ALGOL survey program

Valerie G. Coulson†, I. D. Hill† and Valerie F. Hillier*

ALGOL survey program (Asp) is one of many programs now in existence for survey analysis. It differs from most of the others in that the user's requirements are specified as part of the program, not as part of the data. This has the advantage of giving the user the power of a compiler to assist him, but the disadvantages that the user must know something of programming, and that the program has to be re-compiled for every new set of requirements.

(Received November 1971; Revised March 1972)

The Department of Social and Preventive Medicine at Manchester University has, for many years, been involved in survey work, offering advice and assistance to research workers wishing to process their survey data. Such workers often collect large quantities of complex data, often coded in a complex fashion, which would be best processed on a computer, but lack the time and programming knowledge necessary to undertake the task of writing a complete program themselves. Furthermore if they were to do it themselves it would be largely wasted effort since the bulk of the programming can be done once and for all. A program has been available (Department of Social and Preventive Medicine, 1969) for some time, which could most easily be described as being an enriched version of the programming language Atlas Autocode (Brooker, Rohl and Clark, 1966), in that all the facilities of Atlas Autocode are available as well as the additional facilities for survey analysis. This program was written in a mixture of Atlas Autocode and Atlas Machine code and as such was, at the time, available only on the Atlas computer (though a version is now available on the ICL 1906 computer). It was felt that an ALGOL system, incorporating similar facilities, might have a wider use and might be more machine independent, or at least easier to transfer between machines.

Desirable features

Desirable features for any system for tabulating survey data are:

1. Data checking facilities so that data incorrectly coded, or recorded, may be rejected from the analysis and listed, in order that spurious results are not generated;
2. Facilities for producing both frequency tables and tables that have as their entry item the value of a variable, or any function of variables;
3. A variety of printing modes so that the work of calculating percentages, or the χ^2 statistic, by hand does not have to be done after the computer output has been received;
4. The capacity to deal easily with tables of more than two dimensions;
5. The ability to store data for use on more than one occasion without having to read cards many times, and also to store tables for subsequent updating;
6. Automatic layout of tables, complete with automatic splitting whenever a table is too wide for the page;
7. Facilities for naming variables, and categories within variables, so that the tabular output is directly meaningful to the user.

Types of survey analysis program

Survey analysis programs in general fall into three main groups:

1. Programs such as the BMD series (Dixon, 1967) which are self-contained, where the user specifies requirements by means of special data cards which precede the survey data. The main disadvantage of this type of system is that the specification of requirements is in such a heavily coded form as not to be at all meaningful without constant reference to a manual.

   A further disadvantage of BMD in particular seems to be that it might be necessary to use several programs of the series to perform an analysis. However the programs are compatible in that the way of specifying requirements remains the same throughout the series;
2. Special computer languages designed solely for survey, or other statistical, analysis, such as MVC (Colin, 1964; Singh, 1968) or ASCOP (Cooper, 1967; 1969). These accept instructions in a fairly English-like form, but require the user to learn a special language (which is not usable for more general purposes), and may not be easily transferable from one computer to another. This is specially true of MVC with its own compiler, as distinct from ASCOP, that depends on an interpreter written in FORTRAN;
3. Systems that are an enrichment of an already available, general purpose, language. The main advantage of such a system is that it provides all the facilities of the general purpose language, if required, yet the more difficult part of the programming task has been done. Transfer between computers is also relatively easy.

   The user is required to learn some programming, but only relatively simple operations are usually needed, and the language may be useful to know for other purposes.

Asp

ALGOL survey program (Asp) is of this third type. It does require the user to write some ALGOL, but while the full power of ALGOL is available for those capable of using it, successful analyses may be made with a much more limited repertoire, merely working from examples rather than from an ALGOL textbook.

Whereas the 'Special Compiler' facility of Atlas allowed the Atlas Autocode program to be treated as if it were itself a compiler, a rather different approach has been used for the ALGOL version, so as to be independent of differences between operating systems.

This approach consists of a fixed ALGOL program, in which four procedures are missing. The user specifies his requirements by writing these four procedures, which are then combined with the fixed program before compilation.

The advantages of this approach have to be paid for because the program has to be re-compiled for every job, since the

†Medical Research Council, Computer Unit, 242 Pentonville Road, London N1 9LB.
*Department of Social and Preventive Medicine, University of Manchester, York Place, Manchester M13 0JJ.

ALGOL program is not complete until the user's procedures have been supplied.

So as to avoid having to compile, every time, too many facilities that will not be used, three versions of the program have been devised. These are called Miniasp, Midiasp and Maxiasp.

Miniasp

The Miniasp program contains facilities for reading data from cards, checking for faults and inconsistencies in the data, forming frequency tables in any number of dimensions, and printing these tables either as frequencies or as percentages (if percentages are requested, the frequencies are printed also unless the user deliberately suppresses them).

