Journal of Modern Applied Statistical Methods

Volume 2 | Issue 1 Article 4

5-1-2003

The Way Ahead In Qualitative Computing

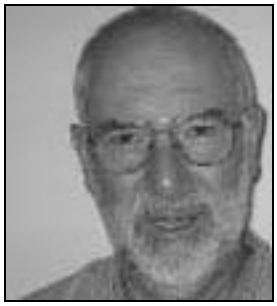
Tom Richards
QSR International

Lyn Richards
QSR International

Journal of Modern Applied Statistical Methods
May, 2003, Vol. 2, No. 1, 16-26

1538 – 9472/03/$30.00

The Way Ahead In Qualitative Computing



QSR International
Melbourne, Australia

Tom Richards Lyn Richards

Specialized computer programs for Qualitative Research in social sciences have greatly changed ways of doing QR, the reliability and comprehensiveness of results, the ability to inspect and challenge a researcher's working, and the relationship with quantitative methods in social research. This article explores these claims in the context of N6 (NUD*IST) and NVivo, the two programs designed by the authors; and considers possible future developments in the field.

Key words: NUD*IST, NVivo, qualitative research, qualitative computing

Introduction

Qualitative Research (QR) has always centered on the analysis of conversational interviews, field notes and recorded conversations. Its raw data are people talking, and the people can be the researchers with their field notes and conversational turns, as much as the interviewees or subjects. Interviews may be one-on-one, or in groups, the records may be live transcripts or historical recollections. Questionnaires may be used, but mainly as topic prompts expecting prose responses not ticked boxes.

Tom Richards is Chief Scientist at QSR International, and designer of NUD*IST and NVivo. He has a D. Phil. in Logic from Oxford University, and many publications on logic, computer science and methodology. Lyn Richards is founder and Director of Research Services at QSR. She has published books and papers on family sociology, qualitative research and QSR's software. This article is based on a presentation given to the American Educational Research Association, SIG Professors of Educational Research, Chicago, April 21, 2003.

The methods and techniques of doing QR are often corralled into a number of schools, Ethnography, Grounded Theory, Phenomenology, and others. From our point of view these are seen as laying stress on different parts of the research process, and the aim of a developer of software for QR is to ensure there are enough tools to keep them all happy. Their actual practices, viewed as tool-users, have much in common: they just prefer to make different products or build them in different ways because they have different research goals.

QR was done manually until about twenty years ago with the rise of the word processor. Preferred techniques involved typing up the interviews or other raw data, and coding or flagging passages about topics of interest with the goal of gathering together all the passages on a given topic. Coding was done by making marginal notes, or photocopying into file folders, or making notes on system cards. This usually required a messy desk or a large living-room floor as a sorting ground. Needless to say these practices were rickety: clerical and management processes were onerous and scarcely fail-safe. Whilst you might do your initial coding thoroughly, it becomes hard to be sure, for example, that you'd

compared thoroughly how a particular viewpoint is presented by people with different demographics or sets of opinions – just because sorting the data into multiple such groups, often cross-cutting, then trying to do side-by-side comparisons, is so hard. Even trying to find vaguely remembered passages about this or that was a matter of luck. These and many other such difficulties we could call the access problem.

Moreover there is the revision problem. Revising your coding in the light of experience was virtually impossible because of the rigidity of handling coding imposed by paper records and coding management. Using manual methods also meant it was impossible to link the data systematically with quantitative research. Demographic data about respondents, or ticked response boxes, could be analyzed in SPSS; but studying interesting qualitative issues arising in conversational interviews with the respondents, in a way that sorted and compared those discussions using the demographic data, was very difficult. Only simple relations could be effectively investigated. Call this the qual-quant problem.

All of this meant that effective QR was best done with small data sets (by no means a bad thing, *n* is not often an important parameter in QR), or conclusions were impressionistic and bolstered by "juicy quotes" rather than dispassionate analyses. Checkability and the reaching of agreement suffered too: disputes over the conclusions reached by a researcher were hard to resolve since there was no way of reviewing the analysis steps. It was more a matter of starting again with the raw data.

