Gower, J. C. 1962. The handling of multiway tables on computers. *Computer Journal.* 4 (4), pp. 280-286.

The handling of multiway tables on computers

By J. C. Gower

A notation is proposed which concisely describes many forms of table manipulation in terms of a concept called *scanning*. Methods of programming a scanning subroutine are discussed and examples are given of the use of the scanning notation to formulate typical table operations in symbolic form. It is often convenient to store marginal values (usually totals or means) with the main table, and this requires some modification of the notation and the subroutine. Proposals are outlined for inclusion of scanning facilities in automatic programming languages and examples are given of the form this could take in a language related to the Mercury Autocode compiler.

Introduction

Little attention has been given to the systematic handling of multiway tables on computers. In recent contributions to this journal Yates and Simpson (1960 and 1961) have discussed the table manipulations required in their general program for the analysis of surveys. Lieblein (1959) has given methods for the analysis of variance of a general factorial system, and Hartley (1956) has outlined a program for providing the analysis of a general class of experimental designs. These are all particular applications, and in this paper the emphasis is on the underlying counting processes needed to extract tabular values from multiway tables in order to deal systematically with these values in particular problems.

Notation for Multiway Tables

In the terminology of factorial experiments, an n-way table has *factors* 1, 2, . . ., n and each factor can assume a number of values called its *levels*. The ith factor has L_i levels $l_i = 0, 1, 2, \ldots, (L_i - 1)$. The table may be stored so that the content $x(l_1, l_2, \ldots, l_n)$ of the cell with levels $l_1, l_2, \ldots, l_n$ is stored in an address

$$s(l_1, l_2, \ldots, l_n) = A + l_1 + l_2L_1 + l_3L_1L_2 + \ldots + l_nL_1L_2 \ldots L_{n-1} \quad (1)$$

where A is the address of the first cell of the table. An address and its contents may be referred to as s and x_s (or x), respectively, when no ambiguity arises.

Scanning over a set of factors $i, j, k, \ldots$, etc., is defined as a process that generates the ordered set of addresses $\boldsymbol{s}$ in which $l_i, l_j, l_k, \ldots$ take in turn all the $L_iL_jL_k \ldots$ combinations of their possible values, the levels of all other factors remaining unchanged at preassigned levels, i.e. the factors take in turn the values 0, 0, 0, . . .; 1, 0, 0, . . .; . . . $L_i - 1$, 0, 0, . . .; 0, 1, 0, . . .; 1, 1, 0, . . .; etc. Thus the addresses are produced in ascending order of s.

The scanning process will be written symbolically as

$$\boldsymbol{S}[i, j, k, \ldots : H] \quad (2)$$

which is read as "Scan over the factors $i, j, k, \ldots$" The set of addresses $\boldsymbol{s}$ produced by a scan will define a set of tabular values $\boldsymbol{x}_s$. Normally, in table manipulations, some arithmetical or organizational *operation* H will be required on these values. For example $\boldsymbol{S}[3: \Sigma_s x_s]$ means "scan over factor 3 and sum the contents of the addresses so defined."

Often two or more scans are required, one scan being part of the operation of the other. Written symbolically this becomes

$$\boldsymbol{S}[l, m, n, \ldots : K_1\boldsymbol{S}[i, j, k, \ldots : H]K_2] \quad (3)$$

In (3) the inner scan over $i, j, k, \ldots$ will be referred to as the *minor scan* and the outer scan over $l, m, n, \ldots$ as the *major scan*; the corresponding operations are the *minor* and *major operations*. The minor scan and its operation H form part of the major operation, the remainder of the major operation being made up of a part K_1 preceding the minor scan, and a part K_2 following it. In practice, circumstances rarely, if ever, arise where there are any factors common to the major and minor scans, and in what follows it is assumed that major and minor scans are always over disjoint sets of factors. Frequently $l, m, n, \ldots$ and $i, j, k, \ldots$ will account for all the n factors, and this leads to the convenient notation $\overline{i}$ read as "not i" so that (3) would be written

$$\boldsymbol{S}[\overline{i, j, k, \ldots} : K_1\boldsymbol{S}[i, j, k, \ldots : H]K_2] \quad (4)$$

which means that the major scan is over all those factors not occurring in the minor scan.

