5

Codes and coding: principles and practice

Coding provides a means of purposefully managing, locating, identifying, sifting, sorting, and querying data. It is not a mechanistic, data reduction process, but rather one designed to stimulate and facilitate analysis. Either explicitly or implicitly, it is a necessary step in most approaches to qualitative analysis, yet forms of coding, approaches taken to coding, and specific purposes for coding vary enormously.

In this chapter, develop an understanding of:

- when and how to make use of coding;
- different kinds of coding;
- strategies for managing the process of coding;
- how coding strategies differ in different situations;
- reliability and validity in coding.

Read this chapter in association with the next. As you begin to apply codes, you will need also to know how to identify and name coding categories.

Using codes to work with data

Coding is a fundamental skill for qualitative analysis. Coding is not an end in itself, but a purposeful step to somewhere. It provides a means of access to evidence; it is a tool for querying data, for testing assumptions and conclusions. But what makes for proficient coding? And how does one become a proficient coder? Four things stand out for me in reply to these two questions: responsiveness to data; focus on purpose; learning through observation of and discussion with experienced others; and practice. You *can* learn to code effectively and well.

When you code for qualitative analysis, you will label a passage of data with a code based on your understanding of what that passage is about. The label (code) is then used both to *represent* and to *access* that passage along

with other data that are the same or similar. When you review the code, you have access to and can retrieve all the data represented by that code. Qualitative coding is about data retention, rather than data reduction (Richards, 2009).

Coding typically moves through at least two major stages: an initial stage of identification and labelling, variously referred to as first-level, initial, or open coding (using *a priori* or emergent codes); and a second stage of refining or interpreting to develop more analytical categories or clusters, often referred to as focused coding (Saldaña, 2009). Initially your focus will be on working through your sources; later in the project you will be focusing your thinking more around your codes and coded data.

Different kinds of codes are used to help you focus and develop ideas. Codes range from being purely descriptive of circumstances (sometimes referred to as structural codes) or descriptive of actions or events or experiences, to categorising topics or issues, through to naming more interpretive or analytical concepts (Gibbs, 2007; Richards, 2009). Thus, this event occurred *in the playground*, it describes an act of one child *ridiculing* another, it is about *bullying*, it is a reflection of *cultural stereotyping.* Coding is 'entirely dependent on close reading – they are not mechanical tasks. It is not good enough to skim a transcript or set of field notes and to have a broad sense of "what it's all about", cherrypicking bits of data for quotation' (Hammersley & Atkinson, 2007: 162). When codes involve interpreting data within their context, they give meaning to data. The task of coding assists the researcher to break out of an 'imprisonment in the story' to see new connections and alternative ways of framing and interpreting a text or situation (Maxwell & Miller, 2008: 469).

The evolving understanding that occurs as you move from data to description to analysis means that coding occurs in a cyclical, or recursive, process (Miles & Huberman, 1994). First, new data may add new categories to the analysis. Categories that appear interesting but which are inadequately developed prompt you to seek further data. Inevitably, initial codes will be revised as work proceeds, necessitating review of sources coded earlier. Finally, categories that are developed during coding will be further reviewed and refined before final analysis proceeds. 'Codes are organizing principles that are not set in stone. They are our own creations, in that we identify and select them ourselves; they are tools to think with. They can be expanded, changed, or scrapped altogether as our ideas develop through repeated interactions with the data' (Coffey & Atkinson, 1996: 32).

Tips for coders:

- Codes can be descriptive, topical, or analytical; codes can be used to serve many purposes.
- Use coding to stimulate thinking about what is going on in the data.

- ✓ Coding helps you to attend to the detail of the data. This may be the most valuable contribution of the exercise of coding data, for some researchers.
- ✓ Your coding will develop, so never be afraid to reconfigure a code or coding system (archive the earlier version for security).

Use coding to:

- manage (keep track of) your data;
- build ideas from your data;
- facilitate asking questions of your data.

Coding to manage (keep track of) data

When you go to use a reference book, two of the most valuable tools in being able to make use of the information in that book are the contents page and an adequate index. Coding data has been likened to indexing a book (Kelle, 2004; Patton, 2002). An entry in an index provides a pointer to where something can be found in the text. Indexing doesn't imply or involve segmenting text (cutting it up); rather, it is a reference to a place in the original source where you can find relevant material. Indexing is an especially useful strategy if you are working on paper, as it allows the records to be maintained in their entirety, while also allowing you to access and review all the material on a particular topic (Hammersley & Atkinson, 2007). Retrieving everything you know about a particular topic or drawing comparisons across groups or contexts for each coding category becomes rather complicated if you rely on indexing, however.

