APPLICATIONS IN COMPUTING FOR SOCIAL ANTHROPOLOGISTS

MICHAEL D. FISCHER

Applications in computing for social anthropologists

As increasing numbers of social anthropologists use computers for word processing, interest in other applications inevitably follows. *Applications in Computing for Social Anthropologists* addresses this interest and encourages researchers to make full use of their computers to help them organize data. First, the author discusses computing applications in relation to research activities shared by all anthropologists—ethnographic fieldwork, management and analysis of fieldnotes and the use of visual and aural material. The author then illustrates the way in which computer-based representations can satisfy the requirements of anthropological methods with a detailed examination of ways of representing kinship relations in an original way. New developments in the representation of visual and aural data on computers as well as possible applications of knowledge-based models are also introduced. In a clear and sympathetic style, Michael Fischer provides an excellent resource which contextualizes computing applications in a form suitable for tackling the problems currently faced by social anthropologists.

Michael D.Fischer formerly worked in the computer industry and is now Director of the Centre for Social Anthropology and Computing at the University of Kent at Canterbury.

Association of Social Anthropologists AS A Research Methods in Social Anthropology

Panel of Honorary Editors:

A.L.Epstein James J.Fox Clifford Geertz
Adam Kuper Marilyn Strathern

Series Editor:

Anthony Good

University of Edinburgh

Research practices in the study of kinship
Alan Barnard and Anthony Good (Academic Press) 1984

Ethnographic research
A guide to general conduct
R.F.Ellen (ed.) (Academic Press) 1984

Observing the economy
C.A.Gregory and J.C.Altman (Routledge) 1989

Oral traditions and the verbal arts
A guide to research practices
Ruth Finnegan (Routledge) 1992

Applications in computing for social anthropologists

Michael D.Fischer

London and New York

First published 1994
by Routledge
11 New Fetter Lane, London EC4P 4EE

This edition published in the Taylor & Francis e-Library, 2005.

Simultaneously published in the USA and Canada
by Routledge
29 West 35th Street, New York, NY 10001

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data
Fischer, Michael D.
Applications in computing for social anthropologists/Michael D.Fischer.
p. cm.—(ASA research methods in social anthropology)
Includes bibliographical references.
1. Ethnology-Data processing. 2. Ethnology-Mathematical models.
I. Title. II. Series: ASA research methods in social anthropology (Routledge (Firm))
GN346.5.F57 1993
306′.0285–dc20 93–10306
CIP

ISBN 0-203-45108-2 Master e-book ISBN

ISBN 0-203-45645-9 (Adobe eReader Format)
ISBN 0-415-01818-8 (hbk)
ISBN 0-415-01819-6 (pbk)

Contents

Illustrations

FIGURES

TABLES

Foreword

Roy Ellen

That somebody with no more than an average acquaintance with computing applications in anthropology should be introducing this volume may require some explanation. Part of the answer lies with the fact that I was responsible, as founding editor of the ASA Research Methods Series, for commissioning Michael Fischer to write what is now before you. The other part lies in precisely what might seem on the surface to disqualify me: that I am no expert. For many anthropologists, computing—apart from mechanical word-processing operations—remains a fairly closed book, and something which many are still not convinced offers anything that would justify the time they might expect to invest in finding out. Despite being a tolerably competent 'end user', and being fairly open-minded about how computers might improve my professional practice, I share the general lack of confidence, technical gaucheness and jokey scepticism of some of my colleagues. I am, in short, exactly the kind of person at whom the volume is directed; and for me it works.

Michael Fischer's credentials for writing this book will be clear to all who know him; probably no one else in the British Isles at the time of writing has the necessary qualifications, or could write with such freshness, authority and wealth of practical know-how. Fischer was appointed in 1985 to fill a post at the University of Kent at Canterbury created with the establishment by John Davis of what is now the Centre for Social Anthropology and Computing (CSAC). He is presently Director of the centre, a role to which he brings the experience of teaching anthropological computing to some seven cohorts of undergraduates and graduate students, a stalwart track-record in encouraging and converting sometimes reluctant or apparently untutorable colleagues, an ability to attract major research awards with a central (and not merely supportive) computing focus, an enviable record of dissemination through both hard and electronic media, and a rare technical understanding of both hardware and programming skills. Most importantly, though, Michael Fischer is not just some user-friendly computer scientist but someone who lives for anthropology, and who has mainstream ethnographic interests. It is because of this that what he writes is so accessible, informed and practical. He has encountered the problems we all encounter in writing and managing fieldnotes, in grappling with the intricacies of kinship, in appreciating the ultimately multifaceted, reflexive and problematic quality of our data. He is also well aware of the methodological divisions within the subject which have a direct bearing on what he advocates. What is perhaps unusually significant about this book, is not that it is a timely, readable and

professional introduction to its subject (it is all of these things), but that its author actually sheds new light on our research practices, on ‘doing anthropology’, by virtue of the sharp focus that an intelligent use of computing demands that we bring. At the same time the book manages to make a contribution to the ethnography of computer use. Thus, this is a volume which not only demystifies and opens-up a new field in anthropological research practices to the growing constituency of those looking for a lead, but also serves to scrutinize our existing, more conventional, practice.

Acknowledgements

In any book on anthropological methodology it is impossible to acknowledge those who have contributed so much, because their knowledge is more often imparted through experience than formal writing and teaching. Much of this book is a compilation of such experiences, and I apologize for being unable to remember all my co-authors in these experiences, especially my students, past and present, who have honed much of this material.

I thank the Economic and Social Research Council (under grants R000231113, R00231953, R000233509 and R000233933), the Tri-Council HCI/ Cog.Sci. Initiative (under grant SPG8920734), and the Leverhulme Foundation for supporting many of these experiences, without which this book would have appeared much sooner, if in a more impoverished form.

The Government of Pakistan has provided generous support for my various projects in their country over the past decade. I especially wish to thank Tayyab Yazdani Malik, Tahir Yazdani Malik, Mazhar Ali Khan and Younis Khan for their friendship, hospitality and support for my research, and for acting as facilitators on many occasions.

The Government of the Cook Islands has been a gracious host to my research, and I especially wish to thank Kauraka Kauraka for his assistance.

Although my experience with computers dates to 1965, due to a forgotten benefactor at Collins Radio Inc., my real education is due to Michael Schucking and Charles Lohf, who believed, as I did, that a computer business could be built on $5 and dreams.

Ira Buchler kindly provided much of the intellectual basis of my research, Dick Schaedel its focus and Henry Selby whatever strength it has. Henry also advised me to disguise any computing ability until safely tenured, which would have served well had John Davis not radically altered the landscape, with a vision of computing in ‘ordinary’ anthropology.

I cannot thank John Davis enough for his confidence and support in writing this book, though I fear it is not what either of us envisioned. John Demoss, Gucharan Khanna, Dwight Read and Douglas White have all provided many ideas and inspirations. I also thank my colleagues at the University of Kent for countless hours of critical discussion and for their continuing support.

Finally, this book would never been have begun or completed without the effort and support of my colleague and wife, Wenonah Lyon, whose contribution is immeasurable.

As always, I am responsible for all shortcomings of this work.

Chapter 1
Perspectives and resources

> One can not move from the informal to the formal by formal means alone.
>
> (Fortune program, UNIX V7)

1.1
COMPUTING IN SOCIAL ANTHROPOLOGY

Social anthropology has been very slow to take up computer-based methods, although there has been an increase in recent years. There are many possible reasons for this, but there are some we can dismiss out of hand. Some anthropologists attribute this attitude to the posture of anthropology between the humanities and the biological sciences. However, our closest relatives in both camps have been active users of computers for nearly forty years. There has been an active list of journals for computing in the humanities since the mid-1950s, pre-dating active dialogue by mainstream biologists. It cannot be due to complacency with our methods, as these have been in crisis since the early 1970s.

Assuming that anthropologists as a group are not defective in some important way, it would appear a reasonable hypothesis is that, by and large, computer-based methods have not been able to address major issues that most concern anthropologists (Sugita 1987: 9–14). It is the premise of this book that the development of new technology for representation with inexpensive and powerful computers has greatly increased the potential for computing for 'ordinary' anthropology. This is confirmed to some degree by an increase in conference papers on research which includes computer use, a scattering of research reports and notes in the journals and a recent, edited volume on computer use in anthropology (Boone and Wood 1992).

But there is another issue which has inhibited use of computers in social anthropology; the lack of computing methods which are distinctly 'anthropological'. Kippen, responding to a research report on the use of computers in the field, remarks:

> while Dyson and Dyson-Hudson…rovided excellent technical information regarding portable systems that can be operated under difficult fieldwork

> conditions…would the essence of their report have been any different had it been entitled 'Computers for Botanical Fieldwork'? Almost certainly not.
>
> (1988b: 317)

When I first read this I was somewhat disturbed, not only because I had published several pieces sharing this 'fault', but because it struck me that Kippen was basically right in many ways. However, I do not think this is an indictment. What makes anthropology 'anthropological' is the subject matter; methods are inherently discipline-free, as the propensity of anthropologists to borrow should confirm. This was aptly illustrated by Hymes when he presented a quotation from a geologist discussing the impact of computing on geology, but deleting references to geology to show the analogy to the situation of anthropology (Hymes 1965a: 27n).

I have attempted three basic organizing principles in this book. First, I have addressed aspects of applied computing which I have found capable of addressing anthropological problems. I have avoided saying much about numerical applications, but only because there is no shortage of information on these. I have not discussed specific computer programs, rather discussing generic program types and how these might apply to anthropological data, since specific 'brand name' programs change very rapidly, both in capability and even in their existence.

Second, I have tried to address computing issues in terms of anthropological issues. The intent is not to contradict or contravene computer science, but simply to view these principles from a vantage point more favourable to anthropologists. In this respect I over-simplify some issues and elaborate others.

Third, I have tried to balance the book to address the needs of anthropologists with little prior knowledge of computing, as well as those anthropologists considering increasing the use of computers in their work. This is not a completely successful marriage, but I hope it is adequate.

This book is a guide to computer-assisted research methods in social anthropology, not a tutorial on computing. I assume that you have used a computer at least for word processing, and are familiar with the most basic operations of computing. The book is not self-contained; to pursue any specific project you will require further specific information on precise computer programs, hardware requirements and interpretations, from colleagues, local support people or, as is common, graduate students.

In this chapter I briefly introduce some general issues of applying computers to social anthropology, ending the chapter with a set of recommendations for levels of use, corresponding to a software and hardware tool kit.

Mainstream computing has been developed around four paradigms; arithmetic and numerical processing, data organization and processing, writing and document preparation (sometimes in the guise of 'word processing') and symbolic computing, the latter including an area referred to as artificial intelligence.

The dominant use of computers in social anthropology until 1980 was numerical processing, most often embodied in SPSS *(Statistical Package for*

the Social Sciences). Numerical methods apply to social anthropology in important ways; there are things of significance that we can count—demographic data, crop yields, hours of activity. However, many anthropologists have apparently decided they could do their analysis without investing time and resources into computing. Computers have the potential to make numerical methods more acceptable to anthropologists who currently avoid them and vastly to improve the research of those who do not. I shall only make occasional reference to numerical methods in this book because this area is relatively well covered in the anthropological literature. For recent contributions on quantitative computing in anthropology see Boone and Wood (1992) and the journal *Quantitative Anthropology*. For more information on numerical methods in general see Johnson (1978), De Meur (1986), Mitchell (1980) and White (1973), and additional references in the Bibliography.

Data management has focused mainly on maintaining collections of data which fit into fairly rigid categories. This is a growing area of application in anthropology, building on a growing number of applications which avoid the requirement for rigid structuring of information. Data management of both sorts probably has productive scope for most anthropologists in the future, since everyone has data to keep track of. Even fieldnotes fit easily into the more flexible formats. Traditional data management is discussed in Section 2.3. Fieldnote management is the subject of Chapter 4.

Writing is the main area of penetration in social anthropology. This has already had a major impact on the discipline, since more and more publishers are requiring high-quality camera-ready quality. Related to writing there are a number of additional resources, such as bibliographic databases, spell-check programs thesaurus programs, and outlining applications. There is a lot of support for this area and discussion is limited to a few special features in Section 1.3.1.1.

Symbolic computing is now widely accepted by computer scientists, but was viewed with suspicion during much of its long development. In the late 1970s this image began to change, in part because of the advent of the 'expert' system, a program which simulates the advice of a human expert within a narrow subject domain. The methods used to do this were considered rather interesting at the time to computer scientists, but were rather mundane to someone trained in anthropology and linguistics, being nothing more than a simple grammar of a knowledge domain. Of course, just at the time when non-anthropologists were finding a use for this approach to knowledge representation, anthropologists were beginning to abandon it.

Symbolic computing and knowledge representation are central to the core interests of social anthropology. This technology has taken some time to filter out of computing science in ways which are accessible to anthropologists, but opens up many possibilities for addressing research problems in anthropology with computers. These issues are discussed in Section 1.2.2 and in Chapter 8.

More flexible and improved implementations of all four paradigms, especially the inclusion of a full range of representational forms—from text to video—and representational styles—from informal to formal—vastly improves the value of computers to anthropologists, as well as reducing the investment in time required to use these new methods. Many of these improvements are very recent

in terms of general availability—arriving on the mass market in the mid-1980s and achieving mass acceptance in the early 1990s—and will continue to develop (Dow 1992).

1.2 COMPUTERS AND QUALITATIVE ANTHROPOLOGICAL DATA

Computing techniques are becoming more widespread in social anthropology and other disciplines with traditionally strong qualitative components, though not as quickly as some had thought (Hymes 1965b; Freedman 1978). The effectiveness of the new technology is currently limited by the problem of adequately representing data on a computer (Agar 1983; Fischer 1986; Kubo 1987; Sugita 1987; Kippen 1989). Current representation techniques for anthropological data on computers are often small advances over what was once done (and sometimes better done) with index cards. This inability to represent information limits the methodological possibilities. Symbolic representation is aimed at developing representation techniques which provide the researcher using a computer with greater flexibility in methodology.

This section discusses some of the problems encountered in representing and analysing qualitative anthropological data using a computer. Three basic themes are discussed: first, a discussion of the social construction of computing; second, the form of representations in general; and third, a discussion of the distinction between weak and strong representations, the latter incorporating knowledge about organization within the representation. The point of the third theme is that this distinction is not the same as an informal to formal transition, and that strong representations need not be explicit or have explicit, deterministic outcomes.

1.2.1 Symbolic representation

By representation I do not mean presentation in the form of diagrams, literal text or tables (although I do not exclude these), but a kind of modelling whereby a combination of elementary (perhaps atomic) constituents and an organizational structure are used to describe and model some object, structure or phenomenon. This is not a reductionist approach. Although partly by definition, any complex phenomenon has properties that cannot be predicted from the elementary constituents alone and their local organization. A model must capture these properties to be at least descriptively adequate. A ‘higher-order’ model (relative to the constituents) is required to access and interpret this model—there is information in the model that is not derivable from the constituents.

The representation of ethnographic and other cultural materials has consistently been an issue in social anthropology, particularly during the past two decades (Holy and Stuchlik 1985; Sperber 1985). Many problems have emerged and representation of these materials on a computer has been no less problematic. An important aspect of any representation rests on its purpose. It is difficult to represent anything in ‘universal’ format since the range of usage is

difficult to presuppose, and any representation 'fixes' the range of possible operations that can be performed on the representation. Most representation involves models or 'approaches', the latter being metamodels. Many researchers are contemptuous of 'approaches', but a brief look at one particular artefact of our own culture, computer programming languages, clearly demonstrates the value of an 'approach' for describing a problem (Fischer 1987:3).

Ultimately all modern computers manipulate signs within a very simple framework. The 'machine' aspect of a computer generally consists of a means of representing a large number of signs, a means of storing signs and sequences of signs and a unit that can compare, modify and arrange signs. At this simple level there is little structure to the signs, other than that imposed on them—they are simply markers, with no inherent signification. There is no such 'basic' computer generally available, only elaborations and variations on the above theme. Each artefactual computer is one engineer's (or more often a group of engineers) conception of what a computer should be. Thus even at the most basic available level different 'world views' emerge. Ultimately, any problem or model that can be represented on one given computer can be represented on another (assuming similar memory, etc.), but we find that there are advocates of one system over another, despite the behavioural identity of the systems.

For a variety of reasons most people find the basic level of the computer difficult to interact with and use a 'higher level' approach. These approaches generally embody an alternative metaphor or approach to representing problems, especially problems of a particular sort. These higher-level languages have their advocates for the different world views, although any task that is representable in one language is representable in another.

However, the different approaches are not without foundation. Although there exists the ability to represent any computable problem on any computer, there are varying degrees of difficulty in doing so. It is this point that is interesting, that a modelling/representational approach alters the complexity of a given representation, sometimes to the point of making the representation possible, rather than theoretically possible. Each approach is a metamodel, a perspective for selecting one model over another for a given representational task, and we find real behavioural differences in the ability of people to represent different situations using different approaches (Zeigler 1979).

There is much the same problem in any discipline. An approach is required to define 'what' and 'how' we are to represent our subject matter. This section, a perspective on representing anthropological data on computers, could easily end with a short message: there is no different problem of representation on a computer than exists for representations in general, the problem is the representation itself. However, I shall not be so merciful, and a computer is a good medium within which to discuss this problem.

There are many levels of representation possible in the computer medium, ranging from passive recording of the information as it would be on paper (although the actual means is complex (human language), the complexity lies outside the medium) to active representations with transformational properties (where the complexity of the organization is represented within the medium). It is the latter category that creates the most difficulty, for unlike the former, it appears to require a conscious apprehension of the organization for the

representation—it requires a model. This continuum of representation can be characterized by a corresponding increase in the amount of structural representation (or knowledge) in the representation; a continuum from weak to strong representation.

1.2.2 Knowledge representation in anthropology

Knowledge is the anthropologist's stock-in-trade. Malcolm Crick (1982) argues against the formation of an anthropology of knowledge on the grounds that it would be impossible to isolate such an integral aspect of anthropology to one sub-discipline. It could be said that anthropology is the description and representation of knowledge possessed by human groups. Unfortunately, such knowledge in one's own society appears to be very difficult to describe or represent, and it is even more difficult to describe the knowledge of other societies as the internal context that allows interpretation of partial systems of representation such as language or ritual is lacking, except, perhaps, to some small degree within the ethnographer. The addition of computer-based methods to the anthropologist's repertoire suggests a new requirement; methods of representation of knowledge in a new medium.

The data or information that anthropologists attempt to relate is quite complex, because of the number of factors involved and the dual nature of the knowledge—if it can be called knowledge at all. However, anthropologists have developed usual and characteristic methods of representing the knowledge gained in the field. In the past written accounts have been the primary method of representation, with some use of formal analytic presentations, although few of the latter are considered to have broad, general representational value or general acceptance among anthropologists. The primary purpose of representation within a computing environment is to use the computer as a tool for extracting and examining relationships within and between domains of knowledge; in a sense facilitating the creation of knowledge about knowledge. One of the barriers anthropologists face in representing problems using a computer is the lack of suitable tools for representation.

I.Rossi (1982) observes that anthropology is distinct from other biological and physical sciences in two ways. First, anthropologists can directly and intimately interact with their subject matter, and second, while both the physical sciences and anthropology use models based on symbolic systems, symbols in the physical sciences are related to things whilst those in anthropology are related to other symbols.

Although it is not possible at this time to define exactly what knowledge is, the latter observation is perhaps the basis for a working definition of knowledge, or more specifically, what a representation of knowledge is. One way to look at knowledge is to say it is the result of an active process of connecting systems of symbols together; knowledge is not a static collection of facts, but the situationally specific outcome of relating facts and other knowledge. This is, of course, a gross simplification of whatever knowledge is, but it is sufficient for an initial examination of methods of representation, as a representation does not

necessarily claim to be a model of what knowledge is but only a model of what knowledge does in a particular context (Ellen 1986).

Barr and Feigenbaum (1982) present a good review of representation techniques used in artificial intelligence (AI), a sub-discipline of computer science which has developed a number of methods to simulate intelligence (by internally defined criteria). They offer this operational definition of representing knowledge:

> a representation of knowledge is a combination of data structures and interpretive procedures that…will lead to 'knowledgeable' behaviour.
>
> (vol.1: 143)

This definition is not entirely satisfactory from an anthropological perspective due to the emphasis on behaviour rather than the system of knowledge. But it captures the central idea that information plus rules (or processes) results in knowledge.

In some sense the anthropologist working in the field is attempting to acquire and analyse an alien representation of knowledge. The goals of AI and anthropology are not identical—most anthropologists are not interested in writing programs that have knowledgeable behaviour for that purpose; they are interested in representing knowledge. However, AI researchers do represent the knowledge of lawyers and doctors; are these techniques useful to represent the knowledge of the Nualu, Zuwaya or Panjabi?

The computer as a representational medium offers some exciting possibilities. The purpose of a representation is twofold. It acts as a means of recording and testing ones ideas about some domain and it is a means of communicating ones ideas to others who might have an interest in the domain. Traditional representations in anthropology rely on substantial amounts of text and a high level of training and expertise to interpret it. A computer representation has the potential to represent some of the interpretation as well, relying less on the combined judgment of the researchers and their audience: the computer representation of the knowledge structure is more explicit, less abstract.[1] The representation can be tested and probed. This allows discussion and criticism to focus directly on the interpretative aspect of the representation, rather than on interpretations of the interpretation as so often happens.

1.2.3 Computers and representation

Most people in industrialized societies have incorporated a view of computers as 'number crunching' machines—overgrown adding machines of a sort. This view has been modified over the past decade with the development of microcomputers, with their games, text processing and database capabilities. Often this has led to another extreme view, with computers seen as overgrown typewriters. The reasons for this are largely historical; computers and the funding for computers came first from defence and second from business, and they were developed by mathematicians and then engineers. This has left its mark.

The computer is not, and cannot be, a finished tool. Although the ideas grew up from the mathematics of Babbage, Lovelace and Turing, the implementation came through the ranks of engineering, physical sciences and, later, business. Quite literally the computer most people deal with is an artefact of some engineers' symbolic conception of a modelling language environment. To some extent this can be ignored as it has been demonstrated that all symbolic systems that have been represented on a computer are equivalent in power of representation. However, this is much like saying all cultures are equivalent in expressive power. As the anthropologist knows from hard experience, the acquisition of another culture sufficient to express their own cultural notions is a non-trivial task. Generally, to use a computer as a research tool you must be prepared to translate the problem as conceived into another representational system; a system that is an artefact of a different problem environment.

Computers are essentially 'symbol-crunching' machines (Fischer 1987), whose abilities were first enlisted to crunch a particular type of symbol, a number. Although present-day computers are optimized to perform very simple numerical operations, such as addition and multiplication, to perform any slightly complex numerical operation requires a model of that activity to be implemented. An enormous amount of effort has been put into proving that algorithms do indeed perform the anticipated result. Most of this work has only been successful when the algorithm is severely curtailed and formalized. Algorithms are simply not provable in general; it is not possible to be certain that the algorithm is an accurate depiction of the model on formal grounds alone.

The symbol processing abilities of the computer for ordinary symbol processing are quite primitive—difference, equality and, if one allows, logical operations of *and, or, exclusive or* and *not*. More complex operations require a very good model of how the operation can be built up or approximated from the primitive operations. Part of the work can be accomplished using so-called high-level programming languages. These languages are an intermediate model of building blocks for creating relations and defining symbols.

The first 'programming languages' consisted of rewiring the machine for each new task. In the 1950s J.Von Neuman defined an operational method of having the computer store a program as a set of instructions in the computer. Soon the notion of 'auto-coding', or new sets of instructions whose actions were the translation of these new instructions into other sets of instructions, resulted in so-called 'high-level' programming. The idea of high-level programming was that a problem-oriented set of instructions could be constructed from the more primitive instructions provided by the engineer.

Predictably, the first high-level languages were directed toward arithmetic formula translation (hence FORTRAN). In the number-crunching realm FORTRAN was, and still is, a favourite language, because it simplifies many of the arithmetic and elementary algebraic operations and because it incorporates a means of re-using methods worked out for similar problems, called libraries. It also formalized a notion of data types. A data type is simply a model of a particular type of information, with corresponding operations on that information. FORTRAN includes integer, floating point and complex number data types, established so that the programmer can think and represent numbers in a familiar symbolic representation, rather than translating and manipulating these data

types using the lowest computational representation. That they are models is evident, as ultimately they are represented in the fundamental symbolic representations of the computer.

Other early languages in another 'tradition' are SNOBOL and LISP. SNOBOL was devised essentially for symbol processing, and had appropriate data types. Besides numeric data types, SNOBOL had other data types, all of which were represented by sequences of alphabetic characters, called *strings*. The primitive type in SNOBOL is just this, some arbitrary set of characters. Derivative types are patterns, which are sets of strings to match, records, which are attributional sets, and tables, which are associative sets. The basic operation of SNOBOL is to match strings or patterns, and it includes the ability to delete, replace or add to specific sequences within a string.

LISP is popular among those who do symbolic processing, computational linguists and AI researchers among others. LISP has two basic data types, atoms and lists. An atom is some indivisible sequence of characters and a list is one or more atoms (including the NIL atom) or lists, related by being in the list. Its enthusiasts usually declare its extreme flexibility in representing very complex data types, such as semantic networks and genealogical structures.

A more recent symbolic language, Prolog, made accessible another concept, declarative programming, at a very high level. Declarative programming contrasts with procedural programming. In procedural programming the programmer defines a problem as a series of steps, a process, which will have as its result a 'solution' to that problem. The resulting program is itself declarative, in that the procedural details are 'hidden' at that level. You declare your intent by selection of the program, and the program will produce a result. Prolog, based on predicate logic, is an attempt to define a language in which relationships between data structures are declared and problems are posed as descriptions of goals, with minimal reference to the processes required to achieve the goals (e.g. solve the problem). In Chapter 7 we shall see how Prolog can 'solve' problems of calculating genealogical relationships.

Each of these languages represents an 'approach' to solving particular types of problems, and leads us to different structural solutions to the same problem. FORTRAN is very good if your programming needs largely depend on accurate arithmetic. SNOBOL (and similar languages) are very good if you need to do complex textual transformations, and LISP and Prolog are superb for the representation and manipulation of relationships and structures. However, all current programming languages are in theory equivalent—each can address the same problems and arrive at the same results. In practice the importance of different specialized programming approaches is valued by programmers. Although the occasional programmer will do list processing or textual transformation using FORTRAN, just to show it can be done, most would decline this task and choose a different approach or a different language.

1.2.4
Some issues in computing applications for anthropology

Most computer-based models in social anthropology have been peripheral models —i.e. they have been primarily models of the ecology (Buchler *et al.* 1986), the local demography (Dyke and MacCluer 1973) or the economic context (Fischer 1980; Eades 1988) within which the effects of a social practice are evaluated. Despite the domination of cognitive, semantic and symbolic issues in current ethnography (Crick 1982), few attempts have been made to apply computer models to knowledge representation, although there is a small but growing literature (Miranda 1967; Coxon and Chalmers 1973; Guillet 1989a, 1989b; Benfer *et al.* 1991; Fischer and Finkelstein 1991; Read and Behrens 1992).

The essential problem in applying such models is the inaccessibility or non-existence of methods of representing qualitative and symbolic data so that data can be explored with either formal rigour or even a reasonable degree of control. The particular historical development of computing resources has led to the development of facilities oriented to either quantitative representations, organized lists or simple representations of texts. These are not adequate for the human sciences (Sugita 1987: 14).

Unfortunately computer tools to analyse complex situations are generally hand-crafted for each application. Anthropological data is usually collected piecemeal, and at best only semi-systematically. Although the anthropologist usually has a specific purpose underlying portions of the data collection, by the nature of the anthropological data collection process many other materials are gathered as well, many of which will prove more valuable in the long run than the specific purpose materials.

These complications in the data collection process are an artefact of the goals of anthropological research; not only to investigate specific processes and attributes but also to provide an overall realistic, usable description of the group under study that can be used for broader analysis within that same group or comparison with other groups.

Anthropologists have used computers as tools for analysis for over thirty years. Interest has been higher than the actual amount of work produced, but there has been a substantial amount of computer-related work to date. Perhaps the most serious barrier to anthropologists wanting to incorporate computer-based methods into their work is the huge amount of data preparation necessary prior to analysis using conventional application programs (Davis 1984b: 308–9). Although some of this preparation is similar to that in other disciplines, it is intensified by the methodology of anthropological data collection.

This is not to say that anthropologists are unique in having this problem but that it is endemic among anthropologists. Little ‘straightforward’ data is suitable for the ‘row and column’ style entry that most application software expects. It is true that an appropriate database program can aid in sorting out some problems (Section 2.3), but to perform many analyses requires acrobatic feats of reformatting data to represent static views of the overall structure or process, requiring enormous management effort to achieve the desired analysis.

There can be no solution to the basic problem of having large amounts of varied information. However much of the management problem can be

simplified if there are problem-solving environments designed for this information. An obvious strategy for computing applications in anthropology is to adapt an existing method of representation directly to a computer. This is effective when the dynamic properties of a computer medium can be exploited to ease or speed up the existing method. One such method in anthropology is the representation of genealogies. In the usual form, genealogical diagrams have proven to be useful in illustrating and elucidating points of social organization, terminology and wider relations of kinship in the society. However, in this form they are hideously difficult to construct for any sizable population (over one hundred or so) (Davis 1984b: 298), and once constructed are difficult to recast for purposes of demonstrating alternative views of the same set of information. However, if our representation of genealogical material is such that we can selectively examine either specific relationships or specific relationship types which correspond to our own specifications, then we can investigate genealogical aspects of populations numbering thousands (see Chapters 6 and 7). Moreover, if we can incorporate our findings back into the representation, we have a 'stronger' representation; a representation which contains not only the original data but which also incorporates our own knowledge about that data.

It is inevitable that when we work with data we will begin with 'weak' representations, collections of data which have no interpretation without knowledge external to the data. When we analyse this data we create successively 'stronger' representations as we impose a more interpretive context. For example, as we classify data (on whatever basis) we are imposing (or limiting) the range of possible interpretations. When the analysis reaches a stage where a paper or monograph can be produced, the processes of selection impose yet more constraints.

To apply computers to the process of analysing social data, we must evaluate how the computer-based methods will interact with this process of developing stronger and stronger representations. There are potential benefits and potential dangers.

A benefit is that while we can create very strong representations indeed, stronger than any we can create with pencil and paper because the representation itself can independently generate behaviours, these strong representations can be directly imposed on the weakest forms of representation available, without losing this weak level. That is, the strong representation can be built in layers over the original data, where the original data is still available for inspection at any point in the analysis. This level of flexibility is possible with manual methods but very difficult in practice; all too often we impose classificatory and other restrictions of interpretation on the original data, and replace the original data with these.

A danger is that computer programs incorporate the methods which yield these stronger representations, which may not be good representations. The more complex the method, the more danger that something is being added that we do not intend. This becomes worse when a program is applied which is not well understood by the analyst. As Miranda remarks:

> Unfortunately, much too much sociological garbage is processed in the hope that the magic of electronics will change it into scientific truth. And a

> sad side of the fad is that the mention of a computer number or that of the esoteric name of a program, modestly relegated to a footnote, lends indeed unwarranted authority to conclusions that even a nineteenth-century positivist would have swept aside as fallacious.
>
> (1967:77)

Fortunately (I suppose) most current computer applications are quite weak in their approach to representation as tools—what they do is quite comprehensible. This is especially true of interactive programs where you build up the representation in small steps. However, because of the more dynamic nature of computer-based documents, it is possible for you to build stronger representations using these 'weak' tools and still have the confidence that it is your representation and not an artefact of the computer. This, for example, is the basis of hypertext/hypermedia documents (Section 2.3.4; Section 4.6.2 onwards), which are built using programs which supply you with a set of tools for interlinking data in complex ways, but where you must build each link.

Finally, with simulation (Section 8.1), expert systems and production systems (Section 8.2) we have opportunities to build very strong representations in which we represent enough information for the representation to 'animate' and generate behaviours, which can either be checked against actual data or can be used as a source of 'artificial' data (Dyke 1981: 203–4) to develop and test methods against. These kinds of computer methods are, as yet, experimental and only partially proven as useful tools, but certainly represent areas for research. The discussion can be summarised as follows:

1 The computer is simply the actualization of a method of modelling or representation. There is nothing that can be done with a computer that could not have been done without one, but many such tasks would require an enormous amount of effort to do otherwise. Thus, in effect the computer represents the only way to accomplish some representational tasks. In addition there is now effectively no existing alternative form of representation that has properties that cannot be effectively duplicated using a computer (other than the organic brain), usually with at least some benefits from enhanced ability to access the information in the representation.
2 Computers are a medium for representing symbols and relations between symbols. Perhaps more important from our perspective, computers can represent processes and active relationships between symbols; not only can static relations be represented, but relationships of change and variation can be articulated as well.
3 Because of the dynamic properties of the computer medium, new possibilities for representation emerge. Although we can make use of these properties to translate our 'usual' means of representation, the development of new methodologies is necessary to take full advantage of the representation.
4 The interpretation of traditional methods of representation lies entirely within the observer of the representation. I call this form of representation a weak representation. Within the computer medium it is possible to represent

some of these interpretations as well. In other words, we can assign a set of interpretations to the representation and examine their effect on the representation. I call this a strong representation.

5 The computer medium permits a higher degree of complexity to be represented within the representation than with traditional forms. Indeed it may provide us with the first substantial tool for the study of complexity.

6 The weak/strong classification of representation appears to imply an informal/ formal dimension as well, e.g., to represent more of the interpretation of the representation appears to suggest that more is formally understood about the representation. However, as the examples of simulations, expert systems and production systems suggest, it is possible to have a representation of an overall system, without an overall formal account of that system.

1.3 CONCEPTUALIZING COMPUTER RESOURCES: TOOL KITS

The problem most people have with computers is equating them to most other machines with which they have an acquaintance. Using the 'machine' model, it might seem that all you need to learn are which buttons to press and you would 'know' how to use a computer. The problem with this approach is that computers are not machines in the usual sense of our experience. They are machines for representing machines. If you intend to use a computer for one dedicated, repetitive task, such as word processing, then it is indeed possible to learn a fairly restricted set of operations and function very effectively.

Using a computer in a research environment makes rather different demands on the anthropologist. First, the number of possible tasks in research is large, and cannot be easily anticipated in a single application so specific as to automate the process. Second, it is important for the researcher to have a good deal of control over the representations used by the computer applications employed to avoid results which are an artefact of the chosen application. Third, it is unlikely that the computer can be used for more than a portion of the research process, and the researcher must understand when to apply a method and how appropriate it is.

There is, then, scope for an enormous amount of complexity. Computers without programs provide very low-level resources for action (e.g., each possible action does very little) which have a great deal of flexibility but which require a great deal of understanding of the computer and its actual construction. Because each application must be built out of these very small parts, it takes a lot of training, skill and experience to apply them efficiently, especially in a research environment which makes new demands frequently. This is alleviated to some extent by high-level computer languages, but even at this level considerable time will be required to make full use of the computer system.

There is, thankfully, a happy medium. Although the possible range of applications which might be useful to social anthropology is very large, there are many research tasks from which most anthropologists can benefit, drawing on a

relatively small set of computer applications used together to solve a variety of related problems.

We can consider as tools a set of computer applications (programs) which can accept and create data of different sorts directly from the user or from other applications in the suite. The existence of compatible input and output standards for different kinds of data mean that applications can be used together to meet an objective. At any given stage of data collection, management, analysis or presentation the researcher can apply an appropriate computer application to the process as required with a relatively small investment in time.

A note on operating systems

An operating system is the software responsible for access to and management of the resources of the computer hardware and system software. We need not discuss operating systems in much detail since even the expert user has little direct contact at this level, but there is a relationship between operating systems and facilities available at the level of interaction with the computer.

Hardware resources include devices for storing data, such as hard-disk drives and devices for printing output from programs, as well as handling keyboards and display screens. Software resources are mainly programs which are supplied by the user, and a means for connecting programs to documents where the results of using a program are stored. These documents are usually called files and are represented as a discrete object by the operating system, usually by a user-defined name.

From our point of view the most important property of an operating system is the ease with which it is possible to use the computer for different jobs. Most of the interaction with the operating system is indirect, through the user interface. Currently this interface takes two basic forms, textual and graphical.

Text-based interfaces are derived from older terminal interfaces, which had extremely limited display capabilities, usually twenty-five or so lines of alpha-numeric characters. Resources are invoked through typing words and symbols, called commands. Programs are usually accessed by typing their name, followed by other information the program may require such as a document name or a sub-instruction (or option) the user may wish to invoke.

Graphical operating systems typically use some kind of ‘pointing’ device, such as a mouse, pen or puck, which moves a pointer on the screen to indicate a ‘hot’ area. Programs are usually accessed by placing the pointer on an image which represents the program (or a document created by the application) and pressing a button.

There is no inherent ‘extra’ power from one ‘approach’ or the other, although beginners usually find graphical operating systems easier to use, and some experts claim that the text system is better for their advanced use. However, since there are many more beginners than experts, graphical operating systems are beginning to dominate the marketplace.

1.3.1
Software tool kits

There is no definitive tool kit. The range of applications will vary between anthropologists, as do skill levels and resources which can be devoted to acquiring necessary tools. In any case, although the current functional range of tools is likely to survive for some years, there are new applications with new uses constantly being introduced. A tool kit is a set of tools suitable for your requirements. However, to set some constant context for discussion we have to deal with some specifics. As examples, I define three general software tool kits, for basic, mid-user and advanced application areas. Basic versions of most of these applications are available at no cost or low cost, and commercial versions for most range from £50–£500.

Versions of programs used in the examples in this book are readily available for most mid-to high-end computers. In any case, the tool kit components (and my classifications of them) are exemplary rather than prescriptive; these do not exclude any specialized 'plug 'n' go' programs you may find for your particular needs, such as a graphical kinship editor. Not everyone will want all of the tool kit components. For example, if you never work with statistical analysis of data you may want to give the statistics program a miss. The classification into basic, mid-user and advanced tool kits is based on the time investment required for productive use and the relative expense rather than a reflection of the ability required for use.

1.3.1.1
Basic-user tool kit

This tool kit includes programs for basic tasks which require little investment of time to use at some level, and most important are individually productive in many segments of the research process. These tools are adequate for transferring many traditional methods in anthropology to the computer, where they can then be managed and manipulated. Most of these applications are so pervasive in computing circles that very low (or no) cost is involved in acquiring all of them. Most require relatively few resources and can run on very small computers, such as notebook or palmtop computers.

Wordprocessor/text editor Probably the most common application in use by anthropologists, this is also the most popular tool for data entry, organization and writing up results. Word processors generally have facilities for text entry and formatting text and advanced wordprocessors should be able to represent different character fonts simultaneously (such as English and Arabic). They should also be able to store and display graphics and photorealistic images. Some can also store sound notes. Most computer applications you might want to use to process the information you enter will require what is sometimes called ASCII text or 'plain-text'. This simply means that these programs can only work with alphabetic data, not the formatting you have imposed, other than lines and perhaps paragraphs. Your wordprocessor should be able to make plain text documents for this reason.

Text database manager Software for managing and analysing documents which consist mainly of text, either divided into small sections, full length manuscripts or even sets of full length documents. Further information and applications are discussed in Chapter 4, Section 4.6.3.

Spreadsheet calculator Software useful for basic arithmetic, basic quantitative data management and analysis, some kinds of quantitative modelling, including simple simulations and ancillary tasks such as field accounting. See Section 2.4.1 for a basic example.

Draw/paint program Software for making diagrams and drawings on the computer, possibly for inclusion in wordprocessor documents or database programs. See Section 5.2.

Map-making software Software for making and displaying maps, and performing some kinds of analysis based on maps. See Section 5.4 and Section 3.2.7.2.

Statistics program Software for statistical analysis of data such as that obtained by questionnaire or survey. There is not much information on these programs in this book, since there are many other sources (for a recent reference see articles in Boone and Wood (1992)).

Communication software Software for transmitting data from one computer to another. These programs are used for interactive communication between a terminal or micro-computer and another terminal or computer at some remote location. There are also programs which take advantage of the growing facility of accessing data archives, especially for academic and commercial researchers. See Section 2.6.

Disk utilities for data recovery and integrity These are programs for recovering data from damaged computer disks and for finding problems with disks in advance. Any user of computers who keeps their own data on floppy disk or hard disk should have these programs, or access to them.

1.3.1.2
Mid-user tool kit

In addition to the programs in the basic tool kit, the mid-user tool kit includes programs which either require more investment in time or money or open more opportunities for using programs in tandem. These applications focus on improved results, higher levels of management and more automation. Most of these programs are either specialized versions of basic programs, such as the illustration program or desktop publishing program, or require more organizational skill, such as the relational database manager or hypertext/hypermedia program. These tools are very productive for those anthropologists willing to make the investment in time and money. For the intermediate tool kit add the following:

Database manager Software to manage, reformat and perform simple transformations on data which can be represented either as ‘rows and columns’ (flat file database) or as a set of relationships between ‘row and column’ data (relational database). See Section 2.3.

Hypertext/hypermedia manager A program which includes most of the operations of a textual database but has many more facilities for making linkages between sections of texts and includes simple to use but powerful programming facilities. Hypermedia managers can also include images and sounds. See Sections 2.3.4 and 4.6.5.

Bibliography management A specialized database for managing bibliographic references. Many have the facility to look up keywords in your text, insert a citation in a form you specify and construct a finished bibliography.

Illustration program A more advanced form of draw/paint program which has finer control over producing detailed drawings.

Text-manipulation tools Assorted programs for counting words, creating indexes, Keyword in Context (KWIC) indexes and other tasks. See Section 4.6.

Macro package A utility program which can reproduce a set of actions you perform on the computer. These are usually associated with a unused key on the keyboard, so that repetitive operations are made quicker.

Desktop publishing program A program for producing publications. These differ from wordprocessors by having much more control of layout of text, graphics and pictures on the page.

1.3.1.3
Advanced tool kit

The addition of these tools opens a wide range of new possibilities for anthropological description and analysis. They add the capability to work with aural and visual data at a high level, the ability to do advanced modelling of indigenous knowledge-based systems and the ability to write new applications through the use of a high-level programming language. Although some of these tools are not very expensive, the best versions can be and the computing platform required for using them can represent a very large financial investment, especially for the aural and visual tools. For the expert tool kit the following might be added.

Image analysis and manipulation program Software for working with photorealistic computer images. Basically a very high-end paint package with a great deal of control over colour, remapping colours, touching up images, sharpening and blurring images, resizing and preparing for printing or specific applications which can only retrieve and display these images.

Video/image-capture/presentation Software which works with hardware to capture images from printed pages, photographic slides, photographic negatives, Photo-CD discs and video tape and video camera. See Section 3.3 and Section 5.3.

Sound capture/presentation Software which works with hardware to record sound. See Section 3.3.

Expert system shell A program for writing expert systems based on rules. See Section 8.2.

Simulation package A program which supports the writing of simulations. See Section 8.1.

Programming language; e.g. Prolog (or C, Pascal or FORTRAN) Computer software for writing new programs for your own purposes. Some examples of the programming language Prolog are given in Chapters 7 and 8.

1.3.1.4
Expert tool kit

Add your own.

1.3.2
Hardware tool kit

The most significant limitation on computer use is the software. It follows that, ideally, an appropriate hardware tool kit must be closely matched to the particular mix of applications you intend to use. Unfortunately, most people acquire a computer and then attempt to find software to meet their needs as they arise. Fortunately, you can add additional components to most general purpose computers to meet new software demands. There are two basic kinds of components; those which are considered to be a part of a general purpose computer system and those which are considered to be extensions of the system. Over time, components of the latter category move to the former. For example, in the late 1970s a floppy disk drive was a desirable extension, now it is a fundamental component of all general purpose microcomputers.

Because of the dependance of software on hardware, I propose different levels of exemplary hardware tool kits which are suitable for the analogous software tool kits. Quantified components such as memory and storage capacity are reasonable values as of 1993.

1.3.2.1
Basic set

Almost any general purpose computer can support some level of the basic software tool kit. All microcomputers available in the early 1980s could support these applications, the current generation more so. However, a general stand-alone computer system expandible to the larger software tool kits should probably include the capacity to install 4–10 megabytes of main memory (RAM), with 2–4 megabytes installed to run easy-to-use operating systems, a 40–80 megabyte hard-disk drive (larger ones can be added later) and a high resolution video display.

Very small (in size and weight) computers can support these requirements easily, and can be used in the field where there is power available for charging batteries or where solar recharging is adequate. Most battery-operated notebook/laptop computers operate between two and four hours on a set of charged batteries. Using a 12 watt solar panel (weight 500–1,000 grams) it takes about 4–8 hours to charge a set of batteries. With a proper adapter (you can use an automobile cigarette lighter adapter for your machine) you can operate 20–80 hours off a charged car or truck battery, which are available in many field sites.

In the near future, there are unlikely to be major improvements in the amount of time you can operate off a general purpose portable computer's internal batteries, as the market is slowing development in this direction because of lack of demand.

For field use where no or little power is available, more specialized computers may be used. There are a number of small notebook/laptop and hand-held/ palmtop computers which can operate on 2–4 AA size batteries for several weeks. These usually have no disk storage, using solid-state storage devices instead, which can typically amount to 2–4 megabytes of long-term storage. This situation may change rapidly, since both higher capacity solid-state devices and very small low-power hard-disk drives are becoming available. The current limitation of 2–4 megabytes is more than adequate for several months' fieldnotes, amounting to approximately 3,000–6,000 pages without considering duplicate copies of the notes.

You will also need access to a suitable printer at some point. For basic printing, dot-matrix impact printers may continue to be used for a while, but these are rapidly being superseded by inexpensive ink-jet printers and laser printers. These have the advantage of being able to create camera-ready copy which includes different typefaces, fonts and graphic material.

For colour printing, intermediate quality is available from colour ink-jet printers, which currently produce results about as good as well-composed, shaded and inked drawings. Other options are available, but are expensive. This is a technology which will move quickly.

1.3.2.2
Intermediate set

The intermediate hardware set corresponds to the mid-user software tool kit. The applications in this set are mainly specialized or advanced versions of the applications in the basic tool kit. In hardware terms you can probably run all of these with the general system outlined above, but will achieve much better performance with more main memory (RAM). These are also much more demanding of the computer's processing power, and while a 'bargain' system may be adequate for the basic tool kit, something more powerful is desirable for the intermediate set. Besides the higher requirements of the mid-user software, there is also probable greater use. An operation which takes an hour too long once a week is supportable, ten times a week is less so.

1.3.2.3
Advanced set

The applications in the advanced tool kits are production tools, used for developing and processing materials, perhaps for use by the mid-user tool kit. Production tools, in general, require more computing resources than systems which need only to manipulate the materials symbolically.

In particular, to support the visual and aural data discussed in the advanced software tool kit, additional components are required. First, to work with this

material you want the fastest computer possible within given weight and size limitations imposed by the context of use. You will require a high-quality colour or greyscale video display. The minimum amount of main memory for working with reasonable sound and still image material is probably about 8 megabytes and 20 megabytes will be appreciated in most cases of serious use. For motion video, 20 megabytes is probably the minimum and 64–128 megabytes is not surplus. Likewise you will require mass storage devices with quite large capacity. Currently one minute of video with sound in a small window at half the normal frame rate takes about 4 megabytes. One minute of sound takes between 500 kilobytes and 10 megabytes depending on quality. A fast hard-disk drive with 300–500 megabytes is reasonable, with a secondary removable media optical disk drive with a capacity between 128 and 1,000 megabytes. You will also require additional hardware for capturing images and sound (Section 5.5).

Note that these are the levels required for capturing, manipulating and processing these materials, not what is required simply to access them or symbolically manipulate them in a database or similar context. This can be handled by most computers capable of supporting the mid-user software at a comfortable level.

1.4 GENERAL FEATURES OF COMPUTER SYSTEMS AND PERIPHERALS

Although software is the key to effective use of computers in social anthropology, some understanding of computer hardware is the basis of understanding what kinds of software requirements are realistic in particular situations. This is especially important for many applications in social anthropology, where minimum computing power is often not adequate.

The task determines the kind of hardware required. Many anthropologists, for budgetary or other reasons, tend to think of their requirements as being satisfied by a single computer. This is not the best way to think of the situation. Computers are tools for accomplishing goals, and it is as unrealistic to consider using a single tool for specialized operations with the computer as the tool as it is to consider taking a single adjustable wrench to a mechanical job. Ideally, a researcher will have access to a range of computers, from pen-driven ‘note-books’ to a powerful workstation.

There are a number of possible typologies we can bring to the task of deciding hardware needs. Whatever typology we adopt, it must be relative, since the quantitative measures are rapidly changing at the same time as higher demands are made on the computer resources. In 1982, for example, 64 kilobytes of main memory was a lot, relative to the demands of software of the time. In 1992 many people consider 4 megabytes as the minimum memory required to run ‘modern’ applications. However relative features may be, you will need to specify your requirements in terms of what your likely needs will be, using terms others are likely to use. One possible typology which can be used as a starting point for consultation includes the following attributes.

Processor speed

The processor speed is the speed at which the computer processor can carry out operations such as accessing memory or executing basic instructions. Speed is highly variable, and somewhat correlated with cost. Quantified statements of processor speed are misleading, especially since poor design in the rest of the system can offset any advantage a particular processor might have. You must, however, determine that a particular machine, in practice, is up to the tasks you intend to use it for. Speed is not an absolute limit on most computer applications. A high processor speed is not essential for operation of any program, but for some programs, especially those which relate to visual data, speed can make the difference between satisfactory performance and impractically slow performance. If, for example, your main application is text processing, then processor speed is a minor feature. If you are working with digitized video sequences, adequate processor speed is essential. Faster machines also make possible 'higher-level' software, where the computer can perform operations at a more symbolic level. The language Prolog, which is the basis of some of the examples in this book, is much more effective with a higher-speed computer.

Memory

Current computers require storage space for the active parts of the programs they are executing and space to store data structures. The amount of direct memory available to the computer has a direct impact on what the computer can do. Computers with very small amounts of memory can do quite a lot, but require great skill on the part of the programmer and user to do so. Modern software tends to be quite large and to use great amounts of memory. The amount of memory also contributes to the effective speed of the computer, since directly accessible memory is very fast. With the exception of word processing and numerical applications, most anthropologically interesting applications require fairly large amounts of memory.

Mass storage

Mass storage is the amount of 'off-line' storage available. Currently, this is most often in the form of magnetic or magnetic-optical disk drives. Mass storage provides a place for the long-term storage of programs and data which can be accessed by the computer. Storage size can range from a few hundred thousand bytes to several billion bytes. Some slower media can store trillions of bytes, enough to store every word written by or about anthropology since the mid-nineteenth century with room to spare. Like memory, mass storage has a very direct bearing on what kinds of tasks are practical or possible with a given computer configuration. Although the levels of textual material used by most anthropologists are fairly easy to accommodate using quite modest resources of a few megabytes, or even less for short fieldwork, some data, such as digitized photographs or moving video, require what is currently vast amounts, hundreds of megabytes or more.

Input capabilities

For a computer to be much use you must get information into it. Today most people think of this in terms of keyboards, which were adequate when most computers had terrible text-oriented user-interfaces and most data were purely textual descriptions. There are many other possibilities. One innovation which has won favour is the use of pointing devices, where you can work with data in the form of structural representations on screen. This has made programs such as wordprocessors almost trivial in operation. For some kinds of data direct structure reference with a pointing device greatly expands the possibilities of computer applications. For example, kinship data can be input from the keyboard using textual representations, but it is possible to use the conventional graphical representation of the kinship diagram. This information can be presented on the screen and a pointing device (such as a mouse or pen) can be used to identify structures in the diagram. There are other kinds of conventional data in social anthropology, such as audio or visual material and the direct input of measurements where these are central, which are either difficult or impossible to input without special equipment. However, a number of mid-range computers are beginning to come on the market with built-in microphones for audio input, and the price of adding video input is dropping rapidly.

Output capabilities

Once information has been input to the computer system, there must be a means of relating the information, or transformations of the information, in a form useful to you. The more conventional forms of output are printers and video screens. These come with a wide range of capabilities, some capable of displaying only text, others capable of reproducing photographic quality pictures. Other forms of output are audio, ranging from simple sounds to complex multivoice multi-channel reproductions, or synthesized audio. Another important form of output is to other computers, discussed in the connectivity section below.

Portability

Portability can be of considerable importance to anthropologists, since the portability of equipment can easily make the difference between being able to use a computer in the field or not. Even if size and weight at the field site itself is not an issue, lugging a large appliance around the world is not conducive for most field situations. Portable computers can be quite variable in this respect, ranging from perhaps 50 grams for a limited purpose computer to perhaps 10 kilograms. The source of power is also an issue. Some limited purpose computers can operate for up to thirty days on four AA alkaline batteries. Some portable machines cannot operate for useful amounts of time from batteries, largely owing to the demands of special output devices such as high-quality colour monitors or because the output devices incorporate a lot of specialized hardware to enhance speed or include a large amount of mass storage. However, there are now a number of small powerful computers which have daily power

demands which can be satisfied by solar charging of batteries. We can expect the power/ capability/size to improve over the coming years.

Connectivity

As the number of different computer-controlled tools increases, and the number of computers in general multiplies, connectivity has become an important issue. If you enter information on one computer it is important to be able to move it to another. This is especially the case where different computers have different basic functions and capabilities. For example, I have used up to three different computers on a given bit of field research: a small 300 gram hand-held computer to store past field data, census records and other information on my field site, as well as for recording rough notes and organizing time; a 1.5 kilogram notebook computer which is convenient for typing out fuller sets of notes and which operates for several weeks on a set of batteries; and either a laptop computer which has a wide range of computational abilities, but which requires constant recharging of batteries, or a 'transportable' system which requires mains power to operate a high-quality colour display to display visual field material in the course of some interviews. It is essential that the movement of data between these be smooth and trouble free.

While not in the field connectivity is still a very important issue, and becoming more important. Most university or other institutional computer systems are now networked, so that data can be distributed across a large number of machines. This applies to access to larger mainframes, but also to the data stored on your colleagues' computers. It is now a trivial exercise to 'log on' to virtually any properly connected computer in the world from another, and to transfer data in either direction. To take advantage of this level of connectivity requires a machine with the proper capabilities to match the networks you need to access.

Compatibility

Related to connectivity is compatibility. At the level of the different tasks, the systems that exchange data must be compatible. This is becoming much less of a problem than in the past, either because specific pieces of software are available for a range of different computers, or because the task depends only on a consistent file format, which can be maintained by special software which exists only to maintain compatibility. If you work in a group it is important to have equipment that is at least data compatible, and preferably application compatible. Many computers have software which enables them to emulate most other computer systems to some extent, if a bit slowly.

Compatibility is specially a problem in the research environment, simply because so much of the data processing is necessarily idiosyncratic, relative to 'conventional' data processing. Research applications often require special hardware or expensive software which is unlikely to be available generally. This can create significant barriers to dissemination of data, despite its electronic form. This can be alleviated to some extent by trying to use software

which can be enhanced by special hardware, but will still operate at some level of functionality using a software emulation of the hardware. This is relatively common, since most developers recognize the need for higher requirements for ‘development systems’, and the need for supporting migration to less appointed machines.

Ease of use

The more difficult a computer system is to use, the more difficult it is to apply it to a particular job. Ease of use is also, unfortunately, a difficult criteria to establish, since what is easy for one person is not so easy for others. This is demonstrated by the rather fanatical disagreements that emerge when proponents of different systems congregate. However, it is important, if possible, to gain some familiarity with different available systems before selecting one, so that you can make your own decision rather than following the advice of someone who may or may not share your value judgments or may be limited by their own experience.

Support

Support is an extremely important category, especially for the person who intends to be mainly a consumer of computer resources, rather than an author. There are different levels of support: you need support from your vendor, in case something goes wrong with the equipment; and you need support from programmers if you are using software, since it will require modification if it is part of a research programme.

If there is some degree of choice in equipment, researchers, in particular, should see what others in their area of speciality are using; you are more likely to get software directly applicable to your research. While there is usually a computing service in research institutions, their choice may not be your choice. But unless you are very skilled you must find support somewhere.

Chapter 2
Applications for ethnographic data processing

2.1
INTRODUCTION

In 1965 *The Use of Computers in Anthropology* (Hymes 1965b) appeared, the proceedings of a workshop which included anthropologists, linguists and computer scientists. This book is still uncomfortably current, despite the apparent progress in computing over the thirty years since that meeting. The substantive applications discussed and illustrated, the concerns expressed and the hopes for the future reflect much of the current literature; textual content analysis, statistical processing, the creation of databases, modelling, simulation and predictions of data archives. What we can presently add to this are largely simple elaborations and improvements in accessibility.

In some ways this similarity is comforting—it suggests that the fundamentals of anthropology remain the same, suggesting that perhaps we are, after all, becoming a discipline with a cumulative legacy. In other ways the similarity is depressing—we have not moved very far towards advancing the discipline using computers. And in other ways the similarity is false—there is a vast difference, both quantitatively and qualitatively, in how pervasively we can apply computers to our discipline, and certainly in terms of potential.

However we regard this, it illustrates an important point; computing applications in anthropology are largely driven by needs from within the discipline, not because the technology is available. We cannot expect basic needs to change just because computing has become less expensive and more accessible. What is at issue is identifying how our needs might be met (Sugita 1987; Eguchi 1987; Kippen 1988a; Dow 1992).

We can devise four basic elements to anthropological research: collection of information, analysis of that information, theory building and testing and reporting the results. These are not sequential; because of the time invested in anthropological research, these phases become intricately inter-woven. Anthropologists have the following particular problems with respect to their data.

1 Much of the data is contained in fieldnotes, which are in chronological order rather than by subject, are irregular in structure and contain many different subjects distributed within them.
2 The investment in and intensity of most of the data collection effort, often over a period of years, demands a high 're-use' value. The same information may need to be reorganized in many ways to derive maximum value.
3 For ethical, economic and practical reasons anthropologists can rarely perform experiments. Data is generally collected passively, without any form of systematic sampling.
4 The greatest strength and weakness of anthropological data is its dependence on the researcher. Not only must we rely on the researcher to collect data in a satisfactory manner, but we depend heavily on the knowledge of the researcher to assess what the data is and what its significance is when it is applied.

Although anthropology is not unique in this last point—there are many disciplines which depend on the subjective impressions of their practitioners—it is only within anthropology that subjective knowledge is routinely taken for objective knowledge *and* where the anthropologist is obliged, whether impossible or not, to minimize the impact of their own bias and prior conceptions on this subjective knowledge. It is in this latter respect that computers may ultimately have the most impact.

This chapter examines issues relating to the analysis of anthropological data using computers, and describes a number of computer-based resources which may be applied. These resources include tools for data management, data modelling and data transformation, as well as resources for improving communication of ideas and information within the discipline.

2.2 COMPUTER-AIDED PROCESSING OF ETHNOGRAPHIC DATA

At one level it would be ideal if we could use a computer without having to make any accommodations to our current ideas, methods and data. This level is, of course, magical. Put in other terms, we could say it would be nice to do anthropology without having to worry about methods and data, and indeed some have apparently taken this course, also magical. All methods, manual or not, make requirements on the underlying ideas, methods and form and content of information they are compatible with. So what we are actually discussing is how we must accommodate differently for computer-based processing over conventional methods.

Computers superficially complicate matters, because to apply a method embedded in a computer program we need not understand how the method works to get a result; we need only organize data in the correct form. Lack of knowledge regarding the method may be satisfactory for purely applied work, where fitting results into a pre-determined form is one of the principle objectives, but it is poorly suited to a research environment, where we must also

supply an interpretation to the results of the method. It is thus essential to understand precisely what a particular computer-based method is doing when it is processing data, and to have confidence that this is indeed what is happening. This confidence can be accomplished by testing against a range of known (manually verified) cases.

This section discusses a conceptual basis for relating our conceptual needs to those which can be dealt with by computer, and the basis of a method for converting between different conceptual systems.

2.2.1 Methods and data

Ideally the data and its format and preparation of data for computer processing would be identical to that for conventional analysis. While, within limits, the content of the data will be the same, in practice modifications to the format and preparation of the data are usually required to accommodate efficient processing and ease of program design. If you plan to use an existing program, the modifications are likely to be greater than if a program is being written by or for you to address your specific research needs.

The principal reasons for modifications are of three sorts. First, computer programmers (and hence computer programs) have a general intolerance for variation in the structure of data elements. If we are preparing data for manual use each entry may be a little different; a note is recorded in the margin, some bit of information is emphasized, abbreviations of different sorts are employed, additional criteria for some classes of people are recorded or some information is not available for some people. It is possible to accommodate all of these in a program specifically written for a given set of data, but at the cost of increasing both the complexity of the program (and the time to write it and test it against all possible input forms) and the possibility of undetected errors in the data. This does not mean that you have to develop a single format for each case. For example, to represent information about a given person it is quite reasonable to have one structure for demographic data and another structure for relationship information. But all demographic data should be structured in a uniform manner, and all relationship data should be uniform in structure.

Second, people are capable of making sophisticated inferences from data which are very difficult to accommodate in a program. For example, one well-known anthropologist recorded as a single data item either the wife's father's lineage or the wife's village name, depending on whether she was from the village or had immigrated to the village. This worked well for the anthropologist because he knew each person and all the village lineages and source villages, and it was perfectly obvious which was which. For the computer program to make this same inference requires all this knowledge to be represented in the program. An alternative is to have another data item which distinguishes the two, but this also creates complexity in the program and in the interpretation of the reports generated by the program. Different types of information should occupy different structural positions.

