LES CAHIERS DE L'ANALYSE DES DONNÉES

J.-P. BENZÉCRI
F. BENZÉCRI

Sources de programmes d'analyse de données en langage PASCAL : (III) : élaboration de tableaux divers (IIIB) : création du tableau de Burt à partir du tableau des numéros de modalité

Les cahiers de l'analyse des données, tome 22, n° 3 (1997), p. 271-280

<http://www.numdam.org/item?id=CAD_1997__22_3_271_0>

Les Cahiers de l'Analyse des Données
Vol. XXII - 1997 - n°3 - pp.271 280.

SOURCES DE PROGRAMMES D'ANALYSE DE DONNÉES EN LANGAGE PASCAL: (III) : ÉLABORATION DE TABLEAUX DIVERS (IIIB) : CRÉATION DU TABLEAU DE BURT À PARTIR DU TABLEAU DES NUMÉROS DE MODALITÉ

[SOURCES PASCAL (IIIB)]

J.-P. & F. BENZÉCRI

0 Création du tableau de BURT: le programme 'zBurt'

On sait qu'à partir d'un tableau de base I×J croisant un ensemble I d'individus et un ensemble J de variables, le programme de codage 'zrang' élabore des tableaux de diverses formes (cf. IIIA). Pour créer le tableau de BURT, le programme 'zBurt' utilise, essentiellement, le tableau des numéros de modalités; i.e. un tableau I×J, donnant, à l'intersection de la ligne i et de la colonne j, le numéro de la modalité, de la variable j, afférente à l'individu i.

De façon précise, si le tableau de base est: 'disq:tab', le tableau des numéros de modalité est: 'disq:tabM.z', dans le cas d'un codage logique, en [0, 1]; ou: 'disq:tabS.w', pour un codage barycentrique, avec des numéros continus de modalité (si une valeur de variable tombe entre deux pivots).

Le tableau des numéros de modalité ne suffit pas pour créer le tableau de BURT. En effet, ce tableau ne contient pas les sigles des modalités des différentes variables. De plus, si une variable, j, est éliminée du codage, ce qui revient à lui attribuer zéro modalité, la colonne, j, n'en subsiste pas moins dans le tableau des numéros de modalité, avec des rangs au lieu des numéros de modalité (cf. IIIA, §6.2). Le tableau: 'disq:tabDcodx', fournit à 'zBurt', pour chacune des variables, le nombre de ses modalités avec les sigles de celles-ci.

Dans la publication de la source de 'zBurt', nous suivrons strictement le listage. En effet, d'une part, les procédures sont énumérées dans l'ordre où elles sont appelées; et le programme principal est, en quelque sorte, réduit à être une table des matières. D'autre part, le lecteur des articles précédents de la série des sources reconnaîtra, sans peine, des notations toutes déjà employées ailleurs.

Les cahiers de l'analyse des données 0339-3097/97/03/© Elsevier, Paris

1 Déclarations; et procédures de lecture sur fichier de texte

```
program zBurt;
uses memtypes,quickdraw,osintf,toolintf,sane,uver5;
const mmax=300;
type
colber=array[0..mmax]of longint;pcol=^colber;
cowber=array[0..mmax]of single ;pcow=^cowber;
var fin:file of integer;ft:text;
carac,repc,rpf,bon,rpz,rpw,suB:char;nomba,nomf,titre,vacua:str255;
erl,izl,i,j,jj,jp,m,mp,c,carj,carm,lu,nu,sgn:integer;ret:real;nw:single;
sig:zigle;psgl:pzgl;ttri:stringptr;ffli:plng;ffw:pww;ptrr:ptr;
tampi:array[1..titab]of integer;
sigj:array[1..jmax]of zigle;sigm:array[1..mmax]of zigle;
mdj,nmj,dj:array[1..jmax]of integer;
colm:array[0..mmax]of pcol;cowm:array[0..mmax]of pcow;
wdj:array[1..jmax]of single;
ffl:longint;
procedure lecsig;   begin    sig[0]:=0;
   while (ord(carac) in [0..32]) and not eof(ft) do  read(ft, carac);
   while not ((ord(carac) in [0..32]) or (sig[0]=4)) do begin
     sig[0]:=sig[0]+1;sig[sig[0]]:=ord(carac);
     if eof(ft) then carac:=' ' else read(ft, carac) end;
   while not ((ord(carac) in [0..32]) or eof(ft)) do read(ft, carac);
   if eof(ft) then carac:=' ';end;
procedure lecnombre;begin ffl:=0;sgn:=1;
      while not((ord(carac) in [48..57]) or eof(ft)) do begin
          read(ft,carac);
          if not(ord(carac) in [48..57]) then sgn:=1;
          if (carac='-') then sgn:=-1;end;
      while (ord(carac) in [48..57]) do begin
          ffl:=(10*ffl)+(ord(carac)-48);
          if eof(ft) then carac:=' ' else read(ft,carac) end;
      ffl:=sgn*ffl; end;
```