Coding takes you a step further than indexing: as well as helping you locate data, it serves as a way of *sorting* and ordering data – 'the undigested complexity' of field notes, transcripts, documents, pictures, ideas – so that you can keep track of what you have on any topic and easily find what you want, when you want it (Patton, 2002). By placing similar material together into its own place, you can review everything that was said or that is known about a topic or an issue without having to sift through each of the original sources (Box 5.1).

- ▶ Sort your data according to any or all of what type of data they are, the period they relate to, the setting they came from. Material might be placed in piles, boxes, or files or stored in computerised coding categories: here are the first set of interviews with community leaders and those are the later ones; there are the notes from each series of planning meetings; these blog entries have been sorted according to the web archive from which they came.
- ▶ Sort your notes and other data also according to particular topics: this is about the level of physical amenity in the area; here are data relating to cooperation (or lack of) between service providers.

Make extra copies (if necessary), so you can put each piece into as many places as are needed. Alternatively, if you use qualitative software, the same part of the data can be coded to as many categories as are needed to fully describe it.

Box 5.1

Coding to manage data

Whenever I have to bring varied sources together and sort out the ideas I am gaining from them, I resort to coding those sources. In such cases, the coding is usually designed just as a way of locating data, rather than as a tool for asking questions of the data (querying), although it has the potential to be developed later for deeper analysis.

Some years ago I was asked to redesign a survey that was a fifth and final data collection point in a 10-year community capacity-building project in an area of economic and social disadvantage. The goal for this final data collection was twofold: to provide a final review to be compared with what had been found before; and to contribute to theoretical development about the association between social capital, community intervention, and health outcomes. In order simply to keep track of the kinds of questions that had been asked, what had been found so far, and old and new theoretical perspectives on each of the issues addressed, I coded questions asked in and reports from earlier surveys, summaries of issues raised in interviews by various stakeholders during the project, and notes from the theoretical literature. This coded material, covering everything currently known for this community on each of the areas being investigated, served to inform the development of the final survey instrument and related theoretical questions.

Coding to build ideas

At a descriptive level, naming a code provides a label that represents what passages of data are about, so they can be located. More importantly, however, naming data *connects* them with other data; data are retained. Coding 'leads you from the data to the idea, and from the idea to all the data pertaining to that idea' (Morse & Richards, 2002: 115).

> If sensing a pattern or 'occurrence' can be called seeing, then the encoding of it can be called seeing as. That is, you first make the observation that something important or notable is occurring, and then you classify or describe it … the seeing as provides us with a link between a new or emergent pattern and any and all patterns that we have observed and considered previously. It also provides a link to any and all patterns that others have observed and considered previously through reading. (Boyatzis, 1998: 4)

As all the data about one idea are brought together through coding they are recontextualised: 'Coding is also a way of *fracturing* data, breaking data up, and disaggregating records. Once coded, the data look different, as they are seen and heard through the category rather than the research event' (Morse & Richards, 2002: 115). Recontextualisation in this way is not about loss of context, but about seeing data in a *new* context (Box 5.2).

Box 5.2

Building an idea (and analysis) through coding

Shane is an experienced historian. I had asked him about his journey into research and he responded by giving an account of an experience of working with primary sources in his second year as an undergraduate.

For some time I struggled with how to code Shane's references to working with old newspapers:

> And I enjoyed that a lot. I enjoyed the sensuous feel of century-old newsprint … Maybe I wouldn't have been so keen on it today, because there is this sensual quality to using aged papers. It's not nearly as much fun with microfilm.

Picking up on his emotional responses was not a problem, but there was greater significance in this: what exactly was it about being involved in working with data that gave rise to the degree of passion that he was expressing? What I was seeing here then 'rang a bell' with something that had caught my attention long ago from Frost and Stablein's (1992) book, *Doing exemplary research.* In their final chapter, these authors summed up what they had learned about the defining characteristics of exemplary research. They called the second of these 'handling your own rat' to express the idea that excellent research comes from being in direct contact with one's data, and doing one's own data analysis. Shane, here, was talking about being in direct, literal, physical and intellectual contact with his data, and about loving that experience. The code I arrived at, then, was *intimate contact with data.* As I reviewed his interview, I could see the same idea expressed again when he was talking about gathering visual material:

> You don't have photographs in newspapers really until the thirties, but it's looking at albums and private collections. And that's lovely work, it's intellectually demanding, looking at pictures.

Identifying and naming this code alerted me to other instances where this could be seen. Stuart provided one:

> but research is I think most rewarding because you're tackling a problem that you yourself have invented. I still draw my own graphs and stuff, like I massage my own data – I like doing all that kind of stuff. I guess it's just thrilling.