The Rise of Qualitative Computing

If the above characterized QR without computers, how did computing help? Early experiments with electronic files in a word processor improved on the manual situation. Codes could be inserted [like this] in the text, and word search would find all the instances of a code, enabling inspection of their passages. This greatly ameliorated the access problem, but clerical organization of codes, and their comparison, remained elusive. These problems led to the rise of the early dedicated QR programs, which basically provided tools for coding text documents, storing the coding references (usually to lines), and using them to find and display all passages referred to by a given code such as 'playground bullying'. From the first dedicated QR programs, simple Boolean searches were supported, thus you could find all passages coded by both 'playground bullying' and 'fear of going to school'. These features were much prized, because researchers could explore, with confidence of completeness, hunches about relationships between different situations or concerns or attitudes; and that is the way qualitative theories are built and tested.

This process came to be known as code-and-retrieve, and because it was computationally simple to program, became the hallmark of computer-based QR. As we shall see however, this was a somewhat limiting approach to QR. For one thing, researchers couldn't edit the text of their data any more, because to do so would invalidate the coding references made to the text passages; yet flexibility of amending, adding to, fleshing out, the text was a desirable tool for qualitative researchers that word-processing had provided.

Nevertheless code-and-retrieve has formed the core of all QR programs to date. Many of the current software offerings however provide much more than that. This, and the future, is what the rest of this paper will look at, in the context of QSR's two QR programs.

Methodology

QSR has two products for qualitative researchers, NVivo and NUD*IST (Non-numerical Unstructured Data Indexing Searching and Theorizing, a name given when it was being programmed by one of us (TJR) for sole use by the other of us). Its latest version is known as N6 in an attempt to suppress a name, which, however memorable, definitely should not be searched for using a Web search engine! NUD*IST was first used by LR in the early 1980s, and went commercial in 1986 with the sale of one license (on a university mainframe and with scroll-mode display!). NVivo was launched in 2000. These are very different products, and aimed to support different work practices, as will be described below. Right now however, our aim is to set out how these products both go beyond the code-and-retrieve paradigm just described.

Edit-While-You-Code

We pointed out above that a restriction imposed by code-and-retrieve was that you

couldn't edit a document – it was frozen. The reason: editing would, by adding or removing text, invalidate the references made by coding to passages in the document. Add a hundred characters at a given point and every reference by every code to passages later in the document will now pick up text a hundred characters before what it used to. Back in the days of paper the problem was different and not so bad. If you coded by photocopying passages to folders of codes, then if you altered the original the coded copy in the folder was unaltered, but might no longer be faithful to the altered original.

Researchers do want to make corrections to interview transcripts, to do partial transcription and flesh it out later as the direction of research indicates, to edit out privacy-infringing material, to add clarifications and greater detail to field notes. Researchers also want to code while they are typing up the transcription, because that's often when they have their best thoughts about what the text is saying and implying and hinting and suggesting. The restriction that all your data documents must be complete and final before you dare to add one code, is a strait-jacketing QR cannot accept.

Aside from the ability to add text at the end of document, which doesn't upset any existing coding, N4 and onwards has provided the ability to edit individual lines or paragraphs – the text units that are the smallest chunks of text that can be coded. NVivo however codes all the way down to individual characters, and moreover supports rich text documents, not just plain text as in N6. Despite this, NVivo supports full editability. Its Document Browsers, where you look at the text of a document, have full editing controls plus controls over the "richness" of the text – font, letter style and color and size, etc. And using the text editor does not in any way invalidate existing coding: NVivo's way of recording coding keeps up with editing changes. So for the first time ever, researchers can feel completely free to modify their documents, and to code them while writing them up.

Nodes – Going Beyond Code-and-Retrieve

The world of QR, including QR computing, talks of codes as the labels attached to and describing the contents of, passages of text. The process of coding is the labeling of the text, and retrieval of a code involves presenting, somehow or other, the passages referenced by the code.

But both of QSR's products store coding at nodes. These are containers for topics, ideas, places, people, and attitudes, indeed anything that may be relevant to the QR project at hand. There may, for example, be a node 'Schools' which has under it sub-nodes for the schools in the project 'Valley High' and 'Hilltop Primary' for example. 'Valley High' might contain just a memo written by the researcher describing the school and its problems, and 'Schools' contain nothing – it's there just as a generic locator for the nodes for individual schools (this demonstrates why we chose the word 'node' for these entities, and why the two programs can organize nodes in a tree-structured hierarchy like a library catalog or a taxonomy).

Many nodes will however contain coding. If an interviewee talked about Hilltop Primary, it's appropriate to code that passage at the 'Hilltop Primary' node. And of course some nodes are intended primarily for coding, such as 'angry' (marking where interviewees displayed anger) or 'reports of bullying'.