Midiasp

The Midiasp program contains all the facilities of Miniasp, but additionally allows frequency tables to be printed as a χ value for each cell together with a total χ^2; and also allows the construction and printing of mean tables, standard deviation tables, and expression tables where the means, standard deviations, and expressions may be formed from any functions of any of the variables, tabulated by any of the variables.

A feature of the output of a table of means and standard deviations is that each cell of the table contains the three values—frequency, mean and standard deviation (see **Fig. 1**). This is much handier, for eventual use of such a table, than three separate tables: one for the frequencies, one for the means and one for the standard deviations.

Maxiasp

The Maxiasp program contains all the facilities of Midiasp, with the addition of facilities to allow data, or tables, to be stored on magnetic tape (or disc) for re-use in a further run.

EXAMPLE
FREQUENCY TABLE NUMBER 1
MEAN TABLE NUMBER 2
STANDARD DEVIATION TABLE NUMBER 3

CV[2]	CV[1] 1	2	3	4	TOTAL MEAN STAN.DEV
0	17	7	5	1	30
	4.55	3.79	5.77	5.71	4.62
	0.53	0.48	0.56	*	0.82
1	22	15	10	5	52
	6.12	4.43	6.90	6.66	5.84
	0.46	0.44	0.48	0.87	1.07
2	12	20	19	13	64
	6.29	4.73	6.96	7.41	6.23
	0.47	0.57	0.52	0.53	1.19
3	10	15	23	11	59
	4.69	3.55	5.40	6.07	4.94
	0.58	0.57	0.39	0.61	1.04
4	5	9	19	18	51
	4.87	3.63	5.68	6.12	5.39
	0.48	0.54	0.39	0.44	1.00
5	0	2	13	16	31
	*	4.91	6.31	6.90	6.52
	*	0.23	0.53	0.66	0.77
TOTAL	66	68	89	64	287
MEAN	5.44	4.17	6.12	6.60	5.61
STAN.DEV.	0.92	0.72	0.78	0.77	1.20

STANDARD DEVIATION WITHIN GROUPS = 0.51
DEGREES OF FREEDOM = 264

Fig. 1. Example of output of a table of frequencies, means and standard deviations

Programming

Asp was originally written for the ICL Atlas computer using Atlas ALGOL, with the addition of code procedures for reading individual characters and for using magnetic tapes. It has now been transferred, with little trouble, to an ICL 1903A.

There are three features of the programming that are, perhaps, sufficiently unusual to be worth explaining in some detail.

The first is the device that allows the user to specify tables of any number of dimensions, without violating the ALGOL rule that a procedure can have only a fixed number of parameters. This is done by means of a function, T, using itself as its own parameter. Thus $T(2, 0)$ indicates a one-dimensional table based on variable number 2, whereas $T(3, T(1, T(5, 0)))$ indicates a three-dimensional table based on variables 3, 1 and 5. The principle of this is similar to that given by Hill (1971).

The second feature is the use of ALGOL block structure to give 'default' versions of the procedures that the user is expected to supply. The user's procedures are declared and used within an inner block. However, procedures with the same names but with dummy bodies also exist in an outer block. If the user declares his procedures, these get used, but if a particular procedure is not required the user can simply ignore it and the rules of ALGOL automatically call for the outer block dummy version, so the program is still complete.

The third feature is that the user's procedure *tabulations*, which is used for each data card read to update the tables, is also used initially, before reading the data for the first subject, to set up arrays of pointers for later use both in forming tables and in printing them out. It might be thought that machine code would be needed for this purpose, but, in fact, it was found possible to do it entirely in ALGOL, thus preserving machine independence.

Even the size of these arrays is not known until the user's procedures have been examined, and since the arrays need to exist in the same block as the user's procedures a special device is necessary, which is based on the following idea:

```
begin integer H, J;
J := 0;
for H := −1, 0 do
  begin array A[0:J];
  if H < 0 then carry out all operations needed to determine
    the required upper bound and set J to this value, while
    avoiding all operations that actually use A
  else
    do the real job using A which will now have the required
    upper bound
  end
end
```

Categorised variables

Raw variables will be read from cards, derived variables may be formed from these, and tables will be formed from these variables. Such variables are held in an integer array called *CV*, whose upper bound is specified by the user. *CV* stands for Categorised Variables—the word *Categorised* being intended to indicate integer type. The user, therefore, refers to the variables in the form *CV*[*subscript*]. The name *CV* is not at choice.

At present, tabular output is in terms of this *CV* notation, and the marginal scales are purely numerical. It is hoped that a later version will allow the user to specify names, both for scales, and for individual categories within any scale, so as to make the output more directly intelligible.