The massage metaphor Stuart uses here in describing how he handles his data reinforces the image of 'skin-to-skin' intimacy. And then, from Paul, participating in a discussion with other experienced researchers:

> getting the data and thinking about what it actually means, you know, and being able to get other people's data and think about that too, and then try and shuffle it into some sort of order, so that it makes sense beyond – you know, the bits and pieces, that's what it's about.

From Susan, talking about experimenting with chemical compounds:

> and then we had to try and understand why every time you looked at it, it seemed to be adopting some different shape and because of that, more out of interest, because we couldn't understand it and it just got us sucked in, why on earth is this doing this to us.

(Continued)

(Continued)

And from Beverley in a focus group discussion:

> but I can remember when I got results in from that – you sort of get bogged down in all this psychological theory and you, you, you put all these hypotheses together and you make all these predictions – ah hah! – now are you going to get them? … and there were the differences! Wow! They really exist. The *p* is zero zero zero!

But being close to your data is not always a positive experience! Beverley again:

> other people seem to do experiments and write them up and do the next experiment and write it up and – I don't seem to do that. I seem to sort of be having to go back to my initial dataset and because it was a qualitative one and because I'm an amateur at it, you know ((self-deprecating tone)) so I kept going back and saying 'What am I going to do with this stuff?' you know, and sort of finding new stuff there and thinking this isn't supposed to be my whole PhD, this is only supposed to be the beginning, but – ((deep breath))

Intimacy can be threatening to the fumbling novice!

Now thinking primarily in terms of the category, I widened the net a little. For example, Frank talked with strong emotion about being immersed in research and the consequent stimulation of ideas, although for him it was also very much about sharing that immersion with a colleague. As I continued to work, the code for *intimacy with data* morphed into a code for *immersion*, and as I reviewed other related codes, I found many more passages which spoke of immersion that had not originally been coded as such, each of which enriched my understanding. Now I was thinking about and working with a concept rather than cases, although there was constant reference back to the original data for checking on accuracy of interpretation. (I later came to see immersion as a necessary but not sufficient dimension of obsession – a characteristic that was itself widely considered to be necessary for 'good research' and 'really getting somewhere'.)

Coding to facilitate asking questions of the data

When data are coded, it becomes possible to ask questions of them, beyond simply retrieving all the data identified by a particular code. Asking questions of the data – challenging the data – is perhaps one of the most important skills you can develop. Be ruthless in asking 'So what?' or 'What's going on here?' or 'Why is that?' For example, you might start by simply *comparing* reports of a local event by new and longer-term residents of the area, or you could compare the range of different issues experienced in living in the area that are reported by these two groups. Chapter 9 provides many examples of ways of doing this.

Alternatively, you might consider whether residents who raise an issue of 'hoons' creating traffic noise are also the people who complain about the dirty state of the local shopping precinct. And did they say whether one or other or both of these problems have always been there or are they both recent? Or, do those who enjoy the country-like atmosphere of the area in which they live also enjoy entertaining at home? What are the styles of entertaining, or social life

more generally, that are enjoyed by residents of different areas, and to what extent is this related to their attitude to the area? These kind of *relational* data questions are addressed in Chapter 10. Coding data is a facilitator for answering all these kinds of questions.

Writing analytic memos and keeping an audit trail

Throughout the process of coding, building memos about concepts is critical. Note why *this* code is important to consider, any regular or especially notable associations it appears to have with other codes, questions that arise in relation to the concept it represents, ideas that come from theoretical or fanciful comparisons, and so on. Memos are especially important for higher-level concepts; these will contribute significantly to the eventual writing up of the project. By recording also the circumstances under which a concept comes up for discussion, and links between this concept and another, you will contribute to later theory building and reporting your analysis.

In addition to memos relating to specific codes, keep notes in a journal or research diary about decisions made in relation to coding (cf. Figure 4.1). These will be important later in helping to explain and justify how you arrived at the conclusions you have reached.

Memos about concepts can be recorded in a notebook, on a sheet in an indexed folder, using a word processor, or using qualitative software (Figure 5.1). Further examples of ways to record memos were illustrated in Chapter 4.

