Analyzing Qualitative Data with Computer Software

Eben A. Weitzman

Objective. To provide health services researchers with an overview of the qualitative data analysis process and the role of software within it; to provide a principled approach to choosing among software packages to support qualitative data analysis; to alert researchers to the potential benefits and limitations of such software; and to provide an overview of the developments to be expected in the field in the near future.
Data Sources, Study Design, Methods. This article does not include reports of empirical research.
Conclusions. Software for qualitative data analysis can benefit the researcher in terms of speed, consistency, rigor, and access to analytic methods not available by hand. Software, however, is not a replacement for methodological training.

Key Words. Qualitative research, qualitative data analysis, software, CAQDAS, mixed-methods research

As health services researchers increasingly turn to qualitative and mixed (qualitative and quantitative) research methods, there is increasing interest in finding and using new tools and methods for the analysis of qualitative data. As recently as the mid-1980s, most qualitative researchers were carrying out the mechanics of their analyses by hand: typing up field notes and interviews, photocopying them, marking them up with markers or pencils, cutting and pasting the marked segments onto file cards, sorting and shuffling the cards, and typing up their analyses. Some were beginning to use word processors for the typing work, and just a few were beginning to experiment with database programs for storing and accessing their text. Most qualitative methods textbooks at the time (e.g., Bogdan and Biklen 1982; Goetz and LeCompte 1984; Lofland and Lofland 1984; Miles and Huberman 1984) made little, if any, reference to the use of computers. A couple of programs designed specifically for the analysis of qualitative data were just beginning to appear (Drass 1980; Seidel and Clark 1984; Shelly and Sibert 1985).

But the landscape has changed dramatically. There has been an outpouring of journal articles, a series of international conferences on computers and qualitative methodology, thoughtful books on the topic (Fielding and Lee 1991, 1998; Kelle 1995; Tesch 1990; Weitzman and Miles 1995b), and special journal issues (Mangabeira 1996; Tesch 1991). At the time the late Matt Miles and I wrote *Computer Programs for Qualitative Data Analysis* (Weitzman and Miles 1995b), we reviewed no fewer than 24 different programs that were useful for analyzing qualitative data. Half of those programs had been developed specifically for the analysis of qualitative data, while the other half had been developed for more general-purpose applications such as text search and storage. Since then, the field has continued to grow rapidly. Programs are being revised at a regular rate, and new programs appear on the scene at the rate of one or two a year. Yet the software varies widely, and it remains very much the case that there is no one best program.

In this context, a researcher needs to be able to make a *principled choice* of software: one that matches the capabilities of the software to the specific needs of the researcher and the project. To that end, this article provides an overview of the range of uses of computer software in qualitative research; describes the major types of software available; provides an approach to needs assessment that can match software choice to specific researchers and projects; addresses some common questions about whether and why researchers should use software; and lays out some of the future developments that are either hoped for or clearly on the horizon.

THE ROLE OF COMPUTERS IN QUALITATIVE RESEARCH

In order to be able to discuss the role of computers in qualitative research, it will be helpful to identify the major steps in the analysis process. The model in Figure 1, and the description that follows, are intended as a general

---

Portions of this article are adapted from Miles and Weitzman (1996) and Weitzman and Miles (1995b). Most of the specific programs alluded to here are reviewed in detail (sometimes in slightly earlier versions) in Weitzman and Miles (1995b).

Address correspondence to Eben A. Weitzman, Ph.D., Assistant Professor, Graduate Programs in Dispute Resolution, University of Massachusetts Boston, 100 Morrissey Blvd., Boston, MA 02125. This article, submitted to *Health Services Research* on February 8, 1999, was revised and accepted for publication on June 24, 1999.

representation of those steps, one that sees many and wide variations in practice, and *not* as any sort of prescriptive model.

