ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XIV, 4 (1975)

ALGORITHM 40

ANNA BARTKOWIAK (Wroclaw)

SEARCH FOR MARGINAL MEANS AT GIVEN FACTOR LEVELS IN AN n-WAY TABLE CONTAINING DATA SCORES AND ALL MARGINAL MEANS

1. Procedure declaration. Given a one-dimensional array containing data scores and all marginal means, procedure *searchmeans* enables, for declared factors, to obtain, in the lexicographical order, all single means, all double means, all triple means etc., up to all l-tuples of marginal means.

Data:

l — integer indicating with how many factors at last the marginal means should be calculated;

$n1$ — integer indicating how many factors (counted from the first) should be entered in the calculated l-tuples of factors;

n — number of factors under investigation;

$f[1:n]$ — factor levels;

$maxf$ — the largest value in the array f;

$a[0:\prod_{i=1}^{n}(f[i]+1)-1]$ — array containing data scores and all marginal means, calculated, e.g., by procedure *means* (see [1]);

combi — identifier of the procedure generating distinct combinations of the first n integers, taken r at a time (see [2]); the heading of this procedure should be

procedure *combi*$(n, r, ind, first)$;
integer n, r; **integer array** *ind*; **Boolean** *first*;

Procedure *combi* should generate distinct combinations of the first n integers, taken r at a time, starting with an initial combination $1, 2, \ldots, r$.

The actual combination should be stored in the integer array $ind[1:r]$. Each call of *combi*, after the first, must have in *ind* the previous generated combination and change it into another combination.

The Boolean variable *first* is set **true** in *printmeans* before the first call of *combi*. The variable *first* remains **false** untill all combinations have been generated. Then *combi* sets it **false**.

setup — identifier of the procedure setting the values of the local arrays *m*, *fl* described as follows:

$$fl[i] = f[i] \quad \text{for } i = 1, \ldots, n,$$

$$m[n] = 1, \quad m[i] = \prod_{j=i+1}^{n} (f[j]+1) \quad \text{for } i = 1, \ldots, m-1.$$

The heading of this procedure should be

procedure *setup*(n, f, fl, m);
value n; **integer** n; **integer array** f, fl, m;

address — identifier of the procedure calculating the address of the desired marginal mean stored in *a*. The heading of *address* should be

integer procedure *address*(n, f, fl, m);
integer n; **integer array** f, fl, h;

The value of *address* is set as follows:

$$address := \sum_{i=1}^{n} (f[i]-fl[i]) \times m[i].$$

Results:

Results are obtained by procedure *printmeans* which is called several times during the run of *searchmeans* yielding subsequent r-tuples of marginal means. Procedure *printmeans* should be headed as follows:

procedure *printmeans*$(r, ind, fl, a1, p)$;
value r, p; **integer** r, p; **integer array** ind, fl; **array** $a1$;

We get by every call of *printmeans*:

r — number of factors for which the marginal means are calculated (r satisfies the condition $1 \leqslant r \leqslant n$);

$ind[1:n]$ — first r values of *ind* indicate the factors for which actually the marginal means are calculated ($1 \leqslant ind[i] \leqslant n$ for $i = 1, \ldots, r$);

$fl[1:n]$ — auxiliary factor levels; to get the true factor levels $fa[i]$ for the calculated means, we have to calculate them by the formula

$$fa[i] = f[i]-fl[i]+1 \quad \text{for } i = ind[1], ind[2], \ldots, ind[r-1];$$

the last value of this array is always equal to the highest level of the r-th factor under consideration;

a1[1:maxf] — first *ind[r]* values of that array give the desired marginal means;

p — number of data scores from which each already calculated marginal mean is obtained.

```
procedure searchmeans(l,n1,n,f,maxf,a,combi,setup,address,
 printmeans);
 value l,n1,n,maxf;
 integer l,n1,n,maxf;
 integer array f;
 array a;
 procedure combi,setup,printmeans;
 integer procedure address;
 begin
  integer i,j,k,fk,mk,p,q,r,s;
  integer array b,f1,m,ind[1:n];
  array a1[1:maxf];
  Boolean first;
  setup(n,f,f1,m);
  for r:=1 step 1 until l do
   begin
    first:=true;
newcomb:
    combi(n1,r,ind,first);
    if first
     then go to fin;
    for i:=1 step 1 until n do
     b[i]:=f1[i]:=0;
    p:=1;
    for i:=1 step 1 until r do
     begin
      j:=ind[i];
      p:=p×f[j];
      f1[j]:=f[j];
```

```
        b[j]:=1
       end i;
      s:=address(n,f,fl,m);
      k:=n+1;
et:   k:=k-1;
      if k=0
       then go to fin;
      if b[k]=0
       then go to et;
      mk:=m[k];
      fk:=f[k];
      q:=s;
      for i:=1 step 1 until fk do
       begin
        a1[i]:=a[q];
        q:=q+mk
       end i;
      printmeans(r,ind,fl,a1,p);
et1:  k:=k-1;
      if k=0
       then go to fin;
      if b[k]=0
       then go to et1;
      s:=s+m[k];
      if fl[k] gt 1
       then
       begin
        fl[k]:=fl[k]-1;
        k:=n+1;
        go to et
```

```
        end fl[k] gt 0;
      fk:=fl[k]:=f[k];
      s:=s-fk×m[k];
      go to et1;
fin:if not first
      then go to newcomb
    end r
  end searchmeans
```