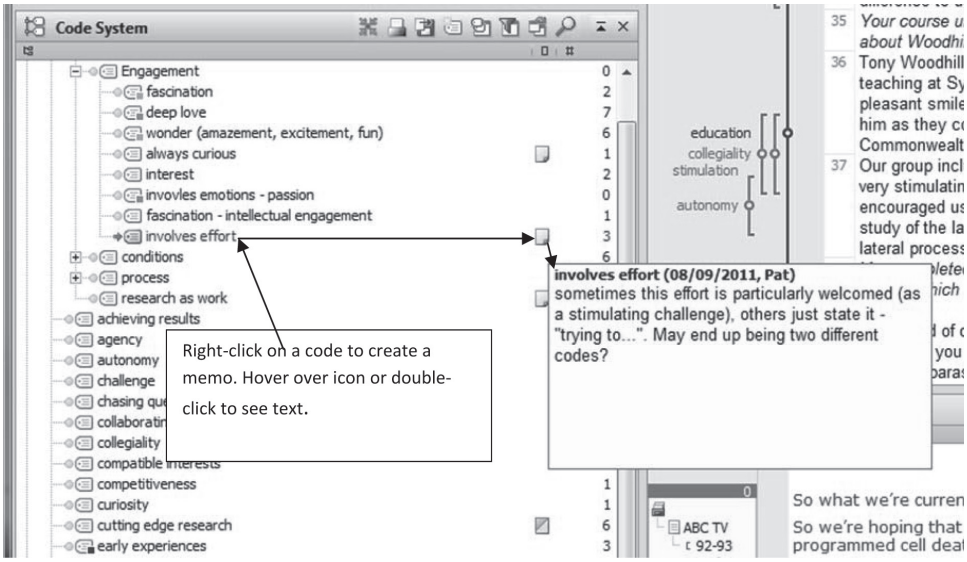

Figure 5.1 Linking a memo to a code in MAXQDA

Tips for coders:

- Use codes to help keep track of data as well as to cover their content, and ask questions relating to the content.
- Record the decisions you make about coding.
- Record ideas you have while you are coding about analysis and questions to ask.

Practical tools for coding

The strategies described in Chapter 4 were designed primarily to help you become familiar with each data source as it became available to you. Now it is time to work systematically through each data source, to build a database of evidence that will provide you with insights into the experiences or processes you want to understand, and serve as a resource for testing and supporting the conclusions you will develop.

There are multiple strategies that researchers have developed for coding data. In this section I review a range of possibilities for both manual and computer systems.[1] If this is your first attempt to systematically code data:

- Experiment to start with, until you find a strategy you are comfortable with.
- Start small: avoid stepping into a major project with a strategy that you haven't tried before.
- Try running some simple analyses based on your first sample of coding, to ensure it will work for your purposes.
- Spend some time working through your first few sources with someone who is more experienced; become an apprentice to develop competency in the craft of coding.

'Pencil-and-paper' strategies for coding

Historically researchers have had to rely on a range of pencil-and-paper tools, and these have served to facilitate many fine ethnographies, case studies, grounded theories, and other analyses. For some researchers, the tactile contact with paper is important, as is their ability to sit with data surrounding them – giving them a sense of physically handling and juggling the words or pictures as they tease out ideas contained in them, and a sense of understanding the whole picture. Alternative methods for working 'manually' follow.

Indexing

Indexing involves creating labels for identifiable passages of text, and making a record based on the labels. (Note that multiple labels can be applied to any passage.) To use this method:

[1]Novice researchers often (mistakenly) imagine that if they have a small amount of data, it is not worth using a computer for coding and analysis. The smallest project for which I used a computer consisted of nine pages only of data, but using it was invaluable for helping me to organise the concepts contained in those nine pages.

- Sequentially number the lines, paragraphs, pages, or other units in your sources.
- Record your code, accompanied by the appropriate sequential number to identify both the source and the specific location within it. If you find more passages on the same topic, add to your previous record, just as an index for a book is built. If you don't have a predetermined coding system to work from, then you will probably find this easiest to do:
  - by creating a card for each code, on which you record reference details for each passage you find for that code, along with a brief reminder of the content of each passage (Turner, 1981);
  - as a list into which you can insert new codes, using your word processor or a spreadsheet.
- Record also associations between codes as you go (e.g., the other codes you are using for the same passages).

Marking transcripts or other physical records

Pencil and paper coders format their data so that they have wide margins in which to record codes. Use margins to:

- Write in as many keywords as are necessary to describe the adjacent passage or text or element of the picture; and/or
- Read through sufficient data to be able to decide at least the kinds of codes you might want to use, and develop a shorthand code system from those; then use the margin space to write in the codes you are using (this does not allow for much flexibility).[2]

✓ Always keep a clean copy of the original data, as a backup resource.

- Keep a journal in which you record notes about the content of each code, and of the associations you are noting between it and other codes.