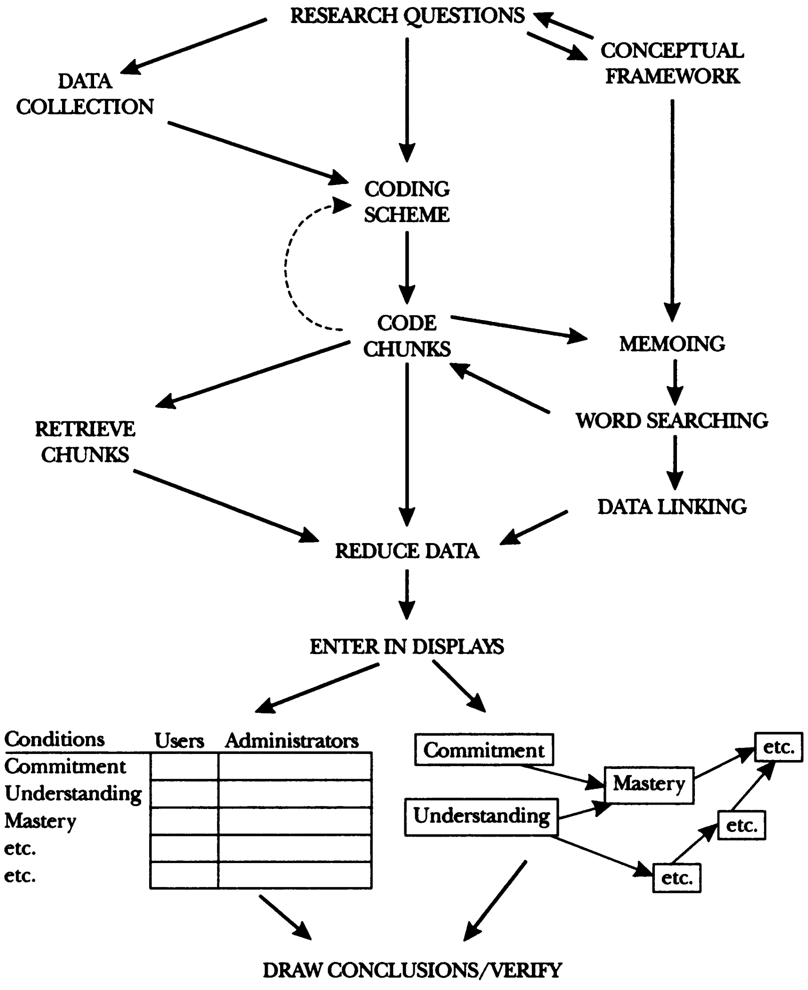
THE ANALYSIS PROCESS: FROM RESEARCH QUESTIONS TO CONCLUSIONS

In general, researchers begin with a set of research questions and move toward reaching conclusions. Data are collected in order to answer the research questions, and in qualitative studies the data are often voluminous. The researcher then faces the task of somehow reducing the data into a form in which it can be examined for patterns and relationships.

In most approaches, the researcher will go about this through developing some sort of *coding scheme*: a set of tags or labels representing the conceptual categories into which to sort the data. These may be developed either *a priori* from the conceptual framework driving the study, inductively as the analysis proceeds and the analyst begins to identify issues in the data, or by some combination of the two. Segments of the data are marked with relevant codes (coded). In many cases, researchers write memos as they code, recording emerging ideas and early conclusions about both theory and methods. As insights accrue, it often becomes useful to search back through the data for places where specific words or phrases are used, and to locate related phenomena in the text, in order to both code these new chunks and check the validity of emerging conclusions. It may sometimes be useful to create pointers (or *links*) between different places in the text where the same issues arise, as, for example, when in one interview a patient describes an episode that is elsewhere also described by his or her caregiver. This whole process will in many cases give rise to modification of any *a priori* coding scheme, and many researchers follow an iterative process, making repeated coding passes through the data.

The next step in reducing the data is often to retrieve the chunks of text associated with particular codes, reading these passages to refine the understanding of that conceptual category. It may also be desirable to retrieve text according to *combinations* of codes—for example, to see where a concept having to do with a particular caregiver attitude, say, "empathizes with patient," coincides with a particular context, say, "extended one-on-one caregiver-patient contact." The researcher would then be able to read all of the chunks (text passages) where both of these codes had been applied, to

Figure 1: From Research Questions to Conclusions



see what, if any, relationship might exist between the two. Depending on the nature of the study, it might also be desirable to identify all of the *cases* where both of these codes apply, even if the two concepts do not happen to emerge from the same text chunk.

In this way, the researcher begins to be able to write summaries of the main conceptual issues that appear in the data. These preliminary write-ups have the virtue of being much smaller in physical length than the original transcripts, making them that much more accessible to the researcher's scrutiny.

They also have the disadvantage of being one step removed from the original data. It is therefore important for the researcher to have the ability to examine and re-examine the underlying data as analysis and write-up proceeds, in order to continually check interpretations against the data. Finally, many researchers enter their summaries into displays, for example, text matrices or network diagrams, that aid in identifying patterns and relationships in the data (Miles and Huberman 1994). In Figure 1, the displays are offered as an example, and the categories are adapted from the work of Huberman and Miles (Huberman and Miles 1983, 1984; Miles and Huberman 1994). The example matrix is composed of rows representing several conditions for success of a school innovation (commitment, understanding, mastery) and columns representing some key stakeholder groups: users of the innovation (teachers) and the school administrators. The cells of the matrix would be filled with summaries of each stakeholder group's views of each condition, allowing the researcher to look for patterns across the data. The network diagram, similarly, shows the researcher's representation of the connections among the different conditions.

From these summaries of the data, which may now exist in memos, code "definitions," mini write-ups, and/or displays, the researcher draws conclusions. Depending on the methodology being employed, the researcher may then embark on a new round of searching through the data to verify whether the conclusions reached are in fact supported by the data. And, finally, a report is produced.

In closing this discussion, it is important to emphasize that it is not intended as a substitute for methodological training, but as a generalized and quite simplified description. The researcher embarking on his or her first qualitative study will need to seek out training and readings on the methodology involved in both data collection and analysis in order to fully understand the content and process of the steps alluded to.

USES OF COMPUTER SOFTWARE IN QUALITATIVE ANALYSIS

Perhaps obviously, many of the tasks just described can be greatly assisted by the judicious use of computers. Computers can help in virtually all phases of this process, from taking notes to writing reports. For example, computers can be used for:

1. *Making notes* in the field;
2. *Writing* up or transcribing field notes;

3. *Editing*: correcting, extending, or revising field notes;
4. *Coding*: attaching key words or tags to segments of text to permit later retrieval;
5. *Storage*: keeping text in an organized database;
6. *Search and retrieval*: locating relevant segments of text and making them available for inspection;
7. *Data "linking"*: connecting relevant data segments to each other to form categories, clusters, or networks of information;
8. *Memoing*: writing reflective commentaries on some aspect of the data as a basis for deeper analysis;
9. *Content analysis*: counting frequencies, sequences, or locations of words and phrases;
10. *Data display*: placing selected or reduced data in a condensed, organized format, such as a matrix or network, for inspection;
11. *Conclusion-drawing and verification*: aiding the analyst to interpret displayed data and to test or confirm findings;
12. *Theory-building*: developing systematic, conceptually coherent explanations of findings and testing hypotheses;
13. *Graphic mapping*: creating diagrams that depict findings or theories; and
14. *Report-writing*: interim and final (adapted from Miles and Huberman 1994: 44).