If in one of the scans a scan over all factors is called for then the other is null; that is, no change occurs in the scan count.

At the beginning of all scanning operations, scanned factors are assumed to be at zero level and on completion of any scan (major or minor) the levels of the factors occurring in that scan are to be reset to zero.

Rescue procedures may require a scan to be re-entered at the levels reached for the last available values of any variables being calculated. This will cause no special difficulty.

Scan operations are often repetitive, the same manipulation being required for different factors of the table. The symbol $\boldsymbol{R}$ will be used to denote repetitive scans so that we have situations such as

$$\underset{i}{\boldsymbol{R}}\boldsymbol{S}[\overline{i} : K_1\boldsymbol{S}[i : H]K_2]. \quad (5)$$

This defines a minor scan over the ith factor and a major scan over all other factors, the whole being repeated n times with i set to 1, 2, . . . n in turn.

Table 1

A Scan Subroutine based on a Packed Count

FUNCTION	EXPLANATION
Plant Operations Link	
Plant Exit Link	
$R_4 = R_2$ and R_1	Part of count specified by $\Pi \rightarrow R_4$
$R_1 = R_4 \not\equiv R_1$	Part of count not specified by $\Pi \rightarrow R_1$
$R_4 = (R_2 \not\equiv (-1)) \not\equiv R_4 + 1$	Count increased by one in least significant digit
$R_1 = (R_4 \not\equiv R_2) + R_1$	Unjustified count $\rightarrow R_1$
$R_4 = (R_4 \not\equiv (-1))$ and M	R_4 contains a 1 in the carry digit of every factor which has carried. The count is actually zero (and not $16 - L_i$)
$R_5 = (2^{-4}R_4)$	Expanded form of $R_4 \rightarrow R_5$
$R_5 = R_4 - R_5$	
$R_1 = (R_5$ and R_2 and $Z) \not\equiv R_1$	Count in R_1 justified
If $[(R_1 - Z) = 0]$ go to Exit Link.	The count is complete and the levels of all factors in π
Otherwise convert $(R_1 - Z)$ to address form	have been reset to zero, i.e. $(16 - L_i)$
Exit to Operations Link	

Repetitive scans over sets of factors are also required. Factor sets are most easily expressed in binary notation when working with computers. Thus 1101 represents factors 1, 3 and 4. In all there are $2^n - 1$ factor combinations or 2^n combinations if the null set is included. Factor sets will sometimes be represented in bold type (e.g. $\boldsymbol{i}$). Repetitive scans over all possible factor sets will be written as $\boldsymbol{RS}[\boldsymbol{i} : H]$ or if the null set is to be included $\boldsymbol{i}$ $\boldsymbol{R_0S}[\boldsymbol{i} : H]$. The factor sets $\boldsymbol{i}$ are to be produced in $\boldsymbol{i}$ ascending order of magnitude when interpreted as binary numbers.

Programming Details

It is important that scanning should be made as fast as possible. With crude programs an inordinate amount of time can be spent in picking up the tabular values, so that calculations spend more time in scanning than in computing operations. In this section two methods of programming are discussed which will be suitable for different circumstances or with different computers.

The basic requirements of a scan subroutine are (i) a multiple count over all specified factors and (ii) means of converting this count into an address. Provision must be made for two links, one for entry to the operation and the other for the final exit when all addresses required for the scan have been produced. At this point all factors which have been scanned must be reset to their zero levels. Because the factor sets for major and minor scan are disjoint it is possible to use the same orders for both scans without the counts interfering with each other. The factors occurring in a particular scan are specified by the digits of a parameter π: thus $\pi = 1011$ means that scanning occurs over factors 1, 2 and 4, but not 3 or any other possible factor.