Sorting to create codes

For the first sorting strategy you will need to make multiple copies of each source (ensuring that an intact original is kept on file). Clearly label each copy of each page with essential identifying information:

- Cut out passages from *copies of* the original that are about particular topics or issues or points of relevance with regard to the goals of your project. *Make sure each piece is labelled with its source ID, page, and paragraph number as you copy or cut it from the page.* Make as many copies as you need for the number of categories that apply to that segment.
- Place these in separate piles or hang folders, to represent the various categories.

A second sorting approach uses either short segments of text, or statements drawn from the text that are exemplar quotes regarding ideas present throughout the text. Paste these exemplars onto small cards so that the identifying and

[2]Examples of manual coding structures can be found in Miles and Huberman (1994: 59–60) and Patton (2002: 464, 516–17).

descriptive information about the source or the context of the quote can be recorded on the back of the card (Bernard & Ryan, 2010).

- Sort the cards into groups, based on similarity of content. This might be done by spreading them across a table and gradually pushing them into piles, then naming the piles.
- Alternatively, have pairs of researchers working on the sorting task, with their conversations being recorded as they do so.

The discussions coders have about the piles or the pairs of quotes are as valuable to the researcher as the final sorts are, in providing an understanding of the issues covered by the quotes (Box 5.3).

Box 5.3

Working as a team to create and sort 'chunks of data'

Here is how Scott, Bergeman, Verney, Longenbaker, Markey, and Bisconti described their sorting process:

> Working with another member of the research team, we separated the transcribed interviews into units of meaning. Constantly comparing these units with each other, these chunks of data were grouped into emergent categories. A rule for inclusion and exclusion was written for each category, describing the essence of the units in that category. Eventually in this iterative process, the rules were developed into propositional statements that served as tentative findings for a specific theme. The research team then synthesized the propositional statements into an overall understanding of the phenomena. To build rigor into the work, the research team consulted with peer debriefers. (2007: 248–9)

Problems in using paper methods for coding

When you are coding, it is ideal to have both the benefit of being able to see in one place all the data pertaining to a particular code, and also to be able to see the data for which a particular combination of codes applies. A difficulty in deciding about methods of coding, if you are limited to manual (pencil-and-paper or cut-and-paste) approaches, is that while each method has particular benefits, it loses some or all of the benefit that pertains to the other. Thus:

- Indexing or marking techniques, provided an accompanying tally sheet and journal record is maintained, give the best access to patterns of association in the data, but make it more difficult to assemble original data associated with either individual codes or combinations of codes.
- Sorting techniques give the most rapid access to all the evidence gathered relating to a particular code, but are less helpful when it comes to looking at the connection of codes.

Either way, full benefit will depend closely on the care with which you record what you are noticing and thinking as you code. Otherwise, reports will be

descriptive (a list of annotated concepts or 'themes') without analyses of associations between themes, or analytical without descriptive depth to provide context and interpretive understanding for the reader.

Using a computer for coding

Software tools to support analysis have now become readily available and are increasingly being adopted by researchers analysing qualitative data. Essentially, using computer software, especially those programs developed specifically for qualitative data analysis, will give you the benefits of *both* the indexing/marking and sorting approaches outlined above, *and more*, with much greater flexibility and speed.

Using standard MS Office software

Everyday MS Office software can assist with sorting and indexing text, albeit *in a limited way*.

Sorting text using Word

At a very simple level, one might *copy* and paste passages of text from an original source into another document where the material is organised by topic (or, less conveniently, into separate documents for each topic). Some text might be pasted multiple times, if multiple codes apply to it.

- Use styled headings to identify the topics for the sorted text. Doing so allows you to (a) use the Document Map (Navigation pane) to hyperlink to any heading in the document, for rapid access; and (b) to create a table of contents for the topics in the document (with page numbers).
- Record the source ID and paragraph numbers for the original version each time you paste a passage of text.
- If you are pasting the text into multiple places, include (with the ID information) a comment with each paste to indicate where else it has been placed.

An alternative approach is to insert keywords into the text, so that relevant passages for any topic can be found using Word's Find function. While these methods can help you to locate all your material on a particular topic, they are not helpful for finding patterns of relationships between categories.

Indexing with Excel

Indexing can be recorded in Excel (or alternative spreadsheet) rather than on paper or cards. This will facilitate pattern searching within and across sources, as well as retrieval (Figure 5.2 shows coding for the passage shown in Figure 5.3).

- Ensure your sources have numbered paragraphs.
- Each time you identify something in the source to be coded, use a new row in the spreadsheet to record (in column A) the source ID combined with the *single* paragraph number for the material being coded.