Third, computers are usually resorted to when there are a large number of cases. Computers may be able to handle a large number of cases, but the quality of the analysis is only as good as the input data and the model which underlies the transformations to the input data. Errors in data entry in a large data set are inevitable, and one important use of computer-based methods is error identification and correction. This process is greatly aided if additional information is included with the data to identify the source from which the data is derived.

The main constraints of using a computer for analysis relate to the format in which the data is presented to the computer, not the essential content of the data. If you intend to use an existing computer application to analyse your data, you almost certainly will have to meet specific format constraints on the structure of your data. As yet, there are no universal formats. This is usually not a very serious problem, but it can lead to clerical complications if you have already entered the data into a computer format and it is not compatible with the application. If you intend to use several different applications, then almost certainly the data will require reformatting for each program. Fortunately, it is not necessary to re-enter data (unless an application requires a data item which you have not previously entered). Instead you use computer applications for data management to transform the existing structure into the structure required (Section 2.3.3).

To summarize, it is usually best, if possible, to put data into a uniform structure where the interpretation of each data item in the structure is uniform (each item is of the same sort) and to include additional data items to identify the source of the data. A computer program can accept many different kinds of data, but each kind should follow these conventions.

2.2.2 Conceptualizing the problem

Before we can define how the data is to be presented to the computer for processing, we must first define:

1. the model and abstract model categories which assign the data a meaning or interpretation;
2. the analytic goals;
3. the analytic procedures we intend to apply to satisfy the analytic goals;
4. the abstract categories of the model that are necessary for the analytic procedures;
5. the concrete data collected which corresponds to these abstract categories, and the method of deriving the categories from the data.

Any useful computer program can be conceptualized as a set of operations which transforms data into a form which is considered the result, or output, of the program. Some of the transformations result in a loss of information, such as the sum of a column of numbers or a classificatory term, others simply re-order the information. For the most part, the ‘meaning’ of the transformations, and how

well 'meaning' is preserved over transformations, is external to the computer program, existing, if at all, in the head of the person who writes the program, and hopefully that of the person using the program as well. Computer programs operate on data structures, rather than on data.

Computing scientists discuss data abstraction a great deal. Data abstraction refers to the ability of the program to present data in a functional or model form suitable for the conceptual framework of the programmer or user, while 'hiding' the details of implementation of data structures from the programmer and the user of the program (Burnard 1987: 64–8).

All computer programs assume some model for the data, with the ideal being a model which is relatively close to the one the prospective user is likely to employ conceptually. The program itself might not internally store the data using this model, but it will appear to do so to an outside observer.

There is a limit to how abstract the model can be, imposed by performance constraints, time constraints on the part of the programmer and an inability to easily deal with models which are not logically complete. Efficient, and trustworthy, programs must have available all the data required for an operation. However, the form in which the data is represented within the computer application need not be the form of the data supplied to the program; the abstract data model need only be derivable from the input data model.

2.2.3 Defining problem requirements

The data which must be available are based on a specific analysis, or range of analyses, that you plan to undertake. The analyses you can undertake are based, in turn, on the data available and the quality of that data. It is not helpful to collect data, prepare a data set and then look for procedures and tools to process the data, you should plan ahead (Davis 1984b: 308). Anthropologists have become increasingly aware that when they collect data they do so in correspondence to an explicit or implicit theory or model. Analyses carried out using a computer may require specific data; however, these are not dependent on computing considerations but on conceptual and analytic considerations. You are responsible for determining that the analytic method is consistent with the assumptions which underlie the data you have available. The program need not employ an identical model to the model used to guide data collection, but the program's model should be equivalent to (or derivable from) your model.

A schema for specifying the form that input data should take is therefore related to the conceptual and analytic procedures you intend to apply to the data; in effect you must construct a specification of the specific analyses you intend to perform. An outline of this process might take the following form:

1 A brief description of what the purpose of the analysis is and a specification of the results required to achieve this purpose.
2 A brief description of the abstract data model from which these results might be achieved:

(a) the abstract categories of information the analysis requires;
(b) a definition of each category and a list or description of the range of values expected for the category;
(c) the data and method by which you expect to derive the abstract category.

3 A description of the transformations required to relate the abstract data model to the results.

This is, of course, no more than should be done for any method, computer-based or not.

2.3 DATABASE APPLICATIONS

Computer database applications in social anthropology probably represent the single most appropriate computer-based resource for anthropologists. All anthropologists have data, and many have data that has been under-used because of the difficulty of access. Database applications also require less expertise than many other computing applications. The most accessible forms of database applications for computers are rectangular, hierarchical and relational databases. An excellent introduction to these is found in Burnard (1987), Bagg (1992) and Wilson (1992).

Briefly, a rectangular (or 'flat file') database is a simple collection of *cases* (representing *entities*) for which there are identical, fixed, categories of information *(attributes)* associated with each case. Rectangular databases have the advantage that they are relatively easy to set up and simple to query, but they are limited in the kinds of information that can be extracted from them. In general, it is not possible to refer directly or indirectly to other cases within the database, except as a reference category. It would be difficult then to extract information about the sisters of men who hold a specific office or position, unless this information was explicitly coded within the case. This represents a weak form of representation, because almost all the knowledge needed to interpret the data lies outside the database, either in other written work or in the head of the researcher (Section 1.2.4).

A hierarchical database is a rectangular database with a sub-case structure. As an example, if the basic case structure represents a family, within a family we can have a variable number of households with a variable number of members. These usually require that each level of case be structured as if it were a rectangular database; each case has identical categories of information and there is little or no inter-case referencing. This is a stronger (though still weak) form of representation than rectangular databases, as we can at least express hierarchical relations between different structures. More of the context for interpretation is represented within the database.

A relational database shares with the former that it is entity/case based, but has the important addition that it represents attributes as relationships. This permits both radical reorganization of the entity relationships within the database and the referencing of information between cases. The relational database model is a

much stronger form of representation than the rectangular or hierarchical model. Although it remains entity oriented, different types of entity can be represented within the same database, and it is possible to represent many more kinds of inter-entity relationships than the hierarchical database. Unlike the rectangular and hierarchical databases, the relational database is more dynamic; it facilitates multiple views and uses of the same material, while the former present a 'flat' view of the data with far fewer options for transformation.

This section describes the background to these different database models, and illustrates some of the potentials and limitations of each. Topics include design concepts for representing data, basic terminology, representing relationships and representing data with a more active structure than these conventional databases using hypertext.

2.3.1 Design concepts for representing data units

The kinds of processing you can do with data depends very much on its structure, and how much of this structure you can communicate to the program you are using to process the data. Common kinds of data processing, beyond basic entry and retrieval of information, include selective searching based on multiple keywords, sorting data into a new order, content-based counting and establishing correlations (or their absence) between data elements.

If the information you have is basically structured as a natural language text—perfectly well structured for humans—most computing applications will regard this as 'unstructured' material. Although there is a lot of research in computing science into 'understanding' natural language texts, very little of this has manifest itself as yet in commercially available computer applications, although we might expect some progress in this direction in future. If your data is mostly textual, then the best current application type to use is the same as that for fieldnotes, a textual database, described in some detail in Section 4.6.3. A textual database will give you the capacity to make relatively complex searches, add classificatory terms to each card and select and sort by these terms, and many have some capacity for making counts and producing tables of correspondence.

Although textual databases can be very useful with a variety of 'free format' data common in ethnographic research, most computer applications for working with collections of data expect rather more structure, and, in return, can perform more specialized kinds of searching, retrieval and processing.

It is at this juncture that we must rethink our categories (and our technology). In his discussion of designing data models for museum collections, Burnard warns that,

> users…tend to see their information in terms of the particular data structure in which it has previously been physically instantiated (index cards, ledgers, textual description, etc.). Even when computerization has been embarked upon specifically in order to overcome some informational restriction inherent in the physical constraints of a manual system, it is not unusual to find users requesting that those very constraints should be

> eternally perpetuated in the very system which is supposed to do away with them.
>
> (1987:65)

The problem is that some of the features we associate with our current methods are artefacts of the technology we are using, rather than an intrinsic part of why we use these methods. We want the transition from our traditional technology to computer-based technology to be as simple as possible, and we want to expand our capabilities in fieldwork. Sometimes we can accommodate both of these objectives, and sometimes not. In the early stages of using a computer it is probably best initially to adopt computer methods that closely resemble those you are already using, but as your experience with computers grows be ready to adopt new approaches. With graphical operating systems this can be a fairly easy strategy to follow, since these operating systems have as a part of their philosophy providing *at least* the features we associate with paper and graphite technology. At the same time they make possible new methods of representation which are not possible with conventional media.

In our case we have to make the transition from ethnographic data to the kinds of resources available on computers. For example, a card in a card index uses a physical surface to relate the different bits of information on the card as representing a single entity the information relates to, such as a person, place or event. To convert this system over to a computer we must use an analogue to the card which has at least this same organizing behaviour. This function is most commonly provided by a class of computer programs generally referred to as database management systems (DBMS). There are a number of different types of DBMS, each of which is designed for different kinds of data content and structure.

2.3.2 Terminology and structures for data

A common term in computing circles for a data structure corresponding to a specific instance of our index card is a *record,* itself composed of one or more *data fields* each of which contains a specific value in any given instance. A record can be conceptualized as an organizing principle for information represented in fields. This latter sense of record is called a *record type* or *relation* and provides context for interpreting the information in the fields. A collection of record instances of the same type can be called a *table* The set of tables you are using for a particular analysis is a *database*. See Bagg (1992:2–9), Burnard (1987:66–8) and Wilson (1992:77–80) for further discussion of terminology and data models.

Data fields can range from being quite simple—a single number or word—to being themselves records or references to records. For example, we can design a ‘household’ record to represent a collection of information about households, and a ‘personal’ record to refer to a collection of information about individuals. The household record may be self-contained, it may include the personal records of the household members or we may have a relational mechanism to connect the household record to the people who are a part of the household. So, depending

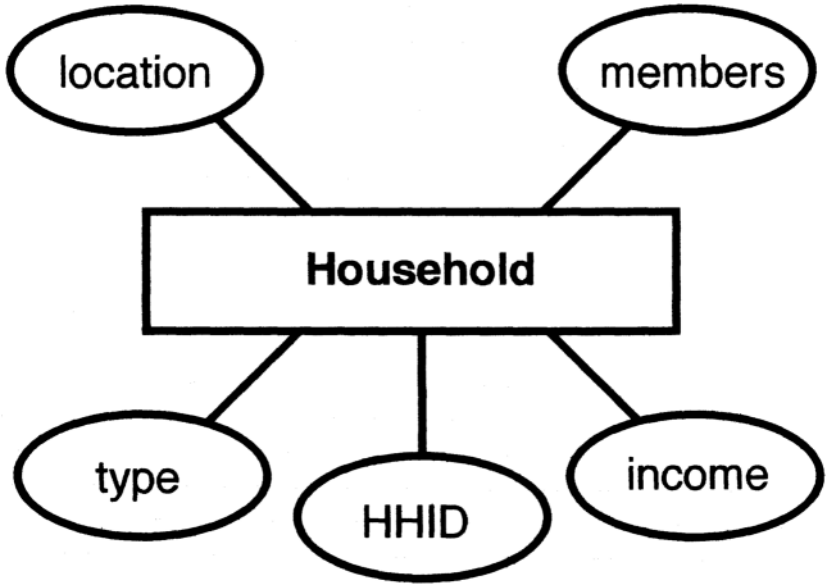


Figure 2.1 Schematic of household table

Source: Adapted from Bagg 1992:3

on how we define the relation 'household' it may be represented by a simple household record, a household record containing personal records or a collection of household records and personal records. Or, in addition to structured information, the household data unit may also include references to a household at different times, drawings, references to conversations with household members and various kinds of visual and aural recordings relating to a given household.

We can divide data records types into three rough categories:

1 record types with a fixed or regular structure;
2 record types with an irregular structure;
3 record types which have both a regular (possibly fixed) portion and an irregular (or idiosyncratic) portion.

Fixed structure indicates that each individual record within a table will be modelled using exactly the same structure. For example, we can represent a household register in terms of a card which contains a fixed set of categories of information for each household, such as 'Housing type', 'Number of household members', 'Location' etc. (Figure 2.1). This kind of structure can be easily incorporated into a rectangular fixed-format (or 'flat') database manager. Data in this form is easy to export to a statistics program or spreadsheet calculator (Section 2.4.1), which tend to assume rather simple and discrete data structures.

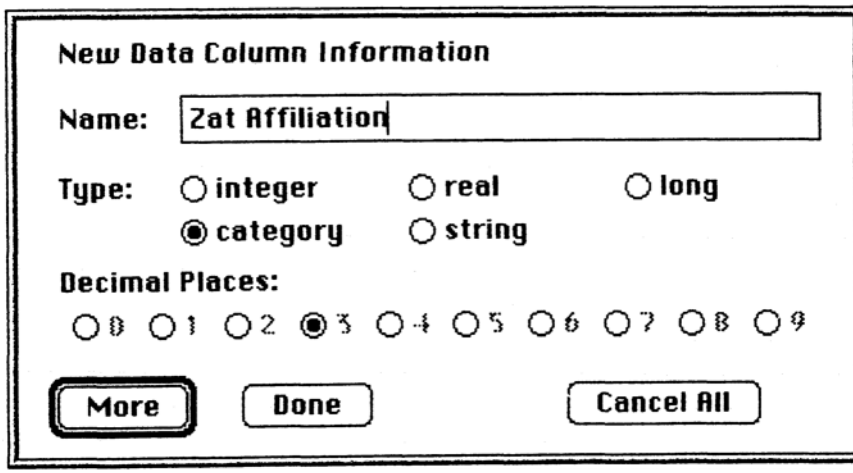


Figure 2.2 Data definition dialog for flat file database

If your data consists mainly of numbers, and the number of records is relatively small, say less than 1,000, you may find that the database management capabilities of a spreadsheet or interactive statistics program adequate for that data.

In computing terms *fixed structure* means precisely that. What appears fixed and regular to ordinary human beings is not necessarily so when we confront the computer programs which other human beings have provided. It is still not unusual when setting up a fixed structure database for a computer program to have to specify not only what type of data will be put in a particular position (word, integer number, fixed decimal point number), but how big the number will be, or how long the word will be. When inputting data to the program based on this specification of the structure, deviations from this original definition will be rejected as 'invalid' (often quite rudely). A fixed structure for most fixed format DBMS implies that each record will have exactly the same categories of data, with the contents of these categories in exactly the same format. Figure 2.2 is a typical data type definition setup for an interactive fixed format database program. The data types given are *integer, real* and *long* for numbers, *category* for a restricted range of alphabetic data and *string* for unrestricted alphabetic data. Figure 2.3 shows a table view of the beginning of a database with records representing individual information. In this example Name and House Number are type string and Block, Age and Sex are integers, although Block and Sex could be given as categories.

Although a fixed format database is easy to define, and easy to analyse within its limits, many (perhaps most) kinds of anthropological data do not easily fit into a fixed structure, especially in the research design and data collection phases of fieldwork. This is particularly the case in social anthropology, where we often define units in terms of relationships within and between units, in addition to simple discrete observable attributes. Although the fixed format may be too

mqps 1988

	Name	Block	Age	Sex	House Number and Status	Min
1	Nazeeran	1	35	1	II 721 wi 1	
2	Saeeda	1	35	1	II 424 wi 1	
3	M Ishliaq	1	28	2	II 588+589 he 1	
4	Saeeda	1	25	1	II 588+589 wi 1	
5	Shamshad	1	30	1	II 528+529 wi 1	
6	Ghafoor	1	35	2	II 528+529 he 1	
7	Saeeda	1	40	1	II 517+518 wi 1	
8	Hamif	1	35	2	II 517+518 he 1	
9	Tahim	1	35	1	II 381+382 he 1	
10	Aisha	1	30	1	II 381+382 wi 1	
11	Habib	1	40	2	II 332+333 he 1	
12	Surrya	1	35	1	II 332+333 wi 1	
13	Tayyab	1	40	2	II 421 he 1	
14	Mah Para	1	35	1	II 421 wi 1	
15	Shamin	1	35	1	II 241 wi 1	

Figure 2.3 Table view of fixed format data records

restrictive, in many cases there is still a regular structure we can use to represent the information in the record.

Records with a regular format can be composed of:

1 a fixed set of fields (data categories), of which only a subset of these are likely to be associated with any specific record;
2 fields with an indeterminate number of instances for each record; or
3 fields with no specific structure, which can take different formats in different records.

For example, if we want to assess as part of a household survey the different kinds of household items owned, this can be a very large number of items with a different set for each household. Also, in designating household members, there will be a different number of people for each household. And it is likely that we will include a field which contains notes on anything we find of interest about the household, especially of an idiosyncratic nature. These kinds of structure are rather common, especially in the field, since we cannot usually pre-determine the range of responses or information which might be encountered until they are encountered. Extending the kinds of categories as they are found permits a regular, systematic approach to defining units in a exploratory context, but requires more complex approaches for computer representation.

2.3.3 Representing data relationships

There are two basic approaches to working with records of regular structure. First, we can represent it as a fixed format record, creating a record which includes all possible fields and including a 'not applicable' code for those fields which were not actually associated with a given record. The primary advantage of this approach is that we can use the same programs that are used for fixed format data. There are many disadvantages. A great amount of clutter appears in such codings because of numerous place holders. Either a great deal of revision must be done as you must redefine the record (which many fixed format DBMS make a disagreeable procedure) or you must go back to earlier records and manually update these new fields with the 'not applicable' value; alternatively you could find or write a program that adds these to earlier records. Finally, many kinds of regular structure do not fit easily into a fixed format, even given the stated disadvantages, and even if these can be represented in a fixed structure some detail or potential for generalization is lost. A common case of this in anthropological data is the difficulty of making relationship associations; 'all men who have a sister with a hand-weaving loom' or 'women with an elder brother who is unmarried'. Although it is possible to code these kinds of facts into a flat database, the number of such facts that might be interesting in some future analysis is rather large, cannot easily be anticipated in advance and would lead to an enormous amount of data entry to cover the many inappropriate and inapplicable cases.

Generally, a better approach is the use of a relational database manager (Brent 1985; Bagg 1992; Wilson 1992), or where relational requirements are minimal, a textual database with some relational capabilities (Section 4.6.3). A fixed format DBMS generally has a one-to-one correspondence between a conceptual record or relation and the fields which define it (Figure 2.1). In a relational database the concept of record is maintained not necessarily in terms of contiguously stored, discrete fields; rather it can be constructed in terms of a set (one or more) of sub-records whose data fields can be related to construct the conceptual data record (or relation). Risking over-simplification, a relational database can be seen as one or more (usually more) fixed format tables which can be combined to define complex records. You can, of course, refer to the records of a single table as records. Using a relational database management system (RDBMS) it is possible to construct new record types by combining different fields from different relations.

A record is itself a collection of data fields. Besides the discrete content data fields, there is additional information which establishes a link between a record and related records through a unique reference name or number for each record instance, called a *key*. It is called a *primary key* when it stored in the record for which it is the identifier, and a *foreign key* when it is stored in another record as a reference. By associating a field with a value for this reference we can retrieve this referenced record using the key. As an example, information relating to a household and its members can be represented as a set of two record types, as shown in Table 2.1. Each relation has a field 'HHid'. For the household relation this is its unique identifier, a primary key, and for the person relation it is a

Table 2.1 Relational record structure for household and person

Record type	*Structure*				
Household	HHid	Type	Location	Income	
Person	Pid	Name	Age	Sex	HHid

Source: Adapted from Bagg 1992: 5

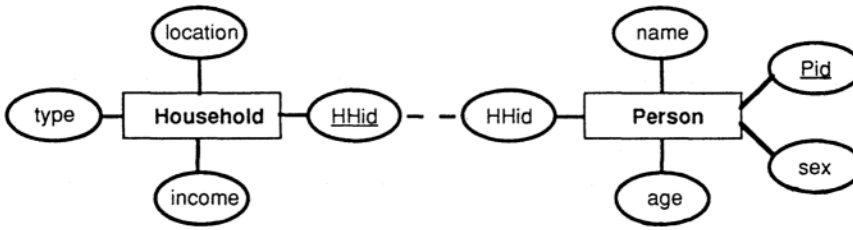


Figure 2.4 Relational record structure for household and person

Source: Adapted from Bagg 1992: 5

reference to the household the person belongs to, a foreign key. With this structure we can create reports for people and households independently. For instance, we can create a list of households with income below a certain level, or a list of people sorted by age and sex. If we want to create a list of people who live in households with a given level of income, we can instruct the RDBMS to list people whose household has a income field value within the desired level. This is possible because the 'HHid' in the person record will match the 'HHid' for the appropriate household in each case (Figure 2.4).

The most common means of instructing the RDBMS to construct a list, search for a specific record, or some other operation is through a control language, generally called a query language. There are many possible query languages which different database management programs might use, but the current trend is towards convergence towards a *de facto* 'standard', SQL (pronounced sequel) which is an acronym for Structured Query Language. As with most *de facto* standards, SQL has a number of minor variations in each implementation, but most follow the following form (in the Ingres dialect).

To make our query to list people whose household income is less than Rs1,000 we might type:

```
select p.name p.age, p.sex, h.income

from person p, household h

where h.HHid=p.HHid and h.income < 1000

order by h.income
```

The bold face terms are SQL *keywords*. The **select** line effectively defines the structure of a new relation (which can be given a name if necessary). The 'p' and 'h' on the **select** line will represent the *person* and *household* relations (see **from** line), and the '.' notation indicates that the following term is a field name within the attached relation. 'p.name' references to the name field of the p relation. The **from** line sets 'p' to represent the *person* relation and 'h' the *household* relation. The **where** line specifies a condition for accepting a candidate for the newly defined relation, in this case by matching the *HHid* fields from the *household* relation and the *person* relation and then checking to see if the household income is less than 1,000. Finally the **order by** line controls how the resulting relation will be sorted for presentation, since the majority of these new relations will simply be examined and/or printed, which generally means that you will want the output ordered. It would be possible to further order the output by age as well, which would sort the records first by income and then within the income groups by age.

This *query* will create a new table with fields name, age, sex and household income sorted by household income. This table can be exported to a spreadsheet or statistics program which requires fixed format data for further analysis. Besides simple queries, there are a number of SQL operations which will find averages and totals and perform basic arithmetic. Queries can be embedded in queries.

This makes RDBMS a powerful tool for reformating data for export to other computer applications, such as a statistics program, which have more powerful (or at least faster) analytic facilities but which have very limited capacities for managing data structures.

The advantages of this power becomes more evident if we consider the record structure in Table 2.2. This is a portion of a data structure I developed to examine households and membership and income sources over time (Lyon and Fischer, forthcoming). For this reason virtually each relation has a field 'Year' or 'Begin Year' and 'End Year'. These help me develop different views of the community in different years, following household through different houses, members and heads, and people through different households and marriages. The 'Marriage ID (of Parents)' field in the person relation is enough information to reconstruct kinship relationships (in conjunction with the 'Sex' field) between households at different stages of development of the community (following the model developed in Chapter 6). Although constructing SQL queries for some of these relationships can lead to complex statements, it is far easier than trying to work these relationships out using a simpler DBMS.

Only information which is present is recorded. It is possible with a relational database manager to write out special purpose sets of these categories in a fixed structure for import into a fixed format DBMS, a spreadsheet calculator or a statistics program. With a relational database we can produce fixed format reports consisting of complex categories of information corresponding to criteria such as 'occupation of brother's wife's brother in 1985', assuming we have the data relations 'brother', 'wife' and 'occupation' associated with each person.

Simple regular structures can be handled by many plain-text database managers (Section 4.6.3). These are much less efficient than most relational database managers for highly structured data, but have the advantage of greater

Table 2.2 Complex relational structure for household and person

Record type	*Structure*			
Person	Person ID	Year of Birth	Sex	Marriage ID (of Parents)
Marriage Link	Marriage ID	Person ID		
Marriage Tenure	Marriage ID	Begin Year	End Year	
Occupation Link	Person ID	Occupation	Year	
Household	Household ID	Number of Members	Year	
HH House Link	Household ID	House ID	Begin Year	End Year
HH Person Link	Household ID	Person ID	Begin Year	End Year
HH Income Link	Household ID	HHincome	Year	
HH Head Link	Household ID	Person ID	Year	

ease in revising the structure as you proceed with data collection, since plain-text databases make fewer formal demands with respect to structure and value. They also make it possible to combine regular data with free-format text, which helps to maintain context. After completion of your database using a plain-text database manager, it is fairly easy to export to a relational database or spreadsheet for further reporting and analysis.

2.3.4 Hypertext: representing irregular and idiosyncratic data with objects

Relational database applications support a more structured model for the maintenance and analysis of social data, and have been widely used for several years. Although little analysis is done directly with these applications, the researcher can dynamically reformat data for input to other applications, while insuring the integrity of the database. Most are presently limited to textual data, with a few supporting simple graphics. Relational databases can be difficult to accommodate to social research if the study includes much idiosyncratic information about subjects, since to take advantage of the power of the relational database application it is necessary that a uniform (if flexible) structure be imposed on the data. Hypertext systems make no such assumptions about data, but this entails that much of the structure be imposed manually.

Irregular data structure is, as you might expect, rather difficult to describe in a regular manner. It cannot really be completely irregular, or it could not be represented. However, I am using irregular more with respect to conventional

computing resources than an intrinsic property of the information. Record types which include idiosyncratic information relating to an object or event the record represents are irregular in this sense. The other sense of 'irregular' I am using refers to data which has a number of different 'senses' and structures of which it is a part, and where you can make decisions and access these data directly.

Much data collected in the field has both regular and irregular information. The irregular information may take the form of specific notes made at the time of collection or transcription, unusual details which are not likely candidates for a general usage, or non-textual data, such as sketches, photographic reproductions, video records or sound recordings (Section 3.3), which can be recorded directly into computer-based documents managed by a hypertext/hypermedia program. Hypertext documents with this degree of flexibility appear to be a resource of immense potential for anthropologists, especially those who already work with non-textual materials, as well as those with complex textual material.

The concept of hypertext more or less predates computers and is generally attributed to Vannevar Bush (Nielsen 1990:29), who conceptualized a system with associative indexing which directly ties two (data) objects together so that one is reachable from the other. Bush visualized a system where information was stored on microfilm and linked mechanically. His system was never built, but with the advent of computers similar systems were developed until the mid-1980s when commercial systems became available.

The basic idea of hypertext is that any portion of the 'text' (which now conventionally includes non-textual data such as images) can be 'objectified' and linked to another such object. As an example, if we have two pages of fieldnotes, we can designate any sequence of words in one note as an object and link it to the other page of notes (or any number of pages of notes), or to an objectified sequence of words in that note. Likewise, we can objectify a sketch map, or any portion of a sketch map, and link it to another object, and/or vice versa. A good general introduction to hypertext is Nielsen (1990).

The commercial development of hypertext systems provides another important tool for studies based on texts, as well as other types of data management and analysis. In a hypertext document, the text can be dynamically reorganized to suit specific research needs. Rather than a single linear order, processed piecemeal to provide concordances or Key Word in Context (KWIC) reports (Section 4.5), hypertext can maintain multiple organizations and multiple orders for the content of the text. These orders can be automatically compiled (to a limited extent), or represent material that the researcher has selected, while preserving direct access to the original context of the material. In addition to text, hypertext systems can usually support human and computer drawn graphics, digitized photographs and recorded sounds. With a suitable hypertext program such as Guide, or HyperCard, textual material relating an episode such as a neighbourhood argument can be integrated with digitized video frames of the event itself, with the video linked to other frames, the relevant text, and vice versa. Related developments are interactive videodisc collections, such as that produced by A. Macfarlane (Macfarlane *et al.* 1987; Macfarlane 1990) who collected on videodisc about one-third of the known textual and visual material on the Naga, a society in India (Section 2.5).

It may well be that the best suited applications for working with irregular or partially irregular information are hypertext programs (see also Section 4.6.5). Most dedicated plain-text database managers are suitable where only text is used in a compilation and where only limited and structured linkage is required, but hypertext programs can be used where non-textual data is to be included in addition. Many plain-text databases are built using hypertext programs, so hypertext is probably the best general purpose approach. With these programs it is possible to search and retrieve for inspection the irregular information based on searches using the regular information, and also to export the regular information for processing by other programs.

Hypertext programs have another advantage, which to me is compelling, at least for use in the field. Besides the capacity to incorporate data of different types and formats in a single hypertext document, most hypertext programs have highly interactive means of modifying individual data units, and facilities for making systematic or *ad hoc* links to other hypertext documents at any time in the process of creation, modification or analysis. Most computing analogues to conventional data representation have advantages to paper with respect to speed of access and analysis, but are relatively cumbersome to manipulate except in highly restricted ways, often requiring substantial experience to use flexibly. Hypertext on a sufficiently powerful computer is better than paper; not only is access and modification virtually unrestricted, but hypertext documents can directly represent references to other documents as actions; activate the reference and the referenced data unit is available for inspection and modification. A single data unit can be referenced in many different places in the hypertext document as if there were multiple copies. However, unlike copies, if any modification is made to the data unit, this change is reflected wherever there is a reference to the unit (unless a 'true' copy is made).

For example, I have constructed a set of hypertext documents representing my field material from Lahore in 1982 (Figure 2.5). These include my fieldnotes, a book draft, surveys on marriage choice, a socioeconomic survey, a household survey, medical records from a clinic, passive and active surveys for malaria, a domestic animal register, some digitized photographs and geographical information relating to households. These are interconnected in a number of ways. The fieldnotes and book draft can be searched by keyword. There are a number of places in the text of these that reference the various surveys, for example where I am writing about a particular person who has clinic records or a household record. Likewise, it is easy to move from survey record to survey record relating to a person or household. There is a 'front end' which does cross-tabulations of the surveys. The resulting tables have 'active' cells. I can point at a cell, click the mouse, and a list of people or households (as appropriate) is displayed. One of these can be chosen, and links to survey records of the person or household can be directly examined, or references in the notes relevant to the person or household can be collated.

Little of this interlinkage between data units was automatic (unlike the links between table cells and data units); it has taken a great deal of work to create the links. But most of these were made in the ordinary course of using my research material. The links are more or less the equivalent to the side-notes or indexes I

would have made otherwise. In other words, I was able to directly represent in the documents my discoveries about these documents.

2.4 QUANTITATIVE METHODS

Until recently probably the major use of computers in anthropology, aside from word processing, was for quantitative applications. Indeed until the early 1980s it was difficult to find any application to anthropology which did not have a significant quantitative dimension. For recent contributions on quantitative computing in anthropology see articles in Boone and Wood (1992), which has a good bibliography on numerical methods, and the journal *Quantitative Anthropology*. For anthropological numerical methods in general see Johnson (1978), De Meur (1986), Mitchell (1980), and White 1973.

This section covers two topics. First, a description of spreadsheet calculators, which are simple, general and flexible enough to be used by anyone for basic quantitative tasks, whether these be simple statistical analyses or simply keeping accounts for fieldwork. Spreadsheet calculators are one of the few applications which are sufficiently standardized to make a detailed example useful. Second, a brief description of a development which addresses one of the most serious problems of applying quantitative methods to anthropology, the implausibility of meeting standard sampling requirements and a more direct method for relating quantitative analysis to qualitative analysis.

2.4.1 Spreadsheet calculators

Spreadsheet calculators as a class of computer program are useful for a wide range of numerical tasks, ranging from table preparation to simulation modelling. Although the majority of social anthropologists are not primarily numerical in their approach, there are few who make no use of numerical methods, even if this is restricted to the compilation of tables to represent aggregated information. Spreadsheet calculators greatly improve the handling of tabular information, as well as simplifying the preparation of tables for publication.

Spreadsheet calculators are simple and general enough to support those anthropologists who regularly use numerical methods in association with their research and provide a resource to simplify numerical applications for other anthropologists. Because spreadsheet documents can usually be saved in interchange formats compatible with different spreadsheet programs, it is fairly easy to exchange small databases and models with other anthropologists. Most spreadsheets produce printed output suitable for camera-ready copy (with the proper printer (Section 1.3.2.1)).

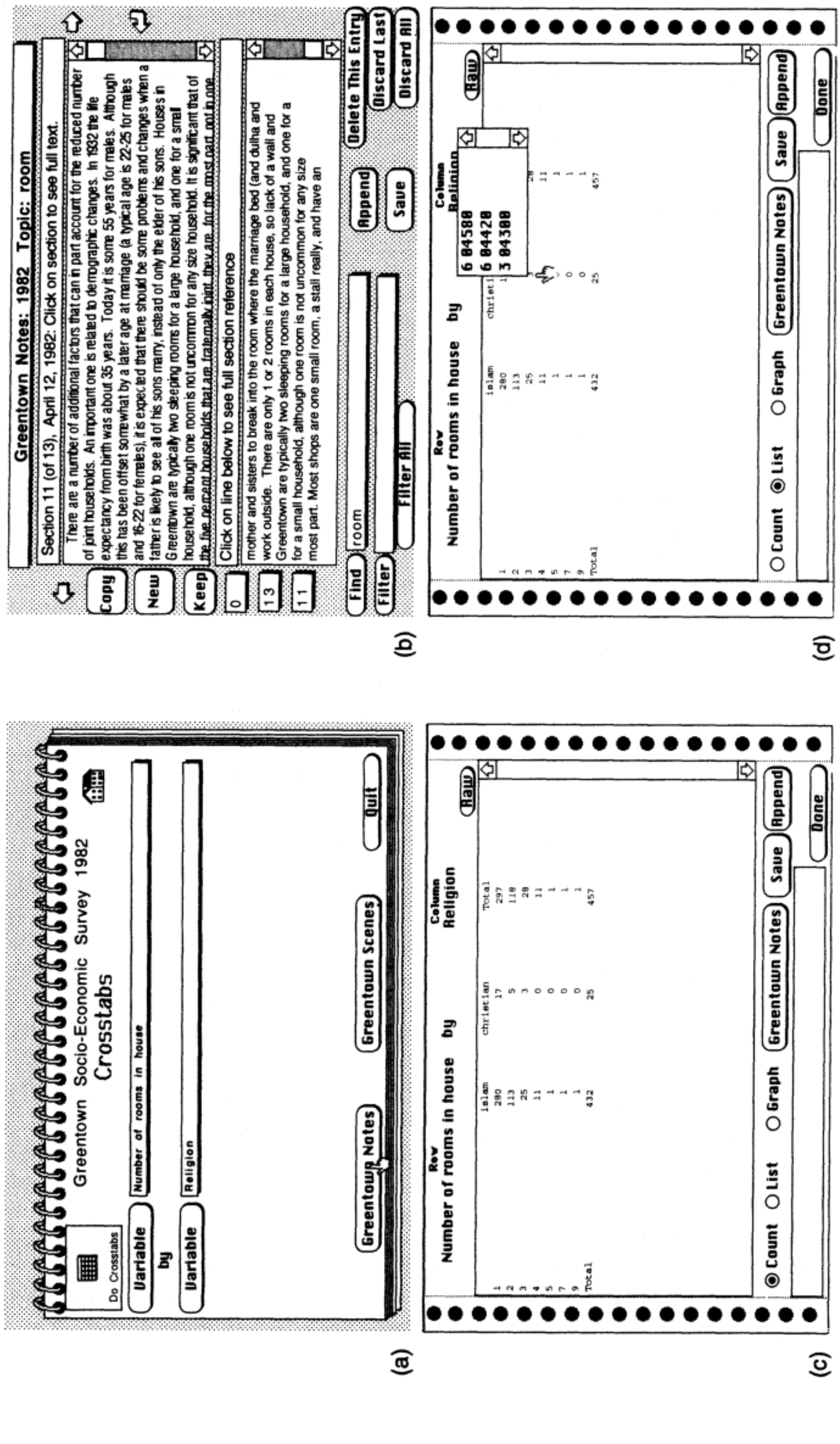
The computer-based spreadsheet calculator was the invention of D. Brickland in 1976, based on an accountants ledger spreadsheet. This first program, marketed as VisiCalc™, has the distinction of being the first application available on microcomputers which was not available on larger computing platforms, and contributed considerably to the success of Apple Computer Inc., whose Apple II was the original computer on which the program operated. More than any other single 'event' this program set the context for the 'micro-revolution' in the early 1980s.

As with most useful general purpose tools, the underlying idea is simple and powerful. A spreadsheet is a model of a large sheet of paper divided into cells which are organized into rows and columns. Each cell will accept an expression; which may be a literal number, word or formula. Conventionally, most spreadsheets designate columns with alphabetic letters and rows with numbers, as in Figure 2.6. Individual cells are designated by the column and row the cell intersects; 'A1' for the upper left-hand corner, 'B1' for the cell immediately to the right of 'A1'. There are usually a large number of cells available, usually in the order of 2,600 to over 65,000 (depending on the amount of memory available).

The original purpose of the spreadsheet calculator was as an aid for computing and laying out tabular reports. The important innovation was with respect to how the values of cells could be designated. Cells could have a constant value, such as a particular numerical value or an alphabetic string (usually a label), but they could also have a formula as their value, yielding a result based on one or more of the other cells in the spreadsheet. In tabular reports this supports the basic but useful operation of calculating the row totals, column totals and grand total based on formulae in the marginal cells.

However, what led to eventual success was that when the constant cells in the layouts are given new values, the formula values are automatically recalculated. This means that it is possible to layout speculative reports and 'play' with the numbers to project the results of different eventualities, so-called 'what-if' analysis. In other words, it is possible to model with a spreadsheet, as well as carrying out repetitive tabular calculations. The spreadsheet concept is sufficiently powerful to represent a new approach to non-procedural programming, conceptually representing problems in a form accessible to non-computer professionals.

Since their introduction spreadsheet calculators have been vastly expanded in functionality. In addition to basic spreadsheet operations, most spreadsheet calculators include facilities for manipulating textual data, limited database operations and graphing and programming facilities, and a few include word processing and desktop publishing capabilities.


(a)
Do Crosstabs
Greentown Socio-Economic Survey 1982
Crosstabs
Variable
Number of rooms in house
by
Variable
Religion
Greentown Notes
Greentown Scenes
Quit
(b)
Greentown Notes: 1982 Topic: room
Section 11 (of 13), April 12, 1982: Click on section to see full text.
Copy
New
Keep
Click on line below to see full section reference
Find
Filter
room
Filter All
Append
Save
Delete This Entry
Discard Last
Discard All
(c)
Row
Number of rooms in house
by
Column
Religion
Raw
Count
List
Graph
Greentown Notes
Save
Append
Done
(d)
Row
Number of rooms in house
by
Column
Religion
Raw
Count
List
Graph
Greentown Notes
Save
Append
Done

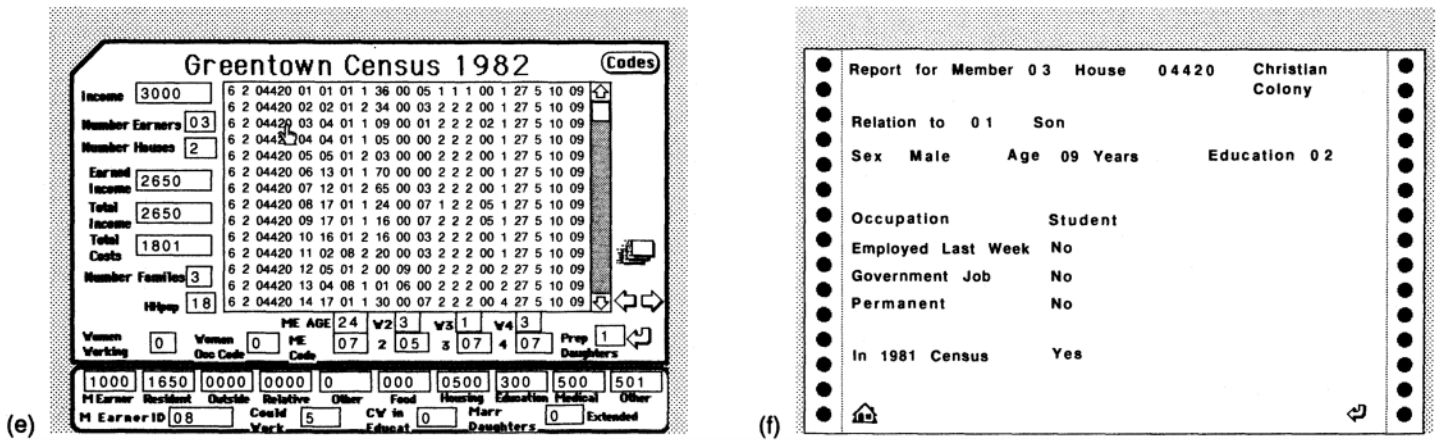


Figure 2.5 Program displays illustrating different levels of data access in field database, unstructured notes, table summary of structured data, household and individual data records: (a) opening display; (b) fieldnotes access; (c) cross-tabulationM of structured data; (d) list of households in table cell; (e) household record; (f) individual census record

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						

Figure 2.6 Spreadsheet layout

2.4.1.1
A simple anthropological application

This section illustrates the operation of a spreadsheet calculator using a simple application useful to social anthropologists. The example is intended only to convey a sense of how spreadsheet calculators work by demonstrating how some common models can be translated into spreadsheet form.

One basic use is the representation of tabular material. Goody and Buckley (1980), investigating the division of agricultural labour by sex, adapted data from the *Ethnographic Atlas* (Murdock 1967) to examine comparative interactions with other social and cultural variables. I develop an example based on the relationship between gender-based participation in agriculture and community size (Goody and Buckley 1980: 35, data drawn from their Table 2).

Data is entered into a spreadsheet by *selecting* a cell, using either keyboard commands or a pointing device depending on the program, and typing the value for the selected cell. If we enter just the labels and body data for the selected variables, we get the layout represented in Figure 2.7(a). The row and column totals could be entered directly as a literal number, but we can also enter a formula for each total. Formulae are usually entered by typing an '=' followed by the formula. References to other cells in the formula are designated by using their column and row. For the first row total the formula entered into cell 'E4' would be '=B4+C4+D4' (in most spreadsheets the cell references can be entered by selection, rather than manually typing the reference). The totals in Figure 2.7(b) are automatically calculated using the formulae in Figure 2.7 (c).

	A	B	C	D	E
1		Community Size and Sex Roles in Agriculture			
2					
3		Female Participation	Male Participation	Equal Participation	Total
4	0-99	35.00	17.00	33.00	
5	100-399	56.00	32.00	36.00	
6	400-1000	19.00	16.00	15.00	
7	1000+	7.00	55.00	25.00	
8	Total				

(a)

	A	B	C	D	E
1		Community Size and Sex Roles in Agriculture			
2					
3		Female Participation	Male Participation	Equal Participation	Total
4	0-99	35.00	17.00	33.00	85.00
5	100-399	56.00	32.00	36.00	124.00
6	400-1000	19.00	16.00	15.00	50.00
7	1000+	7.00	55.00	25.00	87.00
8	Total	117.00	120.00	109.00	346.00

(b)

	A	B	C	D	E
1		Community Size and Sex Roles in Agriculture			
2					
3		Female Participation	Male Participation	Equal Participation	Total
4	0-99	35.00	17.00	33.00	=B4+C4+D4
5	100-399	56.00	32.00	36.00	=B5+C5+D5
6	400-1000	19.00	16.00	15.00	=B6+C6+D6
7	1000+	7.00	55.00	25.00	=B7+C7+D7
8	Total	=B4+B5+B6+B7	=C4+C5+C6+C7	=D4+D5+D6+D7	=B8+C8+D8

(c)

Figure 2.7 (a) Spreadsheet with labels and data entered; (b) completed spreadsheet with totals formulae entered; (c) completed spreadsheet showing marginal totals formulae entered

2.4.1.2 Transformation

The process described in the last section might appear to be a great deal of trouble just to deal with a small table. The advantage of representing even a simple example such as this within a spreadsheet is that it is now available for further analysis. Figure 2.8(a) shows a new table (in the same spreadsheet; note the new row references) of row percentages, which is composed of references to the original data in Figure 2.7(a).

Figure 2.8(b) shows the formulae which compute the values. The cell formulae *and* their spatial organization constitute a computer program for calculating a table of cell percentages relative to their respective row totals. Even this program might appear to entail rather a lot of typing for such a simple task. However, in most cases a spreadsheet program is highly repetitive in that there are relatively few kinds of cell. In the present example there are label cells, row total cells, a total cell and body cells.

In creating the spreadsheet portion in Figure 2.8(a), I directly entered only five cells, one for each kind of cell (plus the subtitle, which is unique), and then copied each 'prototype cell' into other cells of the same type. For example, I directly entered the value for 'B14' and then instructed the program to copy this definition into all the other body cells (each program does this differently; check the manual for the specific method). This is possible because each body cell performs an identical operation using different source cell references. For example, the operation represented for the body cells in Figure 2.8(a) is equivalent to the more conventional formula:

$$\text{row percent cell}_{ij} = \frac{\text{data cell}_{ij}}{\text{row total}_{j}} \times 100$$

Normally, references are relative, which means that if a cell definition from one cell is copied by the program into another cell, the column and row of cell references in the copied definition are updated relative to the new cell. In the case of the percentage formula in Figure 2.8(b), the reference to column 'E' must be fixed in each cell (since it represents the row total), but the row reference must change to that of the cell into which it is copied. A common convention for fixing the row or column references is to place the character '$' in front of the reference to be fixed. For example, '$E4' designates a fixed reference to column 'E' and a relative reference to row '4'.

Similar tables could be made for column percentages or total percentages. For column percentages a prototype formula entered into cell 'B14' must reference cell 'B$8' rather than cell '$E4', to keep the reference to the column totals. A table for total percentages must use the reference '$E$8' to freeze the reference to the single cell 'E8', which contains the total for the table.

	A	B	C	D	E
12		Community Size and Sex Roles in Agriculture			
13	Row Percentages				
14		Female Participation	Male Participation	Equal Participation	Total
15	0-99	41.18	20.00	38.82	100.00
16	100-399	45.16	25.81	29.03	100.00
17	400-1000	38.00	32.00	30.00	100.00
18	1000+	8.05	63.22	28.74	100.00
19					

(a)

	A	B	C	D	E
12		=B1			
13	Row Percentages				
14		=B3	=C3	=D3	=E3
15	=A4	=B4/$E4*100	=C4/$E4*100	=D4/$E4*100	=B15+C15+D15
16	=A5	=B5/$E5*100	=C5/$E5*100	=D5/$E5*100	=B16+C16+D16
17	=A6	=B6/$E6*100	=C6/$E6*100	=D6/$E6*100	=B17+C17+D17
18	=A7	=B7/$E7*100	=C7/$E7*100	=D7/$E7*100	=B18+C18+D18
19					

(b)

Figure 2.8 (a) Row percentages of data in Figure 2.7(a); (b) formulae used to create the report in part (a)

2.4.1.3 Modelling

Besides creating transformations of the original data, we can apply models as an aid to analysis. For example, we can model the internal table values we might expect to find if there were no relationship between size of community and gender-based participation in agricultural labour. One established numerical procedure for developing such a model is based on the relationship:

$$\text{expected value for cell} = \text{column total} \times \frac{\text{row total}}{\text{grand total}}$$

This is equivalent to

$$\text{expected cell}_{ij} = \frac{\text{column total}_i \times \text{row total}_j}{\text{total}}.$$

Applied across the data in Figure 2.7(b) we get the table in Figure 2.9(a), using the formulae in Figure 2.9(b). This model yields a table in which two conditions are satisfied. First, the internal structure of each *individual* variable is retained, i.e. there are the same number of cases for each value of the variable. Second, the values are distributed in the cells of the table based on the relative proportions of the row and column values in which the cell lies, which is what we expect if there is no tendency for a value of one variable to imply a specific value in the other.

2.4.1.4 Evaluation

We can evaluate the model of no interaction by building another table which compares the observed data with the expected data, using the model of χ^2 (chi-square), based on the relationship:

$$\chi^2 \text{ cell}_{ij} = \frac{(\text{observed cell}_{ij} - \text{expected cell}_{ij})^2}{\text{expected cell}_{ij}}$$

This results in the table in Figure 2.10(a), in which each cell is a contribution to the overall χ^2 value found in the **'Total'** column and row. Note that at the bottom of the table there is a reference to degrees of freedom for the table, entered as a constant formula in this case for information, and an indication of the significance of the statistic at $p < 0.01$. The latter is calculated using a conditional formula in cell 'C37' (Figure 2.10(b)).

Conditional formula in spreadsheet calculators usually take the form:

if(condition expression, true expression, false expression)

where the condition is usually a comparison using '=', '<' and '>', but can be any expression (constant or formula) which evaluates to zero (false) or not zero (true). In Figure 2.10(b) cell 'C37' compares the total χ^2 with the critical value for $p < 0.01$, giving the value 'yes' or 'no' depending on the result of the comparison.

If this still seems a bit verbose simply to solve a problem of establishing

	A	B	C	D	E
20		Community Size and Sex Roles in Agriculture			
21	Expected Values if no interaction				
22		Female Participation	Male Participation	Equal Participation	Total
23	0-99	28.74	29.48	26.78	85.00
24	100-399	41.93	43.01	39.06	124.00
25	400-1000	16.91	17.34	15.75	50.00
26	1000+	29.42	30.17	27.41	87.00
27	Total	117.00	120.00	109.00	346.00

(a)

	A	B	C	D	E
20		=B1			
21	Expected Values if no interaction				
22		=B3	=C3	=D3	=E3
23	=A4	=B$8*$E4/$E$8	=C$8*$E4/$E$8	=D$8*$E4/$E$8	=B23+C23+D23
24	=A5	=B$8*$E5/$E$8	=C$8*$E5/$E$8	=D$8*$E5/$E$8	=B24+C24+D24
25	=A6	=B$8*$E6/$E$8	=C$8*$E6/$E$8	=D$8*$E6/$E$8	=B25+C25+D25
26	=A7	=B$8*$E7/$E$8	=C$8*$E7/$E$8	=D$8*$E7/$E$8	=B26+C26+D26
27	=A8	=B23+B24+B25+B26	=C23+C24+C25+C26	=D23+D24+D25+D26	=E8

(b)

Figure 2.9 (a) Expected values relative to column and row totals in Figure 2.7(b); (b) formulae used to create part (a)

	A	B	C	D	E
29		Community Size and Sex Roles in Agriculture			
30	Chi Square Contributions				
31		Female Participation	Male Participation	Equal Participation	Total
32	0-99	1.36	5.28	1.45	8.09
33	100-399	4.72	2.82	0.24	7.78
34	400-1000	0.26	0.10	0.04	0.40
35	1000+	17.08	20.43	0.21	37.72
36	Total	23.43	28.63	1.93	53.99
37					
38	df (Degrees of Freedom)		6.00		
39	P < .01		Yes		

(a)

	A	B	C	D	E
29		=B1			
30	Chi Square Contributions				
31		=B3	=C3	=D3	=E3
32	=A4	=(B4-B23)^2/B23	=(C4-C23)^2/C23	=(D4-D23)^2/D23	=B32+C32+D32
33	=A5	=(B5-B24)^2/B24	=(C5-C24)^2/C24	=(D5-D24)^2/D24	=B33+C33+D33
34	=A6	=(B6-B25)^2/B25	=(C6-C25)^2/C25	=(D6-D25)^2/D25	=B34+C34+D34
35	=A7	=(B7-B26)^2/B26	=(C7-C26)^2/C26	=(D7-D26)^2/D26	=B35+C35+D35
36	=A8	=B32+B33+B34+B35	=C32+C33+C34+C35	=D32+D33+D34+D35	=B36+C36+D36
37					
38	df (Degrees of Freedom)		=3*2		
39	P < .01		=if(E36>16.812,"Yes","No")		

(b)

Figure 2.10 (a) Contributions to , Total and Significance; (b) formulae used to create part (a)

evidence for an interrelationship between community size and the sex roles of labour, we have accomplished a bit more than this. Because the only substantive constant references are in the original table represented in Figure 2.7(b), if we change the labels and data values to those of any other three by four table, we have a new report. In other words, we have written a program which can be applied to any table of the appropriate dimensions and which calculates a table of row percentages, expected values and χ^2 relative to the new data. Indeed, if we were to define the original table area as a six by six table, were willing to put up with a few blank or zero cells on the edges of the table, developed some conditional expressions for computing the χ^2 contributions (to protect against 'division by zero' errors) and developed a conditional mechanism to insert the correct critical value for the statistic (or if we simply manually inserted the value), we would have a general program for any table up to six rows by six columns.

Generalization usually has a price; you must understand the tools and possible contexts of use much better than in a less general case. Because of the use of the spatial layout in a spreadsheet it is relatively easy to lay out a general solution for a specific spatial form, and rather more difficult where the form must change. Spreadsheet calculators offer the advantage that they are accessible at a relatively basic level for doing basic work, and have the potential for quite sophisticated applications once more experience and knowledge is acquired.

2.4.2 Logic and variation

One means of analysing qualitative material such as ethnographic data results in a series of logical relations and entailments which appear to characterize relations in the data. While the use of logic as a formal language for the representation of scientific descriptions needs little justification, given a successful run of some 2,000 years, there are some specific problems with applying logic to data as opposed to idealized scenarios. A formal language, by definition, controls the form and structure of arguments. The problem of classification or measurement which establishes the mapping of the formal structure to observations is crucial. It is important to have a formal language which has adequate mechanisms for representing essential aspects of the phenomena under investigation (Fischer and Finkelstein 1991).

For the most part, aspects of research relating to logical representation have focused on methodologies for defining specifications of ethnographic data suitable for formal expression and subsequent analysis (D'Andrade 1976). One area of logic which does require attention with respect to social science data is variation. Any systematically collected body of data about humans has variation, and a formal language which cannot represent this aspect of human productions is of limited value. Logic, as it has generally been used, is designated for qualitative analysis and not for quantitative studies, where variation is generally 'dealt with'. Much variation exists simply because there are often a large number of distinct means available to people for achieving a desired result. Thus, even if we can establish a dependent variable or goal state, we may find few, if any,

significant correlations to any independent variables, although because of our qualitative analysis we are convinced they participate in the 'solution' of the problem. Indeed it is sometimes the case that both the presence and absence of a particular independent variable may be a necessary component in different derivations, which defies most statistical techniques.

Behrens (1990) introduces a method based on using computers to integrate the relationship between quantitative and logical analysis, providing a quantitatively sound foundation for qualitative analysis. In order to incorporate both rich ethnographic data and survey data into formal representations, Behrens developed an method which he calls BART ART (Boolean Analysis with Randomization Tests). Essentially Behrens combines computationally intensive randomization tests (Johnson and Behrens 1989) with Boolean minimization algorithms (Ragin 1987). Randomization tests are used to develop a model truth table at a given level of significance by assigning most probable outcomes to conditions. Boolean minimization is then applied to remove redundancy from the model truth table, which results in a formal model which can be used to express logical relationships in the data, which themselves can be related to other qualitative data for confirmation or rejection.

BART overcomes a number of quantitative problems such as sample sizes and dependence among cases when analysing ethnographic surveys or data derived from informal interviews by using randomization techniques, which derive confidence intervals directly from the data by repeated randomized sampling rather than by comparison with standard parametric distributions whose use is often suspect because our data (or its collection) violates some crucial assumptions underlying these distributions.

BART focuses on establishing the significance of sequences of bivariate variables and their relation to a bivariate outcome. For example, if we have four variables we want to relate with an outcome, BART establishes the predictive value and significance of each of the sixteen possible states of the four variable group with respect to the outcome with a value of true (or false). Those state sets which fall within the accepted confidence interval are then subjected to Boolean minimization to yield a minimal disjunctive normal form model to represent the original data. This model represents a formal model of the original quantitative data, and can be subjected to further analysis. For example, you can apply De Morgan's Law to establish a model of the opposite state of the outcome variable, which can then be tested against the data as a check on the model. Or the model can be tested against, or integrated into, other model elements which may have been derived using qualitative methods. Because the model is a formal model, it is a 'strong' representation of the quantitative data, with results which have direct interpretation with respect to richer ethnographic data.

Although this is altogether too brief an account, I mention this for two reasons. First, it demonstrates that computers can make possible new possibilities for analysis (both quantitative and qualitative) which break down many of the objections one camp aims towards the other. Second, it illustrates that real advancements in new applications of computing are more likely to happen when anthropologists are directly involved in the computational details, as Read and Behrens conclude:

> Writing software and not just being a consumer of software is feasible and necessary if the full potential of the microcomputer for anthropological research is to be realized.
>
> (1992: 250)

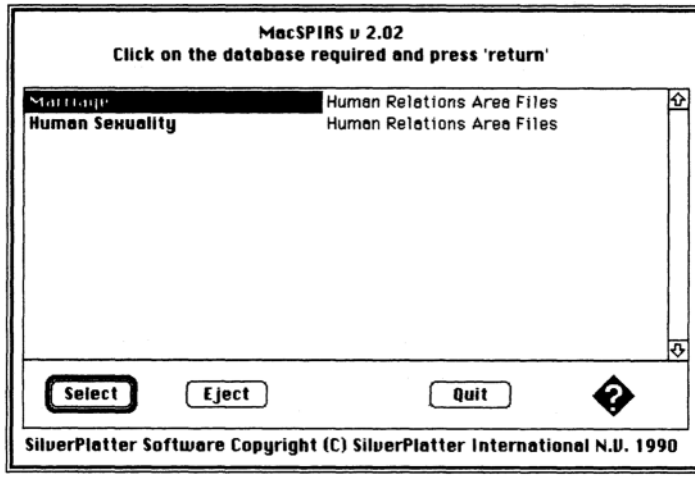
2.5 CD-ROM

A CD-ROM (Compact Disc Read Only Memory) disc is used to distribute large quantities of data, using the same technology as the more familiar digital music discs. The basic advantages of CD-ROM are a large storage capacity, about 700 megabytes (= 2^{20} bytes=1,048,576 bytes) at a relatively low cost. In more familiar terms this represents about 350,000 pages of single-spaced text, about enough for all of the text of all the volumes of all anthropological journals, complete with a detailed index.

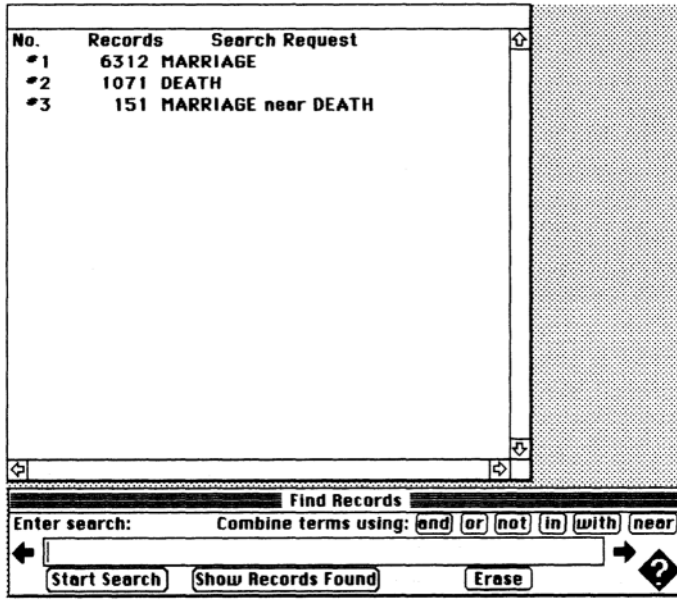
CD-ROM is a technology which has the power to transform social anthropology. For perhaps the first time in history it is possible to create and inexpensively distribute huge amounts of complex information with few constraints of length and inclusion of photographs and, using hypertext, giving the author the opportunity to encode alternative paths through the material, or even allowing the reader to construct their own. The opportunities for ethnographic publishing suggest documents which include the finished text, with links to the original field data the text is based upon.

CD-ROM, as the name implies, is a read-only media; there is no direct means to alter the contents once the disc is produced. It is random access storage, which means that any data on the disc can be accessed in any order. It is very slow compared with most other forms of computer disk storage, though much faster than magnetic tape. Many computer scientists are already predicting the demise of CD-ROM in the near future because of these defects and predictions of technically superior storage technology. This is unlikely, although as I write this I am reminded of an article of the late 1970s in *Kilobaud,* a now defunct computer hobbiest magazine, entitled ‘Paper tape is here to stay’. However, any company which can manufacture the music discs can manufacture the data discs. With the rise of CD-I (Compact Disc Interactive), the players for the data discs will merge with the players for the music discs. The recent introduction of Photo-CD™ by Kodak Corporation, and its promotion by the commercial leaders in operating systems, will help ensure the position of CD-ROM for some time. The means for manufacture and the means for accessing the discs are quite independent of the computer industry. Although technically superior read/write media are likely to emerge by the end of the century, it is unlikely that these will be able to compete with the very low cost of the CD-ROM disc, currently less than $1.50 each in a quantity of 1,000, or expect to find as many people capable of using these newer format discs. It is already possible to produce CD-ROM compatible discs which you create yourself, although this is presently very expensive.

