APPENDIX 3. MASH MACHINE READABLE CODEBOOK STANDARD

Introduction

This appendix describes a machine readable codebook standard for the MASH system, discusses its implementation on the PDP-10 computer, and contains an example of codebook content and output format.

The standard which has been defined and is described here is somewhat more general than what is required to support the operation of the MASH system. It is believed that this standard is applicable to a substantial number of micro data files, and that it will find application not only within MASH but also by other systems as well as being useful for defining and using other data files. While this appendix does not attempt to provide a rigorous definition of the standard, such a definition appears in MASH project documentation.

The concepts and standard described here have been motivated by an increasingly common requirement in social science computing to process larger and structurally more complex data files. Concepts in this standard have been developed to deal with problems encountered in defining and processing such files. Other concepts have been borrowed from parallel work in social science computing. The proposed standard represents a compromise of ideals; more general data structures and value structures have been sacrificed in order to achieve simplicity of representation and reasonably simple implementation of codebook processing programs. One important restriction inherent in the standard is that it is only applicable to micro populations, i.e. collections of observations on individual units. The units themselves, of course, will generally be aggregations of more micro observations, e.g. state data is aggregated county data, and individual earnings on a yearly basis is an aggregation of the monthly or weekly flows.

The problems inherent in providing adequate data file documentation are not new problems, but present solutions are not very satisfactory. The growth and success of commercial data processing has depended upon the development of adequate data file documentation for its purposes, and initial higher level commercial data processing languages such as Comtran and Cobol included data file definition sections. The data division of a Cobol program contains a basic machine readable codebook describing the files that are read and written by the program. Codebooks have also been a basic processing tool for survey files generated by survey research organizations; however these codebooks have until recently been used primarily for the data coding process only.

Data file definition and documentation in social science research has generally received lower priority than obtaining statistical and tabular outputs from data files. Most individual research in the social sciences has involved only a moderate amount of data which has not required thorough documentation. The active life of such data files has generally been limited to the duration of the researcher's interest in it. Researchers have generally not given high priority to maintaining their data sources and providing accessibility to them; indeed, as argued in [S9], there has been little professional or other incentive to do so.

A number of recent trends have made the systematic processing and documentation of social science data files more important. There has been a gradual but steady increase in both the number of data files that are becoming available and relevant for social research, and the size of these files is increasing. Private data sources, government agencies, and universities and data archives have all contributed to this increase in the supply of machine readable information. Concurrently, the demand for this information is increasing, and the number of social scientists and policy makers who are learning about the use of computers is growing. More complex and larger data files are being judged useful for research, and the uses to which they are put are becoming more intricate. Finally, although the members of the social science research community have become more mobile and their research interests and outputs more interconnected, the computers, data sources, and software available to them have remained relatively rigid and immobile.

The increase in the supply of potentially useful social science data and the growth in demand for access to it has encouraged the development of machine readable documentation of data files for two important and quite different purposes: (1) the retrieval of information *about* data files; and (2) the retrieval of information *from within* data files. An example of the first purpose is the retrieval of the *text* of certain questions asked in the course of a series of sample surveys. These questions might then be scanned for non-trivial key words and then catalogued by key word as a guide to retrieval of questions dealing with specific concepts or objects. The second purpose is concerned with retrieval of *data values* from a data file using the codebook as an aid in retrieval; the codebook is referenced to obtain the location, length, mode, values, and special conditions for the item to be retrieved. Both purposes rely upon a codebook document that describes the data file in some detail.

As the amount of available social and economic data increases, the demand for disseminating detailed information about the data also increases. Textual information included within a machine readable codebook is well suited to this requirement. It can be classified, sorted, indexed, and searched for occurrences of specific subjects, categories, responses, areas of application, interview techniques or survey units, thus providing a comprehensive and current compilation of existing data sources. The text portion of a data file codebook can be made dynamic by establishing a commentary section in which the pertinent observations of its users are entered as they gain experience with the data file.

The growth in use of machine readable data files by social scientists has encouraged the development of more efficient methods of accessing these files and transferring them between computer installations. Historically, the production and use of large data files has been individualistic rather than systematic, and the transfer of both data files and processing programs between computer installations has often been moderately difficult at best. Standardization of machine readable documentation for micro data files would simplify these operations to some extent; the codebook would standardize the presentation of information (the codebook file itself) about other information (the data file). It is also both possible and desirable to simplify the data storage and retrieval functions of computer programs by building mechanisms to access data file contents indirectly through that file's codebook. Other advances in computing practice that are realizable are on-line source data capture, consistency checking by codebook, semi-automatic data cleaning procedures, and automatic output generation. All of these procedures are, of course, limited by the amount of detailed information that the codebook contains.

