riclage;begin
       with riri do begin sis := sql;
         for a := 1 to carf do Fac[a]:=Fpa^[a];
         Fac[carf+1]:=fre;Fac[carf+2]:=d2;
         for u:=1 to dir do write(firi,ptrcl^[u]);end;end;
```

Il nous suffit de dire que la procédure 'riclage' (analogue à 'lignage') écrit, sur un fichier binaire, les résultats afférents à un individu: facteurs (mais non contributions); 'poid' (fre); et distance au carré, 'dis2' (d2).

## 2.2 La procédure principale de création des fichiers de facteurs: procédure 'sortir'

### 2.2.1 Vérification du nombre de facteurs non triviaux

```
procedure sortir;begin
ptrc1:=ptrc(@riri);ttri:=stringptr(@tampi);sgli:=pzgl(@tampi);
w:=aamax;if (carj-1<w) then w:=carj-1;
for a:=w downto 1 do begin res:=0;
   for j:=1 to carj do res:=res+(Phaj[a]^[j]*fj[j]);res:=sqr(res);
   if (lama[a]<=res) then w:=a-1 end;
```

Le nombre des facteurs susceptibles d'être écrits sur 'fin' est, au plus, w=aamax=93. Reste à vérifier le nombre, w, de facteurs non triviaux dont on dispose; ou, en terme géométrique, la dimension du nuage principal: N(I) ou N(J).

On sait, *a priori*, que: w≤(carj-1), nombre des colonnes principales diminué de un. Plus précisément, dans l'analyse d'un tableau de BURT (ou d'un tableau de BURT généralisé), avec carQ variables, codée chacune suivant un bloc de modalités, le nombre total des modalités de l'ensemble des blocs étant carJ, on a : w≤(carJ-carQ).

Ainsi qu'on l'a annoncé au §1 (et comme il sera précisé au §3.3), la matrice à diagonaliser a été calculée de telle sorte que le facteur trivial, constant et égal à 1, sorte, ici, avec la valeur propre zéro. Si les calculs étaient d'une précision absolue, la dimension, w, du nuage, ne serait autre que le nombre des valeurs propres non nulles.

En fait, même si la précision effectivement atteinte ne laisse rien à désirer, on attend qu'après w valeurs propres non nulles, afférentes aux facteurs à conserver, il y ait (carj-w) valeurs propres quasi nulles, associées à des facteurs dépourvus de sens, ou pseudo-facteurs; reconnaissables à ce que s'y mêlent, en quelque proportion, le facteur trivial, constant et égal à 1, qui est, aux erreurs près, dans le même sous-espace propre: lama=0. De ce mélange, il résulte qu'alors que les facteurs significatifs, $\varphi_\alpha$, ont une moyenne nulle, les pseudo-facteurs, $\psi_\alpha$, ont une moyenne non nulle, $\mu_\alpha$.

Après transition pour calculer les $F_\alpha(i)$, cette moyenne se retrouvera telle quelle, apportant au moment d'ordre 2 un terme supplémentaire: $(\mu_\alpha)^2$=res, s'ajoutant à sa valeur normale, $\lambda_\alpha$. C'est pourquoi on a pris comme critère d'élimination: lama[a]<res; critère qu'on ne pourrait justifier qu'en connaissant la struture des erreurs faites dans SYMQR, mais dont est sûr, *a priori*, que, d'une part, il serait correct si la précision était parfaite; et que, d'autre part, il éliminera tout facteur affecté d'une perturbation supérieure à la contribution lama[a] qu'il est censé apporter à l'inertie.

Dans la pratique commune, le critère fixe correctement le nombre des facteurs significatifs: on a pu s'en assurer dans les analyses de tableaux de BURT; ou quand: (cari<carj), auquel cas on doit avoir w≤(cari-1), inégalité non prise en compte par le programme 'qori'.

```
if (w<carf) then carf:=w;
dir:=7+(2*carf);
for a:=1 to carf do for j:=carj+1 to cartj do Phaj[a]^[j]:=0;
for j:=carj+1 to cartj do begin fj[j]:=0;
   for i:=1 to cari do fj[j]:=fj[j]+k1ji[j]^[i];
   fj[j]:=fj[j]/mtot end;
carel:=cari;che:='i';sqe:=zigler('i');writeln;ecriclun;
```

Aux instructions qui suivent la vérification des facteurs, nous ne consacrerons pas de § particulier, mais seulement un bref commentaire.