Because of the development of CD-ROM as a commercial product, it is

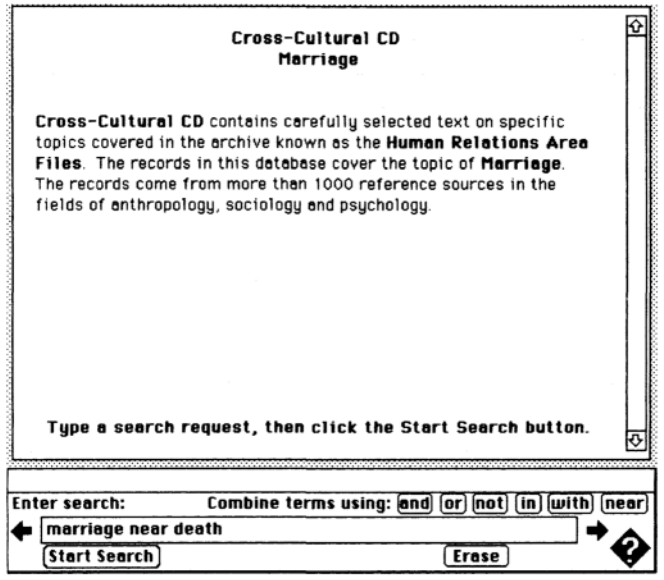


(a)

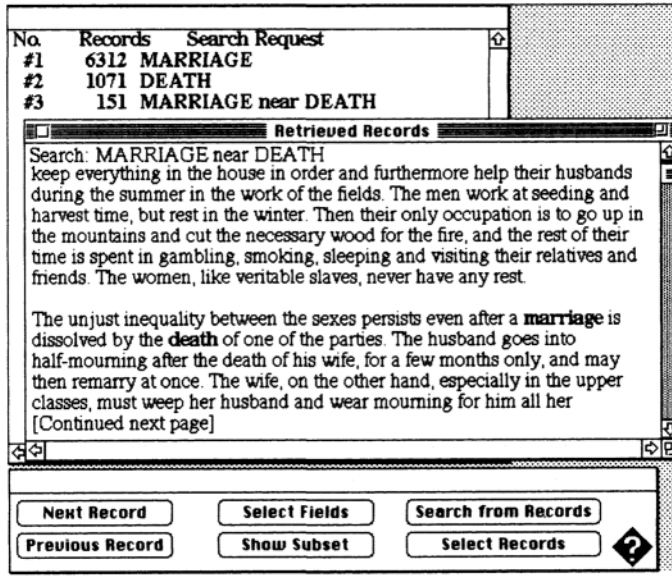


(c)

Figure 2.11 Program for accessing HRAF CD-ROM database on Marriage: (a) database selection from HRAF (two databases per CD-ROM); (b) search terms entered to find relevant sections; (c) results of search term, 'marriage near death'; (d) display of reference from keyword search



(b)



(d)

basically very easy to operate. In general, you put a CD-ROM disc into your CD-ROM player and run a specific computer application intended to access the information on the disc. Many discs are documents for hypertext programs

(Sections 2.3.4 and 4.6.5). By 1989 CD-ROM was already the basis of at least one product of direct relevance to social anthropology, with the launch of a series of discs produced by Human Relations Areas Files (HRAF), which includes pages from over 1,000 ethnographic sources relevant to topics such as aging, marriage, sexual practices and child socialization. These pages more or less represent what one would receive from a request on one of these topics from the ethnographic source pages of the paper-based HRAF files, although the references tend to be older ones (prior to 1950), presumably to limit copyright liability. Using a computer program similar to a plain-text database (Section 4.6.3) or hypertext editor, the contents of each disc can be searched for sections which match keywords, and the corresponding sections retrieved (Figure 2.11). This results in a formidable educational and research tool for comparing directly the ethnographic descriptions of specific topics across the sample of cultures covered by HRAF. At the time of writing the price for each of the discs is rather high for the amount of information included, about £900 per disc. Each disc contains about 10,000 pages of text and a detailed index, and no illustrations or plates, about 8 per cent of the capacity of the disc.

CD-ROM is the direct descendant of an earlier technology, Videodisc™, also pioneered by the Dutch company Philips Electronics in the early 1970s. Videodisc technology, which was used to record high-quality video and sound, fell victim to a combination of technical problems and competition from videotape recorders/players. The technology continued in small niche markets, mostly in interactive training productions and interactive video games due to the ability to quickly retrieve specific sequences of the recording in any order. This facility, in conjunction with computer control, made possible applications where the continuation of the video programme was contingent on the action or choice of the viewer. For example, a training film could continue either with successful resolution or disaster, depending on decisions made by the trainee.

Random access to the frames on the videodisc was exploited by A.Macfarlane (1990) to produce the Naga videodisc set. Although he originally intended to put all of the available material on the Naga on a single videodisc, owing to a combination of technical constraints and an under-estimate of the amount of material available, he completed a set of three discs containing about one-third of the available material. The disc contains video sequences, film, still photographs, sound recordings and textual data. The front-end program for the computer accepts keywords from the user and retrieves matching material from the videodisc. Thus a wide range of multimedia data can be retrieved for research and educational purposes.

There are a number of similar projects underway at this time, which should be appearing over the next few years. Some of these have much the same objective as the Naga project, but using the more widespread CD-ROM technology. Others are aimed more at relating standard ethnographic work, such as a project with which I am associated which will include the fieldnotes, texts, maps, photos and longitudinal field surveys from over forty years research by Professor P.Stirling on two Turkish villages.

All the new CD-ROM-based ethnographic projects of which I am aware are based on a hypertext program called Hypercard, which has particular advantages for academic authors wanting to create their own CD-ROM hypertext documents,

since once the work of assembling the ethnographic document is completed, the author can send the hard-disk drive containing the document(s) to a CD-ROM manufacturer for pressing with no further preparation or the need to write a special program to access their data, and the program (or a clone) for executing the document is widely available and distributed for most microcomputers (see also Section 2.6).

In CD-ROM technology the information on the disc is simply that; a very large amount of information on a disc. The information on the disc is organized into files, much like other storage technology. How the information is used is determined by computer programs installed on individual computers, not on the disc or installed in the player (which is itself a computer). There is, at present, little standardization of how these programs operate, although discs produced by individual publishers often work in a similar fashion.

A new development in this area, drawing both from the multimedia aspects of videodisc and the data orientation of CD-ROM, is the CD-I technology, also under development by Phillips Electronics. Using the same disc as CD-ROM, the principal advance is storing the means of access and programming on the disc itself, to be stored and executed by the player, accepting input from the user either directly on the player console or indirectly via another computer. This, in effect, establishes a standard for how information is stored and accessed from the disc, as well as a standard for how the user can access the data.

Another move to standardization, principally for the text-based CD-ROM, is being promoted by Sony Corporation. They have proposed a standard for electronic books, using the smaller 3 inch CD (holding a mere 300 megabytes, or about 100,000 pages of text). In 1991 they introduced a small portable product called Dataman™, to access these discs and the introduction of this machine was followed by the introduction of a number of electronic books for the machine, mainly encyclopedia, dictionaries (uni- and multi-lingual), directories and other reference works.

2.6
DATA ARCHIVES

As more and more projects in anthropology use computers to analyse data, potentially one recurring problem in anthropology may be alleviated. Traditionally, anthropologists have maintained their data manually in the form of rough notes, interview transcripts etc., and have published only analyses of these with a limited amount of data for illustration. Most data is collected by individual anthropologists, and is only available to others in a highly processed form. There are some very good reasons for this practice: as the research data includes information about specific individuals it is important not to release the raw data. Indeed in many countries one has a legal obligation not to release such information. From a scientific perspective this is not a satisfactory state of affairs, but anthropologists have few resources for preparing data for release to others. With the material on computer media many of these problems are lessened. It is relatively easy to create copies of specific sections of the data, removing information which can be used to identify individuals, and in some extreme and

sensitive cases, applying specific transformations of the data to alter it to disguise personal and locational references. The latter may seem extreme, but human livelihoods or even lives can literally depend on such a disguise.

As computer-based data resources became available, disciplinary initiatives were set up to collect and distribute them. For example, the Economic and Social Research Council of the United Kingdom maintains a quantitative data archive for the social sciences, and is considering one for qualitative data. Some older data archive centres which existed before computing became economical have converted to computer archives. HRAF began in the 1950s as a manually maintained archive of ethnographic information, based on a coding scheme and index by G.P.Murdock. For some years the coded data has been available on computer media. The 'raw' data have recently begun being transferred (pages from about 1,000 ethnographic works) to CD-ROM (Section 2.5). There are currently several projects in anthropology to set up specialized data archives for visual data.

Recent developments in computer communications infrastructure make these archives especially relevant. Besides inexpensive modems, which can be used to connect two computers any where at the end of a phone line, there are now several world-wide networks which connect most of the universities in wealthier countries together, as well as commercial links into these networks. For example, I can now make a local phone call in Islamabad and log on to my computer in Canterbury. From my desk in Canterbury I can access a data archive in Australia, examine files at that location and transfer these back to my computer. There are now specialized applications which can access network-wide indexes of available material, give you a list corresponding to a keyword request you have made and fetch those files you want. There is enormous potential for anthropologists to share data, once the barriers to this practice erode within the discipline. You can learn more about this from your local computer service.

At another level, communications technology opens up the potential for collaborative research involving researchers at sites around the world, where messages can be exchanged not only via electronic mail (E-Mail), but by file transfer as well using standardized file transfer protocols (FTP) available via BITNET and Internet, among others. Nor need anthropologists rely on the large institutional resources. A computer connected to a phone can be accessed by any other computer connected to a phone, and there are a number of programs which can automate exchange of files and mail through this 'roll your own' network. User groups for most microcomputers can be very helpful with setting up an *ad hoc* communications network. For further information see Dow (1992:278–9) and Khanna (1988).

Chapter 3
Fieldwork and ethnographic research
In the field

3.1
INTRODUCTION

Fieldwork is the basis of ethnographic research. If computers are to become an important tool for anthropological research, the use of the computer must begin during fieldwork. Otherwise, as Davis suggests, 'striking savings in time and accuracy are achieved only after a high initial cost in preparation' (Davis 1984b: 308). So long as data is recorded in one form, only to be transcribed to another, then preparing field data for computers competes with other tasks such as indexing notes, transcribing audio tapes and deciphering genealogies, and can potentially be the most time consuming of these tasks. But, '[the costs of preparation] diminish as you plan and as you use the machine for more jobs' (ibid.: 309). The larger tasks might be the initial incentive for computer use, but the benefits emerge from the multitude of smaller jobs which would not themselves have justified the cost in time and money. One of the ways to diminish the barrier of preparation is to use computers throughout the research process.

Although 'striking savings in time and accuracy' are important, these are not the most compelling arguments for using computers in anthropological research, in the field or out, as Kippen remarks:

> I believe that the enormous potential of computers to improve the capacity of anthropologists to gather, store, and analyse information has yet to be demonstrated. Clearly, it is necessary to look beyond wordprocessors and databases to other systems, such as knowledge-based systems, that have the power to change the ways in which we as researchers operate.
>
> (1988b: 318)

Our most important goal in using computers as a tool for research must be to do better anthropology and not simply more. If you are satisfied with the state of ethnographic research, there is little purpose for introducing the additional cost and time for learning. The best way to introduce computing into our research is to first replicate what we have done before, but greater benefits will come when computers are used to do things we could not do before, not only for the

amount of time these would have involved but because these could not easily have been conceptualized prior to the opportunities the computer as a tool can facilitate.

3.1.1 Ethnographic data—how conventional?

In Section 3.2.1 I have adapted a schema Ellen presents for classifying 'permanent' written field records (1984b: 282). I take his account as representative of the way anthropologists present field records and their guidelines for producing and maintaining these records, a view in concurrence with Sanjek (1990a: 235). But, in practice, just how conventional is this schema? For example, consider the category Ellen refers to as 'conventional notebooks' (1984b: 282), which I take to be, more or less, fieldnotes (although Ellen seems careful not to do so).

Jackson (1990) conducted systematic interviews of a non-random sample of seventy users of fieldnotes, sixty-three anthropologists and seven other social scientists. She encountered considerable variation in what are regarded as fieldnotes (at least for her sample—anthropologists on the East Coast of the United States). She reported that

> What respondents consider to be fieldnotes varies greatly. Some will include notes taken on readings or photocopied archival material; one person even showed me a fieldnote in the form of a ceramic dish for roasting sausages. Some give local assistants blank notebooks and ask them to keep fieldnotes. Others' far more narrow definitions exclude even the transcripts of taped interviews or field diaries…Clearly, what a 'fieldnote' is precisely is not part of our profession's culture, although many respondents seem to believe it is.
>
> (1990:6)

In her fuller discussion it becomes clear that there are two main sources of variation. First, precisely where, in the classification of field material, does any particular kind of information belong. This is a matter of naming rather than of substance, possibly broadening our inclusive definition to the entire range of written records and beyond (and perhaps accounting for Ellen's caution). The second source of variation relates to whether the respondent placed value on or even recorded fieldnotes, entering entirely into the domain of what Davis refers to as 'Intuitive knowledge' (1984b: 304), Ottenberg as 'Headnotes' (1990: 144) and Jackson, quoting one respondent, as 'I am a fieldnote' (1990: 21). Jackson also describes a range of 'personal' attachments and a mystique which many of her respondents ascribe to fieldnotes (ibid.).

Seidel (1991), author of *The Ethnograph* (1988), a popular program which supports the analysis of ethnographic texts, discusses conflicts he has reconciling promoting the use of computers for ethnographic research while being aware that his program is used by some researchers as a tool to do ethnographic analysis, rather than a tool to assist ethnographic analysis. They become 'distanced'

from their data by defining ethnographic analysis as those tasks the program performs (1991: 114–15) or taking refuge in the 'deification' of data (ibid.: 112–14).

The writing and use of ethnographic field material can apparently go astray with or without the involvement of computers. Using computers cannot solve the problem of patchy notes (although they can be used to assess how patchy the notes are (Fischer and Finkelstein 1991)) and are unlikely to assist those who depend entirely on their memory (but see Section 8.2). For some people computerized notes may not be able to supply the mystique of notebooks and they may not be able to develop personal attachments to computer data files. And it is possible that a program designed to enhance access to fieldnotes might provide an avenue for abuse.

It would be unwise to dismiss these problems. A major consequence of using computers for field records might be to make us aware of these peripheral aspects of producing and using these records. One of the things that applied anthropologists are forever telling organizations is that regardless of what their planners, accountants and managers say the structure of the organization is supposed to be, it is better to base policy on the organization as it operates rather than how it is supposed to operate, translating the results back into organization-ese if required by company reports. The failure of many a computer system situated in an organization has been due to the inability of that system to cope with actual practice.

But is anthropology and the conduct of anthropological research a fiction? There are many critics who appear to tell us so, although their criticism and our consciousness of that criticism tends to reduce the fictional danger over time. We must not lose sight of an important disciplinary assumption. We may take it for granted that the ethnographer and the ethnographic texts are intertwined to the extent that others may not make much out of the texts without the ethnographer. We may disagree about what is being unveiled. But we must also assume that the resources the ethnographer uses in the retelling of the material meet some general standard. If Ellen's presentation is indeed a fiction, we shall hope it is a useful one.

3.1.2 Organization of the chapter

This chapter presumes that a computer will be available in the field, where issues differ somewhat from the study. If field access to a computer is not possible (or the field research has already been done), then, of course, both preparation and analysis can take place at your home site. In any case, it is important, in advance when possible, to match intended forms of processing data with plans for the collection of that data. In ethnographic research this is not always possible, nor in all cases desirable, but certain methods of analysis are simply not possible without specific data.

It is certainly not my purpose to say what you must collect or how you must analyse it. (This would be pointless because I cannot anticipate all my own research needs.) Any specific methods given are illustrative rather than

pre scriptive. In a research environment methods must come from the researcher. My intent is to inform, to identify what resources are available to work with conven- tional categories of data and to suggest how some lesser-used categories of data might become more important. Unlike archaeology, linguistic anthropology and physical anthropology, the literature for social anthropology is rather sparse in applications of computers to data other than fieldnotes and other texts, survey data and historical documents. However, all means discussed are accessible, and have been used in the field by anthropologists, although some require more expertise than others.

3.1.3
Past and present feasiblilty of field computing

Early efforts by Weinberg and Weinberg (1972) and by Brown and Werner (1974) were attempted by shuttling information in and out of the field site for processing, but this was, understandably, not persuasive. The rapid development of microcomputers from the late 1970s—increases in power in conjunction with reductions in size and cost—has made the use of computers in the field practical, and increasing numbers of anthropologists have taken computers into the field since the late 1970s (Tomajczyk 1985; Sutton 1984; Powlesland 1986; Guillet 1985; Dyson-Hudson and Dyson-Hudson 1986; Ellen and Fischer 1987; Agar 1983; Case 1984; Werner 1982).

These efforts were not without the drawing of blood. Many of these systems weighed over 50 pounds, and upwards of 100 with accessories, power supplies and spares. Equipment designed to operate in laboratories or homes at temperatures between 15°C to 27°C with relative humidity between 40 per cent and 80 per cent (non-precipitating) found itself in temperatures from –10°C to 50°C and relative humidity of 100 per cent (precipitating), with wily anthropologists luring insects from the disk drive with peanut butter (Dow 1987). No one who has participated has suggested it was not worth the trouble, but that it was trouble no one would deny.

Thus, for most anthropologists taking a computer to the field has become feasible only recently, not because it was technically impossible previously, but because the logistical difficulty of operating or accessing computers in most field situations was unacceptable to most anthropologists.

Most of the problems with hardware are, if not solved, under control. There are battery-powered ‘notebook’ computers the size of an A4 book, weighing less than 3 kilograms and capable of almost anything larger ‘desktop’ computers can do. A4 size ‘notepad’ computers weighing somewhat less than 2 kilograms, can perform complex tasks and store a year’s fieldnotes, as can their smaller brethren, the ‘palmtop’ computers, weighing less than 350 grams, having 256k–512k RAM, being powered for six weeks on two AA-sized batteries and having a non-volatile storage capacity equivalent to 1,000 A4 pages on a removable (and replaceable) 20 gram wafer solid-state ‘disk’. Interfaces for direct entry via hand printing with a pen are available, which work much like conventional paper pads, down to the ability to draw in the margins. Although initially more trouble, small hand-held machines with a one-handed ‘chord’ keyboard can be used for

note entry, with blind text entry at speeds at least as fast as conventional typing and with the ability to off-load the text to a larger computer (say a palmtop) for analysis. In short, an anthropologist can carry into the field more computing power and storage in a 350 gram package the size of a Prince Albert tobacco can than was available in 1980 using a desktop microcomputer.

3.2 COMPUTING ASPECTS OF WRITTEN DATA PRODUCED DURING FIELDWORK

Most anthropologists will find written records are the easiest to work with using conventional computing techniques, at least in the sense of the amount of knowledge about computers required for the job. Almost any computer, of any size, can be of some benefit to this process. In this section we shall look briefly at the requirements of ethnography and how these can be reconciled with computer-based tools appropriate to working with written materials.

Ideally, in doing ethnography we would not be restricted in any way in terms of the information we record and our subsequent access to this information. In practice the very tools which are used (the means of writing and organizing this writing) appear to encumber the process, not only because of the time required to create a record of our observations and experiences, but also because the volume of writing becomes so great that it is difficult to keep an overall grasp of what has been recorded, much less the detail.

Part of this problem is our human inability to remember in a non-selective manner (D'Andrade 1973) and part is technology, the pen, paper and writing we use to improve our ability to remember. You can use computer applications to address the technological problems as a supplement to, or as a partial or total replacement for, our conventional technology.

In this section we examine the issues of data planning, collection, maintenance and minimal analysis in the field. Some processing and analytic issues are discussed in Chapter 2 and fieldnotes are discussed in greater detail in Chapter 4. It is unlikely that most anthropologists will want or need to use all of these methods in any one field project. This is a discussion of computing methods available for, and appropriate to, the field, rather than what one should do in the field. In each of the entries basic issues, the kinds of software necessary to address these issues and, where necessary, an outline of the hardware requirements as these appear in 1993 are presented. Guidelines are intended as just that, and it is essential that you seek specific advice from a computing support person or through further reading. Software issues are, in part, based on conceptual and practical issues relating to the problems addressed. Hardware is simply a means of implementation, and is subject to rapid changes in capability and price. I have tried to limit speculation to hardware developments that are more or less inevitable in a time span of two to three years.[2]

3.2.1
Types of written records produced during fieldwork

In terms of 'conventional' ethnographic research, most data,[3] and the most important data, is collected in the form of written records. Ellen identifies seven main types of written records produced during fieldwork (Table 3.1). These forms of data, while not exhaustive, serve as a basis for discussing computer applications in the field, which must necessarily be oriented towards the overall fieldwork process.

In general, computers adapt very well to assisting in the collection and analysis of written materials, in part because of the activeness of humanities researchers in devising methods of working with historical and literary texts, and in part because of the commercial 'office revolution' in the 1980s which resulted in the widespread distribution of quality computer-based tools for producing and accessing texts, as well as the provision of tools for representing complex information and facilities for making reports based on that information. Some of these tools can be applied directly to our material, with little accommodation on our part, some require that we structure data in ways which are otherwise unnecessary and some require an entirely different conceptual basis to be used effectively.

3.2.2
Temporary notes

Temporary notes are, currently, the least likely data to be entered onto computer in a form other than their contribution to the other types of written records. Most temporary notes are written into notebooks and, being temporary, there is little point for re-entry just for the sake of reproducing the notes. Some researchers, myself included, enter the temporary notes into a wordprocessor verbatim and then expand these, but do not systematically retain a verbatim computer copy (cf. Pfaffenberger 1988: 33).

You can, of course, enter the temporary notes directly into a computer. Many ethnographers find taking notes on paper distracting to both themselves and their consultants. It is unlikely that a computer will be less so. In many cases there is so little 'connected' text (as opposed to telegraphic prose, outlines, charts and drawings) involved in a day's temporary notes that there may be little value added if the presence of a computer is likely to detract at all. In my own fieldwork I have recently been taking some notes on a small hand-held computer which is as unobtrusive as a pen and notebook (once I have initially demonstrated it to my consultant). It saves no time whatsoever if I transfer these notes, unaltered, to my larger laptop computer later in the day, because the time required to set up the file transfer hardware and software takes about as long as typing the notes in (although with far less tedium). The major advantage is that I can partially fill the notes out on the hand-held computer at odd times during the day, wherever I might be, which does save time, since otherwise I would be writing the expansion in the notebook and retyping the expanded version later in the day. Also useful, I have all previous fieldnotes and card indexes (from current

Table 3.1 Types of written record produced during fieldwork

Temporary notes
Conventional notebooks, general and specialized according to topic or kind of record
Diaries, personal and register of events
Card indexes, such as a name directory or dictionary entries
Questionnaire returns
Special records, e.g. register of taped material, special survey material, genealogical charts
Maps and diagrams

Source: adapted from Ellen 1984b: 282

and previous research at the same site) on the hand-held computer as reference for writing the expanded notes, as well as for preparing visits and interviews.

Two kinds of specialized hand-held computers may better justify the direct entry of notes onto computer compatible equipment. The first are small, relatively inexpensive 'chord' keyboard entry computers, although I know of no anthropologists using these. These weigh only 100–200 grams and are operated using one hand and a small number of keys. Special key patterns have to be learned which correspond to letters. Some people have little trouble with this, achieving relatively high rates of speed in a few weeks. Others find it very difficult. For successful users, notes can be entered faster (40–80 wpm) than most people can write. More important, notes can be taken in any position, even with the machine in a pocket, matching Margaret Mead's alleged ability to write notes in her handbag. Once in the machine the notes can be edited and read directly from the display, or transferred to a larger machine for expansion into the day's fieldnotes.

The second development is of small, powerful and relatively expensive hand-held computers which accept input in the form of hand printing with a pen. As of 1993, these are too error prone to replace pen and notebook in ethnographic situations if scrawl is most likely to describe the standards of penmanship. However, if you can print fairly neatly, the better of these can record copy which is good enough for rough notes. Indeed the best of these pen-driven systems have abilities of contextual linking to contents of other notes and databases which suggests that, properly prepared, such a system marks some progress towards a process of automatically filling the notes out with personal details, location details and time reference information, as well as automatically posting reminders to follow up issues so marked in the notes. These systems also support the drawing of diagrams and the linking of these to textual and database references. With more development these pen-driven hand-held systems have the potential to serve as the complete hardware base for field-based anthropologists.

3.2.3
Conventional notebooks, diaries and interview transcripts

Diaries and interview transcripts etc. are suitable for computer-based entry in the field, using standard wordprocessors (Section 1.3.1.1) and hypertext editors for entry (Section 2.3.4 and 4.6.5) and/or simple textual database managers (Section 4.6.3) which are easily adapted by a casual user for these purposes (Section 2.3). They are also the data from which much post-fieldwork benefit will come, since the preparation time will be considerably lessened for data which is among the more tedious and which requires the most sensitivity and specialization of (anthropological) knowledge to enter;[4] information which should usually be entered by yourself on return. See Chapter 4 for a review and discussion of principles and methods of fieldnote entry, management and analysis.

3.2.4
Card indexes

Card indexes are a common cellulose-based information technology (CBIT) for storing and accessing a variety of kinds of data in the field, including word lists, bits of information on individuals and names of objects, plants and animals. Among the advantages of card indexes in the field are flexibility in inserting new material and the option of sorting the cards into different orders for different purposes. Card indexes are particularly easy to implement using conventional computer programs. The type of program you should use depends on the kind of information and how it is structured on the card equivalent.

Notebook computers are capable of dealing with any of the options given in the following discussion. If your field situation requires very low power use, notebook computers capable of operating for several weeks on standard batteries can be used for most of these options, if on a smaller scale, but probably adequate for a season's fieldwork.

To simply replicate the function and operation of a manual card index on a computer is very simple; you can enter the information using a basic text editor or wordprocessor (Section 1.3.1.1) and use the 'Find' operation of the editor to locate sections of the document by doing a word search and, if desired, copy selected material into another document. This has the advantage of using a basic tool for two purposes, entry and retrieval, and permitting you to lay the information out in any way that you like which is compatible with the wordprocessor or text editor. Given that the more recent wordprocessors have the capability to store and display high-quality images, and even video and sound (Sections 3.3.2 and 5.3), this can be a flexible structure indeed. However, there are severe constraints if your document becomes too large (and thus slow) or if you want to process the information further.

If you want to automate storing the results of a word search, most text-based computers have at least one program which will accept textual keyword(s) and search through your text-file created using a text editor, displaying on the screen or printer or storing in another text file, the lines in the source file which contain the keyword(s) (Table 3.2). If your card index, word lists or inventories

are simple in structure, this may be adequate. If a single line is not adequate to represent the contents of your card index application, there are other programs which will work with larger sections, separated by some distinctive marker you type in after each entry. If you have stored the output of these searches in a new file, other programs can be used to count the number of lines or to sort these files into new orders.

For more complex data-processing requirements see Section 2.3. In particular, for fieldwork purposes consult Sections 2.3.4 and 4.6.5, which describe uses of hypertext editors. The principal advantage of hypertext for fieldwork is the extreme flexibility for representing different kinds of textual and non-textual data, together with flexibility in changing the design as your work progresses.

3.2.5
Questionnaire and survey returns

Software for the analysis of survey data is perhaps the best known application of computers to social science research, for many the only known application. There are a broad range of applications for statistical analysis, questionnaire tabulation and generating tables for examination or publication. The kinds of program you require depends on the complexity of the survey instrument and the size of your intended sample. For surveys with mainly numerical data and where the number of cases is a few hundred, a spreadsheet calculator may be adequate (Section 2.4.1). For more complex surveys with linked or optional questions, a statistics program will probably be required. If the survey is especially complex, you may in addition need some kind of database management system (Section 2.3) which you can use to create simpler data structures for the spreadsheet or statistics program.

The value of dealing in the field with questionnaire and survey returns depends on the use you intend for these while in the field, the sensitivity of their

Table 3.2 Results of two keyword searches of a text file representing an Urdu/Panjabi word list

Input File (Urdu/Panjabi)	*Search for*	*Output*
admi, man	*boy*	bucha, boy
aurat, woman		lerka, boy
bucha, boy		
buchi, girl	*lerk*	lerka, boy
...		lerki, girl
lerka, boy		
lerki, girl		
log, people		

content and the means you have available to enter these into a computer. A problem, both in the field and out, is the preparation of survey data for the computer. If you intend to simply look over the questionnaires in the field, and you must enter these yourself from the paper forms, then it may not be a justifiable expenditure of valuable field time. If you think a particular questionnaire might be useful for planning further research while in the field, then it may be worth the time, if you have included an appropriate spreadsheet or statistical package in your software tool kit. In many cases for planning field research an analysis of a relatively small sample (less than fifty) of the questionnaires or survey instruments will probably be adequate.

In some cases it may be possible to have someone else enter the questionnaires into your computer, or another computer which can produce compatible files. For example, in many countries banks and public utilities have spare data entry capacity and will enter the material for you at a reasonable rate. If the questionnaire is a long one or covers more than fifty cases, it is a good idea to pay for it to be entered twice and use a program which compares two files to identify errors (which are likely, if not certain), or pay for this service as well (which is always available if they take in outside work).

There are also various pieces of equipment which can be used to 'automatically' enter forms from surveys which require mainly yes/no or multiple-choice responses, using some kind of mark-sense form most commonly manifest as a computer card similar to the old punched card. The response is recorded on a special form, marking with a pencil in a particular position, and directly read with a machine. This is an expensive but simple solution. You must get such a reader with software and hardware interfaces which suit your machine, test it for accuracy before leaving for the field and take along a suitable supply of special forms with you. Another option which can be used with forms on which choices are recorded on plain paper are optical character readers which can read carefully hand-printed numbers, or if you know a programmer, a program can be written which will spatially interpret the marks on a form which has been entered using an image scanner/digitizer (Section 5.5).

If you (or your field assistants) are filling in the questionnaires in the course of interviews, direct entry of the questionnaires into small notebook or hand-held 'palmtop' computers can be a productive option. A form representing the questionairre can be 'programmed' into the small computer and entries made and recorded quickly.

Green (1988) developed a program based on an expert system (Section 8.2) to assist in converting textual material such as interviews into questionnaire type data, by having the anthropologists develop a set of rules for answering the questions from a transcript and encoding these in the program. The program then asks the questions for each case, providing standardized assistance in classifying each case with respect to the question. This has the advantage that each questionnaire is produced in a consistent manner from relatively unstructured data.

3.2.6 Special records

Ellen (1984b: 282) includes in this category materials such as registers of taped material, special survey material and genealogical charts, and most other kinds of written data of a 'special' sort. In the strictly written sense, most of these can be adequately dealt with using either textual databases or more structured database management systems as in Section 2.3, possibly in the case of genealogical data using the programming language Prolog (Chapter 6), and for special surveys which incorporate visual and aural materials a hypertext application (Sections 2.3.4 and 4.6.5).

For many kinds of special records such as these it is sometimes useful to write a dedicated application for entry. This can be done with fairly limited programming skill using a hypertext editor program or most database management systems, or you can specify the function of the application to a programming partner.

As an example, for genealogical material a simple graphical editor can be used for entering relationships directly in the form of kinship diagrams and attaching information about the household members with the kinship symbols on the screen. The program illustrated in Figure 3.1 was written using a common hypertext program (Hypercard) in about two hours, most of which was spent making the icons for the different symbols. It underwent a slight modification after some suggestions in a workshop to improve the control in manipulating the symbols, taking another twenty minutes.

Similarly, the application in Figure 3.2, written using the same hypertext program, is dedicated towards making a register of the contents of audio and video tapes. It simply is a form for entering textual notes and tape index counter readings which can be used as I am listening to the tape, and for adding crossreferences afterward. It is not a general purpose application but it took only about twenty minutes to originally construct, with modifications added as needed while in use.

In both of these applications the data can be later output into more conventional forms if required.

3.2.7 Maps and diagrams

Most anthropologists seem to generate a lot of diagrams or other graphical representations while in the field, if with varying degrees of skill. For some kinds of research diagrams are essential; as a illustration of ethnographic detail (the placement of a decoration in a house), to denote certain kinds of relationships (genealogies) or for creating representations as an interviewing aid (drawings of material culture). See Section 5.2 for further information on drawing using a computer, Sections 3.3.1 and 5.3 for computer-based photographic and video images and Section 5.4 for further discussion of maps.

3.2.7.1
Applicability of diagrams to fieldwork

Although the use of computers for making diagrams is of great value, especially in the presentation of ethnographic material, its use in the field very much relates to the role of drawings in your own fieldwork.

If rough diagrams begin life on paper, there may be little point in converting these to rough diagrams in the computer, except for the possible benefit of integration into your fieldnote database.You might, of course, want to catalogue references to these using a database. If you restrict yourself to rough diagrams, however, the effort is not very great. If you include an image scanner/digitizer (Section 5.5) in your hardware tool kit, the effort is rather less since you are not drawing but copying a drawing onto the computer. The latter will usually be edited using *paint* type programs (Section 5.2.2), although there are some applications which will attempt to convert these for *draw* type programs (Section 5.2.1). Quality problems in the field are of rather less importance than for published work.

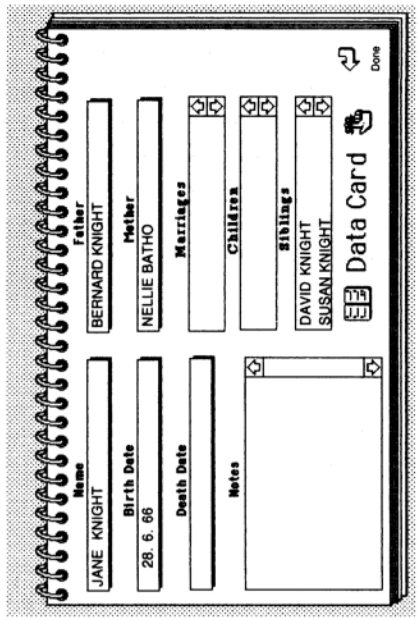
Even with special software and hardware for this purpose, it takes considerable time and effort to prepare a good diagram for publication or lectures, and of the many rough diagrams you are likely to make in the field, only a few will be selected for this purpose. High-quality drawings should be done after leaving the field, unless you have a particular purpose in mind.

I use a limited number of drawings as interview aids when discussing material culture, and find this very useful. From a printed copy I elicit ‘corrections’ to my drawings, helping me to separate etic detail from emic detail, and attach the various responses to the computer-based drawings (in a hypertext program, Section 3.2.6), using these as one means of access. It works well for me, especially given my limited skill in freehand drawing with pen and paper.

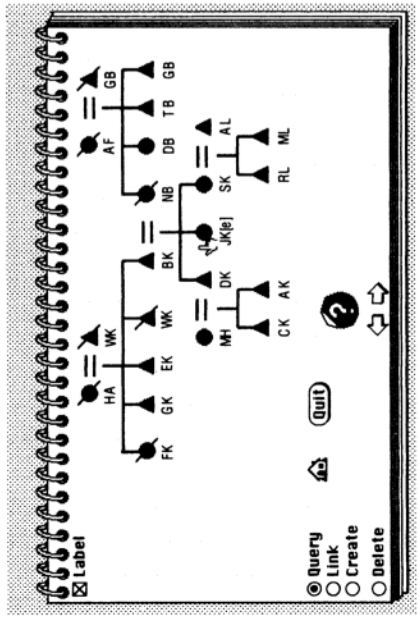
3.2.7.2
Special purpose graphic tools

There are an increasing number of programs which employ graphical tools potentially useful in fieldwork. These include programs for drawing and manipulating maps, programs for direct genealogical data entry in the form of diagrams, such as that described in Section 3.2.6, programs which can assist in the building and testing of taxonomies, based on recent work on cladistics, and graphic editors for the direct input of musical notation, as well as facilities for playing back music in this form.

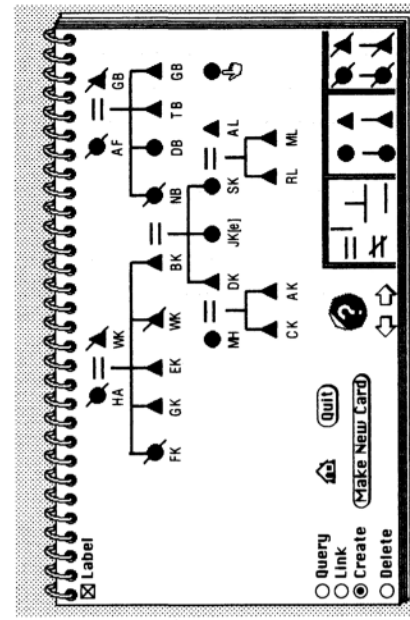
Maps, even rough ones, have considerable utility when on a computer. Drawing a good original map is difficult, even on a computer. But the sorts of rough maps (in appearance, not necessarily in accuracy) that are commonly used in the field are relatively easily entered (see Section 5.4). Use of maps in the field include monitoring cultivated areas, plotting social groups and kinship links, or other cultural variables, against the map and tracking the coverage of surveys and questionnaires. Using a printer, the map can be reproduced in a variety of sizes, with different textual legends as required. Maps are discussed in more detail in Section 5.4.


Name
JANE KNIGHT
Birth Date
28. 6. 66
Death Date
Notes
Father
BERNARD KNIGHT
Mother
NELLIE BATHO
Marriages
Children
Siblings
DAVID KNIGHT
SUSAN KNIGHT
Data Card
Done

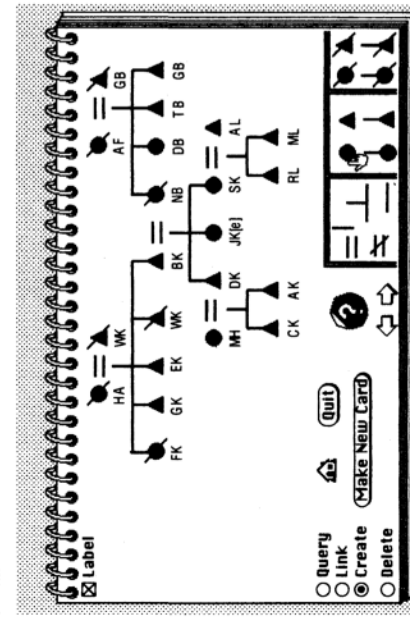
(b)


Label
Query
Link
Create
Delete
Quit

(a)


Label
Query
Link
Create
Delete
Quit
Make New Card

(d)


Label
Query
Link
Create
Delete
Quit
Make New Card

(c)

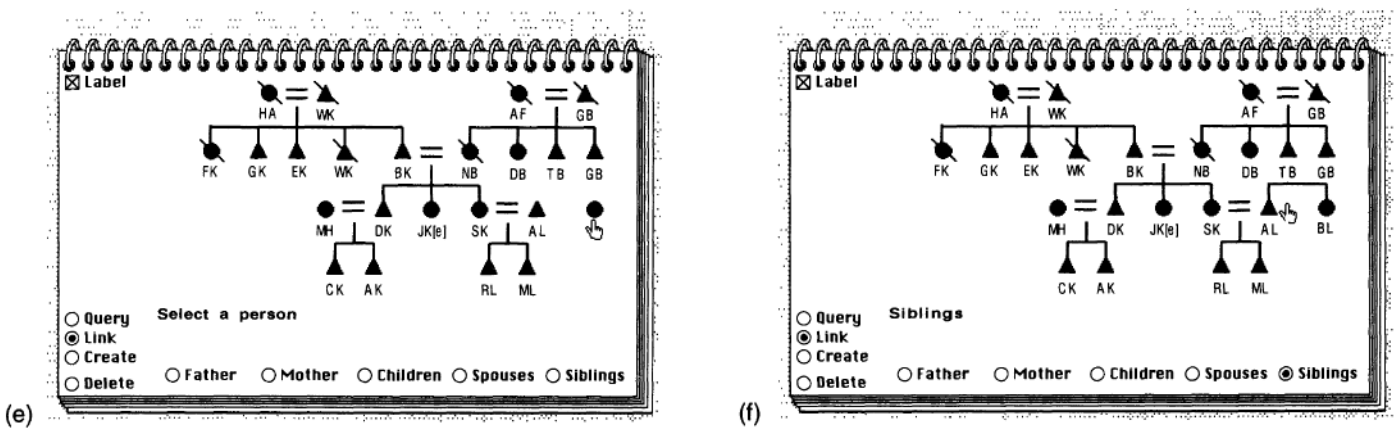
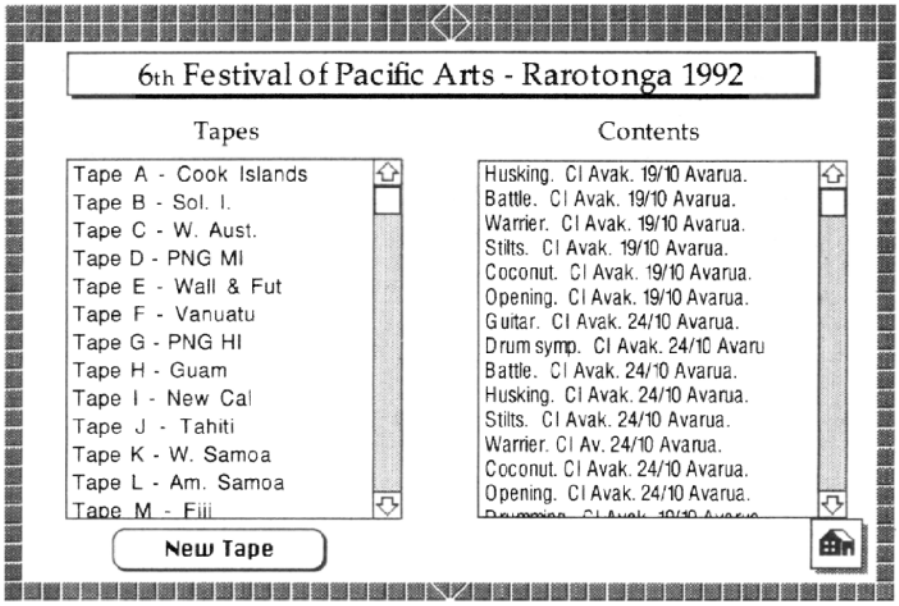
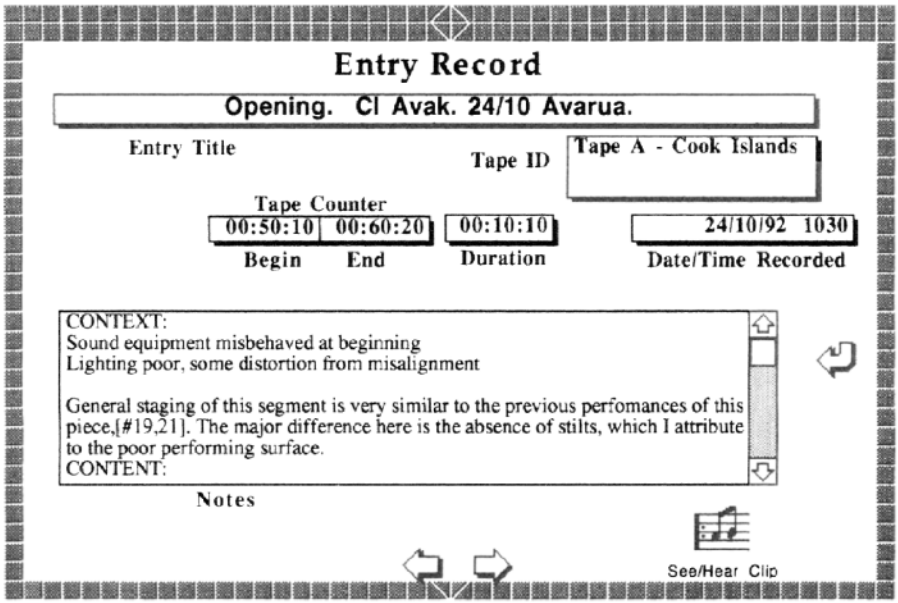


Figure 3.1 Simple hypertext program for kinship data entry: (a) selecting 'JK' for database query; (b) report for person query; (c) selecting icon to create new person entry; (d) placing new person entry; (e) preparing to link new person entry; (f) making a 'Sibling' link

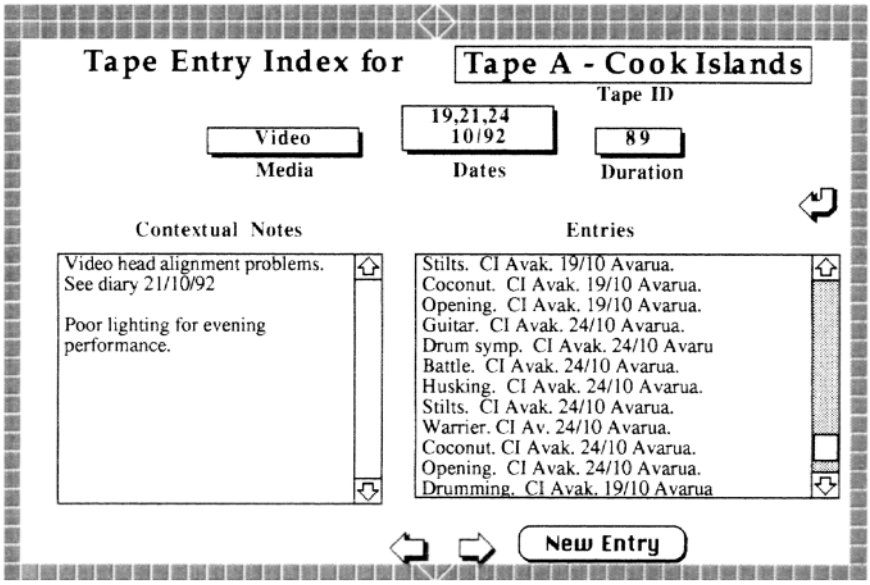


(a)



(c)

Figure 3.2 Simple hypertext program for videotape register: (a) main index for videotape register; (b) tape index entry; (c) entry record; (d) video/sound clip


Tape Entry Index for
Tape A - Cook Islands
Tape ID
Video
Media
19,21,24
10/92
Dates
89
Duration
Contextual Notes
Video head alignment problems.
See diary 21/10/92
Poor lighting for evening
performance.
Entries
Stilts. CI Avak. 19/10 Avarua.
Coconut. CI Avak. 19/10 Avarua.
Opening. CI Avak. 19/10 Avarua.
Guitar. CI Avak. 24/10 Avarua.
Drum symp. CI Avak. 24/10 Avaru
Battle. CI Avak. 24/10 Avarua.
Husking. CI Avak. 24/10 Avarua.
Stilts. CI Avak. 24/10 Avarua.
Warrier. CI Av. 24/10 Avarua.
Coconut. CI Avak. 24/10 Avarua.
Opening. CI Avak. 24/10 Avarua.
Drumming. CI Avak. 19/10 Avarua
New Entry

(b)


Clip
Opening. CI Avak. 24/10 Avarua.

(d)

The value of graphical input of data in the field depends very much on your research. The strength is that the data input format (and thus at least one database record) is the same as the report or output format. If you tend to record genealogical relationships in diagrams in the field, then a genealogical editor may well make sense as a means of transferring data from notebook to computer, even more so if you intend direct entry working with a consultant. The graphical structure of the data represents structure as well, and this can help ensure that you have included everyone and recorded all the relevant data. On the other hand, if you mainly record genealogical relationships in textual form (recommended in any case if you are recording genealogies of more than twenty or so people at a time), and want to be able to see genealogical relationships between different groups and dwellings in a village or several villages, then the methods in Chapter 6 will probably serve as well, or better, since genealogical diagrams of too great a scale tend to obscure relationships rather than bring them out. Certainly, as with any tools you intend to take to the field, you should learn and evaluate these well before depending upon them in a 'live' situation.

3.3 OTHER TYPES OF DATA PRODUCED DURING FIELDWORK

In addition to the written data in Section 3.2, there are other common kinds of field data:

1 Photographs, cine film, video;
2 Audio recordings;
3 Material culture; artefacts, measurements.

These are briefly discussed in this section, and most have more extensive coverage in Chapter 5. What most of these have in common is that, for most anthropologists, they are at present somewhat peripheral activities, if well established (Blacking 1984: 199). Even in the case of photographs and to some extent audio recordings, which most anthropologists make, these are under-used for research, either in the field or out. On the other hand, most anthropologists I have spoken to are interested in using these materials but have found them cumbersome in practice, so the material mainly sits in drawers.

Computers are available with the power, peripherals and software to address information of all sorts, not simply numbers, words, formulae and coded texts. This approach to computing is just beginning in the sense of being practical for ordinary users. In anthropology there is but one completed work (Macfarlane 1990) which incorporates computer accessible images and ethnographic texts as an ethnographic report, but there are a number of research projects in progress and proposed research under consideration. These initial projects are drawn mostly from researchers with a pre-existing interest in visual anthropology or folklore, but I believe that the application of computer technology is ultimately likely to radically change both the practise of fieldwork, ethnographic research and ethnographic reporting.

3.3.1
Visual recordings: photographs, cine film and video recordings

Anthropologists incorporate images into their research to a widely varying degree. Some take a few photographs simply for the purpose of illustrating some aspects of their notes and other data. Others use visual records as a central part of their research. Those in the former category may find little use for a computer other than perhaps the creation of a special register or card index of photographs and legends, and giving their photographs names and referring to these in the notes or other data resources.

If visual records are an important aspect of your research, there are many more possibilities. The technical details of when, where and how to acquire images in the form of photographs or video and the basic methods of analysis and use are no different from more conventional media for representation. These issues are well covered in Collier and Collier (1986) and Jackson (1987), as well as briefly discussed by Blacking (1984). However, the range of analysis possible and the ease of access are more extensive. These range from text/code-based computer-assisted classification using authority lists (taxonomic thesaurus of classification terms for subject domains), to cross-references between fieldnotes and people, places and events depicted in the records, to the incorporation of the images into a computer representation which can not only be accessed and viewed on screen from databases but can serve as a interactive element for data entry relevant to the images and objects in the images. Images can be incorporated into most computer applications which operate on graphical operating systems, including wordprocessors and database programs. Images can be resized and otherwise manipulated using a paint program capable of editing colour images.

Although it is well-established technology, at present the capability to digitize and incorporate digital images comes at a price, in both equipment and weight (and money, of course). The computers are not a problem, since any notebook computer with greyscale or colour display can display images with adequate detail for most field use. There are, however, two basic problems with digital still images. First, a method of acquiring and digitizing images is needed. There are a number of devices for digitizing images, almost all of which are larger than one would like for fieldwork. The best all round field solution, if you do not require the best possible quality, is probably to use a video camcorder, with an attachment for slides and negatives if required. This camera, in conjunction with a *video frame-grabber* peripheral, will enable you to digitize stills directly from the camcorder. The frame-grabber should come supplied with software for this purpose. At present there are no frame-grabbing devices intended for portable use, but most can be coaxed to run off a battery by a competent technician. I expect this to be rather less of a problem in the very near future, with this capability built into more expensive notebook computers. You are also likely to require more memory than you would require otherwise. With images of the resolution discussed, 8 megabytes of memory is probably the minimum required.

The advantage of this arrangement is mainly in terms of portability and time investment in the field. Hi-8 or SVHS-C camcorders can be quite small and light, as little as 900 grams plus batteries and easily pocketable. The resulting

image from a digitized frame will fill most computer screens with a full-colour (or greyscale) image rather better in definition than a good home television. You can depend entirely on the camcorder for visual documentation (and sound as well) if the quality is adequate (it will not be any better once you leave the field). You can also take conventional slides, and once developed these can be transferred to the camcorder using a slide-scanning attachment available from all major camcorder manufacturers. This leaves you the option of making better quality stills once out of the field.

The second problem is storing the images on a disk. Even when highly compressed, a full-screen image will occupy about 20,000–100,000 bytes of memory (the equivalent of about 5,000–20,000 words, rather than the proverbial one). This amounts to 20–50 images per megabyte. If you produce about 2,000 images in a short season of fieldwork, this requires about 100–200 megabytes for disk storage. Given that you have other storage requirements, this is rather a lot, but is an available option for most full-featured notebook computers.

For videotape, support for partial or even full transcription at frame level is available, either through computer-controlled players or through direct representation on the computer. Indeed, full transcription may not be necessary in some cases because under computer control specific video frames or sequences can be displayed on the monitor as a response to database queries, along with notes and markings added by the researcher.

Because of current costs and limitations, using digitized video in the field has to be viewed in terms of application. For most researchers the principal application of displaying or making video in the field include the following.

1 *An interviewing aid.* This suggests that you have to plan your requirements based on a system which you can either get your consultants to, or take to your consultants. If you can prepare the materials in advance this may require only the capacity to display video. Software required can be as simple as a wordprocessor, although special purpose applications exist for this purpose (Section 5.3).
2 *Interim analysis of activities with a visual element.* This will require facilities for producing video, although not necessarily of the highest quality. The kinds of analysis can range from frame by frame comparisons and documentation for interactional studies to creating video clips with very coarse temporal quality; rates as low as one or even one-half video frame per second can give a good overview of the structure of an event such as a ritual or other ceremony.
3 *Data acquisition.* This will require facilities for producing video clips, though not necessarily of high quality. For example, with a temporal quality rate of one frame per minute or so you can record activity at a location for a day and analyse patterns of space use, traffic density or other periodic activity.
4 *Indexing videotape.* This will require facilities for producing video clips, though not of high temporal quality. An index consisting of one frame per ten seconds for thirty hours of tape results in a computer document about 40 megabytes in size. Time codes can be associated with the frames, which

makes it very easy to locate sections of tape which you can watch using more conventional video equipment, including video decks which can be directly controlled by the computer.

In all cases video clips can be manipulated by other computer tools, such as database management systems (Section 2.3), or inserted into word processing documents or hypermedia authoring programs (Section 2.3.4), as well as specialized programs for working with video material. With the system level support indicated below, there is support in each of these applications for taking smaller clips from the original clips, regardless of what application they are embedded within, and installing these new sub-clips into the same or other application. This is done by reference, rather than making a copy, so the new sub-clip does not add appreciably to the storage requirements. A five minute video sequence can thus be broken into smaller and smaller sequences within an application such as a wordprocessor, where these can be documented, while retaining the entire clip for reference. There are a number of computer applications specifically for managing and manipulating still images, video clips and relevant textual documentation.

One problem with discussing conventional or computer-based visual materials with consultants in the field is adequately recording which image or image components are under discussion at any given point in the discussion. This problem is greatly compounded with video. I have experimentally used computer-based tools for displaying visual materials during an interview, digitally recording the discussion into the visual document, as well as the location and movement of on-screen pointing devices, such as a mouse, over the image under discussion. At a later time the interview is played back, including movement of the on-screen pointer to indicate what image was under discussion, or a part of the interview pertaining to a specific image or video clip is retrieved by activating that image or video clip.

It is much simpler to display video clips (or still images for that matter) than to produce them while in the field. Any mid-range computer, notebook or desktop, can display digitized video at some level using free or inexpensive system level software.[5] It is important to have system level support for video, so that you can use video clips in your other programs which are not designed to support this kind of data. You can paste clips into wordprocessor documents and databases, as well as programs designed to use video directly.

You will get very poor quality from a monochrome screen, and surprisingly good results from a 16-level greyscale screen, although 256 colours or greys are much better. For notebook computers with flat screens, it is much better to have an active matrix screen rather than a passive matrix screen,[6] although the former adds a lot to the cost. Some computers can output to a conventional portable colour television, which is usually inadequate for text but displays digital video in a plausible manner.

The problems relating to producing digital video in the field at present are similar to still images, only compounded by requirements for more special hardware, memory and much more disk storage. Although the use of digitized video in the field is one of my current research interests, I can recommend this

only if you are willing to make a large investment in time and money, as at present it requires a lot of equipment and expense. Although I shall summarize current hardware considerations, you should consult a support person about the latest developments as the terrain is changing almost daily at the moment.

Presently 20 megabytes of RAM memory is fairly comfortable for video work involving small image size (about one-quarter of a normal screen, approximately 320 pixels by 240 pixels). This memory is not required to play back the video sequences, only to produce them. Any amount of memory up to 1 gigabyte (2^{30} = 1 gigabyte = 1,073,741,824 bytes) would be very welcome.

In terms of disk storage, one minute of quarter-screen size video at 15 frames per second requires about 4 megabytes of storage. Full-screen video (640 x 480) requires almost four times as much. Currently, this factor alone vastly reduces the portability of a system for working with video, although it can be overcome.

If you want to take video documents to consultants, but can produce video at a base, you might consider two different computers. With current software support digitized video at 10–12 frames per second (fps) is easy and acceptable, if crude, for ethnographic work, and 15 fps is possible with faster notebook computers. Note that video at both these rates will be rather crude, but is good enough to be useful. The standards for conventional video are 25 fps for PAL and SECAM and 30 fps for NTSC video standards. It is possible to attain these speeds, even at full-screen resolution with expensive hardware support, but in my opinion it is not at present a reasonable option in the field unless there are exceptional reasons.

For further discussion of digital video see Section 5.3.

3.3.2 Audio recordings

The range of options for audio recordings follow similar lines to visual recordings (Section 3.3.1). For the serious user of audiotape similar options for computer representation and control are available. Computer-controlled audio decks can locate and play tape sequences on demand from either direct requests by the user as a result of searching a database or from a fieldnote reference. Audio of reasonable to high quality can be stored directly on disk, and a number of programs for editing and modification are available.

Although the options are similar, digital audio recording directly to the computer poses few problems other than disk storage, and this is not on the scale of digital video. One minute of audio suitable for voice or low-quality music reproduction can be stored in about ½ megabyte of disk (with some compression). At compact disk quality one minute is about 10 megabytes. The equipment necessary for low-to-medium fidelity sound is quite inexpensive and trivial in weight, size and power requirements.

Like images and video, sounds can be imported into word processing documents, databases and other computer tools. Audio clips can be manipulated by other computer tools, such as database management systems (Section 2.3), or inserted into word processing documents or hypertext authoring programs (Section 2.3.4), as well as specialized programs for working with audio material.

With system level support, there is support in each of these applications for taking smaller clips from the original sound clips, regardless of what application they are embedded within, and installing these new sub-clips into the same or other application. This is done by reference, rather than making a copy, so the new sub-clip does not add appreciably to the storage requirements. A twenty minute audio sequence can thus be broken into smaller and smaller sequences within an application such as a wordprocessor, where these can be documented, while retaining the entire clip for reference.

Like video, in some cases there may be no need to fully transcribe audio material in this form, since direct and immediate access to the audio clips can be directed from a computer-based document, while retaining the context of the clip.

3.3.3
Material culture, artefacts, physical measurements

Standard computer applications can greatly ease the difficulty of cataloguing, describing and referencing instances of material culture, including photographs, artefacts, botanical and biological specimens. The use of authority lists (recommended for manual management) simplifies the problem of systematic treatment, as well as access to the references or even images of the materials (Section 3.3.1). As with other data, active cross-references can be linked to other record types, which also improves access.

Although measurements of length, distance, temperature, humidity etc. are usually transcribed from instruments to paper (or a computer), if you use a lot of measurements of a particular sort, it may be advantageous to log measurements directly into a computer database using a special version of the instrument which hooks directly to a notebook (or smaller) computer. Otherwise guidelines for special registers are adequate (Section 3.2.6).

3.4
ANCILLARY FIELD ACTIVITIES

There are a number of activities in the field which, while not research, contribute to the process of research. Although you would never take a computer into the field for these purposes, they are worth considering once you have done so.

These generally cover basic functions, such as:

1 vocabulary drills, the word lists you develop can easily become electronic flash cards;
2 field accounting for expenses, which can make good use of either database programs or spreadsheet calculators, especially handy when using multiple currencies (Ellen 1984a);
3 making appointment files, for which special purpose programs are available;
4 making official-looking correspondence, which has been of considerable benefit to me on several occasions;
5 occasional entertainment, for instance you may fancy a game of chess from time to time—video games should be avoided by those weak of will.

Chapter 4
Fieldnote and textual data

4.1
RESOURCES FOR FIELDNOTES AND OTHER TEXTUAL INFORMATION

Fieldnotes are a fundamental tool for ethnographic research. The application of computers to the production and use of field notes is a basic requirement if computers are to greatly benefit ethnographers.

The range of information which might be included within fieldnotes is quite diverse, from notes on observations, interviews, house inventories and genealogical data to diagrams of house plans and the layout of agricultural plots (Ellen 1984b: 282). This chapter focuses on the use of computers to support the production and analysis of notes, interview transcripts and other textual documents—the textual components of fieldnotes. I have attempted to sidestep aspects of theory, focusing more on operations than the value of these operations, which is more or less dependent on the individual researcher. For a good overview of computer-based textual methods for qualitative ethnographic research, you should consult Pfaffenberger (1988). For a broader range of views Fielding and Lee (1991) is recommended.

Fieldnotes are, by their very nature, very cumbersome to use. They are a record of observations, narratives and new insights, primarily organized by chronology rather than topic. Indeed, a part of their value is in this order, recording the ethnographer's development of different ideas and lines of inquiry. Although difficult to use, the fieldnotes are usually the most significant source of information both in the field and out: constantly referred to, updated and cursed for incompleteness (cf. Jackson 1990).

A conventional method to make fieldnotes more accessible is to produce indexes, based upon classification codes and keywords chosen by the ethnographer. Although indexing fieldnotes is a favourite activity for those days when you just cannot bear to face yet another difficult day, the notes are rarely indexed to satisfaction, since the indexing process itself is somewhat dependent on the stage of the investigation (Ellen 1984b; Seidel 1991).

Computer programs can enhance the use and analysis of textual documents using methods such as automatic indexing, concordances and high-speed search. The main drawback is that these documents must be available to the

computer and its programs in 'machine-readable' form. This basically means someone must type the manuscript into a computer. This is most easily done, of course, if a computer is available at the time of consolidating the notes or transcribing the interviews: in the field. However, paper notebooks are relatively cheap and portable, rarely malfunction and require little power. Computers are relatively expensive, are generally less portable, occasionally malfunction and require rather more power. To justify these disadvantages, computers must yield significant benefits.