The development of the MASH machine readable codebook standard was initially encouraged by the activities of the Council of Social Science Data Archives and by work done at The Brookings Institution on the development of the BEAST language [B2]. An initial implementation of a codebook standard was developed in 1967-68 at Brookings to deal with the problem of producing versions of the 1966 and 1967 Surveys of Economic Opportunity files that would be suitable for microanalytic research purposes [S2]. Codebooks developed using the initial standard were used successfully for many of the applications listed above. The present standard was developed in 1970, and resulted from a major modification to the initial standard; the modification itself was prompted both by the use of machine readable codebooks in MASH and by the increased uses such codebooks might enjoy.

A set of programs have been written for performing basic manipulations with codebook files. One program allows its user to build a codebook file interactively; another reformats the information in a codebook file into a format that is easy to read and therefore suitable for printing and for file documentation. Furthermore, since codebook files are sequential "line" or "card image" in structure, any procedure or program available to manipulate such files may be applied to codebook files.

Data File Structures

Micro data files consist of collections of data on individual units of observation, often called *enumeration units.* Some examples of enumeration units used in social science research are families, persons, manufacturing plants, schools, states, commodities, transactions, and standard metropolitan statistical areas. Terms in current use that have a meaning similar or identical to enumeration unit include "population unit," "unit of analysis," "observation," "case," and "entity."

For every micro data file, there are a set of *attributes* associated with the enumeration unit of the file. For example, a survey of persons might obtain values for the person attributes age, race, sex, address, income, and marital status. Attributes of a survey of manufacturing plants include number of employees, weekly manufacturing payroll, value of shipments, major product identification, and the state in which it is located. Terms in current use having a meaning similar or identical to attribute include "variable," "measurement," and "characteristic."

Each attribute of an enumeration unit has a range of *values*, any one of which it may take on. This set of values is the *domain* of the attribute. For example, the attribute "age of person, measured in years" may assume the values 1, 2, ... , 99 in many sample surveys. Attributes may take on a fixed, discrete set of values, or they may be continuous attributes such as income.

Using the above terminology, a micro data file consists of a structured set of values assumed by a collection of attributes for a population of enumeration units.

The simplest type of file structure for micro data is the *rectangular file structure.* Such a structure may be thought of as a rectangular array, or matrix, of values. The data values in each row of the array correspond to the values of all attributes for one enumeration unit, and the data values in each column of the array are the values taken on by one attribute as it ranges over the collection of enumeration units. Much sample survey data generated by survey research centers is properly classified as micro data having a rectangular data structure.

Although many social science data files are either rectangularly structured or can be transformed into a rectangular structure with little effort, there exist an increasing number of files for which this transformation cannot be performed without introducing substantial inefficiencies in both the cost of retrieving and processing the data and in the quantity of computer memory resources used. For example, a sample survey that gathers data both on a family and a person basis cannot be restructured rectangularly in an efficient manner because the number of persons in a family varies among families. The information could be separated into two rectangular files, but at the cost of losing efficient links between persons and families. Although the enumeration unit of the survey is the family, the data file contains collections of data on two different entities.

The concept of *segment* provides the additional power to define more general file structures. A *segment* is a collection of attributes which are related in some substantive or organizational manner to the information within an enumeration unit. Each attribute of the enumeration unit is contained in exactly one

segment type; this containment is often expressed using the possessive case, e.g. age of person, income of family, and location of school. Within a data file, information may be subdivided into one or more segment types, and segment types can occur one or more times within an enumeration unit. One segment type, often referred to as the *root segment,* can occur only once, and its occurrence in the order of a sequential data file signals the end of information for the preceding unit of enumeration, and the beginning of information describing a new enumeration unit.

In the above example of the sample survey that obtained data about both families and persons, an appropriate organization of data would include two segment types, a family segment type and a person segment type. The data for an enumeration unit would then consist of one occurrence of a family segment type, and a variable number of occurrences of person segment types. Each segment type would necessarily contain an attribute that would allow its type to be identified by a processing program. One of the attributes in the family segment might be "number of persons in family."