D'après w, on corrige, s'il y a lieu, carf, nombre des facteurs demandés sur fichier binaire; puis on calcule 'dir', nombre de mots, de 2 octets, pour l'enregistrement d'un individu sur ce fichier (cf. 'riclage', §2.1.4): 3 mots pour le sigle; et 2 mots pour chaque quantité réelle, facteur, 'poid' ou d2.

Le tableau des facteurs sur Js est mis à zéro: il ne sera calculé qu'après celui des facteurs sur Ip; donc par double transition à partir des facteurs sur Jp. On calcule des fréquences pour les éléments de Js.

Et, par les instruction de la ligne: carel:=cari;..., on entreprend de traiter l'ensemble Ip, des lignes principales; le fichier binaire des facteurs pour Ip est ouvert par 'ecriclun' (cf. §2.1.3). Mais avant de calculer les facteurs, qui seront écrits simultanément sur fichier binaire et sur listage, il faut écrire l'en-tête du listage, par le bloc d'instructions expliqué ci-après.

### 2.2.2 Création du listage des facteurs et écriture de l'en-tête

```
nomf:=concat(nomba,'cortx'); rewrite(ft,nomf);
writeln(ft,titre);writeln(ft,'trace  :  ',trace:4);
b1:=((w-1) div 15)+1;res:=0;
for b:=1 to b1 do begin
  w1:=((b-1)*15)+1;w2:=w1+14;
  if (w<w2) then w2:=w;
  write(ft,'rang   : ');
  for a:=w1 to w2 do write(ft,a:6);writeln(ft);
  write(ft,'lambda : ');
  for a:=w1 to w2 do write(ft,round(10000*lama[a]):6);
  writeln(ft,'   e-4');write(ft,'taux   : ');
  for a:=w1 to w2 do write(ft,round(10000*lama[a]/trace):6);
  writeln(ft,'   e-4');write(ft,'cumul  : ');
  for a:=w1 to w2 do begin res:=res+lama[a];
      write(ft,round(10000*res/trace):6) end;
  writeln(ft,'   e-4') end;
if (carf<care) then care:=carf;
for a:=1 to 19+(14*care) do write(ft,'_'); writeln(ft);
write(ft,chr(124),'SIGI',chr(124),' QLT PDS INR',chr(124));
for a:=1 to care do write(ft,'  F',a:2,' CO2 CTR',chr(124)); writeln(ft);
for a:=1 to 19+(14*care) do write(ft,'_'); writeln(ft);
```

Pour le listage, le bloc d'instructions, objet du §2.2.2, joue le même rôle que joue, pour un fichier binaire de facteurs, la procédure 'ecriclun'. Mais 'ecriclun' peut être appelée quatre fois (pour: Ip, Is, Jp, Js); tandis que l'ouverture du listage ne se fait qu'une fois (tous les tableaux de facteurs y étant écrits successivement) ce qui nous a dispensé d'en faire une procédure.

On sait qu'après une ligne de titre, le listage donne la trace; puis, par blocs de quatre lignes, étiquetées {rang, lambda, taux, cumul}, les rangs, valeurs propres, taux d'inertie et cumuls (les réels étant en: e-4); avec, dans chaque bloc, 15 colonnes, jusqu'à épuisement de l'ensemble des w facteurs, (i.e. de la suite de valeurs propres qui est écrite, également, par 'ecriclun').

Puis, de même que la procédure 'ecriclun' écrit des sigles, on écrit, sur le listage, des têtes de colonnes, les sigles étant, ici, encadrés de traits: '|' ou '_'.