For many anthropologists, a small computer might be justified in the field for no other reason than to enter notes using a wordprocessor. Writing or typing fieldnotes into triplicate notebooks, as Ellen suggests (1984b: 282), is neither easy nor fun. Once entered into a wordprocessor, or perhaps a specialized fieldnote application, as many copies as required can be printed, merged into other documents based on subject headings or copied onto diskettes for safe deposit or mailing.

Most anthropologists who have used computers in the field take most of their rough notes using paper, entering and consolidating these onto the computer in the evening. There are still a few conditions not ideal for using a computer capable of reliably storing fieldnotes; room temperature in the Antarctic, for example (although at least one anthropologist has used a computer there). There may be special hazards for computers in the field—dust, heat, humidity, insects and mildew, among others—but if you can use a typewriter of any description in your field conditions, then you should be able to use a computer there as well. A computer is more than a typewriter replacement. A better description is 'a typewriter, copier, filing cabinet and infinitely large wastebasket'.[7]

The simplest way to enter notes is to type the notes with a wordprocessor exactly as you would have written or typed them on a piece of paper (Section 1.3.1.1). This is the model which underlies wordprocessors, so wordprocessors are fairly well adapted to this particular approach. Even the most basic wordprocessor is useful for note entry. Unfortunately, there is little you can do with a wordprocessor alone, other than enter, edit, copy and review your notes. Pfaffenberger suggests that one of the problems with wordprocessors is that, for the most part, they were originally developed to 'de-skill' typing, not for writing (1988: 18). Wordprocessors were certainly not developed to support ethnographic research. Other programs will be required for comprehensive search, access and analytic support.

There are alternatives to using a word processing program which may be advantageous. Text-oriented database programs (Section 4.6.3) usually have quite reasonable facilities for the entry and editing of text, and collect a number of operations[8] to assist in the organization of material such as fieldnotes, as well as incorporating many text-oriented classifying, searching and collation operations (Section 4.6).

The most basic, and common, use of a 'computerized notebook' is to locate sections of the notes relevant to some topic of interest, replacing the more traditional indexing and eyes-over-pages methods. There are a range of partial computing solutions to these problems, some simple and some quite sophisticated. All are partial solutions, because no existing computer-based method can refer to the meaning of the notes in any useful way (even two

anthropologists can have difficulty in agreeing on this). Computer programs are restricted to referencing literal structural features of the notes (Pfaffenberger 1988: 41). If you type in the notes as you might with a typewriter, these structural features are limited to a model implicit in the technology of writing on paper: letters, words, lines, paragraphs, sections, pages and chapters. With such 'raw' texts most contemporary computer software is incapable of identifying structures more complex than paragraphs, or even lines, unless user-defined boundaries are indicated.

The content of a text can only be referenced by most programs in terms of explicit lexical features (ibid.). In a ordinary text, most programs are limited to indexing, locating or counting specific words or phrases which appear in the notes. If you impose a more discrete structure, then more useful work can be performed by the computer with programs designed to exploit that structure. Additional structure can be as simple as including distinctive classification codes and keywords as you consolidate your notes (Section 4.5), as is suggested for conventional fieldnotes (Ellen 1984b; Sanjek 1990b; Seidel 1991), or as complex as imposing direct links between entries that you judge to be related (Section 4.6.5) or including formalized statements of content (Section 8.2).

Computer operations for the analytic reduction of notes are obviously for much more specialized use: a list of the frequencies of words in a text does not suit everyone's analytic needs. For others this is quite useful. Literary scholars have long used word frequencies to gain clues about authorship. Some have carried this even further, considering the frequencies of letters in the document. The problem with most analytic procedures for anthropological analysis of fieldnotes is that, despite considerable noise to the contrary, fieldnotes are not a text in the same sense as a literary text,[9] and the specific words and patterns of words are consequently of less interest. Unfortunately, this is all the computer can respond to unaided by additional references to structure. Pfaffenberger notes, 'In qualitative data, the significant patterns are not principally encoded in any form the computer can detect, namely in instances or absences of lexical items' (1988: 41). Most anthropological uses of analytic methods will depend on you imposing specific structure for this purpose (Section 4.6).

Fieldnotes serve another important purpose in anthropological analysis: to contextualize other material collected in the field and the writing we do based on our analysis of the fieldnotes. Other types of information are greatly enhanced if the note references relevant to the information can be easily retrieved (Section 4.4, Chapters 2 and 3). Computers are capable of representing almost all of the conventional forms of records that we make of the field; besides written records (and their structure) these include photographic material, video tape, audiotape and maps (Section 3.3, Chapter 5). High-performance computer environments have operations which support all of these data types, not only for display but as objects which can be interactively annotated and interlinked (Section 4.6.5). For example, with available hardware and software you can write notes, include audio samples or annotations in the text and include synchronized, full-colour photographs or video sequences in the document (Section 3.3.1). If you have an inexhaustible supply of money, you can have a staggering array of data acquisition devices attached to your computer, with these inputs integrated into a single document.

4.2
CAUTIONS AND ENCOURAGEMENT

Many ethnographers are reluctant to trust their field notes to a computer in the field, fearing 'computer wipeout' (Sanjek 1990c: 38), although this view is countered by Trotter (1992: 55–8). In some circumstances, especially in the past, this attitude was not altogether unjustified. For example, in my first fieldwork in Pakistan in 1982–3, due to unique circumstances I had an inadequate supply of computer diskettes (of very low capacity) with no way to get more. This factor, in conjunction with having a mains-powered computer in an area that only sometimes had power and having no printer, I decided against entering my fieldnotes into my computer, though I did keep summaries and indexes of the notes on computer files. The notes might have fit, but along with other more pressing needs for diskettes, I decided I could not ensure that I could securely store my notes and have guaranteed access to them at all times.

Now, with the advent of very small, reliable computers and printers this kind of situation should occur less often. Even if the computer fails while in the field, if you have a printer the worst outcome is that you have paper copies of your fieldnotes. Computer disk loss or failure is a more likely hazard. If you make daily copies of your notes onto diskettes with the same care as your paper notes, there is little chance of losing more than a few hours' work. In some ways the data is more secure. I generally carry an extra copy of my notes on a single diskette with me at all times in the field. Therefore, there are not just one or two copies to be stolen, destroyed in a fire or lost.

Having said this, it is only fair to mention that everyone I have known has lost work due to carelessness at least once. The loss does seems to have a curative effect. Diskettes should be treated with a bit of care. Data loss on diskettes is almost always the result of careless treatment. This is not to say there are not real hazards, especially with the conditions in many field sites.[10] Diskettes should always be kept in sealed airtight boxes, exposed only when necessary. Disk drive slots should be covered with plastic tape, when not in use, to discourage dust and insects. Despite having a habit that computer disks are alleged to dislike, I have lost data on only three diskettes over fifteen years of use, and never in the field. One was left on the back seat of a car at 40°C, another was left next to a phone which rang (erasing much of it) and the third was probably due to an airport X-ray machine (the X-rays are not really a problem, the conveyor belt and motor can be). In the third case I was able to retrieve the data using a disk utility for that purpose (Section 1.3.1.1). More likely hazards are losing a diskette or having some accident occur to it, such as a fire or a dousing in water. If you ensure that your notes are on more than one diskette, and that all the diskettes are not in one physical location, then you are exceedingly unlikely to lose the computer copy of your notes. The paper copy faces the same hazards as always.

Computers require power, but there are many full-featured computers which can be maintained by charging batteries with solar electricity. More specialized low-power models can operate for up to one year of daily use on perhaps 20–30 AA (LR6) alkaline batteries. Solid-state disks record information with security at least as good as paper. Solid-state flash memory, for example, safely stores the contents of up to twenty notebooks (350,000 words) on a chip which weighs

about 20 grams and is 1.5 cm x 2.3 cm, for ten years with no power. With the current state of technology, a low-power computer useful for note taking and assistance in analysis of notes (e.g. a full-sized keyboard, 2–4 million characters storage capacity and plausible LCD (Liquid Crystal Diode) display) weighs as little as 1–2 kilograms, and can be expected to drop further. Even smaller computers are available which accept hand printing directly,[11] with a weight as little as 500 grams.

4.3
COMPUTING RESOURCES FOR NOTES

Although there are a wide range of computer-based operations which you can use to assist with entering, maintaining, accessing and analysing your notes, some will be more appropriate than others, depending on your particular approach to notes and what you use them for and depending on how much structure and information you have added to your notes, since some computer operations will require specific conventions to do anything very useful.

You must, of course, make any judgments on the applicability of any particular operation to your research. This is more of a point than a truism. In both quantitative and qualitative approaches to analysis there has been a long history of abuse by graduate students and senior staff alike. Methods are only valid when they relate to specific analytic models; they are a part of an overall research process. Researchers constantly misuse quantitative methods such as statistics by applying inappropriate methods to inappropriate samples and making inappropriate interpretations of the results. The introduction of computers only compounds this problem by distancing researchers from the analytic processes (Johnson and Johnson 1990: 175–6). It became possible to apply a statistical method ‘because it is there’, rather than for a theoretical purpose.

4.3.1
Defining contents and properties of fieldnotes

Seidel, as the author of a popular program for qualitative content analysis,[12] relates his concern for computer-assisted qualitative analysis:

> My concern is that qualitative data analysis might get reduced to this [data reduction], and that qualitative researchers might start working in this manner, not because it is the best or most appropriate way to proceed, but because the technology makes it easy for them to work in this way.
>
> (1991: 115)

The greatest trap in which one can be caught is to proceed with research and analysis using a singular formula, regardless of the situation to which the formula is to be applied. It is doubly important to evaluate a computer tool for its appropriateness. Moreover, computer programs generally are a part of the process, and none are total solutions, regardless of the claims within promotional

literature. You must know the assumptions within the program (these are often very simple, but crucial) and match these to the results you wish to achieve.

To make productive use of computers at the core of your research, you must select computer methods that are compatible with that research. Poor results will come from the use of inappropriate techniques. To specify your requirements for computing resources to assist in access and analysis of fieldnotes or other textual materials, you must first identify the properties of fieldnotes and how these properties fit into your goals. In short, you specify the information in your fieldnotes, the purposes you currently use this information for, the operations you currently undertake to accomplish these purposes and the benefits you want to achieve by the use of a computer.

Although Jackson (1990) describes considerable variation in how anthropologists practice the processes, identified by Clifford (1990), of 'inscribing', 'transcribing' and 'describing' the components of fieldnotes, most anthropologists still seem to aspire more or less to the standards described by Ellen (1984b) and Sanjek (1990b). The principal variation seems to focus on the word fieldnote, which Ellen avoids by referring to 'written records' and 'conventional notebooks' (1984b: 282). For our purposes, the exact placement of textual documents into categories is not strictly necessary. Notes, diaries, transcripts and other texts share many properties and requirements in the context of fieldwork. I shall refer collectively to these as notes or fieldnotes, except where distinctions are important.

4.3.2
Contents of fieldnotes

Ellen lists the following range of information which might appear in general notes:

(a) Verbatim texts: words and phrases taken from informants [with additional information to contextualize them].
(b) Translations of verbatim statements.
(c) Generalizations about behaviour in particular situations.
(d) Crude summary translations from informants' and actors' statements; these are usually highly selective.
(e) Descriptions of actions (or activities) perceived (for example, rituals). These are affected by our own cultural construction of what we regard to be 'events'.
(f) Interpretation. Without it concise and full description is impossible.
(g) Diagrams and illustrations in support of (a) to (e).
(h) Quantified statements (e.g. lists of numbers), relating to (e).

(1984b: 285)

This information, while varied, has one important structuring principle common to all field materials. Fieldnotes are the result of a process. They are produced over time and have a chronological structure which is of considerable importance. The process of research has an impact on the content and use of

notes. Earlier notes will often be more general in interests, more focused to the original proposal; developmental. As the research progresses and the body of information increases, notes begin to become more focused on a smaller range of areas of observations, more detail is taken for granted, general observations are noted more briefly, literally being ticked as having occurred. There is often a shift where parallel lines of reasoning become convergent, where fewer and fewer categories account for more and more cases. The chronology must be preserved, not only because it helps document the development of the particular models used to select information, but also to preserve the transition from mostly etic to mostly emic representation which will transpire in most ethnographic research.

4.3.3
Conceptual operations on fieldnotes

Some basic processes associated with the production of fieldnotes (derived from Ellen 1984b and Sanjek 1990b) are as follows:

1 Rough notes are taken in the course of fieldwork.
2 The rough notes are transcribed and expanded into notes in the evening, filling in additional information from memory, tape recordings or other devices.
3 The notes are classified and keywords are inserted.
4 The notes are consolidated with other notes by cross-reference.
5 The notes are consolidated with other field data such as tapes, photographs, survey data and interview transcripts, with possibly some cross-reference.
6 The notes may be copied and the copies stored into rough categories.
7 The notes are indexed according to the classification in point 3.

Most of these production processes are intended to facilitate the use of notes during the field research and afterward in ‘writing up’ (or down). Uses of notes include the following:

1 Looking up matters of detail. This is time consuming, depending to some extent on keywords and to a large extent on memory and muscle.
2 Finding fieldnotes relating to areas of classification. This is important in the formation of hypotheses or, for those who do not have hypotheses, the formation of ideas. These are located in part using the indexes for the notes, and in part using similar processes to those in 1.
3 Finding fieldnotes related by person, place or episode. Since much of what is observed are processes and many of these processes extend over weeks and months, it is often necessary to assemble these into an account of the overall process or episode. If there are cross-references, these can be activated. These notes are also located in the manner of 1.
4 Identifying written context for interviews, surveys, genealogical records, photographs, video etc. This depends on the presence of a separate log of

such materials, which can locate primary references. Secondary references are located as in 1.

5 General browsing of notes with no specific objective in mind other than to reflect on what has been written, drawn or otherwise represented.

From these two lists it is clear that, content aside, most of the structural operations in the production of fieldnotes accommodate the operations associated with the use of fieldnotes. That is, the use of fieldnotes dominates the production of fieldnotes, at least in terms of our design. Added structure included in the production of fieldnotes represents an attempt to solve problems of use which emerge because of the complex contents of fieldnotes and shifts in emphasis which take place over the course of fieldwork (Section 4.3.2).

Even the content of fieldnotes is generally selected to be useful at some future time. Fieldnotes are not so much a conversation with ourselves, as a conversion of experience into a tale with an unknown plot and no conclusion. Fieldnotes, and the writers of fieldnotes, do not create meaning but rather 'defer meaning' until 'the silent tomb speaks' in the context assembled by many units which are 'the same but not identical'.

Most of the work of using fieldnotes is assembling this context. Although browsing through the notes can be a continually edifying experience, to do the work we call ethnography we must develop a set of samenesses within which we can identify differences. Thus the most common operation in the use of fieldnotes is 'Find'.

Finding relevant notes using a computer is not a trivial exercise, and requires no less structure than the manual method of use. The three basic methods which we manually use with fieldnotes are (a) searching through the notes for relevant notes, (b) looking up notes from an index we have been updating, and (c) remembering relevant notes and using a combination of (a) and (b) to find them. We have the distinct advantage of being able to read the notes and to comprehend some form of overall meaning associated with this reading. At present this is not an operation available for general use on computers, but to a limited extent may become possible in the future.

Computers do possess a number of properties which make it very easy to locate instances of literal text in the notes. This is the basic operation used, in some form, by virtually all present computer programs which support access to and analysis of documents. This suggests that most computer programs of this sort are likely to be used as a support tool, where the computer finds the literal text and people decide how to exploit this capability, either manually or by using other supporting computer operations incorporated in the program to store or otherwise manage the results.

Relying on the basic 'Find' operation has a number of limitations. The most fundamental is that although there is a correlation between a word and a set of meanings, fieldnotes are not mere collections of words structured by writing, but are heavily contextualized, which makes 'meaning' a difficult proposition. Looking up words with a computer is a statistical exercise, where some of the notes retrieved will be 'hits' and others 'misses'. Still, with the right program a

computer can find and record all instances in a matter of seconds, leaving you to work out which are useful.

The number of hits relative to misses can be increased. The two main methods are similar to methods anthropologists use with manual notes: classification codes and reference to context. Classification codes are the same as those we use in fieldnotes. Classifications attempt to denote a broad dimension of meaning which can be be applied to a unit of text in the form of a specific code word, such as 'MARRIAGE' or 'DEATH'. If you do a literal search for 'DEATH' you will pick up all notes which incorporate the code 'DEATH' within the note text. If the find operation in use is case sensitive,[13] then capitalizing the code terms is adequate to distinguish 'DEATH' from 'death' and 'Death'. However, when looking for literal text, 'death' and 'Death' are also considered to be different. Most find operations in programs make case-sensitivity optional, because most of the time it is a nuisance. A simple way to insure you can distinguish an instance of the code from an instance of the word is to prepend or append a rarely used punctuation character to the code term, e.g. '%DEATH'.[14]

Classifiers capture only one dimension of meaning and as research progresses you are likely to need greater specificity. You might, for example, want to examine notes which are classified with respect to both death and marriage. Conceptually, one solution is to find notes which contain the code '%DEATH' and then to search these notes to see if '%MARRIAGE' also appears. The implementation is a bit more complex.

The first problem is that we have to define the boundaries of a note in terms which are easy for most computer programs to use, a literal marker. If you type the notes as a normal text there is no purely lexical means of detecting when one note ends and the next begins. With some of the conventions (and non-conventions) that anthropologists use, such as noting place only where it changes and using half-a-dozen ways to represent dates, it is nearly impossible to write even a complex program which can always find note boundaries. Without note boundaries there is no way to determine that a given note contains two key words. We could partially ignore this problem with a single note, because the word is in the note and if the computer shows you the word embedded in the note this is adequate in many cases.[15] The simplest method of indicating notes is to mark note boundaries using a specific lexical convention. This can be something brief, such as '{' (if we are careful not to use this bracket elsewhere in the notes), or something more explicit to human eyes, e.g. '{BEGIN'.[16]

A second problem is related. Meanings can be correlated to a large number of different words. Any given search with a single word will miss relevant references because another word will be used. Thus there is always a 'missed' category of notes.

A third problem is related to the morphology of written language, e.g. words take different forms, sometime irregular, in different grammatical contexts. This can increase the number of missed notes.

4.4
SOFTWARE REQUIREMENTS FOR NOTE ENTRY

There are advantages to entering notes with the wordprocessor of your choice, taking account of the recommendations in Section 1.3.1.1. The reason for this is not so much the virtue often given by designers of programs for note analysis, 'Most researchers already have wordprocessors and would prefer not to learn to use another one' (Davies 1991: 62), but more an issue of flexibility. Good wordprocessors are just not that hard to learn. Many good access and analysis programs have no (or poor) facilities for text entry, especially the kind of word processing operations an academic author might write for a program, and these programs might not exist if the programmers had to replicate word processing operations in each program. Many researchers would probably not use these programs if they had to endure the kind of wordprocessor an academic author might write.

There are alternatives to using a word processing program which can be advantageous. Text-oriented database programs (Section 4.6.3) usually have quite reasonable facilities for the entry and editing of text, and collect a number of operations to assist in the organization of material such as fieldnotes, as well as providing many of the text-oriented classifying, searching and collation operations discussed in Section 4.6. Most of these programs have idiosyncratic file formats which are incompatible with most other programs. If you choose this alternative, make sure that the program is capable of writing the text out into more conventional files, so that other programs can be used to access and analyse your notes.

Many support programs expect input text organized into lines or paragraphs, with no special features such as diagrams, fonts, diacritics, typestyles or special tab settings, so many of the 'advanced' features of a wordprocessor cannot be used by these programs. These features may still be useful, since you can review and print the notes with the wordprocessor, but if you use these features the wordprocessor or entry program should be able to create a file which consists of just ordinary text with no special characters or formatting.

In no case should you be worse off than if you wrote the notes into notebooks. For practical reasons, some practices such as writing in the margins might be substituted for, especially since the text in computer form will be easy to edit. Unlike a notebook or a typed document, a computer-based document is never 'finished', despite the concern of Clifford (1990). If you are insistent about marginal notes, there are wordprocessors which support these. Diagrams should be supported, but high-quality diagrams are not still not particularly easy to draw without special hardware (Section 5.5).

4.5
CONVENTIONS FOR NOTE ENTRY AND CONSOLIDATION

Your intention to use computers should not determine or 'control' the content of your fieldnotes in any significant way. However, most anthropologists impose

considerable structure on their notes, spending considerable time classifying, coding and creating indexes for them (Ellen 1984b: 285–8).

There are several ways you can adapt your note entry and consolidation process to a computer. Most of the programs you might use to support your access and analysis of notes will expect some combination of the following six kinds of structure and/or information to be imposed on the note. These are presented in an order that tends to be roughly cumulative, but there are programs which expect these in just about any combination. You must figure out what you need and then try to find (or design) programs or combinations of programs which fill these needs.

The first kind of structure is more or less a 'conventional' note. You enter what you would have entered had you been following your usual method. This might be similar to that shown in Figure 4.1(a). The advantage is that entry requirements are simple and there are a lot of generic program types that support this kind of structure; index-making programs, concordances, keyword in context (KWIC), word frequency and simple text matching search programs, among others. The disadvantage is that the kind of information that can be reported is limited with respect to context. Because there is no computer-discernable structural information to divide the notes into units or the file into notes, the text is addressed in terms of words, lines, paragraphs and pages, which may not be adequate for your purposes (Pfaffenberger 1988: 39–43).

This situation can be improved by indicating where the note begins and where the note ends (Figure 4.1(b)). Although we can usually deduce this simple bit of information, most computer programmers expect you to tell them, in a manner determined either by the programmer or, more ideally, by yourself, designating a character or set of characters as a boundary marker, such as typing '%END' after each fieldnote entry or, as in Figure 4.1(b), '@N' for the beginning of a note. The same set of programs are used as for the conventional case. Some search programs (Section 4.6.3) can take advantage of this structure to allow more complex conditions for the search (two or more terms), and report in terms of more meaningful units. Some programs may require both a beginning and an ending delimiter. When you use programs which need to know where note boundaries are, these programs will have a means of letting you designate the special boundary marker you employed.

Since programs can only index and find lexical terms, you can improve the resolution of your search by explicitly marking the codes and classifiers you have chosen for your note sections. Again, we can usually distinguish between a code and the content of the note. Computer programmers usually expect you to conform to a convention which lets them identify a code in a program, or allows you to designate what the conventional indicator will be. This can be as simple as appending a character or two on either side of the code, e.g.' @Ethnic@' (Figure 4.2(a)), or using an introductory keyword, e.g. 'C: Ethnic' (Figure 4.2(b)). Again, the same set of programs apply, but with greater resolution (you get more of what you want and less of what you do not want) (cf. Pfaffenberger 1988:39). Applying this level of structure lets you and the program agree on what is a code and what is not, and gives you the ability to identify sections of note with code classifiers. An extension of identifying codes is to identify the scope of the code: what segment of

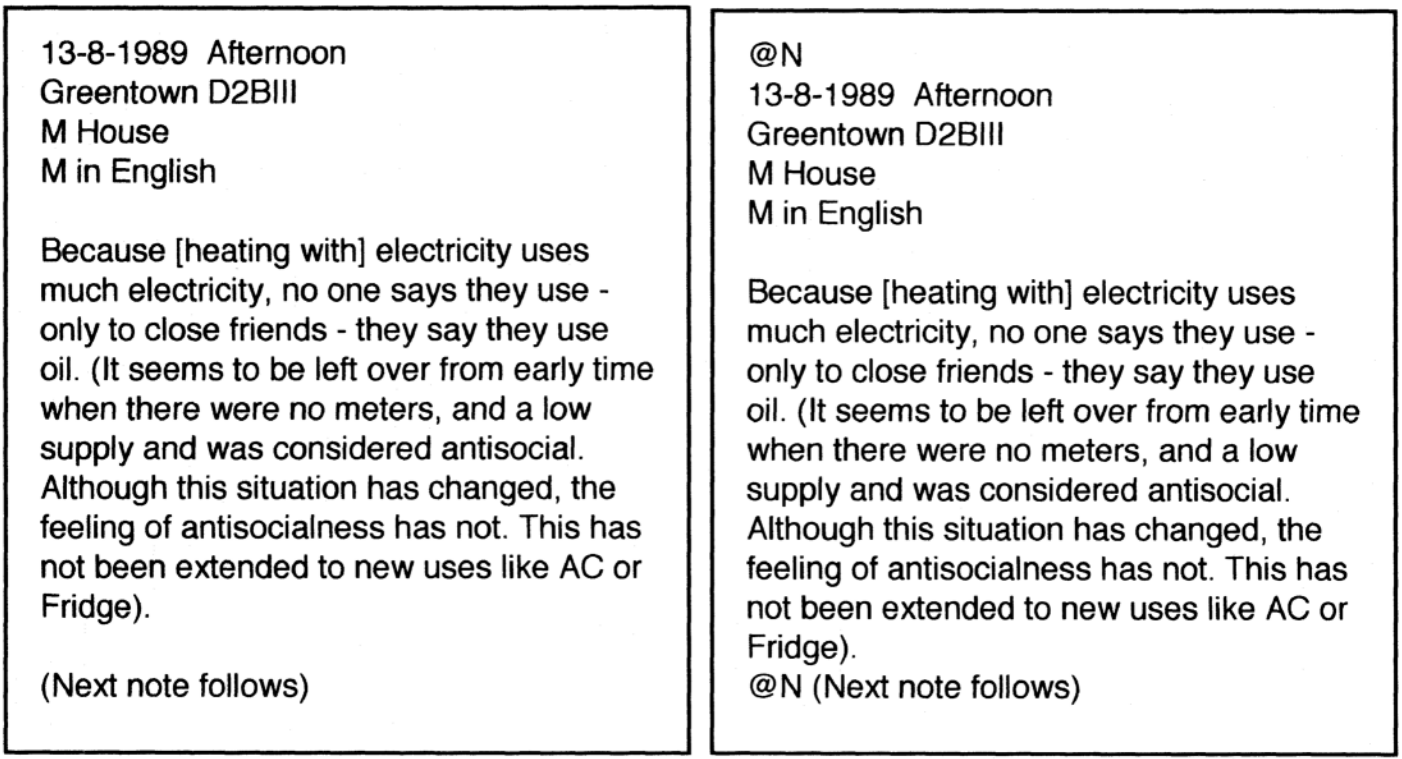


Figure 4.1 (a) A 'conventional' note; (b) note with lexical boundary code

the note does the code apply to? If you always classify whole notes then this is equivalent to indicating note boundaries. If you associate classifiers with smaller sections of notes then a device similar to that used for note boundaries is needed, i.e. another set of conventions or special characters.

Keywords are more specific and idiosyncratic than codes, and usually designate the content of relatively small sections of a note (Sanjek 1990b). These are treated in the same manner as codes by most programs, although many programs permit you to selectively search for keywords or codes respectively if you use different lexical conventions to indicate keywords (Figure 4.2(a)), or the program provides an explicit convention for keywords (Figure 4.2(b)).

Since programs can only index and find lexical terms, you can improve the resolution of your search by explicitly marking the codes and classifiers you have chosen for your note sections. Again, we can usually distinguish between a code and the content of the note. Computer programmers usually expect you to conform to a convention which lets them identify a code in a program, or allows you to designate what the conventional indicator will be. This can be as simple as appending a character or two on either side of the code, e.g.' @Ethnic@' (Figure 4.2(a)), or using an introductory keyword, e.g. 'C: Ethnic' (Figure 4.2(b)). Again, the same set of programs apply, but with greater resolution (you get more of what you want and less of what you do not want) (cf. Pfaffenberger 1988:39). Applying this level of structure lets you and the program agree on what is a code and what is not, and gives you the ability to identify sections of note with code classifiers. An extension of identifying codes is to identify the scope of the code: what segment of the note does the code apply to? If you always classify whole notes then this is equivalent to indicating note boundaries. If you associate classifiers with smaller sections of notes then a device similar to that used for note boundaries is needed, i.e. another set of conventions or special characters.

Keywords are more specific and idiosyncratic than codes, and usually designate the content of relatively small sections of a note (Sanjek 1990b). These are treated in the same manner as codes by most programs, although many programs permit you to selectively search for keywords or codes respectively if you use different lexical conventions to indicate keywords (Figure 4.2(a)), or the program provides an explicit convention for keywords (Figure 4.2(b)).

Some programs make use of explicit fields (Figure 4.3). Fields are structured slots which identify and introduce information of a particular sort, and which programs (and programmers) can refer to and report. To take advantage of this level of structure requires a special kind of program such as a text-oriented database management system (TDBMS, Section 4.6.3). In a TDBMS fieldnote record you might include a 'Date' field in your notes (e.g. 'Date: 21/2/93'). At a later time you can limit searches to notes that occur on a date, before a date, after a date or between two dates. Although it may be obvious to you that this is a date, without an identifying label, it will certainly not be obvious to a computer program. This sort of identifier is crucial to get best use out of many programs, since it is the only clue to some of the rough semantic and pragmatic content of the note. Some programs have special fields and meanings for things like date, speaker, place and identification number, and have a built in notion that things

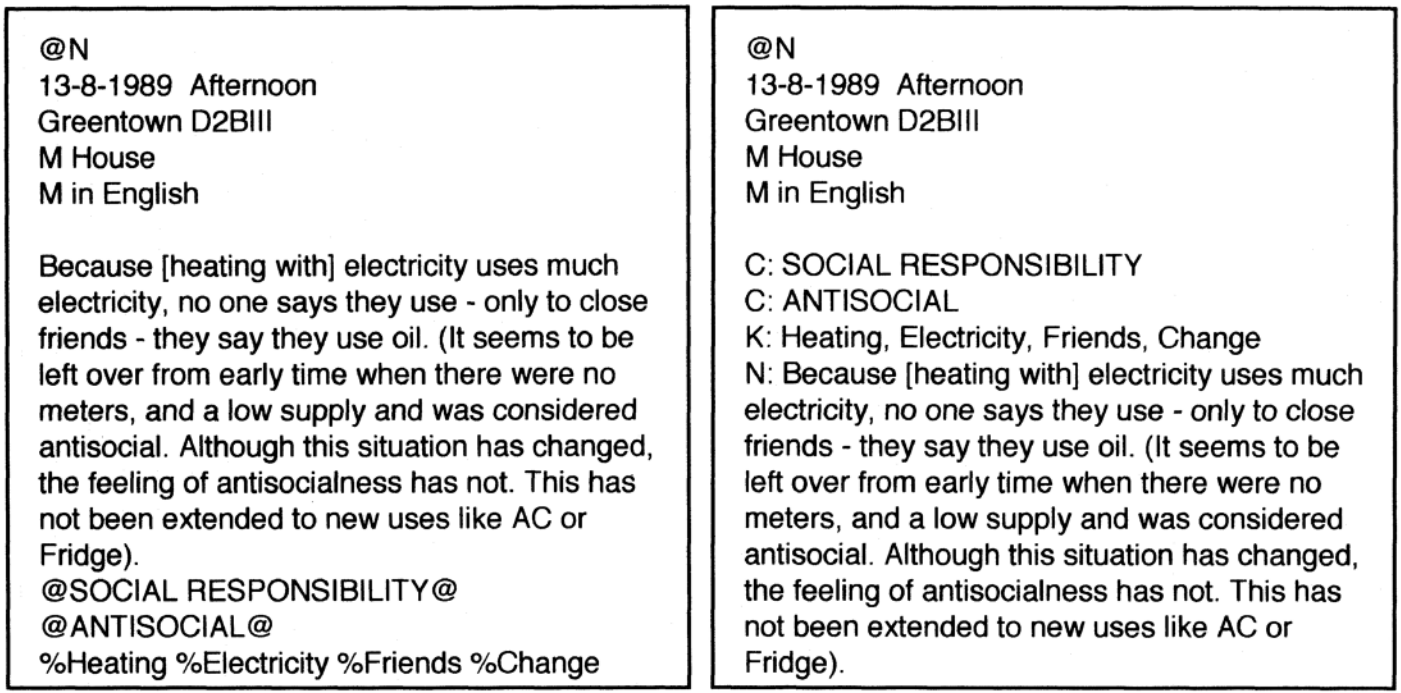
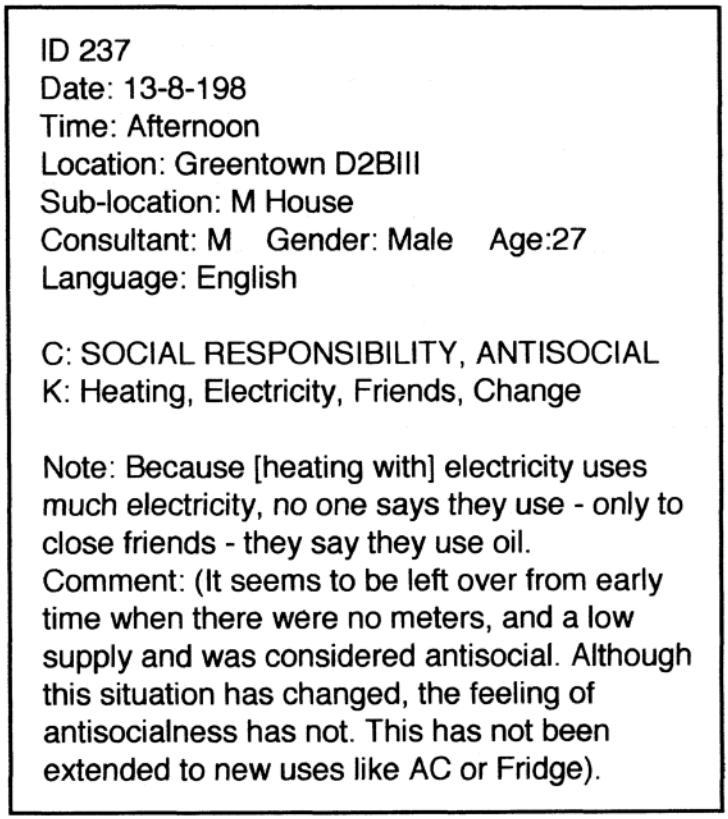


Figure 4.2 (a) Note with lexically coded classifiers and keywords; (b) note with explicitly coded classifiers and keywords


ID 237
Date: 13-8-198
Time: Afternoon
Location: Greentown D2BIII
Sub-location: M House
Consultant: M Gender: Male Age:27
Language: English

C: SOCIAL RESPONSIBILITY, ANTISOCIAL
K: Heating, Electricity, Friends, Change

Note: Because [heating with] electricity uses much electricity, no one says they use - only to close friends - they say they use oil.
Comment: (It seems to be left over from early time when there were no meters, and a low supply and was considered antisocial. Although this situation has changed, the feeling of antisocialness has not. This has not been extended to new uses like AC or Fridge).

Figure 4.3 Structured note with fields

identified as a date can be interpreted as a number with special meaning. Other programs permit you to designate all the fields, and you must give them a type so that any special interpretation required can be made. Most programs of this sort will include a date type.

Another kind of representation is used in experimental programs, such as hypertext, and knowledge-based representations (Benfer *et al.* 1991; Fischer and Finkelstein 1991). This is best characterized as explicitly encoding some aspects of the structure of the semantic and pragmatic content of the notes, if not in full than at least to a significant degree. There are two main methods of doing this with material based on texts.

Hypertext (Section 4.6.5) is currently the more practical of the two, in the sense that good commercial software and a large amount of experimental academic software is available and more will become available. Hypertext more or less allows you to record your understanding of how the different sections of the text fit together, so that this structure can be explicitly reproduced instead of statistically reproduced as in the conventional index and search operations.

Knowledge-based representations involve some way of translating the natural-language text into a more formal structure which can be interpreted to the point of letting you address issues of content, entailment and association. This is intended to provide an unambiguous statement by the ethnographer regarding what they are attempting to say. This approach is discussed in Section 8.2.

4.6 TOOLS FOR ACCESSING NOTES

4.6.1 Wordprocessors

If you use a wordprocessor to enter your notes you will soon have a rather large collection of document files. In a year's fieldwork an ethnographer can easily write 1,500–3,000 pages of fieldnotes and diary entries. Most wordprocessors do not work well with thousands of pages in a single document file. With a one-hundred page limit per document file, 2,000 pages of fieldnotes will occupy between twenty and fifty different primary files (not all will be 'full'). If you put copies of notes into topic-oriented files, another 60–150 files may be generated. Also there should be at least two independent back-up copies on different disks in different locations at any one time. If you intend to use additional computer tools to access and retrieve notes, there are additional text-only (ASCII) files.

Obviously, you must plan how to organize and maintain these files. Naming files is important. Do not name files 'new notes' or 'latest notes'. Develop a system of naming primary files and copies of primary files which is clear and consistent. Use of dedicated note-handling programs (Section 4.6.4) or hypertext editors (Section 4.6.5) will eliminate some of these organizational problems.

4.6.2 Basic search programs

Basic search programs are a class of programs which have existed for at least forty years. They are not specifically intended for access to notes, but more generally for locating literal text in files. They accept search terms (queries), including some range of Boolean queries (Section 4.1), and search through a designated set of files. Most programs support either a large number of files or very large files, often only limited in size by the general computer system.

To find all instances where the term 'marriage' is contextualized by the term 'divorce' you would start the program and type in a query formed something like:

```
FIND marriage AND divorce
```

Which might be read, 'find all units which contain both the terms marriage and divorce'. What will you get from this query? This depends on what the unit of co-occurrence is.

There is considerable variation among programs in this respect. Like a word-processor, many use either lines or paragraphs as the unit, which does not really

correspond to any usable unit for most research purposes (Pfaffenberger 1988: 41–3). Worse, unlike a wordprocessor, the only context which is commonly reported is the unit itself. More advanced programs have facilities to let you designate, to some extent, both the searching unit and the reporting unit. There are two approaches to this. The more common approach is to let you designate for searching purposes how close the words in the search term must be to each other in words or lines. You might specify that divorce must be within twenty words of marriage. The reporting unit in this case might be designated as ten lines prior to and following the search terms. This adds some flexibility, but is still not really adequate.

Another, more useful, approach is to let you designate specific text which will delimit a search and reporting unit. Unless you want to use paragraphs, this usually means you must have inserted these delimiters in the files while entering them (4.5), or you used a specialized editing program which inserts the delimiters for you.

Although this entails you taking on the task of literally imposing structure on the original note text, it has the benefit that you can retrieve meaningful units (assuming notes are meaningful units). This approach still lacks resolution for many problems with accessing notes. Although it is far better to retrieve notes as a unit rather than some arbitrary unit defined in terms of the structure of text, notes always contain an array of different sorts of information. If we want to limit searching or reporting to one of these sub-units of a note, for instance to a segment which consists of interview material, most search programs cannot accommodate this because although a delimiter may be designated, very few allow a beginning delimiter and ending delimiter, which would be necessary in this case. That is, when writing notes each type of entry in the note would require its own beginning and ending delimiter.

Another problem with most searching programs is that these are both over and under inclusive. These find search terms, usually with little reference to context, and subsequently locate entries which are of little interest (Pfaffenberger 1988: 39). They look explicitly for the search terms, missing entries which are relevant, but fail to correspond to the specific terms (ibid.: 40–1). This problem can be limited somewhat by adding explicit classificatory terms, usually with a special keyword identifier consisting of one or two unusual characters, e.g. '%% Kinship' (Section 4.5).

Classifiers still suffer from the problems of internal note organization. A single entry might have a number of classifiers which correspond to different sections of the notes, the same scope, a scope enclosed within that of other keywords or an overlapping scope with other keywords. Ideally a keyword could be specifically identified with a researcher-specified section of the notes, regardless of its inclusion within another, marked section, or indeed regardless of whether it overlaps another existing segment. Most specialized note access programs satisfy this requirement (Sections 4.6.4 and 4.6.5).

Some search programs search pre-computed indexes of a set of files, rather than the files themselves. Such programs are not necessarily the same as index-building programs, though some perform this operation as well. This variety of search program is not usually difficult to identify, because the distributors normally make quite a noise about its indexing capabilities. They are intended

for people who produce lots of files and have trouble managing them. Most accept search terms, including Boolean queries (Section 4.1), and search through a predetermined set of files (from the index). All support a large number of files, usually between 50 and 250, and some can work with thousands. Some of these programs can be configured to automatically index any new files you may create in specific directories (which can include the entire disk).

In general, these programs find the text very quickly, often with apparently immediate response. The speed is derived from a previously constructed index of all the words in a user-specified list of files. You can usually specify a list of words not to include in the index, and common words (e.g. a, an, the) are excluded automatically. This index is not usually intended for human use, and serves only to support the search function of the program. The results of a search can usually be displayed on the screen, or you can specify a file to put results into. Because the indexes are based on words, it is not easy to search for literal fragments of text which include more than one word, although this can be simulated using a Boolean search expression and a minimum search unit. Most have a provision for 'fuzzy' matches, where you can match a pattern against words, e.g. 'marri*' for all words starting with 'marri', '*marri*' for all words which include 'marri' within, such as married, marriage, remarried, unmarried.

Besides the problems for search programs in general, index-based search programs have another serious drawback. Indexing a large amount of data in a large number of files takes a lot of time. Some programs may take several hours, or even overnight, to index a body of data as large as a year's fieldnotes. The program cannot be used until the index is complete, since the index contains information necessary to find information in the files. After the fieldwork period this may be acceptable, since the notes should not be modified much. In the field the problem is more serious. Some programs can accommodate changes to one file and are able to re-index it individually. Other programs must re-index everything each time a change is made to any file in the index.

The other main approach for general search programs is to scan through a designated set of files without the benefit of an index. Their properties, advantages and faults are very similar to index-based searching programs. The main advantages of a scanning search program is that the text to be located can often be more flexibly specified and not limited to individual words, and because an index is not used there is no time spent building an index after changes have been made to the source documents. The main disadvantage is that scanning search programs are slower in retrieving text from the source documents. The speed of retrieval can range from very slow, perhaps a minute or two to scan a megabyte (220 pages or about 1 million characters) of text, to as little as eight seconds per megabyte. The average time for existing programs is about thirty seconds to one minute per megabyte.

4.6.3
Database management systems and fieldnotes

Database management systems (DBMS) (Section 2.3) are a class of programs for managing information which is relatively highly structured compared with

wordprocessors or search programs (Section 4.6.2). Most conventional DBMS programs (Section 2.3) are too structured for use with textual materials such as notes, diaries and interview texts. However, the models underlying these programs have been applied to a form specialized for working with text.

A program of this sort is usually called a textual DBMS, a free-text DBMS or a text-oriented DBMS. The basic concept behind these programs is to combine the structure associated with a database management program with the flexibility of a search program. Unlike using a wordprocessor, a plethora of files and a search program, many text-oriented DBMS offer a 'total solution', which basically means they include a wordprocessor-like text entry program, operations to design entry form, facilities for

Table 4.1 Possible entry form for fieldnotes

Note ID:	Place:	
Date:	Time:	
Informant Name:	Sex:	Age:
	Status:	
Codes:		
Keywords:		
Note:		

Source: Adapted from Ellen 1984:286

managing whatever files are created on disk, support for locating, sorting and grouping information and facilities for producing reports.

To begin you must design an entry form. With most recent DBMS this is a very simple operation. First you define some labels (sometimes called prompts) for the different categories of information, called fields, which will appear in a record (a complete entry such as a fieldnote). A entry form for fieldnotes might look something like Table 4.1. You will probably have to specify the sort of information each field represents, e.g. Place is literal text, Age may be a number if appropriate, or literal text where other categories might be used, Date is a date (usually a specific type of data for such programs) and Time is a time (also usually a type of data). Note, Codes and Keywords will be designated as variable text or long text, a special type which designates relatively long textual content. The usual maximum of most DBMS for the long text fields is about 30,000 characters (about 12–15 pages of notes), but some are limited only by the amount of storage on the disk. Contrast this with conventional DBMS, which typically allow between 64 and 256 characters as a maximum.

Having defined the form, you can begin entering information. The DBMS will put the form on the screen and you fill in the appropriate blanks with the appropriate information. Features to look out for in text-oriented DBMS are as follows:

1 You should be able to enter information in any field in any order, and edit or re-enter a field if you wish. Control of where you are entering information should be easily under your control using either arrow keys or a pointing device such as a mouse, e.g. you should not have to go through a complex 'conversation' or set of menus every time you want to change something.
2 Support for text entry should be as good as a simple wordprocessor.
3 You should be able to examine other completed records while entering a record (not common).
4 You should be able to change the format of a record. Otherwise you can find yourself with a lot of wasted effort. Make sure you are satisfied on this aspect before going to the field.
5 You should be able to import data from other sources relatively easily.
6 You should be able to export data for use by other programs.

Like the search programs (Section 4.6.2) text-oriented DBMS support searches using literal text, using a Boolean search expression if necessary. In addition searches can also be made by referencing the different categories in the record format. For example, the query:

```
SELECT fieldnote.*

WHERE fieldnote.sex=male

AND fieldnote.age < 27

AND fieldnote.note CONTAINS marriage

AND fieldnote.note CONTAINS divorce.
```

can be read,

> report all the fields of record type *fieldnote* when field *sex* of *fieldnote* is 'male' and field *age* of *fieldnote* is less than 27 and field *note* of *fieldnote* contains 'marriage' and field *note* of *fieldnote* contains 'divorce'.

We have to specify field *note* twice since the expression parsers for most programs are note smart enough to read an expression like 'fieldnote.note CONTAINS marriage AND divorce' in the way that we might intend. If you wanted, you could report the note number only, or any combination of fields, by using a different version of the 'Select' statement. Also some text-oriented TDBMS are relational (Section 2.3.3), allowing queries and reports to refer to several relations or files.

In summary, text-oriented DBMS offer an integrated set of operations which are useful for many of the tasks associated with entry, access to and analysis of fieldnote material and other textual sources. The basic search and report unit is always a record, which should consist of a single note or related text in conjunction with basic structured information such as time, place, consultant information and classificatory information about the note. Retrieval can be based on structured information such as *consultant name, age* or *sex,* unstructured

information (in a computing sense), such as note fields, or a combination of the two. Retrieval can be either a single record, a group of records or a sorted group of records.

There are disadvantages, of course. Many of these programs work with a single file, which can become very large. A season's field notes can easily reach over 4 million characters of storage. Since many of these programs make little use of indexes, they can be relatively slow with large amounts of notes.

Also, databases of this size are much too large for current floppy disks (although this limit may increase soon). This has two consequences. First, these files are difficult to copy onto floppy disks (or small solid-state disks) for security. You will require a special program to do so. Second, you will almost certainly require either a hard-disk drive or a large capacity solid-state disk, as it is useful to have access to all of your notes at one time. Hard disks are less suitable to some field circumstances because of increased power needs, and large capacity solid-state disk drives are expensive.

4.6.4 Programs specifically written to support access and analysis of fieldnotes

Programs explicitly designed to support access to and analysis of fieldnotes and other ethnographic textual sources are perhaps the only generic application which has emerged of disciplinary relevance to social anthropology. These programs are basically text-oriented DBMS with additional operations for ethnographic research, although ethnographic research is fairly narrowly defined in this context. Anthropologists have been involved in the process (Werner 1982; Agar 1983, 1986; Sailer 1984; Pfaffenberger 1988), most programs which focus specifically on the analysis of ethnographic texts have been developed to support some form of 'strip' analysis (Agar 1986) or sociologically inspired models such as 'grounded theory' (Glaser and Strauss 1967; Richards and Richards 1991). When people refer to using computers for qualitative analysis, it is this species of program to which most refer. Although these programs can be useful for anthropologists, the particular formulation of ethnographic analysis these programs promote represents only a portion of what most anthropologists consider to be ethnographic analysis based on fieldnotes.

Pfaffenberger identifies three general processes to be addressed by computer methods for qualitative research; rewriting, coding and comparison (1988: 26). Ellen's conventional approach is more field-oriented; consolidation, classifying, coding and indexing (1984b: 282–8), with a focus more on retrieval or access than comparison or analysis. In part these differences arise from a difference of naming and heading, but there is a more fundamental difference.

Ethnographic analysis based on grounded theory focuses on the discovery of codes and the creation of typologies of codes to describe and relate ethnographic segments or 'strips' (Agar 1986; Pfaffenberger 1988: 27), from which emerges a framework of theory (Pfaffenberger 1988: 28).

This process has some resonance with, and use for, social anthropology. Many of its proponents are anthropologists. However, the process is not adequate

to completely define the kinds of access to field notes required by most anthropologists in the field or afterwards. Anthropologists will use the capability to locate ethnographic instances coded by labels such as 'ethnic interaction', 'conflict' and 'conflicting perspectives' (Pfaffenberger 1988: 34). But anthropologists also require the ability to locate—to be vulgar—facts. By this I mean we must find the answers to specific questions relating to specific events and people; questions such as 'Just who is Abdul's third wife?', 'Who was that guy with Rubina?', 'Who else talked about theft?' or 'How many people live in that household?'.

4.6.5 Hypertext

These are points to consider when entering notes, but is it the best way to adapt to a new medium? Probably not, but it is probably the best way for an anthropologist to begin. As we shall see in other sections, the medium of a computer opens up a huge range of new ways to consider what a document is. Although we may not be aware, the use of pen and paper, typewriter and paper, or even a wordprocessor imposes a hegemony of past technology on the manner in which we organize information. If you type, the transition to a wordprocessor is a fairly friendly one, because wordprocessors are intended to model typing on pieces of paper. Information is entered and displayed in a linear fashion. However, word-processors are an imperfect imitation for the most part. Much more than paper, the view of the document is linear: top to bottom, page 2 to page 3. It is difficult to find operations that correspond to flipping pages. Most computers have screens that show less than half a page. It is difficult to remove a page for later examination. Some wordprocessors can come close to this level of access, but most do not. Although it may be possible to model very accurately a typewriter and paper, this is not necessarily the best manner in which to proceed with a computer. Although you may start with this method, it is not the best place to end.

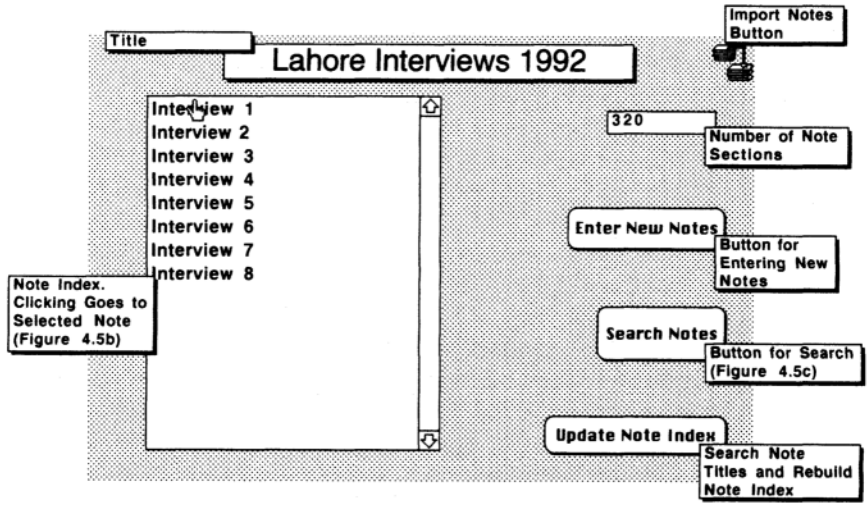
The preceding programs have basically been a medium for replicating a part of what we already do with textual field material: enter and consolidate; classify and code; index; group and compare note entries based on a code or conjunction of codes; scan through for instances of specific detail. These are all ultimately based on the model of a pen and paper, and the various devices we have developed to aid us in accessing such technology.

Hypertext systems attempt to break away from this particular model by redefining the characteristics of a text. Instead of structure superimposed exclusively by different units of information sharing the coincidence of literal bits of text, hypertext is based on a conceptual model where units of information are explicitly linked to other units of information. Where the conventional model is more or less a passive statistical approach to search and access (some of the items will fit your needs, others will not), depending on regularity of form, hypertext extends this with active mechanical links for access, and can deal with quite idiosyncratic forms.

Hypertext attempts to provide a medium where information can be stored embedded within a structure which is intrinsically non-linear, i.e. a document which consists of units of information which can be accessed in many explicit orders, depending on the particular 'thread' the reader is interested in. Units of information are called nodes, and are defined very broadly. They can include text, tables, drawings, photographs, video and sound, and programs, as well as complexes of different node types (Section 2.3.4 and 3.3).

On the surface this sounds ideal for material such as fieldnotes. One of the basic objects of fieldnotes as a means of storing information is that when we come to analyse this material each piece of information will fit into many different structures; we do not want to tie the information to any specific form or model. As we view and review our material we begin to build different frameworks which order the material in the notes. Hypertext provides a means to record structural knowledge so that not only can we record these different views, but we can record the derivation of more abstract views (or analysis) as well.

Little about hypertext is 'automatic'. Hypertexts are 'authored', and the authoring process must be done by someone who is familiar with the material included in the hypertext. In the context of fieldnotes, authoring will consist of typing an entry (called a node) using a text editor provided by the hypertext authoring environment. This node will probably be in the last position of a chronological thread, so that there is a thread which include the notes in chronological order. Indexing and coding can done by linking the node to a node which heads the thread for each index and code term. If a new thread for classification is required, you can begin one. If you need to merge existing threads into a super-category, then a new head node can be created to contain these, while retaining the individual threads. Individual words and phrases can be linked to nodes also. See Figure 4.4 for an example of an hypertext-based fieldnote editor.



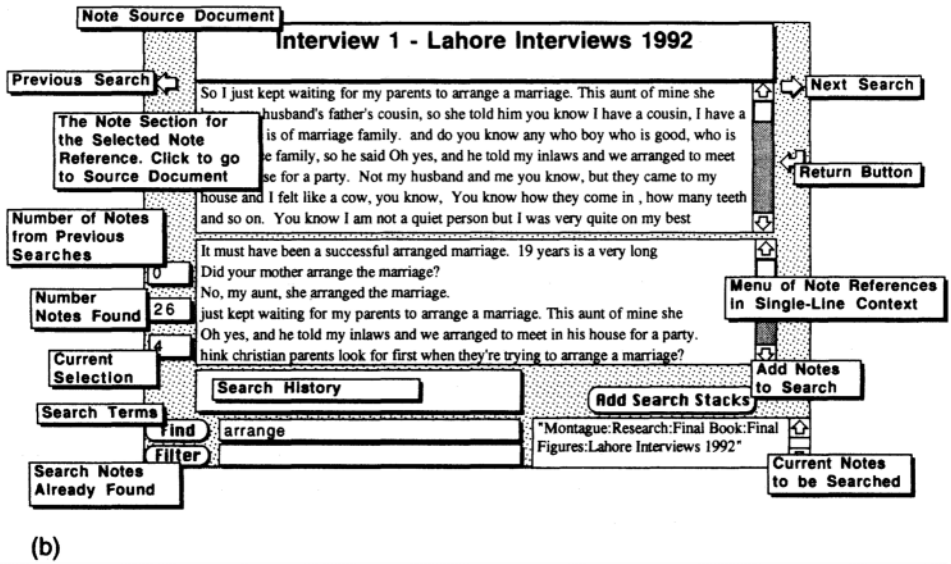
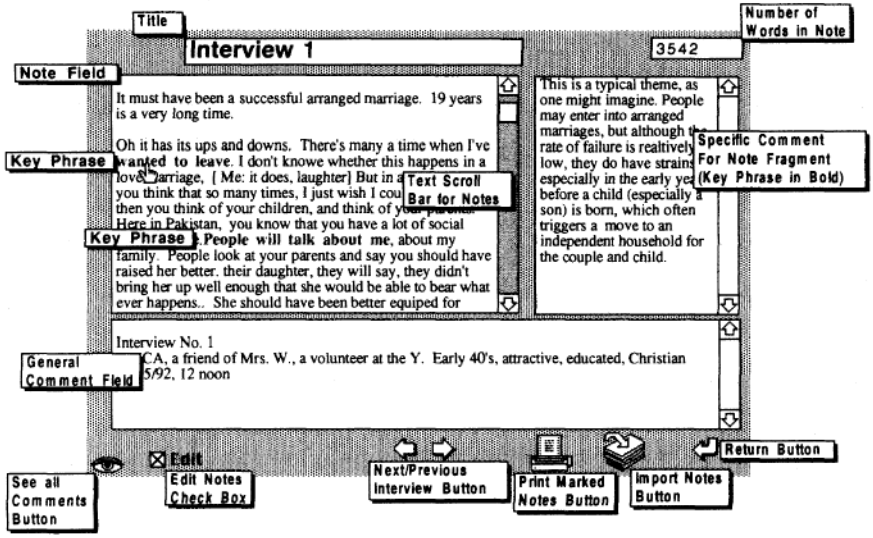


Figure 4.4 Field note search program (examples due to Wenonah Lyon, Lahore, 1992): (a) note index card; (b) a note card; (c) note search card


Title
Interview 1
Number of Words in Note
3542
Note Field
It must have been a successful arranged marriage. 19 years is a very long time.
Oh it has its ups and downs. There's many a time when I've wanted to leave. I don't knowe whether this happens in a love marriage, [Me: it does, laughter] But in a
you think that so many times, I just wish I cou
then you think of your children, and think of y
Here in Pakistan, you know that you have a lot of social
People will talk about me, about my family. People look at your parents and say you should have raised her better. their daughter, they will say, they didn't bring her up well enough that she would be able to bear what ever happens.. She should have been better equiped for
This is a typical theme, as one might imagine. People may enter into arranged marriages, but although th
rate of failure is realtively
low, they do have strain
especially in the early ye
before a child (especially a
son) is born, which often
triggers a move to an
independent household for
the couple and child.
Key Phrase
Key Phrase
Text Scroll Bar for Notes
Specific Comment For Note Fragment (Key Phrase in Bold)
Interview No. 1
CA, a friend of Mrs. W., a volunteer at the Y. Early 40's, attractive, educated, Christian
5/92, 12 noon
General Comment Field
Edit
See all Comments Button
Edit Notes Check Box
Next/Previous Interview Button
Print Marked Notes Button
Import Notes Button
Return Button

(c)

Chapter 5
Ethnographics
Graphics tools for ethnography

5.1
GRAPHICS AND IMAGES IN ANTHROPOLOGY

The use of visual material in social anthropology is well established as a means of documentation and as material for analysis (Collier and Collier 1986). This chapter discusses the capacity and use of computer-based visual representation and analysis, including the simple production of drawings using computers, the incorporation of photographic and video materials, computer-based mapping and the presentation of information in graphical form. The chapter ends with a brief outline of some of the hardware requirements and limits on graphical representation.

Computers used to have a reputation for producing fairly grim displays, with graphics limited to crude bar graphs and drawing pictures of Snoopy using 'X's and 'O's. Fortunately, the trend is for computers (including small notebook computers and even 'palmtops') to have at least decent capabilities for complex visual displays, and this trend is being driven at a faster pace by operating systems depending on graphical capabilities and a growing market for multimedia applications.

As graphical operating systems have become more established, more and more programs are using graphic means of representing information. Sometimes this is more ornamental than necessary, but cumulatively software designers are developing means of representing more information graphically, with the result that computer programs (and the information these represent) have come to resemble objects which can be manipulated like the material tools we are accustomed to, rather than through special textual languages, as was the paradigm before it. More importantly, the introduction of operating system support for graphical representations introduces standardization which obliges software designers to use means of implementing graphical objects in a manner which is compatible with other programs, such as wordprocessors, database management systems and hypertext programs. The ability to create images in one program and use them in another elevates the use and value of graphical information for the ordinary user from a novelty to a useful tool.

The increase in the use of graphical capability opens opportunities for graphical applications in fieldwork and ethnographic representation.

Applications for creating graphical representations fall into two broad categories; general purpose graphic authoring and editing tools and applications specialized for specific representational domains, such as musical notation, representing genealogical drawings, or displaying quantitative and qualitative relationships graphically in the form of graphs, charts and diagrams.

Visual information processing in computing has traditionally been classified into categories corresponding to processes of *image analysis* and *image synthesis* (Hachimura 1987:123–4). *Image analysis* refers to processes which operate *on* images, including content identification, filtering, transformation, measurement, extrapolation and spectrum analysis. *Image synthesis* refers to processes which represent information *as* images, including authoring, plotting, mapping, design, graphing, and animation. Most modern graphical applications have aspects of both these categories: programs which have as their primary function image analysis have facilities for operations such as graphing, animation and authoring, and programs for image synthesis have operations which might include transformations and measurements.

Other than Hachimura (1987) I know of no anthropological work describing automated image content analysis, and as I have no experience of this method myself I shall not discuss this class of application, although there will certainly be applications in the future. Technical information can be found in Bolc and Kulpa (1981) and Pavlidis (1982).

5.2
GENERAL PURPOSE GRAPHICS TOOLS

The most basic way of getting graphical information into a computer is to follow the method we use with pen and paper, substituting a pointing device such as a mouse or tablet (Section 5.5) for a pen and the computer screen for paper. Making drawings with computers takes a little practice, but most programs which support graphic authoring have tools which greatly simplify some operations (such as drawing straight lines and circles or assigning shaded detail). Such a program will not, as yet, make you into an artist, and the computer version of paper has some rather different properties which must be adapted to.

The advantages of drawing directly into the computer relate mostly to what will be done with the result. For some anthropologists, drawing needs might be restricted to simple diagrams, which are much easier to produce in publishable form than using rulers, india ink and transfer lettering. Others may need to produce (or reproduce) field sketches for inclusion into publications or research databases.

General purpose graphic tools also fall into two broad categories, commonly referred to as ‘paint’ programs and ‘draw’ programs. The distinction between these is based on the method of representing the graphical object.