A straightforward scan subroutine may be written by storing the levels of each factor in separate locations. Multiplication may be avoided in conversion of the count by storing appropriate multiples of the levels instead of the actual levels. Thus for factor i the multiple is $L_1L_2 \ldots L_{i-1}$. The count for factor i is then increased by $L_1L_2 \ldots L_{i-1}$ for each unit increase of l_i until it becomes $L_1L_2 \ldots L_i$ when it is reset to zero and unit increase is arranged for the next factor level specified by π. Thus l_i is stored in effect as $l_iL_1L_2 \ldots L_{i-1}$. The address is then obtained by adding the contents of all the separate locations. Alternatively a running count of the value of the address can be kept, with a step back whenever there is carry between factors. A subroutine of this type is particularly easy to set up and has virtually no restrictions on the number of levels allowed to each factor, but is slow on machines with inadequate immediate-access storage.

In many types of problem, when the number of factor levels is not excessive, the counts for the different factors can conveniently be packed into one computer word. The following method of programming, based on a packed count, requires few orders and gives fast operation.

Suppose for definiteness each factor has five bits assigned to it. The count is arranged so that the first factor has the least significant five bits, the second factor the next five bits, and so on reading from right to left of a word. The first four bits associated with each factor indicate the current factor level l_i, stored as $16 - L_i + l_i$ Carry into the fifth bit position will then operate when this factor level exceeds its maximum and reaches L_i; the level indicator of this factor is now zero and the program must be arranged to reset it to $16 - L_i$. The fifth digit is not part of the count but is used as a

signal for carry propagation and is referred to as a *carry digit*.* Two special words are also required:

Z (zero) has $16 - L_i$ in the four count digits for each factor and zero in the carry digit positions;

M (carry mask) has a one in every carry digit position and zero elsewhere.

The counting subroutine uses Π in an expanded form: $\Pi = 01111\ 00000\ 01111\ 01111$ if $\Pi = 1011$ for example. On entry to the subroutine a register R_1 contains the count, the component parts, which are being scanned over, being initially set to zero levels, and R_2 contains the expanded parameter Π. Further registers R_3, R_4 and R_5 are required as working space. The subroutine is given in Table 1, where (-1) indicates a word all of whose bits are 1's.

In this subroutine the chief loss of time will occur in the conversion of $(R_1 - Z)$ to address form. Different methods are suitable for different computers. If the computer is equipped with multiple-radix working the operation is a simple conversion from a multiple-radix number to binary and will cause no difficulty. On the Elliott 401 at Rothamsted Mr. D. H. Rees has added a "sequential addition" order which adds a selection from a list of numbers stored in sequence; the selection is determined by the digits of a control word (1 for addition; 0 for omission). If a *scan list* of the numbers 1, 2, 4, 8, —, L_1, $2L_1$, $4L_1$, $8L_1$, —, L_1L_2, $2L_1L_2$, $4L_1L_2$, $8L_1L_2$, —, etc., is set up, the control word $(R_1 - Z)$ will give the address directly.

Each address given by the above subroutine is after an increase in count. The operation must therefore precede entry to scan, the initial address being that for zero levels of all scanned factors. Alternatively the subroutine may be written in the sequence: address evaluation—operation—increase of count and link detection.

The scan subroutine must be set up before use; for instance, the zero word Z must be constructed, and for conversion the necessary multiple radix indicator or its equivalent must be provided. This can often be done once for all at the outset of a program. If simultaneous reference to two or more tables containing different factors is required, two or more scan lists and Z's must be stored and provision made for using the correct Z and scan count for each table.

These operations have been described in terms of a fixed four bits (excluding the carry digits) for each factor count, but this can clearly be varied. In fact each factor may have a different number of digits assigned to it; this causes slight complications in forming Z and the expanded versions of π and the carry pattern, but these are easily dealt with when multiple radix or equivalent operations are available. For illustrative purposes a packed scan count with four bits for each factor is assumed in what follows.

* Miss Bland of Harwell has programmed a similar scan subroutine without recourse to carry digits: carry is detected by comparing the value of the count before and after carry propagation. This takes a few more orders on the Elliott 401. A further device used by Miss Bland, useful when the computer has a very large store and when few factors are involved, is to define the address by the value of the count and so dispense with the final conversion to binary.

Examples of Operations on a Single Table without Marginal Values

Example (i) *Calculation and Printing of Lists of Marginal Totals*

$$\underset{i}{\boldsymbol{RS}}[i : \boldsymbol{S}[\bar{\imath} : \Sigma_s x_s]\ \text{Print}\ \Sigma x_s]. \tag{6}$$

Expression (6) defines the calculation and printing of the one-way tables of marginal totals.