But there is no one program that will do all of these things best. Therefore, the best choice will depend on your project, your methodology, your own style of analysis and personal preferences, and your team if you have one. Further on you will find a set of questions to guide you through an assessment of your needs. First, however, it will be helpful to offer a typology of software types.

SOFTWARE TYPE FAMILIES

This is a rough sorting of available software into types. There is naturally quite a bit of overlap among categories, with individual programs having functions that would seem to belong to more than one type. However, it is possible to focus on the "heart and soul" of a program: what it mainly is intended for. This categorization scheme was first presented in Weitzman and Miles (1995b).

Text Retrievers

Text retrievers specialize in finding all instances of words and phrases in text, in one or several files. They typically allow you to search for places where two or more words or phrases coincide within a specified distance (a number of words, sentences, pages, etc.) and allow you to sort the resulting passages into different output files and reports. They may do other things as well, such as content analysis functions like counting, displaying keywords in context, or creating concordances (organized lists of all words and phrases in their contexts), or they may allow you to attach annotations or even variable values (for things like demographics or source information) to points in the text. Examples of text retrievers are Sonar Professional, The Text Collector, and ZyINDEX.

Textbase Managers

Textbase managers are database programs specialized for storing text in more or less organized fashion. They are good at holding text together with information about it, and at allowing you to quickly organize and sort your data in a variety of ways, and retrieve it according to different criteria. Some are better suited to highly structured data that can be organized into "records" (i.e., specific cases) and "fields" (variables: information that appears for each case), while others easily manage "free-form" text. They may allow you to define fields in the fixed manner of a traditional database such as Microsoft Access®or FileMaker Pro®, or they may allow significantly more flexibility, for example, allowing different records to have different field structures. Their search operations may be as good as, or sometimes even better than those of some text retrievers. Examples of textbase managers are askSam, Folio Views, Idealist, InfoTree32 XT, and TEXTBASE ALPHA.

Code-and-Retrieve Programs

Code-and-retrieve programs are often developed by qualitative researchers specifically for the purpose of qualitative data analysis. The programs in this category specialize in allowing the researcher to apply category tags (codes) to passages of text, and later retrieve and display the text according to the researcher's coding. These programs have at least some search capacity, allowing you to search either for codes or for words and phrases in the text. They may have a capacity to store memos. Even the weakest of these programs represent a quantum leap forward from the old scissors-and-paper approach: they're more systematic, more thorough, less likely to miss things, more flexible, and much, much faster. Examples of code-and-retrieve programs are HyperQual2, Kwalitan, QUALPRO, Martin, and The Data Collector.

Code-based Theory Builders

Most of these programs are also based on a code-and-retrieve model, but they go beyond the functions of code-and-retrieve programs. They do not, nor would you want them to, build theory for you. Rather, they have special features or routines that go beyond those of code-and-retrieve programs in supporting *your* theory-building efforts. For example, they may allow you to represent relations among codes, build higher-order classifications and categories, or formulate and test theoretical propositions about the data. They may have more powerful memoing features (allowing you, for example, to categorize or code your memos) or more sophisticated search-and-retrieval functions than do code-and-retrieve programs. They may also offer capabilities for "system closure," allowing you to feed results of your analyses (such as search results or memos) back into the system as data. Examples of code-based theory builders are AFTER, AQUAD, ATLAS/ti, Code-A-Text, HyperRESEARCH, NUD•IST, QCA, The Ethnograph, and winMAX. Two of these programs, AQUAD and QCA, support cross-case configural analysis (Ragin 1987), with QCA being dedicated wholly to this method and not having any text-coding capabilities.

Conceptual Network Builders

These programs emphasize the creation and analysis of network displays. Some of them are focused on allowing you to create network drawings, that is, graphic representations of the relationships among concepts. Examples of these are Inspiration, MetaDesign, and Visio. Others are focused on the analysis of cognitive or semantic networks, for example, the program MECA. Still others offer some combination of the two approaches, for example, SemNet and Decision Explorer. Finally, ATLAS/ti, a program also listed under code-based theory builders, has, in addition, a fine graphical network builder connected to the analytic work you do with your text and codes.

Summary

In concluding our discussion of the five main software family types, it is important to emphasize that functions often cross type boundaries. For example, Folio Views can code and retrieve, and has an excellent text search facility. ATLAS/ti, NUD•IST, The Ethnograph, and winMax graphically represent the relationships among codes, although among these, only ATLAS/ti allows you to work with and manipulate the drawing. The Ethnograph and winMAX each have a system for attaching variable values (text, date, numeric, etc.) to text files and/or cases. The implication: do not decide too early which family you want to choose from. Instead, stay focused on the functions you need.

CONCEPTUAL ASSUMPTIONS

One concern many researchers have is that a methodological or conceptual approach will be imposed by the software. In fact, developers do bring assumptions, conceptual frameworks, and sometimes even methodological and theoretical ideologies to the development of their products. These have important implications for the impact that using a program will have on your analyses. However, you need not, and in fact should not, be trapped by these assumptions and frameworks: there are often ways of bending a program to your own purposes. For example, a program may allow you to define only hierarchical relations among codes. You might work around this by creating redundant codes in different parts of the hierarchy, or by keeping track of the extra relationships *you* want to define with memos and network diagrams.

