THE MULTIPLE VARIATE COUNTER

by

ANDREW COLIN

10/-

UNIVERSITY OF LONDON INSTITUTE OF COMPUTER SCIENCE

THE MULTIPLE VARIATE COUNTER

by

ANDREW COLIN

From a copy preserved by Bill Williams and OCRed and re-formatted by Dik Leatherdale.

Original pagination has been preserved throughout but blank pages have been omitted. Modern fonts have been employed in the interests of readability. The contents list, index and internal references have been implemented as clickable links. **`Program and data text`** are differentiated by font from narrative.

Some limited correction of obvious errors has been performed and are shown in red, though trivial changes are not. Please report transcription errors to dik@leatherdale.net.

44 Gordon Square
London W C 1

June 1964

Contents

M. V. C. MARK 5.

Preface

The Multiple Variate Counter, MVC Mark 5, is a general purpose program for analysing the results of statistical surveys with the aid of the ATLAS computer.

MVC has been carefully designed to cope with many kinds of survey from the simplest to the most complex likely to be encountered. The difficulty of using the program is to a large extent commensurate with the complexity of the analysis required. Thus, in the case of a simple investigation, sufficient about MVC can be learned in a few hours by anyone reasonably familiar with elementary statistics and mathematics. For very complicated analysis, a little more study is needed.

Accordingly, this manual is divided into two parts. Part 1, 'Elementary MVC' deals with the primitive aspects of the system in a way which the author hopes is entirely comprehensible to a reader with no knowledge of computers. It also includes sections on Questionnaire design and data preparation. Although everything which these sections contain is 'obvious' even to the extent of being irritating to some readers, they are included because, in the author's experience, many survey analysts tend to arrange these matters in obscure and sometimes ambiguous ways, for reasons which do not stand up to questioning.

Part 2 of the manual ‘Advanced MVC', is considerably more sophisticated in outlook. Its reader must be familiar with the contents of Part 1, and will be greatly assisted by knowledge of some conventional programming language such as Mercury Autocode, Fortran or Algol. ‘Advanced MVC' is concerned mainly with topics such as Multi-dimensional tables, significance tests, and variable structure of individual case records. The user is not recommended to use the facilities described therein unless he fully understands their statistical significance. For this reason, ‘Advanced MVC' is published as a separate booklet.

This manual is incomplete in the sense that it does not describe any of the practical details of computer operation. This omission is due to the fact that at the time of writing, the general operating system for ATLAS is still somewhat fluid, and subject to change. Eventually a supplement dealing with this topic will be issued; but in the meantime, intending users must contact the author at the U.L. Institute of Computer Science before running their surveys.

The manual contains many examples of statistical tables. All the figures in these tables were made up by the author, and bear no relation to any real situation.

This manual supercedes 2 other publications:

1) “MVC Mark 3, a general survey analysis program for Mercury” (issued 1960);
2) A provisional specification of MVC Mark 5 issued in February 1963, which received only a limited circulation.

The author would like to acknowledge the invaluable help given him in this project by his programming assistants, Miss Valerie Weights and Mr. David Minkoff. Without their help, it is doubtful whether the work would ever have been completed.

The author would also like to thank Mr Keith Wolfenden and others for reading and commenting on the manuscript of this manual, and the clerical staff at the U.L. Institute of Computer Science for assisting in its production.

A. J. T. C.

April, 1964

Chapter 1

ELEMENTARY M.V.C.

1.1 Surveys, Questionnaires and Answers

The process of drawing up a list of questions, and the eliciting of answers from a number of carefully selected persons, is known as 'taking a survey'. Survey analysis is the art of processing the answers given by these people, in order to discover relevant facts about them.

In this manual we shall refer frequently to both questions and answers. The individual questions are assembled to form a questionnaire. The word 'questionnaire' has two separate meanings, whose difference is subtle but important. In one sense, as used in the sentence "I designed a Questionnaire to elicit drinking habits" the word is abstract. Conversely, when used in a sentence such as "I want another 5,000 of these questionnaires" the word is concrete, and refers to a piece of paper with questions printed upon it. Usually, the context will make the sense clear; but to avoid all ambiguity, we shall use a capital Q whenever the abstract meaning is intended.

The replies given to any one questionnaire are called a 'set of answers'. Unlike Questionnaires, sets of answers can never be generalised abstractions; they are always groups of symbols actually written on physical questionnaires.

Questionnaires can deal with any topic whatsoever, but the questions in them can nearly always be classified into three simple types, as explained below. This classification system is at the heart of the MVC system, and it is absolutely essential that it should be understood by the intending user.

The first type of question is that requiring a numerical answer. Examples are:

"What is your age?"

or:

"How many pints of beer did you drink last night?".

This type of question is called 'numerical'.

The other two types of question both require non-numerical answers. One of them is the question which has several possible answers, out of which precisely one must be indicated as true. This type is known as a 'polylog' question. Examples of polylog questions are:

```
"Were you born in        a)  Europe
                         b)  Asia
                         c)  Africa
                         d)  North America
                         e)  South or Central America
                         f)  Pacific Islands
                         g)  On the high seas?"
```

or:

```
"Which (in your opinion) is the best actress out of the following:

                         1)  Brigitte Bardot
                         2)  Gina Lollobrigida
                         3)  Diana Dors
                         4)  Sophia Loren?"
```

The last type of question is called 'binary'. A binary question is one

admitting only of the two answers 'yes' or 'no'. Examples are:

"Are you over 21?"

or:

"Do you own a dog?"

The distinction between the last two types was introduced for practical, rather than for theoretical, reasons. Clearly, it would always be possible to consider every binary question as a polylog question with only two possible answers; conversely, every polylog question could be thought of as a list of binary questions, for example:

"Were you born in Europe?

Were you born in Asia?

Etc."

Nevertheless, the three types were retained because they lead to simpler analysis, and to the reduction of redundant information.

Although the great majority of questions fall neatly into one or other of the three groups, there are certain types whose classification is not immediately obvious. Some of these are discussed below.

1) The 'grouped' numerical question, of the type:

```
"Is your weight          a)  Under 7 stone
                         b)  7 — 9 stone
                         c)  10 — 12 stone
                         d)  13 stone or more?"
```

This type must be treated as a polylog. The MVC system provides methods of retrieving the numerical values if required.

2) Questions such as:

```
"Do you read             a)  The Express
                         b)  The Mail
                         c)  The Guardian
                         d)  The Mirror
                         e)  The Worker"
```

In spite of their deceptive appearance, such questions are not true polylogs, since the respondent is not constrained to give precisely one answer. In the example above, the respondent may reply that he reads none, one, or any other number of the newspapers listed. It follows that such a question must be thought of as a list of independent binary questions, such as:

```
"Do you read the Express?"
"Do you read the Mail?"
Etc.
```

3) Questions in which the form of the answer is not specified at all. For example:

"What do you think is the chief cause of road accidents?"

The answers to this type of question are not directly susceptible to analysis by the MVC system, but must be edited beforehand. This editing process must be carried out manually by the instigator of the survey. Thus, actual answers to our example might range from sensible observations such as

"The driving test is too easily passed"

or

"Too many road users"

to trivial replies such as:

"Idiots"

or

"Not me"

In coding these answers, the researcher would attempt to draw up a set of categories such that all the individual replies would fit into one or other of them. The categories would presumably include items such as '**Drink**' ; '**Speeding**' ; '**Lack of skill**', and, of course, '**Other**'. The data supplied to the MVC system would not be the actual answers but their categories, as fitted by the editor. The edited question would be a polylog if each answer were placed in exactly one category; otherwise it would be a group of binaries.

1.2 Data

Perhaps the most difficult task of the survey analyst is the coding of his raw data into a form acceptable to a computer. There are so many methods of doing this that choosing the most suitable one is often a difficult problem.

In this section we shall attempt to lay down some guiding principles for coding survey data. They will only be of any use to those readers who have not yet broken the cardinal rule — which is not to collect or code any data until a general method of analysis has been decided upon.

If the reader has already got a complete pack of punched cards or a reel of paper tape, it is probable — but not entirely certain — that his data will comply with the MVC conventions. He can discover whether this is so by reading the sections on cards (1.2.1) or on punched tape (1.2.2.). We can add here that complete unsuitability requiring extensive recoding of the data is extremely unlikely. Even if the data cannot be read in the normal manner, it will usually be possible to treat it by a series of 'tricks' or special devices. Advice should be obtained before any of these are used.

If the process of taking and analysing a survey is to run smoothly, it should be carried out in an integrated manner. This implies that the method of data preparation and analysis must be taken into consideration when designing the Questionnaire.

The proper starting-point for designing a Questionnaire is a list of facts to be ascertained for each of the subjects surveyed. The translation of this list into a set of questions is superficially an easy task, but nevertheless one which must be done with care. The precise wording of the questions will depend upon whether the questionnaire is to be filled in by the subject himself, or by another person. In the former case, considerable care must be taken to avoid offending the respondent, and causing his replies to be biased. This psychological requirement often conflicts with the recommendations we make below, which advocate the greatest simplicity in all cases. For example, the wording

"How many times have you been to prison?"

is better (from the point of view of analysis) than

"Have you ever been to prison? If so, how many times?"

because it condenses two questions into one. Nevertheless it would probably irritate a person answering the question.

In spite of difficulties of this type, every attempt should be made to follow the advice set out below.

a) Design of questions

In deciding upon the wording of each question, a suitable type (i.e. numerical, polylog or binary) should be selected and the choice should be firmly held in mind. Numerical questions should clearly request a number, and if relevant, the units to be used should be clearly indicated. Polylog questions should include a list of all possible answers (including, if necessary, the category 'unknown'). Binary questions should be phrased so as to allow only the answers 'yes' or 'no'.

In general, each piece of information should be elicited by one question rather than a related group; nevertheless, items such as dates, sums of money in £. s. d. and times of day are best coded as groups of answers. Conversion to a standard unit (such as days, or pence) can be carried out by the computer.

Questions should be designed so as to retain the maximum amount of information. In particular, a category should never be substituted for an actual known value; thus 'age', if known, should never be replaced by 'age group'. Categorisation can be done by the MVC system at a later stage. Conversely, no attempt should be made to include questions whose answers can be logically deduced from other questions; the deduction can also be done automatically by the system. For example, if two of the questions are 'date of birth' and 'date of examination', then the item 'age at examination' should not be included.

Finally, the proposed wording should be carefully checked — preferably by another person — for any ambiguity or lack of clarity.

b) The complete Questionnaire

The complete Questionnaire is constructed by listing the questions in some sensible order. It is essential that each set of answers should be uniquely identifiable. This can be ensured by including the item 'case number', preferably either at the beginning or at the end of the Questionnaire. If the Questionnaire runs to several pages the case number should be stated on each one. The best arrangement (which is not always practicable) is to pre-print different case numbers on each copy of the questionnaire.

An aspect of the utmost importance is the provision of suitable spaces for the answers to each question. It must always be remembered that these answers will eventually be punched on to cards or paper tape by a clerical operator. The overall accuracy of the survey will be greatly increased if the answers are entirely clear and legible. Wherever possible, answers should be indicated by decimal digits or ticks in pre-printed boxes or rings round printed words and 'freehand' should be avoided entirely.

Spaces for answers should be provided at the right-hand side of the Questionnaire. Numerical questions can be provided with a horizontal row of connected boxes, one for each digit of the answer; polylogs with a vertical row of boxes, one for each possible answer; and binary questions with (preferably) a single box in which a tick may be placed to indicate 'yes'.

The answer boxes should be positioned in such a way that the eyes of the punching operator can travel down them regularly without excessive zig-zag. The whole Questionnaire should be generously laid out, and no attempt to use 'odd corners' for any purpose should be made.

The two Questionnaires which follow - one good, one bad - are both intended

to ascertain the following facts about each subject:

SEX

MARITAL STATUS

NUMBER OF CHILDREN

TYPE(S) OF VEHICLE OWNED.

Please put numbers or ticks in the boxes As appropriate.	
1) Are you: a) Single or b) Married or c) Widowed or d) Divorced	☐ ✓ ☐ ☐
2) What is your sex? a) Male b) Female	✓ ☐
3) How many children have you? (if none, put '0')	0 2
4) Do you own a) A Car b) A Motor-cycle c) A Scooter d) A Bicycle e) Any other vehicle	✓ ☐ ☐ ✓ ☐
SUBJECT'S SURVEY NUMBER [Pre-printed]	0 3 1 8

Name:....*Evelyn Blogg*.....