### 2.2.3 Calcul et écriture des résultats afférents à l'ensemble I des lignes, principales et supplémentaires

```
for i:=1 to carti do begin
     if (i=cari+1) then begin
         close(firi);carel:=carti-cari;sqe:=zigler('is');ecriclun;
         writeln(ft,'ci-dessous élément(s) supplémentaire(s)');
         writeln('ci-dessous élément(s) supplémentaire(s)') end;
     sql:=sigli^[i];fre:=fi^[i];
     for j:=1 to carj do ffj[j]:=klji[j]^[i];
     preligni;lignage;riclage;
     if (i<=cari) then
       for j:=carj+1 to cartj do if not(fj[j]=0) then for a:=1 to carf do
      Phaj[a]^[j]:=Phaj[a]^[j]+(Fpa^[a]*klji[j]^[i]/(fj[j]*mtot)) end;
 close(firi);
```

En une seule boucle, for i:=1 to carti do..., sont traités toutes les lignes: d'abord, de 1 à cari, les lignes principales; puis, s'il y a lieu, de cari+1 à carti, les lignes supplémentaires. À cette fin, si (i=cari+1), on interrompt l'écriture du listage (ainsi que l'affichage à l'écran) par une ligne annonçant les éléments supplémentaires. Et l'on ferme le fichier binaire afférent à Ip, pour créer, par 'ecriclun', celui afférent à Is.

Les facteurs, calculés par 'preligni' (cf. §2.1.1), sont écrits sur le listage et affichés à l'écran par 'lignage' (cf. §2.1.2); puis écrits sur le fichier binaire par 'riclage' (cf. §2.1.4).

Tout en traitant complètement l'ensemble I, la boucle calcule progressivement les facteurs sur Js. Plus précisément, la transition appliquée aux facteurs sur Ip, $F_\alpha$, ne donne pas directement les facteurs, $G_\alpha$, sur J, mais le produit $\sqrt{(\lambda_\alpha)}.G_\alpha$; ce que corrigera le bloc d'instruction suivant.

### 2.2.4 Écriture, sur le listage, de l'en-tête du tableau afférent à J; création du fichier binaire pour les facteurs sur Jp; normalisation des facteurs sur Jp et Js

Le calcul et l'écriture des résultats afférents à l'ensemble I des lignes est achevé. La procédure 'sortir' termine, ci-après, le traitement de l'ensemble J des colonnes.

Il n'y a rien, dans l'écriture des en-têtes, qui diffère de ce qui a été fait pour l'ensemble I.

```
for a:=1 to 19+(14*care) do write(ft,'_');writeln(ft);writeln;
for a:=1 to 19+(14*care) do write(ft,'_');writeln(ft);
write(ft,chr(124),'SIGJ',chr(124),' QLT PDS INR',chr(124));
for a:=1 to care do write(ft,'  F',a:2,' CO2 CTR',chr(124)); writeln(ft);
for a:=1 to 19+(14*care) do write(ft,'_'); writeln(ft);
for a:=1 to carf do begin di:= sqrt(lama[a]);
     for j:=1 to carj do Phaj[a]^[j]:=di*Phaj[a]^[j] ;
     if not (di=0) then di:=1/di;
     for j:=carj+1 to cartj do begin Phaj[a]^[j]:=di*Phaj[a]^[j];
       if (fj[j]=0) then Phaj[a]^[j]:=0 end end;
for j:=1 to cartj do rj[j]:=-1;
for i:=1 to cari do if not(fi^[i]=0) then for j :=1 to cartj do
  if not(fj[j]=0) then begin
    res:=k1ji[j]^[i];rj[j]:=rj[j]+(sqr(res/(mtot*fj[j]))/fi^[i]) end;
carel:=carj;che:='j';sqe:=zigler('j');ecriclun;
```

Quant à la normalisation, elle se fait différemment pour Jp et Js. Pour les éléments principaux, on a, dans Phaj[a]^[j], les valeurs, $\varphi_\alpha^j$, des facteurs nomalisés, de variance 1 sur Jp. Les facteurs $G_\alpha(jp)$, de variance $\lambda_\alpha$, sont donc calculés comme les produits $\sqrt{(\lambda_\alpha)}.\varphi_\alpha^{jp}$ . Pour les éléments supplémentaires, au contraire, on a, dans Phaj[a]^[j], les produits: $\sqrt{(\lambda_\alpha)}.G_\alpha(j)$; (cf. §2.2.3). Les facteurs $G_\alpha(js)$ sont donc calculés comme les quotients: $\varphi_\alpha^{js} / \sqrt{(\lambda_\alpha)}$.

On calcule ensuite, dans rj[j], le carré de la distance du $\chi^2$ entre le profil de j et le centre du nuage.

Il faut noter que sonts mis à zéro les résultats afférents aux j de masse nulle.