The fact that developers bring conceptual assumptions to their work is in fact one of the strengths of the field. Many of the developers, particularly of code-and-retrieve and code-based theory-builder programs, are researchers themselves. They have invested enormous intellectual energy in finding the right tools for analyses of different types, and the user can benefit greatly.

It is also true that the design of the software can have an impact on analysis. For example, different programs work with different "metaphors," that is, different ways of presenting the relationships among codes and between codes and text. Kwalitan, NUD•IST, The Ethnograph, and winMAX all allow hierarchical relations among codes. For studies in which you are organizing your conceptual categories hierarchically, these programs offer significant strengths. If you want to represent non-hierarchical relationships, even if you choose to try to work around this metaphor, it may be less comfortable then using a program like ATLAS/ti that explicitly supports more flexible networks. HyperRESEARCH emphasizes the relationship between codes and cases, rather than codes and chunks of text. When you code a chunk of text, you create an entry on what looks like an index card for a particular case. This strongly supports and encourages thinking that stresses case-wise and cross-case phenomena, but it makes it harder to look for and think about relationships among codes *within* a text.

Finally, Code-A-Text offers a quite different set of coding metaphors: (1) codes arranged into "scales" (you can assign only one code from each scale to a segment of text, which is useful if you want to code your text chunks by making mutually exclusive judgments on a variety of factors); (2) codes automatically assigned according to words in the text; and (3) open-ended "interpretations" you write about each text chunk. Working with this collection of coding metaphors could be expected to lead the analyst to consider his or her text in somewhat different ways than with one of the other metaphors described earlier.

Similar issues exist in the choice of other types of software, such as textbase managers. InfoTree32 XT allows you to arrange texts in a hierarchical tree, and lets you drag texts around to rearrange them. Folio Views also allows you to create a hierarchical outline but gives you the additional capability of creating multiple, non-sequential "groupings" of texts that you can activate any time you wish. Folio Views and askSam let you insert fields whenever and wherever you want in any given record, while InfoTree32 XT requires that you use a standard field set (of your own design) throughout the whole database. Idealist not only lets you customize the field set for each record, but you can also create a variety of record types, each with its own set of fields, and mix them in the same database. Finally, while most of these programs show you just one record at a time, Folio Views shows you a word processor–like view in which records appear one right after the other as paragraphs. Clearly, each of these programs shows you a quite different view of your data, may encourage different ways of thinking about your data, and has different strengths and weaknesses. It is also true that a clever user will be able to bend each of these flexible packages to a wide variety of different tasks, overcoming many of the differences among them.

Each of these assumptions is both a benefit, for some modes of analysis, and also a constraint. The key, then, is not to get trapped by the assumptions of the program. If you are aware of what they are, you can be clever and work around them. The program should serve *your* analytic needs, goals, and assumptions, not the other way around. Researchers interested in empirical work on the impact of different programs on research are encouraged to refer to Fielding and Lee (1998).

CHOICE ISSUES

Up to this point, this article has stressed repeatedly that a choice must be made carefully to meet the needs of both project and researcher. Such a choice is based on principles such as that the software should be matched to the structure of the data set, to the style and mechanics of the analysis planned, and to the researcher's level of comfort with computers in general. How then to go about making such a principled, customized choice? Four broad questions, along with two cross-cutting issues, can be asked that represent these principles and should guide the researcher to such a choice (Weitzman and Miles 1995a,b). These guidelines for choice have seen wide use in practice since their original formulation, and have proven to be effective for guiding researchers to appropriate choices.

Specifically, there are four key questions to ask and answer as you move toward choosing one or more software packages:

1. What kind of computer user am I?
2. Am I choosing for one project or the next few years?
3. What kind of project(s) and database(s) will I be working on?
4. What kind of analyses am I planning to do?

In addition to these four key questions, there are two cross-cutting issues to bear in mind:

- How important is it to you to maintain a sense of "closeness" to your data?
- What are your financial constraints when buying software, and the hardware it needs to run on?

With these basic issues clear, you will be able to look at specific programs in a more active, deliberate way, seeing what does or does not meet your needs. (See Table 1, adapted from Weitzman and Miles 1995b, for a worksheet to organize such information. Table 2 provides contact information for the programs mentioned in this article and their demo versions.) You can work your way from answering questions, to the implications of those answers for program choice, to candidate programs. For example, if you are working on a complex evaluation study, with a combination of structured interviews, focus groups, and case studies, you will need strong tools for tracking cases through different documents. You might find good support for this in code structures (particularly programs with highly structured code systems, like NUD•IST, and to a lesser degree in programs with flexible code systems like ATLAS/ti), or through the use of speaker identifiers in programs like The Ethnograph, AFTER, or Code-A-Text.

Question 1: What Kind of a Computer User Are You?