Nodes can also be used to mark cases. If we have ten interviewees, who got interviewed individually a couple of times then in groups, it is useful to collect everything each individual said in one place. This gives rise to case nodes 'Mary', 'Joe', etc., instances of the case type 'Interviewee'.

It's a small step beyond that to use trees of nodes to represent demographic data – called base data trees in N6. (NVivo represents demographic data in tables of so-called attributes). Thus we can have a 'Religion' node, with sub-nodes 'Christian', 'Hindu', 'Jewish', etc. Then if Joe is Jewish, we copy all the coding at the 'Joe' case node to the 'Jewish' node. And the same goes for any other Jewish interviewee case. Why do this? Because now, using the ability to make Boolean combinations of coding at nodes, we can immediately find everything said by Jewish interviewees. And if we have coding at 'Hilltop Primary' and 'reports of bullying' we can find all reports by Jewish interviewees about bullying at Hilltop Primary.

Both N6 and NVivo support importing and exporting such demographic data as tables. For example an SPSS table, whose rows are the

Interviewee cases 'Joe' etc, and whose columns are variables such as 'Religion' etc., can be imported into N6 to create and code up an entire base data tree. Conversely such a tree (which may be created inside N6 rather than imported) can be exported as a table to any table-handling program. NVivo does this more directly with its attribute tables; but in either program a researcher might create a base data type of tree that records research results, perhaps various categories of social, political or educational opinion the researcher has labeled the interviewees with as a result of careful analysis of what they've said. An example would be, for parents or teachers discussing ideal curricula: 'Curriculum priority/vocational', 'Curriculum priority/all-rounder' and 'Curriculum priority/none'. The exported table would record which case (interviewee) belongs to each category.

Nodes with coding represent views onto the textual data of a project that are orthogonal to that provided by documents. Any QR program will let you view the contents of a document, e.g. the first interview with Joe. In NVivo and N6, a Document Browser, like an edit window in Microsoft Word, shows you all the text in that document. A node by contrast refers to all passages that have been coded at it. How do you see such passages? In both N6 and NVivo, and unique to these programs, you can view everything coded at a node (in a browser window) in just the same way as you can view a document. This contrasts with being taken to each document in turn with the coded passage highlighted, or a series of cards holding the different passages. In the Node Browser, you can ask to see not only the passages coded, but as much of the context of those passages as you wish, if that helps to understand them.

Now when you're browsing a document, you can of course code it. Both products provide comprehensive tools for making, viewing and modifying coding in their Document Browsers. But uniquely, they also provide exactly the same coding facilities in their Node Browsers. Since Node Browsers are the place to find and compare nuances in what the node is about, and the place to find what people are or are not talking about in the context of the topic of that node; the Node Browser is the place to code up those nuances and found topics – leading to lots of rich and deep analysis that might well be unrealizable otherwise. This process is called Coding On, and is made possible by "live" Node Browsers that display their text in context and support coding.

Difficult in the days of paper, the advent of the live Node Browser has made Coding On a simple and universally available tool for qualitative researchers, who are still exploring the power it gives them.

Linking: Making the Web of Associations

Edit-while-you-code and the live Node Browser are, in the end, ways of removing fetters from coding. Now we will look at a bunch of tools that are not about coding at all, although the Node system and coding can certainly interact with them. These tools are about making links or associations, involving documents, nodes and other things.

Memos and Links

Most qualitative researchers want to keep notes, commonly called memos, about their data and idea. If you have a one-on-one interview with Joe, you may want to have a memo about how Joe behaved in the interview, your thoughts about Joe, and the like. Most QR programs will support writing such a memo, attaching it to Joe's interview, and adding to it and revising it later.

Such memos can be a valuable source, or indeed explicit repository, of research insights – where the researcher records their evolving thinking about aspects of the project, for instance the rise of a climate of fear and the many ways it interferes with self-esteem. In such a case it seems obvious to link such a memo not to some interview, but to the nodes on fear and self-esteem. N6 and NVivo support that. But more importantly, there is a felt need to code such memos, at anything of research importance they may say. N6 supports this by allowing a memo to be turned into a data document where it can be coded. Obtaining first-class status, if you like. In NVivo, all memos have first-class status anyway. They are no different from any interview document – except that they are called memos.