The above example describes a simple *tree structured* file. Each enumeration unit contains one *root* or *main segment* called "family," and a variable number of second level segments called "person." There exists a *flag* attribute in each segment within the data file that identifies the type of each segment. In addition, there is a *key* attribute in each level 1 segment (family) whose value is equal to the number of level 2 (person) segments associated with that level 1 segment. Tree structured, or *hierarchical,* files are a common form of complex structure used in organizing social science data.

The logical organization of data within a data file, i.e. the *file structure*, should be distinguished from the *physical ordering* of data on the particular storage medium that the data occupy. For example, rectangular files resulting from sample surveys are generally stored sequentially by row, but for certain applications it may be useful to transpose them and store them by column or group them according to a hierarchical pattern. Enumeration units in tree structured files are often stored in *left list order,* i.e. the order that is obtained when the branches of the tree are traced continuously from left to right; however, the segments might equally well be stored sequentially in order of decreasing segment level. Alternatively, the tree structured file might be restructured into a series of rectangular data arrays with inter-array links provided by a number of auxiliary arrays constructed from the file for this purpose. There is a relationship between the logical organization of the data in a file, its physical ordering and representation, the amount of structural information (such as keys and flags) required, and the major uses to which the data will be put.

The number of character positions required to contain all of the data in a segment type often exceeds the length of a typewriter line or the number of columns in a punch card. Since punch cards are an especially convenient form of initially recording observed data, this condition generally results in data for

one segment being entered on multiple lines or punch cards. These lines or cards can then be concatenated within the computer. In order to be able to refer to the original lines or cards, the concept of a *subsegment* is introduced. A subsegment is a character string (usually generated from a typewriter line or punch card) which contains values of a subset of attributes for a specific segment type. For each segment type, subsegments are of equal length and are implicitly numbered 1, 2, ..., N.

Data structures are not limited to being rectangular or hierarchical; there are a wide variety of more complex structures in use at the present time. The MASH codebook standard is designed to describe most efficiently rectangular and hierarchical structures because MASH does not now support the implementation of models requiring more complex data structures. Minor changes in the codebook standard would allow more complex file structures to be defined. However, there would be a considerably larger burden placed upon the processing programs that accessed a more complex data file through such a codebook. The absence of these features should not be construed as a judgement that they are unimportant, but rather that resource limitations and lack of immediate application precluded their development. lt is hoped that such a standard will eventually be extended to encompass more complex structures.

Codebook Structure and Content

This section describes the structure, sequence, and content of the information contained within a MASH machine readable codebook.

Codebook information is organized in a sequence of *lines* having a specific order. Each line consists of a string of alphanumeric characters of up to 70 characters in length. The first character in the string is an identifier that specifies the *line type* of this line.

The term *line* as used here and the term *punch card* are synonymous. Users and computer centers oriented to punch card files may equate the two terms. Since the maximum length of each codebook line is 70 characters, columns 71-80 of a machine readable codebook card deck are not used for any codebook function and may be recorded in any manner, possibly with identification and sequencing information that a user wishes to add.

Each line contained in a machine readable codebook file has a *line type,* defined by its first character. Each line type has its own format for the information it contains, and these formats are all fixed field. The codebook standard defines the following types of lines: (1) *A*ttribute definition lines; (2) *E*nd of data file definition lines; (3) a *F*ile identification line; (4) attribute *N*ote lines; (5) attribute *Q*uestion lines; (6) *S*egment definition lines; (7) file and segment *T*itle lines; (8) attribute *V*alue definition lines; and (9) and end of codebook file (*Z*) line.

Each codebook file begins with its file line (F) and ends with its terminal line (Z). The file line may (optionally) be followed by a variable number of title lines describing the file. Groups of lines, each describing fully one segment type within the file, follow in order of increasing segment identification number. After the last such group of lines describing the segment type with the highest identification number, the end of file definition lines (if any) follow. The Z or terminal line terminates the codebook file.

Each group of lines describing the information in a segment is ordered in the following manner. The first line in the group is a segment line; it is followed (optionally) by a variable number of title lines describing the segment type. The remainder of the lines in the group are organized by attribute. All information pertaining to a specific attribute is grouped together, and the lines containing that information appear in the following order: (1) question lines, if any; (2) note lines (if any) to be printed before the attribute name and values; (3) the attribute line; (4) the value lines for the attribute (at least one); and (5) note lines to be printed after the attribute and value lines. The following paragraphs describe the contents of each codebook line type.