Your present level of computer use is an important factor in your choice of a program. If you are new to computers, your best bet is probably to choose

Table 1: Program Choice Worksheet

Issue	*Answer*	*Implications/Notes*	*Candidate Program(s)*
What Kind of a Computer User Are You? (level 1–4)			
Are You Choosing for One Project or the Next Few Years?			
Data Sources Per Case: Single vs. Multiple			
Single vs. Multiple Cases			
Fixed Records vs. Revised			
Structured vs. Open			
Uniform vs. Diverse Entries			
Size of Database			
What Kind of Analysis is Anticipated?			
Exploratory vs. Confirmatory			
Coding Scheme Firm at Start vs. Evolving			
Multiple vs. Single Coding			
Iterative vs. One Pass			
Fineness of Analysis			
Interest in Context of Data			
Intentions for Displays			
Qualitative Only, or Numbers Included			
Closeness to the data important?			
Cost constraints			

Adapted from Weitzman and Miles (1995b).

Table 2: Software Contact Information

Program Name	*Source*	*Phone*	*WWW Address*
AFTER	NOVA Research Company	(301) 986-1891	www.novaresearch.com
ANSWR	U.S. Centers for Disease Control		www.cdc.gov
AQUAD	Günter L. Huber, University of Tübingen, Germany		www.uni-tuebingen.de/uni/sei/a-ppsy/aquad/aquad.htm
askSam	askSam Systems	(800) 800-1997	www.asksam.com
ATLAS/ti	Scolari	(805) 499-0721	www.atlasti.de
Code-A-Text	Scolari	(805) 499-0721	www.codeatext.u-net.com
Decision Explorer	Scolari	(805) 499-0721	www.banxia.com
EZ-Text	U.S. Centers for Disease Control		www.cdc.gov
Folio Views	Open Market		www.folio.com
HyperQual2	Raymond V. Padilla	(602) 892-9173	
HyperRESEARCH	Researchware, Inc.	(617) 961-3909	www.researchware.com
Idealist	Blackstone Software		
InfoTree32 XT	Next Word Software		www.nextword.com
Inspiration	Inspiration Software	(800) 877-4292	www.inspiration.com
Kwalitan	Vincent Peters		www.kun.nl/methoden/kwalitan
Martin	Simmonds Center for Instruction and Research in Nursing	(608) 263-5336	
MECA	Department of Social and Decision Sciences	(412) 268-3225	
MetaDesign	Meta Software Corporation	(617) 576-6920	www.metasoftware.com
NUD • IST	Scolari	(805) 499-0721	www.scolari.com
QCA	Kriss Drass	(702) 895-0247	
QUALPRO	Bernard Blackman	(904) 668-9865	
SemNet	Kathleen Fisher	(619) 594-4453	
Sonar Professional	Virginia Systems	(804) 739-3200	www.virginiasystems.com
TEXTBASE ALPHA/BETA	Bo Summerlund		www.psy.aau.dk/bos/beta.htm
The Data Collector	Intellimation	(805) 968-2291	
The Ethnograph	Scolari	(805) 499-0721	www.scolari.com
The Text Collector	Dennis V. O'Neill	(415) 398-2255	
WinMAX	Scolari	(805) 499-0721	www.winmax.de
ZyINDEX	Zylab	(800) 544-6339	www.zylab.com

a word processing program with advice from friends, and begin using it, learning to use your computer's operating system (e.g., MS-DOS, Windows, or Mac) and getting comfortable with the idea of creating text, moving around in it, and revising it. That would bring you to what we'll call Level 1. Or you may be acquainted with several different programs, use your operating system easily, and feel comfortable with the idea of exploring and learning new programs (Level 2). Or you may be a person with active interest in the ins and outs of how programs work (Level 3), and feel easy with customization, writing macros, etc. (We will not deal here with the "hacker," a Level 4 person who lives and breathes computing.)

Being more of a novice does not mean you have to choose a "baby" program, or even that you shouldn't choose a very complex program. It does mean, though, allowing for extra learning time, perhaps placing more emphasis on user friendliness, and finding sources of support for your learning, such as friends or colleagues, or on-line discussion groups on the Internet. People at different levels seem to have quite different reactions to the same programs. So, for example, a person at Level 2 or 3 might like a program that puts the maximum information on one screen because this allows her to quickly find what she wants and learn the program very quickly. A person at Level 1 might find all that information overwhelming at first, and might take a little longer to learn that program because of it. But, once the program was learned, would probably benefit from the layout in the same way as the more advanced computer user.

Question 2: Are You Choosing for One Project or the Next Few Years?

A word processor does not care what you are writing about, so most people pick one and stick with it until something better comes along and they feel motivated to learn it. But particular qualitative analysis programs tend to be good for certain types of analyses. Switching will cost you learning time and money. Think about whether you should choose the best program for this project, or the program that best covers the kinds of projects you are considering over the next few years. For example, a particular code-and-retrieve program might look adequate for the current project and be cheaper or look easier to learn. But, if you are likely to need a more fully featured code-based theory-builder down the road, it might make more sense to get started with one of those now (assuming you choose one that includes good code-and-retrieve capabilities).

Question 3: What Kind of Database and Project Are You Working With?

As you look at detailed software features, you need to play them against a series of detailed issues.

Data Sources: Single versus Multiple. You may be collecting data on a case from many different sources (say your case is defined as a patient, and you talk with several nurses, the patient's parents and friends, the doctor, and the patient herself). Some programs are specifically designed to handle data organized like this, others are not designed this way but can handle multiple sources pretty well, and some really do not have the flexibility you'll need. Also, look for programs that are good at making links, such as those with hypertext capability, and that attach "source tags" telling you where information is coming from.