One consideration in choosing an appropriate graphic tool for the field or for other research purposes is to make sure that you can use the drawings in other programs where they are required. For example, if you intend to put the diagrams in a database, make sure the database and the graphic tool use and make diagrams of the same format. Vendors of programs usually include

information like this in the manual. With graphical operating systems compatibility is usually less of a problem, since these provide facilities for transferring information from one program to another.

5.2.1 'Painting' tools

Paint programs are the easier to use for simple work, and in some ways are the more powerful. Paint programs are the closest analogy to pen and paper drawing, and work by providing tools which you use to make marks on a surface. The surface in this case is an area of the display screen, which is divided into rows and columns of small dots, called *pixels*. In this simplest case pixels are either 'black' or 'white' (or 'set' and 'clear'). The drawing is constructed by selectively setting or clearing pixels. This type of image is usually called a *bit-map image*. It is rather tedious to individually set and clear thousands of pixels (a typical display screen has some 300,000 pixels), so the tools associated with paint programs typically serve to create lines and common shapes such as circles and squares, as well as a number of special effects such as 'spray' paint and 'charcoal' textures to the marks. The drawing is edited by 'erasing' pixels, or selecting a group of pixels and moving or copying these to a new location. The addition of colour or shades of grey (greyscale) greatly amplifies the quality and effect a drawing can produce.

Like pen and paper drawings, the objects depicted in the drawing have no identity relative to the media; only people identify the different components which make up the drawing, the computer representation is simply a surface with marks. This provides a great deal of freedom, since the drawing can be selectively and subtly modified at the level of individual pixels. It also results in some constraints; if you want to move, duplicate or resize a portion of the drawing, you must carefully select just those pixels which make up the desired object.

5.2.2 'Drawing' tools

Draw programs work by a process by which the user defines objects, which are then drawn on the 'drawing surface' by the draw program. This is not a difficult process, and for simple drawings it is done using the same tools you would use with a paint program. In effect the draw program records your actions with a tool and stores this as a definition for the object or component you are drawing. The advantage of this approach is that each component in the drawing has status as an object, not only to you but with respect to the draw program as well. This allows you to change the scale and size of these objects, with no degradation of the drawing. Although you can do this with a paint program, the resized section is not as 'good' as the original (Figure 5.1(a)). When you are intending to print the results, this difference is crucial, because a high-quality printer or typesetter will require an image from sixteen to sixty-four times larger than the image you see on a computer screen to make an image of the same dimensions on the

printed page at best quality. The difference in quality is striking (Figure 5.1(b)). More advanced draw type programs can attach information to the different objects, and help you measure not only the scale of the drawing but properties such as area as well.

In terms of basic use, draw programs are more suitable for diagrams and figures where scale is involved, and you may want to experiment with different placements of the diagram components. Paint programs are more suitable for drawing 'realistic' representations, largely because of better 'texturing' control. There are a number of graphic editors which incorporate both paint and draw approaches, usually by having 'paint layers' and 'draw layers' which can be superimposed.

5.2.3
Three-dimensional graphic authoring tools

There is a specialized kind of drawing program used for creating three-dimensional graphics which can be used for representing subjects such as buildings or artefacts. They are generally rather more complex to use than two-dimensional drawing programs, since you are expressing three dimensions on a two-dimensional surface. Currently, most work on the analogy of a lathe or plastic extrusion tool, where you create parts and then attach them.

The advantage of a three-dimensional drawing is that once the image is entered you can display it from different angles, or even from perspectives 'inside' the drawing outwards. In effect, you can build a scale-model of an object or scene on the computer, and examine it from a number of viewpoints (Figure 5.2).

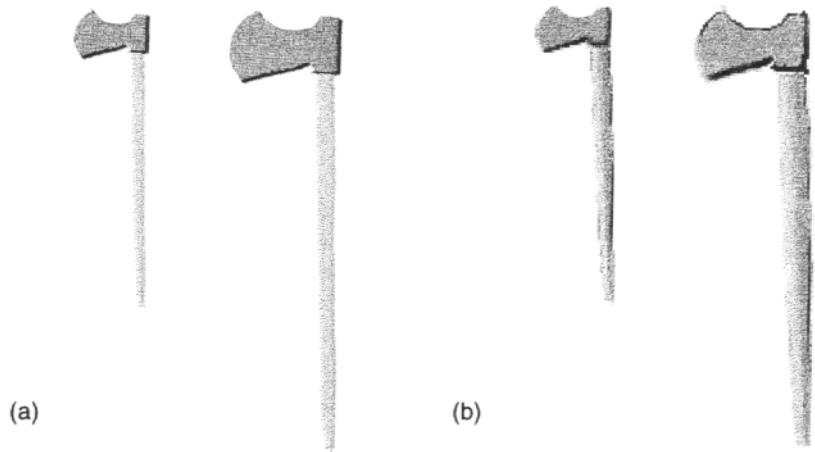


Figure 5.1 Comparison of display and resize quality of Swatti Ax created using (a) draw and (b) paint methods

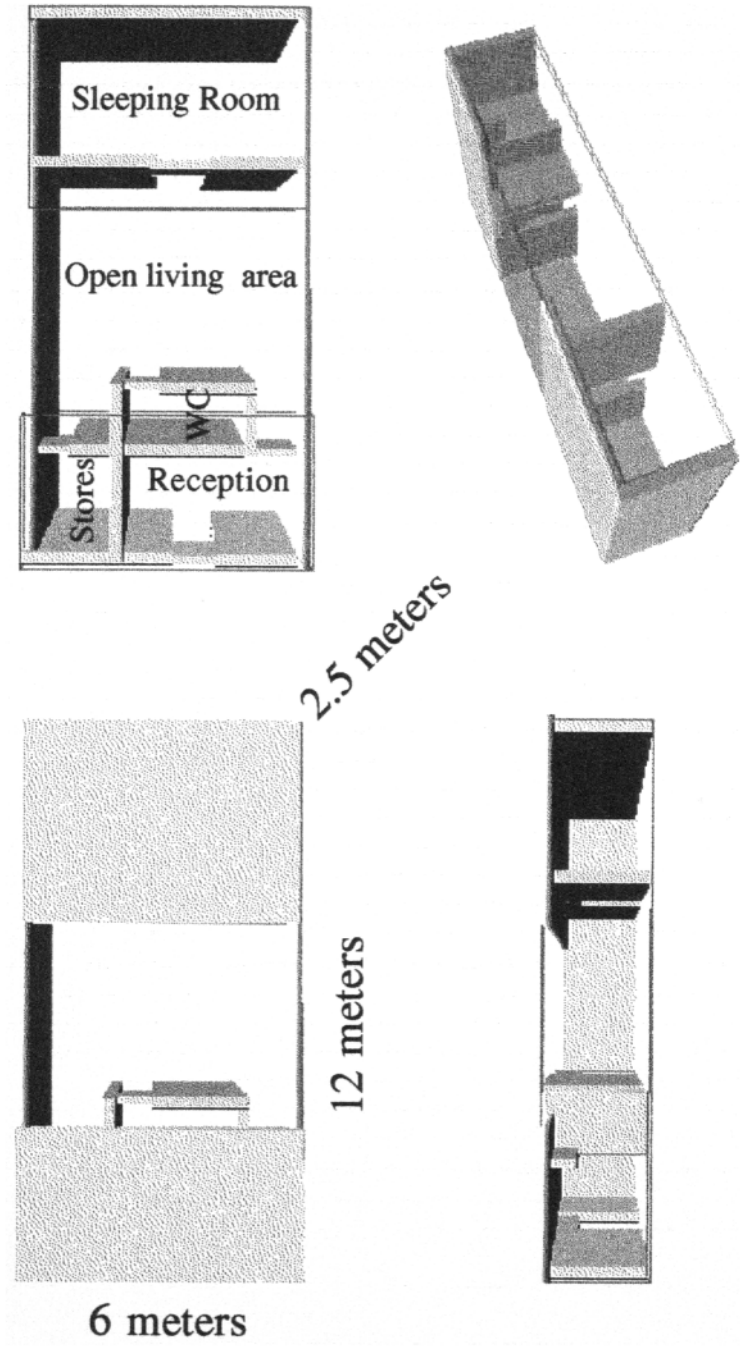


Figure 5.2 An urban Panjabi dwelling

These models can be made quite realistic if you attach colour and texture information to the drawing components, specify light sources in intensity and angle and *render* the image. *Rendering* is sometimes done by the original drawing program, although often the program can only do a rough job of shading and colouring. Specialized rendering programs can create colour representations that are considered *photorealistic* in quality. Once you have rendered the drawing it can usually be converted to a paint-type image, where other kinds of effects can be applied.

If you seriously intend to work with three-dimensional drawings, and especially if you intend to render the results, you are advised to acquire the fastest computer you can afford for this purpose, since this can be a *very* time consuming job for a computer if high levels of quality are desired.

5.2.4 Image processing/retouching tools

These are programs for working with bit-mapped images ('paint' type images), usually images which have been acquired from photographs or video cameras (Sections 3.3.1 and 5.5). These programs are basically enhanced paint programs (and can be used as such) which have a large number of tools for transforming images. They were developed for processing photographs for commercial graphic arts, usually advertising production. They are, however, essential to any anthropologists who want to work with computer-based photographic images.

Although anthropologists have very little interest in some of the commercial uses of these programs, such as 'pasting' the Queen's face onto a past Prime Minister (although these techniques can be quite fun for making creative door signs), most of the other features can be used quite effectively.

One problem with photography in the field is that it is often not in optimal conditions, and there is no opportunity to re-shoot. Many otherwise excellent shots have a 'bright spot', under-exposure, over-exposure, poor focus and other defects. Although most of these cannot be completely compensated for, photographic retouching tools can make many quite unusable images quite presentable. As an example, setting brightness and contrast of an image can correct many of the problems of under-and over-exposure. Slightly out-of-focus images can be sharpened and defects in the perimeter can be cropped.

Another use of these programs is in making montages of examples from portions of images, useful for making sheets for developing typologies or supplying a range of examples. This is also useful when adapting a number of images to share the same *colour palette* when making *indexed-colour* images (Section 5.3.2), so that several images can be shown on the screen at one time. In general, image retouching programs are useful as a tool for converting from one graphics format to another, even between those used on different computer systems. They also are useful for resizing images, especially when making these smaller, although, up to a factor of two or three, they can do a reasonable job of enlargement as well, by sharpening the resulting slightly fuzzy image.

These programs also have a number of very specialized operations for examining the distribution of colours in the spectrum, converting colour models

(useful for producing for publishing), finding edges in the image (useful for classification) and superimposing one image over another. Some have operations useful for making measurements, if you know a basis for setting a scale.

These programs can also be useful for working with data from remote sensing satellites and aerial photographs (Scollar 1978; Hachimura 1987).

5.2.5 Animation and transformation tools

There are a number of programs which have been produced for developing multimedia presentations for business. These basically allow you to define a set of objects (which can include drawn images, photographic images, video images and sounds), often referred to as 'actors', and a background ('stage') and then to define a script for the actors to follow. Most have the capability to build a set of intermediate 'frames' if you give a starting frame and an ending frame. Many have simple to complex programming (scripting) languages associated with them, and can follow contingent paths through the finished presentation.

Programs of this sort can be used to build anything from a simple 'film-strip', an animated multimedia ethnographic presentation or a 'walk-through' model of a village, to models of ritual events, which can be annotated, or indeed shown to informants for judgment.

Another interesting tool preforms an action called *morphing*. Morphing was developed as a special effect used in films for smoothly changing one object to another, and has been used in a number of films and advertisements. One use for this effect for anthropologists is in constructing a range of objects between two emically different objects (which appear etically similar to the ethnographer). As an example, in recent fieldwork in Swat I carried out research on the cultural properties of objects in a mountain community. Using a morphing utility I took drawn pairs of differently named objects which seemed similar to me and created a 'spectrum' of thirty objects which I used to investigate the boundary between the two object types (Figure 5.3).

5.2.6 Graph and chart tools

There are a variety of programs which can display information, especially quantitative information, in a graphical manner. These range from programs which make the familiar bar graphs and the tedious pie charts, to programs which can display and transform multi-dimensional data displays.

Recent trends in object and network representation suggest that similar tools may soon exist for displaying large diagrams of related objects, such as very large genealogies, by using three-dimensional representations or simulated surfaces which distort and compress the image at the periphery, leaving the centre at full size. By providing tools for quickly moving the peripheral data to the centre, larger amounts of information can be maintained on an average-sized display, and still make sense.

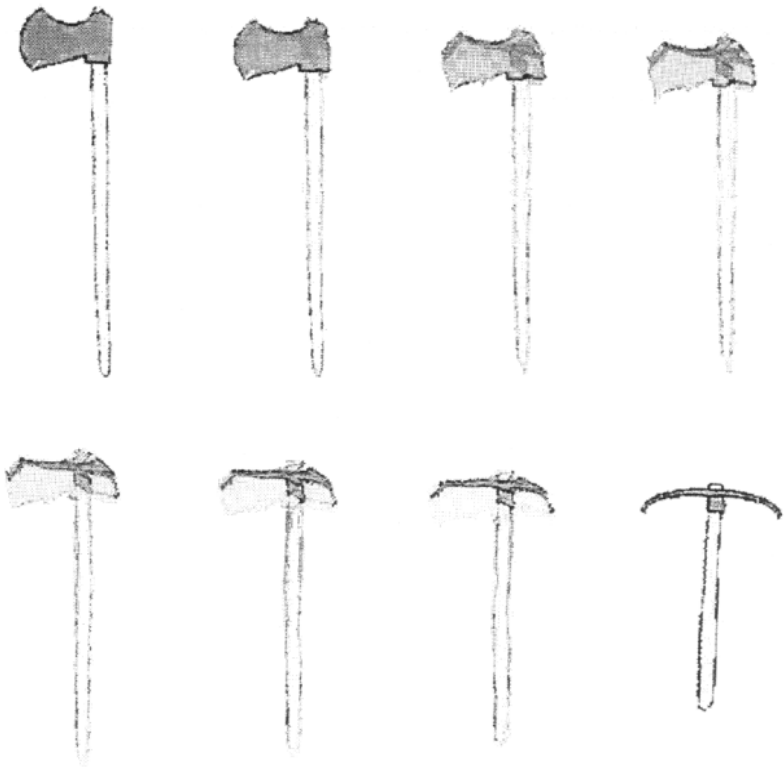


Figure 5.3 Morph of Swatti tools

5.3
USING COMPUTER-BASED IMAGES AND VIDEO

Images have been important in ethnographic research for most of this century. There has been considerable change in how images have been used. Initially images were used to illustrate material culture or performance and to provide a visual reference for people and places. More recently the trend is towards using images to illustrate symbolic aspects of culture; to provide mini-experiences.

Visual anthropology has grown in importance in recent years, although photographic and film methods have been important for most of this century as a means of documenting ethnographic research and presenting some of the results from that research. The use of computers to work with visual materials opens up some interesting possibilities both in the field and in subsequent analysis. Besides their use to document material culture, visual materials can be used directly in the process of accumulating ethnographic data, both as a 'prop' in interviewing ('what are they doing?') and to provide examples to illustrate observations.

The real question here is the overall significance of applying new media to ethnography. Crane (1991) discusses the contention of Marcus and others that ethnography must be essentially writing. The primacy of the text is, of course, important to those who relate ethnography as texts. There is a point here. There is an essential difference between the written text and other forms of 'communication'. Sound recordings and visual recordings are, in a sense, very low level: these are used to record the behaviours observed. But as Marcus points out, film and sound can be manipulated to present particular views of the situation, and because these mimic the behavioural/sensory level of data, somehow greater authenticity is promised. Although I agree in part with his analysis, it is for rather different reasons. Text and textual conventions work well in traditional ethnography and the analysis of ethnographic material, because they constitute an analysis of the material according to conventions and interpretations which have developed within the discipline.

Because of this, language can be used to present essentially formal accounts of aural, visual, olfactory and tactile senses and to communicate these in a way that the actual experiences cannot; as abstractions which can be referenced, cross-referenced, counted and sorted and reliably identified with little ambiguity. Thus, just as Marcus is not asserting that writing is necessarily more truthful, the statements made with language are much more discrete, while those recorded in other media, where the knowledge is embedded in the observer, can be much more phenomenological.

However, visual materials can play an important part in writing. Photographs play an important part in illustrating the text, assisting both the extension and restriction of our imagination. This is an important process, because without these aides there is a tendency to over-abstract in text, playing games with words based on simple abstraction.

Computers can provide a platform where visual and aural information can be well integrated with textual material. Besides acting as illustrations, images can be be actively linked to other texts, further explanation and sounds, or to other images. Embedded video can illustrate points, accompanied with explanatory or descriptive texts. The abstraction, the text, can be integrated with the low-level images of ethnographic data, manageable in a form which is impossible to replicate using pen, paper, photographs and sound recordings.

The equipment required to work with visual materials in the field, even under difficult circumstances, is not excessive if moderate quality is required. The main additions, besides a video source such as a still video camera or a small video camcorder, are a computer screen capable of displaying levels of grey (Section 5.3.2) or colour, equipment for digitizing images and/or video, a large amount of memory (at least 4–8 megabytes) and a substantial storage device for storing the images.

There is no delay in availability of the result. As soon as you have taken the video image it can be transferred to the computer and displayed. Selective enlargements, reductions and enhancements can be applied to the image using image processing software (Section 5.2.4). Using a suitable printer paper copies which are adequate for many purposes can be printed.

However, the use of perhaps £1,000-£2,000 worth of hardware and software for producing simple photographs is about as expensive as using Polaroid™ film. More substantial value comes from the flexible uses of the images.

For further discussion of video and photographic images see Section 3.3.

5.3.1 Usability

The easiest application is the simple use of a computer to increase access to the photographic material itself. One of the problems with taking some 2,000 photographs in the course of fieldwork is that we end up with 2,000 photographs, which are difficult to use effectively due to the simple logistics of placing these in a useful manual filing system. As slides these will occupy some forty trays, as prints some twenty card files.

Using computer media these 2,000 images can be stored on a 50–1,000 megabyte disk (depending on original size and quality required). For fixed disk drives this comes to about 10–50p per image, and for removable media perhaps 2–5p per image.

High-quality images from photographs, slides, video-recorders and still-video cameras can be entered, stored, accessed and viewed on computer equipment with sufficient disk and memory resources. Several ethnographic projects have used such visual materials (Kubo 1987; Macfarlane 1990; Fischer 1993). While not understating the level of resources required, this capability is strictly a matter of cost. The equipment is widely available, with a large number of options, and will become, if not typical, very much cheaper over a very few years. Depending on the quality and format in which an image is stored, from 2,000 to 60,000 images can be stored on a 1 gigabyte disk drive (2^{32} bytes or just over an American billion bytes).

The principal advantages of digital images in ethnographic research are as follows.

1 *Ease of access.* The image (or video sequence) can be recalled from a database, either directly by a reference name, indirectly as a result of a search of keywords or attributes associated with it or sequentually by browsing the image database.
2 *Integration with other information.* Image references can be stored in many databases of a conventional sort, and recalled and viewed in context with the records in that database. Images can be 'pasted' into documents which can be subsequently printed.
3 *Associating information with portions of an image.* Information can be associated with portions of the image, and retrieved contextually by referring to that portion. For example, each of the participants in a neighbourhood argument can be identified by pointing and clicking, and these references can be inserted into a more general database. The participants can later be identified from the image by clicking on the relevant portion of the image, along with other notes or information associated with the image

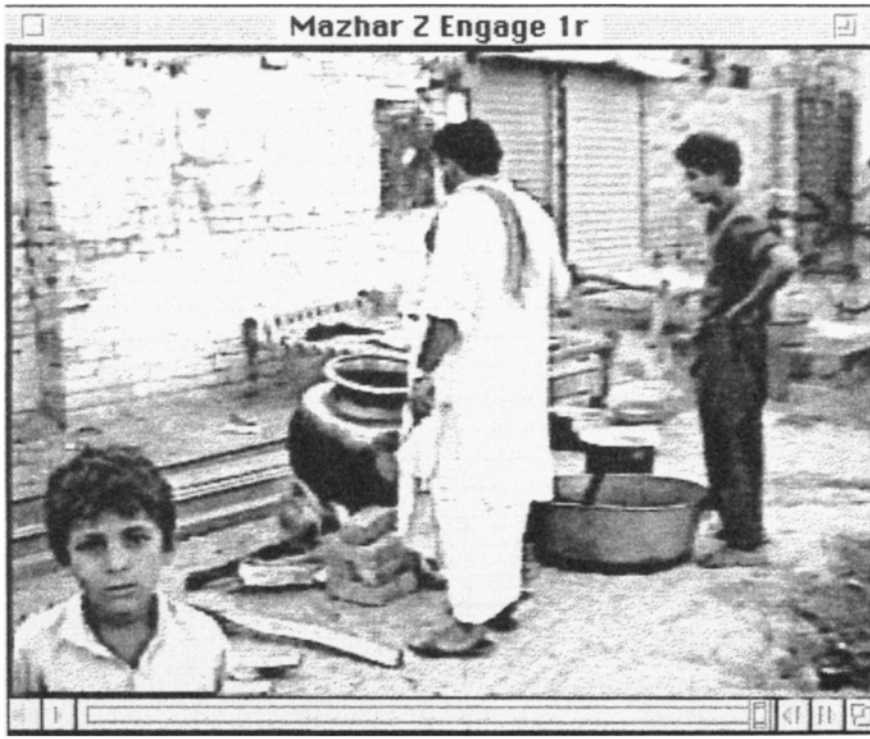


Figure 5.4 Preparing a wedding feast in Lahore: digitized video document in a standard window controller

portion. Portions can also be copied and stored and accessed as independent images, as well as maintaining a link with the original image (Figure 5.4).

4 *Enlarging and enhancing the image.* Images of sufficient density can be enlarged, the focus and contrast improved, the colour balance altered and irrelevant detail removed or subdued. An 'analytically' altered image can enhance desired detail, bring out detail which was 'lost' and provide a image counterpart to verbal notes. Although this opens a whole new area of controversy (what is the ethnographic value of a 'retouched' image), the capability also opens a whole new area of ethnographic expression. Given the small amount of work in this area, the benefits and faults of these processes are yet a matter for anthropologists to assess in the light of practise.

In the use of this material after the field research, there are few disadvantages over more conventional photographic techniques, depending on the hardware available for translating negatives, prints or slides into digital images (Section 3.3.1). The principal disadvantages are expense, some loss of quality, depending on the hardware, and the labour to convert the images. For image

sources such as a video-recorder or still-video camera there are few disadvantages other than expense, as the material can be represented with the same resolution at which it was recorded, the labour is trivial and there should be some improvement in picture quality over conventional television monitors, which tend to be inferior to computer monitors.

The possible impact on field research is twofold. The controversy over the value of photography, cine film and video-recorders in field research is multiplied with the addition of new equipment which must be taken to the field, but this shall be left to the discretion of the researcher (see Blacking, 1984: 200–4 for discussion; also Jackson 1987: 107–27). More interesting is what advantages and disadvantages there are if one does.

5.3.2 Image types

Raster images are composed of rows of *pixels* (PIcture ELements, or 'dots'). Images have attributes of resolution, usually in number of pixels per inch, size, usually the number of rows by the number of pixels in a row (e.g. 640 by 480), and depth, usually the number of different values a pixel can take, generally given in absolute terms (levels) or the number of bits per pixel. Images may be monochrome (1 bit per pixel or two levels, black or white), which are useless for representing near-photographic-quality images, greyscale (2–8 bits per pixel, or 4–256 levels of grey from black to white) or colour (indexed or direct).

Greyscale or *grey level* refers to dividing a scale from black to white into a number of segments or levels, usually powers of two from 2 to 256 (2^1–2^8). Each pixel in the image is associated with a index which indicates which level of grey to display. For greyscale images of photographic material, two, four, and eight levels are not very useful, sixteen and thirty-two levels are adequate for reference and sixty-four levels is the conventionally agreed minimum for 'serious' work. A format with sixteeen-level greyscale images may be acceptable for use in the field, but probably not for subsequent research use. Greyscale of 8-bit depth, or 256 levels, is the conventional maximum resolution required for greyscale images. The appearance on the screen is superior to a very sharp black and white television screen. There are a number of very good programs which support this image format, and it is quite suitable for representing photographic quality images which do not require colour.

Indexed colour images use a few colours (4–4096) from a palette which contains references to possible colours (usually from 4096 to 16,777,216 (or more)). The number associated with each pixel references a colour selected from the palette. For example, a 6-bit indexed colour image can represent any of sixty-four colours within the image. Indexed colour of 2–6 bits is not generally useful for presenting photographic materials. Indexed colour with 8 bits of resolution is drawn from a palette of 256 colours, usually selected from 16,777,216 colours although some inferior standards draw from rather fewer and should be avoided. The quality of 8-bit indexed images is surprisingly good (usually somewhat better than a colour television picture), especially for subjects where the colour variation is low. Associated with each image is a palette, which should be

adaptive to the image, i.e. it should be constructed to fit the colours which appear in the original image. However, images of 8-bit depth may appear 'cartoon-like' if there is too much colour variation in the original image. They are also difficult to adjust or enhance without introducing distortions. There are also some limitations for display of multiple images on some hardware. If precise colour is not required an 8-bit indexed format is adequate for many research purposes.

Direct colour images store all colour information directly with each pixel, rather than indirectly as with indexed colour. The two most common formats are 16 and 24 bits per pixel (the latter is sometimes called 32-bit colour, though there is as yet no standard model for using the upper 8 bits. This is likely to change.) Colour information is usually encoded in one of two standard formats, RGB (Red, Green and Blue) or CMYK (Cyan, Magenta, Yellow and Black). RGB, evolved directly from television standards, is the more common for display on computer screens. CMYK was developed by the printing industry, and is used for images intended for typesetting. Very few microcomputer software packages can display anything other than RGB. A few can convert RGB to CMYK and vice versa, but the conversion is not exact.

Colour is usually encoded in three planes, each similar to a greyscale image, corresponding to one of the three primary colours. Black can be synthesized from CMY (Cyan, Magenta and Yellow) or RGB. Black is included in the CMYK model for printing because synthesizing black from CMY ink is possible but wasteful, and the final results are not as good as special black printing ink. Images with a format of 16 bits encode thirty-two levels of each primary colour per pixel, and may be unsuitable for some photorealistic images, though they are usually adequate if precise information about texture and fine details of colour are not necessary. Images with a 24-bit format encode 256 levels of each primary colour per pixel. With ideal display hardware these images can rival photographs for detail and quality, and on average equipment results are nearly as good as High Definition Television.

5.4
WORKING WITH MAPS

Maps have long been used by anthropologists, as a tool for both fieldwork and analysis. The level of sophistication used varies widely, from a simple, scratchedout drawing of the layout of a village or neighbourhood, to carefully scaled maps of agricultural plots and trade routes.

Computers can be a powerful tool for employing maps. There is a wide range of software available for the representation and analysis of data lined to map coordinates and maps. Unfortunately much of this software is either very expensive (£2,000–£20,000) and designed to operate on expensive graphics workstations, or works with maps of inadequate resolution for the kind of microanalysis which anthropologists tend to do. Less expensive programmes include simple digitized maps which typically have a resolution no better than 6 minutes of arc or 0.1 degrees, which at the equator is nearly 12 kilometres. More expensive programs use maps with resolutions of as little as 1 km, or 30 seconds of arc at the equator.

Anthropologists are likely to use fine-scale maps of a relatively small area, ranging from a village to a valley. They are also interested in details that may not appear on maps which have been prepared by others, such as individual dwellings, agricultural plots, footpaths, irrigation canals etc. Most programs designed to work with maps have methods of importing user-prepared custom maps. Less expensive programs will usually require some trickery, where the scale of the map is related to a much larger scale, with 0.1 of a degree of arc in terms of the program relating to 0.1 of a second of arc on the user's scale. The program can then be used to display the map, and any measurements can be re-scaled manually to the user's 'true' scale. More expensive programs usually have a completely customizable scale.

5.4.1 Creating maps

If you are looking for large-scale maps, computer readable maps are often available from government survey organizations at relatively low cost. You will probably need to write a program to convert these maps into a form you can use, or have someone else do this for you. It is likely that this will become less of a problem in the future.

If a very small area is required, such as your field site, usually the only way to get such a map is to make it. As with the manual version there are often starting points. Reasonable scale maps may be available in a survey office or some other official source. An aerial photograph can often provide a useful start (Scollar 1975). Sometimes the only solution is to survey the area, creating the map from scratch.

Once an appropriate map is found or created, a representation of the map must be communicated to the computer. The most direct way is to draw the original map directly on the computer. Some mapping programs provide facilities for drawing maps directly. Less expensive programs usually have less powerful facilities, and many have none. Most of these programs can import maps drawn using a draw program (Section 5.2.2) or a computer-aided design (CAD) program. You should not use a paint program for most purposes other than creating overlays (Section 5.4.4). Draw and CAD programmes record how the image was drawn, representing the drawing as a set of objects which can be manipulated. Objects can be reproduced smoothly at different sizes and resolutions. This is particularly important for producing printed output. What looks reasonable on the screen will look fairly rough when printed on a medium- or high-quality printer (Figure 5.5). It is also important if the map is going to be displayed at different sizes. Paint programs record an image as a set of dots (or pixels) which are not related by the program. A circle in a draw program is represented by a method for drawing a circle. A circle in a paint package is represented as a set of pixels which look like a circle, which can only be manipulated crudely.

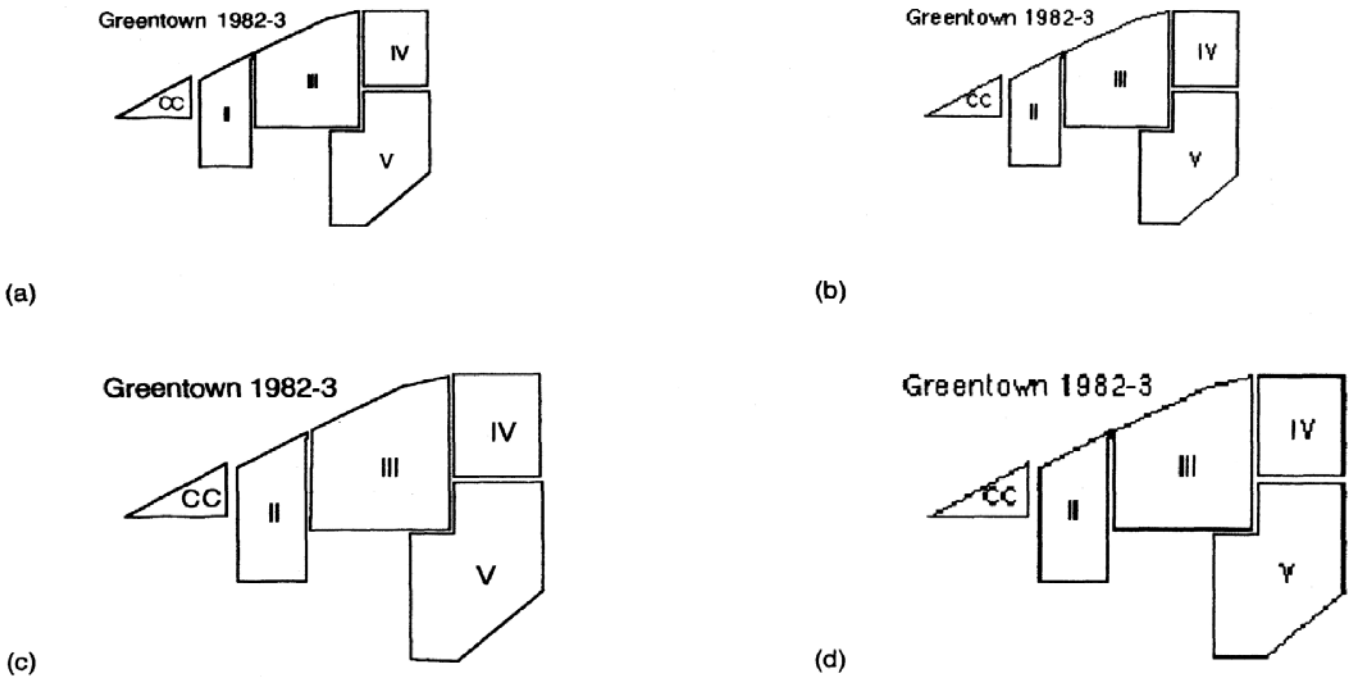


Figure 5.5 Comparison of display quality of field site maps created using draw and paint methods: (a) map created using draw method; (b) map created using paint method; (c) draw map enlarged by 33 per cent; (d) paint map enlarged by 33 per cent

If a base map is available it can be transferred more easily by making a transparency of the map and taping it to the monitor (or indeed taping the map to the monitor), which for many of us is rather easier than free-hand drawing on a computer. The map can then be traced directly on the screen. If the map is larger than the monitor, it can be done in sections, taking care to join the sections on the drawing. This has some disadvantages if one of the sub-areas or features of the map is divided between sections, because the geometry of the sub-area is difficult to reproduce later since it was drawn in sections. A river is represented as perhaps four lines, rather than as a single object. Most draw packages allow grouping different objects to form a new single object.

A variant of this method which is becoming popular is to digitize the base map using an image scanner, creating a bit-map of the map, similar to what would be created by a paint package. The bit-map is then displayed in the draw program (most draw programs can represent a bit-map as a single object) and traced over using drawing operations. The advantage of this is that if the machine has sufficient memory the entire map image can be held in memory, and the draw package can be used to represent each sub-area and feature in the map as a single object (most draw programs will scroll the image automatically when the drawing tool reaches the boundary of the drawing on the screen). Some map programs and a few draw programs have a feature which automatically traces (autotrace) closed surfaces in a bit-map. This will work only if the sub-area is entirely bounded with no gaps in the outline. Scanned images are often not as good as we might like, and it may be necessary to touch up the bit-map using a paint package prior to attempting to autotrace. Another problem is that most autotrace programs trace inside the boundary. Tracing contiguous bounded areas will thus create two objects which do not quite bound each other, the gap represented by the width of the boundary line. Some of the more sophisticated programs can make some corrections for this problem.

Another method, which works extremely well with most CAD programs and more expensive mapping programs, is using a digitizing tablet. A digitizing tablet is basically an electronic drawing pad. These usually have a surface ranging in size from B5 to A0 paper sizes. The most common sizes for inexpensive tablets (£400) are A4 and A3. Larger tablets are very expensive, but it may be possible to get access to one in a university geography, architecture or electronics department. The main advantage with using a digitizing tablet is that the map can be transferred with more accuracy. Most digitizing tablets have an accuracy of 0. 1 mm or better, and the device used to draw or trace on a tablet is generally easier to use than a mouse. Digitizing tablets can also be used to create a list of numerical coordinates, which some mapping packages require for import, especially those which tend to be less expensive or are not designed to work with windowing environments.

Most methods of using computer-based maps reflect practice with using maps manually. The primary resources we can exploit by putting maps on a computer are using the map as a means of retrieving information, plotting references to information on a map surface and making measurements directly from a map.

5.4.2
Plotting data on map surfaces

Plotting data onto a map can be a useful way to examine the distribution of social variables such as clan membership, or to reveal geographical patterns of distribution of other systems such as kinship networks. In principle this is a relatively simple procedure. Almost all of the more expensive Geographical Information Systems (GIS) can perform this operation (Dueker 1987), and many cheaper programs can do so to some extent, although the control over what points are plotted can be somewhat limited.

However, any program which permits the plotting of points onto the screen, plotter or printing device can be used to make such plots. The problem is divided into three stages. The first is data entry where a geographical reference is recorded for each possible data item to be plotted. For simple two-dimensional maps this will usually be a horizontal and vertical offset onto a plane. These are usually referred to as rectangular coordinates. The values of the offsets can be based on absolute coordinates, such as the latitude and longitude system in common use for geographical notation, or in terms of a custom scale that the user devises for a particular situation, which might be measured in terms of feet, metres, miles or kilometres from a particular reference point. Absolute coordinate systems are the easiest to work with given most existing computer software.

Another absolute coordinate notation sometimes used is a polar coordinate. A polar coordinate is denoted by a distance r and an angle q. The basic idea is that a point can be defined relative to some origin point by specifying a distance from the origin point and an angle relative to the coordinate axes. Polar coordinates can be converted to rectangular coordinates using the formulae:

$$x = r \sin q$$
$$y = r \cos q$$

Polar coordinates are useful in a number of situations. They can be used as a simple form of manual point entry where points are being entered from a traditional map, especially a map which has not been laid out on a labelled grid. Rather than drawing a grid over the map and then looking up xy pairs from the grid, the points can be located by selecting an origin point, and using a protractor and a ruler the distance and angle of each point to be recorded can be measured, providing the polar coordinate for these points. Polar coordinates can be directly plotted by some programs, or the xy pairs can be calculated using the above formulae (using a spreadsheet calculator).

Polar coordinates are also useful when the map area is large enough for the curvature of the earth to be a potential problem, although this is usually not the case for anthropological maps. Polar coordinates are easier to project.

A three-dimensional coordinate system is necessary when altitude must be recorded. While xyz coordinates can sometimes be used, this is often cumbersome both to collect and to plot, since only the more advanced programs usually have this capability and there are fewer generic plotting utilities which can utilize the third dimension. It is also difficult to represent the third dimension on a map, which is conventionally viewed from directly above.

A common solution to this problem is to use topographic notation. This is basically a map which has regions of elevation marked in some manner. Often this takes the form of concentric regions, where each division represents a particular elevation. It may also be done by using shading as a marker for elevation. If the original map is marked out in this way the *z* coordinate can be a reference to the altitude band. If the input map coordinates are recorded with explicit *z* coordinates, the altitude bands can be drawn to facilitate representation of this dimension. Often a simpler solution is to mark up the representation of the map with the altitude bands, and then to record only the *xy* pairs. This can be useful if the only objective is to view the distribution of points over an area, since the elevation can be determined visually. If actual references to the altitude components are required, then the *z* coordinate must usually be coded with the data.

If the map is digitized, it is usually possible to indicate the xy coordinate for each point by either using a GIS application which includes this function or using a CAD or draw program and recording the *xy* coordinate in association with a unique reference to the point. These references might be house numbers, plot identification numbers or some other naming convention which might be used to identify the object to be plotted.

Once the coordinate data is entered there must be a mechanism for selecting points to plot. Again GIS programs usually include this function, since these usually contain specialized database operations for this purpose. Most database programs, either flat file or relational, can be used both to select objects for plotting based on data values in the record and to produce a list of *xy* coordinate pairs and other information that might be used to produce the plot.

Once the points to be plotted are selected, then they must be plotted, either by the specialized GIS program or by some other plotting utility.

To illustrate this process consider the following example in Prolog (see Chapter 7):

```
mapref(abdul,12,14). /* mapref(name,x,y). A definition is required
for each name */
property(abdul,'#'). /* property(name, symbol). One is required for
each name */
plot_property(Name) :-
property(Name,Symbol),
mapref(Name,Xcord,Ycord),
plot_name(Symbol,Xcord,Ycord),/* user supplied plotting routine*/
fail.
plot_property(_).
```

Another program, which plots all the kin of an ego could be written as follows:

```
mapref(abdul, 12,24). /* this is a data description. One is required for
each ego */
/* using the is_kin function defined in chapter 7 */
```

```
plot_kin(X) :- /* plot kin of X*/
is_kin(Name,Rel,X),
mapref(Name,Xcord,Ycord),
plot_name(Rel,Xcord,Ycord),
fail.
plot_kin(_).
```

5.4.3 Measuring area

Many programs can compute the area of a selected region of a map. This is quite useful for assisting the study of land use. Using measurements from accurate maps fairly good estimates of land holdings can be calculated, distances between areas can be determined and, by plotting against other features such as different sorts of land, quantities of different sorts of land can be estimated.

5.4.4 Overlaying

There are several kinds of overlay techniques used with maps. A common one is to overlay a vector-based map with a facsimile of a map derived from a conventional or satellite photograph. Photographs, though they represent a 'real' terrain, are often difficult to use as maps. Maps are a specific kind of means of representing spatial relationships, and rather than appearing 'real' these models of an area relate a clear picture of some of the relationships present, at the expense of some detail which must be lost to accomplish this. Although some researchers view an ideal map as a photorealistic image of the area itself, this may or may not be considered a map, and experience suggests that an image is less informative about the spatial relationships that a map is designed to relate. However, maps cannot represent most of the physical information that a photographic image can. By integrating the two forms of representation using a computer the best features of both can be achieved.

There are some significant problems with this integration. Vector-based maps are resizable, since the shapes and relationships are recorded by designating line segments. Photorealistic images can be resized to some extent, but there are currently severe problems with storage. There are programs which can do a good job of reducing a base image to some arbitrary size, especially if it is stored in greyscale or RGB colour. However, to permit much flexibility the image must be at a very small scale. This can result in a prohibitively large storage size for a single image which is expensive in both time and storage space to enable resizing.

Another strategy is to store a number of images at different scales, which is less flexible than resizing a single image but is more manageable with contemporary hardware and software. The principal problem here is the amount of management required to organize the images.

5.5 HARDWARE

The basic hardware requirements for a system capable of creating, storing and manipulating images are as follows:

1 *Any medium to high-end computer.* This should not pose a problem, since key hardware will not be available for a completely unsuitable computer. In general, however, the faster the machine is and the more memory it has, the better. Images are huge, even by computer standards, and manipulating images requires a large number of operations per pixel. Unless you are working with simple outline drawn images, the absolute minimum memory capacity should be 4 megabytes (1 megabyte=2^{20} bytes or 1,048,576 bytes of memory), and 8–32 megabytes will be useful.
2 *Display hardware and matching monitor.* There are a large number of factors to consider in terms of the size of the image and the amount of colour detail possible. Most computers include some sort of display hardware, but unless it was ordered for the specific purpose of working with photorealistic images, more is likely to be necessary.

The typology for display hardware is basically the same as for images. Ideally, the image format need not be identical to the hardware format: the hardware should attempt to display the image, even if the format of the image has a higher depth than the hardware is capable of. Images with a 24-bit colour format should be displayed on 8-bit indexed hardware, and vice versa. If the hardware cannot do this, then a software package should be available for conversion between the two formats.

For research purposes, the ideal display hardware which should be considered is as follows:

1 8-bit or 256-level greyscale if your requirements do not require colour. Use of greyscale instead of colour greatly reduces the cost of the system, and a slower system can perform well because of lessened requirements. However, reasonable results for some material can be accomplished with as few as sixteen grey levels, especially in field conditions.
2 8-bit indexed colour if your requirements do not require precise colour and maximum detail. Also, on most 8-bit hardware only one image can be shown properly at a time, since a palette is specific to each image (to achieve acceptable results) and the hardware can use only one palette at a time. Other images on the screen remain, but take on a grotesque appearance. It is possible that improvements in hardware might reduce this restriction.
3 24-bit colour if high quality is essential, or if more than one image must be displayed on the screen at one time. Since the hardware does not require a palette, a number of 8-bit images can be displayed simultaneously.

Monitors vary in terms of colour capability, resolution and size. Obviously, if the application requires colour, a colour monitor is required. There is no distinction

between monitors for 8-bit and 24-bit colour display hardware. There are good and bad monitors. Important points to consider are as follows:

1 How bright is the monitor relative to the lighting conditions in which it will be used? Are all colours (or shades of grey) equally bright? Do you have to turn the brightness up all the way, or is there reserve? Is the contrast good? How adjustable is the contrast?
2 Is the picture badly distorted or out of focus at the corners and edges? (All displays have this problem to some degree.)
3 View a test image made from a negative you own. Compare a photograph from the negative with the display. If your computing services cannot make a test image, there are a large number of graphic arts services who will do so for a small fee. For colour monitors, does 'white' look white, 'red' look red etc.? Most greyscale monitors actually have a 'white' which is blue or yellow. Is this noticeable?

In general larger monitors are better than smaller ones, but the costs rise rapidly. The main advantage of a larger monitor is that you can have more images on the screen at one time, an important consideration in comparative work.

If you want to take images from slides or negatives, there are specialized slide scanners available, which can be quite expensive but give good results. At present all of these require considerable power, although this power can be supplied in the field from an inverter circuit.

Digitizing refers to techniques of entering graphical information. There are several different methods. As a last resort this can be done by hand by simply entering coordinates for the data points. It is much easier to use a special purpose peripheral for this.

Digitizing tablets Digitizing tablets usually consist of a flat surface and a pointing device such as a mouse/puck or pen. The surface contains a number of receptors that pick up signals from the pointing device and then report the location via an interface port, either on a continuous basis or when the user presses a button. This is usually done with precision; most common tablets have a resolution of 0.1mm. They come in a variety of sizes, ranging from 6 inches by 9 inches to 6 feet by 4 feet and even larger. The smaller varieties at the time of publication cost £300 or more.

There are a variety of formats that the digitizing tablet may use to report the coordinates to the computer. Most have the capacity to send in a form which is identical to that which a human user would input using a keyboard. This makes it reasonably easy to get points in a human readable form, which can be transformed by a program for input to different programs. There are also usually more efficient methods of encoding the information, which require specialized software to decode. There are standards of a sort, and many programs have been written to use specific digitizing tablets.

CAD programs usually work from digitizing tablets, and make the task of entering coordinates much easier. Once the coordinates have been entered the CAD program has facilities to edit the coordinates, which usually include operations for measuring the distance between points, calculating the area of

enclosed segments and various other useful things. CAD packages are quite reasonable tools with which to draw plans of sites, dwellings or artefacts. One advantage of this approach of entering coordinate data is that it can usually be made larger or smaller without a loss of detail. CAD programs can usually make a printed copy of the work on a conventional dot-matrix printer or a pen plotter, or even a laser printer.

Image digitizer Unlike a digitizing tablet, which records the coordinate information, image digitizers translate an image of the object into data about the brightness or darkness and colour of a large number of points into which the image sensing device is divided. By reproducing this configuration of points on a video screen or printer, the image is reproduced. An advantage of this approach is that it is relatively quick, indeed it is possible to do many images per second. Depending on the capabilities of the hardware, a very detailed image can be captured, with better hardware the detail approaches that of a printed photograph. The disadvantages are that the image, at least as captured, is simply a collection of points, with no detailed information about the structure of these points. With a digitizing tablet, an image is drawn as a series of polygons, which have an independent identity as a set of points that can be addressed as a unit. The image, as received by an image digitizer, has no such sub-units. Because of this it is difficult to change the size of the image without losing detail. There is software which can aid in the translation of the digitized image into an object orientation, although it usually requires a great deal of interaction with an operator.

A digitized image is useful for a number of purposes. It can be measured and compared with other images, with resulting images consisting of common or different aspects of the two. With the use of a program, different aspects can be emphasized or diminished (computer-aided tomography (CAT)). These are used in medical facilities to make recordings of body segments. These images are usually three dimensional, with greyscale information (the intensity of the image point). The resolution of a CAT scan is usually 128 x 128 x 128 with 256 levels of grey from black to white. It is expected that this resolution will increase to 256 x 256 x 256 in the near future. Although to microcomputer owners accustomed to 640 x 480 resolution this may seem rather low, the greyscale information makes the display appear detailed. Although at this time CAT scanners are prohibitively expensive for most anthropologists, it is possible to use excess capacity to record images. The resulting images are recorded on magnetic tape, which can be read by most central computing services (Khanna 1988). With CAT scan data, slices of the image can be examined, density information recovered and area and volume measurements made, although there is currently little commercially available software (at an affordable cost) for this purpose.

For further discussion of video and photographic image acquisition see Section 3.3.1.

Chapter 6
Kinship applications

6.1
INTRODUCTION

The principal goal of this chapter, using the familiar territory of kinship, is to outline how to specify your research requirements in a form suitable for matching up with available computing resources by a research assistant, a computing-centre advisor or yourself. For reasons of space and because of the requirements of research discussed in Section 1.1, a full range of possible techniques cannot be covered. Chapter 7 implements simple, but useful, programs, using the computer programming language Prolog, based on some of the specifications described in this chapter.

There are a number of reasons to introduce computer-based methods of analysis using kinship material. Not least the familiarity of the subject matter: the study of kinship is the greatest common denominator across the different fractions of social anthropology. In general, an anthropologist will begin with a set of models, ideas and representations which, if not agreed upon, are at least familiar: '[kinship] appears to be the one area of anthropological discourse where the ground rules are clearly laid down' (Barnard and Good 1984: 2). Unfamiliar methodology is more easily presented using familiar content. Kinship studies, too, satisfy (and are easily shown to satisfy) the requirements for computer-based analysis. Most computing methods require relatively clear schema for successful application, and many of the models and methods anthropologists have used for analysing kinship meet this criterium.

Certainly kinship-related computer applications represent the earliest efforts by anthropologists to use computers in research (Kunstadter *et al.* 1963; Coult and Randolph 1965; Gilbert and Hammel 1966; Hackenberg 1967). Anthropologists have shown considerable interest in the use of computers for analysing kinship and genealogical data. Everyone seems to have 'too much' kinship information and computers have been suggested as a means to make this intractable quantity of data do some work (Gilbert 1971: 135–7).

Just as kinship is one of the most complex areas of social anthropology, the computing resources required to assist in the analysis of kinship data are relatively complex and varied. Anthropologists have used computer-based methods in:

1 creating and maintaining databases;
2 analysing genealogical data or models in connection with other data or models;
3 presentation of kinship and relational diagrams.

Computer-based databases of relationships have been developed to draw out specific examples of relationships or to establish relationships between two or more people. This has been an application on the wish-list of many anthropologists for some time (Coult and Randolph 1965; Gilbert 1971; Chagnon 1974). There have been significant difficulties; issues include constraints on the size of the database (how many people and relationships can be represented) and how to specify the kinds of relationships in which you are interested without being 'swamped with types of relationships which do not interest' you (Gilbert 1971:136).

Computers have been used to help analyse genealogical data or models in context with other data or models. Relationship databases support further analysis based on other variables associated with the group or unit under study. This may be further supported by computer-generated diagrams (Section 6.5) or incorporated with simulation modelling (Section 8.1). Surprisingly, there has been relatively little published work (but see Davis 1987; Ottenheimer 1988; Bagg 1991; Read and Behrens 1992; White and Jorion 1992) or even speculation (Gilbert 1971; Chagnon 1974; Davis 1984b). A notable exception has been the use of computer-based modelling to examine how demographic structure influences marriage behaviour and vice versa (Kunstadter *et al.* 1963; Randolph and Coult 1968; Hammel *et al.* 1979; Dyke 1986). It may be that the apparent difficulty intimated in the past (Findler and McKinzie 1969; Gilbert 1971:137) with using computers for flexibly analysing genealogical data has limited speculation in this area.

Computers have also also been sparsely used for theoretical work in kinship, either with terminologies or structure. Examples of such work include Read and Behren's (1989,1992) kinship algebra expert system, Ottenheimer's (1988) work on modelling, applications derived from graph theory (Garbett 1980; Hage and Harary 1991) and statistical studies aiming to develop behavioural correlations.

A number of programs have been written by anthropologists which can assist in the preparation of kinship diagrams for most computers (Ryan 1985; Ottenheimer 1985, 1988). Two basic approaches to automatic diagram support have been developed: theoretically motivated diagrams present principles of kinship structure using imaginary individuals and actual kinship diagrams present examples of relationships and diagrams between the people in an area of study. It is easier to present an idealized kinship chart than to deal with actual populations. Besides the more obvious problems of deviance from conventions, there are problems of different kin cluster sizes, sheer quantity of people and deciding precisely what kinds of relationships to diagram. By diagramming actual people in actual relationships, we are introducing both mechanical and conceptual problems. For example, in a physical layout it is difficult to present both the spouse and sibling relationship clearly in the same diagram. This is true in non-computer-generated diagrams as well, and the ordinary solution

has been to introduce only relationships significant to the current discussion. (This is the computer-based solution as well.) A mechanical problem is solved conceptually. Unfortunately, the reverse rarely proves true: conceptual problems are not solved by mechanical means, even when the machine is a computer. A clear conceptual model of the material is necessary for any analysis traditionally or computer based. The first step in computer-assisted analysis is a clear development of the structural scheme we apply to a body of data.

6.2
DEFINING CONCEPTUAL REQUIREMENTS

As discussed in Section 2.2, we must first define the conceptual terms to which we intend to apply a computer-based analytic procedure. A 'structural schema' is a set of objects, their properties and the relationships between the objects and properties. The conceptual terms must be determined, in whole, by scientific requirements rather than computing requirements. The structure and definition of conceptual terms are independent of whether or not a computer is to be used. It may be the case that an external schema which is adequate to represent a given conceptual schema will be difficult to implement using existing or custom purpose-designed computer applications. Using a computer may not be useful in analysing that domain. Computers are not applicable to all problems in part or whole.

Most simple-mindedly, anthropologists do anthropology, and they do it using conceptual schemes determined by disciplinary conventions. The first consequence of this is that anthropological structural schema take precedence over computer structural schema. If the fit is good, the anthropologist has an efficient and useful tool. The second consequence is that the anthropologist needs at least enough appreciation of computer structural schema to recognize problems (and felicities) in the fit of the two schema. In the domain of kinship, anthropologists have traditionally conceived a schema which can be usefully and productively represented on a computer. The applications discussed in Section 6.1 mainly use existing schema associated with genealogical modelling in social anthropology (Barnard and Good 1984: 23–6).

The structural aspects of the genealogical model are relatively uniform; we record the genealogical connections between individuals as a means of describing some aspect of social relations which has been of value in social anthropology for describing social structure. The conceptual schema influences the collection and interpretation of the structural model which results from establishing the genealogical connections.

A computer application may provide a generalized record-keeping and relationship-establishing function common to all sorts of schemata. An anthropologist may simply want to establish links between individuals in a population using defined kin categories. This will influence and simplify preparation of reports, error checks and audit functions. This is a generic function of the computer, and the computer support functions required are limited.

However, a computing application may include functions which are specifically related to the conceptual schema intended. If there are special interpretations or assumptions included in the anthropological investigation of the material, specific support requirements in the computer applications may be required. For example, if an application is simply required to establish links between individuals in a population, the support requirements for any particular schemata are limited. If we wish to perform an analysis of a more specialized sort, such as calculating inbreeding coefficients on a purported biological pedigree, specific support requirements are required.

For this reason it is usually best to define requirements in terms of generic functions and specialized functions. All forms of genealogical analysis require the establishment of different sorts of links between individuals, so this is a generic function. Very few analytic models require that these links be defined as biological links, as is required to legitimately calculate inbreeding coefficients. However, we do not necessarily require two independent applications for these two purposes. One generic application can calculate the genealogical connections and the results input to another more specialized application which assigns the specialized interpretations and operations on the proposed genealogical structure.

In terms of a genealogical study, the conceptual schema would consist of definitions of people, the kinds of links recorded and maintained and associated theoretical statements about how these interact or are defined in terms of each other. Barnard and Good present one conceptual schema based on common genealogical models. This consists of primitive objects, people, a set of properties of people (sex, relative age), a primitive set of relationships between people (F, M, B, Z, S, D, H, W) and a set of rules for building compound relationships (FF, MM, FB, MB etc.) based on primitive relationships via other people (Barnard and Good 1984: 3–9).

This might appear to be a great deal of detail, but it is nothing more than any anthropologist applies to a genealogical model. However, in ordinary discourse much of this detail can be left unstated simply because it is so conventional. Computer programs will, however, require some form of this knowledge of conventions to be incorporated into the program. It is important to be able to specify this detail at the conceptual level, especially if you are attempting to specify your problem to a programmer. One particularly problematic aspect of genealogy programs and computer programmers (who are not also anthropologists) is that the unsaid detail will often be inferred by the programmer since ‘every one’ understands genealogies and family trees and such. Their inferences, derived from their particular genealogical models, will rarely match your requirements.

Even if you intend to write the program yourself, it is a good idea to record this information. It will provide a base to which you can compare the eventual computer implementation of the schema. As with other analytic methods, there will be simplifications and compromises, and it is useful to identify explicitly where there are, since deviations from the conceptual schema impose limitations for interpretation of the results.

In specifying the conceptual schema you can use formal looking statements and diagrams, such as in Figure 6.1. This can be rather difficult and cumbersome

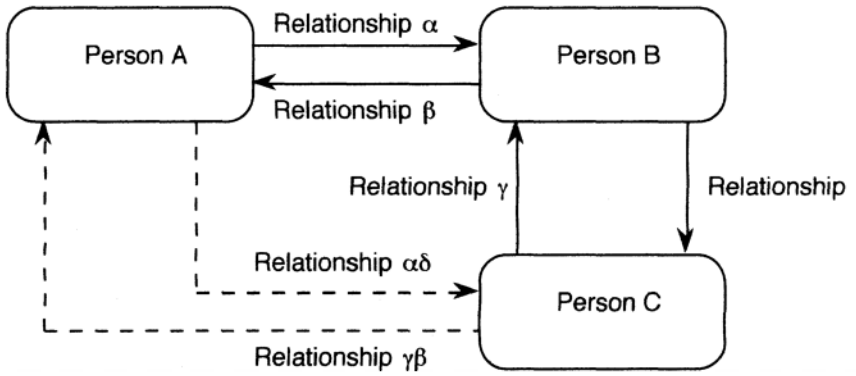


Figure 6.1 Abstract conceptual schema for relationships

for complex definitions, and is not strictly necessary. Many people will be more comfortable clearly writing the

Table 6.1 Informal specification of conceptual schema for genealogical model

I	The object of the schema is to model people and relationships between people. People will be represented by a set of properties, specifically, each person is an individual entity with properties of gender, age and generation.
II	Relations between people are of two sorts.
A	Basic relationships.
(i)	Basic relationships correspond to Parent, Child, Sibling and Spouse, and where necessary, the gender-marked relationships which correspond to the categories Father, Mother, Son, Daughter, Brother, Sister, Husband and Wife, relative to a given individual denoted as Ego. These relationships are defined in terms of indigenous social judgments.
(ii)	Each relationship has a reciprocal relationship; e.g. if an individual has a relationship Father relative to an Ego, Ego has the reciprocal relationship Son or Daughter to that individual. This reciprocal relationship is different in the case of Parent–Child, is the same in the case of Sibling–Sibling or Spouse–Spouse, is different in the more gender-marked forms if the genders are different and may be different in the case of Siblings if relative age is different (and significant).
B	Compound relationships.
(i)	Compound relationships are established between people by at least one common intermediate relationship, which itself may be compound. Compound relationships are sequences of basic relationships which relate one person to the other. There may be more than one compound relationship between any two individuals.
(ii)	Reciprocals for compound relationships are the sequence of reciprocal relationships between the same people.

schema using their native language with examples of each type of definition or rule and a few simple diagrams where useful (Table 6.1). As long as the specification is clear, relatively complete and relatively unambiguous, just about any form will be satisfactory and useful.

Table 6.1 presents an example of informal specifications for a genealogical model. This model will be customized for the group involved: in my own work, birth order, multiple marriages and other information are significant in analysis. This information will be included in your informal specifications.

In summary, theoretical characterization of the research problem defines conceptual schema. If scientific representation of the area under study appears to be adequately represented in the computer, we outline precisely what general and specific functions a computer can perform in the analysis of the material. If we decide that computer-based analysis is possible and useful, we translate our information into a form computers can use (programs) or computer programmers can understand (detailed specifications), or look for commercially available programs.

In the next section, we shall discuss more completely how this is done, and the advantages and disadvantages to each approach. (The optimistic might think that simply buying a software package is easy. It is not.)

6.3 MODELLING GENEALOGICAL LINKS BETWEEN PEOPLE

The modelling of genealogies is an ideal task for computer assistance. Genealogical applications are among the earliest applications of computers to ethnographic data attempted by anthropologists, dating from Kunstadter's work on the interaction of demographic structure and marriage preference (Kunstadter *et al.* 1963) and more explicitly by Coult and Randolph's discussion of computer-based methods and models of genealogical space (Coult and Randolph 1965) and their further application of these to problems of Bedouin kinship and social organization (Randolph and Coult 1968).

Despite this initial interest, there is no generally available computer application for modelling given genealogical links in 'real' populations. There is commercial software for assisting with the compilation of 'family trees', but these generally fall into the domain of programs for data entry and retrieval. Where some calculation of relationships is performed, these are under assumptions which are unacceptable to most anthropologists. Anthropologists have written programs for this purpose, but most of these are unpublished and many of these contain assumptions which are specific to their field material, such as tracing only patrilineal links. Ryan (1988) has written a program which determines links in a very general and configurable manner in conjunction with genealogical diagramming (Section 6.5), but it is not well suited to large populations (greater than 300) due to constraints of diagramming. Findler and McKinzie (1969, 1970) developed some programs for tracing genealogical links, but there are some serious problems with using them on non-ideal data which contain ambiguity or missing information, multiple spouses or missing parents. Gilbert (1971), who wrote the programs used by himself and Hammel in their early work on demographic interactions with marriage choice (Gilbert and Hammel 1966), offers some useful advice and speculation on the subject, but concludes that the problem is rather difficult for a general solution (Gilbert 1971:

136); there is no one way to reckon genealogical relations which will satisfy the varying research needs of anthropologists. Instead, he suggests that in order to gain the flexibility anthropologists require they must become programmers (ibid.: 137).