Address:....*Flat 37A, .94 .Fitzallan .Sq, ..Manchester ..12*.....

Have you ever been married?....*Yes*.....

If so, are you married now?....*Yes*.....

If not, is your former spouse still living?....✓.....

Have you any children ?....*Yes*.....

If so, how many?....*2*.....

What vehicles do you own?....*Car, Scooter, Moped*.....

These two examples make obvious the value of a well-designed Questionnaire; with the first, all ambiguities are avoided, and the task of the person who prepares the data for the computer is limited to copying the characters in the boxes. With the second, this task involves the deduction of the subject's sex from his or her name, and of the marital status from a group of three questions. In addition, there is the difficulty of deciding whether an item such as a 'moped' is to be classed as a scooter, a bicycle, or 'any other vehicle'. It is clear that a questionnaire of the second type will lead to a considerable number of errors in the data. Furthermore, when coding is done by a commercial concern, charges for well laid out Questionnaires are usually very much less than for poorly constructed ones.

Optimum layouts for answer boxes will be discussed in the next section. Meanwhile, let it be stressed that it is impossible to attach too much importance to Questionnaire design.

One of the prime decisions to be made during the course of a survey relates to the medium to be used for supplying data to the computer. In practice, the choice lies between punched paper tape on which the unit of information is the character, and punched cards, which are themselves natural units of information.

There are several factors affecting this choice, of which the foremost is the availability to the user of punched card equipment. In general, it seems that punched cards are more suitable than punched tape for the large majority of surveys. Firstly, they can be pre-edited and sorted by conventional punched card machinery, and secondly, the correction of known errors in data is extremely simple. Tape, on the other hand, is difficult to handle in large quantities, and the correction of errors is an extremely lengthy and tedious process. Nevertheless, paper tape is acceptable for small surveys (up to about 250 cases), and can be recommended in cases where the user is obliged to prepare his own data.

(Operation of punched card machinery is a highly skilled professional task, whereas paper tape equipment can be mastered by anyone in a few minutes).

The two sections which follow are concerned with the recording of data on cards, and tape, respectively. It is only necessary to read the appropriate one of these.

1.2.1.1 Card Layout and Notation

The MVC system is designed to handle standard 80-column cards. Each column

in a card has 12 positions in which holes can be punched. These are known, from the top downwards, as U (for "upper"), L (for "lower"), 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.

Any of the 960 hole sites on a card can be identified by its column number, a solidus, and its row number. Thus the four corner hole sites are known as 1/U, 80/U, 80/9 and 1/9.

In some applications, a natural unit of information is formed not of one, but of several, cards. Two notations are available for distinguishing the various cards from one another. The first assumes that the several cards are equivalent to one "long" card, so that the first column on the second card is "81", the first on the third "161", and so on. In the other notation, a column number in the range 1 — 80 is preceded by a "card number" and a period. Thus column 7 of card 5 is represented by "5.7". Clearly, the dot in this construction is not a decimal point. The two notations may be used interchangeably. The following pairs of equivalents are intended to illustrate them:

45/3 = 1.45/3

115/L = 2.35/L

The MVC system assumes that the hole sites in a card are ordered. In any one column, the "natural" order is from top to bottom; i.e., U, L and 0 to 9. The "first" hole site in any column is deemed adjacent to the "last" one in the previous column.

A "field" of a card is the area of consecutive hole sites in which any item of information is punched.

1.2.1.2 General Strategy

When card coding is used, data from each case is punched on to one or more cards. The number of cards used will depend upon the amount of data present in each questionnaire. In planning the layout of information on the cards, the following rules must be obeyed:

1) Every card must be uniquely identifiable. This means that all cards must carry, in corresponding areas, the case numbers of the subjects they refer to; and, if there is more than one card per case, each card must bear the "card in case" number as well.

2) The fields assigned to the various items of information must be constant. Thus, the answer to any particular question must always be punched in the same area of each set of cards. This rule is necessary because the MVC system identifies answers from their positions on the card.

As well as these immutable rules, there are certain recommendations which the user would be wise to follow:

1) "Double punching"; that is, the presence of more than one hole per column, should be avoided. The reason is that although the MVC system is capable of dealing with double-punched cards, they do not lend themselves easily to the more mechanical operations, such as sorting or tabulating, which are normally carried out by conventional punched card machinery.

2) It is unwise to attempt to "squash" too much information on to one card. Cards are certainly much cheaper than the difficulties and delays, not to say the possible ambiguities, caused by information which is too condensed.

1.2.1.3 Coding numerical questions

Numerical answers to any one question must always occupy the same number of consecutive columns. They must be punched as integers, and be justified to the right. Initial zeroes must be filled in. Thus, if four answers to some question are 25000, 7805, 340 and 5, they must be punched as **25000**, **07805**, **00340** and **00005**, respectively.

Numbers may be punched either as decimal, or duodecimal (scale of 12) quantities. In the former mode (which will be used in the vast majority of cases), each digit is represented by one of the holes **0** to **9** in one column. In the duodecimal mode, which can be used for — say — inches, pence, or months, the hole sites in a column, in their defined natural order, represent the duodecimal digits nought to eleven.

1.2.1.4 Coding binary questions

Each binary question in a questionnaire is assigned one hole site. A hole in that position implies that the answer was "yes".

1.2.1.5 Coding polylog questions

Each polylog question is assigned a field of two or more consecutive hole sites. The number of hole sites in the field is equal to the number of possible answers to the polylog; and the actual answer in any particular case is indicated by a hole in the corresponding position.

1.2.1.6 Planning card layout

The card layout is best decided by the following procedure:

1) Decide how many cards are required for each questionnaire. Allow 1 column for each binary answer, each polylog answer*, and each digit of each numerical answer.

2) Decide on the position of the case number on each card. It is customary to assign the first few columns for this purpose. If the number of cards per case exceeds one, decide on a position for the "card in case" number. It is common to use the last column for this purpose.

3) Assign actual fields to each of the expected answers. It is important that these fields should be in the same order as the corresponding questions in the questionnaire. Avoid any "compression", special codings, and the like.

4) Check carefully for possible slips such as assigning one column to two different purposes. Such mistakes can be very expensive if not caught early.

1.2.1.7 Design of questionnaires

In designing the answer boxes from which data is to be transferred to cards, it is important to distinguish between items which are to occupy separate columns, and those which go on the same column. A possible technique is to use a horizontal separation for items on separate columns only. Needless to say, it is also vital to arrange the coding so that the punched card operator can work straight down the page.

* If a polylog has more than 12 possible answers, more than 1 column will be required

This is illustrated below:

1) CASE NUMBER	Cols 1-3 ☐☐☐
2) What is your age?	Cols 4-6 ☐☐☐
3) Are you: a) Single b) Married c) Widowed d) Separated e) Divorced	☐ 7/0 ☐ 7/1 ☐ 7/2 ☐ 7/3 ☐ 7/4
4) At what age did you leave school?	Cols 8-9 ☐☐
5) Did you attend: a) University b) Technical College c) Theological College d) Medical School	☐ 10/0 ☐ 11/0 ☐ 12/0 ☐ 13/0
6) If you attended a university, did you get a) a First b) a Second (upper) c) a Second (lower) d) a Third e) a Fourth f) a 'pass' degree g) A fail h) Not applicable	☐ 14/0 ☐ 14/1 ☐ 14/2 ☐ 14/3 ☐ 14/4 ☐ 14/5 ☐ 14/6 ☐ 14/7
7) What is your present income (before tax)	Cols 15-19 ☐☐☐☐☐

1.2.2 Coding data on punched tape

Punched tape is usually prepared on a flexowriter, or a similar machine produced by another firm. Appendix 1 of this book contains a description of these machines, and readers who are unfamiliar with them are advised to read it.

In preparing data on a flexowriter, any single character on the keyboard may be used freely, but superimposed pairs (other than those containing "erase") are banned. "Newline", and "space" and its synonyms may not be used as significant data characters, although they may be employed to "punctuate" the data.

1.2.2.1 The notion of "records" and "items"

When data is punched on to tape, information on each case is said to comprise a "record". Each record must be terminated by a special symbol which is not then used in any other context. If the records are of the right length, "newline" forms a suitable terminator; but if they are too long (or too short) some other symbol, such as (for example) "`*`" or "`/`" must be chosen.

Within each record, data may be coded by either of two conventions.

In the "character" convention, the record consists of a simple sequence of characters, all of which are significant. The various answers are identified by their position in this sequence, so that the number of characters allotted to each one must be constant for all possible answers to that particular question.

In the "item" convention, each item forms a unit of information by itself, and must be separated from the next item by a "separator". Like terminators, separators must be specially chosen characters which are not to be used in any other context. The system identifies the various items by their "item number" in the record, so that each item may use a variable number of characters. The order in which the items are punched must of course be fixed.

The decision on which of these two conventions is to be used again depends on circumstances. In general, records in the character convention will be much more compact than corresponding records in the "item" convention, but they will require more careful preparation, and will be more difficult to check. This can be seen from the examples given at the end of this section. It is recommended that the character convention be used for medium-sized surveys, where the operator of the flexowriter is relatively skilled. The item convention is only recommended for very small surveys.

1.2.2.2 Coding of numerical data

In the "character" convention, numerical answers to any one question must always occupy the same number of characters. They must be justified to the right. Initial zeroes must be filled in, but a decimal point, and initial + or – are permitted, and count in the number of characters.

In the "item" convention, numerical answers may be punched in any reasonable way. Signs and decimal points are permitted. Very large numbers may be punched in "floating point form", with suitable decimal exponents, in the format:

argument `|` exponent

For example, `5.69|7` stands for 5.69×10^7, or `56, 900, 000`.

1.2.2.3 Coding Binary and Polylog data

Both these types are coded in a very similar way. Each possible answer is assigned one or more characteristic symbols, and the appropriate group is included in each record. If the character convention is in force, then the number of symbols in each of the possible answers to any one question must be constant.

1.2.2.4 Design of Questionnaires

If the character convention is in use, the number of characters in the record is of prime importance, and the questionnaire must be designed accordingly. It is recommended that for numerical answers, rows of boxes with the exact number of digits be provided. For polylog and binary questions, boxes with the possible answers already in them should be included. In answering the questionnaire, the subject (or interviewer) would fill in the numerical answers, and indicate the polylog and binary answers by ringing the appropriate box.

If the item convention is to be used, the design of the questionnaire is not so highly critical. We suggest that blank lines be left for numerical answers, and replies to polylog and binary questions could be indicated by ringing a suitable word, which would then be copied directly to the punched tape record.

We shall give two examples of the same questionnaire: one suitable for the character convention, and one intended for the item convention.

1)	What is your age?	2 4
2)	How many years have you been in your present job?	0 5
3)	Do you consider the lighting on your job:	a) Good A b) Fair B c) Bad C d) Terrible D
4)	Do you wear glasses?	Y (N)
5)	Do you think that carrots	a) Help + b) are indifferent to 0 c) Hinder - eyesight?
Case number:		1 2 3

1) What is your age?.....

2) How long have you been in your present job?

(years)

3) Do you consider the lighting on your job:

GOOD FAIR

BAD TERRIBLE

4) Do you wear glasses YES NO

5) 5) What effect do you think that carrots have on eyesight

HELP INDIFFERENT

HINDER

Case Number:..123.....

Below, we give some records which might possibly have been prepared from answers to the questionnaires above. Both sets refer to the same group of cases.

1) In the character convention: The terminator is '*****'.

```
2405AN+123*4621DY-124*3701BN0125*1902CY+126*6244CY0127*
```

2) In the item convention:

The separator is ',' and the terminator is 'newline'.

```
24, 5, GOOD, NO, GOOD, 123
46, 21, TERRIBLE, YES, BAD, 124
37, 1, FAIR, NO, INDIFFERENT, 125
19, 2, BAD, YES, GOOD, 126
62, 44, BAD, YES, INDIFFERENT, 127
```

1.3 The MVC Language.

When a computer is used to analyse a survey, it must be supplied with a description of the survey itself, as well as the actual data to be analysed. The description is written in the MVC language.