Single versus Multiple Cases. If you have multiple cases, you usually will want to sort them out according to different patterns or configurations, and/or work with only some of the cases, and/or do cross-case comparisons. Multicase studies can get complicated. For example, your cases might be patients (and you might have data from multiple sources for each patient). Your patients might all be "nested in" (grouped by) care units, which might be nested within hospitals, which in turn might be nested in cities. Look for software that will easily select different portions of the database, and/or that will do configurational analysis across your cases; software that can help you create multiple-case matrix displays is also useful.

Fixed Records versus Revised. Will you be working with data that are fixed (such as official documents or survey responses), or data that will be revised (with corrections, added codes, annotations, memos, etc.)? Some programs make database revision easy, and others are quite rigid: revising can use up a lot of time and energy. Some will let you revise annotations and coding easily, but not the underlying text, and some will let you revise both.

Structured versus Open. Are your data strictly organized (e.g., responses to a standard questionnaire or interview), or free-form (running field notes, participant observation, etc.)? Highly organized data can usually be more easily, quickly, and powerfully managed in programs set up to accommodate them—for example, those with well-defined "records" for each case and "fields" (or variables) with data for each record. Free-form text demands a more flexible program. There are programs that specialize in one or the other type of data, and some that work fairly well with either.

Uniform versus Diverse Entries. Your data may all come from interviews.

Or you may have information of many sorts: documents, observations, questionnaires, pictures, audiotapes, videotapes (this issue overlaps with single versus multiple sources, above). Some programs handle diverse data types easily, and others are narrow and stern in their requirements. If you will have diverse entries, look for software designed to handle multiple sources and types of data, with good source-tags, and good linking features in a hypertext mode. The ability to handle "off-line" data—referring you to material not actually loaded into your program, such as whole books or manuals—is a plus. Many programs can be tricked into doing this if you are clever about it. For example, you might set up an empty document on your computer corresponding as a proxy to each "off-line" document. You could enter chapter numbers, or some other indexing information into this blank proxy document. Then, as you read the off-line document, you could code the corresponding indexing information (e.g., chapter number) in the proxy. When you do a search, the program will retrieve the reference in the proxy document, and you will know where to look in the offline document. (This strategy is adapted from one suggested in the manual for NUD•IST.)

Size of Database. A program's database capacity may be expressed in terms of numbers of cases, numbers of data documents (files), size of individual files, and/or total database size, often expressed in kilobytes (K) or megabytes (MB). (Roughly, consider that a *single*-spaced page of printed text is about 2 to 3K). Estimate your total size in whatever terms the program's limits are expressed, and at least double it. Most programs today are generous in terms of total database size. A few are still stingy when it comes to the size of individual texts. For example, in some programs when a document goes beyond about ten pages, the program will insist on breaking it into smaller chunks, or will open it only in a "read-only mode" browser.

Question 4: What Kind of Analysis Is Anticipated?

Your choice of software also depends on how you expect to go about the analysis. This does not mean a detailed analysis plan, but a general sense of the style and approach you are expecting to use.

Exploratory versus Confirmatory. Are you mainly planning to poke around in your data to see what they are like, evolving your ideas inductively? Or do you have some specific hypotheses in mind linked to an existing theory to test deductively? If the former, it is especially important to have features of fast and powerful search and retrieval, and easy coding and revision, along with good text and/or graphic display.

If, on the other hand, you have a beginning theory and want to test some specific hypotheses, programs with strong theory-building and testing features are more appropriate. Look for programs that test propositions or those that help you develop and extend conceptual networks.

Coding Scheme Firm at Start versus Evolving. Does your study have a fairly well defined *a priori* scheme for codes (categories, keywords), perhaps theory-derived, that you will apply to your data? Or will such a scheme evolve as you go, in a grounded theory style, using the "constant comparative" method (Strauss and Corbin 1998). If the latter, it is especially important to have on-screen coding (rather than having to code on hard copy) and to have features supporting easy or automated revision of codes. Hypertext linking capabilities are also helpful here. "Automated" coding (in which the program applies a code according to a rule you set up, such as when a certain phrase, or a combination of other codes, exists) can be helpful in either case.

Multiple versus Single Coding. Some programs let you assign several different codes to the same segment of text, including higher-order codes, and may let you overlap or nest coded "chunks" (the ranges of text you apply codes to). Others are stern: one chunk, one code. Still other programs will let you apply more than one code to a chunk, but will not "know" that there are multiple codes on the chunk: they will treat it like two chunks, one for each code.

Iterative versus One Pass. Do you want—and do you have the time—to keep walking through your data several times, taking different and revised cuts? Or will you limit yourself to one pass? An iterative intent should point you to programs that are flexible, invite repeated runs, make coding revision easy, have good search and autocoding features, and can make a log of your work as you go. (See also the question of whether your records are fixed or revisable during analysis.)

Fineness of Analysis. Will your analysis focus on specific words? Or lines of text? Or sentences? Paragraphs? Pages? Whole files? Look to see what the program permits (or requires, or forbids) you to do. How flexible is it? Can you look at *varying* sizes of chunks in your data? Can you define free-form segments with ease? Some programs make you choose the size of your codable segments when you first import the data; others let you mix and match chunk sizes as you go.

Interest in Context of Data. When the program pulls out chunks of text in response to your search requests, how much surrounding information do you want to have? Do you need only the word, phrase, or line itself? Do you want the preceding and following lines/sentences/paragraphs? Do you want to see the entire file? Do you need to be able to jump right to that place in the file and do some work on it (e.g., code, edit, annotate)? Do you want the information to be marked with a "source tag" that tells you where it came from (e.g., "Interview 3 with Janice Chang, page 22, line 6")? Programs vary widely on this.