- You will benefit from always using the same number of digits for numbers; so, for example, if your source has 597 paragraphs, start numbering with 001 (this impacts on sorting). At the same time, don't start an ID with 0, as Excel will drop leading 0s.
- In column B for that row, record the code. If you want to code it with three codes, use three rows.
- If the code applies to more than one paragraph, repeat the code in a new row for each one.
- Use a third column to record a very brief summary of the content for each code for each unit of data, or to paste in a brief quote from the original text. Format the column so the text wraps in the cells if you are using more than a few words.
- You can attach a brief memo to a cell in Excel. Cells with a memo are marked with a triangle in the top right corner, and the memo can be seen by hovering over the cell with your mouse pointer.

For retrieval:

- Use the software's sort function (cf. Chapter 3) on column B to find all the paragraphs in which material for a particular code appears. Sort on column A to find all the codes that co-occur within each paragraph of data (some of these may be incidental and have no significance).

	A	B	C	D	E	F
1	**Source ID**	**Code**	**Text**			
2	A2	career	In and out of research "I drifted"			
3	A2	educ	hons PhD postdoc			
4	A2	opp	offered postdoc			
5	A3	career	not planned			
6	A3	uncertainty	about research			
7	A5	goals	prefers teaching			
8						

Pat:
reactive rather than proactive

Figure 5.2 Recording indexing in Excel, to allow sorting by paragraph or by code

Using qualitative software

When you code using qualitative software, you create within the software a storage area for each topic or concept, in which references to large or small segments of text, image, sound, or video are recorded.

Most software has a similar approach to the coding process:

- Import the sources to be coded (this usually *copies* the source into the project file).
- Create coding categories either before or during coding. These can be changed and reorganised at any time without loss of data.

- Use drag-and-drop to code highlighted text in the imported source at a particular code (Figure 5.3).

- Sources can be edited after import, but it is better to set them up appropriately beforehand.
- Sources can be of different types, including text, video, audio, and images, and in various formats.[3]
- Most software offers ways of automating routine coding (such as for who is speaking in a focus group, or what question is being answered in a questionnaire). These documents usually require special formatting *before* they are imported.

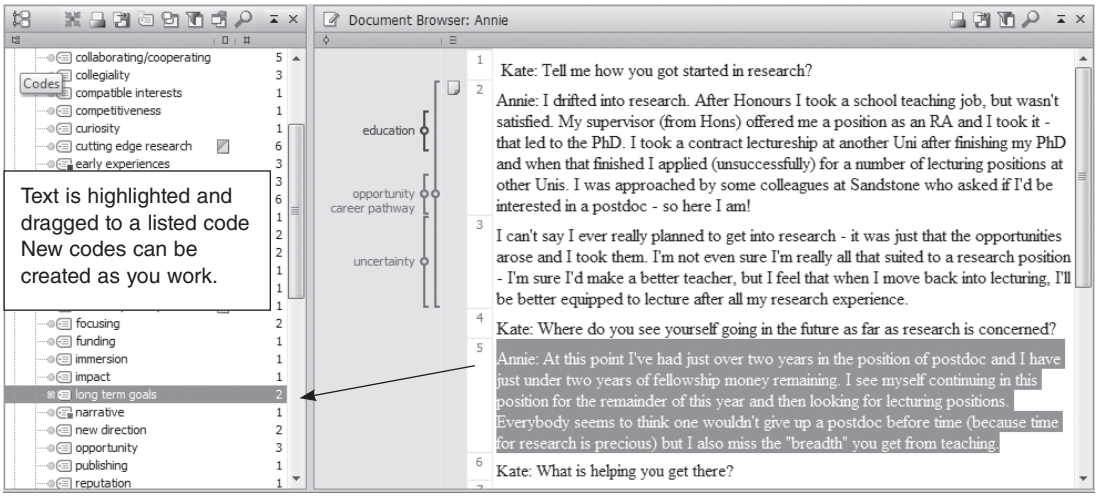

Figure 5.3 Coding using MAXQDA

What is stored, when you code, are not actual segments of data, but *references* to the exact location of the coded data in the source file. Using those references, the software is able to locate and retrieve all the coded passages for a particular code or combination of codes from the source records (Figure 5.4). The passages themselves are never copied, cut, or physically moved as they are coded. Unlike cut-up photocopies on the sitting room floor or in hanging files:

- the source always remains intact;
- information about the source of a quote is always preserved and is displayed whenever a segment is displayed;

[3]Exactly what can be imported, in what format, depends on the software being used.