If all the marginal totals are required (i.e. grand total, one-way, two-way and higher order tables of totals) (6) becomes

$$\boldsymbol{R}_0\underset{\boldsymbol{i}}{\boldsymbol{S}}[\boldsymbol{i} : \boldsymbol{S}[\bar{\boldsymbol{\imath}} : \Sigma_s x_s]\ \text{Print}\ \Sigma x_s]. \tag{7}$$

For a good tabular layout additional instructions have to be included to print table headings and to allow L_i tabular entries per line. In (7) no attempt is made to save computer time by deriving 3-way tables from 4-way tables and so on. This is done by expression (12) below.

Example (ii) *Analysis of Variance*

If in (7) $i, j, k \ldots$ are the factors in the current specification of $\boldsymbol{R}$ then each total Σx_s printed is the sum of the same number $N(i, j, k \ldots)$ of tabular entries. The major operation may be extended to calculate quantities

$$Q(i, j, k \ldots) = \frac{\Sigma(\Sigma x_s)^2}{N(i, j, k \ldots)}.$$

These are known as *crude sums of squares* and may be stored in a sequence of stores modified by the unexpanded scan parameter π. The $Q(i, j, k \ldots)$ thus form a 2^n table derived from the original table and, to conform with our earlier notation, the entries in the table will be written as $y(m_1, m_2, m_3, \ldots, m_n)$ where $m_r = 1$ if r is a factor occurring in $S(i, j, k \ldots)$, and $m_r = 0$ otherwise. The problem of the analysis of variance is to calculate *corrected sums of squares* $Q'(i, j, k \ldots)$ which are defined by formulae of the type

$$\begin{aligned} Q'(i, j, k) &= Q(i-1)(j-1)(k-1) \\ &= Q(i, j, k) - Q(j, k) - Q(i, k) \\ &\quad - Q(i, j) - Q(i) - Q(j) - Q(k) - Q(0). \end{aligned}$$

$Q(0)$ is sometimes known as the *correction for the mean* and is the grand total squared divided by the total number of cells in the original table.

The corrected sums of squares are given by

$$\underset{i}{\boldsymbol{RS}}[\bar{\boldsymbol{\imath}} : \boldsymbol{S}[i : y_t = y_t - y_s]. \tag{8}$$

In each minor scan of (8) two successive addresses s and t of the scan count are available and the specified operation overwrites the second value y_t by the amount $(y_t - y_s)$ by which it exceeds the first value y_s.

It may be noted that the scanning system of formula (8) can be used with slight modification to form the effects of a 2^n factorial experiment (for definition see, for example, Cochran and Cox, 1957). In this case $Q(i, j, k, \ldots)$ must be the treatment mean over the ith, jth, kth, . . . factors of the original table, and the minor operation is

$$\left.\begin{aligned} y_s &= y_t + y_s \\ y_t &= y_t - y_s \end{aligned}\right\} \qquad (9)$$

This is equivalent to the well-known Yates plus-minus technique, but is better for computer work since the derived values overwrite the old ones, and thus only two additional words of storage space are required instead of 2^n words.

Example (iii) *Fitting a Multidimensional Surface by Orthogonal Polynomials*

This problem has recently been discussed in this *Journal* (Cadwell, 1961) for the bivariate case. The methods given can clearly be directly extended to more dimensions. The values to which the surface is to be fitted are assumed to form an n-way table, factors 1, 2, 3, . . . corresponding to the rectangular co-ordinate system $x, y, z, \ldots$

The required scanning is

$$\underset{i}{\boldsymbol{RS}}[\bar{\imath}: \boldsymbol{S}[i : \boldsymbol{y} = \boldsymbol{x}_s]\boldsymbol{z} = p(\boldsymbol{y})\boldsymbol{S}[i : \boldsymbol{x}_s = \boldsymbol{z}]]. \qquad (10)$$

Temporary storage space $\boldsymbol{y}$ (i.e. $y_1, y_2, \ldots, y_{L_i}$) is used to store the values $\boldsymbol{x}_s$ obtained by the first minor scan. These values are then operated on to obtain the coefficients of the fitted polynomials of the required degrees—this is represented by $\boldsymbol{z} = p(\boldsymbol{y})$ (see Clenshaw, 1960). These coefficients are then written back into the original table by the second minor scan. If the degree d of the polynomial associated with factor i is less than $(L_i - 1)$ this scheme is inefficient, so a method for reducing the dimensions of the table should be incorporated in the program at each change of i. To do this, factor i must be restricted to its first $d + 1$ levels; this is easily arranged by adjusting the Z word.