The MVC language is normally typed on a flexowriter (which is described in Appendix 1). The conventions for typing MVC specifications are slightly different from those used for typing data (see section 1.2.2). In particular, the character set is more restricted, but certain compound (i.e. superimposed) characters are permitted. The MVC character set was chosen so that MVC specifications can therefore be typed on any flexowriter or similar machine associated with ATLAS.

1.3.1 The MVC character set

The MVC character set contains the following symbols:

All letters (both upper and lower cases)

All decimal digits

Other symbols:

+ - * / = > < , . ( ) [ ] | : ? ' _ (underline) /// (erase)

Non-printing symbols:

"space", "backspace", "lower case", "uppercase", "newline" and "tabulate". (The use of "tabulate" is discouraged, except at the beginning of lines).

Compound symbols: (produced with the aid of "backspace")

All underlined letters;

≠ (= and /)

≤ (< and _)

≥ (> and _)

; (, and :)

Within "quotations" (1.3.3.4), the above conventions do not apply, and any single character on the flexowriter keyboard may be used.

1.3.2 Interpretation conventions.

1. The system always interprets each line according to its printed appearance. It follows that the order in which the components of compound characters are typed is immaterial.

2. Both "underline", by itself, and "tabulate" are synonymous with "space". All sequences of "spaces" or synonyms are equivalent to one space. Except in quotations, the upper and lower case forms of any letter are synonymous.

3. The symbol /// (erase), whether by itself or superimposed on any other character, is always ignored.

Any sequence; beginning with "**comment**" and ending with ";" is always ignored.

Spaces are always ignored except in "names" (1.3.3.2).

"Newline" is always ignored.

1.3.3 MVC words.

As in a human language, the individual characters in an MVC text are usually combined to form "words". There are four main types, which are described below:

1.3.3.1 Numbers.

Groups of decimal digits and the symbol "." can be assembled, in the conventional fashion, to form numbers. Examples are:

45 or 467.34 or 0.0006

All the numbers in an MVC text must contain an integral part. Thus 7, 5.8 or 0.13 are legal words, whereas .98 is not.

1.3.3.2 Names.

In the MVC system, the various entities in a survey are referred to by names. Subject to certain rules, names are chosen arbitrarily by the user. Examples are:

```
B5
Age Group
SCORING ABILITY
Eats crisps in bed
jAnUaRy 19Th 1964
X167 Y435 Z65
Chewing gum consumption in 1963
```

The rules governing the construction of names are these:

1) They must contain only letters, digits and spaces.

2) They must start with a letter.

3) They must contain at least 2 and at most 63 symbols.

4) They must not end with 'space'.

Names which differ only by the presence or absence of spaces are distinct from one another. Nevertheless, the use of two names which are only different in this respect (such as '**INCOMETAX**' and '**INCOME TAX**') is not recommended, for obvious reasons.

1.3.3.3 System words.

An MVC specification must refer to certain fixed concepts. For a few of these, there exist single symbols (for example, '**+**' and '**–**' to specify the concepts of addition and subtraction, respectively). For the majority, however, no single symbols are available; instead, their concepts are referred to by 'system words'. A system word is quite simply the name of the concept. In a text, it is always underlined to distinguish it from any user's name consisting of the same letters. Examples of system words are **not**; **U**; **comment** and **tabulate**.

Although the upper case and lower case forms of any letter are synonymous,

in this book we shall follow the convention of writing users' names in capitals, and system words in lower case letters.

It must be stressed that system words are a fixed feature of the MVC language. The user is not free to invent his own system words, but must use those provided.

1.3.3.4 Quotations.

In an MVC text it is sometimes necessary to quote a sequence of characters which is not part of the text itself. This may be done by putting the sequence inside single quotation marks, thus:

```
'This is a quotation'
```

or

```
'The £ — % problem'
```

Unlike the rest of the MVC text, quotations may include any single characters except for '.

1.4. Variables

In the MVC system, the Questionnaire being analysed is represented by a collection of variables. Each variable corresponds to one of the questions in the Questionnaire. The individual variables are referred to by names arbitrarily chosen by the user. Usually, the variables will be assigned names which indicate their nature; for example,

HAIR COLOUR; or **HEIGHT**.

In some applications, of course, the user might prefer to call his variables **X1**, **X2**, and so on; this is permissible.

Each answer to a question in a questionnaire defines a value for the corresponding variable. Each completed set of answers will therefore define a new list of values for the variables; in general, this list will be different from the list defined by any other completed questionnaire.

The values which variables may assume are not necessarily numerical; this will be explained below.

There are three types of variable, corresponding exactly to the three types of question. They are;

1.4.1. Numerical variables.

Numerical variables are used to represent numerical questions. They take the (numerical) values of the answers to these questions.

1.4.2. Binary variables.

Binary variables correspond to binary questions. They can only take two values, **TRUE** for the answer 'yes', and **FALSE** for the answer 'no'.

1.4.3. Polylog variables.

Polylog questions are represented by polylog variables. The values which a polylog variable may assume correspond to the answers which may be given to the question. These answers can be given names, and referred to by these names, in the same way as variables themselves.

To illustrate these concepts, consider this questionnaire:

Questions		Hypothetical sets of answers				
		1	2	3	4	5
1) What is your age?		52	43	67	19	7
2) Do you wear galsses?		Yes	No	No	Yes	No
3) Are you left-handed?		No	Yes	No	No	No
4) Is your hair	a) Red	Yes				
	b) Fair				Yes	
	c) Brown					Yes
	d) Black		Yes			
	e) White					
Or are you	f) Bald			Yes		
5) What is your weight (lbs)		201	154	177	114	89
6) Were you born in	a) England	Yes		Yes	Yes	
	b) Scotand					
	c) Wales					
	d) Ireland		Yes			
	e) Elsewhere					Yes

In preparing this questionnaire for analysis, one would assign types and names to the variables, and, where applicable, to the answers as well. This might be done as follows:

Question number	Name and type of variable	Answer names
1	AGE (Numerical)	
2	SPECTACLES (binary)	
3	LEFTHANDED (binary)	
4	COLOUR OF HAIR (polylog)	RED, FAIR, BROWN, BLACK, WHITE, BALD
5	WEIGHT (numerical)	
6	COUNTRY OF BIRTH (polylog)	ENGLAND, SCOTLAND,WALES, IRELAND, ELSEWHERE.

.

During the analysis, each of the hypothetical sets of answers given above would be represented by an array of values for the named variables. These values would be:

Variable	Case 1	Case 2	Case 3	Case 4	Case 5
AGE	52	43	67	19	7
SPECTACLES	True	False	False	True	False
LEFTHAND-EDNESS	False	True	False	False	False
COLOUR OF HAIR	Red	Brown	Bald	Fair	Brown
WEIGHT	201	154	177	114	89
COUNTRY OF BIRTH	ENGLAND	IRELAND	ENGLAND	ENGLAND	ELSEWHERE

Variables where values correspond directly to the answers given to a questionnaire are called “raw variables”. All the variables listed in the table above are of this type.

1.5 Raw Variable Definitions

If the reader has successfully followed the text to this point, he will appreciate that each question in the Questionnaire has three attributes:

1) a type (i.e., Numerical, Binary, or Polylog)
2) a name arbitrarily assigned (e.g. "number of cats in household")
3) an area of the case record where answers to that question are punched (e.g., "columns 34 — 37" or "item 16").

In addition, polylog questions have a fourth attribute, namely, the names assigned to the various possible answers.

The description of each variable (i.e. its list of attributes) is indicated to the MVC system by means of a "raw definition".

Each raw definition explicitly states the name of the variable, and the area of the case record in which it is punched. The type is stated implicitly by the position of the definition; because all definitions of the same type are gathered together under a suitable heading.

When typed, the components of raw definitions are separated by commas, and the definitions themselves are terminated by semicolons. The list of answer names which must be included in polylog definitions is preceded by a colon.

Since cards and punched tape are essentially different media, it is necessary to use slightly different formats in specifying raw variables. Below, we shall discuss each format in turn.

1.5.1 Raw definitions for card input

Before any definitions for data on cards can be written, it is necessary to declare the expected type of input. This is done by the announcement:

read cards; (i.e. as opposed to "**read tape**")

The definitions of variable values punched on cards must include explicit column numbers and hole site positions. These are written in the notation explained in section 1.2.1.1.

1.5.1.1 Raw numerical card definitions

Definitions for raw numerical variables are always grouped together under the heading

raw numerical;

Each definition consists of the following items:

1) The name of the variable.
2) The number of the most significant column allotted to the answer.
3) The number of digits in the answer.
4) Optionally, the system word "**dozen**". This signifies that the quantity is punched on the scale of 12.

Examples of raw numerical definitions are:

YEAR OF BIRTH, 5, 4; (i.e. 4 cols starting at col 5)

BEER CONSUMPTION, 2.46, 2;

PRICE PER HALF PINT IN PENCE, 2.48, 1, dozen;

1.5.1.2 Raw binary card definitions

The group heading for binary definitions is

raw binary;

Each definition consists of the name of the variable, and the hole site assigned to it in the case record. The variable will take the value "true" if a hole is actually punched in the indicated position.

Examples are:

SMOKES PIPE, 57/U;

READS FINANCIAL TIMES, 133/4;

PASSED 11 PLUS, 4.34/0;

1.5.1.3 Raw polylog card definitions

Raw polylog definitions must be collected together under the title

raw polylog;

A raw polylog definition is constructed from the following components:

1) The name of the variable.

2) The "first" hole site in the assigned field.

3) The number of hole sites in the field (i.e. the number of possibilities).

4) A list of names for the various answers.

Some possible examples are:

SOUND LEVEL, 5/0, 4: QUIET, MEDIUM, NOISY, VERY NOISY;

FAVOURITE TYPE OF PIPE, 56/0, 3: WOODEN, CLAY, MEERSCHAUM;

SEX, 2/5, 3: MALE, FEMALE, NOT KNOWN;

1.5.1.4 Case number declarations

Every MVC specification which deals with data punched on cards must include a "case number declaration". This is an announcement of the position which the case number occupies on each card, and takes the form

case number x, y;

where **x** is the first column assigned to the case number, and **y** is the number of columns.

Although similar in format, the case number declaration is not a definition. Its purpose is to allow the system to recognise cards which belong to the same case, and to check that the cases are correctly presented. If the case number is required as a variable, it must be separately specified by a definition such as

CASE NUMBER 1, 4;

1.5.1.5 A complete example.

As an example of a complete set of raw definitions, we shall quote those which apply to the questionnaire shown in section 1.2.1.7.

```
read cards;
raw numerical;
AGE,4,3;
AGE LEFT SCHOOL,8,2;
INCOME,15,5;
raw binary;
UNIVERSITY,10/0;
TECHNICAL,11/0;
THEOLOGICAL, 12/0;
MEDICAL, 13/0;
raw polylog;
CIVIL STATUS,7/0,5: SINGLE, MARRIED, WIDOWED,
SEPARATED, DIVORCED;
TYPE OF DEGREE,14/0,8: CI,C2U,C2L,C3,C4,PASS,FAIL,NA;
case number 1,3;
```

1.5.2 Raw definitions for Tape input.

Definitions of variables punched on tape must be preceded by declarations of the type of punching used, and of the separators and terminators used to punctuate the record.

The punching convention is simply indicated by one of the two announcements

`read characters;` (i.e., the character convention)

or

`read items;` (i.e., the item convention).

The chosen punctuation symbols are listed in a "record format" statement. Since the permissible symbols may be any of those available on the flexowriter keyboard, and need not necessarily belong to the MVC character set, they must be made into "quotations" (1.3.3.4). Three "unquotable" characters must be indicated by name; viz newline, space and prime (i.e., the symbol ').

The user is permitted to specify as many alternative terminators (or separators) as he wishes. When listed, these alternatives are separated by commas, which carry the meaning 'or'.

A typical format statement used in conjunction with the character convention is

`terminator: '.';` It specifies that each record is terminated by a full stop. Similarly, a format statement used with the item convention might be

`separator:space, terminator: ',';` This signifies that individual items are separated by spaces, and that entire records are terminated by commas.

Another format statement, including alternatives, might be:

```
separators: '.' , '/' , prime, ':' , terminators: '?' , newline;
```

These statements should be carefully examined, and the precise punctuation noted. Both the system words separator and terminator may be used in either their singular or their plural forms.