### 2.2.5 Écriture des résultats afférents à l'ensemble J des colonnes, principales et supplémentaires

```
for j:=1 to cartj do begin
    if (j=carj+1) then begin
        close(firi);carel:=cartj-carj;sqe:=zigler('js');ecriclun;
        writeln(ft,'ci dessous element(s) supplementaire(s)');
        writeln('ci dessous element(s) supplementaire(s)') end;
    if (rj[j]<0) then rj[j]:=0;d2:=rj[j];fre:=fj[j];sql:=spgj^[j];
    for a:=1 to carf do Fpa^[a]:=Phaj[a]^[j];
    lignage;riclage; end;
for a:=1 to 19+(14*care) do write(ft,'_');writeln(ft);
close(firi);close(ft);end;
```

On procède comme pour l'ensemble I des lignes. En une seule boucle, for j:=1 to cartj do..., sont traités toutes les colonnes: d'abord, de 1 à carj, les colonnes principales; puis, s'il y a lieu, de carj+1 à cartj, les colonnes supplémentaires. À cette fin, si (j=carj+1), on interrompt l'écriture par une ligne annonçant les éléments supplémentaires. Et l'on ferme le fichier binaire afférent à Jp, pour créer, par 'ecriclun', celui afférent à Js.

Les facteurs, déjà calculés et normalisés (cf. §2.2.4), sont écrits sur le listage et affichés à l'écran par 'lignage' (cf. §2.1.2); puis écrits sur le fichier binaire par 'riclage' (cf. §2.1.4).

La procédure 'sortir' s'achève par la fermeture des fichiers.

### 3 Introduction du tableau des données et calcul de la matrice à diagonaliser

Le segment {$S init}, qui appelle l'unité {$U ulire}, et le segment {$S tabl}, qui appelle l'unité {$U uens}, sont considérés ensemble dans le §3.

### 3.1 Choix du tableau des données et mise en mémoire centrale: la procédure 'litaber'

```
{$S init} {$U ulire5}
procedure litab(var pj:ptr;pk,pn,pt:ptr;var rc:char;
     var ic,jc:integer;var pi:ptr);external;
procedure litaber;begin carti:=0;
  litab(p2j,@t1k,@nomba,@titre,repc,carti,cartj,p1i);
  if (repc='O') then begin
     sig1i:=pzgi(p1i);spgj:=pzgi(p2j);new(Fpa);
     memi:=carti;if (memi<cartj) then memi:=cartj;memi:=memi+2;
     if (memi<ibmax) then sig1t:=pzgi(newptr(6*memi)) else new(sig1t);
     pr1a:=ptni(newptr(2*memi));s1u:=ptni(newptr(2*memi));
     if (memi<iamax) then fi:=ptdri(newptr(xtn*(2+carti))) else new(fi);
     for j:=0 to cartj do k1ji[j]:=pti(t1k[j]);
     write('nombre de facteurs à garder sur fichier = ');readln(carf);
     write('nombre de facteurs à écrire sur listage = ');readln(care);
     if (carf<1) then carf:=1;if (aamax<carf) then carf:=aamax;
     if (care<1) then care:=1;if (10<care) then care:=10 end;end;
```

La procédure 'litaber' commence par l'appel de 'litab' avec les pointeurs appropriés (cf. IA§2.1). Si la réponse de confirmation, repc, est: 'O', 'oui', on installe, en mémoire centrale, pour le programme principal, les accès au tableau lu par 'litab': sig1i et spgj pointent, respectivement, vers les sigles des lignes et des colonnes; les k1ji[j] pointent vers les suites de nombres (single) afférentes aux colonnes.

Les calculs de vecteurs propres et de facteurs se faisant en format 'extended', on crée, sous ce forrmat, une ligne auxiliaire, Fpa, de longueur jmax.