2. Method used. The order in which the marginal means should be put in the n-way table is explained in [1]. Procedure *searchmeans* uses, first, procedure *setup* which sets the auxiliary arrays *fl* and *m*. Then, for subsequent values of $r = 1, 2, \ldots, l$, the following calculations are done:

First, set the value of r equal 1. Next:

(a) By the call of procedure *combi*, calculate the appropriate combination of the first n integers, taken r at a time (the initial combination is $1, 2, \ldots, r$).

(b) Set an auxiliary array $b[1:n]$, putting 1 for those factors which are in the combination already generated, and 0 elsewhere.

(c) Get, by the call of procedure *address*, the address of the first desired marginal mean at lowest levels of factors appearing in the actual combination. Pick up the element $a[s]$ corresponding to that address and place it as the first element into the array *a1*.

(d) Changing appropriately the levels of *fl* and augmenting the value s by appropriate values of $m[i]$, where i denotes the ulterior (last) factor appearing in the combination, get subsequent marginal means and place them into the array *a1*.

(e) When all levels of the last factor are exhausted, store the actual values of the array *a1* by the call of procedure *printmeans*.

(f) Now re-establish the initial level of the last factor and repeat points (c)-(e) for a higher level of the former factors as many times till all the combinations of all factors are exhausted.

(h) If $r < l$, jump to point (a) with $r := r+1$.

3. Certification. Let us call procedure *searchmeans* with the following values:

$$l = 2, \quad n1 = 3, \quad n = 3, \quad f = [2, 3, 4], \quad maxf = 4,$$

$$
\begin{aligned}
a = [&6.5, \quad 2.7, \quad 4.0, \quad 4.1, \quad 4.325,\\
&5.2, \quad 4.5, \quad 4.1, \quad 3.4, \quad 4.300,\\
&5.6, \quad 4.1, \quad 3.6, \quad 5.5, \quad 4.700,\\
&5.767, \; 3.767, \; 3.900, \; 4.333, \; 4.442,\\
&6.5, \quad 4.2, \quad 4.7, \quad 4.4, \quad 4.950,\\
&5.1, \quad 3.5, \quad 4.9, \quad 5.2, \quad 4.675,\\
&6.1, \quad 3.2, \quad 3.7, \quad 3.8, \quad 4.200,\\
&5.900, \; 3.633, \; 4.433, \; 4.467, \; 4.608,\\
&6.500, \; 3.450, \; 4.350, \; 4.250, \; 4.637,\\
&5.150, \; 4.000, \; 4.500, \; 4.300, \; 4.487,\\
&5.850, \; 3.650, \; 3.650, \; 4.650, \; 4.450,\\
&5.833, \; 3.700, \; 4.167, \; 4.400, \; 4.525].
\end{aligned}
$$

For each call only the first r elements of *ind* are defined indicating, and their factors are actually dealed with. Further elements are undefined (casual) and their value is denoted below by XXX.

Analogically, the array *a1*, containing the means desired, is defined only at the first $f[ind[r]]$ places. If $f[ind[r]]$ is less than *maxf*, the upper subscript bound of *a1*, further elements of *a1* are casual (undefined), and their value is denoted below by XXX.

Procedure *printmeans* is called during the run of *searchmeans* 10 times and we get the following results:

The first call of *printmeans* brings the single means for the first factor:

$$
\begin{gathered}
r = 1, \quad ind[1:3] = [1, XXX, XXX], \quad fl[1:3] = [2, 0, 0],\\
a1[1:4] = [4.442, 4.608, XXX, XXX] = [x_{1..}, x_{2..}, XXX, XXX],\\
p = 12.
\end{gathered}
$$

The second call of *printmeans* brings the single means for the second factor:

$$
\begin{gathered}
r = 1, \quad ind[1:3] = [2, XXX, XXX], \quad fl[1:3] = [0, 3, 0],\\
a1[1:4] = [4.637, 4.487, 4.450, XXX] = [x_{.1.}, x_{.2.}, x_{.3.}, XXX],\\
p = 8.
\end{gathered}
$$

The third call of *printmeans* brings the single means for the third factor:

$$
\begin{gathered}
r = 1, \quad ind[1:3] = [3, XXX, XXX], \quad fl[1:3] = [0, 0, 4].\\
a1[1:4] = [5.833, 3.700, 4.167, 4.400] = [x_{..1}, x_{..2}, x_{..3}, x_{..4}], \quad p = 6.
\end{gathered}
$$