Since the 'character' and 'item' may employ different formats

the structure of raw definitions in the two conventions is slightly different. In 'character' definitions, the characters in the record are number notionally 1,2,3,.... and are referred to by these numbers; similarly, in 'item' definitions, the items are referred to by the numbers 1,2,3, '1' indicating the first item in the record.

1.5.2.1 Raw Numerical Tape Definitions.

The definition of a raw numerical variable punched on tape contains the name assigned to the variable, and an indication of its position in the record. In the item convention, this is given by the item number alone; but in the character convention, it is necessary to state both the most significant character number and the expected number of characters (which is, of course, constant). Examples are:

```
ELECTRICITY USED,6;        (item convention)
MPG, 54, 2;                (character convention)
```

Raw numerical definitions are gathered together under the heading

```
raw numerical;
```

1.5.2.2 Raw Binary Tape definitions.

The group heading for binary definitions is:

```
raw binary;
```

The definition of a raw binary variable contains a name, and a corresponding position in the case record (i.e. the item number in the 'item' convention, and the position of the first character in the 'character' convention). Since the user is entirely free to choose the actual symbols that represent the two possible answers to the binary question, the definition must also include a specification of the chosen symbols. These are indicated as quotations, and are placed in the order 'answer meaning 'yes' / 'answer meaning 'no'.

A typical binary definition for the character mode might be:

```
RETIRED, 45, "+",
```

and for the item mode:

```
OWNS CAR,7, 'YES', 'NO';
```

It is evident that in a character mode definition the number of symbols in each possible answer must be the same.

1.5.2.3 Raw Polylog Tape Definitions.

These are in some respects similar to the raw binary definitions. Each one consists of the following components:_

1) The name assigned to the variable.

2) A position in the case record (item or character number)

3) The number of possible answers.

4) Quotations giving the possible forms of the answers

5) A list of names for the answers.

Some examples are:

```
          AGREEMENT WITH ATTITUDE,17,5,'--','—','?', '+','++':
          STRONGLY DISAGREE, DISAGREE, DONT KNOW, AGREE,
          STRONGLY AGREE;
or        SECURITY, 34,5,A, B , C , D, E: TOP SECRET, SECRET,
```

```
CONFIDENTIAL,RESTRICTED,UNCLASSIFIED;
```

The relationship between the list of answer formats and the list of answer names may appear somewhat obscure, and must be clarified.

The answer names of a polylog are words in the MVC language. They must be constructed according to the proper conventions, and, once having been included in a definition, they may be used elsewhere in the specification, for instance, in the description of a required contingency table. On the other hand, each answer format is in no sense an MVC word; it is merely a 'picture' of a possible arrangement of the symbols in the data. It is not liable to any of the constraints which apply to the structure of names. Furthermore, each name in a specification must be distinct from any other; but there is no reason why two different questions may not be represented in the data by similar sets of symbols. (The answers are of course distinct by virtue of their position.)

It will sometimes happen, particularly in the item mode, that the answer formats are identical to the answer names. If this occurs both lists must nevertheless be quoted in full. This may lead to a polylog definition such as:

```
DRINK,9,3,'BEER','WINE','SPIRITS': BEER,WINE,SPIRITS;
```

Polylog definitions are collected together under the heading

```
raw polylog;
```

1.5.2.4 Case number declarations.

Every set of raw definitions must be associated with a 'case number declaration'. This is an announcement of the position of the case number in each record. It is not a definition, but serves to permit the system to check that all the cases in the survey are presented correctly.

In the item convention, the declaration takes the form:

```
case number x;
```

where **x** is the item number of the case number.

Similarly, the form of the declaration in the character convention is:

```
case number y, z;
```

where **y** is the position of the first character of the case number, and **z** is the number of characters it contains.

If the case number of a record is required for purposes of analysis, then it must be separately defined by a definition such as

```
CASE NUMBER 7;
```

1.5.2.5 Complete examples.

As complete examples of sets of raw definitions, we shall quote those applicable to the examples given in section 1.2.2.4.

1. The CHARACTER convention

```
read characters;
terminator:"*";
raw numerical;
```

```
AGE, 1,2;
YEARS IN PRESENT JOB, 3, 2;
raw polylog;
LIGHTING,5,4,'A','B','C','D': GOOD,FAIR,BAD,TERRIBLE;
CARROTS,7,3, + , O , - : HELP, INDIFFERENT,HINDER;
raw binary;
GLASSES,6,'Y','N';
case number 8,3;
```

2. The ITEM convention.

```
read items;
separator:',', terminator:newline;
raw numerical;
AGE, 1;
YEARS 1N PRESENT JOB,2;
raw polylog;
LIGHTING,3,4,'GOOD','FAIR','BAD',ITERRIBLE':GOOD,FAIR BAD,
TERRIBLE;
CARROTS, 5, 3, 'HELP', 'INDIFFERENT', 'HINDER' : HELP, 1NDIF-
FERENT, HINDER;
raw binary;
GLASSES,4,'YES','NO';
casenumber 6;
```

1.6 Boolean Particles

A notion of fundamental importance in the MVC system is that of BOOLEAN PARTICLES. These are named after the Irish mathematician George Boole. Although the ideas presented below may be unfamiliar to some readers, they should present no difficulty, because they are mainly formalisations of every day 'common sense'.

A Boolean particle is an 'object', analogous to a variable, which admits of only two values; 'true' and 'false'. It can be formed in four ways:

1) It may be a binary variable.

2) It can be an answer to a polylog question. The value of such a Boolean particle is 'true' if that answer was actually given, and 'false' otherwise.

3) It may be an arithmetical relationship, such as '=' or '>' between two numerical variables, or between a numerical variable and a number. The particle takes the value 'true' if the relation is satisfied. When written, particles of this type must always be enclosed in round brackets.

4) It may be one of the Boolean constants '**T**' (for 'true') or '**F**' (for 'false').

To illustrate the concept of Boolean particles, we shall use the Questionnaire quoted in section 1.4, (and repeated below for the convenience of the reader).

Questions		Hypothetical sets of answers				
		1	2	3	4	5
1) What is your age?		52	43	67	19	7
2) Do you wear galsses?		Yes	No	No	Yes	No
3) Are you left-handed?		No	Yes	No	No	No
4) Is your hair	a) Red	Yes				
	b) Fair				Yes	
	c) Brown					Yes
	d) Black		Yes			
	e) White					
Or are you	f) Bald			Yes		
5) What is your weight (lbs)		201	154	177	114	89
6) Were you born in	a) England	Yes		Yes	Yes	
	b) Scotand					
	c) Wales					
	d) Ireland		Yes			
	e) Elsewhere					Yes

The variables derived from this Questionnaire were:

AGE (numerical)

SPECTACLES (binary)

LEFTHANDED (binary)

COLOUR OF HAIR (polylog, possible answers **RED**, **FAIR**, **BROWN**, **BLACK**, **WHITE**, **BALD**)

WEIGHT (numerical)

and

COUNTRY OF BIRTH (polylog, possible answers **ENGLAND**, **SCOTLAND**, **WALES**, **IRELAND**, **ELSEWHERE**)

The table below contains some examples of each type of Boolean particle, together with the values the particles take for each of the hypothetical sets of answers.

Type	Particle	Values				
		Case 1	Case 2	Case 3	Case 4	Case 5
1 (binary)	**SPECTACLES**	**T**	**F**	**F**	**T**	**F**
	LEFTHANDED	**F**	**T**	**F**	**F**	**F**
2 (polylog answer)	**RED**	**T**	**F**	**F**	**F**	**F**
	BALD	**F**	**F**	**T**	**F**	**F**
	IRELAND	**F**	**T**	**F**	**F**	**F**
3 (relation-ship)	**(AGE > 50)**	**T**	**F**	**T**	**F**	**F**
	(WEIGHT < 160)	**F**	**T**	**F**	**T**	**T**
4 (constant)	**T**	**T**	**T**	**T**	**T**	**T**
	F	**F**	**F**	**F**	**F**	**F**

Note that the particle '**SPECTACLES**' corresponds roughly to the English statement 'This person wears glasses'. It takes the value '**T**' whenever the statement is true. Similarly the particle '**RED**' corresponds to the statement 'This person has red hair'. Thus, in case 1, **RED** = **T** because **COLOUR OF HAIR = RED**; but **BALD** = **F** because **COLOUR OF HAIR ≠ BALD**. In general, any Boolean particle corresponds to a statement which may or may not be true. A Boolean particle may be said to define a set or group among the cases in a survey, namely, those cases for which it is true.

1.7 Boolean expressions.

In English, statements can be combined with the aid of conjunctions such as 'and', 'not', 'also', and so on. In an analogous, but much more rigidly defined way, Boolean particles can be combined to form expressions. The syntax, or set of rules for constructing them is very similar to the set governing the construction of ordinary algebraic expressions; but the values which Boolean expressions may take are limited to **T** and **F**, like the Boolean particles themselves.

The simplest form of Boolean expression consists of just one particle, and takes the value of this particle. More complicated expressions are assembled from particles and 'operators', which correspond to the algebraic signs '+', '/', etc. Brackets may also be used.

The Boolean operators are '**and**', '**or**' and '**not**'. (These groups of symbols are actual system words in MVC). '**not**' operates on one particle(just as √ operates on one number), and **and** and or operate on two particles each (just as + or ×). Since the particles the operators act upon can only take the two values **T** and **F**, their effect can be completely defined by listing the results for all possible combinations of particle values.

not T = F	**F or F = F**	**F and F = F**
not F = T	**T or F = T**	**T and T = F**
	F or T = T	**F and T = F**
	T or T = T	**T and T = T**

In any practical application, a Boolean expression has a meaning which is usually obvious. Thus, for example, the expression

not LEFTHANDED

is true for all persons who are not lefthanded; or

BLACK and WALES

is true for all black-haired Welshmen. (In this context the word 'man' is used in the sense 'a member of the human race'.)

Many Boolean expressions contain more than one operator. In order to give such expressions the precise meaning which the corresponding English statements sometimes lack, it is necessary to specify the order in which the operators are to be applied. This is

first, the **not**'s;

second, the **and**'s;

third, the **or**'s.

This rule is closely analogous to the algebraic convention whereby in an expression such as 'A × B + C × D', the multiplications are carried out before the additions. That the rule is necessary is clearly shown by ambiguous statements such as My telephone is not working and red.

If these precedence rules are applied, the expression

not BALD or SPECTACLES and ENGLAND

turns out to define the class which includes a) All men with hair;

and b) All bespectacled Englishmen.

Note that bald bespectacled Englishmen are included.

In the Boolean expressions, the order in which the operators are applied can be changed by using brackets. The rule here is that everything inside a pair of brackets is evaluated first. Naturally, the presence of brackets can affect the meaning of certain expressions considerably;

For example,

```
not (BALD or SPECTACLES) and ENGLAND
```

now defines the class of Englishmen who are neither bald nor wear glasses; and

```
(not BALD or SPECTACLES) and ENGLAND
```

defines the class of Englishmen who have hair or wear glasses (or both).

These examples would repay careful study.

The user of MVC is concerned not so much with deciphering existing Boolean expressions as with writing down his own in order to define classes in which he is interested. As the following examples show, this is easily done.

1) To define the class of 'Irishmen over 65 years old'.

```
All Irishmen          = IRELAND.

All men over 65       = (AGE > 65)

Irishmen over 65      = IRELAND and (AGE > 65)
```

2) To define the class of 'Celtic Teenagers who are not lefthanded'.

```
All Celts             = SCOTLAND or WALES or IRELAND
All over 12           = (AGE > 12)
All under 20          = (AGE < 20)
All teenagers         = (AGE > 12) and (AGE < 20)
All Celtic Teenagers  =
(SCOTLAND or WALES or IRELAND) and (AGE > 12) and (AGE < 20)
```

The required expression is

```
(SCOTLAND or WALES or IRELAND) and (AGE > 12) and (AGE < 20)
and not LEFTHANDED
```

In this expression, the brackets round the definition of 'Celts' is necessary, for without them, the expression would change its meaning. Whenever there is any doubt, the brackets may be used freely. As long as they match, there is no penalty for redundant pairs.