Le nombre 'memi' est défini comme: 2 + sup(carti, cartj). Pour les pointeurs sig1t, pr1a, s1u, on réserve des zones de mémoire pouvant loger, respectivement, des sigles ou des entiers, au nombre de: memi. Pour chacun des deux ensembles, E=I et E=J, lors du choix des éléments supplémentaires, le nombre s1u^[e] prend la valeur 0 ou 1 selon que l'élément de rang e de E est principal ou supplémentaire. Ce choix étant fait, les éléments de E reçoivent un nouvel ordre, en deux blocs consécutifs Ep et Es: principaux d'abord, supplémentaires ensuite, l'ordre initial étant conservé au sein de chaque bloc. Le nouveau rang, de l'élément qui avait initialement le rang e, est noté dans: pr1a^[e]. Grâce à pr1a^, ainsi qu'au tableau auxiliaire de sigles, sig1t^, on peut mettre le tableau des données, avec ses sigles et ses nombres, sous la forme déterminée par le nouvel ordre des colonnes et des lignes.

On réserve, pour le pointeur fi, une zone de mémoire pouvant contenir une suite de (carti+1) réels, au format 'extended'.

La procédure d'entrée, 'litaber', se termine en demandant à l'utilsateur de fixer le nombre des facteurs à conserver: d'une part, carf, sur fichier binaire; d'autre part, care, sur lisatge écrit. On a vu, dans 'uver', que carf est limité à aamax=93; pour le listage, le nombre care=10 a été choisi, initialement, afin de tenir dans une ligne de l'imprimante 'imagewriter'; il va sans dire que ces bornes n'ont rien d'essentiel.

### 3.2 Choix et rangement des éléments supplémentaires

Après exécution de 'litaber', on doit, avant d'entreprendre les calculs proprement dits, ranger les données selon le choix des éléments supplémentaires; et d'abord, faire ce choix. Soit par un dialogue à l'écran, régi par la procédure 'ensembler', déjà expliquée en IA§3, avec l'unité 'uens.p'; soit d'après un fichier auxiliaire, lu par la procédure 'suplirer', propre à 'qori'.

#### 3.2.1 Désignation des éléments supplémentaires par un fichier de commande: la procédure 'suplirer'

```
{$S tabl} {$U uens5}
procedure ensembler(carte:integer;plmt,padj,psiglt,psu:ptr);external;
procedure lecnombre;begin ffl:=0;
  while not ((ord(carac) in [48..57]) or eof(ft)) do read(ft,carac);
  while (ord(carac) in [48..57]) do begin
       ffl:=(10*ffl)+(ord(carac)-48);
       if eof(ft) then carac:=' ' else read(ft,carac) end;end;
procedure suplirer;var ii:integer;begin
  writeln('on lit le fichier ',nomf);
  reset(ft,nomf);readln(ft,nomf);ia:=0;carac:=' ';
  while not eof(ft) do begin lecnombre;i:=ffl;ic:=0;
      if (i>10000) then begin i:=i mod 10000;ic:=1;end;
      if (ia=0) then ia:=i;
      if not((i<=0) or (carte<i)) then
        for ii:=ia to i do slu^[ii]:=1;
      ia:=0;if (ic=1) then ia:=i end;
  close(ft);end;
```

En bref, le fichier de commande éventuel, se compose d'une ligne de titre qui est lue d'abord, puis d'une suite de nombres entiers positifs. Ces nombres étant lus par une version simplifiée de la procédure 'lecnombre' (qui sert dans 'ulire' pour un tableau de données: cf. I§2.5.2), on peut y mêler de commentaires, pourvu que ceux-ci soient écrits sans chiffres.

Avec ces nombres, les rangs des éléments supplémentaires sont spécifié soit isolément, soit par blocs. De façon précise, un nombre 'ffl', inférieur o égal au cardinal 'carte' de l'ensemble E (I ou J) considéré, désigne simplemer un élément en supplément; mais si ffl est supérieur à 10000, le reste, ffl mo 10000, est pris comme valeur initiale pour un bloc se terminant par le nombi qui sera lu ensuite. Par exemple, avec carte= 100, la suite de nombres:

{2, 5, 153, 10044, 73, 10020, 35, 10444, 11}

commande la mise en supplément du sous-ensemble:

{2, 5, 20..35, 44..73}

en particulier, 153 étant supérieur à 100, est perdu; le nombre 10444 tendrait à définir un bloc commençant au rang 444 et se terminant à 11; mais un tel bloc est vide, parce que 444>11.

Il est clair qu'un fichier de commande écrit sans erreur s'exécute selon les intentions de l'opérateur; mais d'autre part, la procédure est protégée contre l'écriture au-delà de la zone allouée au pointeur: slu.