- changes made to the source are (usually) immediately reflected in the coded text segment;[4] and

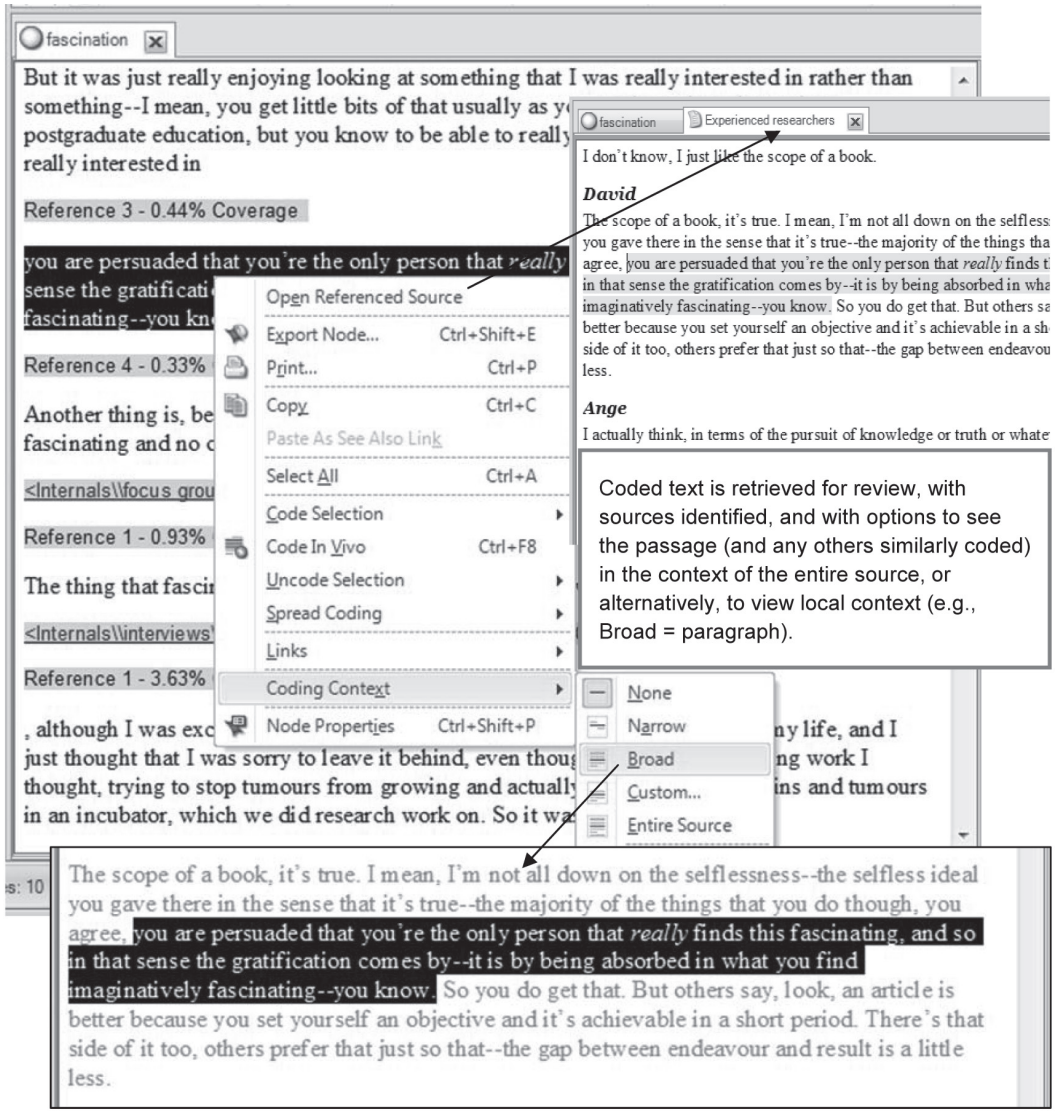

Figure 5.4 Code and context retrieval using NVivo

[4]Qualitative software programs vary in the degree of flexibility and integrity they have with regard to editing of texts already imported, particularly once they have been coded. Some are better than others at adjusting the coding that has already been applied to take account of the editing changes. Check this feature carefully before making changes to text that has been coded in the software you are using.

- a coded segment can immediately be seen together with other similarly coded material (Figure 5.4); but also
- it is always possible to view the coded passage in its original context; and
- passages can (and should!) be coded with multiple codes, with queries rapidly able to find passages coded by co-occurring codes.

Computer-based coding of qualitative data is potentially more complex and more detailed than manual thematic sorting. Rather than identifying a passage with one all-inclusive thematic code, when using a computer the coder can separately identify, by using several codes on the same passage, what is going on, contextual factors such as when, where, and who is involved, what attitudes, feelings, or responses are being revealed, what issues are raised, what the outcome is, and so on, with the confidence that the computer can reconnect codes used for the same passage of text. Attending to these different aspects as you work through the data encourages a greater level of attention to nuances in the data and often leads to deeper insights than would otherwise occur. Immediately data are coded in this way, almost limitless possibilities for review, sorting, and sifting of text segments become possible (Bazeley, 2007; Bazeley & Jackson, in press; Lewins & Silver, in press).