1.8 Arithmetical expressions.

A construction which is often used in the MVC system is the Arithmetical Expression. Arithmetical expressions are built up from the names of numerical variables, numbers, brackets, and the algebraic signs `+`, `-`, `/`, `|` (exponentiation), and `*` (which means multiplication and always must be stated explicitly). Under some conditions they may contain Boolean expressions.

A full description of the relevant syntax is given in appendix 2 to this book. Although it may appear complicated, it is designed so that anything which 'looks alright' is automatically valid. The appendix should only be referred to if the reader is in doubt.

In general, arithmetical expressions are evaluated for each case submitted for analysis. During the evaluation, the actual values of the numerical variables named in the expression are used; so that the value of the expression is a number which depends upon the value of these variables. For example; suppose that each subject in a survey had taken four exam papers, and that the variables representing the scores were

```
ENGLISH MARKS
FRENCH MARKS
MATHS MARKS
HISTORY MARKS
```

Then the total marks for all subjects would be given by the expression

```
ENGLISH MARKS + FRENCH MARKS + MATH MARKS + HISTORY MARKS
```

And the average mark (assuming that the maximum marks were the same in each case) would be written as:

```
(ENGLISH MARKS + FRENCH MARKS + MATHS MARKS + HISTORY MARKS) / 4
```

The value of the average mark would be computed for each person in the survey. It would clearly depend on the values of the individual marks and would vary from case to case.

Boolean expressions can be included in arithmetical expressions with the aid of the conversion operator C. This is used in the construction 'C (any Boolean expression or particle)', which has the (numerical) value `1` if the expression is true, and the value `0` if it is false. Thus, `C(T) = 1`, and `C(F) = 0`.

There will be so many examples of arithmetical expressions in the following sections that their inclusion here would be redundant.

1.9 Derived variables.

In survey analyses it frequently happens that the data read from the case records is not directly suitable for tabulation. For example, it may be required to tabulate certain quantities against — say — income groups, whereas the data contains, for each case, the income as a numerical quantity. Again, while a questionnaire may contain a person's weight in stones and pounds, the actual quantity to be dealt with might be his weight in kilograms. Yet again, a person might state his religion as one of a dozen possibilities, whereas the survey requires merely the simple classification 'Christian/not Christian'.

The requirement is one in which the computer begins to show its considerable superiority over conventional card sorting and tabulating machinery. For the latter, it is usually necessary to calculate the various quantities needed by hand, before punching the data. Although the calculations are generally simple, they become extremely tedious and prone to errors if the survey sample is at all large. If, on the other hand, a computer is being used, it is only necessary to punch the actual answers, (i.e. the 'raw data') given in each questionnaire. Any transformation on these values which may be required before tabulation, can be carried out automatically by the computer.

To permit these calculations, the MVC system allows for two classes of variables. Raw variables (as stated previously) are those whose values are read directly from the case record. Variables whose values are to be computed from the values of the raw variables are known as derived variables.

Derived variables are also of three types; numerical, binary and polylog. Each derived variable is specified by a definition which links an arbitrary name with a formula for computing its value.

1.9.1 Derived Numerical Variables.

Derived numerical variables are defined with the aid of expressions. The definitions are written as follows:

1) The name of the variable being defined.

2) The compound symbol '`:=`' pronounced 'is defined as'.

3) An arithmetical expression.

4) A semicolon.

The value assigned to the defined variable is that obtained by evaluating the expression.

Some examples of derived numerical variable definitions follow.

a) `TOBACCO IN OZ := 0.035*CIGARETTES + PIPE TOBACCO + 0.5*CIGARS;`

This definition might arise in a medical survey. The numerical variables represent the questions 'How many cigarettes per day do you smoke?', 'How many ounces of pipe tobacco?', and 'How many cigars?'. The derived variable '`TOBACCO IN OZ`' is intended to be a general measure of tobacco consumption independent of the actual type of tobacco used. It is expressed in terms of 'ounces of tobacco', The constants `0.035` and `0.5` are introduced because cigarettes and cigars are supposed to weigh 0.035 and 0.5 ounces respectively.

b) `WEIGHT IN KG := (14*STONES + POUNDS + OUNCES/16)/2.205;`

This definition is typical of those which might be used to reduce quantities given in several units to one common measure.

c) **SKIN AREA := 71.84 * WEIGHT IN KGS | 0.425 * HEIGHT IN CM | 0.725;**

This is a representation of Duboys' Formula

$S = 71.84\ W^{0.425} \times H^{0.725}$.

It expresses a quantity which is very difficult to measure in terms of two other easily determined numbers.

d) **DISTANCE := sqrt (NORTHING * NORTHING + EASTING * EASTING);**

This definition illustrates the use of the special function '**sqrt**'. This is one of a standard set listed in appendix 2.

e) **WEIGHT GAIN := (1 – 2 * C(LOSING WEIGHT)) * WEIGHT CHANGE**

This definition shows the way in which signed numbers are handled in the MVC system.

When a numerical quantity in the data is expected to be either positive or negative, then on cards it is best represented by a raw numerical variable which gives its absolute value, and a raw binary variable which gives its sign. These two answers can then be combined to give one derived variable, as in the example above. In this example, it is assumed that the subject's change in weight is represented by the variable **WEIGHT CHANGE** (which must always have a positive value). The binary variable **LOSING WEIGHT** is used to represent the direction of this change, and has the value **T** if this is negative. It will be seen that the contents of the brackets

(1 – 2 * C(LOSING WEIGHT))

will have the value **+1** if there is a real gain in weight (because then **LOSING WEIGHT = F**, and **C(F) = 0**), and **–1** if there is a weight loss (for a similar reason).

There are several different ways in which such a definition can be written. Among them is:

```
WEIGHT GAIN := WEIGHT CHANGE * C(not LOSING WEIGHT) – WEIGHT CHANGE *
                    C(LOSING WEIGHT);
```

When punching signed decimal numbers on to cards, it is quite permissible to place the binary answer corresponding to the sign in the 'upper' or 'lower' hole site in one of the columns used for punching the absolute value of the number. When using punched tape this problem does not arise because '**+**' and '**–**'are permissible symbols.

In the MVC specification, all the derived numerical definitions must be collected together under the heading

derived numerical;

1.9.2 Derived Binary Variables.

Derived binary variables are defined in much the same way as derived numerical variables, except that a Boolean expression is used in place of an arithmetic metical one. Some examples are:

a) **CHRISTIAN := ANGLICAN or ROMAN CATHOLIC or ORTHODOX or NONCONFORMIST;**

b) **ADULT := (AGE ≥ 21);**

The group heading for derived binary definitions is

derived binary

,

1.9.3 Derived Polylogs

The main use of derived polylog variables is in grouping quantities which in the raw data are given as simple numbers. Thus, for example, the raw numerical variable '**INTELLIGENCE QUOTIENT**' might be used to define a derived polylog whose several possible (and mutually exclusive) values might be '**MORON**, **STUPID**, **AVERAGE**, **BRIGHT**, **GENIUS**'.

The definitions of derived polylogs which are quantisations of numerical variables take the following form:

1) The name of the polylog being defined.

2) The symbol '**:=**'.

3) The name of the numerical variable being quantised.

4) An arithmetical relation such as **>** or **<**.

5) A list of numbers separated by commas.

6) A list of answer names.

Thus we might put:

```
IQ GROUP := INTELLIGENCE < 60, 90, 110, 140, 500: MORON,STUPID,
                                AVERAGE,BRIGHT,GENIUS;
```

The following table gives the values which this polylog would take for several values of intelligence:

INTELLIGENCE	IQ GROUP
70	STUPID
130	BRIGHT
143	GENIUS
107	AVERAGE

The last value in this table is arrived at as follows:

1) The relation (**INTELLIGENCE < 60**) is not satisfied.

2) The relation (**INTELLIGENCE < 90**) is also not satisfied.

3) The relation (**INTELLIGENCE < 110**) is satisfied.

110 is the third number in the list, so the polylog is consequently assigned the third possible answer.

Another example of a derived polylog of this sort is:

```
EXAM PLACING := MARKS ≤ 40, 80, 100: FAIL, PASS, DISTINCTION;
```

The other use of derived polylogs is in grouping together various binary possibilities. For example, a person may give his country of origin as one out of several hundred possibilities, whereas what is required for the survey analysis is his native continent. The definition of this type, of derived polylog contains a list of Boolean expressions, each of which is true for at least one of the required groupings. The format is this:

1) The name of the polylog being defined.

2) The symbol '**:=**'.

3) A list of Boolean expressions, separated by commas.

4) A list of answer names, preceded by a colon, and terminated by a semicolon.

To provide an example, consider the raw polylog **COUNTRY OF BIRTH**,

to which the possible answers were **ENGLAND**, **SCOTLAND**, **WALES**, **IRELAND** and **ELSEWHERE**. For the purpose of analysis, it might be necessary to 'lump together' the Scots, Welsh and Irish. To do this we would define the derived polylog, '**ETHNIC GROUP**' as follows:

```
ETHNIC GROUP := ENGLAND, (SCOTLAND or WALES or IRELAND), ELSEWHERE:
                    ENGLISH, CELTIC, OTHERS;
```

When this type of variable is being evaluated, the Boolean expressions which define its value are worked out, in order, one by one, until one whose value is **T** is found. The position of the first 'true' expression in the list determines the value of the polylog, and when it has been found, all further expressions in the list are ignored. This implies two rules about the list of expressions:

1) At least one of them must always be true.

2) More than one expression may be true, but only the first is taken account of.

These rules exist so that whenever a grouping is to be '**CLASS A**, **CLASS B**,...., All other classes'.

then the expression indicating 'All other classes' may be written down simply as the Boolean constant **T**. For example:

```
PREMATURE HAIR CHANGE := (AGE < 40) and BALD, (AGE < 40) and WHITE, T:
                    HAIR LOST, HAIR WHITE, HAIR NORMAL;
```

Derived polylogs which correspond to grouped numerical quantities may also be written down in this format, but they will generally occupy more space. For example:

```
EXAM PLACING := (MARKS < 40), (MARKS < 80), T: FAIL, PASS,
                                                DISTINCTION;
```

A third use of derived polylog definitions is the grouping together of related binary questions. For example, suppose that (in spite of a recommendation given in a previous section) a questionnaire contained the following questions:

```
Q1)     Are you married?          ⎫    Answer
Q2)     Have you been married?    ⎬    Yes or
Q3)     Have you been divorced?   ⎭    No
```

From the Questions, the survey analyst might require the four mutually exclusive possibilities:

```
SINGLE
MARRIED
WIDOWED
DIVORCED
```

This could be defined as follows:

```
STATUS := not Q1 and not Q2, Q1, not Q1 and Q2 and not Q3, T:
          SINGLE, MARRIED, WIDOWED, DIVORCED;
```

As might be expected, the group heading for derived polylog definitions is: **derived polylog**;

1.10 The complete set of definitions.

Survey data presented to the MVC system is completely described by the set of raw and derived definitions. In writing them, it is necessary to observe certain precautions.

The same name must never be used for two different variables, or even for two different polylog answer names. (If this were done, expressions involving these names would be ambiguous). While the choice of different names for variables should never give any difficulty, some trouble might arise if there were many polylog questions which admitted of the answers '**unknown**' or '**other**'. To avoid ambiguity, we suggest that such categories be given titles such as '**UNKNOWN AGE**' or '**UNKNOWN INCOME**' or '**OTHER VEHICLE**'.

When each case is analysed, the definitions are invoked in the order in which they are written. It is therefore important that no expression should contain the name of any variable which has not yet been defined. This implies that raw definitions should be placed before derived definitions. If any derived definitions involve other derived variables — this is permissible — then these other derived variables must be defined first.

1.11 Errors and the checking facilities.

Large bodies of data, such as those supplied to survey analysis systems are highly prone to error. Although the effect of a small number of randomly scattered mistakes on the final results is not usually catastrophic, there is nevertheless every justification for ensuring that the data analysed is as free from error as possible.

The three major sources or errors in survey data are:

1) Incorrect answers to questions.

2) Incorrect punching of answers.

3) Malfunctioning of the computer itself.

Most of the work of avoiding errors should be done before the data is presented to the computer at all. There are many measures which lead to increased reliability. The main ones are unambiguous wording of questions, arrangement of the questionnaire so that the answers can be punched directly without intermediate 'hand copying', and verification of card punching.