6.3.1
Specifying the genealogical 'engine'

Most of this prior work has employed the genealogical 'engine' in conjunction with other computer-based analytic components. There are, however, occasions where the simple ability to establish the kinship links between individuals or to find the individuals who stand in a specific relationship or set of relationships to ego would be useful in the context of manual analysis. There are also potential computer applications which would be useful in the collection and maintenance of genealogical information in the field (Weinberg and Weinberg 1972; Chagnon 1974; Davis 1984b). For example:

- translating an ego specific genealogy to a general, normalized, format;
- consolidating two or more egocentric genealogies to a normalized form;
- checking consistency of two or more egocentric genealogies which overlap;
- figuring genealogical links using different descent criteria;
- selectively including, excluding or limiting different kinds of links in terms of generation, affinity or collaterality;
- evaluating alternative structures based on ambiguous data, such as vital events records;
- translating terminological notation to 'etic' notation and vice versa.

Each of these auxiliary operations depend on the capacity to establish links between individuals. As suggested by Gilbert (1971: 135–7), there is no general 'solution' to the problem of establishing links, simply because genealogical analysis is based on a methodology, not a procedure. The method has changed over time to reflect the experience and criticism of anthropologists to meet research requirements. So although much of the original method set out by Rivers in *Notes and Queries* (1912) has been retained, there have been considerable modifications such as those suggested by Barnard and Good (1984: 30–3). Basically, attempting to define a computer-based medium in which to work with the genealogical method, with the ability to change, extend or remove specific kinds of genealogical links, requires a level of flexibility which is only easily attainable at the level of a computer programming language.

To develop a program, to be written either by yourself or by your programming partner, which can accomplish the task of establishing genealogical links we must first produce a specification of the program. A program specification is one of the following:

1 a fairly explicit description of the task at the conceptual level (Section 6.2). This establishes the purpose of the program and describes the conceptual universe of the program; what the various terms and elements in the

eventual program will refer to. Following the conceptual description in Table 6.1, we can define that the purpose of our program is to establish a list of people who stand in one or more genealogical 'positions' relative to a specified ego. The conceptual schema relates to people and the kinship relationships they recognize as existing at some moment in time and space. Moreover, we are positing a genealogical space within which we can establish an etic position for each person, which will serve as the basis for evaluating the emic judgments of indigenous consultants with respect to the operation of kinship in different circumstances.

2 An explicit description of a model. This includes the following:

(a) a description of the data structures available for processing which will correspond to the initial data stored in the program or data to be input to the program by the user or from a computer data file (initial data model or input data model). This is not necessarily the structure of the data you have collected, but must, of course, be compatible with your data (see Section 6.4). The example specification in Section 6.3.3 assumes data structures which link people with the relationships Child and Spouse will be available, along with the Gender of each person. The relationship between this abstract data model and the data which is input to a program is discussed in Section 6.4.

(b) A description of how the model terms not included in the initial data will be derived. For the example specification in Section 6.3.3 this will include the derivation of the basic genealogical relationships (F, M, Z etc.) from the primitive relationships Child, Spouse and Gender, and compound relationships such as FB or MZ from these primitive relationships.

(c) A description of the procedures which manipulate the initial or derived structures to achieve the purpose of the program. This includes specifying the initial condition and initialization of the program (reading in the data from a computer file, for example) and how to recognize that the objective has been accomplished.

6.3.2 Specification models

There are many different forms a specification can take and different models are used for specification (Burnard 1987). Specifying a program is basically the task of explaining exactly what a program must do to accomplish the research objectives required of it. Specifications for computing applications are usually a conjunction of assertions in the form of data, ethnographic, imaginary or simulated, and operations on that data. A number of different programs can satisfy a single computing applications specification. If properly written, however, the specifications for computing applications can form a basis for specifications for program applications.

For the present purpose we shall use an informal version of the interpreted predicate logic model (ibid.: 66; Kowalski 1984). This model translates well into

rules of the sort that are common and explicable to anthropologists (though sometimes much criticized). These sorts of rules are familiar to both anthropologists and their programming partners, and are relatively easy to implement in existing computer programming languages and relational database languages, especially the programming language Prolog which is used to implement these specifications as programs in Chapter 7.

The anthropologist provides the material for analysis and the program which performs the analysis. Specifying research objectives required of the program can only be done by the person who possesses such knowledge. Many a project has floundered because it was left to the unfortunate programming partner to make crucial decisions about a problem area of which they had only a marginal understanding.

The main problem inexperienced computer users have with specifying a program is the level of detail required. For example, in defining a basic relationship between two people chosen from the population (e.g. the data for a population input to the program) we might write:

If A is Parent of B then B is Child of A.

where A and B represent some two people from the population. Many people would be satisfied with this as a valid specification of the link between Parent and Child. But it is not logically complete relative to our conceptual schema in which the only case where B can be derived as a Child of A is when A is Parent of B. We have specified one of a possible many conditions for when B is a Child of A. To make this exclusive we can revise this statement to:

Iff A is Parent of B then B is Child of A.

where 'iff' is read 'if and only if'. This statement is now logically complete relative to our intention.

There is a 'positive' bias in programs which is commonly used to imply closure by not supplying data or rules which violate closure. Thus at the level of the program, if the only rule implemented for deriving Child is:

If A is Parent of B then B is Child of A.

then exclusivity is implicit. It is still best at the level of specification to indicate the exclusiveness to differentiate these cases from those where more than one specification rule has the same conclusion, such as:

If A is Father of B then B is Child of A.
If A is Mother of B then B is Child of A.

6.3.3
A first specification

In making the specification you write down the set of rules and data available which are consistent with your goals in an analysis and which do not 'compromise' the meaning and interpretations you intend. You should note the

implied assumptions (iff) and identify terms which will be defined from initial data and terms which will be derived from these.

For convenience of exposition (and to avoid too many abstract symbols) the following example specification uses the English terms (without italics) for basic genealogical relationships, and upper case letters for variables which can denote any person from the population to be examined. Ego is used to denote the person to whom the derived relationship is relative. Ego ranges over the same values as variables denoted by upper case letters.

I The purpose of the program is to establish a list of people who stand in one or more genealogical 'positions' relative to a specified ego. The conceptual schema underlying the program intends to denote people and the kinship relationships they recognize as existing at some moment in time and space. The model also includes a genealogical space within which an etic position is established for each person.

II Data will be supplied for Gender, which ranges over the values male and female, and the relations Child and Spouse. The basic genealogical terms Father, Mother, Son, Daughter, Brother, Sister, Husband and Wife will be derived from these relations. More complex relations will be derived from these derived basic genealogical terms.

iff Ego is a Child of B then B is Parent of Ego.

iff A is a Child of Ego and A is male then A is Son of Ego.

iff A is a Child of Ego and A is female then A is Daughter of Ego.

iff A is a Parent of Ego and A is female then A is Mother of Ego.

iff A is a Parent of Ego and A is male then A is Father of Ego.

For the relation sibling we could try the following:

iff A is Father of Ego and B is Mother of Ego and A is Father of C and B is Mother of C then C is Sibling of Ego.

However, this specification of Sibling is not complete. When we are using variable terms, such as A, B, C, Ego above, whose values are drawn from the set of people under consideration, there is nothing to prevent the same person being assigned to two (or more) variables. This is not a problem for relationships which are not of the same type, but this definition for Sibling will result in a program which will agree that Ego is Sibling of Ego. To correct this, use the amended statement:

iff C is not Ego and A is Father of Ego and B is Mother of Ego and A is Father of C and B is Mother of C then C is Sibling of Ego.

iff A is Sibling of Ego and A is male then A is Brother of Ego.

iff A is Sibling of Ego and A is female then A is Sister of Ego.

iff A is Spouse of Ego and A is male then A is Husband of Ego.

iff A is Spouse of Ego and A is female then A is Wife of Ego.

This specifies a means for deriving the basic etic links from a given data model, which might seem quite a bit of work just to specify what was known already. Except for fairly specialized analyses, e.g. comparing the household structure of large populations, we would be unlikely to employ a computer just to represent the basic 'nuclear' relationships. The point of the specification so far is simply to provide the basic knowledge required to derive more distant genealogical links, which are not known or known only for a fraction of the population.We continue along the same lines as before:

iff A is Father of Ego and B is Father of A then B is Father's Father of Ego.

iff A is Father of Ego and B is Mother of A then B is Father's Mother of Ego.

iff A is Mother of Ego and B is Father of A then B is Mother's Father of Ego.

iff A is Mother of Ego and B is Mother of A then B is Mother's Mother of Ego.

iff A is Father of Ego and B is Brother of A then B is Father's Brother of Ego.

iff A is Father of Ego and B is Sister of A then B is Father's Sister of Ego.

iff A is Mother of Ego and B is Brother of A then B is Mother's Brother of Ego.

iff A is Mother of Ego and B is Sister of A then B is Mother's Sister of Ego.

…

iff A is Mother of Ego and B is Mother of A and C is Sister of B then C is Mother's Mother's Sister of Ego.

…

Although this can become rather tedious (computer applications are often tedious), it is one simple method to specify precisely the relationships and links you need to represent and to ignore those with which you are not interested. Only relationships for which specifications exist will be identified. The exact extent of tracing relationships through the dimensions of generation, collaterality and affinity can be controlled. In situations where you are interested in more complex and far-ranging relationships more abstract specifications can be written.

III A procedure to list the set of kin corresponding to the definitions set out in II, relative to a given, ego can proceed as follows:

Initialize the data specified in II for Gender, Child and Spouse relationships for a specific population of people.

Set Ego to a specific person value.

For each specification rule in II, except Child, Parent, Sibling and Spouse, trace the links in the rule until it is either impossible to continue with the rule (no person exists in the population for a specific genealogical position relative to some link in the rule) OR all the links in the rule are found to exist between the specific people in the population. If the rule is satisfied store the following information in a list: the derived relationship in the rule (e.g. Father's Brother) and a list of the people who satisfied the rule (e.g. 'Abdul, Mustaffa' for the Father's Brother rule if Abdul is the Father and Mustaffa is the Brother of Abdul). If the rule can apply more than once (there may be more than one Brother, Sister, Wife, Husband etc.) then continue until it can no longer be satisfied.

After all rules have been applied, output the list of stored people and relationships (on a computer terminal or perhaps to a computer data file).

6.3.4
Problems with specifications to be applied to data

The (quite informal) specification in Section 6.3.3 can be translated into a computer program relatively easily after some discussion with a programming partner, since plenty of detail is assumed. It has some flexibility for extensions. For example, modifications to which relationships should be reported can be easily effected. However, this specification is tied not only to a particular conceptual model, but also to a particular view or model of the data that might be available to it.

Although another specification and program can be devised to reformulate a number of varieties of 'real' data to the abstract data format of the example, the abstract model must be consistent with our objective: the description of kinship relations within a (finite) population.

Although this specification is generally adequate for representing genealogical links within some ideal populations, there are some practical and logistical problems with its application to most data derived from ethnographic collections.

1 Genealogical data is often incomplete, and necessarily so because of the 'horizon' of a finite population. For example, we specified that siblings were people who shared two parents. This is fine so long as we never have the situation where one or both parents fail to be recorded. This is, of course, inevitable, because a genealogy must stop somewhere, and this is unlikely to be at a point where every person is an 'only child'.
2 Kinship data is not usually collected in the format set out in the specification. The method recommended by Barnard and Good (1984: 26–33) specifies a minimum set of thirty-nine relationships to be elicited from a

single consultant. Most ethnographers do not try the patience of their consultants by recording genealogical information from each member of a family or household unit unless there is a compelling reason to do so. It is therefore necessary to derive genealogies for the non-consultants who appear in genealogies from the collected genealogies.

3 The specification presented in Section 6.3.4 is not adequate to deal with many kinds of genealogical data which anthropologists might need to represent. There is, for example, no reference to relative birth order of the children of a union, the order of spousal unions, nor any means of referring to the status of a particular union (e.g. married, divorced, widowed, informal, adulterous). These can be corrected fairly easily by adding these attributes to the specification. However, more serious problems emerge, for example, when more than one union has occurred between the same couple, and this is an emic factor in indigenous judgments.

4 Where the genealogical data is gathered from documents, or even from consultants, there are problems in reliably establishing the identity and details of some (or all) of the people indicated in the data. In this case it may be necessary to allow more flexible evaluation rules to suggest contingent candidates for links. The specification in Section 6.3.4 has no provision for contingent data.

Addressing these issues is not insurmountable, although (4) is intrinsically problematic. Incomplete data from 'horizon' effects is often resolved by linking to imaginary people who are not part of the data set. Other solutions have required that bogus records be inserted in the data set in order to maintain the integrity of linkages. The relationship between the format of data required within the model *vis-à-vis* the form of the input data can be addressed by writing another program segment which translates the input data into the 'normalized' form required by the specification (Section 6.4). The inadequacy of the specification in Section 6.3.3 can be rectified by examining in greater detail the requirements of representing genealogical data, and producing a specification with greater power. The problems of using genealogical data derived from documentary sources (and other sources of contingent data) is intrinsically problematic, but can be approached by a rather more flexible and broader range of abstract data models.

6.3.5 An example

Gilbert (1971) presents a framework which suggests reasonable solutions to the three problems in Section 6.3.4. Limitations of both hardware and software at the time dictated much of this framework. Currently, expanded memory capability on inexpensive computers and the development of relational models in software offer alternative solutions to some of the problems with his approach. An extended

Table 6.2 Input data record format suggested by Gilbert

Ego	*Sex*	*Father*
	Mother	*Sibling*
	Spouse 1	*First Child*
	Spouse 2	*First Child*
	. . .	

Source: Gilbert 1971: 132

discussion of Gilbert's 1971 work presents a good example of a specification model and the interaction of hardware and software in research solutions. Current hardware and software developments suggest a different approach, one which will be discussed after Gilbert's material.

Gilbert's model required the data to be input as in Table 6.2, where, apart from Sex, all variables (e.g. Ego, Mother, First Child) are unique references to people with the appropriate relationships. He addresses the incomplete data and 'horizon' problem by pointing all 'unknown' male cases to a single computer-generated bogus record he calls 'Mr Zero' (there is also a 'Ms Zero'). What he is suggesting is more of a missing data indicator which happens to take the form of a data record with all zeros. This solution imposes no real conditions on the user preparing a data set, and all such references are fairly easy to ignore later when the genealogical database is interrogated.

Sibling refers to the next sibling in some order (usually birth), unless there is none, in which case this field refers to the first sibling in the order (which may be Ego). These references can be followed to construct the set of siblings. $Spouse_1$… $Spouse_n$ refers to the spouses, in order, that this Ego has at the moment or (possibly) in the past. There is no status (e.g. divorced, married) for the marriage, although a status variable could be added after each spouse. First Child is the reference to the first child of each union. Gilbert's data model determines the other children by following the Sibling reference. He is thus using a 'trick' of structure to encode both children and siblings and their relative order with relatively low redundancy. Each data record points to the First Child of each union, and each Ego points to the next sibling in relative order (with the last pointing to the first). With this scheme it becomes possible to recover the following additional information using the informal sketch method included.

1 Set of children of a union in relative order. Start with Eldest Child. Follow Sibling reference until Eldest Child repeats.
2 Number of children. As (1) with count.
3 Set of siblings relative to ego. Follow Sibling reference until Ego.
4 Number of siblings. As (3) with count.
5 Set of siblings in relative order. Look at Father record (if present). Find Mother (if present). Start with First Child. Follow reference, excluding Ego, until First Child repeats.
6 Elder siblings. As (5) but stop when reference is Ego.

7 Younger Siblings. As (3) but stop when Eldest Child is reached. Eldest Child is determined using method in (5).

To add a person to a database in Gilbert's format is relatively straightforward, if a little complex: Ego's data record is inserted. If Ego is the eldest child of a union, Ego is referenced on both the Father's record and the Mother's record. Otherwise Ego is referenced in the Sibling field of the next Eldest Sibling's record. If Ego is married to one or more spouses, Ego is referenced in each of these records, and visa versa.

Gilbert's model was representative of the time in which it was devised,[17] and it addresses most of the problems in Section 6.3.4 (excepting the fit of Gilbert's input data model with the collection methods of ethnographers).

The main problems with it are its complexity and fragility. It has a relatively complex structure for the data. The complexity cannot be alleviated to any significant degree; the conceptual structure the data represents is itself complex. The location of the complexity can be shifted however, from idiosyncratic structures in the format of data records to a more regular form, which will also make the structure less fragile.

The structure depends upon sibling reference, and it is relatively easy to misplace a sibling reference. A great deal of Gilbert's problem was working exclusively with what is often called a 'flat' data structure: all data relating to a case is aggregated in a single record relating to that case. Aggregated data is also more inflexible. When data is aggregated in the form of cases, it is almost impossible to disaggregate the data and reform it in new social units, which limits the manipulations possible. (Anyone who has worked with census information and tried to go from household records to family records, or vice versa, knows how very difficult this can be.)

I shall propose an alternative form of recording genealogical information which is more robust, less fragile and radically disaggregated—leading to greater ease in forming new social units with the data.

6.3.6
A more robust example

The fragileness of the referential structure can be improved, using a data model which is now relatively common and which is compatible both with the interpreted predicate logic model presented in Section 6.3.2 and with the relational model (Section 2.3.3) which is in common use for data base management. This compatibility is not coincidental, since the relational calculus upon which the relational model is based is an elaborated subset of the predicate calculus. An advantage of relations as a model of data is that data is disaggregated in a form which is easy to relate to its interpretation, and which can be present for some cases but need not be present for all. New data types (relations) can be added to an existing database with no changes being necessary (or desirable!) to data already present. The data can be structured in one form and logically specifiable expressions can be used to create different 'views' of the data, if these views are a possible interpretation of the expression when applied

Table 6.3 Representation of Gilbert's record struture

Relation Name	*Key*	*Value1*	*Value2*
Gender	Ego	{male,female,unknown}	
Spouse_Set	Ego	Spouse_Set ID	ordinal of union
Sib_Set	Ego	Parent's Spouse_Set ID	ordinal of sibling position

to the database. For example, in the example in Section 6.3.5, from the basic predicates (or relations) in the data model, Child, Spouse and Gender, a view of the data was created as the abstract data model using a number of expressions which specified the basic relationships Father, Mother, Son, Daughter, Brother, Sister, Husband and Wife.

Using this disaggregated structure for the input data model we can, for example, represent all the information in Gilbert's record structure, including the special constraints on sibling references, with the set of three relations (or predicates) shown in Table 6.3.

This schema aims to record all information from the reference of a single ego, making no direct reference to other people and instead using indirect references to a Spouse_Set using a Spouse_Set ID, which is simply a unique identifier for a particular spousal union (or other kind of union if desired), and a Sib_Set reference using the Parent's Spouse_Set ID reference. Using this scheme only data for a given ego need be inserted into the database at any particular time. Children can be found by looking for a Sib_Set record which uses one of Ego's Spouse_Set IDs. Siblings can be found independent of the presence of parents by locating all people who share Ego's Parent's Spouse_Set ID. This can be particularly useful if it is necessary for analytic or ethnographic reasons to separate out children allocated to the same parents in different unions or where the union status has changed, e.g. to differentiate children born to a union which existed before a formal recognition of union from those born after the union was recognized, and is achieved by assigning the two unions (or union statuses) different Spouse_Set IDs. The primary benefit for all cases is that bogus parent records or spouse records need not be constructed; only an ID need be assigned.

A minimum record for a given unrelated ego (in the data) would involve the input of two data elements, Ego and the Gender value for Ego. A once married Ego with parents involves eight data elements, Ego three times, the Gender of Ego, the Spouse_Set ID (a name given to this particular union) and the Parent's Spouse_Set ID (the ID of the parent union), along with ordinal indicators for the Spouse_Set and Sib_Set. Additional spouses will require three additional data elements each. Gilbert's model required only seven per case, plus two elements per additional spouse, but he was encoding both the order of spouses and the order of siblings by the position of the data records, an error prone process in

which the errors are difficult to detect. All 'horizon' cases would require the creation of bogus records to provide sibling links.

An objection that might be lodged against the newly proposed input data model is that it is not 'natural' or does not reflect ordinary analytic intuitions. There are two possible defences. First, there is analytic support for the structure, since it mirrors fairly accurately the conventions used in genealogical diagrams, establishing a set of spouses and set of siblings and a link between them. Second, it is only the input data model. It is not necessary to provide raw data in this form. Rather, the input data model must be adequate to represent the information in the raw data and compatible with the abstract data model used internally, that of the eight basic relationships.

The abstract data model of the specification in Section 6.3.4 for basic relationships could be rewritten (briefly) to incorporate this new input data schema as:

parent	iff Spouse_ID of A is Parent's Spouse_ID of Ego then A is a Parent of Ego.
father	iff A is a Parent of Ego and Gender of A is Male then A is Father of Ego.
mother	iff A is a Parent of Ego and Gender of A is Female then A is a Mother of Ego.
sibling	iff A is not Ego and Parent's Spouse_ID of A is Parent's Spouse_ID of Ego then A is a Sibling of Ego.
brother	iff A is a Sibling of Ego and Gender of A is Male then A is Brother of Ego.
sister	iff A is a Sibling of Ego and Gender of A is Female then A is Sister of Ego.
child	iff Ego is a Parent of A, A is a Child of Ego.
son	iff A is a Child of Ego and Gender of A is Male then A is a Son of Ego.
daughter	iff A is a Child of Ego and Gender of A is Female then A is a Daughter of Ego.
spouse	iff Spouse_ID of A is Spouse_ID of Ego then A is a Spouse of Ego.
husband	iff A is a Spouse of Ego and A is Male then A is a Husband of Ego.
wife	iff A is a Spouse of Ego and A is Female then A is a Wife of Ego.

These definitions are at least as 'natural' as the previous definitions, and in the new specification we can incorporate all the additional information of Gilbert's specification, as in Section 6.3.5:

1 *Set of children of a union in relative order.* Set of people where Parent's Spouse_ID of their Sib_Set is Spouse_ID of Ego's Spouse_Set in order of their Sib_Set ordinal value.
2 *Number of children.* The number of people where Parent's Spouse_ID of their Sib_Set is Spouse_ID of Ego's Spouse_Set.
3 *Set of siblings relative to ego.* Set of people with same Parent's Spouse_ID as Ego, except Ego.
4 *Number of siblings.* Number of people with same Parent's Spouse_ID as Ego, except Ego.
5 *Set of siblings in relative order.* Set of people with same Parent's Spouse_ID, except Ego in order of their Sib_Set ordinal value.
6 *Elder siblings.* Set of people with same Parent's Spouse_ID, with Sib_Set ordinal value greater than Sib_Set ordinal value of Ego.
7 *Younger Siblings.* Set of people with same Parent's Spouse_ID, with Sib_Set ordinal value less than Sib_Set ordinal value of Ego.

Comparing the descriptions specified here with those required as a consequence of Gilbert's data model, we see that these are defined entirely in terms of structural relationships between data elements and not in terms of the procedures required to identify the relationships; Gilbert's model requires a greater familiarity with the way a particular computer language works than the former, which is defined in terms of the data model which must be understood by the designer (researcher) in either case.

Another advantage of the schema is that it is fairly easy to include additional data for all or selected cases without altering the other items in the database. For example, the following information on the beginning and ending of unions can be independently included:

Union_Begin Spouse_Set ID date of union (married, informal, etc.)
Union_End Spouse_Set ID date of end (widowed, divorced, etc.)

By specifying union and its status (with marriage as one possible value) traditional anthropological issues, such as matrifocality, can be ignored or further explored. This structure provides information on each Spouse_Set, assuming that this information will be agreed by all partners in the union. If there is likely to be a difference of opinion about the status or dates, then Ego must also be included in the structure. Among the Nayar, for example, partners to the union would assign a different status to that union (Dumont 1983); this is not a business database where things must reconcile. Lack of reconciliation is one of the most interesting problems in the field, in fact, and must certainly be accommodated, e.g.

Union Ego Sid Doe Status

Diachronic comparisons are possible, if reasonably accurate information is available, and the addition of year of birth, year of death, beginning of union and end of union dates are included. The scheme can be extended with additional

information on each Ego, such as land ownership, or on other structures necessary for the particular analysis, such as clan, lineage or section attributes.

6.4 PREPARATION OF DATA FOR GENEALOGICAL PROCESSING

To begin any analysis using a computer, the data upon which the analysis is to be based must be represented in an arbitrary form which an appropriate computer application can use. Ideally, a computer would accept data in the same form in which it was originally collected. This was the speculation of Gilbert (1971), who predicted that 'soon' genealogical data might be input directly in the form of diagrams using scanners. Unfortunately, this has not come to pass (yet) and it is rarely the case that the original form of the data will resemble that used for data input.

If you intend to use an existing application, such as those by Gilbert (1971) or Ryan (1988), these applications will have specific formats in which the input data must appear. If you use the programs in Chapter 7, or you design and write your own application, each program will have some pre-existing format for the data, although you will have control over this format.

Even if you have entered your data directly into a computer, it is unlikely to be in the correct format unless you have taken the required format into consideration at the time of entry. In most cases it would not be desirable to record the data directly in such a format. The data collection model usually has quite different demands from the data analysis model, whether the analysis is to be undertaken by computer or manually. For example, if the data collection model conforms to the recommendations of Barnard and Good (1984: 30) in Table 6.4, a sample (a very orderly one at that) of the data as collected might appear as in Table 6.5. For manual or computer analysis the data must be converted from a set of extended genealogies relative to a few people, to some sort of 'normalized' form from which the data can be cast relative to an arbitrary person. The usual options for conversion are as follows:

1 Hand process the data from the collection model into the required form, and enter the data using a text editor. This most resembles the 'traditional' process of copying, cutting, pasting, indexing and sorting.
2 Hand process the data from the collection model into the required form, and use a computer program which emulates a data entry form, which will organize the entry of data and perform some simple error checks on the values entered, improving the 'correctness' of the data entry. There are a number of readily available programs available for most computers for this purpose.
3 Code and enter the data in a form close to the collection model, using a text editor or a forms program, and use an application to translate it to the required form. This, in effect, will require a customized program specified by yourself or a computing partner. This is a particularly easy type of

program to design, assuming that the collection model itself is fairly orderly and logically consistent with the required target format.

6.5
PRESENTATION: KINSHIP DIAGRAMS

When anthropologists think of a genealogical diagram drawing program they are

Table 6.4 Suggested order of procedure in genealogy collection

(1) F, (2) FP, (3) FG, (4) FGE, (5) FGC, (6) FGCE, (7) FGCC
(8) M (9) MP, (10) MG (11) MGE, (12) MGC, (13) MGCE, (14) MGCC
(15) E, (16) EP, (17) C, (18) CE, (19) CEP, (20) CC, (21) CCE, (22) CCEP
(23) EG, (24) EGE, (25) EGEP, (26) EGC, (27) EGCE
(28) G, (29) GE, (30) GEP, (31) GC, (32) GCE, (33) GCEP, (34) GCC, (35) GCCE
(36) GEG, (37) GEGE, (38) GEGC, (39) GECGE

Source: reproduced from Barnard and Good 1984: 30
Note: G, Sibling; E, Spouse; P, Parent; C, Child; F, Father; M, Mother.

Table 6.5 Sample of genealogical data in format of Barnard and Good 1984: 30

House 705 D2BIII, Consultant/Ego: Muzhar Ali Khan, Male, 27, years b. 1960, Head, engraver	
F	Kazeem Ali Khan, pio, Male, 55 years at death, b. *c.* 1930, clerk on depot
FP	Qadar Ali Khan, baba, Male, 60 years at death, b. *c.* 1905, farmer
	Saifia Khan, mama, Female, 57 years at death, b. *c.* 1910
FG	Abdul Ali Khan, taiya, Male, 61 years, b. *c.* 1927
	Mustaffa Ali Khan, chacha, Male, 58 years, b.*c.* 1930

usually considering something that will automatically generate diagrams from data, rather than having to designate positions themselves. While this is highly desirable, it is quite difficult to accomplish. (At least, it is very difficult to accomplish in a way which results in the diagram which was wanted, especially if the results are intended for publication.) There have been a number of attempts to develop such a program, but none can do this without some form of additional input from the user. The reason for this is quite simple: there is no single correct way to draw a diagram for a given group of people. This is especially so if the relationships include a large number of people who are interrelated. Since this covers most of the cases for which a diagram is desired, it poses a problem.

The most basic way to produce a simple kinship diagram is to 'manually' draw it using a computer-based drawing tool (see drawing tools and plotting tools in Sections 5.2 and 5.4). Although this requires only the most basic level of support from the computer, it is far easier than producing such diagrams manually using india ink or transfer lettering. The principal advantage of choosing this method to produce diagrams is that a simple tool can be used with none of the constraints likely to be imposed by many computer programs which do more of the work of diagram preparation. The disadvantage is that even small diagrams are still quite tedious, and major modifications are difficult once the diagram has been drawn. However, even if you plan to do your diagrams in this manner, the issues which apply to semi-automatic or automatic preparation in the following discussion are worth considering, since all most of these do is the same operation you intend to pursue manually (as is the case with almost all the applications we shall consider in this book).

There are four general steps for computer-generated kinship diagrams.

1 Structure and description of data.
2 Calculating the positions of diagram objects.
3 Calculating the relationships between objects
4 Drawing the results.

6.5.1
Structure and description of data

This is a crucial design issue for any computer application. Every computer program design will require a specific form for inputting the data. In selecting a pre-existing computer program for drawing kinship diagrams you must consider the form that the input data will take, and ensure that it is compatible with your own data. The form will depend heavily on exactly how much the diagramming program does. At the simplest

Table 6.6 Possible data for kinship diagram

```
Male, Mustafa, 1, 9
Female, Saifia, 1, 11
Female, Nadia, 2, 8
Male, Nadeem, 2, 10
Male, Younis, 2, 12
Spouse, Marriage1, Mustafa, Safia,
Sibling, Sibset1, Nadia, Nadeem, Younis,
Descendant, Nuclear1, Marriage1, Sibset1
```

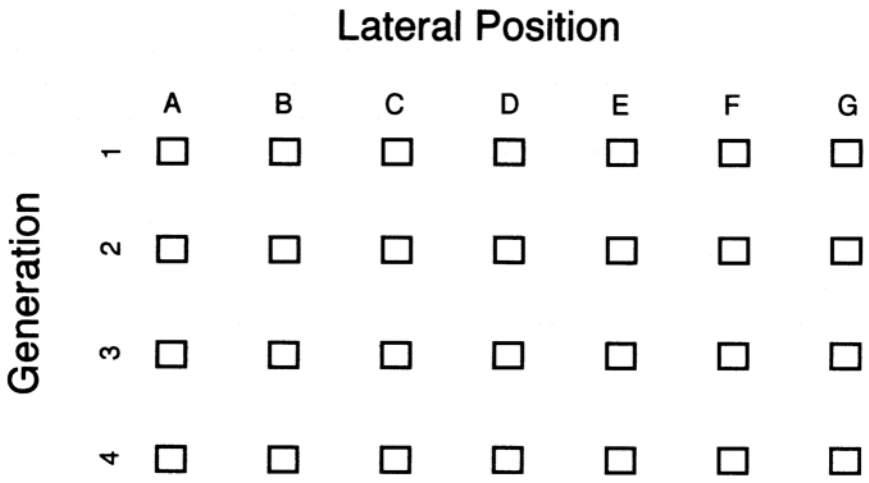


Figure 6.2 Kinship model grid

level (which may be the most satisfactory for the production of diagrams for publication) you designate the names, sexes and physical positions on a grid of each of the people you want to relate. You also include information about which to connect and what kind of connection to make; you perform components (2) and (3) above by including this information in the data. For example, data might take the form shown in Table 6.6. This form of data implies a program which uses the (arbitrary) representational model given in Figure 6.2.

The vertical dimension is descent and the horizontal dimension collaterality. The program will interpret Male to be drawn as '△' and Female as 'O'. Thus Mustafa will be drawn as '△' at Generation row 1, Collateral column G. Similarly

Nadia will be drawn as 'O' at Generation row 2, Collateral column E. The last three statements of the data designate lines to be drawn between the symbols in the diagram. M1 is defined to be a line which connects Mustafa and Safia and is of type Spouse, which might be drawn as '=' or, following the conventions recommended by Barnard and Good (1984:6), as '⌣'. S1 is defined to be a line which connects Nadia, Nadeem and Younis and is of type Sibling, drawn as '⌐┬┐'. N1 is defined to be a line which connects M1 to S1 and is of type Descendant, drawn as 'I' from the centre of M1 to the centre of S1. The resulting diagram will appear as Figure 6.3. A small program which implements this model appears in Chapter 7.

6.5.2 Calculating the positions of diagram objects

The most successful interactive program is G-Net written by N. Ryan (Ryan 1988). G-Net is very general, and is intended for any data which can be represented in hierarchical trees or graphs. It has some optimizations for genealogies. Ryan's general solution is to draw a reasonable diagram from the

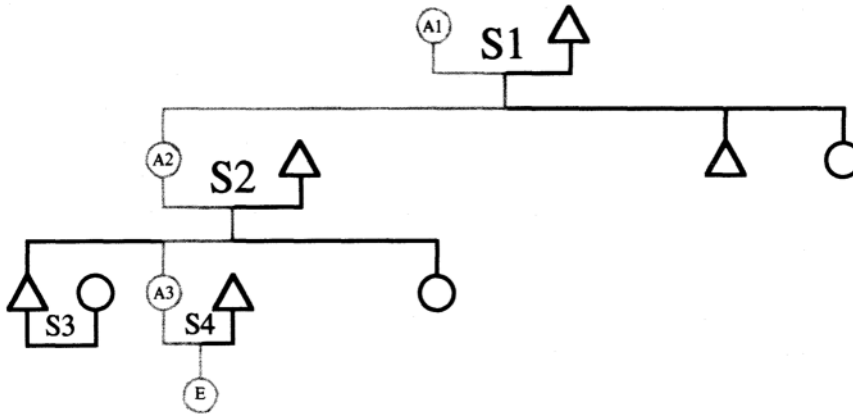


Figure 6.3 Kinship model diagram

data and then to provide editing facilities to the user which allow modifications to the diagram. These modifications include not showing some relationships and allowing the user to interactively move the symbols on the screen until these are in the desired location.

The data requirements are different than those for Section 6.5.1 above. In Section 6.5.1 the data included the symbol name, the symbol form and the location of the symbol on a model grid. A program which is intended to calculate

Table 6.7 Revised sample data

Male, Mustafa
Female, Saifia
Female, Nadia
Male, Nadeem
Male, Younis
Spouse, Marriage1, Mustafa, Safia,
Sibling, Sibset1, Nadia, Nadeem, Younis,
Descendant, Nuclear1, Marriage1, Sibset1

the positions does not require the locational information. Instead it requires data on the way in which the people in the diagram are related to each other. This can take a wide range of forms, but to keep things simple we shall use a model of the data which closely follows the former example (Table 6.7).

The data is exactly the same except that the location data has been omitted. The difference is in the assumptions that the programmer must make in interpreting the data. In the former example the interpretation was more or less literal. Call

this symbol Mustafa, Mustafa is male, Mustafa is at 1, G. Although we could establish relationships from that data, there was no need for the programmer to encode this knowledge into the program. In the present case representing this knowledge is precisely what must be done. A interpretative task is removed from the end user and incorporated into the program. A degree of control is removed from the end user as well.

6.5.3 Calculating the relationships between objects

Calculating the relationships between objects is necessary both to position each symbol in the diagram and to draw the appropriate lines between the symbols. How this is to be done is based on what data is available. There are at least three reasonable approaches.

The first is implied by the example data in Table 6.2. This data model requires that three basic relationships be present in the data: spouses, sibling sets and spouse-set to sibling sets. From this information more complex relationships can be deduced based on a set of rules for each deduction. This approach is a problem if the data is not already organized in terms of sibsets, as is often the case. The second model requires a different data model where the relationship data is in terms of spouses and child relationships. There are two variations, one where the child relationship is to the parent(s) and another where the child relationship is to the spouse unit. The third model lists a number of given relationships from a given ego's reference (e.g. father, mother, paternal uncle, maternal aunt, paternal uncle's wife).

Although each of these is adequate for defining and drawing basic diagrams, for analytic purposes each of these data structures presents problems for a corresponding interpreting program. This is important if we want to do some analysis of the data before producing a diagram, as in Chapter 7.

Some of these problems are shared. For example, the first two models will present problems with the contrast between the rules established for reckoning kinship relationships, as opposed to what people actually recognize or use. The third can be used to deduce a plausible structure which represents how a particular ego sees the kinship universe, but may well fall victim to conflicts between different egos' views of the same people. When using these methods you are producing a theoretical model of the kinship relationships, one which might be compared with different egos' accounts.

Another problem which can emerge, especially with the third approach, is when there are multiple relationships between people in the group. Some programs cannot work with endogamous populations and require the elimination of all circular relationships (e.g. where people are related through two different ancestors). This is, of course, not acceptable to most anthropologists.

Whatever approach you choose (or is imposed upon you by the program you are using), you must be aware of the rules for deducing relationships and make certain that these are acceptable for your purposes. Otherwise you may well find that you have invested a great deal of time in data preparation, perhaps only to produce diagrams which are inaccurate and useless.

This will be an important issue when modelling of kinship relations is carried out. For the purpose of diagram preparation the example approach will serve. Whatever model is in use, the positions must be calculated from this relationship information. For the purposes of our example we shall assume that spouses will appear centred over their children.

6.5.4
Drawing the results

Drawing the results is the goal of the process. The principal point is to produce a diagram in a style which is appropriate for the purpose. This is much easier said than done. Any programmer can produce a program which will draw a 'correct' diagram given a set of objects, their locations and the type of relationship. However, in most cases kinship diagrams are intended to illustrate specific kinds of relationships, not just to 'connect the dots'.

The main problems are representing marriages and aligning generations when the diagram is not rooted with a single ancestor. Representing marriage links is not too difficult if the spouse is not otherwise related within the diagram, but can be a serious problem if they are. Any of the three suggested techniques can be interactively used to present a single diagram suitable for publication. Computer-generated diagrams for purposes of presentation are quite possible but to generate a number of diagrams to be used as analytic tools requires careful specification of the purpose of the analysis.

Chapter 7
Kinship programs

7.1
INTODUCTION

In Section 6.2 we outlined the conceptual specification for programs which would identify the people corresponding to specific (etic) genealogical positions relative to some ego and which would list the genealogical terms describing the relationship(s) between an ego and a specific person.

In this chapter we shall examine some of the ways in which this specification can be implemented in the programming language Prolog (the specification can be implemented in virtually any computer programming language). The approach to implementation was chosen based on the likelihood that the methods would be accessible to anthropologists.

The resulting programs are relatively straightforward and effective, at the cost of some efficiency and risk of derision by programmers. The first method (Section 7.3.1) corresponds to what has been called ‘straight-line coding’, because little or no use is made of the algorithmic capabilities of the programming language. This form of coding is, however, more defensible in Prolog, which makes impossible some of the worst excesses possible in other programming languages. This method is suited to those who will be using such techniques only occasionally, because there is relatively little to understand about the computing and program language if the problem itself is well understood by the anthropologist-programmer. It does, however, limit the degree to which the computer can be used in a given problem area, since much of the value of computers lies in the ability to represent generative descriptions of complex structures.

A second method (Section 7.3.3) uses the capacity of the programming language to write more general (and powerful) generative definitions of structures and operations on structures, but requires much greater ability on the part of the programmer. Again, these do not necessarily correspond to ‘best programming practice’. A balance is struck between style and efficiency and intelligibility for the non-programmer. Even though these are not written in the most compact and generative manner possible, they will initially prove difficult to understand. (Indeed, it is unlikely that the beginner will understand these methods enough to write new programs based on them, without

considerable effort and resort to introductory books on Prolog (see Bratko (1986) for a full introduction. Kononenko and Lavrac (1989) provide a quick introduction with examples.).)

The programs are included because they (a) illustrate the implementation of specifications such as those we produced in Chapter 6, and (b) serve as a working model and starting point for discussion with a programming partner. They provide examples of attacking familiar problems which can be useful if you do develop programming further. Perhaps most importantly, as written they perform tasks which should be useful and are immediately accessible by typing the program text into the computer.

7.2 THE INPUT DATA MODEL

7.2.1 Representing the input data model

Table 7.1 describes some partial data for a genealogy, which corresponds to the input data model outlined in Section 6.3.6. Although data could be prepared in this form, the purpose of the model is not to determine the form in which the data is collected, or even the form in which data will be presented to the program, but to represent the information requirements of the program (in the terms of Section 6.2 the abstract data model) in a relatively non-redundant form. Whatever form the input data takes, it must be transformable into the terms of the input data model.

The *Relation* in each case designates a collection of information. The other headings label a category of information within this collection. The different headings are associated by the relations. The relation *gender* associates each *Person ID* with a *Gender* value. The relation *spouse_set* associates each *Person ID* with a *Union ID* (a name for a specific union or marriage) and *Union Order*. In this case *Union Order* is a simple number designating the ordinal of this particular union (first, second, third marriage), but it could be more complex if required in a specific social context. The relation *sib_set* associates a person with their parent's *Union ID* and the ordinal of their position in the sibling set (first born etc.).

7.2.2 Representing relations in prolog

The basic building block in Prolog is called a clause, which can take forms which include:

```
P.

P(X).

P(X,Y).
```

Table 7.1 Data fragment for input data model from Section 6.3.6

Relation	*Person ID*	*Gender*	
gender	abdul	male	
gender	mustaffa	male	
gender	hamiz	male	
gender	sufia	female	
gender	humza	male	
gender	nadya	female	
Relation	*Union ID*	*Person ID*	*Union Order*
spouse_set	1	mustaffa	1
spouse_set	2	hamiz	1
spouse_set	2	sufia	2
spouse_set	5	humza	1
spouse_set	10	abdul	1
spouse_set	10	nadya	1
Relation	*Union ID*	*Person ID*	*Sibling Order*
sib_set	1	abdul	3
sib_set	2	mustaffa	2
sib_set	3	hamiz	1
sib_set	4	sufia	2
sib_set	2	humza	1
sib_set	5	nadya	2

P(X,Y,Z).

P(X,Y), Q(Y,Z).

P(X,Y):-Q(Y,X).

where *P* is any predicate and *X* and *Y* are variables which range over some domain. The predicate P asserts a relationship between the arguments, if any, and can be evaluated by Prolog as either true or false. The relationship asserted by a predicate can be of any sort: 'male(X)' /X is male/; 'mother(X,Y)' /X is the mother of Y/; 'younger(X,Y)' /X is younger than Y/. The programmer, by their authoring of the predicates, states the relationships and is responsible for ensuring that conclusions drawn from evaluating the predicates have a

reasonable interpretation outside the program. For obvious reasons, names are used which are meaningful to the programmer. These names are not meaningful to Prolog. For example, we might have two named predicates, *spouse* and *gender:*

spouse(X,Y) /X is a spouse of Y/

gender(X,Y) /the gender of X is Y/

Prolog must have some basis for assigning values of *true* or *false* to statements consisting of or including these predicates. The most basic method of supplying this information is to insert concrete instances of the predicates by replacing the variables with data.

spouse(mary, john).

spouse(carl, janet).

gender(mary, female).

gender(john, male).

gender(carl, male).

gender(janet, female).

This form of predicate instance is, obviously, data. The anthropologist selects the predicate names and types in the information. At this early stage the anthropologist can access the information in four ways, by instructing Prolog to evaluate:

spouse(mary, john). /is mary the spouse of john?/

spouse(john,mary). /is john the spouse of mary?/

spouse(F, john). /who is the spouse of john?/

spouse(mary, K). /mary is the spouse of whom?/

spouse(V,L). /who is the spouse of whom?/

The first query is simply evaluated as *true,* because this is an instance previously supplied to Prolog. The second query is evaluated as *false*. There is no instance which matches 'spouse(john,mary)'. Prolog, in evaluating this query, can make no use of the knowledge that *spouse* is a reciprocal term, because this knowledge has not been made available. A simple (but wasteful) means of establishing this relationship is to add the instance 'spouse(john,mary)' to the program.

The terms F, *K, V* and *L* in the latter three examples are variables. Any term which begins with an uppercase letter (A-Z) is treated as a variable by Prolog. When variables are present, information from each matching instance of the predicate is assigned to the variable from the corresponding position. The latter three will be evaluated as *true,* but the presence of variables in the query will

give more information. The third query supplies the additional information that the value of the variable *F* is 'mary'. The fourth that *K* is 'john'. The fifth is evaluated as *true* twice, the first time with 'V=mary' and 'L=john' and the second time with 'V=carl' and 'L=janet'.

This is mildly useful, particularly if a large number of spouses are under investigation. Much more useful are compound clauses. If, rather than spouses, we want to list wives, we would make the query:

```
spouse(Wife,Ego), gender(Wife,female).
```

Prolog will evaluate 'spouse(Wife,Ego)' as being twice *true*. The first time *Wife* will be bound to 'mary' and *Ego* to 'john'. The *comma* instructs Prolog to evaluate the second clause with these same variable bindings. Treating 'gender (Wife, female)' as 'gender(mary, female)', this clause will evaluate as true and the variable bindings can be examined to determine who this wife is and to whom. The second time the 'spouse(Wife, Ego)' clause is evaluated as *true,* with *Wife* bound to 'carl' and *Ego* to 'janet'. Evaluating the second clause with these bindings results in evaluating 'gender(carl,female)' as *false,* therefore the compound clause is false and no variable bindings are available for examination.

Quite complex compound clauses can be formed, but if you intend to use the same clause more than once, or use it as the basis of an even more complex clause, the clause can be added to the program as a rule. We can define a predicate *wife* with the following instance in the program:

```
wife(Wife, Ego):-spouse(Wife,Ego), gender(Wife,female).
```

This can be read: 'wife(Wife, Ego)' is true when the compound clause is true. If the compound clause is true, then the variables in 'wife(Wife,Ego)' will be bound to the values which led to this evaluation. If you make the query 'wife (X,Y)', this will be evaluated precisely as the previous example, reporting that *X* is 'mary' and *Y* is 'john'. The variables in the query need not be those in the definition. Prolog attaches no meaning to variable names other than the establishment of uniqueness in a clause or compound clause; within a clause all variable instances with the same name will be bound to the same value.

The present definition of *wife* only finds wives when these are in the first position of the spouse clause. This can be corrected by adding a second definition of *wife:*

```
wife(Wife, Ego):-spouse(Ego,Wife), gender(Wife,female).
```

Any predicate can have any number of instances, either of a concrete form or defined in terms of compound clauses. When evaluating the query 'wife (Wife,Ego)', the first instance will be tried and then the second.

7.2.3 Defining the input data model in Prolog

Table 7.2 shows one way the data in Table 7.1 can be represented in Prolog. A relation name becomes a predicate name in Prolog. Prolog has no information about the predicates *gender, spouse_set* and *sib_set* in Table 7.2, other than the

following: there is a predicate named *gender* which is applied to two arguments. Any other meaning must be established by conventions which are applied in the program. It may be obvious (to people) from the predicate form of *gender* that the first position is a person's name and the second is an English gender term, but Prolog makes no such presumptions. The predicate 'x(y,z)' would have served as well, so long as we remember that *x* is a predicate which relates a gender *z* to a given person *y,* which gender value *z* stands for and who *y* is.

Table 7.2 Prolog representation of Table 7.1

```
gender(abdul, male).
gender(mustaffa, male).
gender(hamiz, male).
gender(sufia, female).
gender(humza, male).
gender(nadya, female).

spouse_set(1, mustaffa, 1).
spouse_set(2, hamiz, 1).
spouse_set(2, sufia, 2).
spouse_set(5, humza, 1).
spouse_set(10, abdul, 1).
spouse_set(10, nadya, 1).

sib_set(1, abdul, 3).
sib_set(2, mustaffa, 2).
sib_set(3, hamiz, 1).
sib_set(4, sufia, 2).
sib_set(2, humza, 1).
sib_set(5, nadya, 2).
```

7.3 REPRESENTING THE ABSTRACT DATA MODEL

7.3.1 The abstract data model

The need for a separate input data model and abstract data model arises from the basic concepts of genealogical analysis (among others). Genealogical relations are conceptualized as being relative to a specific individual, but it is useful to have the freedom to designate and vary which individual that might be. To input all the data directly in this form would be very wasteful. For a single genealogical

interview of thirty people, this might consist of 870 entries (thirty people with twenty-nine relations each). Using the proposed input data model in Section 7.2.3 requires ninety entries at most, and contains all the information required to construct these 870 relationships.

The abstract data model is the set of schema or rules required to transform the normalized input data format into ego-relative terms. Following the specification in Section 6.3.6, Program 7.1 is a Prolog implementation of instances which define the basic genealogical categories from the normalized input data.

These rules define the basic relationships, designated by a single letter corresponding to the notation suggested by Barnard and Good (1984: 4, Table 1. 1), with the exception of using lowercase letters for convenience due to the special meaning of uppercase letters in Prolog. For example, compare the Prolog rule (predicate) for p /parent/ with the specification for parent in Section 6.3.6:

parent iff Spouse_ID of A is Parent's Spouse_ID of Ego then A is Parent of Ego

```
p(Parent,Ego) :-
sib_set(Spset,Ego,Order),
spouse_set(Spset,Parent,Union).
```

1 The goal of the rule is to determine if Parent is the parent of Ego. In Prolog this goal is written p(Parent, Ego).
2 The construction ':-' is written after the goal. This can be read as 'when' or 'if and introduces the set of clauses which specify how to satisfy the goal statement.
3 The clauses necessary to establish the goal statement are separated by commas and ended with a full stop ('.').

If the information in Table 7.2 and Program 7.1 are consulted as a program in Prolog, the following query can be evaluated:

p(X,Y). Who is a parent of Whom, or list all the parent-child pairs/

To evaluate this query, the first goal in the definition 'sib_set(Spset, Ego, Order)' will be evaluated. Since *sib_set* is defined by instances of data this clause will be evaluated as:

sib_set(l, abdul, 3)

Ego is now bound to 'abdul', *Spset* to 1 and *Order* to 3. The second clause, 'spouse_set(Spset,Parent,Union)', begins with *Spset* set to 1, or 'spouse_set(1, Parent,Union)'. This is evaluated as:

spouse_set(l, mustaffa, 1).

So *Parent* is now bound to 'mustaffa'. Since both clauses have evaluated true, the goal is satisfied and *Ego* and *Parent* are bound to the specific values 'abdul' and 'mustaffa'.

Prolog is called a non-deterministic language because there can be more than one solution to a problem. For this query there are six:

p(mustaffa, abdul).

p(hamiz, mustaffa).

p(sufia, mustaffa).

p(hamiz, humza).

p(sufia, humza).

p(humza, nadya).

with the order of evaluation:

```
p(X,Y).
    sib_set(1, abdul, 3)
        spouse_set(1, mustaffa, 1)      → p(mustaffa, abdul).
    sib_set(2, mustaffa, 2)
        spouse_set(2, hamiz, 1)         → p(hamiz, mustaffa).
        spouse_set(2, sufia, 2)         → p(sufia, mustaffa).
    sib_set(3, hamiz, 1)
        spouse_set(3,X,Union)           → FAIL (no solution, no report).
    sib_set(4, sufia, 2)
        spouse_set(3,X,Union)           → FAIL (no solution, no report).
    sib_set(2, humza, 1)
        spouse_set(2, hamiz, 1)         → p(hamiz, humza).
        spouse_set(2, sufia, 2)         → p(sufia, humza).
    sib_set(5, nadya, 2)
        spouse_set(5, humza, 1)         → p(humza, nadya).
```

In the evaluation of the query 'p(hamiz, Ego)', the variable *Parent* will be bound to 'hamiz' wherever it appears in the definition, which will follow the order of evaluation:

```
p(hamiz,Ego).
    sib_set(1, abdul, 3)
        spouse_set(1, hamiz, Union)      → FAIL.
    sib_set(2, mustaffa, 2)
        spouse_set(2, hamiz, 1)          → p(hamiz, mustaffa).
    sib_set(3, hamiz, 1)
        spouse_set(3,hamiz,Union)        → FAIL.
    sib_set(4, sufia, 2)
        spouse_set(3,hamiz,Union)        → FAIL.
    sib_set(2, humza, 1)
        spouse_set(2, hamiz, 1)          → p(hamiz, humza).
    sib_set(5, nadya, 2)
        spouse_set(5, hamiz, 1)          → FAIL.
```

The query 'p(Q, nadya).' will be evaluated as:

```
p(Q, nadya).
    sib_set(5, nadya, 2)
        spouse_set(5, humza, 1)          — p(humza, nadya).
```

These evaluation sequences illustrate three important points. First, although the variable names carry no special meaning in Prolog, a given name has the same value throughout a rule: once a value is assigned to a variable it 'sticks' through all subsequent clauses. Second, the variable names used in the query need not match the variable names used in defining the rule. In Prolog the variables are related to the position in which they appear in the goal statement and query, not the specific form of the variable. Third, the computational efficiency of a definition can vary depending on the order of the defining clauses (although the resulting answer will not). This is not important if there are six clauses to evaluate, but can be an issue if there are one or two thousand.

Once a rule is defined in Prolog, it can be used within the definition of another rule just as if it were defined in terms of data. Making rules establishes knowledge about interrelationships in the data within the program. Having established how to derive a parent relationship from the input data model, the rule can be used to define other rules for father, mother and child relationships.

The remainder of the rules in Program 7.1 should be clear, except for the construction 'Ego\= Sib' in the *g* rule (sibling) and 'Ego\= Spouse' in the *e* rule (spouse). 'Ego\= Sib' can be read *'Ego* is not equal to *Sib'*. This situation arises because we are using the predicate *sib_set* twice in the definition, and although *Ego* and *Sib* are different variables they can have the same value. Unless a person can be considered their own sibling, then this situation must be blocked by using the 'not equal to' predicate at the end of the definition. It must be at the end because it is the *sib_set* predicates which bind the variables to values.

In many cases other kinds of conceptual distinctions are necessary to accommodate significant relationships in some societies. For example, many societies distinguish between elder and younger siblings. The input data model

Program 7.1 Prolog implementation of the Abstract Data Model

```
p(Parent,Ego):-
      sib_set(Spset,Ego,Order),
      spouse_set(Spset,Parent,Union).

f(Father,Ego):-p(Father,Ego), gender(Father,male).
m(Mother,Ego):-p(Mother,Ego), gender(Mother,female).

g(Sib,Ego):-sib_set(Spset,Ego,OrderSib), sib_set(Spset,Sib,OrderEgo), Ego \= Sib.

b(Bro,Ego):-g(Bro,Ego), gender(Bro, male).
z(Sis,Ego):-g(Sis,Ego), gender(Sis, female).

c(Child,Ego):-p(Ego,Child).

s(Son,Ego):-c(Son,Ego), gender(Son, male).
d(Daug,Ego):-c(Daug,Ego), gender(Daug, female).

e(Spouse,Ego):-
      spouse_set(Spset,Ego,MorderEgo),
      spouse_set(Spset, Spouse,MorderSpouse), Ego \= Spouse.

w(Wife,Ego):-
      e(Wife,Ego), gender(Wife, female).
h(Husband,Ego):-
      e(Husband,Ego), gender(Husband, male).
```

Program 7.2 Prolog rules for younger and elder sibling

```
gy(Sib,Ego):-
      sib_set(Spset,Ego,Ordego),
      sib_set(Spset,Sib,Ordsib),
      Ordsib < Ordego.
ge(Sib,Ego):-
      sib_set(Spset,Ego,Ordego),
      sib_set(Spset,Sib,Ordsib),
      Ordsib > Ordego.
```

encodes the necessary information in the *sib_set* predicate in the *Order* parameter. Rules for making this distinction in the abstract data model are in Program 7.2. In these rules the *Order* parameter for each are compared. The '<' is read as less than and '>' as greater than. Because *Order* cannot be equal, it is not necessary to check that *Ego* and *Sib* are not the same value, although it would not disturb anything to do so.

7.3.2 Finding complex relationships based on the abstract data model

7.3.2.1 A simple version

Program 7.3(a) is a portion of a simple, if inelegant, program for finding arbitrarily complex ego-centric relationships between people. The program is composed of as many definitions of a new predicate, *relation,* as is required to identify all the relationships that are of interest.

The basic form is:

```
relation(Rel,Ego,Relname) :- link1(L1,Ego), link2(L2,Ll),…, linkn
(Rel,L2).
```

Because the kind of relationship is 'hard-wired' in the definition, the *Relname* is supplied as a fixed term. The main disadvantage to this approach is that a large number of relation predicates (over 200) must be written to cover the genderspecific expansion of the thirty-nine types of relationship recommended as a minimum by Barnard and Good (1984: 30) reproduced in Table 6.4. It can also be a tedious exercise to modify if only subsets of relationships are needed for a specific analysis, although separate versions of the program can be developed for different needs with the aid of a text editor and multiple copies of an original. The advantages are:

1 the approach is easy to understand;
2 the order in which relationships are reported is directly under the researcher's control. Prolog evaluates all the applicable (matching) relation predicates in the order these appear in the original program text;
3 only the relationships of interest need be included (and reported).

The tedium of entering the predicates can be reduced somewhat by the use of a good text editor, duplicating previous similar clauses and amending these. Be careful, because this is also a potential source of systematic error. It is not necessary to use variable names such as *MM* or *MMZ,* as appear in the program text, but this is helpful when referring back to the program and looks rather better than *X* and *Y*. Also, as in the last definition in Program 7.3(a), other definitions of *relation* can be used to find the last link to the potential relative.

In many cases it is necessary (or highly desirable) to know not only the relationship but the linking relatives as well. Program 7.3(b) demonstrates a simple modification for this purpose, adding a list to store the linking individuals. A list in Prolog acts as a single object, but can contain a complex structure which includes other lists. In this case the list is a simple one, containing the linking individuals separated by commas. Lists are surrounded by left and right square brackets, '[' and ']'. In the parameter list of *relation,* the list is a single

Program 7.3(a) Simple program for finding complex etic relationships

```
relation(Rel,Ego,'F') :- f(Rel,Ego).
relation(Rel,Ego,'M') :- m(Rel,Ego).
relation(Rel,Ego,'B') :- b(Rel,Ego).
relation(Rel,Ego,'Z') :- z(Rel,Ego).
relation(Rel,Ego,'S') :- s(Rel,Ego).
relation(Rel,Ego,'D') :- d(Rel,Ego).
relation(Rel,Ego,'W') :- w(Rel,Ego).
relation(Rel,Ego,'H') :- h(Rel,Ego).
relation(Rel,Ego,'P') :- p(Rel,Ego).
relation(Rel,Ego,'C') :- c(Rel,Ego).
relation(Rel,Ego,'G') :- g(Rel,Ego).
relation(Rel,Ego,'E') :- e(Rel,Ego).

relation(Rel,Ego,'MZ') :- m(M,Ego), z(Rel,M).
relation(Rel,Ego,'MB') :- m(M,Ego), b(Rel,M).
relation(Rel,Ego,'FZ') :- f(F,Ego), z(Rel,F).
relation(Rel,Ego,'FB') :- f(F,Ego), b(Rel,F).

relation(Rel,Ego,'MM') :- m(M,Ego), m(Rel,M).
relation(Rel,Ego,'MF') :- m(M,Ego), f(Rel,M).
relation(Rel,Ego,'FM') :- f(F,Ego), m(Rel,F).
relation(Rel,Ego,'FF') :- f(F,Ego), f(Rel,F).
relation(Rel,Ego,'MMZ') :- m(M,Ego), m(MM,M), z(Rel,MM).
relation(Rel,Ego,'MFZ') :- m(M,Ego), f(FF,M), z(Rel,FF).
relation(Rel,Ego,'MMB') :- m(M,Ego), m(MM,M), b(Rel,MM).
relation(Rel,Ego,'MFB') :- m(M,Ego), f(FF,M), b(Rel,FF).

relation(Rel,Ego,'MMZD') :- m(M,Ego), m(MM,M), z(MMZ,MM), d(Rel,MMZ).

or

relation(Rel,Ego,'MMZD') :- relation(MMZ,Ego,'MMZ'), d(Rel,MMZ).

. . .
```

parameter, regardless of how many values are within it. For example, consider the query:

relation(nadya, abdul, Relation, Links).

and the responses:

Relation='W', Links=[abdul,nadya].

Relation='FBD', Links=[abdul,mustaffa,humza,nadya].