A memo can be linked to several nodes and documents, so that when you are browsing them you can see they have memos and you can open them in new Browsers. In addition a memo can be linked to any point in the text of a document where it may be relevant, so you see a

little link icon in the text and can access it from there. These in-text links, and others we will be talking about, are all visible in Node Browsers too, and can be accessed from there. And if a memo is rather general in nature, such as a research plan or summary, it needn't be linked anywhere at all, but will still be listed along with all other documents in NVivo's user interface. After all, it is a document. And since it's a document, it can contain links of its own. In this way we can build up a web of links between documents and documents, memos and other documents, nodes and memos or documents. For many researchers, these provide a new way, different from coding, for associating and exploring ideas, topics and themes.

Links can also be to nodes, which provides a sort of converse of coding. If a passage in an interview refers to Joe's peculiar views about sport in the school curriculum, we can insert a link right there in the text, called an extract, to the passage in Joe's interview where he expresses those views. When you set up an extract, the passage being extracted gets put into a node, the extract node, which is what the link in the text jumps you to.

Hyperlinking to Other Data

So now in NVivo we can put into the text a fabric of links, joining documents (whether memos or not) to each other, nodes to each other, and between documents and nodes. In addition to such links, marked by little icons in the text, there are more standard hyperlinks to short comments or, significantly, to computer files and web documents. This means that material of any sort at all can be referenced at any place in a document – pictures, web pages, spreadsheets, movies, … and opened there in its appropriate program. This provides the ability to code such linked items as wholes, by simply coding the hyperlink in the document. In the case of audio and video files, by judicious use of programs that will "snip up" such files, you can attach just the relevant part of a video, for example where Joe is getting worked up about school sport, to the hyperlink. For many researchers, this way of handling the coding of videos is preferable to coding the video file directly. Moreover such links, like document and node links, are always visible and live when presented in Node Browsers, not just in the editable Document Browser. The ability to make associations, and to link the web of associations with nodes and coding, is now comprehensive.

Beyond Retrieval: Asking Questions

Retrieving the text coded at a node may be interesting and illuminating, and lead to a lot of valuable coding-on; but it doesn't show you anything new – you did all that coding. But finding simple Boolean combinations of coding does offer new knowledge. Simple intersection (and) is particularly effective: Given a demographic code such as 'gender/male' and a "thematic" node such as 'bullying' intersection will show us everything the males have said about bullying. We can by the same procedure put that result alongside what the female interviewees have said about bullying – a contrast likely to be productive of insights to code-on.

A couple of thematic nodes such as 'playground' and 'bullying' can be intersected to see what's said about bullying in the playground, and a similar search will lead to a contrast with bullying in the classroom. Using the Node Browser facility to see retrievals in context will counteract the way that intersection narrows down its finds.

Several nodes need to be intersected to answer some questions, such as "What do Jewish fathers have to say about playground bullying?" (intersecting four nodes).

N6 and NVivo handle all these searches using a facility called a Search Tool. This supports all other Boolean search operators, so that for example you can ask "What is said about bullying that is not in the playground?" A large range of proximity searches are also provided so that you can ask questions like "Amongst the people who talk about bullying, what do they say about fear of attending school?" – a simple example. The fact that nodes can be organized hierarchically for cataloguing purposes is not forgotten either. So if the 'curriculum priority' node has sub-nodes 'vocational', 'all-rounder' and 'none', you can ask to retrieve all the curriculum priority views (nodes below the 'curriculum priority' node) and see them together.

Well, where, how, do you see them together? From the earliest versions of NUD*IST, and in NVivo, the results of any search for any combination of coding has always been stored at a

node. This sort of reflexivity, where results of analyses get stored as new data, is called system closure. It allows the researcher to view the results in a Node Browser, and hence code on – a very fruitful activity with the results of interesting searches. It also allows new questions to be asked involving the search results at any later date. For instance, having stored the answer to "What do Jewish fathers have to say about playground bullying?" as a node, you might ask "Amongst the Jewish fathers who spoke out on playground bullying, what do they have to say about other forms of bullying?" – a proximity search. Such questions are crucial in QR, but how would you get answers to them in a paper-and-file-cabinet research project?