The MVC system has two optional facilities for the detection of errors. Firstly, it can check that the format of each answer is valid and, secondly, it can check that each answer lies within reasonable limits, or is not logically contradicted by other answers.

If specified by the user, the MVC checking mechanisms are applied to each case after the values of its variables have been determined, but before it is used for tabulation. Whenever a case fails to pass these checks, it is ''rejected''. When this occurs, the relevant case number and all the values of the variables are printed immediately, and the case is not included in any subsequent tabulations.

1.11.1 The format check.

When the value of any variable is derived from a case record, its format is automatically examined for correctness. If it breaks the rules given in section 1.2, the variable is assigned the special value, 'wrong'. Some mistakes which would give rise to this value are, for instance, empty polylog fields on cards; or on tape, non-decimal characters within numbers, or polylog answers which do not correspond to any of the given formats.

The order to reject from the analysis any cases which contain 'wrong' values is given by the single word:

```
check;
```

1.11.2 The consistency checks.

The consistency checks enable variables whose values are legal in format to be compared against predetermined limits, or against the values of other variables. For example, a consistency check could be used to reject all subjects who claimed that their age was over 110, or vegetarians who gave their favourite food as beef.

Each check to be applied must be stated explicitly in the specification. It takes the form of a Boolean expression, preceded by the system word '**reject**'. The case is actually rejected if the expression is found to be true.

Examples of check statements are:

```
reject (NUMBER OF CHILDREN > 20);
reject VEGETARIAN and (BEEF or MUTTON or PORK or CHICKEN) ;
reject (HEIGHT < 48) or (HEIGHT > 84);
```

An alternative form of the checking statement consists of the word '**accept**' followed by a Boolean expression. In this case, the subject is rejected if the expression is found to have the value 'false'.

In accordance with their function, checking statements are best placed after any definitions. It should be stressed that unless it is explicitly specified, no checking, not even for 'wrong' values, takes place.

1.11.3 Computer faults.

A third possible source of errors in analysis is incorrect functioning of the computer itself. However, it is most unlikely that this cause could lead to incorrect results without giving some indication of the error. If the computer makes a mistake, it will usually stop. If any of the peripheral devices, such as card or tape readers, begin to malfunction, then the system will in all probability reject large numbers of cases which are in fact correct.

In practice, therefore, undetected computer errors are so rare that they can be conveniently forgotten.

1.12 Specifying tabulations.

All the previous sections of this manual have been concerned with the task of reading, editing, transforming and checking data, so as to prepare it for tabulation. We now reach the specification of the actual tables themselves.

The basic task of any tabulating system is to divide the cases presented to it into several classes, or categories, and to count the number of cases in each class.

The most elementary form of table description in MVC is simply a Boolean expression. This is called a 'table entry', and causes the number of cases in the class defined by the expression to be counted.

To indicate that a Boolean expression is actually a table entry, it must be included in a 'table specification'. This consists of a 'table heading', which contains general information about the table; a 'table body', which consists (in the simplest case) of a list of table entries preceded by the word '**count**'; and finally the table terminator '**finish**'.

The only mandatory item in the table heading is the 'title' This is a 'headline' which will eventually be attached to the table generated by this specification. In the heading, it is given as a quotation, preceded by the system word '**title**'. The other items which can be included in the table heading will be explained in due course.

A very simple table specification is given below. It refers to the Questionnaire first given on page 16. Its title is '**General Count**', and its task is to count the 'yes's' to all the non-numerical questions in this survey.

```
title    'GENERAL COUNT'
count    SPECTACLES, LEFTHANDED, RED, FAIR
         BROWN, BLACK, BALD, WHITE,
         ENGLAND, SCOTLAND, WALES, IRELAND, ELSEWHERE;

finish;
```

This specification will generate a table similar to this one:

GENERAL COUNT	
TOTAL	976
SPECTACLES	460
LEFTHANDED	70
RED	105
FAIR	233
BROWN	198
BLACK	201
WHITE	78
BALD	161
ENGLAND	579
SCOTLAND	166
WALES	100
IRELAND	64
ELSEWHERE	67

In this table, the entry '**BALD 161**' is a direct result of including the Boolean expression '**BALD**' in the table specification. It implies that out of the survey sample, 161 persons answered 'Bald' to the question 'Colour of Hair'. The extra line '**TOTAL 976**' represents the total number of cases admitted to the tabulation. In this case the total corresponds to the size of the survey sample, less any cases that were rejected. It should be stressed that the **TOTAL** is not the sum of the numbers in the column.

1.13 Short cuts.

In a practical survey analysis, so many different categories or groups must be counted that the use of some short-hand method of naming them is mandatory. The MVC system includes several condensed notations for this purpose.

1.13.1 Polylog names in tables.

The simplest shorthand notation is the inclusion of a polylog question name in the table. In this context, it is exactly synonymous to the list of its answers. Thus, the table specification:

```
title 'GENERAL COUNT';
count SPECTACLES, LEFTHANDED,
COLOUR OF HAIR, COUNTRY OF BIRTH;
finish;
```

would produce exactly the same table as the longer specification given in the previous section.

In a table, a polylog question name must stand alone; as it is not a Boolean particle, it must not be included in any 'expression'. For example, the construction

```
LEFTHANDED and COLOUR OF HAIR
```

is illegal.

1.13.2 The entry criterion.

There is a frequent need for tables which refer to a selected portion of a survey sample. For example, an investigator might be interested in the distribution of hair colour among — say — Scotsmen. This could be arranged by writing each of the table entries in such a way as to exclude all but the selected portion — e.g.,

```
count SCOTLAND and RED, SCOTLAND and BROWN, . . etc.
```

The shorthand way is to put an 'entry criterion' into the table heading. This consists of the symbol '**include**' followed by a Boolean expression. Its function is to admit only the cases defined by this expression to the subsequent table. The use of an entry criterion is analogous to the process of factorisation. Effectively, a common factor is removed from all the table entries and placed in the table heading instead.

The distribution of hair colour among Scotsmen should be tabulated by a specification such as:

```
title 'SCOTTISH HAIR COLOUR';
include SCOTLAND;
count HAIR COLOUR;
finish;
```

This would generate a table of the following appearance:

SCOTTISH HAIR COLOUR	
SCOTLAND	
TOTAL	166
RED	34
FAIR	26
BROWN	46
BLACK	17
WHITE	23
BALD	20

The entries in this table imply that there were 166 Scotsmen in the survey sample, of whom 34 had red hair, 26 fair, and so on.

It should be understood that the entry criterion is in no way complementary to the checking facilities. A case which fails to gain admission to one table is not thereby rejected from the survey; it may, in fact, be admitted to another table later on.

1.13.3 Simple table splits.

Another common requirement in table making is the comparison of corresponding statistics for two or more mutually exclusive groups. For example, we may require figures for — say — men as opposed to women, or for each of the social classes.

Sets of mutually exclusive tables of this type can be specified in a compact way by using 'table splits'. A simple table split consists of a list of mutually exclusive Boolean expressions, each of which defines one of the classes required. The table entries which follow apply to all the tables defined in the split, as does also any entry criterion which may be present.

Table splits are included in the table heading and are written as follows:

The system words '**tabulate by**'.

A list of Boolean expressions which must be mutually exclusive.

The terminating symbol ';'.

A typical table might be:

```
tabulate by PEDESTRIAN, CYCLIST. DRIVER;
```

Such a table split would cause the generation of three tables, relating to the classes '**PEDESTRIAN**', '**CYCLIST**' and '**DRIVER**'. These classes would have to be defined so that they were mutually exclusive.

It is usually possible to replace the list of Boolean expressions in a table split by a single polylog question name, which in this context is synonymous with its list of answer names. For example, the aim of the following table specification is an assessment of the prevalence of left-handedness and spectacle wearing among persons of different nationalities. The division of the survey population into different groups is accomplished by such a table split.

```
title        'QUALITIES BY COUNTRY';
tabulate by  COUNTRY OF BIRTH;
count        LEFTHANDED, not LEFTHANDED,
             SPECTACLES, not SPECTACLES;
finish
```

This specification would result in a table of the following appearance:

QUALITIES BY COUNTRY

	ENGLAND	SCOTLAND	WALES	IRELAND	ELSEWHERE	TOTAL
TOTAL	579	166	100	64	67	976
LEFTHANDED	68	34	3	19	2	126
NOT LEFTHANDED	511	132	97	45	65	850
SPECTACLES	312	76	10	18	44	460
NOT SPECTACLES	267	90	90	46	23	516

If the two mutually exclusive classes to be defined in a table split correspond to the 'yes' and 'no' answers in a binary question, the basic 'Boolean Expression' form of the table split is used. Thus we could put, for instance:

```
tabulate SPECTACLES, not SPECTACLES;
```

1.13.4 Parallel Tables.

Very often a survey analysis requires several tables or sets of tables, which are independent (i.e. not necessarily mutually exclusive), but which would, if written out in full, make use of the same table entries. For example, details of sweet and tobacco consumption may be required for men versus women, children versus adults, social class by social class, and so on.

A group of tables of this sort can be specified compactly by writing several table splits in one table heading. To follow up our example (for which we assume suitable variables to have been defined), the specification:

```
title       'CONFECTIONARY AND TOBACCO CONSUMPTION';
tabulate by (AGE<21), (AGE≥21);
tabulate by SEX;
tabulate by UPPER, MIDDLE, WORKING;
count       EAT SWEETS, EAT CHOCHOCOLATE, SMOKE CIGARETTES,
            SMOKE PIPE, SMOKE CIGARS;
finish;
```

would yield a set of tables as follows.

CONFECTIONARY AND TOBACCO CONSUMPTION

	(AGE<21)	(AGE≥21)	MALE	FEMALE	UPPER	MIDDLE	WORKING	TOTAL
TOTAL	40	63	52	51	14	70	19	103
EAT SWEETS	30	24	32	22	10	29	5	54
EAT CHOCOLATE	35	42	30	47	12	48	17	77
SMOKE CIGARETTES	10	54	40	24	8	38	18	64
SMOKE PIPE	1	1	9	1	5	4	1	10
SMOKE CIGARS	0	5	5	0	4	1	0	5

This completes the description of the condensed notations available in the MVC system. Great care should always be taken using these notations, because it is easy to make the error of specifying an unnecessary large and diversified table, even with a very short specification

1.14 Ratios (percentages).

In order to make the significance of figures in tables clearer, they are sometimes expressed as ratios, instead of absolute quantities. When this is done, it is common practice to multiply the ratios by some constant, such as 100 or 1000, so as to convert them into percentages or 'parts per thousand'.

The usefulness of this technique is shown in these two tables:

	AGE < 21	AGE ≥ 21	TOTAL
EAT SWEETS	30	24	54
DON'T EAT SWEETS	10	39	49
TOTAL	40	63	103

1. Raw totals

	AGE < 21	AGE ≥ 21	TOTAL
EAT SWEETS	75	38	52
DON'T EAT SWEETS	25	62	48
TOTAL	100	100	100

2. Percentages

The raw totals in an MVC table can be converted into ratios in four different ways, which differ in their choice of 'reference quantities'. The totals can be formed as fractions of

a) The total number of cases in the survey (the 'total ratio');

or b) The total number of cases admitted to this group of tables (the 'table ratio');

or c) The column total (the 'column ratio');

or d) The row total (the 'row ratio').

The total ratio and the table ratio are only essentially different if the table under consideration has an entry criterion. The column ratio and the row ratio are only applicable to groups of tables formed by table splits.

The significance of these different types of ratio can be illustrated by reference to the table on page 41.

Entity	Value	Meaning
Raw entry	76	Our sample contains 76 Scots who wear spectacles.
Total ratio	76/976 = 0.08	8% of the sample consists of bespectacled Scotsmen.
Column ratio	76/166 = 0.46	46% of all Scotsmen in the sample wear spectacles.
Row ratio	76/460 = 0.17	17% of all the people in the sample who wear spectacles are Scottish.

When the tables are prepared, the ratios may be written alongside the raw totals from which they are derived, or they may be placed in a separate table. The MVC system makes it impossible to suppress the raw totals entirely, as is done with some methods of survey analysis. In general, this practice is to be

heavily discouraged, for it may lead to assertions such as

'100% of red-haired left-handed Scotsmen wear glasses',

whereas an inspection of the raw totals would have shown that the survey sample contained only one red-haired left-handed Scotsman.