Program 7.3(b) Program 7.3(a) modified to record people in relationships

```
relation(Rel,Ego,'F',[Ego,Rel]) :- f(Rel,Ego).
relation(Rel,Ego,'M',[Ego,Rel]) :- m(Rel,Ego).
relation(Rel,Ego,'B',[Ego,Rel]) :- b(Rel,Ego).
relation(Rel,Ego,'Z',[Ego,Rel]) :- z(Rel,Ego).
relation(Rel,Ego,'S',[Ego,Rel]) :- s(Rel,Ego).
relation(Rel,Ego,'D',[Ego,Rel]) :- d(Rel,Ego).
relation(Rel,Ego,'W',[Ego,Rel]) :- w(Rel,Ego).
relation(Rel,Ego,'H',[Ego,Rel]) :- h(Rel,Ego).
relation(Rel,Ego,'P',[Ego,Rel]) :- p(Rel,Ego).
relation(Rel,Ego,'C',[Ego,Rel]) :- c(Rel,Ego).
relation(Rel,Ego,'G',[Ego,Rel]) :- g(Rel,Ego).
relation(Rel,Ego,'E',[Ego,Rel]) :- e(Rel,Ego).

relation(Rel,Ego,'MZ',[Ego,M,Rel]) :- m(M,Ego), z(Rel,M).
relation(Rel,Ego,'MB',[Ego,M,Rel]) :- m(M,Ego), b(Rel,M).
relation(Rel,Ego,'FZ',[Ego,F,Rel]) :- f(F,Ego), z(Rel,F).
relation(Rel,Ego,'FB',[Ego,F,Rel]) :- f(F,Ego), b(Rel,F).

relation(Rel,Ego,'MM',[Ego,M,Rel]) :- m(M,Ego), m(Rel,M).
relation(Rel,Ego,'MF',[Ego,M,Rel]) :- m(M,Ego), f(Rel,M).
relation(Rel,Ego,'FM',[Ego,F,Rel]) :- f(F,Ego), m(Rel,F).
relation(Rel,Ego,'FF',[Ego,F,Rel]) :- f(F,Ego), f(Rel,F).

relation(Rel,Ego,'MMZ',[Ego,M,MM,Rel]) :- m(M,Ego), m(MM,M), z(Rel,MM).
relation(Rel,Ego,'MFZ',[Ego,M,MF,Rel]) :- m(M,Ego), f(MF,M), z(Rel,FF).
relation(Rel,Ego,'MMB',[Ego,M,MM,Rel]) :- m(M,Ego), m(MM,M), b(Rel,MM).
relation(Rel,Ego,'MFB',[Ego,M,MF,Rel]) :- m(M,Ego), f(MF,M), b(Rel,FF).

relation(Rel,Ego,'MMZD',[Ego,M,MM,MMZ,Rel]) :-
    m(M,Ego), m(MM,M), z(MMZ,MM), d(Rel,MMZ).

or

relation(Rel,Ego,'MMZD',[Ego,M,MM,MMZ,Rel]) :-
    relation(MMZ,Ego,'MMZD',[Ego,M,MM,MMZ]), d(Rel,MMZ).

. . .
```

7.3.2.2 Another stage

Although Program 7.3(b) will suffice for some research purposes, it is extremely cumbersome if relationships of reasonable depth are required. There are at least 460 possible gender-specific relationships (including affines) with up to four links ('MMZD'), and at least 1,580 with up to five links.

Prolog has powerful mechanisms for specifying generative predicates, where specific cases can be generated rather than listed as in Program 7.3(b). This increases the complexity of defining rules. Program 7.4(a) is a first draft of a simple generative program for finding relationships. It operates on the non-

deterministic properties of Prolog interpretation, in which all possible positive solutions are produced.

For example, the first *relations* rule finds all one-term relationships by trying each instance of the *simple_rel* predicate in turn. Those which succeed are reported, binding the relevant variable as in Program 7.3(b). This single *relations* rule 'replaces' eight *relation* rules from Program 7.3(b). The second *relations* rule finds all two-term relationships, replacing sixty-four *relation* rules (Program 7.4(a) finds relationships such as 'WH' and 'FS', and even 'WW', as it stands).

The four relations rules will attempt to locate 4,096 relationships, although some of these are impossible or unlikely, such as 'HH' or 'HWHW'. The query:

```
?- relations(nadya, abdul, T, L).
```

reports seven relationships:

```
№1 T    ='W', L=[abdul, nadya]
№2 T    ='FBD', L=[abdul, mustaffa, humza, nadya]
№3 T    ='FSW', L=[abdul, mustaffa, abdul, nadya]
№4 T    ='WFD', L=[abdul, nadya, humza, nadya]
№5 T    ='WHW', L=[abdul, nadya, abdul, nadya]
№6 T    ='FFSD', L=[abdul, mustaffa, hamiz, humza, nadya]
№7 T    ='FMSD', L=[abdul, mustaffa, sufia, humza, nadya]
```

To control the range of possible relationships the transitions between terms must be controlled. Program 7.4(b) contains an additional predicate, *transition,* which generates valid transitions for genealogical terms and revisions to the *relations* predicate which use these transitions. Again, because of the generative nature of Prolog, all the transitions will be used, if necessary. The order of the transitions is only relevant to the order in which relationships will be reported.

Valid transitions are, of course, a matter for the researcher. The given *transition* predicates should be altered as necessary. For example, if affinal relations are not required, remove the transitions for 'H' and 'W', and relationships such as 'WF' will not be considered.

Program 7.4(b) may be slow if it is used on a database with a large number of people. This is less of a problem if the relationship to be found is not known in advance, for instance if the relationships between two people are at issue (e.g. '?-relations(abdul, nadya, T, L)') and cannot easily be remedied in any case. But if the relation is known in advance, the process of finding people who correspond to this relationship can be greatly accelerated. The problem with the program as it is written so far is that if the relationship is given in the query, the relationship is not checked until the end of the *relations* rule. After all cases of the relevant relationship are found, the program will continue until it has exhausted all possibilities. With up to five-term relationships there are 1,580 possible simple and complex connections. On a large database of people as many as 7,264 simple relationships (e.g. calls to *simple_rel*) might be evaluated. At 100 evaluations per

Program 7.4(a) A generative relationship program

```
simple_rel(Rel,Ego,'F') :- f(Rel,Ego).
simple_rel(Rel,Ego,'M') :- m(Rel,Ego).
simple_rel(Rel,Ego,'B') :- b(Rel,Ego).
simple_rel(Rel,Ego,'Z') :- z(Rel,Ego).
simple_rel(Rel,Ego,'S') :- s(Rel,Ego).
simple_rel(Rel,Ego,'D') :- d(Rel,Ego).
simple_rel(Rel,Ego,'W') :- w(Rel,Ego).
simple_rel(Rel,Ego,'H') :- h(Rel,Ego).

relations(Rel,Ego,Gen,[Ego,Rel]) :- simple_rel(Rel,Ego,Gen).

relations(Rel,Ego,Gen,[Ego,L1,Rel]) :-
      simple_rel(L1,Ego,T1), simple_rel(Rel,L1,T2),
      stringof([T1,T2],Gen).

relations(Rel,Ego,Gen,[Ego,L1,L2,Rel]) :-
      simple_rel(L1,Ego,T1), simple_rel(L2,L1,T2),
      simple_rel(Rel,L2,T3),
      stringof([T1,T2,T3],Gen).

relations(Rel,Ego,Gen,[Ego,L1,L2,L3,Rel]) :-
      simple_rel(L1,Ego,T1), simple_rel(L2,L1,T2),
      simple_rel(L3,L2,T3), simple_rel(Rel,L3,T4),
      stringof([T1,T2,T3,T4],Gen).

[ If your Prolog lacks stringof you can use the following definition, a method
using name predicate, which should be standard (refer to your manual) ]

stringof(Charlist, Atom) :-
      nonvar(Atom),!,
      name(Atom, Bytelist),
      strof1(Bytelist,Charlist).

strof1([ ],[ ]).

strof1([T|Rest], [C|More]) :-
      name(C,[T]),
      strof1(Rest,More).

stringof(Charlist,Atom) :-
      nonvar(Charlist),
      strof1(Bytelist,Charlist),
      name(Atom,Bytelist).
```

second, the total search might amount to over a minute. In practice, these maximums are unlikely to occur for demographic reasons, but unnecessary delays will occur.

Program 7.4(c) contains a new predicate, *relation,* and revisions to *relations* to remedy this problem. The predicate *relation* checks to see if the relationship is a variable or not, i.e. if the query has a term or a variable in that position. If it is not

Program 7.4(b) Additions and revisions to generative relationship program

```
transition('F','F').   transition('F','M').   transition('F','B').   transition('F','Z').
transition('M','F').   transition('M','M').   transition('M','B').   transition('M,'Z').
transition('B','S').   transition('B','D').   transition('B','W').
transition('Z','S').   transition('Z','D').   transition('Z','H').
transition('S','S').   transition('S','D').   transition('S','W').
transition('D','S').   transition('D','D').   transition('D','H').
transition('H','B').   transition('H','Z').   transition('H','F').   transition('H','M').
transition('W','B').   transition('W','Z').   transition('W','F').   transition('W','M').

relations(Rel,Ego,Gen,[Ego,Rel]) :- simple_rel(Rel,Ego,Gen).

relations(Rel,Ego,Gen,[Ego,L1,Rel]) :-
    simple_rel(L1,Ego,T1), transition(T1,T2),simple_rel(Rel,L1,T2),
    stringof([T1,T2],Gen).

relations(Rel,Ego,Gen,[Ego,L1,L2,Rel]) :-
    simple_rel(L1,Ego,T1), transition(T1,T2),simple_rel(L2,L1,T2),
    transition(T2,T3), simple_rel(Rel,L2,T3),
    stringof([T1,T2,T3],Gen).

relations(Rel,Ego,Gen,[Ego,L1,L2,L3,Rel]) :-
    simple_rel(L1,Ego,T1), transition(T1,T2),
    simple_rel(L2,L1,T2), transition(T2,T3),
    simple_rel(L3,L2,T3), transition(T3,T4), simple_rel(Rel,L3,T4),
    stringof([T1,T2,T3,T4],Gen).
```

a variable, the *stringof* predicate converts the genealogical term (or atom in Prologese) into a list of the letters in the term (*stringof* is built in to most Prologs. See Program 7.4(a) if it is not.) This binds the variables *T1, T2* etc. in the rules to these specific simple relationships, so that alternatives will not be searched. It
also sets the length of the list to a specific value, so that if 'FBD' is the query relationship, only the three-term rule will apply.

The additions to the *relations* rules also opens up the possibility of greater control over the relations to be examined. For example, the query:

?-relations (Rel, abdul, Term, Link, ['F', 'B', T]).

will find only relationships via FB, FBD, FBS and FBW with the current transition control. However, by specifying the list as three long, only three-term relationships can be reported. Prolog has a special list notation that removes this restriction:

?-relations (Rel, abdul, Term, Link, ['F', 'B'| Rest]).

Program 7.4(c) Additions and revisions to improve efficiency of generative relationship program

```
relation(Rel,Ego,Gen,Link) :-
     nonvar(Gen), stringof(Gens,Gen), relations(Rel,Ego,Gen,Link,Gens).
relation(Rel,Ego,Gen,Link) :-
     var(Gen), relations(Rel,Ego,Gen,Link,Gens).

relations(Rel,Ego,Gen,[Ego,Rel],[Gen]) :- simple_rel(Rel,Ego,Gen).

relations(Rel,Ego,Gen,[Ego,L1,Rel],[T1,T2]) :-
     simple_rel(L1,Ego,T1), transition(T1,T2),simple_rel(Rel,L1,T2),
     stringof([T1,T2],Gen).

relations(Rel,Ego,Gen,[Ego,L1,L2,Rel],[T1,T2,T3]) :-
     simple_rel(L1,Ego,T1), transition(T1,T2),simple_rel(L2,L1,T2),
     transition(T2,T3), simple_rel(Rel,L2,T3),
     stringof([T1,T2,T3],Gen).

relations(Rel,Ego,Gen,[Ego,L1,L2,L3,Rel],[T1,T2,T3,T4]) :-
     simple_rel(L1,Ego,T1), transition(T1,T2),
     simple_rel(L2,L1,T2), transition(T2,T3),
     simple_rel(L3,L2,T3), transition(T3,T4), simple_rel(Rel,L3,T4),
     stringof([T1,T2,T3,T4],Gen).

[Define more relations definitions for longer links if necessary.]
```

The vertical bar 'I' indicates that the remainder of the list, from zero to *n* elements, will be bound to the variable *Rest* (or whatever name is chosen). This form of the query will report all two, three and higher term relationships.

It is also possible to specify individuals in the *Link* list in the same manner:

```
?-relations (Rel, abdul, Term, [X, hamiz| More], ['F', 'B'| Rest]).
```

so that relationships through that person only will be considered, if any. If the additional control is not needed, then the *relation* predicate can be used.

7.3.3 Controlling the search

Program 7.4(c) illustrates the requirements of making the program both efficient and permitting more control over the relationships which are searched for, since most of the time we probably do not want to see all possible relationships. Program 7.5(a) introduces the idea of 'generate and test' for finding specific relationships. The primary addition is a predicate *valid_rel* which is a set of predicates in which you list those relations of interest. There are changes to *relation* which use *valid_rel* in the second *relation* definition to generate values to pass to the first definition of *relation*. It is assumed that any term you select for the first definition is valid. Note that the second *relation* predicate makes a call

Program 7.5(a) Program modifications to Program 7.4(c) to control search

```
valid_rel('F').          valid_rel('FF').
valid_rel('FFF').        valid_rel('FM').
valid_rel('FMF').        valid_rel('FMM').
valid_rel('FB').         valid_rel('FZ').
valid_rel('FBS').        valid_rel('FBD').
valid_rel('FZS').        valid_rel('FZD').
valid_rel('S').          valid_rel('SS').
valid_rel('SSS').

relation(Rel,Ego,Gen,Link) :-
    nonvar(Gen), stringof(Gens,Gen), relations(Rel,Ego,Gen,Link,Gens).
relation(Rel,Ego,Gen,Link) :-
    var(Gen), valid_rel(Gen), relation(Rel,Ego,Gen,Link).

[The relations predicate is defined in Program 7.4c]
```

to *relation*. This is an elementary example of *recursion,* defining a predicate in terms of itself, although in this case it is trivial since the call can only be successful for the first *relation* predicate, since the term corresponding to *Gen* will not be a variable since it is bound before the call.

Program 7.5(a) has much the same problem as Program 7.3(b); you have to specify each kind of relationship you want to investigate, and there are a large number of these. Using recursion we can introduce another kind of control over the kinds of relationships to search for. In addition to asserting relationships using *valid_rel,* we can control term generation in terms of affinity, collaterality and ascending and descending generation. We can, for example, generate all terms with no more than one affinal link, or no more than two ascending generations, or relations which are strictly lineal. In Program 7.5(b), the predicate *relation* is modified to use a new predicate *gen_rel,* which generates relationship terms, based on a specification by the user, of how many links exist through affines, collaterals and ascending and descending relations. As an example, the query:

relation (Rel, Ego, Term, Link, 1, 1, 1, 1).

will report all people in the database related with at most one link of each type; F, FB, FBS, FBD, FBW, B, BW, BWF etc. It does this using the recursive routine *grel1,* which uses the *transition* predicate defined in Program 7.4(b), and a predicate *valid_trans* which subtracts one from the appropriate variable as the links are followed. When all the parameters *Affine, Collateral, Ascending, Descending* are zero, then all calls to *valid_trans* fail and the recursion in *grel1* ceases. This is a more typical use of recursion, since the same definition *of grell* is called by itself to follow the next link; literally an operation which is defined in terms of itself. Prolog makes considerable use of recursion, and to fully use

the language it must be understood. Because of its importance, it is well covered in all textbooks on Prolog.

7.3.4
Relationship terminologies

We can introduce the use of emic relationship categories into this program by the simple inclusion of a list of predicates with terms as one argument and the etic kin string as the other, such as:

```
terms(father, 'F').          terms(mother, 'M').
terms(grandfather, 'FF').    terms(grandmother, 'GM').
terms(uncle, 'FB').          terms(uncle, 'MB').
```

and so on. These can be used where desired to either classify output terms into emic categories or to generate search terms from emic categories.

Prolog is also an excellent language for working with terminological properties themselves, although a treatment of this here is beyond the scope and word limits of this book. See Read and Behrens (1989,1992) for a description of their program KAES, which assists the user in determining algebraic descriptions of relationship terminologies.

7.4
ADDENDUM: DRAWING KINSHIP DIAGRAMS

Drawing kinship diagrams automatically is a non-trivial task, not because it is difficult to draw *a* diagram, but because it is difficult to draw the diagram you want. As Read and Behrens (1992) suggest, drawing diagrams using a drawing program (Section 5.2.2) is sometimes the easiest way to get a diagram of a particular sort.

However, for all their flaws, computer-generated diagrams have considerable attraction, and considerable value for some research purposes, since although they might not be suitable for publication, they can relate a lot of information and they can be drawn quickly.

There are no generally available programs at present for plotting kinship diagrams, although quite a number have been written to assist in research. The basics are not difficult, however, and it is often possible to get some assistance in developing such a program.

Programs 7.6(a) and 7.6(b) together form a small Prolog program for drawing simple kinship diagrams. You will need to be fairly familiar with Prolog to follow it, though we have covered most of the features in it in the previous examples. It assumes the same data structures for relationships as developed in this chapter. It is intended simply as an example and as a basis for more elaborate programs, since it illustrates the two basic operations: positioning people (Program 7.6(a)) and then drawing them at these locations and connecting them with appropriate lines (Program 7.6(b)).

Program 7.6(a) is fairly portable, which means it should work in almost any version of Prolog. It is also fairly general, and is likely to resemble any program routine which positions. One problem it does not deal with adequately is the placement of spouses whose nuclear families lay in the scope of the diagram, since in some cases they should be placed with the nuclear family and in others nearer the spouse. This can be added on a basis which reflects the analytic biases your particular analysis requires. The program makes no assumptions about screen or paper size, instead locating people on an abstract unit grid. It is up to the drawing program to translate these grid coordinates into actual screen or paper locations.

There are only two predicates that you need access, *clr_loc* and *calc_all*. clr_loc simply clears out the results of any previous calculations, so that a fresh set of locations can be set. *calc_all* is called with the form:

calc_all([List of Sibset IDs], Generation, Start width, End width).

where the first argument is a list of sibsets to calculate, *Generation,* the relative generation of these sibsets, which should all be at the same level (e.g. '0', '1', '2')- A second call can be issued for other unrelated sibsets of a different relative generation. At the first call, *Start width* should be '0', and for subsequent calls should be the value of *End width* for a previous set of sibs (in the same diagram). This program makes considerable use of recursion (Section 7.3.3) to trace the links of descent and affinity. These recursive predicates (such as *calc_sibsets*) follow a common pattern in Prolog, consisting of at least two definitions of the predicate and, where the first matches the terminal condition for the recursion, breaking the cycle. This is usually by finding an empty list ('[]'), since recursion is most often used with lists.

Program 7.6(b) draws a diagram based on the calculations in Program 7.6(a). Although it is illustrative, it is specific to a particular 'dialect' of Prolog, MacProlog. This lack of generality results from the means of drawing, which tends to be idiosyncratic by program, although most will have equivalent drawing operations. In this example the first three predicates define the symbols for females, males and neuter genders, which are drawn in the predicate *plot_sym*. The *sibline, spouseline* and *descentline* predicates define the line shapes for these lines, which are drawn in *plot_ sibline, plot_ spouseline,* and *plot_ descentline* respectively. MacProlog does its actual drawing with a built-in predicate, *add_pic,'* which uses these definitions. You will have to implement something compatible with your version of Prolog, or another language if you translate the algorithm.

The rest of the program is fairly portable, except for *to_ grid,* which translates the abstract unit grid of Program 7.6(a) to a unit compatible with the drawing program, usually in terms of absolute pixels or dots on the screen or printer. In this case I used thirty-five pixels for the vertical translation, which is about 1/2 inch on my screen, and thirty-six pixels for the width.

Plot_all will make the entire diagram. Note the use of the *fail* predicate in *plot_people* to step through all the locations. This is a common mechanism in Prolog because it lacks a general 'loop' operation. This works by getting each value, plotting it and then failing, which causes Prolog to try the next value of *location*. When all the locations are dealt with, the next definition of *plot_ people* is tried, which is designed to simply succeed, so that the remainder of *the plot_all* predicate can be interpreted. *plot_sibsets* works in a similar manner.

Program 7.5(b) Program modifications to Program 7.5(a) to control search using parameters

```
relation(Rel,Ego,Gen,Link, Affine,Collateral,Ascending,Descending) :-
    nonvar(Gen), stringof(Gens,Gen), relations(Rel,Ego,Gen,Link,Gens).
relation(Rel,Ego,Gen,Link, Affine,Collateral,Ascending,Descending) :-
    var(Gen),
    gen_rel(Gen, Affine,Collateral,Ascending,Descending),
    relation(Rel,Ego,Gen,Link).

[The relations predicate is defined as in Program 7.4c. valid_rel as in Program 7.5a]

gen_rel(Rel1, Affine,Collateral,Ascending,Descending) :-
    valid_rel(Rel1).

gen_rel(Rel1, Affine,Collateral,Ascending,Descending) :-
    simple_rel(R),
    valid_trans(R,Affine,Collateral,Ascending,Descending, A1,C1,As1,De1),
    grel1(R,Rel,A1,C1,As1,De1),
    stringof(Rel,Rel1),
    not(valid_rel(Rel1)).

grel1(R,[R],Affine,Collateral,Ascending,Descending).

grel1(R,[R|Rel],Affine,Collateral,Ascending,Descending) :-
    transition(R,N),
    valid_trans(N,Affine,Collateral,Ascending,Descending, A1,C1,As1,De1),
    /* fails if not valid*/
    grel1(N,Rel,A1,C1,As1,De1).

valid_trans('W',Affine,Coll,Ascend,Descend, A1,Coll,Ascend,Descend) :-
    Affine \= 0,
    A1 is Affine – 1.
valid_trans('H',Affine,Coll,Ascend,Descend, A1,Coll,Ascend,Descend) :-
    Affine \= 0,
    A1 is Affine – 1.

valid_trans('B',Affine,Coll,Ascend,Descend, Affine,C1,Ascend,Descend) :-
    Coll \= 0,
    C1 is Coll – 1.
valid_trans('Z',Affine,Coll,Ascend,Descend, Affine,C1,Ascend,Descend) :-
    Coll \= 0,
    C1 is Coll – 1.

valid_trans('F',Affine,Coll,Ascend,Descend, Affine,Coll,As1,Descend) :-
    Ascend \= 0,
    As1 is Ascend – 1.
valid_trans('M',Affine,Coll,Ascend,Descend, Affine,Coll,As1,Descend) :-
    Ascend \= 0,
    As1 is Ascend – 1.

valid_trans('S',Affine,Coll,Ascend,Descend, Affine,Coll,Ascend,De1) :-
    Descend \= 0,
    De1 is Descend – 1.
valid_trans('D',Affine,Coll,Ascend,Descend, Affine,Coll,Ascend,De1) :-
    Descend \= 0,
    De1 is Descend – 1.
```

Program 7.6(a) Program to calculate positions on a genealogical grid for specified sibling groups

```
% Calculate positions of a rooted ancestor tree in Grid Coordinates
calc_sibset(Spouse_Id, Gen, In_width, Out_width) :-
   get_sibs(Spouse_Id,Sibs),
   calc_people(Sibs,Gen,In_width,Out_width, Locs),
   set_sib(Spouse_Id,Locs).

calc_people([ ],_,Width,Width,[ ]).
calc_people([Person|Rest],Gen, In,Out, [Loc|Slocs]) :-
   calc_person(Person,Gen,In,O1,Loc),
   calc_people(Rest,Gen, O1,Out, Slocs).

calc_person(Person, Gen, In, Out,Loc) :-
   get_spouses(Person,Spouses),          % get all spouses past and present
   G1 is Gen + 1,
   calc_sibsets(Spouses, G1, In, O1),    % calculate for each set of children
   calc_loc(In, O1, R),                  % terminal nodes are treated differently.
   count_spouses(Spouses,N), Out is O1 + N*0.5 + 0.5,
   Loc = (Person, G1, R), set_loc(Loc),!.

count_spouses([ ],0).
count_spouses([X|T],N) :- count_spouses(T,N1), N is N1 + 1.

calc_loc(In, In, In).                    % terminal node case.
calc_loc(In, Out, R) :-                  % not a terminal case e.g. descendents
   R is (Out – In) / 2 + In – 0.5.       % centre over descendents with room
                                         % for spouse(s)
calc_sibsets([ ],_,Width,Width).
calc_sibsets([Sibs|Rest],Gen, In,Out) :-
   calc_sibset(Sibs,Gen,In,O1), calc_sibsets(Rest,Gen, O1,Out).

/* the following definitions use findall, which should be fairly general. If it is not
   present in your version of Prolog, look for a function which collects all
   solutions to a clause. bagof will usually do, setof will not, as it sorts the
   results, losing the birth order if you entered the siblings by birth order */

get_sibs(Spouse_Id, Sibs) :- findall(X,sibset(Spouse_Id,X),Sibs).
get_spouses(Person, Spouses) :- findall(Id,spouse_info( _,Person,Id,_ ),Spouses).

set_loc((Person,Gen,Col)) :-
   not(location(Person,_)),              % don't set if location is already set
   assertz(location(Person,(Gen,Col))),!.
set_loc((Person,Gen,Col)) :-
   location(Person,(Gen,C1)), C1 \= Col,!,  % Reset if location is already set
   retract(location(Person,(Gen,C1))), assertz(location(Person,(Gen,Col))).
set_loc((Person,Gen,Col)).               % if is already in database then succeed

set_sib(Sib_id, Sibs) :-
   not(sib_loc(Sib_id,_)),               % don't reset if location is already set
   assertz(sib_loc(Sib_id,Sibs)).
set_sib(Sib_id, Sibs).                   % if is already in database then succeed

clr_loc :-
   retractall(location(X,(Y,Z))),        % removes all location-3 clauses
   retractall(sib_loc(X,Y)).
```

Program 7.6(b) MacProlog program to plot genealogical data calculated in Program 7.6(a)

```
femalesym(Y,X,trans(Y,X,circle(5,0,5))).
malesym(Y,X,trans(Y,X,poly([(0,0),(9,5),(9,-5)]))).
neutersym(Y,X,trans(Y,X,poly([(0,0),(6,6),(12,0),(6,-6)]))).
sibline(Y,X,D,L,trans(Y,X,lines([(0,0),(L,0),(L,D),(0,D)]))).
spouseline(Y,X,D,L,La,Lb,trans(Y,X,lines([(9,0),(La,0),(La,D),(Lb,D)]))).
descentline(Y,X,D,trans(Y,X,line((-9,D),(-18,D)))).

plot_all :- clear_lines, plot_ people, plot_sibsets.

plot_people :-
    location(P,(Y,X)), plot_sym(P,(Y,X)), fail.
plot_people.

plot_sym(P,(Y,X)) :-
    female(P), to_grid(Y,X,Y1,X1), add_pic(femalesym(Y1,X1)).
plot_sym(P,(Y,X)) :-
    male(P), to_grid(Y,X,Y1,X1), add_pic(malesym(Y1,X1)).
plot_sym(P,(Y,X)) :-
    not(male(P)), not(female(P)),
    to_grid(Y,X,Y1,X1), add_pic(neutersym(Y1,X1)).

plot_sibsets :-
    sib_loc(S_id,Sibs), sib_lines(Sibs),
    descent_line(Sibs), spouse_line(S_id,Sibs),
    fail.
plot_sibsets.

sib_lines([]).
sib_lines([(P1,Y1,X1)|[(P2,Y2,X2)|Rest]]) :-
    plot_sibline((Y1,X1),(Y2,X2)),
    sib_lines([(P2,Y2,X2)|Rest]).
sib_lines([(P,Y,X)|Rest]) :- plot_sibline((Y,X),(Y,X)).

plot_sibline((Y1,X1),(Y2,X2)) :-
```

```
    D1 is X2 – X1, to_grid(0,D1,_,D),
    to_grid(Y1,X1,Y,X),
    add_pic(sibline(Y,X,D,-9)).

spouse_line(S_id,Kids) :-
    spouses(S_id,P1,P2,_),
    spouse_loc(P1, P2, L1, L2,Kids),!,
    plot_spouseline(L1,L2),!.
spouse_line(_).

spouse_loc(P1,P2,L1,L2,_) :- location(P1,L1), location(P2,L2).

spouse_loc(P1,P2,(Y,X1),(Y,X),Kids) :-
    not(location(P1,_)),
    location(P2,(Y,X)), adj_loc(Y,X,X1,Kids),
    plot_sym(P1,(Y,X1)), set_loc((P1, Y,X1)).

spouse_loc(P1,P2,(Y,X),(Y,X1),Kids) :-
    not(location(P2,_)),
    location(P1,(Y,X)), adj_loc(Y,X,X1,Kids),
    plot_sym(P2,(Y,X1)), set_loc((P2, Y,X1)).

adj_loc(Y,X,Xout) :- Xout is X + 0.5, not(location(_,(Y,Xout))),!.
adj_loc(Y,X,Xout) :- X1 is X + 0.5, adj_loc(Y,X1,Xout).

adj_loc(Y,X,Xout,[]) :- adj_loc(Y,X,Xout).
adj_loc(Y,X,Xout,Kids) :-
    descent_span(Kids,(Y1,X1),(Y2,X2)),
    Xout is X1 + (X2 – X1)/2 + 0.25,
    not(location(_,(Y,Xout))),!.
adj_loc(Y,X,Xout,Kids) :- X1 is X + 0.5, adj_loc(Y,X1,Xout).

plot_spouseline((Y1,X1),(Y2,X2)) :-
    D1 is X2 – X1, max(Y1,Y2,Ymin),
    to_grid(Ymin,X1,Y,X), to_grid(0,D1,_,D),
    to_grid(Y1,X1,Ya,Xa), to_grid(Y2,X2,Yb,Xb),
    store_line(Y,Xa,D,17,L),
    La is Ya-Y+L, Lb is Yb-Y+L-(L-18+9),
    add_ pic(spouseline(Y,X,D,L, La,Lb)).
plot_spouseline(_,_).

min(A,B,A) :- A =< B. min(A,B,B) :- B <  A.
max(A,B,A) :- A >= B. max(A,B,B) :- B >  A.

store_line(Y,X,D,L,L2) :-
    Y1 is Y + L, /* get the offset of the actual line */
    X1 is X + D, /* get the endpoint */
    do_store_line(Y1,X,Y1,X1,L,L2).

do_store_line(Ya,X1a,Yb,X1b,L,L) :-
    not(line_k(Ya,X2a,Yb,X2b)),!,
    assert(line_k(Ya,X1a,Yb,X1b)).

do_store_line(Ya,X1a,Yb,X1b,L,L) :- line_k(Ya,X1a,Yb,X1b),!.

do_store_line(Ya,X1a,Yb,X1b,L,L) :-
    line_k(Ya,X2a,Yb,X2b),
```

```
    (X1a > X2b; X1b < X2a),!,
assert(line_k(Ya,X1a,Yb,X1b)).

do_store_line(Ya,X1a,Yb,X1b,L,L2) :-
    Y1a is Ya + 3, Y1b is Yb + 3, L1 is L + 3,
    do_store_line(Y1a,X1a,Y1b,X1b,L1,L2).

clear_lines :- retractall(line_k(A,B,C,D)).

descent_span([(_,Y,X)],(Y,X),(Y,X)).
descent_span([(_,Vert,First)|Sibs], (Vert,First), Last) :-
    descent_span(Sibs,(Y1,F1),Last).
descent_line(Sibs) :-
    descent_span(Sibs,First,Last),
    plot_descentline(First,Last).
descent_line([]).

plot_descentline((Y1,X1),(Y2,X2)) :-
    D1 is (X2 – X1)/2, to_grid(0,D1,_,D),
    to_grid(Y1,X1,Y,X), add_pic(descentline(Y,X,D)).

to_grid(Y1,X1,Y3,X3) :-
    Y2 is Y1 * 35, X2 is X1 *36,
    int(Y2,Y3), int(X2,X3).
```

Chapter 8
Computer-based simulation and modelling

8.1
SIMULATION

When we apply a rule or set of rules to a set of input information, we derive a set of results corresponding to the inputs. The process of deriving this set of instances of applying the rule can be called simulation. The rule is a model which describes relationships, the application of the rule to generate an outcome is a simulation.

For example, if we take a rule adapted from Islamic Shariat law:

> If *talaq* is said three times in succession to the wife before two male witnesses the marriage is dissolved, otherwise the marriage continues.

and apply this rule to the information *'talaq* was said twice to the wife before two male witnesses' we arrive at the result that the marriage continues.

Simulations of this sort are nothing new to anthropologists—we do them all the time in our head or on paper to validate our analyses against data, to explore the properties of our models or to extrapolate our models to new situations. Such simulation, predating computers, has been used in anthropology at least since the nineteenth century (Mulvaney 1970). Simulation, with caution and reservation, is used to observe rituals, ceremonies and activities which for some reason cannot be observed in the ordinary course of fieldwork (Clammer 1984: 72–3; Ellen 1984b: 274). Indeed, formal interviews and 'set-ups' (Jackson 1987: 41) meet this sense of simulation to some extent. Finally, we practice simulation each time we 'play out' our models and analyses in our minds or on paper, testing against observed data and evaluating the results.

The use of computers for simulation modelling was one of the first encounters that social anthropologists had with computing (Kunstadter *et al.* 1963; Gilbert and Hammel 1966), not only because simulation met more or less the conception of what computers did in the early 1960s, but also because anthropologists at that time were beginning to explore the use of more sophisticated models and attempting to apply a more systematic perspective to anthropology.

8.1.1 Introduction

Simulation is a kind of modelling which is useful for a wide range of problems and situations. It has applications to both quantitative and qualitative problems with either very good data or very little data. It has important implications for basically non-experimental disciplines such as social anthropology, providing a means of exploring problems which could never be observed to order. Simulation is not a panacea for all of our problems, but it can be an important tool for the social researcher aware of its limitations.

Simulations are distinguished from other kinds of models more in terms of goal than form. Simulations are typically used for problems which are seen as complex and intractable, where no direct means of evaluation are known or the conventional means of evaluation is extremely difficult to execute, or which requires interactive decisions by the investigator during the course of the model.

In social anthropology the most common (and successful) simulations have been based on the interaction of models of prescriptive or preferential marriage, incest or other social phenomena with either demographic models or ecological models (or both) (Kunstadter *et al.* 1963; Hammel and Gilbert 1965; Coult and Randolph 1965; MacCluer and Dyke 1976; Black 1978; Buchler *et al.* 1986). The fundamental idea underlying these simulations is to investigate the performance of social models in context with 'well-understood' models, including the ethnographic model of collection.

Although simulations can be quite abstract and analytic, most anthropologists tend to favour those which are fairly concrete. One reason for this is the emphasis of social anthropology on structural relationships between individuals. If you are investigating the feasibility of literal prescribed matrilateral cross-cousin marriage (c.f. Kunstadter *et al.* 1963), then you must usually simulate a population as a set of people, not as a simple aggregate. Each simulated person must have at least a mother and father, an age, a gender, a marital status, be subject to birth, marriage and death and have, in some cases, a history.

A simulation animates our models to produce data which we can use to evaluate these models. This is of course possible to do without computers, but is a very time consuming effort. Although most simulations have been applied to theoretical situations where simulation was most useful precisely because it was not possible to observe these directly, in the past few years simulations have been applied back to the field with promising results.

Lansing (1991) describes a simulation which resulted from his fieldwork in Bali, regarding the role of water temples and the rituals associated with these and the regulation and conservation of irrigation water for rice cultivation and, more controversially, their role in pest control. Although a large part of the simulation related to ecological parameters, the overall significance depended heavily on ethnographic data relating to how the water temples functioned ritually as well, and how information flowed from the water temples to the peasants who used irrigation water for their crops. It appears that one of the results of the simulation project was the provision of a basis for reversing official policy towards the water temple system by the state and development agencies, which

are now recognized by the state and 'have regained informal control of cropping patterns in most of Bali' (Lansing 1991: 125).

Kippen (1988a) applied a novel version of simulation, using a production system/expert system (Section 8.2) to represent indigenous knowledge about improvising *tabla* music, animating this model and creating not a literal set of recordings but an improvizational 'performance' by Kippen's model; literally something new but conforming to a pattern which his expert consultants (*tabla* musicians) could make judgments about, criticize and set a context for Kippen to elicit new information on which to base modifications to the expert system rules.

In the past we could argue that there was no real way to produce a directly testable model. We cannot yet produce formally provable models, but there is no reason, though, why, at a micro-level, we cannot make statements about what we believe we know, and evaluate these with respect to what we think should be the outcome. Analysis should at least be subjectable to a test of the internal consistency of the representation, regardless of how we want to argue about the external reliability or lack thereof.

8.1.2 Uses of simulation modelling

Simulation can be defined in either a broad or a narrow sense. In the narrow sense simulation is a model where we are attempting to describe the behaviour of a system by incrementally and interactionally applying a number of models against some starting situation. In the broader sense, it is any kind of computer model within which some structure is being modified along one or more dimensions. In the usual case one dimension is time, but this is neither sufficient nor necessary for a simulation; it is the modelling of any complex system where observation of change, or an incremental process, is central. Both of these descriptions have in common the modelling of some structure under modification or transformation; the behaviour of some data object along one or more dimensions of change.

Traditionally simulation has been a technique used for quantitative analysis. As with computing techniques in general this is due to the historical development of computing and constraints on our knowledge of how to represent models and information of a qualitative and symbolic form. Designing a simulation involves translating the essential aspects of pre-existing models into a form which can be implemented on a computer so that we can monitor the interaction of the models.

8.1.3 What is a computer simulation?

Despite my assertions in Section 8.1.2, the question of what a computer simulation is, unfortunately, not such an easy question to answer from the literature. As with many terms in the literature, computer simulation is not so much defined as described. Johnson 'defines' a computers simulation as

> a computer program that defines the variables of a system, the range of values those variables may take on, and their interrelations in enough detail for the system to be set in motion to generate some output. The main function of a computer simulation is to explore the properties and implications of a system that is too complex for logical or mathematical analysis…A computer simulation generally has an ad hoc or 'homemade' quality that makes it less rigorous than a mathematical model.
>
> (1978: 186–7)

Nardi offers as a description:

> A computer simulation model…[provides] the investigator with a simplified analogy…for the purpose of better analysing and understanding…[some] phenomenon…it focuses on conducting experiments on a computer in which mathematical or logical operations describing the behaviour of a system over time are of primary importance…. Its very purpose, in fact, is the analysis of change over time…. Computer simulation is a powerful technique, capable of handling large numbers of variables representing complex systems and of simulating the operation of these variables over many cycles.
>
> (1980: 38)

Davies and O'Keefe, in a programming guide for simulation, suggest:

> When the word [simulation] is used by computer scientists, statisticians, and management scientists, they normally refer to the construction of an abstract model representing some system in the real world. The simulation describes the pertinent aspects of the system as a series of equations and relationships, normally embedded in a computer program.
>
> (1989:1)

These descriptions of simulation might be adequate for many purposes, but it is worth a closer look at a more structural definition of simulation in the context of how anthropologists have used them and might use them, if for no other reason than to provide a basis for designing simulations and interpreting the results, a subject treated with extreme coyness in the literature.

Specifying the structure of a simulation is very much like specifying any problem for computer treatment: we have to visualize what we want to get out of a simulation and devise a structure that will fulfil this goal. Most descriptions of simulation, especially those by anthropologists, suggest that a simulation model is:

- used to model systems too complex to model with ordinary analytic models
- comprised of logical or mathematical operations on variables
- able to represent a large number of variables and operations
- holistic in orientation, often used to model aspects of entire systems
- oriented towards representing processes

- oriented towards exploration and experimentation

The first two items suggest that, although simulations are composed of analytic models, simulations are not necessarily analytic models (Johnson 1978: 186–7). In other words, simulations can be informal models defined partly in terms of formal models.[18] The third and fourth items reflect a view that simulations are suitable for modelling complex large-scale systems as well as simple small-scale systems (Johnson 1978: 187; Nardi 1980: 38), and are thus 'capable of greater realism' (Johnson 1978: 187). The fifth item expresses a pervasive attitude that simulations model processes. It is more accurate to say that simulations produce their results from one or more processes. These processes may be used as part of a model of systemic processes, or they may simply be artefacts of the simulation.[19] Most simulations have been used to model processes, but incorporate both relevant and artefactual processes. The last indicates a pre-analytic bias in the use of simulation; simulations are oriented towards producing model data for analysis, not solutions (Buchler *et al.* 1986: 112), although the computer implementation of a simulation model may include analytic models which apply to the model generated data, and simulations may be used in some cases when no other solution is plausible or possible (Dyke 1981: 204).

These properties suggest a number of possible structures, but are not complete. What distinguishes a simulation model from any other model form is not so much the type of model but what we do with the model. In the case of simulations we are interested in the behaviour of a model; instances of application of a model. Simulations do not have solutions in the conventional sense. The most appropriate purpose of a simulation is to generate data, representing the interaction of the models under simulation. The value and purpose of a simulation follows from what is done with this model data (Dyke 1981: 204).

Extending this, I propose a more general structure for computer simulations. Abstractly, a simulation model consists of at least one structure, at least one operation which might act on the structure(s) and at least one opportunity to apply operation(s) to structure(s) (Figure 8.1). It is the applications of one or more models to create one or more instances. An operation may or may not be based on an analytic model; it can be quite *ad hoc*. A simulation is at least one instance of an application of operation to structure. This definition does not differentiate between the application of analytic models, such as a discriminant function derived from social data, and less formal models, such as those derived from so-called qualitative analysis of social data.

8.1.4
Types and uses of simulation

There are a number of approaches to simulation depending on the purpose to which the simulation is intended, the models available and the information available. However, the general form of a simulation is fairly regular. Simulations are almost always used to validate, explore or extrapolate the properties of a

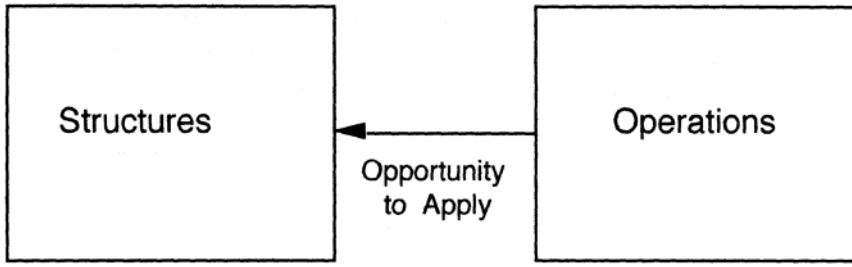


Figure 8.1 Abstract model of simulation

model of some system. The basis of simulation is modelling the behaviour of a model, and thus it is possible to have a very simple simulation

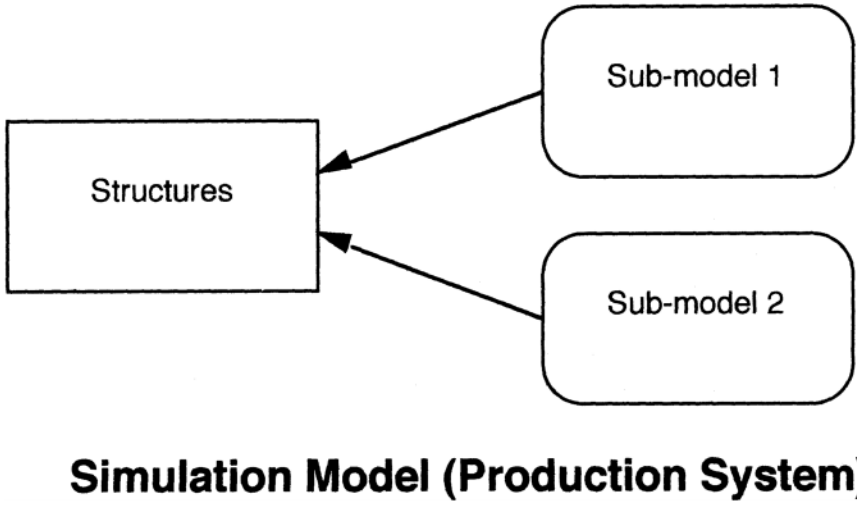


Figure 8.2 Distributed simulation model

consisting of a single model operation applied to a single model object (or structure), However, a simulation is usually a model of a system consisting of at least two sub-models, which interact either by one being directly influenced by the other, direct mutual influence, or where both operate on at least one common object (Figure 8.2).

One general motivation for simulation is that we may lack the means to substantively describe or characterize the interaction of these sub-models using either prose or formal devices, but we can observe their interaction case by case. Simulation is then suitable for those situations which are beyond our normal means to model in a concrete fashion. It is notable that simulations are widely used in applications of the physical sciences in engineering, for although these disciplines have very good models for describing the local interaction of variables, they too, in general, lack the means to describe or predict the interactions that a

complex system might have. Thus airplanes are not built on the basis of first principles of physics, but are designed using these principles, and the designs are refined, first using computer simulations and then using physical simulations such as a wind tunnel.

There are two basic types of simulation components, stochastic and deterministic, which roughly correspond to Lévi-Strauss's distinction between statistical and mechanical models (Lévi-Strauss 1963: 275–81).

Stochastic simulations generally have a probabilistic component, though not usually unconditioned probabilities (e.g. sampling from the normal distribution, Poisson distribution or even catastrophe spaces where the probability density changes with different paths or other spaces). Stochastic simulations, generally, must be performed a number of times until there is an adequate sample of results to evaluate, since any single 'run' of a simulation will have one of many possible outcomes. This sort of simulation is often used as a component in demographic or ecological simulations where many of the components are modelled using statistical criteria. They are especially useful where much of the data upon which they are based has been amenable to statistical analysis, where the local behaviour of many systems is only understood in statistical terms or where it is convenient to model many of the components using statistical or probabilistic models, which is desirable much of the time, even if it is not in theory necessary. It is, for example much easier to model rainfall or warfare with a simple probabilistic model (sampling from an empirical distribution), rather than going to the trouble of an elaborate model which is itself extremely complex simply because we want to estimate the impact of floods or the loss of male population due to warfare as an independent context for our central problem.

Deterministic simulations are those where one solution set exists for a given input situation. Its primary use is to extrapolate and evaluate outcomes given hypothetical inputs or to examine the interaction of a number of inter-dependent deterministic models. Depending on the form of the model used, it may not be by some definitions a simulation proper, but I am including any animation of models within this category.

Polyvalued simulations have more than one outcome for a given starting point, but probability plays no identifiable part in the multiplicity of outcomes (either due to lack of knowledge or because we are not concerned with probability); there are simply many outcomes, representing operations which are not functions. With this type of simulation we are usually concerned with the set of solutions as a whole. This kind of simulation could be used to explore all the possible marriages that might exist in a specific population, and the different structural outcomes of each marriage. This allows us to at least determine the boundaries of a system. It is useful in situations where a number of different rules could apply in a given context.

What can we gain from using computer simulations in our research? Simulation is appropriate where we can reasonably model the circumstances and context of some complex behaviour, and want to explore and evaluate models of that behaviour. For example, we are sometimes concerned with the plausibility of some practice of some stated rules or preferences, say a preference for marriage to FB child. One of the ways we can explore a system of this sort is to examine some model under various known circumstances to compare our own

data with. In the above case we can model a population demographically, state some rules, preferences and conditions for marriage and examine some of the outcomes of applying these to the model population. It is important when using simulation as a part of analysis to avoid two errors posed by Dyke (1981: 202–3). His *methodological error* refers to the process of elaboration of simulations to improve their 'realness' to the point that it is difficult to ascertain the impact of any of the simulation elements. His *heuristic error* refers to concluding that 'life' mimics the simulation. The fact that a given simulation might conform very closely to observed situations does not argue that these operate on the same principles, only that there is some degree of logical similarity between the processes in the simulation and the processes of the simulated system (Dyke 1981: 203).

8.1.5 Qualitative simulations

Often in social anthropology we have data which is impossible to quantify or sample, where we cannot give precise counts or parameters, or even meaningful probabilities. Because of the nature of collecting ethnographic data, data is recorded as it arrives, and only contextual information can be systematically collected in a form that could represent a proper sample from which a probability could be estimated. However, these qualitative collections are the essence of ethnographic collection. They apparently serve most of the needs of social anthropologists and we do not tire of attempting to analyse them. Most simulation techniques are based on numerical and/or statistical criteria, and make little sense in the analysis of marriage rules, sections, totemism or dreams.

It is possible to simulate using only qualitative objects and structures, although the use and evaluation of these simulations are somewhat different, and in many ways more complex than quantitative simulations, especially with respect to validation. In a qualitative simulation we have, at best, categories as parameters (although these may be expressed based on a probability), and the simulation focuses on the relationships between categories and objects within the simulation; where the structure is as important, or more so, than the content.

There are two basic approaches for qualitative simulations. The first is fairly conventional, and simply consists of transforming an input structure into an output structure. This corresponds to the usual design of a quantitative simulation. The second approach is by resolution. With resolution we are simulating a system of rules and conditions and using the simulation to answer specific questions by resolving whether, given the information in the simulation, a specific situation can occur. In the former, we alter the input structure to generate different output structures, and then evaluate these output structures. With resolution, we hypothe size different possible output structures and the simulation tells us whether they are possible or not within the model of the simulation.

8.1.6
Design of simulations

Simulations usually require programming of some sort. Because any computing language that we might use has a limited set of resources available for representing data, information, relationships and transformations that can be applied to these, we must carefully design our simulation. Because we are first interested in a specific problem it is important that the design develops according to the needs of the research, and not from the needs of the computer. Although we will ultimately be limited by the resources available to us on the computer, it is better to make the simplifications and compromises at the later stages rather than the earlier ones of the design. In this way we have considered our requirements in detail, and can better judge how to solve the more mundane problems of implementation. Even if you will not ultimately write the programming code for the simulation yourself, you need to be able to describe it in the terms of stages A, B and C below for the programmer.

A At the earliest stages of the design we want a simple statement of the problem and the objectives for which we want to construct simulations. This should be a prose statement in human readable form which clearly states the meaning and intent of the elements that will be represented in the proposed situations. The purpose of this step is to begin to lay out clear criteria for judging the adequacy of later stages, since these should ultimately be a representation of this initial statement. As an example we begin with the following simple problem: we want to assess the effect of different constraints on mate selection on the structure of monogamous marriage in a closed population. Constraints will include age, status and group or kinship relations. The kinds of structures we are interested in include age structure, relatedness and the proportion unmarried. We shall follow a cohort for ten years.

B The second step is to select the conditions for a specific simulation that will contribute to the statement of purpose. For the first simulation, we shall select the effect of age constraints on marriage structure. Here we need to state fairly precisely exactly what we shall need to write for the simulation. In general we need a list of the agents or objects to which actions or transformations will be attributed, the relationships that can exist between different agents/objects, the rules of interaction and transformation and the kinds of information that will be required. Perhaps more important we need to specify what conditions will constitute the stages of the simulation, what information we need to extract from the simulation, and when. For the example we can use:

Agents: people.

Relations between people: marriage.

Constraints on marriage: male should be 18 or older, and female between 13 and 18, but at least two years younger than male. A male and female can only marry if both are unmarried.

Information required about persons: age, sex, marital status. Information derived about persons: proportion married.

C The third stage is to decide how to represent the elements described in stage B. Here we are getting a step closer to the translation into a computing language, but we still want to remain a bit aloof at this stage with respect to which programming language. However, all existing programming languages share a great deal with a general strategy for representation.

Explicit object definitions are usually represented as a set of categories, and instances of objects are usually represented as a set of values for these categories. For example, we might define the object type PERSON as SEX, AGE and MARITAL STATUS, and a specific instance of PERSON as male, 14, unmarried. In other words we define a data type PERSON which will describe how to access and interpret information about specific PERSONs. These explicit or simple objects are the basic means programming languages use to represent data. Some objects like PERSON consist of several categories, some consist of a single category (which is usually its name) and some have complex categories, which contain not simple values but a reference value that permits us to locate the information in another object. We shall look at references in the next example, but we could have a reference to spouse in the MARITAL STATUS category, instead of the simple married/unmarried value. In this way we would have access to information about spouse as well as ego. However, because of our simple objective in this case, to find the proportion of married to unmarried we do not require this information. Only information that is required need be considered, since our PERSON in this case is a model of a person as required for the objective.

Because of the simple means of representing the married/unmarried relation, we shall not have any interperson relationships, so the representation of the marriage relation is simple.

The constraints on marriage choice will be given as rules. In this case something like: IF age of male greater than or equal to 18 AND the female is between 7 and 5 years younger than male THEN marriage is ok, OTHERWISE not.

Finally we have to consider what kind of information we want to get out of the simulation. In this case it is proportion of married to unmarried males and females, and probably more specifically the proportion of eligible males and females. The former is fairly easy to accomplish: we need only count the number of unmarrieds before we attempt to marry them off and compare this with the number of unmarrieds afterwards. Specifying the proportion of eligible males and females is a bit more complicated, especially if we are strict about eligibility, since eligibility could be construed to be dependent on the prior existence of a male or female of the proper age. A weaker constraint would be to select 18 for males as the eligibility age, and 11 for females. This weakening is reasonable, since we are in a sense doing the simulation to find out the eligibility rate, but there is value in the stronger constraint as well, since an 18–, 19–, and 20–year-old male can marry a 13–year-old female, but a 20 year old could also marry a 15–year-old female, while the 18 and 19 year old could not. Thus the 20 year old can eliminate a possible mate for the younger males.

This latter point raises an important issue about the process of making the marriage decision. How are we to decide the order of choice among the males? This is an issue that emerges over and over in even the simplest simulations or even especially in the simplest simulations, since they often have the most simplified decision models. Sometimes we can decide to use a simple principle, such as oldest (or higher statuses etc.) choose first. It is best if there is some ethnographic evidence for these kinds of principles, but they can be used without if you are willing to accept the bias that is introduced. Another choice, especially if one is intending to run the simulation several times, is to randomly apply the decision, perhaps with a bias towards some principle, say older males are more likely to choose before younger, but not necessarily. In our simple example we shall choose the oldest first principle, but will examine the randomized approach.

8.1.7 Implementation

The examples are represented in the programming language Prolog (see Chapter 7 for description of its basic features; Bratko (1986) is a good introductory text). Prolog has a number of properties that makes it very useful for qualitative and non-deterministic simulations, and is weakest in quantitative simulations. There are some problems with Prolog, because although we shall find it relatively easy to represent the sub-models and their interaction in Prolog, it is often difficult or cumbersome to evaluate the results.

We can define our basic object type in Prolog by using a person fact; person (age,sex,status), which we repeat for every person in our population, where age is a numerical value, sex is male or female and status is married or unmarried. These could be read from a data file, created by an initializing program module (in Prolog a program module is called a *predicate*) or typed in. We could then use a very simple (and unrealistic) marriage rule, 'each unmarried male at or beyond the age of 18 will marry the first female encountered who is 11 years of age or older and who is between 5 and 7 years younger than the male'.

```
/* Database */
person(abdul, 24, male, unmarried).
person(rubina,18,female, unmarried).
/*…*/
person(zarina, 22, female, unmarried).

/* Rules */
marry_all :- /* marry all eligible people */
marry(Male, Female),/* marry a couple */
fail. /* forces evaluation of next couple */
marry_all. /* so that marry_all will succeed after all marriages */
marry(Id_male, Id_female) :- /* marry people if eligible */
eligible(Id_male, Id_female),
change_marital_status(Id_male,Id_female).
eligible(Id_male, Id_female) :- /* check eligibility for marriage */
```

```
person(Id_male,Age_male,male,unmarried),/* unmarried male*/
person(Id_female,Age_female,female,unmarried),/* unmarried
female */
age_check(Age_male,Age_female).
age_check(Age_male,Age_female) :- /* check to see if ages are
compatible */
Age_male >= 18,
Age_male—Age_female =< 7,
Age_male—Age_female >= 5.
change_marital_status(Id_male,Id_female) :- /* change from
unmarried to married status */
retract(person(Id_male,Age_male,male,unmarried)),/* remove old
entry */
assert(person(Id_male,Age_male,male,married)),/* add updated
information */
retract(person(Id_female,Age_female,female,unmarried)),/* ditto */
assert(person(Id_female,Age_female,female,married)).
```

The predicate *marry_all* will attempt to many everyone in the population according to the defined criteria. This does not mean that everyone who is eligible for marriage will be married in the end, because of demographic restrictions of the initial population. One problem with this example, especially from an anthropological perspective, is that all males and females are interchangeable with all other males and females respectively. There is no mechanism to take account of kinship or other relationships, not even such primitive aspects such as sibling-hood! We can accommodate this by adding avoidance for half and full siblings using a person data definition, person (Id, Age, Sex, Marital, Father, Mother):

```
marry(Id_male, Id_female) :- /* marry a couple */
eligible(Id_male, Id_female),
change_marital_status(Id_male,Id_female).
eligible(Id_male, Id_female) :-
is_male(Id_male),
not(is_married(Id_male)),
is_female(Id_female), not(is_married(Id_female)).
not(are_siblings(Id_male,Id_female)),
age_check(Id_male,Id_female).
[% the _ in the following definitions indicates that we are not using
the value for this position.]  is_male(Id) :- person(Id,_,male,_,_,_).
is_female(Id) :- person(Id,_,female,_,_,_).
is_married(Id) :- person(Id,_,_,married,_,_).
get_age(Id,Age) :- person(Id,Age,_,_,_,_).
get_father(Id,Father) :- person(Id,_,_,_,Father,_).
get_mother(Id,Mother) :- person(Id,_,_,_,_,Mother).
are_siblings(Id1,Id2) :- get_father(Id1,Father), get_father
(Id2,Father).
```

```
are_siblings(Id1,Id2) :- get_mother(Id1,Mother), get_mother
(Id2,Mother).
age_check(Id_male,Id_female) :-
get_age(Id_male,Age_male),
get_age(Id_female,Age_female),
Age_male >= 18,
Age_male—Age_female =< 7,
Age_male—Age_female >= 5.
change_marital_status(Id_male,Id_female) :-
retract(person(Id_male,Age_male,male,unmarried,Mf,Mm)),
assert(person(Id_male,Age_male,male,married,Mf,Mm)),
retract(person(Id_female,Age_female,female,unmarried,Ff,Fm)),
assert(person(Id_female,Age_female,female,married,Ff,Fm)).
```

From here we can elaborate the code further to include an absolute preference for FBD (e.g. if an FBD is available marry her, or else marry someone else) by replacing the marry predicate with the following predicates:

```
marry(Id_male,Id_female) :- marry_fbd(Id_male,Id_female).
marry(Id_male,Id_female) :- marry_other(Id_male,Id_female).
marry_fbd(Id_male, Id_female) :- /* marry folks */
is_fbd(Id_male,Id_female),
eligible(Id_male, Id_female),
change_marital_status(Id_male,Id_female).
marry_other(Id_male, Id_female) :- /* marry folks */
eligible(Id_male, Id_female),
change_marital_status(Id_male,Id_female).
is_fbd(Id_male,Id_female) :-
get_father(Id_male,Mf),
get_father(Id_female,Ff),
are_siblings(Mf,Ff).
```

From this we can see that we can alter and elaborate the model to represent what we want. For example, if we want to include the consideration of obligations, we first need a model of obligation, then a representation of obligation and finally a check to see at the time of the marriage decision if an obligation might affect marriage choice. Along the same lines, we might want in the code above to add a deference of the marriage decision until some upper age boundary, say 25. If the male is not already married at 25 he must marry someone, even if an FBD is not available. Or we could add a routine to see if there is the prospect of an FBD becoming available, and if she has an obligation to marry him. In other words, if we can model and represent some aspect of the situation, it can be included in the simulation. One obvious problem with the above example is that we have provided no method of actually monitoring or otherwise getting information about what is happening. Before the simulation is designed it is important to decide what information you are seeking to answer which questions. As with other computing applications, the simulation is a transformational method for

relating input to output. A difference here is that we are interested in the set of transactions that lead to this transformation.

Monitoring the simulation depends on what data is affected. In this case we have limited data to monitor, since all that is changing is the marital status and we are not recording who the marriages are to. For example, we can count the married and unmarried people by gender and age group using the following:

```
count_people(Gender, Low_age, Hi_age, Marital_status, Count) :-
assert(total(C)), fail.
count_people(Gender, Low_age, Hi_age, Marital_status, Count) :-
person(Name,Age, Gender, Marital_status, Father, Mother),
Age >= Low_age,
Age =< Hi_age,
total(C),
retract(total(C)),
C1 is C+1,
assert(total(C1)),
fail.
count_people(Gender, Low_age, Hi_age, Marital_status, Count) :-
total(Count).
```

The first definition of *count_people* initializes the count to zero. It then fails so that the second definition will be tried. The second definition matches the criteria you supply; e.g. 'count_people (female, 15,20,married,Count)'. It fails at the end so that the next person will be examined. The third definition simply reports the result and deletes the reference to the count. This structure works because Prolog always tries the next possible case if it fails. Since we record a case that matches before we fail, we can keep a count. *retract* and *assert* are Prolog predicates which add and remove 'facts' from its database.

If we were to keep track of who was married by extending the *person* structure, then we could also count the number of FBC who were married.

8.1.8
Building blocks for simulations

There are usually a number of different models at work in a given simulation, and indeed this could be taken as a functional definition of simulation: solving a problem by the interaction of at least two models. A simulation is then composed of a number of building blocks whose properties we more or less understand. These models are themselves arranged in a larger interacting model, which represents the larger context of these models. A simulation provides a proving ground for these sub-models. This complicates the general usefulness of simulations, since the results that arise from a simulation are only as good as all of the models that the simulation is built from. This can make the validation of the simulation very problematic indeed.

Validation of a simulation centres on two different aspects: the sub-models and an evaluation model. The sub-models must be independently validated to

establish that they behave as specified. The evaluation model is used to establish that the simulation does or does not have specific validity.

The results of any simulation is, of course, only a descriptive model. Any explanatory power that it might have must be argued based on points that are outside the simulation proper. In anthropology this is generally ethnographic data. As with a model, any simulation is a simplification of the system under study, and in many cases does not even represent any 'real' system at all, rather the simulation is intended to generate model data for an 'ideal' world, which we can then compare our data with, noting where it corresponds and departs from the ideal world. This is a useful technique, especially in the early stages of analysis, since it can be used to establish a sense of how important specific aspects of the context are to the analysis of the data.

The nature of these models used in the simulation depends very much on what the objectives of the simulation are. If all we require is a model that behaves correctly with no explanatory pretensions whatsoever, then our job is relatively easy. This is generally the case for any sub-model whose behaviour is independent, or can be modelled as independent, of the rest of the simulated context. This includes structures such as rain and weather in general, the passage of time and the presence or absence of game, fish, honey or other 'natural' resources. It can also include other sub-models, if they are not the principal object of study. Thus we can include crop yields in this category if we are not interested in the micro-mechanics of corn growth. What we care about in the simulation is that given specific environmental conditions, specific inputs and human labour we shall expect a corn yield. In many simulations all of the sub-models can be descriptive behaviour generators, if we are principally interested in their interaction.