Intentions for Displays. Analysis goes much better when you can see organized, compressed information in one place rather than in page after page of unreduced text. Some programs produce output in list form (lists of text segments, hits, codes, etc.). Some can help you produce matrix displays. They may list text segments or codes for each cell of a matrix, although you will have to actually arrange them in a matrix for display. Look for programs that let you edit, reduce, or summarize hits, before you put them into a text-filled matrix with your word processor. Some programs can give you quantitative data (generally frequencies) in a matrix. Others can give you networks or hierarchical diagrams, the other major form of data display.

Qualitative Only, or Numbers Included. If your data, and/or your analyses include the possibility of number-crunching, look to see whether the program will count things, and/or whether it can share information with quantitative analysis programs such as SPSS or SAS. Think carefully about what kind of quantitative analysis you'll be doing, and make sure the program you are thinking about can arrange the data appropriately. Consider, too, whether programs can link qualitative and quantitative data in a meaningful way (in terms of the analytical approach you are taking).

CROSS-CUTTING ISSUES

The two main cross-cutting issues are closeness to the data and financial resources. Let us dispense quickly with the latter question first. Software varies dramatically in price. The range of prices for the programs we reviewed was $0 to $1,644 per user (Weitzman and Miles 1995b). That is still the range. In addition, programs vary a lot in the hardware they require to run efficiently. You obviously cannot use a program if it is too expensive for you, if it requires a machine you cannot afford, or if it runs on the wrong platform, say, PC instead of Mac. Happily, the U.S. Centers for Disease Control and Prevention now distributes two programs free of charge via the Web: EZ-Text for qualitative surveys and Answr for unstructured text.

The remaining issue, closeness to the data, is more complex. Many researchers fear that working with qualitative data on a computer will have the effect of "distancing" them from their data. This can in fact be the case. You may wind up looking at only small chunks of text at a time, or even just at line-number references to text locations. This is a far cry from the feeling of deep immersion in the data that comes from reading and flipping through piles of paper.

But other programs minimize this effect. They typically keep your data files on screen in front of you at all times; show you your search results by scrolling to the hit, so that you see it in its full context; and allow you to execute most, or all actions from the same data-viewing screen. Programs that allow you to build in hypertext links between different points in your data, provide good facilities for keeping track of your location in the database, display your coding and memoing, and allow you to pull together related data quickly, can in some ways help you get even *closer* to the data than you can with paper transcripts. If you choose with this consideration in mind, software can *help* rather than hinder your work at staying close to the data.

However, having software that enhances a sense of closeness to the data may not be a crucial issue for everyone. Some researchers do not mind relying heavily on printed transcripts to get a feeling of closeness, while others think that such heavy reliance defeats the purpose of qualitative data analysis (QDA) software. Furthermore, some projects simply do not require intense closeness to the data. You may be doing more abstract work and, in fact, *want* to move away from the raw data.

BENEFITS AND LIMITATIONS OF SOFTWARE USE

The use of good QDA software can enhance, in a wide variety of ways, both the speed of the process and the quality of the results of a qualitative or mixed-methods study. But there are also some important limitations to what even the best software will do for you.

Benefits

One of the first benefits you will realize is that using software speeds up analysis tasks *a lot.* There is simply no way to search for important phrases, create text segments and code them, cross-reference memos to text and codes, create cross-references between parts of the text, and so on, nearly as fast by hand as you can with a computer. Qualitative research can be enormously time-consuming, of course, even *with* the use of a computer: you will still have to read and think carefully about your data. But the time savings in the mechanical tasks can substantially lighten the burden.

Second, software can make it possible to do analyses that are not feasible by hand. For example, with the right software it is easy to go back and recode and then look again for the new results of combined-code searches, or quickly gather together all the intersections of codes in your "Themes" list with codes in your "Diagnoses" list to look for patterns of association. You can link multimedia, such as audio or video recordings of interviews and focus groups, to your transcripts so that you can code for non-verbal as well as verbal phenomena. And the "Boolean minimization" logic that underlies Ragin's (1987) cross-case configural analysis would be daunting without the help of a program like QCA or AQUAD.

Software can also help you to ensure consistency and comprehensiveness. Once you have realized halfway through your analyses that some patients make frequent references to death, you can go back and search for all of the synonyms and euphemisms for death, and code them all. That also means that once you have reached a tentative conclusion, you have much greater power to go back and verify it against the actual data. Your likelihood of finding the data that contradict, or lend even better support to your emerging hypothesis, is now much greater.

Finally, software makes it possible to track many facets of the data in connection with each other. For example, if you need to combine demographic data, scale scores, theme codes, and important keywords in the text to find the relevant chunks of text, you can find software that will help you pull all of those things together.

Limitations

However, important limitations also exist. First and foremost is that software will not *do* qualitative analysis. It can find phrases; it can find text with codes you have applied. It can count occurrences of those things. Some software, like SPSS TextSmart, will attempt a quantitatively based categorization of text responses. But there is no software that can make the kinds of interpretive judgments that are inherent in qualitative analysis.

Second, software is no substitute for methodological training. A disturbing number of papers and grant applications have begun to cite a QDA software package as their analytic method. This is both absurd and misleading, and it is dangerous. The choice of a software package in no way determines the analytic operations that will be carried out, the basis that will underlie decisions about coding, the rationale that will be employed for determining relationships among concepts, or the ways in which the resulting conclusions will be evaluated.