The conversion of tables into ratio form can be specified by introducing appropriate sentences into the table headings. They take the form:

1) The word **ratio** followed by solidus ('/').

2) One of the words **total**, **table**, **row** or **column**, followed by a comma.

3) The multiplying constant, followed by a comma.

4) One of the words **combined** or **separate**, followed by a semicolon.

The multiplying constant will, in most cases, be 100. It may, however, be any positive whole number.

The words **combined** and **separate** indicate the way in which the table is to be punched.

When ratios are printed, the quantity 0/0 is represented by '–'. The use of the ratio facility is illustrated below:—

```
title 'TRANSPORT HABITS';
include VEHICLE OWNERS;
tabulate by TYPE OF VEHICLE;
ratio/row, 100, combined;
ratio/table, 1000, separate;
count AGE GROUP;
finish;
```

This will produce:

TRANSPORT HABITS

	BIKE	MOTORCYCLE	CAR	TOTAL
TOTAL	57	117	380	554
	10	21	69	100
UNDER 21	42	89	73	204
	21	44	35	100
UNDER 40	12	20	180	212
	6	9	85	100
OVER 40	3	8	127	138
	2	6	92	100

RATIO/TABLE, 1000

	BIKE	MOTORCYCLE	CAR	TOTAL
TOTAL	103	211	686	1000
UNDER 21	76	161	131	368
UNDER 40	22	36	325	383
OVER 40	5	14	230	249

1.15 Numerical data.

In general, it is possible to use quantitative data (expressed as numerical variables) in a more sophisticated way than qualitative information (expressed as polylog or binary variables). For example, it is possible to derive such statistics as averages and correlations. This section describes the way in which such statistics can be included in table specifications.

Tables in which the individual entries are averages, standard deviations, and so on, are specified in the same way as tables in which the entries are simple counts of non-numerical attributes; the only difference is that instead of the word '**count**', followed by a list of binary and polylog names, we put one of the numerical specifiers, followed by a list of numerical variables. Each table specification may include several such lists.

There are 4 numerical specifiers:

1) '**mean**' which generates the average value of the given variables;

2) 'stdev' which generates the standard deviation of the named variables;

3) '**sum**' which forms the total sum of all the variables, taken over all the cases admitted to that tabulation;

4) '**correlation**' which produces the correlation coefficients between all possible pairs of the listed variables.

As a general example, suppose that we are dealing with a survey taken among children aged 10 — 12, in which a polylog variable is **SEX** (i.e. **MALE or FEMALE**), and in which the numerical variables are **AGE**, **HEIGHT** and **WEIGHT**. Then, this table specification:

```
title       'NUMERICAL STATISTICS';
count        SEX;
mean         AGE, HEIGHT, WEIGHT;
stdev        AGE, HEIGHT, WEIGHT;
correlation AGE, HEIGHT, WEIGHT;
finish;
```

will produce:

```
      NUMERICAL STATISTICS

TOTAL      770
MALE       290
FEMALE     480

MEAN

AGE         11.1
HEIGHT     162.7
WEIGHT      46.9

STDEV

AGE         0.72
HEIGHT      5.60
WEIGHT      7.77

CORRELATION    AGE      HEIGHT

HEIGHT       +0.7372
WEIGHT       +0.8197   +0.9473
```

The 'table split' facility can be used with numerical tables. If such a table contains a '**correlation**' statement which names more than

two numerical variables (i.e., which specifies more than one correlation coefficient) then the resultant tables will not be printed side by side, but below one another, for obvious reasons. In the case of numerical tables which are printed side by side, the 'totals' column on the right contains statistics computed from the sums of the raw totals in the corresponding lines. These represent a kind of weighted average.

For example, the table specification:

```
title         'NUMERICAL TABLE SPLIT';
tabulate by   SEX;
mean          HEIGHT, WEIGHT;
finish;
```

will produce:

NUMERICAL TABLE SPLIT			
TOTAL	MALE	FEMALE	TOTAL
TOTAL	290	480	770
MEAN			
HEIGHT	177.4	153.8	162.7
WEIGHT	47.6	46.5	46.9

In the examples above, it will be noticed that means are printed to one, standard deviations to two, and correlations to four decimal places. This is the 'standard format'. The actual format obtained can be varied at will by including the phrase '**format n**' at the end of statements which specify numerical statistics. '**n**' is a number which indicates the actual number of decimal places required.

For example, the statement:

```
mean HEIGHT, WEIGHT, format 4;
```

would cause the average values of **HEIGHT** and **WEIGHT** to be printed to an accuracy of 4 decimal places.

In using the **format** facility, it is worth remembering that the ATLAS computer performs its calculations to an accuracy of about 1 part in 10^{10}. This means that, for example, it is useless to ask for 10 places of decimals in quantities which are expected to be large.

The omission of a format specification implies that standard format is required.

The numerical specifier **sum** is intended to assist the computation of advanced statistics. Suppose, for example, that each case in a survey contains two numerical values for variables called '**EX**' and '**WY**'. (Mathematically we shall refer to these as 'x' and 'y'). It is required to fit, by least squares, a parabolic regression curve of the form y = ax2 + bx + c to the points given by the various values of the variables.

Now a, b and c can be found by solving the equations

$$\begin{Bmatrix} \Sigma x^4 & \Sigma x^3 & \Sigma x^4 \\ \Sigma x^3 & \Sigma x^3 & \Sigma x \\ \Sigma x^2 & \Sigma x & 1 \end{Bmatrix} \begin{Bmatrix} a \\ b \\ a \end{Bmatrix} = \begin{Bmatrix} \Sigma x^2 y \\ \Sigma xy \\ \Sigma y \end{Bmatrix}$$

As MVC is not an equation solving programme it cannot generate a, b, and c, directly; but it can be used to find coefficients in the equations by doing the necessary summations. This could be done as follows:

1) Among the derived variable specifications we put:

```
EX2   := EX  * EX;
EX3   := EX2 * EX;
EX4   := EX3 * EX;
EXWY  := EX  * WY;
EX2WY := EX2 * WY;
```

2) In a subsequent table specification we put:

```
sum EX, EX2, EX3, EX4, WY, EXWY, EX2WY;
```

The resultant table will then contain the raw sums necessary for fitting the regression curve. The ∑1 term is of course given in the totals line at the head of the table.

1.16 The complete MVC text.

The various items which go to make up an MVC survey specification have all been described in previous sections. They include raw definitions, derived definitions, checking statements, table specifications, and such oddments as case number declarations, and announcements of the type of input medium to be used. When these items are combined into a complete specification, certain conventions must be followed.

The specification must start with a 'job description'. This consists of several lines of information about the program, and is used by the ATLAS operating system. (Its final form is not yet decided).

The first item after the job description must be a type announcement; e.g., '**read characters**'.

The type announcement should be followed by the raw definitions, under their respective headings. It is not necessary to include all definitions of one type in one block; but it is essential that each definition should be preceded by an appropriate heading. Thus the following configuration is legal:

```
raw numerical;
_____
_____          raw numerical definitions.
_____
raw polylog;
_____
_____          raw polylog definitions.
_____
raw numerical;
_____
_____          more raw numerical definitions.
_____
```

If the 'check' facility for wrongly punched data is invoked, the word '**check**' should come after the raw definitions.

Next come the derived definitions. It is important to define any variable only in terms of other variables which have already been defined. The remarks about grouping raw definitions also apply to derived definitions.

Next come the checking statements other than '**check**'. Any number of these may be included.

The checking statements are followed by any number of table specifications. It is sometimes convenient to precede a table specification by a derived definition specially needed for that table; this is permissible.

The specification should end with the following three lines:

```
start;
***Z
\\\\\ ( Erases )
```

Section 1.18 contains a complete example of an MVC specification and the resultant tables.

1.17 A note on the general output format.

Tables generated by the MVC system are always printed by a Line Printer, on sheets 120 characters wide.

The line printer has a somewhat restricted set in comparison to the flexowriter; in particular, it cannot print any lower case letters. All names therefore appear in upper case, irrespective of the way they were written in the specification.

The limited width of the page does not form a restriction on the size of the tables which may be formed. If necessary, the MVC system will automatically split a wide table into two or more sections, which are printed separately but with identifying marks, and which may be stuck together with sellotape.

1.18 A Complete Example

In this final section we shall give a complete example of a simple survey and its analysis.

This imaginary survey is concerned with general habits of potato crisp eating. We wish to discover how variables such as people's favourite brands of crisps, their preferred places for eating them, and the amount they consume are related to such personal characteristics as age and social class.

The variables which are to be determined for each subject are:

1) Social class
2) Age
3) Sex
4) Whether married
5) Whether any children in subject's household

(1–5: Personal Details)

6) Number of packets of potato crisps bought every week
7) Number of packets of crisps eaten per week
8) Favourite places for eating crisps, out of the following
    a) In public houses;
    b) While travelling;
    c) With meals;
    d) Watching television;
    e) In the bath;
    f) At work;
    g) Elsewhere.
9) Favourite brand of crisps, out of Smyth's, Lyon's, Blogg's, or any other.
10) Reasons for eating crisps, out of the following possible ones:
    a) Hunger;
    b) To pass the time.

The first stage in designing an appropriate Questionnaire is to assign suitable types to all the questions. The most suitable for the majority is immediately obvious; thus, questions 2, 6 and 7 are numerical, because they require numbers for answers; 4 and 5 are binary, since they must be answered by 'yes' or 'no'; and questions 1 and 3 must be polylogs, because each subject can only belong to one class or sex.

In answering question 8, the subject may indicate none, one or several of the places listed as those where he habitually eats potato crisps. It follows that this question cannot be treated as a polylog: it must be coded as a group of independent binary questions. The same considerations apply to question 10.

In deciding the type of question 9, we shall neglect the rare occurrence of a person having two or more favourite brands. If we word the question so that each subject must indicate exactly one brand as his personal favourite, then we can treat the question as a polylog.

The results of our preliminary question classification are:

Numerical	polylog	binary
2, 6, 7	1, 3, 9	4, 5, 8, 10

The next stage is to decide on the medium of data representation, and on the

eventual layout of the data. Since we expect to interview about 7000 subjects, the only possible medium is punched cards. (For this quantity of data, paper tape would be grossly unwieldy).

We note that the quantity of data for each case is such that it can easily fit on to one card. We do not therefore need a card-in-case number. Following common convention, we place the case number at the beginning of the card. Since the highest case number will be about 7000, 4 columns are sufficient.

Having filled in the case number, we assign fields to all the questions in the survey. The numerical questions are all such as to need two decimal digits for their answers, so we allot them two columns each. To all the other questions we assign one column each. The fields for each of the polylog questions start at holesite 0; they could equally well have started at holesite 1. In this Questionnaire, we follow the practice of punching related groups of binary questions on to one column. In cases where there is little room to spare on a card, this technique can save a considerable amount of space; but in the present instance, the answers could well have been spread out, one to a column.

The columns are used up, in the same order as the corresponding questions are asked. This ensures that, when coding the data, the card punch operator can work straight down the page.

The complete Questionnaire, which includes suitably worded questions and the result of our card layout decisions, is shown below: note that it includes the category 'Don't know' wherever this is likely to be encountered.