It is interesting to note that (10) is of the same operational form as is required to form the effects of an $L_1 \times L_2 \times L_3 \ldots$ factorial experiment. The operation $\boldsymbol{z} = p(\boldsymbol{y})$ has to be replaced by $\boldsymbol{z} = \boldsymbol{L}_i\boldsymbol{y}$, where $\boldsymbol{L}_i$ is an orthogonal matrix of order L_i.

Example (iv) *Deletion of a Factor Level*

The level l of factor i is to be deleted from a table, the resulting reduced table being written in consecutive store locations, possibly overwriting the original table. To do this, successive values of the table are picked up by scanning over all factors and written back into successive store locations except when $l_i = l$.

Deletion is similar to the problem of amalgamation of factor levels. In both cases the problem is much simplified if one is content to replace deleted levels by marker patterns in the table, so that the same scan can be used on the reduced table.

These and related problems require provisions for identifying and setting the levels of specified factors in the scan subroutine, and cannot be conveniently represented by the symbolism given above, but can be dealt with by the autocode instructions set out below.

Tables including Margins

In many situations the marginal means or totals of multiway tables play a central part in the calculations. It is then desirable to store these means together with the table (see, for instance, Yates and Simpson, 1961). This is simply arranged by assigning an extra level to each factor, for the margins. Thus the address of the cell with levels $l_1, l_2, \ldots, l_n$ becomes

$$\begin{aligned} s'(l_1, l_2, \ldots l_n) = A &+ l_1 + l_2(L_1 + 1) \\ &+ l_3(L_1 + 1)(L_2 + 1) + \ldots \\ &+ l_n(L_1 + 1)(L_2 + 1) \ldots (L_{n-1} + 1). \end{aligned} \qquad (11)$$

The level L_i of l_i defines the store locations of the marginal values of factor i. Alternatively $l_i = 0$ can be used to define the margins for factor i, but the definition adopted here simplifies the printing when the marginal values are required to be printed in their conventional places as the final rows, columns, etc., of the basic table. The multiple-radix conversion requires modification by writing $(L_i + 1)$ in place of L_i in the scan list or its equivalent.

Operations on tables stored in this way require two types of count depending on whether the marginal values are included or not. These will be termed *inclusive* and *exclusive* counts respectively, factors over which an exclusive count is to be taken being denoted by primes (e.g. i'). For exclusive scans the setting of the zero word Z in the scan subroutine must be altered, the zero levels of every factor with an exclusive count being increased by one unit (i.e. $15 - L_i$ for the ith factor).

Example (v) *Evaluation and storage of all the marginal totals of a table*

If a table is stored as in (11) but without its marginal totals, these may be calculated and stored as follows:

$$\boldsymbol{RS}[\bar{\imath}: \boldsymbol{S}[i' : \Sigma_s x_s] \text{ margin } (i) = \Sigma x_s]. \qquad (12)$$

To obtain the address of margin (i) in the major operation, l_i has to be set to L_i, with all other factors at their current values. This is easily done by setting the i'th component $(Z_i + l_i)$ of the count to 15 and converting.

Operations involving more than one Table

We shall restrict ourselves in this section to operations on two tables $A_1(x)$ and $A_2(y)$ to provide a third table $A_3(z)$. If $\boldsymbol{a}$ represents a set of factors common to A_1 and A_2, whilst $\boldsymbol{b}$ is a set present in A_1 but not A_2, and $\boldsymbol{c}$ is present in A_2 but not A_1, then A_3 may be classified by the set of factors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$. Further sets of factors $\boldsymbol{u}$ and $\boldsymbol{v}$ may occur in A_1 and A_2, respectively, which do not directly enter the calculations. These must be set to preassigned levels; usually marginal totals (or means)

are required, denoted by $\boldsymbol{U}$ and $\boldsymbol{V}$ below. A very general operation on two tables may then be written

$$F(x_{\boldsymbol{a},\boldsymbol{b},\boldsymbol{U}}, y_{\boldsymbol{a},\boldsymbol{c},\boldsymbol{V}}) = z_{\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}}. \qquad (13)$$

This is to hold for all levels of factors in $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$.