La constante: mmax=300, est le maximum du nombre total des modalités pour l'ensemble des variables; on a pris: mmax=jmax, nombre maximum de colonnes dans un tableau lu par la procedure: 'litab', (cf. IA§2), afin que le tableau de BURT puisse être analysé par 'qori'; etc.

Les formats: 'colber', et: 'cowber', sont destinés aux colonnes du t. de BURT; le format réel de 'cowber' servant pour le tableau généralisé.

Les tableaux de pointeurs: colm, ou:cowm, donnent accès à l'espace réservé au tableau de BURT, en cours de création (cf. *infra* §3.2).

On a prévu un indice zéro, en haut de chaque colonne (dans les formats: colber, et: cowber), et une colonne zéro en tête du tableau (dans les tableaux de pointeurs: colm, et: cowm), afin que les ordres d'écriture dans le BURT, ou de mise à jour, (cf. *infra*, procédure: 'burt') s'éxécutent même en ligne ou colonne zéro; le zéro étant attribué, par convention, comme numéro de modalité à une variable écartée du codage.

Les tableaux: sigj, et: sigm, déclarés comme: array[1..mmax]of zigle, contiennent, repectivement, (sous le format: zigle, cf. IA§1.2), les sigles des

variables du tableau de base et ceux des modalités en lesquelles chacune de ces variables est découpée.

On note: nmj[j], le nombre de modalités de la variable: j. La suite des blocs de modalités, afférents aux variables successives, est numérotée de 1 à: carm. Ceci posé, le rang de la dernière modalité précédant le début du bloc: j, est: d(j). Le rang de la modalité de j, pour l'individu en cours de traitement, est: mdj[j]; ou, dans le cas continu: wdj[j].

Les procédures: 'lecsig', et: 'lecnombre', sont celles déjà vues dans 'ulire' (cf. IA§2). Elles servent à lire les sigles et les nombres entiers, sur le fichier de codage: 'Dcodx'; ainsi que sur le fichier des numéros de modalité, si celui-ci est en format de texte (cf. §2.1)

2 Choix des données à traiter et vérifications générales

2.1 Existence du fichier des numéros de modalité et du fichier de codage: la procédure 'nommer'

```
procedure nommer;begin
 repc:='N';erl:=0;
 while not ((repc='O') or (erl=6)) do begin repc:='O';
  write('le fichier des données est ');readln(nomba);
  write('a-t-on un codage en [0,1](Z), ou un codage barycentrique(B) ');
  readln(rpw);
  if (rpw='B') then rpw:='S' else rpw:='M';
  if (rpw='S') then
    writeln('  codage barycentrique') else writeln('  codage en [0,1]');
  if (rpw='S') then nomf:=concat(nomba,'S.w') else nomf:=concat(nomba,'M');
  if (rpw='S') then suB:='w' else suB:='z';
  writeln('Vérification pour le fichier des numéros de modalités ',nomf);
  iz1:=1;
  if (rpw='M') then begin
   iz1:=verif(stringptr(@nomf));
    if (iz1=1) then nomf:=concat(nomf,'.z');end;
  repc:=dialof(stringptr(@nomf));
  if (repc='O') then begin writeln(nomf);nomf:=concat(nomba,'Dcodx');
    writeln('Vérification pour le fichier de codage des modalités ',nomf);
    repc:=dialof(stringptr(@nomf)) end;
  if (repc='N') then erl:=erl+1 else begin writeln(nomf);
   write('le nom du fichier des données est il confirmé oui(O) ou non(N) ');
   readln(repc) end end;end;
```

Nous l'avons dit au §0, le fichier des numéros de modalité, créé par 'zrang', peut être: 'disq:tabM.z', ou: 'disq:tabS.w'; ce dernier format (en nombres réels) correspondant au codage barycentrique. Et, selon le cas, le tableau de BURT: 'disq:tabB', sera créé avec le suffixe de format; suB='z', ou: suB='w'.