### 3.2.2 Choix des éléments supplémentaires pour un ensemble E (I ou J): la procédure 'supeler'

```
procedure supeler;begin adj:='suppl';
  writeln('le nombre des ',lmt,'s est',carte:4);
  for i:=1 to carte do slu^[i]:=0;
  write('y a t il ',lmt,' supplémentaire(S) ou non(N) ');readln(reps);
  if (reps='N') then carel:=carte else begin reps:='S';
   if (verif(stringptr(@nomf))=2) then suplirer else
      ensembler(carte,@lmt,@adj,@siglt^,@slu^);
   cares:=0;
   for i:=1 to carte do cares:=cares+slu^[i];carel:=carte-cares;
   if (carel<2) then begin slu^[1]:=0;slu^[2]:=0;cares:=0;
      for i:=1 to carte do cares:=cares+slu^[i];carel:=carte-cares end;
   ip:=0;is:=carel;
   for i:=1 to carte do begin
     if (slu^[i]=0) then begin ip:=ip+1;ih:=ip end
          else begin is:=is+1;ih:=is end;
     prla^[i]:=ih end end;
  rewrite(fin,concat(nomba,che,'pr'));
  for i:=1 to carte do write(fin,1-slu^[i]);close(fin);end;
```

Suivant les conventions expliquées, pour 'ensembler', dans 'uens', la variable 'lmt' a pour valeur la chaîne de caractères: 'colonne' ou 'ligne', selon qu'il s'agit de E=I ou de E=J. Si l'utilisateur répond 'N' (i.e.: "il n'y a pas d'élément supplémentaire"), le nombre: 'carel', des éléments principaux est égal au nombre total 'carte', des éléments de E, et le traitement de cet ensemble sera simplifié d'autant. Sinon, le programme s'enquiert de l'existence d'un fichier de commande, dont le nom, fixé avant l'appel de 'supeler' n'est autre que le nom de base, complété par la lettre de E (che='i' ou che='j'), et le suffixe 'supx' (cf. *infra*). S'il n'y a pas de fichier 'supx', s'engage un dialogue par 'ensembler'.

Une fois fixé l'ensemble Es, éventuellement vide, des éléments supplémentaires, on procède, grâce au tableau: prla, au renumérotage de E, annoncé, ci-dessus, à propos de 'litaber'. Et l'on crée, avec le suffixe 'pr', un fichier numérique où les éléments, pris dans l'ordre initial du tableau de données I×J, sont marqués par 1 (principal, Ep) ou 0 (supplémentaire, Es). Ce fichier sert à 'qorlsup' (cf.ID§0), qui adjoint à une analyse des tableaux en supplément; à 'VACOR' (cf. IID), qui, pour l'interprétation d'une CAH sur Ep, calcule sur le tableau I×J; ainsi qu'à 'carthag' qui représente, sur une carte, dont les subdivisions sont indicées par E, les classes d'une partition de Ep.

### 3.2.3 Appel des procédures de choix des éléments supplémentaires et rangement du tableau des données: la procédure 'tabuler'

```
procedure tabuler;begin
  lmt:='colonne';carte:=cartj;che:='j';
  nomf:=concat(nomba,che,'supx');
  for j:=1 to carte do begin sig1t^[j]:=spgj^[j];g1ji[j]:=k1ji[j] end;
  supeler;
  if (reps='S') then for j:=1 to cartj do begin
     k1ji[pr1a^[j]]:=g1ji[j];spgj^[pr1a^[j]]:=sig1t^[j] end;
  carj:=carel;if (carj<carf+1) then carf:=carj-1;
  lmt:='ligne';carte:=carti;che:='i';
  nomf:=concat(nomba,che,'supx');
  for i:=1 to carte do sig1t^[i]:=sig1i^[i];
  supeler;cari:=carel;
  if (reps='S') then begin
     for i:=1 to carti do sig1i^[pr1a^[i]]:=sig1t^[i];
     for j:=1 to cartj do begin gi:=k1ji[0];k1ji[0]:=k1ji[j];
     for i:=1 to carti do gi^[pr1a^[i]]:=k1ji[j]^[i];k1ji[j]:=gi end end;
  writeln(titre);end;
```

On choisit d'abord les colonnes supplémentaires (E=J, che='j'). Avant d'appeler 'supeler', on calcule par concaténation, le nom 'nomf' d'un éventuel fichier de commande; et on copie, respectivement dans: sig1t^ et g1ji, les sigles des colonnes et les pointeurs qui désignent le contenu numérique de celles-ci.