The File line identifies the data file which is described by the codebook file and to which it corresponds. It contains a mnemonic name for referencing the data file, a title describing it, and the number of distinct segment types contained within it.

The Segment line defines the characteristics of one segment type within the data file. The line includes the segment name, number, label, and number of subsegments contained by it and their length.

The segment line also contains information that allows a program to ascertain the structure of the record sequence. Segments may be identified by flag values within the occurrences of a segment type; such a recognition algorithm may be specified in a segment line by specifying an attribute within the segment in a simple boolean condition. The type of segment is then considered to be identified if the boolean condition is satisfied. Alternatively, the segment may be defined as the node of a part of a tree structure; an attribute within the segment is then specified that contains a count of the number of occurrences of the segment at the next lower level that precede the next occurrence of the segment at the same level.

Title lines contain free form text and are included to encourage accurate and thorough documentation of data files and segment types. Although title lines are optional, they have been used for such documentation as data file abstracts, date, location and circumstances of the file's creation, the method of data collection, and the responsible individual or organization for the creation and maintenance of the data file and its codebook.

The Attribute line is used to define all characteristics of an attribute except the range of values which it may take on. The line contains a mnemonic name and label, mode, a positional field within a segment and subsegment, and a mnemonic reference to the source of the attribute. Fields are also provided for specifying the attribute's position within a compressed binary form of the file; these fields are used by the MASH system since data files corresponding to micro populations are converted to and maintained internally in such a compressed form.

Value lines are associated with each attribute to define the list or range of possible legal values for each attribute. Attributes may assume both a series of discrete values and a continuous range of values. These values are specified using as many value lines as are necessary.

Values are specified by a relational operator and by one or two values specified in the line. Legal values may be defined by: (1) equating the attribute value to a given value; (2) specifying a consecutive sequence of legal integer values by including the end values of the sequence; (3) specifying the end values of a continuous range of legal values; (4) specifying a character string equality condition for character attributes; or (5) allowing all possible contents of the attribute field to be legal.

Each value definition is associated with a label that may be used to attach substantive meaning to the value or values. Also, each value specification based upon equality of values may be tagged with a "special condition" indicator. This indicator makes it possible to label explicitly values that are used to flag a condition, such as "missing data," "not applicable," "don't know whether amount exists," and "amount exists, but don't know how much," rather than nominal, ordinal or cardinal values.

Question lines may be associated with any attribute and may contain free form text of the question or procedure that is used to elicit a value (response) for this attribute (question). Question lines are most appropriately used for documenting data files that have been generated by sample survey interviews. Examples of questions that might be contained in question lines are:

1. How old is ... ?
2. What is the standard industrial classification of the primary product produced by this establishment?
3. Excluding lump sum payments, what was the total dollar value of ...'s income during the last month?

Question lines are included in the codebook standard to encourage complete data file documentation. In addition such information could be used to control an interactive interviewing or data capture process.

Note lines may be associated with any attribute, and are used to contain free form text describing the attribute. Note lines are optional, and as many note lines as desired may be included in the codebook for descriptive purposes.

The End line defines the condition for recognizing the end of the data file which the codebook describes. This condition may be described by count, e.g. terminate reading the file after the 1000'th occurrence of ...; or it may be described by value, e.g. terminate reading the file when the value of attribute ... is 999999. Each end line includes one end condition, and each end condition includes one value which is either a count or is the value in an equality or identity condition. An end line may also specify that the end of the data file is defined by a logical end of file on the file's recording medium rather than by a logical condition defined on its contents.

The end of codebook (Z) line denotes the logical end of the code- book file.

The following pages contain an extract from the formatted print of a codebook describing a micropopulation used by MASH. The population data are originally extracted from the 1970 1-in-1000 Decennial Census file described in Appendix 1.