Plus précisément, on admet que, dans le cas du codage logique en [0, 1], le tableau des numéros (ayant été créé autrement que par 'zrang') puisse être en format de texte. Dans ce dernier cas, 'nommer' donne à: iz1, la valeur 1; sinon: iz1=2.

2.2 Lecture du fichier de codage et vérification du nombre des modalités: la procédure 'vercoder'

La structure du fichier: 'Dcodx', est expliquée dans IIIA§1.1.3.1.

En tête du fichier: 'Dcodx', la procédure: 'vercoder' lit, après deux lignes de titre, le nombre: carj, des colonnes du tableau de base. Puis, dans chacun des blocs de 'Dcodx' afférents à une variable: j, on lit le sigle de celle-ci: sigj[j]; le nombre des modalités qui lui sont attribuées; et les sigles de celles-ci. (Les valeurs des bornes, ou des pivots, lues par l'ordre: read(ft,ret), ne sont pas prises en compte).

```
procedure vercoder;begin
  reset(ft,concat(nomba,'Dcodx'));
  readln(ft,titre);readln(ft,vacua);
  carac:=' ';lecnombre;carj:=ffl;carm:=0;
  for j:=1 to carj do begin dj[j]:=carm;carac:=' ';
     lecsig;sigj[j]:=sig;
      lecnombre;nmj[j]:=ffl;readln(ft,vacua);carac:=' ';
     for m:=1 to nmj[j] do begin carac:=' ';
         carm:=carm+1;lecsig;
          if (carm<=mmax) then sigm[carm]:=sig end;
     for m:=1 to nmj[j] do read(ft,ret);end;
  close(ft);
  if (mmax<carm) then begin repc:='N';
     writeln('ERREUR le nombre de modalités dépasse',mmax:4) end end;
```

Ainsi, la procédure 'vercoder' délimite les blocs de modalités, avec (cf. §1): dj[j] =numéro (éventuellement nul) précédant celui de la 1-ère modalité du bloc afférent à la variable j; conserve le nombre:nmj[j], des modalités du bloc j; et range les sigles de ces modalités dans le tableau: sigm.

On vérifie alors que le nombre total:carm, des modalités demandées n'excède pas: mmax.

Cette dernière éventualité ne peut se présenter si c'est par 'zrang' qu'a été créé le fichier: 'Dcodx', (cf. IIIA§5.3); ou, même, le seul fichier des numéros de modalités, (cf. IIIA§6.1); mais il n'est pas exclu que, notamment par concaténation, soient créés, par l'utilisateur, des fichiers sortant des limites imposées par 'zrang'.

2.3 Concordance des sigles de variable entre le fichier des numéros de modalité et le fichier de codage: les procédures 'teter', 'tetez' et 'verteter'

Selon que le fichier des numéros de modalité est en format de texte (iz1=1) ou en un format binaire (iz2=2), on utilise soit la procédure: 'teter', soit la procédure: 'tetez' pour vérifier que les sigles de variable, lus en tête du tableau des numéros, concordent avec ceux lus par 'vercoder' dans le fichier: 'Dcodx'.

```
procedure teter;begin
 reset(ft,concat(nomba,'M'));
 readln(ft,vacua);carac:=' ';
 lecnombre;if not(carj=ffl mod 1000) then repc:='N';
 if (repc='O') then for j:=1 to carj do begin lecsig;
     if not(sigler(sig)=sigler(sigj[j])) then repc:='N' end;
 if (repc='N') then close(ft);end;
procedure tetez;begin
 reset(fin,concat(nomba,rpw,'.',suB));
 for c:=1 to titab do read(fin,tampi[c]);
 read(fin,lu);if not(carj=lu) then repc:='N';
 if (repc='O') then for j:=1 to carj do begin
     for c:=1 to 3 do read(fin,tampi[c]);
     if not(sigler(psgl^)=sigler(sigj[j])) then repc:='N' end;
 if (repc='N') then close(fin);end;

procedure verteter;begin
  ffli:=plng(@tampi);ffw:=pww(@tampi);psgl:=pzgl(@tampi);ttri:=stringptr(@tampi)
  if (izl=1) then tetez;if (izl=2) then teter;
  if (repc='N') then writeln('ERREUR sur les sigles des variables') end;
```