System closure can have significant effects. Consider text search for example, which is supported in comprehensive ways in both products; involving pattern specifications as well as search strings, and in the case of NVivo, approximation searches to allow for misspellings and the like. Given system closure, text search is presented not merely as a way of displaying the next match in the next document; but as a way of collecting all the finds together, optionally in their sentence or paragraph context, and storing them at a node. This not only allows for the sort of coding-on described above, but also means that text search can be brought into the sort of combinatorial searching just described, since the node holding text search results can be input to a Boolean, proximity or other search. It also means that text search can be used as the first rough pass for coding. You make a node holding the passages found by searching for 'Napoleon' and 'Bonaparte', then add to that by coding when you find indirect references to him.

When you have such comprehensive search tools available, enabling you to ask just about any question expressed in terms of nodes in the project, the task of designing a coding system becomes very much easier, and the resulting system far more flexible. Without such tools, you'd need to ensure you've got all the different responses well catalogued by coding 'Joe on playground bullying', 'Joe on classroom bullying', 'Henry on playground bullying', 'Jewish fathers on classroom bullying'; and so on repetitively to create a morass of combinations of topics and demographics. And all you could do in the end would be to retrieve them, and asking novel questions like "Do older parents have different views on the effect of teacher discipline on the control of playground bullying than younger parents?" would not be possible. You'd simply have to go back to your documentary data and code for that from the beginning.

Whereas, aware of the power of the search tools, you need to code only for some demographics amongst parents interviewed, and for 'parent', 'bullying', 'discipline', 'teacher', 'playground', 'classroom'; then you can ask the questions in the previous paragraph and many others. For instance you find everything said on playground bullying by intersecting 'playground', holding everything said about playgrounds and happenings in them, with 'bullying', holding everything said about bullying. This makes for simple-to-code, clean, easily organizable node systems that lend themselves to powerful searching, and the crucial ability to make unforeseen searches.

A final but very powerful feature of searches needs to be mentioned. Most of the search operators such as intersection can be applied to not just a pair of nodes (to find their intersection) but to two groups of nodes to create a table or qualitative matrix of their pairwise intersections. For example, to find the views of parents of different religious persuasion on the different curriculum priorities, you take the node 'Religion', (below which are 'Jewish' etc.) and the node 'Curriculum priority' (below which are 'vocational' etc.) and you get a table whose cells show what everyone of each religion has said about each curriculum priority. The matrix is stored as coded data, with each cell effectively a node that can be viewed in a Node Browser, where it can be coded-on, used as input to some other search, and so on. A table of numerical data on the cells such as amount of text coded, can be exported to a table-handling program such as SPSS (if the researcher thinks that might be statistically useful data!).

The above outline gives an insight into what N6 and NVivo can do, and a taste of what it's like to work in such a program. There is a great deal more that can be said, but these are complex and powerful programs, and it would be best to visit the literature on them.

How Do N6 and NVivo Differ? Two Worlds of Work.

In spite of all the common features and tools described above, N6 and NVivo are two rather different products that address two rather different ways of working. One simple example has been mentioned: demographics in N6 are handled in base data trees, but in NVivo in tables of attributes of documents or nodes.

The best way to sum up the differences between the two is that N6 and its forebears are designed for rapid access to textual data via coding, whereas NVivo can handle very complex data with a large variety of tools. Think of NVivo as flexible and subtle, suited for deep analyses as in a typical university PhD project; and N6 as containing a single workmanlike tool that nevertheless provides powerful analyses. Let us spell these out.

N6 requires its data documents to be plain ANSI text, whereas NVivo handles rich text in any font at all. Rich text is more attractive than plain, and is needed of course to display hyperlinks and link icons, but aside from that it gives the user the opportunity to mix languages in a document, to do "visual coding" by highlighting, and to use up to nine level of heading to divide a document into nested subsections. While this presents more opportunities to the researcher, it comes at the price of increased storage demands and slower text handling. In large volumes that can matter, whereas the plain text of N6 makes minimal demands.

For purposes of coding, N6 requires all text to be divided sequentially into text units, which the user can define as sentences, lines or paragraphs. These are the smallest passages of text that can be coded, whereas NVivo supports coding right down to the character level. Fine coding presents better opportunities for researchers interested in the details of what people say, enabling them to pick up words, phrases, and stylistic quirks. At the other extreme, coding at the paragraph level in N6 means it is easy to provide coarse coding economically to enormous volumes of text; and typically for large projects with thousands of interviewees, paragraphs are quite small enough thank you.