S'il y a effectivement des éléments supplémentaires, on utilise la fonction de renumérotage, pr1a^, pour donner aux sigles et pointeurs des colonnes, le nouvel ordre, (Jp ∪ Js). Le nombre, carf, des facteurs demandés, est corrigé, s'il se trouve supérieur à (carj-1), nombre, diminué de 1, des colonnes principales.

Pour l'ensemble, E=I, des lignes, on calcule 'nomf' et copie les sigles dans la tableau auxiliaire: sig1t^, déjà utilisé pour les colonnes. Après exécution de 'supeler', on met les sigles des lignes dans l'ordre (Ip ∪ Is).

- Pour les valeurs numériques, on ne peut procéder globalement par pointeur, mais il faut traiter les cartj colonnes l'une après l'autre. La procédure 'ulire' ayant réservé une colonne de rang 0, on dispose d'une colonne surnuméraire; par les instructions:

gi:=k1ji[0] ; k1ji[0]:=k1ji[j] ;

le pointeur vers la colonne surnuméraire est gardé dans gi; et l'actuelle colonne j est destinée à servir, ultérieurement de colonne surnuméraire. Les nombres de la colonne j sont alors copiés, dans l'espace gi^, suivant le nouvel ordre (Ip ∪ Is). Enfin, l'espace vers lequel pointe gi est attribué à la colonne j (réordonnée); tandis que la colonne j initiale, vers laquelle pointe k1ji[0], peut effectivement servir de colonne surnuméraire.

### 3.3 Construction de la matrice à diagonaliser: la procédure 'carrer'

```
procedure carrer;begin
 for j:=1 to carj do fjj[j]:=prtj(newptr(xtn*cartj));
 for j:=1 to carj do for jp:=1 to carj do fjj[j]^[jp]:=0;mtot:=0;
 for i:=1 to cari do begin fi^[i]:=0;
   write(i:4);if (i mod 20 =0) then writeln;
   for j:=1 to carj do begin
     rj[j]:=klji[j]^[i];fi^[i]:=fi^[i]+rj[j] end;
   if not (fi^[i]=0) then begin
     mtot:=mtot+fi^[i];
     for j:=1 to carj do ffj[j]:=rj[j]/fi^[i];
     for j:=1 to carj do for jp:=1 to carj do
       fjj[j]^[jp]:=fjj[j]^[jp]+(ffj[j]*rj[jp]) end end;
 if (mtot=0) then mtot:=1;writeln;
 for i:=cari+1 to carti do begin fi^[i]:=0;
   for j:=1 to carj do  begin
     rj[j]:=klji[j]^[i];fi^[i]:=fi^[i]+rj[j] end end;
 for j:=1 to carj do begin fj[j]:=0;
   for jp:=1 to carj do fj[j]:=fj[j]+fjj[j]^[jp] end;trace:=-1;
 for j:=1 to carj do if (0<fj[j]) then ffj[j]:=1/sqrt(fj[j]) else ffj[j]:=0;
 for j:=1 to carj do begin
   for jp:=1 to carj do fjj[j]^[jp]:=fjj[j]^[jp]*ffj[j]*ffj[jp];
   fj[j]:=fj[j]/mtot;trace:=trace+fjj[j]^[j] end;
 for i:=1 to carti do fi^[i]:=fi^[i]/mtot;
 res:=sqrt(mtot);for j:=1 to carj do ffj[j]:=res*ffj[j];
 for j:=1 to carj do rj[j]:=sqrt(fj[j]);
 for j:=1 to carj do for jp:=1 to carj do
      fjj[j]^[jp]:=fjj[j]^[jp]-(rj[j]*rj[jp]) ;end;
```

La procedure 'carrer' crée d'abord un tableau carré de réels, au format 'extended'. Plus précisément, le nombre des colonnes de ce tableau n'est autre que le nombre, carj, des colonnes principales; mais le nombre des lignes est égal au cardinal, cartj de l'ensemble J tout entier. Car, ainsi qu'on l'a dit en IA§5.1, l'espace auquel fjj donne accès, est pris en compte, dans 'sortir', par l'intermédiaire du pointeur: Phaj; et, cf. §2.2.4, il contient les facteurs sur J tout entier. Le tableau carré carj × carj est ici mis à zéro.