La procédure 'verteter', qui appelle, selon le cas, 'teter' ou 'tetez', avertit l'utilisateur d'une erreur éventuelle; laquelle fait tourner court la création du tableau de BURT.

D'autre part, 'verteter' donne pour valeur aux trois pointeurs: ffli, ffw et ttri, l'adresse initiale du tableau d'entiers: tampi; ce qui permet d'écrire dans l'espace (en mémoire centrale) résevé à: tampi, soit un entier (en format long: par ffli), soit un réel (single: par ffw), soit une chaîne de caractères (sigle ou titre:par ttri).

Ainsi, les informations de nature diverse (titre, sigles, nombres) qui constituent le tableau de BURT, seront transférées, de la mémoire centrale, dans un fichier externe (i.e; sur disque); lequel est déclaré: 'file of integer', qu'il s'agisse d'un BURT usuel (suffixe: '.z', nombres entiers) ou d'un BURT généralisé (suffixe: '.w', nombres réels).

3 Création du tableau de BURT en mémoire centrale

3.1 Lecture des informations afférentes à un individu: les procédures 'verinum', 'liger', 'ligez'

```
procedure verinum;begin
 if ((not (nmj[jj]=0)) and ((nu<1) or (nmj[jj]<nu))) then begin
    bon:='N';vacua:=sigler(sigj[jj]);
    write('ERREUR',vacua:5,' a',nmj[jj]:3,' modalités et pour');
    write(sigler(sig):5,' le numéro est',nu:3);readln(rpf);nu:=0 end;end;
```

Le numéro, nu, de la modalité de la variable jj, afférente à un individu, (tel celui, en cours de traitement, dont le sigle est: sigler(sig)), peut aller de 1 à nmj[jj], (nombre des modalités de cette variable). L'inégalité: $1 \leq nu \leq nmj[jj]$, doit être vérifiée; à la condition, toutefois, que la variable jj soit effectivement retenue pour le découpage, i.e. que: $nmj[jj] \neq 0$.

```
procedure liger;begin lecsig;
 if not(sig[0]=0) then for j:=1 to carj do begin lecnombre;
   nu:=ffl;jj:=j;verinum;
   mdj[j]:=nu+dj[j] end;end;
```

Dans le cas où le fichier des numéros de modalités est en format de texte, il s'agit toujours de codage en [0, 1]; la ligne afférente à un individu est lue, par 'liger', avec les procédures 'lecsig' et 'lecnombre'. On notera que le nombre: mdj[j], est le rang, dans la suite des 'carm' modalités de l'ensemble des variables retenues, de la modalité propre à l'individu considéré. La condition initiale: not(sig[0]=0), élimine le cas où, la lecture du texte étant achevée, il ne resterait plus, au delà du dernier nombre lu, que du blanc (ou, plus généralement, des caractères dont le rang ASCII est de 1 à 32).

```
procedure ligez;begin
 for c:=1 to 3 do read(fin,tampi[c]);sig:=psgl^;
 for j:=1 to carj do begin read(fin,tampi[1]);read(fin,tampi[2]);
   nu:=ffli^;nw:=ffw^;
   if (rpw='S') then begin wdj[j]:=nw+dj[j];
      nu:=trunc(0+nw);if (nu<nw) then nu:=nu+1;if (nw<1) then nu:=0;end;
   jj:=j;verinum;
   mdj[j]:=nu+dj[j] end;end;
```

Dans le cas général, d'un fichier de numéros de modalité en format binaire, le traitement d'un individu commence par la lecture du sigle. Puis les numéros de modalité sont lus dans une boucle. Il faut distinguer entre codage en [0, 1] et codage barycentrique. Dans le premier cas, le numéro entier: nu, est traité par 'ligez' comme le ferait 'liger'.