Expression (13) can be computed by using a separate scan count for each table, but one count with three scan lists, or their equivalent, will suffice if the factors are in the same order in each table. This requires that the same factors occur in the same positions in all three scan lists, with the proviso that the parts of the list referring to the non-existent factors ($\boldsymbol{c}$ in A_1 and $\boldsymbol{b}$ in A_2) are set to zero (i.e. the radices are zero). In this way we have a *simultaneous scan* which produces three addresses s_1, s_2, s_3, one for each table. Simultaneous scans will be written $\boldsymbol{S}_{1,2,3}[\boldsymbol{i} : H]$.

Example (vi) *General form of operations in Yates's and Simpson's general survey program*

Yates and Simpson (1961) restrict themselves to the case in which either $\boldsymbol{b}$ or $\boldsymbol{c}$ or both are zero. If $\boldsymbol{d} = \boldsymbol{b} + \boldsymbol{c}$ the set of additional factors, if any, in either A_1 or A_2 and if the factors $\boldsymbol{u}$, $\boldsymbol{v}$ are set for margins (13) becomes

$$\boldsymbol{S}_{1,2,3}[\boldsymbol{a} : \boldsymbol{S}_{1,2,3}[\boldsymbol{d} : z_{s_3} = F(x_{s_1}, y_{s_2})]]. \qquad (14)$$

Automatic Programming

Table manipulation could be very greatly simplified if facilities for scanning were provided in automatic programming languages. An illustration of this is given below in the form of suggested additions to the Autocode Programs of the Manchester University/Ferranti computers. The only point about these autocodes that need be stressed here is that quantities are of two kinds—variables and indices. Variables are in floating-point form and indices are stored in fixed-point form as integers.

Variables are denoted by the letters *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, *u*, *v*, *w*, *x*, *y*, *z*;

Indices are denoted by the letters *i*, *j*, *k*, *l*, *m*, *n*, *o*, *p*, *q*, *r*, *s*, *t*.

The symbols available for variables are increased by allowing variables to have an index as suffix.

A description of this Autocode has been given by Brooker (1958*a*, 1958*b*, 1959).

To do all the operations required, provision must be made for inclusive and exclusive scans and also for simultaneous scans over three different tables. More than three scans are unnecessary, as operations on any number of tables can practically always be broken down into a set of operations on a pair of tables producing a third table. In what follows a maximum of three scans is provided for.

The following additional Autocode instructions are required:

(i) Set up (n, f_1) — Set up the scan subroutine for the number of factors given in store n, the levels of which are stored in $f_1, f_2, \ldots$, etc. Here n may be an index or an actual integer.

(ii) Set up $(p, q, r, \ldots)$ — This is an alternative to (i), giving the actual numbers of factor levels or the names of the stores containing these levels.

(iii) Set up margin (n, f_1) / Set up margin $(p, q, r, \ldots)$ — These are complementary to orders (i) and (ii). The scan count is set to evaluate the address in s according to formula (11); i.e. it is set to work on tables with marginal values.

In the following orders, scans occur over sets of factors. A single factor can be nominated by an index or an integer, but sets of factors must be nominated in scan parameter form. If the latter system is used, a tag must be set to distinguish the scan parameter from an ordinary index or integer.

(iv) $i = (p, q, r, \ldots)$ — i is set as a scan parameter with tag for the factor numbers given by $p, q, r, \ldots$ which may be indices or integers. i = "tag" + $2^{p-1} + 2^{q-1} + 2^{r-1} + \ldots$ If i (say) is already in scan parameter form, as would be likely if all factor combinations were being scanned in turn, then the tag must be added to p before using it in any of the scan orders.