9-Dec-76
21:23

SEGMENT 3: PERSON

NO.	ATTRIBUTE	LABEL	MODE/ OPER	PRIMARY VALUE	SECOND VALUE	SUBSEG AND POSITION
14.	OTHERINC	Income from other sources	N			0 30 34
		0 or NAP-Under 14 years age	SC NAP	0		
		Income in dollars	BT	1	49950	
		$50,000 or more	EQ	50000		
		NOTES: These values have been recoded from the original Census values for increased similarity to the Surveys of Economic Opportunity.				
15.	PUBWELFR	Income from public assistance	N			0 35 39
		0 or NAP-Under 14 years age	SC NAP	0		
		Income in dollars	BT	1	49950	
		$50,000 or more	EQ	50000		
		NOTES: These values have been recoded from the original Census values for increased similarity to the Surveys of Economic Opportunity.				
16.	RACE	Race of individual	N			0 40 40
		White	EQ	1		
		Black	EQ	2		
		Other	EQ	3		
		NOTES: Value 3 - other - includes the original Census values for American Indian (2), Japanese (3), Chinese (4), Filipino (5), Hawaiian and in Alaska, Aleut (6), Korean and in Alaska, Eskimo (7), and other (8). The values in this attribute have been recoded for increased similarity to the Surveys of Economic Opportunity.				
17.	RFAMHEAD	Family relationship summary	N			0 41 41
		Family head	EQ	1		
		Wife	EQ	2		
		Child of unit head	EQ	3		
		Other relative	EQ	4		
		Unrelated individual	EQ	5		
		NOTES: The values in this attribute have been recoded for increased similarity to the Surveys of Economic Opportunity. Value 3 - child of unit head - includes the original Census attributes:				

child under 18 single (2), child under 18 ever married (3), and child 18 and over (4). Value 4 - other relative - includes other relative under 18 single (5), other relative under 18 ever married (6), and other relative 18 and over (7). Value 5 - unrelated individual - includes primary individual (8), secondary individual (9), and group quarters member (10).

- -

18. RSFAMHD — Subfamily relationship — N — 0 — 42 — 42

Head in a subfamily	EQ	1	
Wife in a subfamily	EQ	2	
Child	EQ	3	
Not in a subfamily	EQ	4	

NOTES: The values in this attribute have been recoded for increased similarity to the Surveys of Economic Opportunity. Value 4 - not in a subfamily - includes the original Census values for in family but not in subfamily (3), primary individual (6), secondary individual in household (7), non-inmate in group quarters (8), and inmate in group quarters (9).

- -

19. SEX — Sex of individual — N — 0 — 43 — 43

Male	EQ	1	
Female	EQ	2	

- -

20. SOSECUR — Income from Soc.Sec./RR retire — N — 0 — 44 — 48

0 or NAP-Under 14 years age	SC NAP	0		
Income in dollars	BT	1	49950	
$50,000 or more	EQ	50000		

NOTES: These values have been recoded from the original Census values for increased similarity to the Surveys of Economic Opportunity. This attribute did not appear on the 1960 census.

- -

21. TIMESWED — Times maried — N — 0 — 49 — 49

Once	EQ	1
More than once	EQ	2
NA (under 14 or never married)	EQ	0

- -

22. WAGES — Earnings last year, tips, etc. — N — 0 — 50 — 54

0 or NAP-Under 14 years age	SC NAP	0		
Income in dollars	BT	1	49950	
$50,000 or more	EQ	50000		

NOTES: These values have been recoded from the original Census values for increased similarity to the Surveys of Economic Opportunity.

- -

23.	WEDSTATE	Marital status	N			0	55	55
		Married, spouse present	EQ	1				
		Married, spouse absent	EQ	2				
		Widowed EQ	4					
		Divorced	EQ	5				
		Separated	EQ	6				
		Never married	EQ	8				

NOTES: Value 8 - never married - includes the original census values for never married (5) and NAP (persons under 14) (6). The values in this attribute have been recoded for increased similarity to the Surveys of Economic Opportunity.

- -

24.	WEEKSW	Weeks worked	N			0	56	56
		13 weeks or less	EQ	2				
		14-26 weeks	EQ	3				
		27-39 weeks	EQ	4				
		40-47 weeks	EQ	5				
		48-49 weeks	EQ	6				
		50-52 weeks	EQ	7				
		NAP-persons under 14	SC	NAP	1			

NOTES: Value 6 - NAP - includes persons who did not work last year. These values were recoded from the original Census values for increased similarity to the Surveys of Economic Opportunity.

- -

25.	WORKCLAS	Class of worker	N			0	57	57
		Private wage worker	EQ	1				
		Government worker	EQ	2				
		Self-emloyed, own business	EQ	4				
		Working without pay	EQ	5				
		NAP-persons under 14 years	SC	NAP	0			

NOTES: Value 2 - government worker - includes the original Census values for Federal Government employees, including armed forces (1), state government employees (2), and local government employees (3). Value 4 - self employed - includes the original Census values for self-employed, own business, not incorporated (4), and self-employed, own business, incorporated (5). These values were recoded for increased similarity to the Surveys of Economic Opportunity.

- -