8.1.9
Generating behaviour: independent events

If we simply need to generate events, such as rainfall, which are independent of other elements in the simulation then a statistical/probabilistic model is often best, especially if the simulation is one which will be run many times to establish its overall behaviour. Thus is usually the case because most social situations involve a great deal of uncertainty, and often the only sensible method of investigating them is to look at a range of solutions. If the event we want to model has a numeric value, such as rainfall, and we have several years of recorded data for rainfall, we can often simply use the mean and standard deviation of the rainfall to generate a value. If we only require a few categories, we can break the probabilities into a table and select from that. Different distributions suit different kinds of data. For example, disease events are often better sampled from a Possion distribution than a normal or binomial distribution.

For example, we can elaborate the simulation in Section 8.1.7 by operating it on an annual cycle. To do this we need to do three things. First we must age everyone one year, we must have some mortality and we must have some births. Aging is easy:

```
age_people :-
person(Name,Age, Gender, Marital_status, Father, Mother),
retract(person(Name,Age, Gender, Marital_status, Father, Mother)),
New_age is Age+1,
asserta(person(Name,New_age, Gender, Marital_status, Father,
Mother)),
fail.
age_people.
```

The others can be simulated in simple cases by applying probabilities of births to women of childbearing age, with appropriate weights for married and unmarried women, and applying mortality to each person based on age.

8.1.10 Evaluating results

The results of a simulation can be evaluated in a number of ways. If the simulation is basically an empirical one, which has a number of random or statistically generated events within it, we can often evaluate the results by using a statistical test such as x^2. Often, though, we are principally interested in using the data to establish some point for which we do not have direct data. Here we are exploring the structural possibilities given what we do know. Simulations are useful for 'what if situations, where we are attempting to extrapolate from what we do know to areas with which we have little or no data. One method of some use in evaluating simulations is examining its structural stability. This is useful where we are (sometimes grossly) estimating a number of values for the different models because the information is simply not available. This is common in ethnography, because out of necessity we collect an account of events that are idiosyncratic to the time which we are in residence, in most studies less than two years and often less than a year. We have, however, some confidence that the behaviour that we observe during our tenure in the situation is not simply idiosyncratic behaviour. The particular events and situations we observe are, but we assume that the responses to these are derived from general principles of the society, and this is usually the object of an ethnographic analysis. Simulation can give us an opportunity to validate some of these assumptions and analyses, because with simulation we can create contexts and situations that did not occur during our study. If the various 'solutions' we find in the social group are likely to represent general processes, we expect that they will work, and the social group will survive in a wide range of possibilities. For example, agricultural practices that lead to the loss of one crop in three are not likely to be considered successful, and probably represent at least a situation where we did not collect enough data. While we cannot be sure of our simulation model, we can establish the various limits which the simulation can adapt to, i.e. the various points at which it breaks down.

8.2 EXPERT SYSTEMS AND ANTHROPOLOGICAL ANALYSIS

The idea of using an expert system, a computer program that simulates a human expert (i.e. an informant), in anthropological analysis has been received by anthropologists with some interest, but with more caution (Davis 1984a: 3). This caution is justified because to most anthropologists the inner workings of the expert system are not known; they are black boxes. But anthropologists should be interested in a model that claims to represent and use human knowledge, if only to evaluate that model. This section describes some of the basic assumptions in contemporary expert systems, discusses their usefulness to anthropology and concludes that many existing expert systems are of limited interest to anthropologists, although the general model underlying expert systems can be used productively.

8.2.1 Introduction

Artificial intelligence (AI) is a multi-disciplinary area in which the goal is to represent intelligence (usually human intelligence) in the modelling environment of a computer. There has been research in AI since there have been computers. It was believed in the 1950s that 'just a few more years' would bring about a revolution in AI, but those few years have receded annually.[20] In the past decade there have been developments in AI that are considered by AI researchers (and others) to be partial successes, among these is the *expert system*. Expert systems are computer-based models that simulate human expertise in a specific area (domain), such as a subset of medicine (Shortliff 1976), exploratory geology (Duda *et al.* 1978) or education (Clancey and Letsinger 1981). Expert systems are claimed by AI researchers to be an important advance, and some claim implications about models of human representation of knowledge and mechanisms of inference (Barr and Feigenbaum 1982).

There is a small but growing literature on the use of expert systems in anthropology. Besides Kippen (1988a), described in Section 8.1.1, Brent (1988) has developed an expert system to assist in statistical analysis. Furbee (1989) describes an expert system for 'folk' classification of soil in the Colca Valley in Peru. Read and Behrens (1992) describe a simple expert system, adopted from Geoghegan (1971), which they developed in 1987 in which they modelled decision making about terms of address used by Bisayan speakers in the Phillipine Islands. Fischer and Finkelstein (1991) wrote a production system which simulated evaluating a potential marriage partner in an arranged marriage in the Panjab, Pakistan. Benfer *et al.* (1991) is a good anthropological introduction to expert systems.

8.2.2 Qualitative and quantitative analysis

Qualitative analysis can be defined as identifying qualitative structures, the states of those qualitative structures and the pattern of changes (transformations) in those states.[21] Quantitative methods can sometimes be used to aid this process, but usually qualitative methods are exclusively used for the analysis of qualitative data and structures for which quantities proper are difficult to define.

Thom (1975) argues that all quantitative analysis assumes a firm qualitative foundation. Before they measure, people must agree that there is something to be measured, and that is a qualitative judgment. Similarly, people must agree that the measure (metric) they use is appropriate and applicable to other phenomena.[22]

As an example, consider per capita income. It is apparently easy enough to agree on the structure, but the metric is another issue. If currency is used as a metric, a poor family in the United States would be a wealthy one in Pakistan. The metric can be further adjusted by considering cost of living, but an acceptable level of living in the United States is not equivalent to one in Pakistan. The problem is not difficult to understand qualitatively; there are different standards in these two places. The two countries' per capita income can be compared quantitatively, but the interpretation of the comparison is qualitative. The quantitative analysis is more difficult to reconcile, and indeed is undecidable without reference to qualitative structures in the two societies.

In most cases quantitative analysis depends on continuity. To quantify a phenomenon meaningfully it is usually necessary to assume that the relation between phenomena and metric can be described by a continuous relation,[23] since a primary goal of quantification is to provide a basis for comparison. For phenomena where the analytic focus is on states this is often misleading or impossible. In most social phenomena there is no continuous relation that can adequately describe the important qualitative relationships. As an example consider income and education. These are variables which are often given a quantitative definition in social research. They are relatively easy to define, and people generally measure income in currency, education in years. But linearity is often assumed and usually there will be a good correlation between them. But it will not be a perfect correlation, as one unit change in the independent variable will not result in some regular linear unit change in the dependent variable. Now this is not terribly shocking, since people do not expect that all the variation in one variable is to be explained by the other, but there is benefit in understanding the relationship between the variables by breaking the relationship into stages and examining the conditions for moving from one stage to the next. For instance, in the United States eleven years of education is minimally better than ten years, but twelve years is far better than eleven. This is due to the local structure of American education; eleven years is pre-graduation and twelve years is postgraduation. The graduating student has a qualitatively changed educational status, the pre-graduating student has not significantly changed status. This type of analysis helps to give a better account of interactions.

Another reason quantitative analysis must depend on qualitative analysis is illustrated in Figure 8.3. The graph shows hypothetical data g and two solutions

fitted to that data. Solution 1 is the better qualitative fit, as the relative shape appears to be the same as the data, but is not as good a fit quantitatively as Solution 2. Solution 2 fits well quantitatively, but probably describes a different underlying mechanism altogether.

8.2.3
Expert systems

An expert system is designed to simulate one aspect of a human expert; the ability to classify phenomena from a set of attributes. The expert system is a classification engine. It is a system that takes information about a particular case or instance within the domain of the system and produces a qualitative result (or

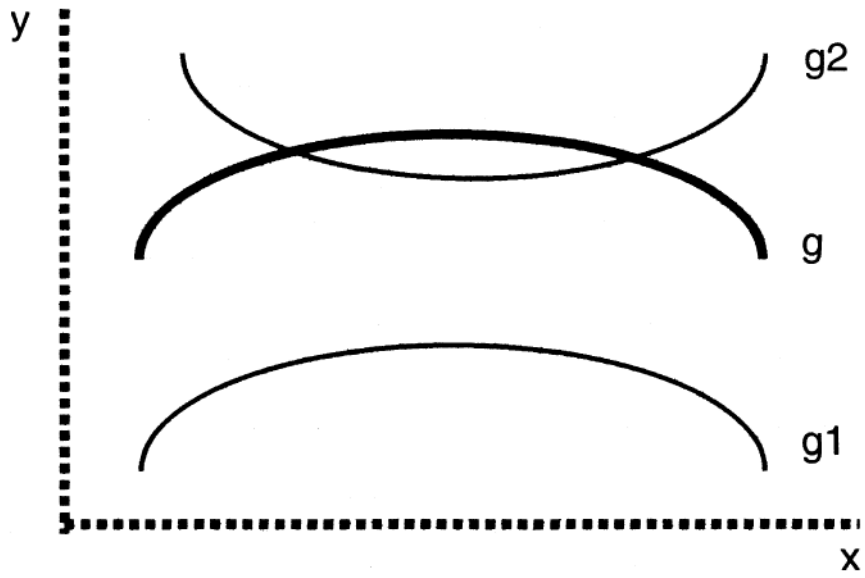


Figure 8.3 Two models of *g*
Source: Adapted from Thom 1975

goal state). It usually has incomplete information, and makes qualitative judgments based on this information. Expert systems are defined in terms of algorithms in a computer program *plus* relationships established by a human expert. This will be interesting to anthropologists if three conditions are met: the computer should arrive at the same conclusions as a native expert; it should arrive at the same conclusions as an anthropologist; and it should do useful jobs.

Expert systems, as a class of computer program, are currently designed to reflect a general model current within the artificial intelligence community; an expert system is not simply a simulation of human expertise, but must be implemented (on a computer) in a particular fashion; it is a product of an AI culture. Ideally an expert system has two primary components (see Figure 8.4):

The knowledge base

The knowledge base is essentially a set of rules describing relations between elements in the domain of knowledge. In the simplest form:

[condition(s) → outcome(s)]

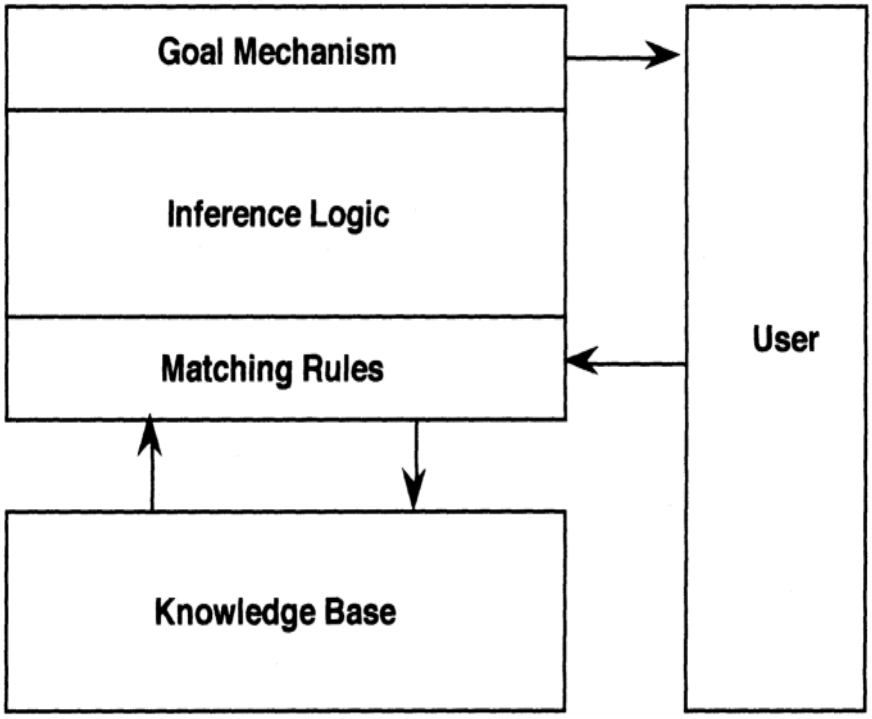


Figure 8.4 Expert system schematic

In spite of this notation causality is not assumed. The rules for deriving an outcome from a set of conditions are always formulated externally by an *expert* usually aided by a *knowledge engineer,* i.e. a specialist in transforming the expert's information into statements suitable for a knowledge base. The knowledge engineer stands to the expert as anthropologists do to their informants. The knowledge that is selected for inclusion in the knowledge base can have a variety of forms, depending on the form of the inference engine.

Most expert system designers consider it important that the rules be easily inserted, modified or deleted from the knowledge base, in any order. They usually consider the rules to be weakly connected: there is no sequencing information about the order in which they can apply and the only connections between them are the use of common terms of *reference*. Thus if one rule determines that a person's residence is patrilocal, and another rule can use that residence information to draw further conclusions, the rules are connected.

The inference engine

An inference engine is a method of using the rules in the knowledge base to derive a conclusion. Using the simple knowledge representation above this might take the form:

if **condition** then add **outcome** to the **context**

Where *outcome* is the conclusion if *condition* is true, and *context* is an area where knowledge is recorded to determine if conditions are true. An *outcome* is often part of another *condition* that matches another rule. In other words, the inference engine takes the rules provided by the knowledge base and uses internal rules of inference to draw a conclusion. The claim is that the internal rules are general to all inference. So the inference engine is a set of rules which are applied to the rules in the knowledge base.

The inference mechanism is thus critical to the outcome; it is responsible for any interrelation of elements beyond the rules in the knowledge base. It is usually based upon some variant of logic, such as first-order logic, fuzzy logic (Zadeh 1965), modal logic (Zeman 1973) or intuitionistic logic, and also usually employs some statistical mechanisms for measurement and classification.

The inference engine is intended to be based on a general model for using knowledge and should not have special knowledge about a particular domain. This model is claimed to be unlike the usual computer program/model structure because the specifics are separated from the methods. This distinction is made for at least two reasons.

1 It makes possible system expertise in different domains by modifying the knowledge base without modifying the inference engine.
2 AI researchers assume that in humans knowledge and inference are separate activities and that inference is prior to knowledge. Hence it is theoretically consistent to separate the two in the computer model.

(Derived from Barr and Feigenbaurn 1982)

In most existing expert systems the knowledge base and the inference engine are not terribly complex in design. The knowledge base determines the set of possible outcomes that the system can consider and the rules for arriving at those outcomes. Although this requires great effort on the part of the human expert and the knowledge engineer, the form of representation is quite simple.

In many systems both outcomes and rules have an objective or subjective probability associated with them, again derived from the human expert. The knowledge base consists of high-level structures derived through the formidable pattern matching and inference skills of humans.

An inference engine has three parts; an identification mechanism an evaluation mechanism, and a goal mechanism. The first two constitute the inference mechanism proper, and the third is for finding efficient paths to an outcome; it does not strictly affect the outcome (unless it is poorly designed) but it selects the *best* condition to request data on rather than requesting all possible conditions in the knowledge base. So the goal mechanism is a search pattern through the possible conditions that apply to a case, and it is the goal mechanism that gives

the expert system the appearance of performing like a human expert by requesting a minimum of information. The inference mechanism gives the expert system the judgment to announce a result consistent with the knowledge base. Most of the successful (externally validated) expert systems use some form of probabilistic model (often Bayesian) as the basis of the inference mechanism, using the probabilities associated with the knowledge base. One common goal mechanism works by finding the goal that is most likely to be true at the current time, and then finding the condition that will give the most information about that goal (as defined by the evaluation mechanism).

8.2.4
A simple example

Consider the factors that influence the marriages arranged by urban Panjabis of Lahore (Fischer 1991; Fischer and Finkelstein 1991). Marriages are arranged in the Panjab by the parents and other relatives of the potential groom or bride. The following factors (not necessarily in this order) appear to be the most important in the evaluation of a possible spouse.

1 *zat* (sometimes glossed as caste)
2 jihez (dowry)
3 intellect
4 education
5 *haq mehr* (bride deposit)
6 beauty
7 *izzat* (honour, respect, responsibility)
8 *baradarie* (clan)
9 *rishtidar* (relative)
10 distance (from natal home)

These are not Panjabi selection criteria, but an anthropologist's measurement or probe of the semantic domain of selection derived from what Panjabis say. In addition, the selection is influenced by the size of social networks and the availability especially of females, who are supposed to be invisible before (and after) marriage except to relatives.

The relationships between these measurements are quite complex, and they are evaluated relatively. For instance, if the *zat* of two candidates is *different,* then what constitutes *enough izzit* will be different in each case. In other words the state of *enough izzit* varies, depending on at least one other value. Amounts measured are not evaluable without other context; there is a high degree of relativity. Moreover, it is probable that different people have different selection models, and each person may have more than one.

To construct an expert system based on this situation we consider the following.

1 *What will the expert system do?* Give a statement of the suitability of possible marriage partners.

2 *How will the expert system do it?* This is a fixed solution (relative to a particular inference mechanism), since an expert system uses the same inference method regardless of the knowledge domain. Initially assume a simple mechanism; internal rules derived from examples of previously considered marriages given a suitability judgment by local experts. These rules can be derived using a statistical mechanism which weights the effect on each marriage of each of the factors. In essence the rules treat each factor as a dimension in a multi-dimensional space, and locate each qualitative state (suitable/not suitable) within that space, given a value for each axis. When the expert system is consulted, the evaluation mechanism will test to see if the input factors required by the inference engine are within a statistically significant distance from the internal rule-derived values. The goal mechanism will find the factor that makes the biggest difference in continuing evaluation, and ask for that information.

This is known as forward chaining because it works from factors to outcomes. Many current expert systems turn the above goal mechanism on its head or side, called backwards chaining and sideways chaining respectively. Backwards chaining is favoured for systems that have a large number of outcomes, much like the above example if all the individuals in the marriage universe are included as part of the knowledge base. In this type of system, the expert system would start attaching probabilities to each person in the base and finding information that would remove a person from consideration. This is called backwards chaining because it works from solutions to factors, and appears more purposeful (Nilsson 1982). In this case a person is the outcome rather than a simple yes or no. Sideways chaining works a bit on both principles, finding both weighted factors and weighted solutions.

3 *What kind of data will it require?* The data is dependent on the kind of inference mechanism used. In this simple case the data will be of the form:

marriage {value of factors 1–10}

where the value will have already been weighted by the human operator; in terms of *too little, too much, enough,* where appropriate, *yes, no, same* and *different*. The weighting in this example is assumed to always be from the son-giving side. This would give us a knowledge base like that in Table 8.1.

In consultation the expert system takes in the knowledge base, creates internal rules and answers the request, which would be for the suitability of a possible marriage. To derive an answer it asks the user to give values for some of the factors until it is possible to determine the qualitative result, and then makes a pronouncement, yes or no. Note that it cannot ask the suitability question itself, as this is the purpose of the system, but requires this for the knowledge base input to form the rules.

8.2.5 A knowledge-based example

The example in Section 8.2.4 is fairly easy to follow, but has little depth because of the immense amount of analysis that is needed to set it up; for it to work it must be told to seek the correct information (the selected criteria) and that is

Table 8.1 Example measurements for marriage model

Factor	*Marriage 1*	*Marriage 2*	*Marriage 3*
zat	same	different	same
jihez	enough	too low	too high
intelligence	enough	too high	too low
relative	yes	no	yes
education	too low	enough	too high
haq	too low	too high	enough
beauty	enough	enough	too little
izzit	enough	too high	too low
bradarie	same	different	same
location	too far	ok	ok
suitability	yes	no	yes

known only after analysis. It also fails to take into account any higher-level ethnographic or ethnological knowledge, it is a purely descriptive model with no explanatory power. Additionally, the particular method described is heavily committed to a particular model in the formulation of rules, and assumes that the results are linearly differentiable; i.e. that each state has a unique coordinate range in the multi-dimensional space.

Most expert systems incorporate higher-level knowledge in the form of explicit rules in the knowledge base. The previous example can be greatly improved in performance by adding rules of the following type to the knowledge base,

1 if *zat* is *same* then *izzat* is *enough*.
2 if *bradarie* is *same* then *izzat* is *enough*.
3 if *relative* is *yes* then *zat* is *same*.
4 if *relative* is *yes* then *bradarie* is *same*.
5 if *distance* is *too far* and *relative* is *yes* then *distance* is *ok*.

and so on. These kinds of rules add information about factors that cannot be taken into account in a regular, statistical method. One might ask why the entire system could not be based from rules like these, freeing the system of dealing with deriving rules from empirical data altogether. The answer is that one can, and most working systems do. However, although the rules appear to be 'higher level', they are no less empirical with respect to the expert system, and provide no explanation for the outcome that is not in the rules to begin with. This defect is usually overcome in expert systems by the expert and the knowledge engineer adding comments to each rule, so that when a user inquires about the reason a

particular conclusion has been reached, comments are displayed for each rule in a successful derivation of the conclusion.

We need not limit ourselves to such simple kinds of factors. For example, consider some rules adapted and simplified from Fischer and Finkelstein (1991):

if girl might be immoral then marriage is not a good risk

if mother is immoral then daughters are probably immoral

if girl is immoral then younger sisters may be immoral

if girl plays suggestive music then girl is immoral

believed: 'older sister of girl played suggestive music'

conclusion: 'marriage may not be a good risk'

Even in this simplified model it is clear that much of the complexity of the computing component is in the goal mechanism, which ideally has no analytic effect on the final outcome, whereas the inference mechanism is relatively simple, using models that are more or less in common usage in descriptive analysis. In spite of this many expert systems do often succeed in making judgments consistent with the human experts they are based on (Michie 1982). They achieve this by representing knowledge as a set of local models, made up of one or more rules, that are only weakly (and informally) interrelated, rather than by having a single large formal model of the expert's knowledge.

Of course, the degree of interrelation varies from system to system. For example, in most learning systems the initial set of structures it is told to learn about have been carefully selected to be independent of each other statistically. In input-rule-based systems, the rules will have been carefully selected. Most successful systems have undergone an enormous amount of tuning and pruning to achieve thier results, using rules similar to the latter example. But the point remains that the knowledge base consists of a large number of conditions and outcomes, which are not generally arranged in a deterministic structure by the human expert, rather they represent bits of information that are connected by the sense of relevance that the human expert gives them. It is the inference engine's role to reconstruct this relevance. Both the former and the latter style of knowledge base share the same assumption: that each outcome has some non-intersecting set of derivations with respect to other outcomes.

Most current expert systems also have a probabilistic component. The knowledge base is for the most part entered in the form of 'higher-level' rules, but objective and subjective probabilities are attached to the conditions and outcomes by the human expert. This is one way to allow the derivation of the outcome to be partial; the outcome need not be absolutely defined with respect to the knowledge base, only defined to some arbitrary degree of probability. This greatly increases the capacity of the expert system to classify, since it is not restricted to finding exact matches to what has been encountered before, but to compare as prototypes, simulating the capacity of human experts to make judgments on new cases.

There are several ways to account for the success of current expert systems. First, since the local models as presented to the expert system are only descriptive models, and the overall system is a performance model, no internal explanation need be generated; the expert system is judged only on its descriptive performance. Second, modern statistical methods are quite powerful descriptively, so one could expect them to be reliable descriptors when used. Third, the knowledge base is created, selected and pruned by humans and consists of human expert judgments. This is also true of information supplied to the expert system while it is operating. So it is assumed that the human can answer the questions asked by the expert system appropriately and correctly. So in many ways the success of contemporary expert systems is a sleight of hand; all the human interaction in the process is taken for granted. But it is fair to say that all the expert system designer is claiming to do is represent the knowledge of a human expert, not to create a human-like expert.

From an anthropologist's point of view the rule-based model is preferable to the statistical one, but it makes no difference to the goal of the expert system, which is simply to descriptively mimic an expert. No current system can do more; expert system writers might claim psychological reality (many do not), but that is a far cry from establishing psychological reality, as the debates (see Burling 1964 and Buchler and Selby 1968) over the new anthropology of the 1960s demonstrate.

Anthropologists may still find possible significance to anthropology in the general model underlying the expert system. A model of some major segment of human action need not be a single large formal model, but can be a series of weakly interacting local models. If these can be stated consistently, anthropologists can explore at least descriptively how the models interact with each other.

8.2.6 Conclusion

The goal of an expert system is to make qualitative judgments, to predict the state of a system relative to contextual data. However, it may not be clear how an expert system can help in qualitative analysis. After all, if you have to provide the model, what is the expert system doing for you? This is not a fair argument as it applies to any computer-based aid. It does nothing that you could not do given pencil and paper, in ten or twenty years. The computer in this role amplifies what can be done.

There are two more serious objections that can be raised. One is the *hidden model* objection, which rests on what happens in the black box of the inference engine to the model or data that was entered. This is a problem only if there is no control over the identification and evaluation mechanisms in the system. In general the other mechanisms are not terribly important; for example, it is not important from an analytic point of view whether the goal mechanism is a forward or backward chaining strategy. That is a description of how the information is ordered and accessed internally, rather than how it is evaluated. However, it is critical to control, or at least to understand, the internal evaluation

method, for the analyst is locked into the limited range of possible models that a given system can accommodate. This is strictly an issue of access to programming skill (Read and Behrens 1992:250).

The second objection is to the formal or theoretical basis of the general model of an expert system. As outlined above, all current expert systems work more or less upon one general macro-method; given a list of symptoms and a list of outcomes the system evaluates the most likely state(s) (outcome) for the system to take at each point of the analysis. The generalized expert system model attempts to achieve this global scope without explicitly laying down all the paths, rather piecing together a unique solution for each unique situation, using only a series of small, local models and a general inference mechanism as the basis. It does this not by incorporating a single exhaustive model relating all possible states to each other, but by using individual instances of information and relating them according to a weak interaction internal model. There is formal support for the weak interaction model in mathematics from Thom (1975), and in anthropology and simulation from Zackary (1980).

The problem with using expert systems in anthropological analysis is created by the split between the knowledge base and the inference engine; in general the non-programmer anthropologist can only control the knowledge base. Regardless of the type of models that the anthropologist sets up in the knowledge base, the inference model must be known to evaluate the interaction of the models as anticipated. This makes the system suspect for analysis unless one knows the inference model in detail, and is satisfied that it realistically represents the assumptions that must be made. This objection is not to the general approach, but to the fixation to a particular global model, the evaluation mechanism. This problem is not unique to expert systems, but arises in any use of simulation to test models: the result of a model must always be tested against another model before it can be interpreted. The properties of the evaluation model must be known and consistent with its purpose. If the problem of control can be overcome then the general expert system model has potential as a means of exploring the interactions of a large number of local models towards a set of global responses; a method of qualitative simulation.

Appendix

The rapid rate of change in computer hardware and software directed this book to be rather generic in its approach to particular programs: in earlier drafts of this book specific software was discussed, only to drop out of existence or become so transformed as to be new. This is likely to be frustrating to many readers, since it is not easy to locate software which meets your requirements. This can be resolved to some extent by reading reviews in the *Anthropology Newsletter, Quantitative Anthropology,* and *World Cultures*. To address this problem the Centre for Social Anthropology and Computing at the University of Kent is hosting an FTP archive on its server, Lucy (in honour of Professor Lucy Mair). You can 'log on' to Lucy using the following information (ask your system administrator to assist and interpret terms):

Internet address: lucy.ukc.ac.uk
login: anonymous
password: [your login name]

Reviews of software are stored in the directory 'reviews'. For latest details of the holdings read the file 'README'. You may also Email:

csa-c@ukc.ac.uk

with the subject line *Reviews* for reviews, *News* for news and *Information* for instructions and a description of holdings.

We are also archiving data sets, field notes and software written by anthropologists for access by the world anthropological community. If you have something to offer, Email mf1 @ukc.ac.uk for details.

Notes

1 Though not, of course, more valid. Simply representing models on a computer says nothing about the validity of those models. Even if descriptive adequacy can be shown for a model, external criteria are required to establish whether that model can be considered 'the' model used by, say, the Zuwaya or Zapotec.
2 It takes new hardware about 5–10 years to move from development to product, so this is not so speculative as some might imagine. I am basically predicting that what already exists will become an affordable product in the near future.
3 I tend to use 'data' and 'information' interchangeably in general contexts. A specific set of data corresponds to a model, whereas information is more general and less directed. Information is potentially many sets of data depending on the models which are applied, as data is in the general sense.
4 Matters of 'authority' aside, I have had exceedingly poor results with textual field material entered by keypunchers and clerical staff, unless, as is becoming the case rather more often, these staff hold relevant degrees. I am also referring to sensitive information about other people which, even with their effective anonymity, I am personally uncomfortable revealing to others.
5 Currently *Video for Windows* for Microsoft operating systems and *Quicktime* for Apple operating systems.
6 Active matrix and passive matrix refer to a display technology for liquid crystal display (LCD) screens. This will be denoted in the promotional literature on notebook computers, and active matrix displays are a common option. Passive matrix displays tend to 'smear' when the screen is changing rapidly, which is irritating at the best of times, and intolerable with video. An active matrix greyscale screen is preferable to a passive matrix colour screen, in my opinion.
7 Wenonah Lyon, personal communication.
8 As discussed in Section 2.2, computer operations should be conceptually distinguished from computer programs. Computer operations define the things that computers do. Computer programs are collections of implemented operations, combined with a user interface and a set of conventions. As Pfaffenberger comments, 'Specific programs come and go…but software genres remain remarkably stable…' (1988:11). The pedigree of almost every operation in this chapter extends back at least forty years.

9 Which perhaps accounts for why literary scholars have used computers for almost forty years, while social anthropologists have virtually ignored the use of computers for qualitative research.
10 While visiting the South Pacific I heard reliable accounts of a fungus which attacks the surface of floppy disks. These have yet to be substantiated, but keeping the disk in a dry box is alleged to be a cure. It is possible that such technohazards may become more prevalent as we introduce such items to different ecosystems.
11 There are disadvantages to hand-printed stylus ('pen') input, principally that most people can quickly learn to type faster than they can print, but under some circumstances this may not hold. Possible advantages of pen entry are that some forms of data entry can be speeded up and rough diagrams can be more easily included. Dual keyboard/pen systems are available in small units.
12 *The Ethnograph* (Seidel 1988). This was the first commercial program written specifically to assist in the analysis of interview material and textual ethnographic material, using a method similar to that described by Agar (1982) as 'strip analysis'.
13 Case sensitive indicates that upper case letters (e.g. 'A' and 'B') are distinguished from lower case letters (correspondingly 'a' and 'b').
14 Some programs may treat your character in some special way which you had not intended. For example, many programs with find operations use '*' in a special way, to indicate a 'wild character', i.e. a character that will match any other character. If this turns out to be the case, you can use a wordprocessor's 'Change' or 'Substitute' operation to change the character to something else.
15 Pfaffenberger (1988) discusses the analytic problems with reporting note context without ethnographer-defined boundaries; instead using conventional structural units such as lines or paragraphs.
16 Some programs may require specific markers for delimiting notes.
17 Gilbert was working with computers with very limited amounts of memory, and a large amount of his discussion relates to avoiding redundancy for the sake of computer storage requirements. At the time of writing this problem is much less urgent. Computers with 1 megabyte are 'small', and inexpensive systems with 4–32 megabytes are common. The focus on redundancy in the text is addressed more at the user's requirements rather than the computer's.
18 The simulation need not constitute an informal model. This characterization is based more on the context of use of simulation; to experiment, explore or validate the properties of a system which one does not understand to some significant extent. A useful simulation often results in a more formal model of the target system as a result of understanding gained by using the simulation.
19 The processes in the simulation model may or may not be intended to represent processes (if any) in the system under study.
20 This enthusiasm is not limited to AI researchers, but is common to all who research with a computer, including anthropologists (see Hymes 1965a). It appears that the promise of computers is easier to conceptualize than the results.
21 This is a structuralist (with a small 's') definition, and is not claimed to be the only definition.
22 Measurement theory has become a sensitive area in physics as well.
23 This is not strictly true, but is a valid statement given the usual form of quantitative analysis in the social sciences.

Bibliography

Agar, M. (1982) Towards an ethnographic language, *American Anthropologist* 84(4): 779–95.

Agar, M. (1983) Microcomputers as field tools. *Computers and the Humanities* 17:19–26.

Agar, M. (1986) *Speaking of Ethnography.* Beverly Hills: Sage.

Bagg, J. (1991) Modelling Kinship and residence from State of Souls registers - experiments with advanced database systems and presentations methods. Paper presented at the 1991 International Association for Humanities Computing Conference, proceedings to appear.

Bagg, J. (1992) Introduction to database systems for anthropologists. *Bulletin of Information on Computing and Anthropology* 1(8): 2–9.

Barnard, A. and A.Good (1984) *Research Practices in the Study of Kinship.* London: Academic Press.

Barr, A. and E.A.Feigenbaum (1982) *The Handbook of Artificial Intelligence,* vols 1, 2 and 3. Stanford: Heuristech Press.

Behrens, Clifford A. (1990) Qualitative and quantitative approaches to the analysis of anthropological data: a new synthesis. *Journal of Quantitative Anthropology* 2: 305–28.

Benfer, R.A., E.E.Brent Jr. and L.Furbee (1991) *Expert Systems.* London: Sage.

Black, S. (1978) Polynesian outliers: a study in the survival of small populations. In I. Hodder (ed.) *Simulation Studies in Archaeology,* pp. 63–76. Cambridge: Cambridge University Press.

Blacking, J. (1984) Preparation for fieldwork: audio-visual equipment in general ethnographic studies. In R.F.Ellen (ed.) *Ethnographic Research: A Guide to General Conduct,* pp. 199–206. London: Academic Press.

Bolc, L. and Z.Kulpa (1981) *Digital Image Processing Systems*. Berlin: Springer.

Boone, M.S. and J.J.Wood (ed.) (1992) *Computer Applications for Anthropologists.* Belmont: Wadsworth.

Boone, M.S., J.Dow and J.J.Wood (1992) Use and study of computer systems in the development of anthropological methodologies. In M.S.Boone, and J.J.Wood (eds) *Computer Applications for Anthropologists* pp. 4–21. Belmont: Wadsworth.

Bratko, I. (1986) *Prolog Programming for Artificial Intelligence.* New York: Addison Wesley.

Brent Jr, E.E. (1985) Relational data base structures and concept formation in the social sciences. *Computing in the Social Sciences* 1(1): 29–49.

Brent, E. (1988) Statistical expert systems: an example. *Journal of Statistical Computation and Simulation* 27:137–51.

Brown, J.A. and B.Werner (1974) An on-site data management system application in field archaeology. *Communications of the ACM* 17:644–6.

Buchler, I.R. and R.M.McKinlay (1969) Decision processes in culture: a linear programming analysis. In I.R.Buckler and H.G.Nutini (eds) *Game Theory in the Behavioral Sciences,* pp. 191–212. Pittsburgh: University of Pittsburgh.

Buchler, I.R. and H.A.Selby (1968) *Kinship and Social Organization.* New York: Macmillan.

Buchler, I., M.D.Fischer and J.R.McGoodwin (1986) Ecological structure, economics and social organisation: the Kapauku. In G.De Meur (ed.) *Advances in mathematical anthropology,* pp. 57–124. London: Routledge.

Burling, R. (1964) Cognition and componential analysis: God's truth or hocus-pocus? *American Anthropologist* 66:20–8.

Burnard, L.D. (1987) Knowledge base or database? Computer applications in ethnology. In J.Raben, S.Sugita and M.Kubo (eds) *Toward a Computer Ethnology* pp. 63–97. Osaka: National Museum of Ethnology.

Burton, M.L. (1973) Recent computer applications in cultural anthropology. *Computers and the Humanities* 7:337–41.

Case, R.P. (1984) Microcomputers in the field: practical considerations. *Byte* 9: 243–4, 246, 248, 250.

Chagnon, N.A. (1974) *Studying the Yanomamö.* New York: Holt, Rinehart and Winston.

Clammer, J. (1984) Approaches to Ethnographic Fieldwork. In R.F. Ellen (ed.) *Ethnographic Research: A Guide to General Conduct* pp. 63–85. London: Academic Press.

Clancey, W.J. and Letsinger, R. (1981) NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. *International Joint Conferences on AI* 7:829–36.

Clifford, J. (1990) Notes on (field)notes. In R.Sanjek (ed.) *Fieldnotes: the makings of anthropology.* pp. 47–70. Ithaca: Cornell University Press.

Colby, B.N. (1971) The shape of narrative concern in Japanese folktales. In Paul Kay (ed.) *Explorations in Mathematical Anthropology.* pp. 117–26. Cambridge, MA: MIT Press.

Collier Jr., J. and M.Collier (1986) *Visual Anthropology. Photography as a Research Method.* Albuquerque: University of New Mexico Press.

Coult, A.D. and R.R.Randolph (1965) Computer models for analyzing genealogical space. *American Anthropologist* 67:21–9.

Coxon, A.P.M. and A.D.Chalmers (1973) *Inquirer Project: Content Analysis as a Social Science Resource.* SSRC Report, Department of Sociology, Edinburgh University.

Crane, G. (1991) Composing culture: the authority of an electronic text. *Current Anthropology* 33(3): 293–302.

Crick, M.R. (1982) Anthropology of knowledge. In B.J.Siegel, A.R.Beals and S.A.Tyler (eds) *Annual Review of Anthropology* pp. 287–313. Palo Alto, CA: Annual Reviews Inc.

D'Andrade, R.G. (1973) Cultural constructions of reality. In L.Nader and T.Maretzki (eds) *Cultural Illness and Health* Washington, DC: American Anthropological Association, Anthropological Studies 9.

D'Andrade, R.G. (1976) A propositional analysis of U.S. American beliefs about illness. In K.H.Basso and H.A.Selby (eds) *Meaning in Anthropology* pp. 155–80. Albuquerque: University of New Mexico Press.

Davies, J.R. (1990) A methodology for the design of computerised qualitative research tools. *Interacting with Computers* 2:33–58.

Davies, J.R. (1991) Automated tools for qualitative research. In N.G. Fielding and R.M. Lee (eds) *Using Computers in Qualitative Research,* pp. 54–72. London: Sage.

Davies, R. and R.O'Keefe (1989) *Simulation Modelling with Pascal.* London: Prentice Hall.

Davis, J. (1984a) Report on the workshop on computing and social anthropology. *Bulletin of Information on Computing and Anthropology* 1:3–7.

Davis, J. (1984b) Data into text. In R.F.Ellen (ed.) *Ethnographic Research: A Guide to General Conduct* pp. 295–318. London: Academic Press.

Davis, J. (1987) *Libyan Politics: Tribe and Revolution.* London: I.B.Tauris.

De Meur, G. (ed.) (1986) *Advances in mathematical anthropology.* pp. 57–124. London: Routledge.

Dow, J. (1983) The combined use of computers and audiotape recorders in storing, managing, and using qualitative verbal ethnographic data. *Journal of Northern Luzon* 8:56–80.

Dow, J. (1987) Hunting and gathering tales. *Computer Assisted Anthropology Newsletter* 2(4): 42–3.

Dow, J. (1992) New directions for computer applications for anthropologists. In M.S. Boone and J.J.Wood (eds) *Computer Applications for Anthropologists* pp. 267–82. Belmont: Wadsworth.

Duda, R.O., P.E.Hart, N.J.Nilsson and G.L.Sutherland (1978) Semantic network representations in rule-based inference systems. In D.A.Waterman and F. Hayes-Roth (eds) *Pattern-directed Inference Systems,* pp. 203–21. New York: Academic Press.

Dueker, K.J. (1987) Geographic information systems and computer-aided mapping. *Journal of the American Planning Association* 53(2): 383–90.

Dumont, L. (1983) *Affinity as a Value: Marriage Alliance in South India with Comparative Essays on Australia.* Chicago, IL: University of Chicago Press.

Dyke, B. (1981) Computer simulation in anthropology. In B.J.Siegel, A.R.Beals and S.A.Tyler (eds) *Annual Review of Anthropology,* pp. 193–207. Palo Alto, CA: Annual Reviews Inc.

Dyke, B. (1986) The effect of marriage rules on mating practices. *Annals of Human Biology* 13: 575.

Dyke, B. and J.W.MacCluer, (eds) (1973) *Computer simulation in human population studies.* New York: Academic Press.

Dyson-Hudson, R. and N.Dyson-Hudson (1986) Computers for anthropological fieldwork. *Current Anthropology* 27:530–2.

Eades, J. (1988) *Simulating Yoruba traders.* Paper given to the American Anthropological Association Annual Meeting, 1988.

Eguchi, P.K. (1987) Fieldworker and computer: an end user's view of computer ethnology. In J.Raben, S.Sugita and M.Kubo (eds) *Toward a Computer Ethnology* pp. 165–75. Osaka: National Museum of Ethnology.

Ellen, R.F. (ed.) (1984a) *Ethnographic Research: A Guide to General Conduct.* London: Academic Press.

Ellen, R.F. (1984b) Producing data. In R.F.Ellen (ed.) *Ethnographic Research: A Guide to General Conduct,* pp. 273–93. London: Academic Press.

Ellen, R.F. (1986) Ethnobiology, cognition and the structure of prehension: some general theoretical notes. *Journal of Ethnobiology* 6(1): 83–98.

Ellen, R.F. and M.Fischer (1987) Computers for anthropological fieldwork. *Current Anthropology* 28:677–9.

Fielding, N.G. and R.M.Lee (ed.) (1991) *Using Computers in Qualitative Research.* London: Sage.

Findler, N.V. and W.R.McKinzie (1969) On a computer program that generates and queries kinship structures. *Behavioral Science* 14:334–43.

Findler, N.V. and W.R.McKinzie (1970) Story of the doctor, the missionary and the computer scientist: computer simulation of a demographical and kinship model. *International* Journal of Comparative Sociology 11:166–70.

Fischer, M.D. (1980) *A general simulation of production and consumption in the valley of Oaxaca, Mexico.* Masters Thesis, University of Texas, Austin.

Fischer, M.D. (1986) Expert systems and anthropological analysis. *Bulletin of Information on Computing and Anthropology* 4:6–14.

Fischer, M.D. (1987) Computer representations of anthropological data: a perspective. *Bulletin of Information on Computing and Anthropology* 6:1–9.

Fischer, M.D. (1991) Marriage and power in an urban Panjabi community in Pakistan: tradition and transition. In H.Donnan and P.Werbner (eds) *Economy and Culture in Pakistan: Migrants and Cities in a Muslim Society.* London: Macmillan.

Fischer, M.D. (1993) Computer-based visual representations in ethnographic fieldwork. *Bulletin of Information on Computing and Anthropology* 9:2–12.

Fischer, M.D. and A.Finkelstein (1991) A case study in social knowledge representation: arranging a marriage in urban Pakistan. In N.Fielding and R.Lee (eds) *Qualitative Knowledge and Computing.* London: Sage.

Fredlund, E.V. and B.Dyke (1976) Measuring marriage preference. *Ethnology* 15:35–45.

Freedman, M. (1978) Social and cultural anthropology. In J.Havet (ed.) *Main Trends of Research in the Social and Human Sciences,* The Hague: Mouton/Unesco.

Furbee, L. (1989) A folk-expert system: soils classification in the Colca Valley, Peru. *Anthropological Quarterly* 62:83–102.

Garbett, G.K. (1980) Graph theory and the analysis of multiplex and manifold relationships. In J.Clyde Mitchell (ed.) *Numerical Techniques in Social Anthropology,* vol. 3, ASA *Essays in Social Anthropology* (ed. Edwin Ardner), pp. 191–234. Philadelphia: Institute for the Study of Human Relationships.

Geoghegan, W. (1971) Information processing systems in culture. In P.Kay (ed.) *Explorations in Mathematical Anthropology.* Cambridge, MA: MIT Press.

Gilbert, J.P. (1971) Computer methods in kinship studies. In P.Kay (ed.) *Explorations in Mathematical Anthropology,* pp. 127–38. Cambridge, MA: MIT Press.

Gilbert, J.P. and E.A.Hammel (1966) Computer simulation and the analysis of problems in kinship and social structure. *American Anthropologist* 68:71–93.

Glaser, B. and Strauss, A. (1967) *The Discovery of Grounded Theory.* Chicago: Aldine.

Goody, J. and J.Buckley (1980) Implications of the sexual division of labor in agriculture. In J.C.Mitchell (ed.) *Numerical Techniques in Social Anthropology,* pp. 33–47. Philadelphia: Institute for the Study of Human Issues.

Green, A. (1988) *The Forms Filling Program.* Unpublished MA dissertation, University of Kent, Canterbury.

Guillet, D. (1985) Microcomputers in fieldwork and the role of the anthropologist. *Human Organization* 44:369–71.

Guillet, D. (1989a) Expert systems applications in anthropology, part one. *Anthropological Quarterly* 62(2): 57–105.

Guillet, D. (1989b) Expert systems applications in anthropology, part two. *Anthropological Quarterly* 62(3): 107–47.

Hachimura, K. (1987) Image processing and computer graphics in the human sciences. In J.Raben, S.Sugita and M.Kubo (eds) *Toward a Computer Ethnology,* pp. 121–49. Osaka: National Museum of Ethnology.

Hackenberg, R.A. (1967) The parameters of an ethnic group: a method for studying the total tribe. *American Anthropologist* 69:478–92.

Hage, P. and F.Harary (1991) *Exchange in Oceania: A Graph Theoretic Analysis.* Oxford: Clarion Press.

Hammel, E.A. (1976) The matrilateral implications of structural cross-cousin marriage. In E.B.Zubrow (ed.) *Demographic Anthropology: Quantitative Approaches* pp. 145–60. Albuquerque: University of New Mexico Press.

Hammel, E.A. and J.Gilbert (1965) Computer simulation of problems in kinship and social structure. In D.Hymes (ed.) *The Use of Computers in Anthropology,* pp. 513–14. London: Mouton.

Hammel, E.A. and D.Hutchinson (1973) Two tests of microsimulation: the effect of an incest tabu on population viability and the effect of age differences between spouses on the skewing of consanguineal relationships between them. In B.Dyke and J.W. MacCluer (eds) *Computer Simulation in Human Population Studies,* pp. 1–13. New York: Academic Press.

Hammel, E.A. and P.Laslett (1974) Comparing household structure over time and between cultures. *Computer Studies in Social History.* 16:73–103.

Hammel, E.A. and K.W.Wachter (1977) Primonuptiality and ultimonuptiality: their effects on stem-family household frequencies. In R.D.Lee (ed.) *Population Patterns in the Past,* pp. 113–34. New York: Academic.

Hammel, E.A., D.W.Hutchinson, K.W.Wachter and R.Z.Deuel (1976) *The SOCSIM Demographic-Sociological Microsimulation Program Operating Manual.* Institute for International Studies.

Hammel, E.A., C.K.McDaniel and K.W.Wachter (1979) Demographic consequences of incest tabus: a microsimulation analysis. *Science* 205:972–7.

Hammel, E.A., C.K.McDaniel and K.W.Wachter (1980) Vice in the Villefranchian: a microsimulation analysis of the demographic effects of incest prohibitions. In B.Dyke and W.Morrill (eds) *Genealogical Demography,* pp. 209–34. New York: Academic Press.

Harrison, S., C.Jardine, J.King, T.King and A.Macfarlane (1979) Reconstructing historical communities by computer. *Current Anthropology* 20(4): 808–9.

Hicks, D. (1984) Preparing for fieldwork: accounting. In R.F.Ellen (ed.) *Ethnographic Research: A Guide to General Conduct,* pp. 190–2. London: Academic Press.

Holy, L. and M.Stuchlik (1985) *Norms and Representations.* Cambridge: Cambridge University Press.

Howell, N. (1979) *Demography of the Dobe !Kung.* New York: Academic Press.

Howell, N. and V.A.Lehotay (1978) AMBUSH: a computer program for stochastic microsimulation of small human populations. *American Anthropologist* 80:905–22.

Hursch, C.J. and J.Hursch (1988) *SQL. The Structured Query Language.* Blue Ridge Summit: Tab Books.

Hymes, D.H. (1965a) Introduction. In D.H.Hymes (ed.) *The Use of Computers in Anthropology: Papers Presented for a Symposium sponsored by the Wenner-Gren Foundation for Anthropological Research, 1962,* The Hague: Mouton.

Hymes, D.H. (ed.) (1965b) *The Use of Computers in Anthropology: Papers Presented for a Symposium sponsored by the Wenner-Gren Foundation for Anthropological Research, 1962.* The Hague: Mouton.

Jackson, B. (1987) *Fieldwork.* Urbana: University of Illinois Press.

Jackson, J.E. (1990) 'I am a fieldnote': fieldnotes as a symbol of professional identity. In R.Sanjek (ed.) *Fieldnotes: the makings of anthropology,* pp. 3–33. Ithaca: Cornell University Press.

Johnson, A. (1978) *Quantification in Cultural Anthropology.* Stanford: Stanford University Press.

Johnson, A. and C.Behrens (1989) Time allocation research and aspects of method in cross-cultural comparison. *Journal of Quantitative Anthropology* 1(2): 234–45.

Johnson, A. and O.R.Johnson (1990) Quality into quantity. In R.Sanjek (ed.) *Fieldnotes: the makings of anthropology,* pp. 161–86. Ithaca: Cornell University Press.

Kay, P. (ed.) (1971) *Explorations in Mathematical Anthropology.* Cambridge, MA: MIT Press.

Khanna, G.S. (1988) On electronic communication for anthropologists. *Current Anthropology* 29(2): 312.

Kippen, J. (1986) Computational techniques in musical analysis. *Bulletin of Information on Computing and Anthropology* 4:1–4.

Kippen, J. (1988a) *The Tabla of Lucknow: A Cultural Analysis of a Musical Tradition,* Cambridge Studies in Ethnomusicology. Cambridge: Cambridge University Press.

Kippen, J. (1988b) On the uses of computers in Anthropological research. *Current Anthropology* 29:317–20.

Kippen, J. (1989) Computers, fieldwork, and the analysis of cultural systems. *Bulletin of Information on Computing and Anthropology* 7:1–7.

Kononenko, I. and N.Lavrac (1989) *Prolog through Examples: A practical programming guide.* Wilmslow: Sigma Press.

Kowalski, R. (1984) Logic as a database language. In J.Longstaff (ed.) *Proceedings of the Third British National Conference on Databases,* Cambridge: Cambridge Unviersity Press.

Kubo, M. (1987) Visual information retrieval system for ethnological research. In J. Raben, S.Sugita and M.Kubo (eds) *Towards a Computer Ethnology,* Senri Ethnological Studies 20. Osaka: National Museum of Ethnology.

Kunstadter, P., R.Buhler, F.Stephan and C.Westoff (1963) Demographic variability and preferential marriage patterns. *American Journal of Physical Anthropology* 21: 511–19.

Lansing, J.S. (1991) *Priests and Programmers. Technologies of power in the engineered landscape of Bali.* Princeton: Princeton University Press.

Lestel, D. (1989) FR: laboratory anthropology and use of artificial-intelligence. *Social Science Information* 28:685–705.

Lévi-Strauss, C. (1963) *Structural Anthropology.* Garden City: Doubleday and Co.

Lyon, W. and M.Fischer (forthcoming) Economic strategies of households in a lower-income community in Lahore, Pakistan. In H.Donnan (ed.) *The Household in Pakistan.*

MacCluer, J.W. (1973) Computer simulation in anthropology and human genetics. In M.H. Crawford and P.L.Workman (eds). *Methods and Theories of Anthropological Genetics,* pp. 220–34. Albuquerque: Universiy of New Mexico Press.

MacCluer, J.W. and B.Dyke (1976) Minimum size of endogamous populations. *Social Biology* 23(1) 1–12.

Macfarlane, A. (1990) The Cambridge experimental videodisc project. *Anthropology Today* 6(1): 9–12.

Macfarlane, A., J.Jacobs, S.Harrison, M.Porter (1987) The Cambridge experimental videodisc project. *Bulletin of Information on Computing and Anthropology* 5:8–10.

Marcus, G.E. and D.Cushman (1982) Ethnographies as texts. *Annual Review of Anthropology* 11:25–69.

Meadows, D.H., D.L.Meadows, J.Randers and W.W.Behrens III (1972) *The Limits of Growth.* New York: Universe Press.

Meeker, R.J. and G.H.Shure, (1968) *Updating some Ground Rules for Man-Machine Simulation.* US Government Research and Development Report AD-672783.

Michie, D. (1982) *Machine Intelligence and Related Topics: An Information Scientist's Weekend Book,* London: Gordon and Breach Science Publishers.

Miranda, P. (1967) Computers in the bush: tools for the automatic analysis of myths. In June Helm (ed.) *Essays on the Verbal and Visual Arts.* pp. 77–83. American Ethnological Society, Proceedings of the 1966 Annual meeting. Seattle: University of Washington Press.

Mitchell, J.C. (1972) *COMPUTER GUIDE 3: Programs for Social Scientists: Social Data Analysis Programs in the UK.* Manchester: National Computing Centre.

Mitchell, J.C. (ed.) (1980) *Numerical Techniques in Social Anthropology.* Philadelphia: Institute for the Study of Human Issues.

Mitchell, J.L. (1974) *Computers in the Humanities.* Edinburgh: Edinburgh University Press.

Mulvaney, D.J. (1970) The anthropologist as tribal elder. *Mankind* 7:207–17.

Murdock, G.P. (1967) *Ethnographic Atlas.* Pittsburg: University of Pittsburg Press.

Nardi, B.A. (1980) The use of computer simulation models for predicting sociocultural change. In Susan Abbott and John van Willigen (eds) *Predicting Sociocultural Change,* pp. 38–56. Athens: University of Georgia Press.

Nielsen, Jakob (1990) *Hypertext and Hypermedia.* London: Academic Press Inc.

Nilsson, N.J. (1982) *Principles of Artificial Intelligence.* New York: Springer-Verlag.

Notes and Queries on Anthropology (4th edn, 1912) London: Royal Anthropological Institute.

Ottenberg, S. (1990) Thirty years of fieldnotes: changing relationships to the text. In R. Sanjek (ed.) *Fieldnotes: the makings of anthropology,* 139–60. Ithaca: Cornell University Press.

Ottenheimer, M. (1985) The world of kinship. *Computer Assisted Anthropology Newsletter* 1(3): 17.

Ottenheimer, M. (1988) Modeling systems of kinship. *World Cultures.* 4:1 (computer program and manual).

Pavlidis, T. (1982) *Algorithms for Graphics and Image Processing.* Rockville, MD: Computer Science Press.

Pfaffenberger, B. (1988) *Microcomputer Applications in Qualitative Research.* Qualitative Methods Series 14. Newbury Park: Sage Publications.

Powlesland, D.J. (1986) On site computing: in the field with the silicon chip. In J.D. Richards (ed.) *Computer Usage in British Archaeology: Report of the Joint IFA/ RCHM Working Party on Computer Usage,* Birmingham: University of Birmingham.

Ragin, C.C. (1987) *The Comparative Method: Moving beyond Qualitative and Quantitative Strategies,* Berkeley, CA: University of California Press.

Randolph, R.R. and A.D.Coult (1968) A computer analysis of Bedouin marriage. *South-western Journal of Anthropology* 24:83–99.

Read, D.W. and C.A.Behrens (1989) Modeling folk knowledge as expert systems. *Anthropological Quarterly* 62(3): 107.

Read, D.W. and C.A.Behrens (1992) Computer representation of cultural constructs: new research tools for the study of kinship terminologies. In M.S. Boone and J.J.Wood (eds) *Computer Applications for Anthropologists,* pp. 228–50. Belmont: Wadsworth.

Read, M.W. (1980) Ethnographic fieldnotes and interview transcripts: *Some Preliminary Observations on the Computer Management of Text.* Cardiff Sociology Research Unit.

Richards, L. and T.Richards (1991) The transformation of qualitative method: computational paradigms and research processes. In N.G. Fielding and R.M.Lee (eds) *Using Computers in Qualitative Research,* pp. 38–53. London: Sage.

Rossi, I. (1982) On the 'Scientific' evidence for the existence of deep structures and their objective and mathematical nature (a training session for Rodney Needham, Ronald Cohen, Peter Caws, and Paul Chaney). In I.Rossi (ed.) *The Logic of Culture: Advances in Structural Theory and Methods,* pp. 235–93. South Hadley: J.F.Bergin.

Ryan, N.S. (1985) Gtree: a system for interactive display and manipulation of genealogical data. *Bulletin of Information on Computing and Anthropology* 3:6–16, University of Kent.

Ryan, N.S. (1988) Browsing through the stratigraphic record. *Computer and Quantitative Methods, Proceedings Computing Applications in Archaeology 1988,* BAR International Series 446(ii): 327–34.

Sailer, L. (1984) Computer-assisted anthropology. *Practicing Anthropology* 6(2): 4–17.

Sanjek, R. (1990a) The secret life of fieldnotes. In R.Sanjek (ed.) *Fieldnotes: the makings of anthropology,* pp. 187–272. Ithaca: Cornell University Press.

Sanjek, R. (1990b) A vocabulary for fieldnotes. In R.Sanjek (ed.) *Fieldnotes: the makings of anthropology,* pp. 92–138. Ithaca: Cornell University Press.

Sanjek, R. (1990c) Fire, loss, and the sorcerer's apprentice. In R. Sanjek (ed.) *Fieldnotes: the makings of anthropology,* pp. 34–46. Ithaca: Cornell University Press.

Sanjek, R. (1990d) (ed.) *Fieldnotes. The Making of Anthropology,* Ithaca: Cornell University Press.

Scollar, I. (1975) Transformation of extreme oblique aerial photographs to maps or plans by conventional means or by computer. In D.R. Wilson (ed.) *Aerial Reconnaissance for Archaeology,* pp. 52–9. London: Council of British Archaeology.

Scollar, I. (1978) Computer image-processing for archaeological air photographs. *World Archaeology* 10(1): 71–87.

Seidel, J.V. (1988) *The Ethnograph Version 3.0* (computer program). Littleton, CO: Qualis Research Associates.

Seidel, J.V. (1991) Method and madness in the application of computer technology to qualitative data analysis. In N.G.Fielding and R.M.Lee (eds) *Using Computers in Qualitative Research,* pp. 107–16. London: Sage.

Shortliff, E.H. (1976) *Computer-Based Medical Consultations: MYCIN.* Amsterdam: North-Holland.

Sperber, D. (1985) *On Anthropological Knowledge.* Cambridge: Cambridge University Press.

Sproull, L.S. and R.F.Sproul (1982) Managing and analyzing behavioral records: explorations in nonnumeric data analysis. *Human Organization* 41:283–90.

Suchman, L.A. (1987) *Plans and situated actions. The problem of human-machine communication.* Cambridge: Cambridge University Press.

Sugita, S. (1987) Computers in ethnological studies: as a tool and an object. In J.Raben, S.Sugita and M.Kubo (eds) *Towards a Computer Ethnology,* Senri Ethnological Studies 20. Osaka: National Museum of Ethnology.

Sutton, D. (1984) Field recording and computers at Pouerua, Inland Bay of Islands. *Newsletter of the New Zealand Archaeological Association* 27:156–65.

Thom, R. (1975) *Structural Stability and Morphogenesis.* Reading: W.A.Benjamin Inc.

Tomajczyk, S.F. (1985) Notes from the field: anthropologist uses picocomputer. *PICO—The Briefcase Computer Report* 1:17–20.

Trotter II, R.T. (1992) Ethnographic data management: a model from a dispersed multi-ethnographer project. In M.S. Boone and J.J.Wood (eds) *Computer Applications for Anthropologists,* pp. 51–62. Belmont: Wadsworth.

Wachter, K.W., E.A.Hammel and T.P. R.Laslett (1978) *Statistical Studies of Historical Social Structure.* New York: Academic Press.

Weinberg, D. and G.H.Weinberg (1972) Using a computer in the field: kinship information. *Social Science Information* 11:37–59.

Werner, O. (1982) Microcomputers in cultural anthropology: APL programs for qualitative analysis. Microcomputers aid in the study of Navajo and other cultures. *Byte* 7:250–80.

White, D.R. (1973) Mathematical anthropology. In J.J.Honigmann (ed.) *Handbook of Social and Cultural Anthropology,* pp. 369–445. Chicago, IL: Rand McNally and Company.

White, D.R. and P.Jorion (1992) Representing and computing kinship: a new approach. *Current Anthropology* 33(4): 454–63.

White, D.R. and G.F.Truex (1988) Anthropology and computing: the challenges of the 1990s. *Social Science Computer Review* 6(4): 481–97.

Wilson, W. (1992) Data entry and management techniques for anthropologists: an example using dBase IV in a historic preservation project. In M.S. Boone and J.J.Wood (eds) *Computer Applications for Anthropologists,* pp. 77–89. Belmont: Wadsworth.

Wobst, H.M. (1974) Boundary conditions for paleolithic social systems: a simulation approach. American Antiquity 39:147–78.

Zackary, W. (1980) *CASP: a cyclical simulation language for anthropology.* Ph.D. Thesis, Temple University, Philadelphia.

Zadeh, L.A. (1965) Fuzzy sets. *Information and Control 8:338–53.*

Zeigler, B.P. (1979) Structuring principles for multifaceted system modelling. In B.P. Zeigler and M.S.Elzas (eds) *Methodology in Systems Modelling and Simulation,* pp. 93–135, Oxford: North-Holland.

Zeman, J. (1973) *Modal Logic: the Lewis-model system.* Oxford: Clarion Press.

Name index

Subject index