(v) "scan (i) … "rescan / "scan not (i) … "rescan — These two sets of instructions are parallel to the [$i = p(q)r$, repeat] sequence already existing in Autocode, and correspond respectively to $\boldsymbol{S}[i : \ldots]$ and $\boldsymbol{S}[\bar{\boldsymbol{\imath}} : \ldots]$ of the previous sections, where, however, i may now represent a set of factors preset by instruction (iv). When "scan" is encountered an address is evaluated and stored in store s; when "rescan" is encountered the scan count is increased and the correct link determined—this is the next order when the current scan is complete and is otherwise the scan order of the preceding uncompleted scan.

(vi) address set $(i = \alpha)$ — This replaces the level l_i of factor i by α; α may be an index, a variable or an integer, and i may be an index or an integer. After the replacement the new address given by the scan count is to be calculated and placed in store s. If i has been set up by (iv) and involves more than one factor (say, 1, 3 and 4) the corresponding

levels will be set if α is an index or constant ($l_1 = l_3 = l_4 = \alpha$), but when α is a variable f (say) then $l_1 = f_1$, $l_2 = f_2$, $l_4 = f_4$.

(vii) Address ($i = \alpha$) — This is similar to (vi) but replacement does not occur; the address corresponding to α is placed in s.

(viii) Exclude (i) — Placed before any "scan" order this gives exclusive counts over the factors given in i. At the end of such a scan sequence i is automatically reset for inclusive counts.

(ix) .....1 (...) — Scans may be numbered with a 1, 2 or 3 immediately preceding the brackets in any of the previous orders, when more than one table is involved. The addresses of scan 1 are given in r, scan 2 in s, and scan 3 in t. This implies that any unnumbered scan is interpreted as scan 2. This could be avoided if Autocode allowed suffices to indices so that s_1, s_2, s_3 would then be convenient for the addresses.

(x) Scan 1, 2, 3 (i, j, k) — A simultaneous scan over factors given in i for scan 1, j for scan 2, and k for scan 3.

Instructions (i) and (ii) are not both necessary, and the two instructions (iii) are redundant as the f_i can always be set to $L_i + 1$ and instruction (i) used. Nevertheless, it is often convenient to accept some redundancy.

Examples

The following examples illustrate the use of these instructions. Tables are assumed to be stored in x_0 onwards, and if more than one table is involved these are in y_0, and z_0 onwards.

The following program generates the effects of a 2^n factorial experiment, as explained in Example (ii) (p. 282), treatment means being stored in x_0 onwards, and the n factor levels, all set equal to 2, being stored in f_1 onwards.

```
set up (n, f1)
--> i = 1(1)n
--> scan not (i)
    t = s
    a = 0
    b = 0
  ->scan (i)
    a = xs + a
    b = xs - b
  -rescan
    xs = b
    xt = a
  ---rescan
-------repeat
```

The formation of corrected sums of squares is even simpler, all references to a and t are deleted. These calculations are both quite lengthy if scanning operations are not available. Examples (i) and (iii) (pp. 282 and 283) can be similarly organized; the existing Autocode deals adequately with their more complex operations. A program for the formation of 2^n effects with a much shorter major operation and no minor scan may be written, if advantage is taken of the fact that there are only two levels 0 and 1 for each factor:

```
set up (n, f1)
--> i = 1(1)n
  ->scan not (i)
    t = s
    address (i = 1)
    xs = xs - xt
    xt = xs + 2xt
  -rescan
-----repeat
```

The operation of deleting a level l of factor i and rewriting the reduced table on top of itself (Example (iv), p. 283) is effected by the following Autocode instructions:

```
set up (n, f1)
q = 0
p = 0
--> scan not (q)
    t = s
    address (i = l)
    jump 1, s = t----
    xp = xs          |
    p = p + 1        |
-1) rescan <---------|
    fi = fi - 1
```

When $q = 0$, "scan not (q)" implies a scan over all factors. The final $f_i = f_i - 1$ is merely to adjust for one level less in factor i in any subsequent calculations.