The above two features, combined with its ability to automate data handling (see below on Command File scripting), mean that N6 can handle enormously large projects, limited in general only by the computer's speed and storage. We know of projects containing tens of thousands of multi-page interview documents, handled well by N6. NVivo would slow down unacceptably on such a big dataset – the recommended maximum is hundreds of documents if they are large and coded to a reasonable level. Of course, even that is no small project.

N6 essentially uses one data type, nodes and their coding, to handle everything – aside from having documents of course. And there's only one analysis tool, the combinatorial Search Tool described above. NVivo on the other hand has not just nodes and their coding, but comprehensive links as described earlier. It also has sets for grouping documents or nodes in any ways at all. And as mentioned, NVivo avoids the need to use nodes and node trees to handle base data by having a comprehensive attribute/value data type. This is used to set up attributes for documents or nodes, and to assign values to individual documents and nodes – string, Boolean, numeric or date.

As to analysis tools in NVivo, sets have a very comprehensive filtering editor, and attributes have live table displays. In addition there is a Show Tool, for finding lists of related items – all the documents with a particular attribute value for example, or all the nodes coding a given document. And there's an Assay Tool for looking at the numbers of documents or nodes in a set that have any selected feature – all presented in tabular format with marginals, ready for export to SPSS or other table-handling package.

Moreover Nvivo's Search Tool is more complex than N6's. Information can be located not just by coding at a node, but in values of attributes, and of course in text search finds. So NVivo's Search Tool supports Boolean, proximity and Matrix searches, as in N6, but can take as input attribute values and text-search patterns as well as nodes with their coding. In N6 a question like 'What do Jewish fathers say about classroom bullying' is framed as intersecting three or four nodes (depending on whether you've coded 'classroom bullying' as one node or preferably two inviting intersection). In NVivo the intersection would be of attribute-values 'Religion=Jewish', 'Role=father', and nodes for classroom bullying as before. And if you want to

find where parents talk about the curriculum you don't have to do a 'curriculum' text search first, save the node then intersect with a node or attribute for being a parent. You just intersect the latter with paragraphs containing the word 'curriculum'.

N6 has a scripting tool called command files, allied with a Command Assistant that helps researchers construct complex series of commands to handle large jobs. These can be used over and over again (with editing if need be to change parameters) to cover repetitive work – in the one project or in a series of essentially similar projects. This provided great speedups for many parts of project work. It can even be used to analyze the comparative performance of many coders in a collaborative or multi-site project. NVivo has no scripting, but provides more interactive tools to assist with some complex routines.

NVivo contains a graphical tool for visual exploration of a project's data and their relations. Nodes, documents, sets, attributes and their values can be placed in layers of a graphical model, each being live to its contents (click on a node in a model to open its browser). In addition to the links and groupings a researcher might draw in a model, links can automatically be added to show which nodes code a given document, which documents have a particular attribute value, and the like. Social scientists use "box-and-line" drawings to display a theory or some process or organization in the world, and the graphical modeler is designed to give them great freedom in preparing such diagrams, live to the underlying data. It makes for a great presentation tool! The workmanlike N6 contains no such graphics.

Both products have an associated "Merge" program designed for combining two separate but essentially similar projects into one. They look, for example to see if two same-named documents in the projects are in fact the same in which case their coding can be combined, otherwise treat them as different and change the name of one on merging. N6 treats this merging as essentially a hands-off "batch" process. You set the parameters and let it run. Merge for NVivo however works far more interactively. Before the merging you are taken through an interactive alignment process of examining all potential clashes (like same-named documents) and deciding what to do. At the end of alignment you can stop, having "sorted out" the two parallel projects so they compare correctly in their document, node and attribute systems, or you can proceed to merge the two of them.

N6, then, is simpler – in its plain text, in its types of data (nodes only) and in its tools and displays. However people working on deep subtle projects, usually in a university research environment, find that compared to N6, NVivo really helps them to soar. It is exhilarating in its richness and flexibility and ways of comparing and showing information. People with simpler needs prefer N6 – there is less to learn and the power remains great. N6 is also the product to use for large projects, which are becoming quite common especially in government or semi-government research organizations, where they are closely allied with extensive quantitative surveys.

These two types of work – simple but powerful and scalable versus complex, flexible and subtle, do effectively divide the field so that most people moving into QR computing recognize which program suits their needs best. These two ways of working, two types of project, are so different that it is unsatisfactory to try to provide one program that handles both excellently – instead you end up with a lowest-common-denominator program.