Mais si le suffixe du fichier des numéros de modalités est: 'S.w', chacun des numéros de modalité est lu, d'abord, comme un réel: nw. Pour une variable effectivement retenue pour le découpage, nw, peut être un entier si la valeur de la variable coincide avec l'un des pivots; ou, encore, si elle précède le 1-er pivot (nw=1), ou est supérieure au dernier (nw=n mj[j]). En général, la variable tombe entre deux pivots consécutifs; et l'on prend alors pour: nu, le numéro du pivot immédiatement supérieur. On notera qu'il y a, à la fois, pour chaque variable, un rang entier de modalité, mdj[j], et un rang continu, wdj[j].

3.2 Mise à jour du tableau de BURT après lecture d'un individu: la procédure 'burt'

Le sigle de l'individu est affiché à l'écran. Pour toute variable non retenue pour le codage (nmj[j]=0), on met à zéro le numéro de la modalité, mdj[j], afférente à l'individu traité.

Il faut prendre garde que, selon la définition usuelle du t. de BURT, kB:

$$kB(m, m') = \sum\{k(i, m) \,.\, k(i, m') \mid i \in I\} ;$$

le traitement d'un individu semble demander une double boucle, indicée par les $carm^2$ couples de modalités: (m, m').

```
procedure burt;
  var wjj,wjp:integer;aoo,aou,auo,auu,rjj,rjp:single;
  begin writeln(sigler(sig));
  for j:=1 to carj do if (nmj[j]=0) then mdj[j]:=0;
  if (rpw='M') then for j:=1 to carj do for jp:=j to carj do
    colm[mdj[j]]^[mdj[jp]]:=colm[mdj[j]]^[mdj[jp]]+1;
  if (rpw='S') then for j:=1 to carj do if not(nmj[j]=0) then
    for jp:=j to carj do if not(nmj[jp]=0) then begin
      wjj:=trunc(0+wdj[j]);wjp:=trunc(0+wdj[jp]);
      rjj:=wdj[j]-wjj;rjp:=wdj[jp]-wjp;
      auu:=rjj*rjp;aou:=rjp-auu;auo:=rjj-auu;aoo:=1+auu-(rjj+rjp);
      if (0<aoo) then cowm[wjj]^[wjp]:=cowm[wjj]^[wjp]+aoo;
      if (0<auo) then cowm[wjj+1]^[wjp]:=cowm[wjj+1]^[wjp]+auo;
      if (0<aou) then cowm[wjj]^[wjp+1]:=cowm[wjj]^[wjp+1]+aou;
      if (0<auu) then cowm[wjj+1]^[wjp+1]:=cowm[wjj+1]^[wjp+1]+auu;end;
end;
```

Il est d'abord clair que ce nombre peut être, à peu près, divisé par deux, en se bornant aux couples (m≤m'); cf. *infra*, §3.3, procédure 'compler'. Mais, on peut faire mieux avec une boucle indicée non sur les couples de modalités mais sur les couples de variables. Économie non négligeable, en l'état présent des moyens de calcul, si l'on code barycentriquement plusieurs milliers d'individus, chacun décrit par des dizaines de variables ayant, au moins, trois modalités.

De façon précise, avec le codage en [0, 1], (i.e. rpw='M'), pour chaque couple (j, jp), avec (j≤jp), on ajoute 1 dans la case: mdj[j]], mdj[jp], du tableau de BURT.

Le calcul est plus complexe avec le codage barycentrique; car, pour chacune des variables j, il faut considérer les deux modalités consécutives, {wjj, wjj+1} (afférentes aux pivots entre lesquels tombe la valeur de j). Pour deux variables, j et jp, il y a quatre cases à modifier. Les quantités à ajouter sont des produits de masses comprises entre 0 et 1; lesquelles, en vertu du principe barycentrique, ne sont autres que la partie fractionnaire du numéro continu (i.e. la différence obtenue de celui-ci en retanchant la partie entière), ou le complément à 1 de cette partie fractionnaire.