If a marker (say -1) can be used for the deleted level, the whole operation is much more simple:

```
set up (n, f1)
address set (i = l)
-> scan not (i)
   xs = -1
-- rescan
```

To calculate and store the marginal totals of a table starting in x_0 as in Example (v) (p. 283), the following instructions are required:

```
set up margin (n, f1)
--> i = 1 (1) n
  ->scan not (i)
    a = 0
    exclude (i)
  ->scan (i)
    a = a + xs
  -rescan
    address (i = f)
    xs = a
  ---rescan
-------repeat
```

With a well-designed compiler, the operation of these Autocode programs should be nearly as fast as the equivalent programs written in machine code.

Conclusion

Scan subroutines have been in use at Rothamsted for over four years, in a variety of statistical programs. They have been a powerful aid to program writing and are particularly suitable for general programs, such as those for the analysis of surveys and experiments, as it is then easy to cope with changes in the numbers of factors and levels. Scanning is, however, also useful for programs written for tables of fixed size. With the packed-count form of scan subroutine suitable for the Elliott 401 at Rothamsted, the speed of operation is very fast and, even for programs operating only on two-way tables, is as fast as equivalent programs not using a scan. For three- or more way tables scanning is very much more convenient.

The Autocode proposed here has not been tried out on a computer, but it would most certainly ease programming considerably without appreciably affecting the time factor.

Acknowledgement

I should like to thank Dr. F. Yates for many suggestions and useful criticism during the course of this work.

References

BROOKER, R. A. (1958*a*). "The Autocode Programs developed for the Manchester University Computers," *The Computer Journal*, Vol. 1, p. 15.

BROOKER, R. A. (1958*b*). "Further Autocode Facilities for the Manchester (Mercury) Computer," *The Computer Journal*, Vol. 1, p. 124.

BROOKER, R. A. (1959). "Mercury Autocode (additional notes)," *The Computer Journal*, Vol. 2, p. xi.

CADWELL, J. H. (1961). "A Least Squares Surface Fitting Program," *The Computer Journal*, Vol. 3, p. 266.

CLENSHAW, C. W. (1960). "Curve Fitting with a Digital Computer," *The Computer Journal*, Vol. 2, p. 170.

COCHRAN, W. G., and COX, G. M. (1957). *Experimental Designs* (2nd edition), New York: John Wiley & Sons.

HARTLEY, H. O. (1956). "A Plan for Programming Analyses of Variance for General Purpose Computers," *Biometrics*, Vol. 12, p. 110.

LIEBLEIN, J. (1959). "A General Analysis of Variance Scheme Applicable to a Computer with a very large Memory," *J.A.C.M.*, Vol. 6, p. 469.

YATES, F., and SIMPSON, H. R. (1960). "A General Program for the Analysis of Surveys," *The Computer Journal*, Vol. 3, p. 136.

YATES, F., and SIMPSON, H. R. (1961). "The Analysis of Surveys: Processing and Printing the Basic Tables," *The Computer Journal*, Vol. 4, p. 20.

Third National Conference
Cardiff – September 4th to September 7th 1962

The next National Conference of The British Computer Society will be held in Cardiff during September 1962 immediately following the Congress of the International Federation for Information Processing in Munich. The Conference will begin on the evening of Tuesday, September 4th, and finish on Friday, September 7th.

A full programme of sessions is being planned to cover many aspects of data processing. It is hoped to give particular attention to actual experience in operating commercial data processing systems, and to cover scientific and technical applications.

A number of social functions are being arranged and if there is sufficient demand visits will be arranged to local places of interest.

Call for Papers

Members of the Society and others are invited to submit details of papers which they are prepared to offer. They can be concerned with any aspects of Data Processing and Digital Computers including the following:—

Business Data Processing—
in commerce and industry,
integration of procedures,
has it profitability?
problem of conversion from one system or machine to another.

Scientific and Engineering Computation—
numerical analysis,
statistics and applied mathematics.

Operational Research—
scientific management,
optimizing routines,
simulation techniques.

Programming—
business and scientific autocodes;
novel techniques.

Selection and training of computer staff

Development of equipment

Automatic Origination of Data

Anyone wishing to offer a paper should send two copies of a synopsis of 300–500 words to:

G. M. Davis,
English Electric Company Ltd.,
Data Processing and Control Systems Division,
Queens House,
Kingsway, London W.C.2,
England

not later than **1st April 1962.**