The fourth call of *printmeans* brings double means calculated at the first level of factor A and at all levels of factor B:

$$
\begin{gathered}
r = 2, \quad ind[1:3] = [1, 2, XXX], \quad fl[1:3] = [2, 3, 0],\\
a1[1:4] = [4.325, 4.300, 4.700, XXX] = [x_{11.}, x_{12.}, x_{13.}, XXX], \quad p = 4.
\end{gathered}
$$

The fifth call of *printmeans* brings subsequent double means for the second level of factor A and for all levels of factor B:

$$r = 2, \quad ind[1:3] = [1, 2, XXX], \quad fl[1:3] = [1, 3, 0],$$

$$a1[1:4] = [4.950, 4.675, 4.200, XXX] = [x_{21.}, x_{22.}, x_{23.}, XXX], \quad p = 4.$$

The sixth call of *printmeans* brings double means at the first level of factor A and at all levels of factor C:

$$r = 2, \quad ind[1:3] = [1, 3, XXX], \quad fl[1:3] = [2, 0, 4],$$

$$a1[1:4] = [5.767, 3.767, 3.900, 4.333] = [x_{1.1}, x_{1.2}, x_{1.3}, x_{1.4}], \quad p = 3.$$

The seventh call of *printmeans* brings subsequent double means at the second level of factor A and at all levels of factor C:

$$r = 2, \quad ind[1:3] = [1, 3, XXX], \quad fl[1:3] = [1, 0, 4],$$

$$a1[1:4] = [5.900, 3.633, 4.4333, 4.467] = [x_{2.1}, x_{2.2}, x_{2.3}, x_{2.4}], \quad p = 3.$$

The eighth call of *printmeans* brings double means for the first level of factor B and for all levels of factor C:

$$r = 2, \quad ind[1:3] = [2, 3, XXX], \quad fl[1:3] = [0, 3, 4],$$

$$a1[1:4] = [6.500, 3.450, 4.350, 4.250] = [x_{.11}, x_{.12}, x_{.13}, x_{.14}], \quad p = 2.$$

The ninth call of *printmeans* brings subsequent double means for the second level of factor B and for all levels of factor C:

$$r = 2, \quad ind[1:3] = [2, 3, XXX], \quad fl[1:3] = [0, 2, 4],$$

$$a1[1:4] = [5.515, 4.000, 4.500, 4.300] = [x_{.21}, x_{.22}, x_{.23}, x_{.24}], \quad p = 2.$$

The tenth call of *printmeans* brings subsequent double means for the third level of factor B and for all levels of factor C:

$$r = 2, \quad ind[1:3] = [2, 3, XXX], \quad fl[1:3] = [0, 1, 4],$$

$$a1[1:4] = [5.85, 3.650, 3.650, 4.650] = [x_{.31}, x_{.32}, x_{.33}, x_{.34}], \quad p = 2.$$

References

[1] A. Bartkowiak, *Calculation of all marginal means from an n-way table*, this fascicle, p. 639-645.

[2] Ch. J. M. Mifsud, *Algorithm 154, Combinations in lexicographical order, procedure COMB1*, CACM 6 (1963), p. 103.

COMPUTING CENTRE
MATHEMATICAL INSTITUTE
UNIVERSITY OF WROCŁAW
50-384 WROCŁAW

Received on 17. 1. 1974

ANNA BARTKOWIAK (Wrocław)

ALGORYTM 40

WYSZUKIWANIE ŚREDNICH MARGINESOWYCH NA DANYCH POZIOMACH CZYNNIKÓW

STRESZCZENIE

Dana jest tablica wielodzielcza, zawierająca dane wraz ze wszystkimi średnim marginesowymi. Tablica taka może być obliczona np. za pomocą procedury *means* [1]. Procedura *searchmeans* umożliwia wyszukiwanie w tej tablicy najpierw wszystkich średnich marginesowych pojedynczych, potem wszystkich średnich marginesowych podwójnych itd., kończąc na wszystkich kombinacjach po l czynników. Kombinacje tworzone są z pierwszych $n1$ czynników.

Dane:

l — górna granica długości kombinacji (obliczone są najpierw kombinacje utworzone z pojedynczych czynników, potem dla par itd.);

$n1$ — liczba czynników, z których są tworzone kombinacje; liczba ta odnosi się do czynników o poziomach określonych pierwszymi $n1$ liczbami w tablicy f;

n — liczba czynników badanych w doświadczeniu;

$f[1:n]$ — poziomy czynników;

$maxf$ — największa liczba poziomów;

$a[0:\prod_{i=1}^{n}(f[i]+1)-1]$ — tablica zawierająca dane i średnie marginesowe obliczone np. za pomocą procedury *means* [1];

combi — nazwa procedury obliczającej kolejne kombinacje po r z n czynników [2];

setup — globalna procedura nadająca wartości pomocniczym tablicom *fl*, *m*, używanym w *printmeans*;

address — globalna funkcja całkowita, znajdująca w jednowymiarowej tablicy a adres szukanej średniej marginesowej.

Wyniki otrzymuje się za pomocą procedury *printmeans* (patrz punkt 2).