3.3 Construction globale du tableau de BURT: les procédures 'burter', 'burtez', 'compler' et 'creburter'

```
procedure burter;begin
 while not(eof(ft) or (bon='N')) do begin
     liger;if not(sig[0]=0) then burt end;
 close(ft);end;
procedure burtez;begin
 while not(eof(fin) or (bon='N')) do begin ligez;if (bon='O') then burt end;
 close(fin);end;
```

Lecture d'un individu et mise à jour du tableau de BURT se répètent, dans une boucle qui tourne tant que n'est pas épuisé le fichier des numéros de modalité: procédure 'burter' pour un fichier de texte; et 'burtez' pour un fichier binaire.

```
procedure compler;begin
 for j:=1 to carj-1 do
   for m:=dj[j]+1 to dj[j]+nmj[j] do
     for mp:=dj[j]+nmj[j]+1 to carm do colm[mp]^[m]:=colm[m]^[mp];end;
```

Le tableau de BURT se compose de blocs rectangulaires (j, jp), croisant les intervalles: {dj[j]+1.. dj[j]+nmj[j]} et {dj[jp]+1..dj[jp]+nmj[jp]}. Plus précisément, pour une variable éliminée du codage, l'intervalle est réduit à l'unique indice: 0 (zéro).

Telle que nous l'avons écrite, la procedure 'burt' tient à jour les blocs (j≤jp): blocs diagonaux, ou blocs situés au-dessus de la diagonale. Pour compléter le tableau, il suffit de créer, par copie, les blocs (j>jp); ce que 'compler' fait en copiant en colonne, sous chaque bloc carré diagonal, les fins de ligne déjà écrites à sa droite.

```
procedure creburter;begin
  if (rpw='M') then ffli^:=0 else ffw^:=0;
  for m:=0 to carm do begin ptrr:=newptr(4*(1+carm));
     colm[m]:=pcol(ptrr);cowm[m]:=pcow(ptrr);
     for mp:=0 to carm do colm[m]^[mp]:=ffli^ end;
  if (iz1=1) then burtez;if (iz1=2) then burter;
  if (bon='O') then compler;end;
```

La procédure 'creburter' affecte d'abord, en mémoire centrale, l'espace requis pour le t. de BURT; puis elle coordonne toutes les opérations déjà décrites dans le présent §3.

Avec un ensemble de valeurs indicées par [0..mmax], chaque colonne occupe (4.(1+carm)) octets. Cet espace peut être lu en format entier (longint), par le pointeur: colm; ou en format réel (single) par: cowm. Sous les deux formats, la colonne mise à zéro contient la même suite d'octets.

4 Écriture du tableau de BURT sur le disque et libération de la mémoire centrale: les procédures 'sorburter' et 'vider'

```
procedure sorburter;begin
    rewrite(fin,concat(nomba,'B.',suB));
    ttri^:=titre;tampi[titab]:=carm;
    for c:=1 to titab do write(fin,tampi[c]);write(fin,carm);
    for m:=1 to carm do begin psg1^:=sigm[m];
       for c:=1 to 3 do write(fin,tampi[c]) end;
    for m:=1 to carm do begin psg1^:=sigm[m];
       for c:=1 to 3 do write(fin,tampi[c]);
      for mp:=1 to carm do begin ffli^:=colm[m]^[mp];
        write(fin,tampi[1]);write(fin,tampi[2]) end end;
    close(fin) end;
```

Pour créer le fichier du tableau de BURT, on n'a pas utilisé la procédure générale 'critab' (expliquée dans IID§5), mais une procédure particulière. En effet, l'existence d'une ligne et d'une colonne de rang zéro, obligerait à recadrer le tableau des pointeurs et les pointeurs eux-mêmes; alors que le format du BURT en mémoire centrale se prête à une écriture immédiate.