How Qualitative Research is Changing

One of the privileges of being the designers of these programs is to travel the world visiting universities and institutions in very many countries to conduct workshops with users and consult on their projects. This gives a unique insight into how qualitative research is changing under the impact of these computing tools over the last decade. Here are some of the headline changes we have observed since about 1990.

The areas employing QR, especially by computer, as a fundamental tool are broadening. Initially projects and people seemed to come from sociology, educational research, and (intriguingly) areas of engineering. Now there is far more qualitative research in business and organizational studies and consulting, demographics-oriented disciplines such as epidemiology, health sciences (which itself has been a burgeoning discipline over this period), and business-based survey research e.g. market research. Interestingly, history and literary studies remain somewhat aloof.

QR by computer (especially if you're using NVivo) is used to handle a research project end-to-end, not just to analyze filed notes or interviews. All project documentation – project plans, progress summaries, and importantly the research summaries and reports and presentations – are kept inside the NVivo project. This reflects the murky dividing line between data and analysis, and the value of using the linking and coding tools in particular to relate "research" to "data".

Size of project has increased enormously. Whilst the median size, perhaps a hundred documents at most, remains unchanged, there is a growing tail of huge projects driven by the desire to provide some sort of qualitative analysis of studies with very large n, and to inform quantitative analysis with such data. Common fields here are government studies, epidemiology and population-wide health studies, global studies by international organizations, and the like. Some specific examples are learning-effectiveness studies of students of a given age across all schools in a state or country, district-by-district analysis of the effects of a new country-wide public safety system, and customer feedback worldwide (where customers are governments) of utilization of major infrastructural capital goods. These may not excite the NVivo-using sociologist who uses QR to develop a theory of social behavior, but their importance, and their need for QR, is great and usually of immediate relevance to communities. They are also all projects suited for N6.

Qualitative-quantitative wars have largely been replaced by collaboration. Some recalcitrant pockets remain, but the change has been remarkable. Of course for many projects on either "side", there is no need for collaboration; but the incidence of collaborative or mixed-method projects as they are being called, is increasing sharply. The presence of software, particularly the NUD*IST line over the years, seems to have been quite instrumental here. Two reasons. One, the qual-quant problem mentioned at the start of this paper has been considerably ameliorated by table-handling facilities within qualitative packages combined with table import/export facilities. Thus for example intriguing numerical patterns arising from a matrix search pitching some demographic attribute-values (themselves imported from survey data) against a range of viewpoints elucidated in interview conversations, can be investigated statistically to test significance or to graph a correspondence analysis.

Reliability is being taken seriously. When the access problem loomed large, researchers tended to erect a number of "monster-barring" defenses here – it's a matter of insight and experience irreplaceable by mere machines, for example. Some defenses by qualitative researchers were quite correct though, for example QR doesn't require a large *n* to give it reliability or validity (though some funding committees still think so). After all a biography (*n*=1) can provide tremendous insight into a personality type, a period of history, or a social situation. What matters more is that a QR project carried out in say N6 provides far more auditing of the conclusions a researcher makes. The use of the Search Tool to find the insightful "core" concepts that give an understanding of the problem at hand, can be traced as the results are preserved as nodes. Another researcher can get into the same project and use the Search Tool on the very nodes the original researcher built, to find counter-examples or problematic cases that challenge the original conclusions. Coding patterns can be studied quite directly to see how even they are across the data, and N6's Command Assistant can even produce a script which will compare the coders in a team to find similarities and differences in their coding patterns.

Analysis is going far deeper. Even with smallish projects, the access problem and the clerical time consumed used to put a close limit on the results discoverable and on their exploration. Now however there is little time cost in exploring a large number of hunches and approaches, of combining them and extending them in many ways; in short in encouraging serendipity then putting the discoveries through rigorous analysis.

Readings

There is a surprisingly small literature on computational QR, given its ubiquity and the effect it has had to change methods. The series of conferences since 1999 at the University of London, Institute of Education on doing research with QSR's software have led to a special journal issue: International Journal of Social Research Methodology (2002a), 5:3.

Amongst its many articles there is a most important discussion of mixed methods by Bazeley (2002a). The evolution of NUD*IST and NVivo is described in T. Richards (2002). An examination of the effect computing has had on QR methods is set out in L. Richards (2002a; 2002b, 1998). Mixed methods are also discussed in Bazeley (2002b). Bazeley & Richards (2000), Morse & Richards (2002), and (Gibbs, 2002) are three books about how to do QR by computer. The first has a gentle mentoring approach for someone new to qualitative computing; the second is more methodology-oriented (ethnography, phenomenology and so on), while the third takes a more standard text-book approach to the subject.