Example

The user's four procedures are called *limits*, *readin*, *tabulations* and *printout*. None has any parameters. In a very simple case,

for which Miniasp would be adequate, they might look as follows:

```
procedure limits;
  begin
  range (1, 1, 5);
    comment this indicates that there is to be a variable called
    CV[1] whose values should not be less than 1 nor more
    than 5;
  range (2, 1, 13); range (3, 0, 4)
  end limits;
procedure readin;
  begin
  cardinput (1, 3);
    comment read in a card, and print out the number con-
    tained in columns 1, 2 and 3 in a list of serial numbers;
  CV[1] := col (6);
    comment the variable CV[1] is read from column 6 of the
    card;
  CV[2] := cols (7, 8);
  CV[3] := col (10) −1;
    comment since the ALGOL compiler is available, any
    expression may be used in defining a variable;
  if CV[3] = 8 then CV[3] := 0;
    comment various such manipulations may be made to
    reach the final values of the variables. The check that
    CV[3] lies within the range 0-4 will not be made until
    exit from this procedure;
  if CV[1] = 1 and CV[2] > 9 then inconsistency (1);
    comment an inconsistency message, quoting the parameter
    1, will be printed if the condition is true;
  end readin;
procedure tabulations;
  begin
  freqtable (1, T(2, 0), true);
    comment frequency table number 1 is to tabulate by the
    variable CV[2]. The true parameter indicates that all
    subjects are to be included;
  freqtable (2, T(1, T(3, 0)), CV[2] < 10);
    comment frequency table number 2 is to be two-dimen-
    sional, tabulating by CV[1] and CV[3]. Subjects are to be
    included in the table only if their CV[2] values are less
    than 10;
  end tabulations;
procedure printout;
  begin
  printfreq(1, 1, 'ABC', 0);
    comment frequency table number 1 is to be printed, on
    output channel number 1, with the heading ABC. The
    final parameter, 0, indicates that plain frequencies are
    required;
  printfreq (2, 1, 'DEF', 2);
    comment frequency table number 2 is to be printed, also
    on output channel number 1, with the heading DEF. The
    final parameter, 2, indicates that percentages of column
    totals are required—plain frequencies will be printed as
    well;
  end printout;
```

This example has deliberately been kept simple for illustrative purposes, but it is when complications arise that the system is seen to its greatest advantage in comparison with some other survey analysis programs.

If variables are required, which are complicated functions of the values read from the data, and perhaps conditional on other variables; or if consistency checks are to be made that call for involved Boolean constructions, then having the power of the ALGOL compiler available to do the work, means that the user can merely write down the desired expressions without much difficulty.

Timing

It is difficult to make an overall judgment of the efficiency of the program, in that we regard human time as being important as well as machine time, but more difficult to measure adequately.

When people compare the efficiency of two computing methods, they often compare merely the computer time taken on the final run, while ignoring not only the human time needed to prepare the run, but also any computer time taken in 'de-bugging' runs.

We shall follow this precedent in quantifying only the computer time of a final run, but it is essential to remember that this is not the whole story.

For purposes of comparison we took a survey job that had recently been run on the ICL 1903A using a version of the XTAB programs (Massey, 1963). It had given a 'Total Mill Time' figure of 56 seconds.

To produce the same answers using Midiasp took a 'Total Mill Time' of 157 seconds. It was found, however, that 40 seconds were taken in compiling (XTAB does not have to re-compile every time), and a further 61 seconds could be saved by removing array bound checking, rather surprisingly giving a net running time of 56 seconds as used by XTAB. In general, we think that the value of array bound checking is such that it should be included, but it is a sobering thought that it can more than double the running time.

There was one feature of the particular job that enabled Asp to show to advantage. The data cards had been punched with a varying number of records per card—anything from one record to four records in a fairly random fashion. To use XTAB a special run had been necessary first, to form a new and more regular data file. The flexibility of Midiasp is such that it was found quite easy to use the cards in their original form, even though the possibility of such a set of data cards had not been envisaged when Asp was written.

Manual

An Asp Manual, that gives all necessary details of the facilities available and how to use them, has been prepared. Copies may be obtained from the Medical Research Council, Computer Unit, on request.

References

BROOKER, R. A., ROHL, J. S., and CLARK, S. R. (1966). The main features of Atlas Autocode. *The Computer Journal*, Vol. 8, pp. 303-310.

COLIN, A. (1964). The multiple variate counter. *The Computer Journal*, Vol. 6, pp. 339-347.

COOPER, B. E. (1967). ASCOP—a Statistical Computing Procedure. *Appl. Statist.*, Vol. 16, pp. 100-110.

COOPER, B. E. (1969). *ASCOP User Manual*. National Computing Centre, Manchester.

DEPARTMENT OF SOCIAL AND PREVENTIVE MEDICINE (1969). *Survey Analysis Users' Guide*. University of Manchester.

DIXON, W. J. (Ed.) (1967). *BMD Biomedical Computer Programs*. University of California Press.

HILL, I. D. (1971). Algorithm AS39—Arrays with a Variable Number of Dimensions. *Appl. Statist.*, Vol. 20, pp. 115-117.

MASSEY, F. J. (1963). *Cross Tabulation Series (XTAB)*. University of Los Angeles.

SINGH, G. (1968). *A manual of the Multiple Variate Counter*. University of London.