Finally, using software for qualitative data analysis requires new learning. Particularly the first time you adopt a qualitative analysis package, you will be learning a whole new kind of program. Think back to what it was like to first learn WordStar or MacWrite (or whatever word processor you started on), and compare that with what it is like now, when you upgrade or learn a new one. You know what a word processor mainly *does*: you just have to find the new buttons/commands and then incorporate an understanding of a few new features. Now you will be starting at the beginning again.

TRENDS AND DEVELOPMENTS

Some very encouraging trends are being traced in ongoing QDA software development. These fall into two rough categories: Catch-Up and Forging Ahead.

Catch-Up

One of the main trends in the world of QDA software is that developers have been working feverishly to make up for the weaknesses in their programs relative to others. Those programs that have been found by reviewers and users to be wanting in search facilities, flexibility in the relationships between text and codes, memoing, data linking, coding display, and qualitative-quantitative linking, are quickly catching up.

Forging Ahead

In addition, several programs, such as Code-A-Text and ATLAS/ti are developing exciting facilities for incorporating multimedia (both audio and video) for analysis along with text. Various programs are moving toward supporting formatted text, the exchange of data among QDA packages, supporting collaboration, and developing new metaphors for providing access to the data.

HOPES

What hopes should we have for future development? Here are a few:

- *Continued catch-up* so that essential features are available in all programs;

- *More flexibility* in configuring programs to meet specific research needs;
- *More "Pencil-level Richness"*—continuing the trend of giving the researcher the same kind of intimacy with the data that you get from working with paper, allowing the researcher to see not just codes but memos and annotations beside the text as well, as in ATLAS/ti;
- *More data exchange* among QDA packages, so that it becomes more feasible to combine complementary programs;
- *More multimedia support*;
- *More ease-of-use,* lower learning curves, and better tutorials;
- *More support for* quantitative and demographic *variables,* and other facilities for tracking individuals through multi-source data sets;
- *More support for* the *needs of narrative analysis,* such as features to track structures in the text; and
- *More and better tools* for collaboration, such as tools for merging and tracking the work of team members, and for the sharing of data (e.g., over the World Wide Web).

CONCLUSION

Software, if chosen appropriately, can be a substantial and important advantage for qualitative research projects. It needs to be selected with careful regard for the needs of individual researchers in terms of working style, structure of the data, and analytic plans. The potential benefits include speed, consistency, rigor, and access to analytic methods not available by hand.

REFERENCES

Bogdan, R. C., and S. K. Biklen. 1982. *Qualitative Research in Education.* Boston: Allyn and Bacon.

Drass, K. A. 1980. "The Analysis of Qualitative Data: A Computer Program." *Urban Life* 9: 322–53.

Fielding, N. G., and R. M. Lee. 1998. *Computer Analysis and Qualitative Research.* London: Sage.

———. 1991. *Using Computers in Qualitative Research.* London: Sage.

Goetz, J. P., and M. D. LeCompte. 1984. *Ethnography and Qualitative Design in Educational Research.* Orlando, FL: Academic Press.

Huberman, A. M., and M. B. Miles. 1984. *Innovation Up Close: How School Improvement Works.* New York: Plenum.

———. 1983. *Innovation Up Cose: A Feld Sudy in 12 School Settings.* Andover, MA: Network.

Kelle, U., ed. 1995. *Computer-Aided Qualitative Data Analysis: Theory, Methods and Practice.* London: Sage.

Lofland, J., and L. H. Lofland. 1984. *Analyzing Social Settings: A Guide to Qualitative Observation and Analysis, 2nd ed.* Belmont, CA: Wadsworth.

Mangabeira, W., ed. 1996. "Qualitative Sociology and Computer Programs: Advent and Diffusion of CAQDAS" (special issue) *Current Sociology* 44 (3).

Miles, M. B., and A. M. Huberman. 1994. *Qualitative Data Analysis: An Expanded Sourcebook, 2nd ed.* Thousand Oaks, CA: Sage.

———. 1984. *Qualitative Data Analysis: A Sourcebook of New Methods.* Newbury Park, CA: Sage.

Miles, M. B., and E. A. Weitzman. 1996. "The State of Qualitative Data Analysis Software: What Do We Need?" Current Sociology 44 (3): 206–24.

Ragin, C. C. 1987. *The Comparative Method: Moving Beyond Qualitative and Quantitative Strategies.* Berkeley, CA: University of California Press.

Seidel, J. V., and J. A. Clark. 1984. "The Ethnograph: A Computer Program for the Analysis of Qualitative Data." *Qualitative Sociology* 7 (1,2): 110–25.

Shelly, A., and E. Sibert. 1985. *The Qualog Users Manual.* Syracuse, NY: Syracuse University, School of Computer and Information Science.

Strauss, A. L., and J. Corbin. 1998. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, 2nd ed.* Thousand Oaks, CA: Sage.

Tesch, R., ed. 1991. "Computers and Qualitative Data II" (special issues, parts 1 and 2) *Qualitative Sociology* 14, no. 3 and no. 4 (fall and winter).

———. 1990. *Qualitative Research: Analysis Types and Software Tools.* New York: The Falmer Press.

Weitzman, E. A., and M. B. Miles. 1995a. "Choosing Software for Qualitative Data Analysis: An Overview." *Cultural Anthropology Methods* 7 (1): 1–5.

———. 1995b. *Computer Programs for Qualitative Data Analysis: A Software Sourcebook.* Thousand Oaks, CA: Sage.t