BLOGGS POTATO CRISP SURVEY

	Case number		3 1 4 2		Cols. 1-4
1.	What is your social class?	a) Upper			5/0
		b) Upper middle		✓	5/1
		c) Lower middle			5/2
		d) Working			5/3
2.	What is your age?				Cols6,7
3.	What is your sex?	a) Male			8/0
		b) Female		✓	8/1
4.	Are you married?			✓	9/0
5.	Are there any children in your household?			✓	10/0
6.	How many packets of crisps did you buy last week?		3 4		Cols.11,12
7.	How many packets of crisps did you eat last week?		0 7		Cols.13,14
8.	Please mark any of the following places in which				
8.	you often eat crisps:	a) In pubs			15/0
		b) While travelling			15/1
		c) With meals		✓	15/2
		d) Watching television		✓	15/3
		e) In the bath			15/4
		f) At work			15/5
		g) Anywhere else			15/6
9.	Which out of the following is your favourite brand of crisp? Please indicate one only.				
		a) Smyths			16/0
		b) Lyons			16/1
		c) Bloggs		✓	16/2
		d) Other			16/3
		e) Don't know			16/4
10.	Please mark any of the sentences below which you think apply to you.				
		a) I eat crisps because I am hungry			17/0
		b) I eat crisps to pass the time.		✓	17/1
		c) I do not know why I eat crisps.			17/2
		d) I do not eat crisps.			17/3

While data is being collected and coded, the MVC specification itself can be drawn up. A possible specification for the potato crisp survey is shown below:

```
JOB                                  ┐
BLOGGS CRISP MARKET SURVEY           │
INPUT                                │
1 BLOGGS CRISP DATA                  │
OUTPUT                               ├  Tentative job description
0 LINE PRINTER 500 LINES             │
TAPE                                 │
1 BLOGGS DATA                        │
COMPILER MVC                         ┘
READ NEW SPECIFICATION
read cards;
raw numerical;
AGE, 6, 2;
CRISPS BOUGHT, 11, 2;
CRISPS EATEN, 13, 2;
raw polylog;
CLASS , 5/0, 4: UPPER, UPPER MIDDLE, LOWER MIDDLE, LOWER;
SEX, 8/0, 2: MALE, FEMALE;
FAVOURITE, 16/0, 5: SMYTHS, LYONS, BLOGGS, OTHER, D K;
raw binary;
MARRIED, 9/0;
CHILDREN, 10/0;
PUBS, 15/0; TRAVEL, 15/1; MEALS, 15/2; TV, 15/3;
BATH, 15/4; WORK, 15/5; ELSEWHERE, 15/6;
HUNGRY, 17/0; PASST1ME, 17/1; DONTKNOW, 17/2; DONT EAT, 17/3;
case number 1, 4;
check;
derived numerical;
CRISPS GIVEN := CRISPS EATEN - CRISPS BOUGHT:
derived polylog;
CONSUMPTION := CRISPS EATEN ≤ 0,3,10,100 : NONE, FEW, SOME, LOTS;
derived binary;
FAMILY := MARRIED and CHILDREN;
reject (AGE < 16) and MARRIED;
reject (AGE < 13) and WORK or PUBS);
reject (CRISPS EATEN > 40);
reject (CRISPS EATEN ≠ 0) and DONT EAT;
title 'GENERAL';
count CLASS, SEX, FAVOURITE,
      MARRIED, CHILDREN, PUBS, TRAVEL, TV, BATH,
```

```
      WORK, ELSEWHERE, HUNGRY, PASSTIME, DONTKNOW, DONT EAT;
finish;
title       'ADOLESCENT CRISP CONSUMPTION';
include     (AGE >= 13) and (AGE <= 19);
tabulate by CONSUMPTION;
count       CLASS, SEX;
finish;
title       'FAVOURITE BRANDS?;
tabulate by SMYTHS, LYONS, BLOGGS;
count       PUBS, TRAVELLING, MEALS,
            TV, BATH, WORK, ELSEWHERE;
finish;
title       'REASONS FOR EATING CRISPS';
tabulate by SEX;
tabulate by (AGE < 21), (AGE >= 21);
tabulate by SOME, FEW, LOTS;
count       HUNGRY, PASSTIME, DONTKNOW;
finish;
title       'TABLE WITH PERCENTAGES';
ratio/row   100, combined;
ratio/table 100, separate;
include     (AGE <= 30);,
tabulate by CONSUMPTION;
count       MARRIED, CHILDREN;
finish;
title       'USE OF NUMERICAL VARIABLES';
tabulate by SEX;
mean        AGE, CRISPS BOUGHT;
stdev       AGE, CRISPS BOUGHT, format 4;
correlation AGE, CRISPS EATEN;
finish;
start;
***Z
\\\\\\\\\\\\
```

As indicated in the previous section, the specification begins with a 'Job Description'. Since the final form is not known at the time of writing, the Job Description in this example is only intended as a broad illustration and must on no account be taken as a model for an actual text.

The raw definitions are entirely straightforward. The only point worth noting is that the 'don't know'' answers to questions 9 and 10 are assigned the different names "**D K**" and "**DONT KNOW**". It would have been illegal to use precisely the same names for both answers. An alternative pair of names would have been "**NO FAVOURITE**" and "**NO REASON**".

In general. the names assigned to variables and their answers appear on

tables generated by the system. In view of this names should be as self-explanatory as possible. If the tables are to be published without further editing (e.g., by a photographic process), it is best to avoid colloquial names such as

"**CLOT**" for **PHLEGM and (CIGARETTES PER DAY > 20)**

The derived definitions are used to specify variables which are to be used in tabulations. Thus, the derived numerical variable **CRISPS GIVEN** indicates the number of crisp packets given away by the subject, and the derived polylog **CONSUMPTION** is a 'grouped' version of the raw numerical variable **CRISPS EATEN**. Its values are:

CRISPS EATEN	CONSUMPTION
0	NONE
1-3	FEW
4-10	SOME
11+	LOTS

Since this text contains only a selection of the tables which would actually be specified, not all the derived variables are actually used.

The checking statements are intended to reject children who claim to be married, or to eat crisps in pubs or at work; subjects who state that they eat an unlikely number of crisps every week; and those who give contradictory answers to questions 7 and 10.

Finally, the tables actually produced by the individual tables specifications are quoted in full below. It is hoped that these are sufficiently self: explanatory to avoid the need for any comment.

GENERAL	
TOTAL	7641
UPPER	242
UPPER MIDDLE	4731
LOWER MIDDLE	2634
LOWER	34
MALE	4017
FEMALE	3624
SMYTHS	7112
LYONS	417
BLOGGS	27
OTHER	5
D K	70
MARRIED	4139
CHILDREN	5007
PUBS	4320
TRAVEL	394
MEALS	612
TV	1048
BATH	7
WORK	2507
ELSEWHERE	3021
HUNGRY	7244
PASSTIME	2891
DONTKNOW	318
DONTEAT	2007

ADOLESCENT CRISP CONSUMPTION

(AGE > 13) and (AGE < 19)

	NONE	FEW	SOME	LOTS	TOTAL
TOTAL	225	310	144	63	742
UPPER	10	50	32	1	93
UPPER MIDDLE	212	197	93	37	539
LOWER MIDDLE	3	86	14	22	125
LOWER	0	7	5	3	15
MALE	53	120	101	40	314
FEMALE	172	190	43	23	428

FAVOURITE BRANDS

	SMYTHS	LYONS	BLOGGS	TOTAL
TOTAL	7112	417	27	7556
PUBS	4302	14	0	4316
TRAVELLING	380	1	13	394
MEALS	610	0	0	610
TV	613	410	25	1048
BATH	7	7	0	14
WORK	2444	57	3	2504
ELSEWHERE	3015	5	0	3020

REASONS FOR EATING CRISPS

	MALE	FEMALE	TOTAL	(AGE<21)	(AGE≥21)	TOTAL	SOME	FEW	LOTS	TOTAL
TOTAL	4017	3624	7641	1037	6604	7641	1072	4316	1119	6507
HUNGRY	3612	3632	7244	994	6250	7244	916	4217	1083	6216
PASSTIME	1437	1454	2891	417	2474	2891	107	2016	224	2347
DONTKNOW	212	106	318	37	281	318	31	117	69	217

TABLE WITH PERCENTAGES

RATIO/ROW, 100

(AGE ≤ 30)

	NONE	FEW	SOME	LOTS	TOTAL
TOTAL	101	1817	821	1073	3812
	3	48	22	28	100
MARRIED	82	800	419	716	2017
	4	40	21	35	100
CHILDREN	7	800	554	1071	2432
	0	33	23	44	100

RATIO/TABLE, 100

	NONE	FEW	SOME	LOTS	TOTAL
TOTAL	3	48	22	28	100
MARRIED	2	21	11	19	53
CHILDREN	0	21	15	28	64

USE OF NUMERICAL VARIABLES

	MALE	FEMALE	TOTAL
TOTAL	4017	3624	7641
MEAN			
AGE	35.7	29.2	32.6
CRISPS BOUGHT	5.3	15.7	10.2
STDEV			
AGE	10.3124	13.4617	11.8061
CRISPS BOUGHT	6.1234	3.4051	4.8342
CORRELATION	AGE	AGE	AGE
CRISPS EATEN	+0.5312	-0.0317	+0.2642

Appendix 1. The Flexowriter.

The device most commonly used to prepare punched tape is the flexowriter, or one of the similar machines manufactured by other firms. The flexowriter has a keyboard similar to that of a typewriter. When operated, it produces not only a printed copy of the matter typed, but a length of punched tape on which each character is represented by a row of holes. The keyboard also has several 'control buttons', which do not produce a printed mark, but cause the machine to take other action instead. They are:

1) Upper case. This sets the platen so that upper case character will be typed
2) Lower case. This is complementary to upper case.
3) New line. This returns the platen to its extreme righthand position and advances it by one line.
4) Backspace. This moves the platen back by one space.
5) Tabulate. This moves the platen forward to the next tabulation stop, wherever that may be.

In addition to these, the flexowriter has a key known as 'erase'. This character is printed, and looks like this: ////. It is commonly superimposed on another character with the aid of the 'backspace' key.

All these special 'characters' are represented by rows of holes on the punched tape.

The details of tape punching machines vary from installation to installation, and should be ascertained on the spot.

The tape produced by a flexowriter is read by the MVC system, the following rules always apply:

1) With the exception noted below, each line of typing is always interpreted according to its printed appearance, full account being taken of backspaces. This rule does not hold if 'tabulate' is followed by a series of ''backspaces', because the computer cannot reconstruct the line. The use of 'tabulate' is therefore strongly frowned upon; but if used, it is equivalent to a single space.
2) The character 'erase', whether by itself or combined with any other character, is always ignored.
3) Characters may be superimposed on themselves without change in meaning.
4) 'Underline' by itself is synonymous with 'space'.
5) Any continuous sequence of spaces or its synonyms is equivalent to one space.

Appendix 2. Arithmetical Expressions

Arithmetical expressions are constructed with numerical constants, previously defined variables, brackets, function designators, and the five operators

| (exponentiation), '/' (division), '*' (multiplication), '+' (addition), and '-' (subtraction). The expression may be of any desired complexity, but must obey the syntactical rules for the construction of arithmetical expressions. In particular, multiplication must be stated explicitly.

A function designator is always followed directly by its argument in brackets. Multiple arguments are separated by commas.

Arithmetical expressions are evaluated in the following order:

1) The contents of brackets (this is to be understood recursively).
2) Functions and initial minus.

3) Exponentiations.

4) Divisions.

5) Multiplications.

6) Additions and subtractions.

Evaluations in each group take place from right to left. There follows a list of functions which are available.

Designator	Meaning
C (Boolean exp.)	**C(T) = 1; C(F) = 0;**
log (x)	$\log_e(x)$
exp (x)	e^x
antilog (x)	
sqrt (x)	The positive square root of x
mod (x)	sqrt (x^2)
intpt (x)	(the integral part of x) if x > 0, and (the integral part of (-x)) if x < 0.
frpt (x)	(the fractional part of x) if x >=0, and (the fractional part of (-x)) if x < 0.
sin (x)	sine (x). x must be in radians.
cos (x)	cosine (x). x must be in radians.
tan (x)	tangent (x). x must be in radians.
arcsin (x)	$\sin^{-1}$ (x). $-\pi/2 \leq$ arcsin (x) $< \pi/2$.
arctan (x)	$\tan^{-1}$ (x). $\pi/2 <=$ arctan (x) $< \pi/2$.
max (x1,x2,...)	The largest argument.
min (x1,x2,...)	The smallest argument.
whichmax (x1,x2,...)	The position of the largest argument.
whichmin (x1,x2,...)	The position of the smallest argument.

The meanings of the last four functions are illustrated in these examples

```
max(3, 17, -89, 0)  = 17;
min(44, 57, -78, 2) = -78;
whichmax(3, 17, -78, 0)  = 2;
whichmin(44, 57, -89, 4) = 3,
```

Numerical constants must be written in one of the following formats:

a) integer

b) integral part. fractional part

In particular, any kind of 'floating point' format is barred. Sequences of the form **'(0.87 * 10 | 5)'** should be avoided, as they will be evaluated every time the expression containing them is used. It is preferable to write **'87000'**.

INDEX

GLOSSARY OFSYSTEM WORDS