On écrit d'abord le titre et le nombre: carm, des lignes et des colonnes. Dans le bloc initial, ou pour chaque ligne, le sigle d'une modalité est copié comme trois entiers. Que le format soit réel ou entier, un nombre, une fois transféré sur le disque, occupe deux entiers (4 octets). On notera que la valeur peut toujours être lue comme un entier (longint): colm[m]^[mp], même dans le cas du codage barycentrique, parce que seul compte la saisie de 4 octets consécutifs, par un pointeur tel que: ffli^.

```
procedure vider;begin for m:=0 to carm do dispose(colm[m]); end;
```

Comme de règle, l'exécution du programme doit s'achever en libérant la mémoire centrale: tels est l'objet de la procédure: 'vider'.

5 Le programme principal 'zBurt'

```
begin  benzecri;rpz:='O';
while not(rpz='N') do begin nommer;
 if (repc='O') then vercoder;
 if (repc='O') then verteter;

 if (repc='O') then begin bon:='O';
  creburter;if (bon='O') then sorburter;
  vider;end;

 if (repc='N') then rpz :='N' else begin
  write('faut il traiter un autre tableau O ou N ');readln(rpz) end end;
readln(rpf);end.
```

Ainsi qu'on l'a annoncé au §0, le programme principal est, en quelque sorte, réduit à être une table des matières.

Après une procédure initiale qui permet, en bref, d'écrire, de lire et de dessiner dans une fenêtre de l'écran du Macintosh (cf. IA§1.3.1), le programme entre dans un boucle dont il ne sort que si le caractère: rpz, prend la valeur: 'N'.

Si les procédures: 'nommer', 'vercoder' et 'verteter' (cf. §2) se sont exécutées sans rencontrer d'obstacle, on a des fichiers de données compatibles entre eux: repc='O'.

À cette condition, le programme 'zBurt' entreprend, par 'creburter', de créer un tableau de BURT, en lisant consécutivement les informations afférentes aux individus. Dans cette étape même, la procédure 'verinum' (cf. §3.1) peut déceler des incompatibilités dans les données. Si tel n'est pas le cas, on a: bon='O', et, par 'sorburter', le tableau de BURT est transféré sur disque. Même si la procédure 'creburter' a tourné court, il convient de libérer la mémoire, par: 'vider'.

Mis à part le cas où: repc='N', le programme offre à l'utilisateur de traiter un autre tableau (par quoi, il faut entendre: un tableau de base; lui-même déjà élaboré, avec un fichier 'Dcodx' et un fichier de numéros de modalité).

En conclusion, nous rappellerons l'inventaire des fichiers lus ou créés par 'zBurt' dans le traitement d'un tableau de base: 'disq:tab'. On doit distinguer deux cas: codage en [0, 1]: cas 'M'; et codage barycentrique: 'S'.

Dans les deux cas, 'zrang' lit le fichier de codage: 'disq:tabDcodx'.

Dans le cas 'M', le tableau des numéros de modalités peut être en format de texte: 'disq:tabM', ou en format binaire: 'disq:tabM.z'; et le tableau de BURT est créé en format binaire: 'disq:tabB.z', pour qu'on y lise des entiers.

Dans le cas 'S', le tableau des numéros continus de modalités est en format binaire: 'disq:tabS.w'; et le tableau de BURT est créé en format binaire: 'disq:tabB.w', pour qu'on y lise des réels.

Références bibliographiques

B. GHERMANI, C. & M. ROUX: "Sur le codage logique des données hétérogènes: présentation de deux programmes permettant de rendre homogènes des données quelconques"; in *CAD*, Vol.II, n°1, pp. 115-118; (1977);

A. El OUADRANI : "Généralisation du tableau de BURT et de l'analyse de ses sous-tableaux dans le cas d'un codage barycentrique", [BURT COD. BARY.]; in *CAD*, Vol.XIX, n°2, pp. 229-246; (1994);

Une étude récente fondée sur l'analyse d'un tableau de BURT généralisé se trouve sous le titre:

G. D. MAÏTI, G. CARGILL : "Variation de l'acidité œsophagienne et des reflux gastro-œsophagiens en fonction de l'âge et des circonstances diagnostiques chez 2739 patients" , [ACIDITÉ ŒSOPHAGIENNE]; in*CAD*, Vol.XXII, n°1, pp. 83–94; (1997).