Vient alors une boucle: for i := 1 to cari do..., qui parcourt la suite des éléments principaux. Dans le tableau fi^, créé par la procédure 'litaber' (cf. §3.1), est mis le total, fi^[i], de la ligne i. Simultanément, on calcule la masse totale, mtot, des lignes principales. Dans la ligne ffj, se trouve le profil, sur Jp, de la ligne principale, i. Quand est achevée l'exécution de cette boucle, le contenu du tableau symétrique fjj n'est autre que ce qu'avec les notations usuelles de l'analyse des correspondances, on écrirait:

$$fjj(j,j') = \Sigma\{k(i,j).k(i,j')/k(i) \mid i \in Ip\} \quad .$$

Dans la boucle: for i:=cari+1 to carti do..., sont calculés les totaux, fi^[i], des lignes supplémentaires.

Dans une première boucle: for j:=1 to carj do..., est calculée la marge, fj, du tableau fjj: chaque terme de cette marge pourrait aussi être calculé par une somme indicée par i, suivant la formule usuelle:

$$fj(j) = \Sigma\{k(i,j) \mid i \in Ip\} \quad ;$$

mais en passant par les valeurs déjà calculées pour fjj (qui est en format 'extended'), on évite que les erreurs d'arrondi sur une longue somme (indicée par Ip) ne mettent en désaccord fjj et sa marge.

Dans les deux boucles suivantes: for j:=1 to carj do..., est calculée la matrice:

$$fjj(j,j') = \Sigma\{k(i,j).k(i,j')/(k(i).\sqrt{(k(j).k(j'))} \mid i \in Ip\} \quad .$$

matrice dont les vecteurs propres, orthogonaux pour la structure euclidienne définie par la somme des carrés des coordonnées, servent à calculer les facteurs normalisés sur Jp; puis, par transition, à calculer les facteurs sur Ip et Is. Simultanément, la distribution de masse, fj, sur Jp, est réduite à avoir pour total 1. Et on calcule la trace de la nouvelle matrice fjj. Il faut prendre garde que, plus précisément, cette trace est diminuée de 1; parce que, à la fin du calcul, la matrice fjj est modifiée pour éliminer le facteur trivial relatif à la valeur propre 1.

Vient alors une boucle: for i := 1 to carti do..., qui, en divisant par mtot les masses fi^[i], réduit celles-ci à avoir pour total 1 sur l'ensemble Ip des éléments principaux.

Enfin, par des boucles: for j:=1 to carj do..., sur l'ensemble Jp des variables principales, on donne forme normalisée à: rj, ffj et fjj.

Dans rj^[j], sont les racines carrées des fréquences (déjà normalisées): fj^[j]. Dans ffj^[j], les inverses de ces racines carrées. Et l'on retranche, de la matrice fjj(j,j'), la matrice des produits $\sqrt{(fj(j).fj(j'))}$.

Ainsi disparaît le vecteur propre trivial relatif à la valeur propre 1; vecteur qui, en l'état du calcul, n'a pas pour composantes une suite de 1, mais la suite: $\{\sqrt{fj(j)} \mid j=1,\ldots, carj\}$. Plus exactement, ce vecteur propre trivial subsiste; mais il est maintenant relatif à la valeur propre 0. Grâce à cette réduction, dans l'analyse de correspondances, les facteurs non triviaux, de moyenne nulle, relatifs à la valeur propre 1 (facteurs associés à des décompositions éventuelles du tableau principal en blocs diagonaux) sortent sans se mêler au facteur trivial.

Celui-ci ne peut perturber que les facteurs afférents à des valeurs propres voisines de zéro; et seulement du fait de l'imprécision des calculs. C'est pourquoi, au début de la procédure 'sortir', cf. §2.2.1, sont éliminés les facteurs Phaj[a] pour lesquels le carré de la moyenne est supérieur à la valeur propre lama[a].

**Référence bibliographique** (voir, aussi, ID, *in fine*)

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).