There are no recent survey books of this fast-changing field. The latest Alexa and Zuell (1999). For a much more comprehensive bibliography of books and articles in the field, visit the following url: http://www.qsrinternational.com/resources/literature/reading.htm.

Conclusion

The world of computing and software is notoriously unpredictable, which is probably why it has such a huge number of gurus doing the predicting. What shape qualitative research programs will take in ten, or even five years' time is very indeterminate. Arrival in the market by a large established software vendor, or the development by some genius of an unforeseen way of doing QR with computers, can upset any prediction. After all the development of new ways of working has been the hallmark of computing in QR in the past, so why not in the future?

On the other hand, the pressures that might shape QR program revisions in the nearer future can be spelled out. Here are some:

The rise of mixed methods, the demand for better qual-quant interaction. This is unlikely to lead to a program that does both, but will lead to innovative thinking on how qualitative programs can better hold up their half of mixed methods. The shape this will take is unforeseeable – something new in research methods may well arise here.

The application of "intelligent" heuristics. Using natural language semantics automatically to code documents to the level of intelligence of a trained researcher, can safely be said to be a very long way off. But there are plenty of more modest artificial intelligence and statistical routines that can be applied to find inductive relationships, to find various sorts of associations between the coding of nodes, and to do data mining as a way of suggesting new and fruitful nodes.

The pressure to handle large projects with large datasets. These can be projects where one person or a small local team is studying huge amounts of data. Or there can be multiple researchers gathering data, or joint projects running in several different sites requiring a unified organization and various levels of comparison of site data.

Handling repetitive or multiple similar projects. Particularly in the business-driven research world, a successful project will modeled for re-application in similar situations, and hence require the easy definition of its model "skeleton" then easy fleshing-out to the new projects. This can include aggregating the repetitions to an "overall" project.

Exploiting the Internet. This is not just finding project data in emails and web pages. The Internet provides a ready-made remote networking and data storage system for people collaborating on projects from multiple sites, and for providing remote and special or customized processing of project data.

New modes of user interaction. QR famously makes huge demands on organizing and displaying data, having huge amounts of disorganized data. Early versions of NUD*IST relied on the scrolling 24 x 80 character display of "glass Teletypes" – which still held sway only 20 years ago. High-resolution color graphics screens, windowing and mousing are all quite recent arrivals, and certainly by no means the last word in user interaction and control. When the next breakthrough arrives it is likely to desert the desktop-and-paper metaphor that the current windowing interface is based on, and provide unforeseen opportunities for novel organization and display of qualitative data.

Given all this, the last word is that we have not yet reached the last word.

References

Alexa, M., & Zuell C. (1999). *A review of software for text analysis*. Mannheim: ZUMA.

Bazeley, P. (2002a). The evolution of a project involving an integrated analysis of structured qualitative and quantitative data: from N3 to NVivo. *International Journal of Social Research Methodology, 5*(3), 229-243.

Bazeley, P. (2002b). Computerized data analysis for mixed methods research. In A. Tashakkori, & C. Teddlie (Eds.), *Handbook of mixed methods for the social and behavioural sciences*. Thousand Oaks, CA: Sage.

Bazeley, P. & Richards, L. (2002). *The NVivo qualitative project book*. London & Thousand Oaks CA, Sage.

Gibbs, G. (2002). *Qualitative data analysis: explorations with NVivo*. Buckingham, Open University Press.

Morse, J., M., Richards, L. (2002). *Readme first for a user's guide to qualitative methods*. Thousand Oaks & London, Sage.

Richards L. (1998). Closeness to data: The changing goals of qualitative data handling. *Qualitative Health Research*, (*8*),3, 319-328.

Richards, L. (2002a). Rigorous, rapid, reliable and qualitative? Computing in qualitative method. *American Journal of Health Behavior*, *26*(6), 425-430.

Richards, L. (2002b). Qualitative computing – a methods revolution? *International Journal of Social Research Methodology*, *5*(3), 263-276.

Richards, T. (2002). An intellectual history of NUD*IST and NVivo. *International Journal of Social Research Methodology*, *5*(3), 199-214.