Problems in using computer software

There are dangers to watch for when using a computer for coding. There is a tendency for users, especially novice users, to:

- create and record too many data in too much detail, simply because the computer can handle them more efficiently than can be done using paper methods;
- rely on coding and retrieving descriptive codes or simple themes, without taking advantage of the linking, memoing, and analysis tools provided by their software;
- become obsessed with the *task* of coding (often referred to as 'the coding trap'), to the exclusion of imaginative and reflective thinking, linking, and memo writing (Johnston, 2006).

The point, as always with computers, is to make the computer work for you; don't work for it!

As with any new software, there is a learning curve when you first start using qualitative software. Basic tasks like coding and code retrieval are usually quite straightforward to learn, but some form of guidance or training is likely to be needed to really benefit from the more sophisticated management and analysis tools provided by software. Developers offer a range of training tools (video tutorials, extensive online help), workshops, webinars, and web-based Q&A forums to assist the process.

Creating a codebook

A codebook holds a digest of your codes and categories, something that assists the lone coder in being consistent in the application of codes. A codebook is essential for team coding, to ensure all team members are applying codes in a

similar way. It will probably be useful to have it in two forms: a full reference version with detailed descriptions of each code and where and how each has been used, and a briefer summary containing the labels you are using for each code and a very brief definition, useful to have nearby as you are coding.

Expect the codebook to be a work in progress, so you will need to reprint it multiple times.[5] This applies even if you start with a predetermined list of relevant codes; it can be surprising how often something that *should* have been relevant turns out to not be so, while other codes need to be elaborated or added.

Codebooks for working manually

Because you don't have rapid access to review material already coded, your codebook will need to contain considerable detail to make it useful as a reference for your coding (MacQueen, McLellan, Kay and Milstein, 1998). Useful information to record includes:

- a short label suitable to apply in a margin;
- both brief and fuller definitions;
- inclusion and exclusion criteria; and
- typical examples (Table 5.1).

Table 5.1 Example of a codebook entry

Code	**STIM – Intellectual stimulation**
Definition	Any evidence of where thinking processes have been 'turned on' by research or research related activity in a way that serves to motivate them to do more.
When to use	Use specifically where something has the effect of stimulating further creative/research thinking or activity. Often overlaps with other intellectual process codes (use more than one where appropriate).
When not to use	e.g., where simply talking about the challenge of a task associated with research, such as admin.
Example	'You stimulate each other, … you are talking research, when you go to bed at night you are thinking it.' 'Getting the data…wanting to know.'

Codebooks for computer software

If you are using a word processor or spreadsheet to record coding, then you will benefit from creating a codebook in a similar way to that for manual-based methods.

[5]The worst situation I ever encountered, in this regard, was the government employee who, after modifying her codes a couple of times, decided that she could now laminate the list, never to change it again. Unsurprisingly, she rapidly became totally bored with the whole task of coding, and never completed the project.

For those working on specialised qualitative analysis software, there are ways of recording the same type of code information in the software, such that it can be easily retrieved as needed, or printed off for reference, as a list of codes or codes with definitions.

- In MAXQDA, definitions, additional commentary, and reflective thoughts for particular codes can be recorded in memos attached to those codes. The code system can be exported or printed with or without the additional memo information.
- In NVivo, definitions for particular codes can be recorded in the properties dialogue box for those codes. The screen display can be customised to show these definitions whenever codes are listed. Additional commentary and other reflective thoughts about a code are recorded in a memo attached to that code. The list of codes with definitions can be exported or printed, as can the text of individual memos.

Including data about participants or other sources

This is a largely mechanical task (for everyone, not just those using computers) that can be dealt with whenever convenient in your project; perhaps you have already done it as part of preparing your data and storing your data records.

For each of your sources and/or units of analysis (if these don't equate to sources), you will almost always want to code or otherwise record some useful classifying information. This will allow you to identify the context of a particular coded passage, for example, whether it was said by a Londoner or a Parisian, when you are thinking about or reporting that passage. More importantly for analysis, it will allow you to make comparisons by sorting all the data with a particular code according to where they have come from, for example, to compare all the comments on cooperation by females with all those from males. If you are using software, you might also be able to use this kind of classification to compare whole patterns of responses, such as the differences in the range of emotions expressed by beginning researchers compared with those who are more experienced. Such comparisons will contribute significantly to your analysis (cf. Chapter 9).

The two issues for now are: what kind of demographic or variable data will be needed; and how best to record the information so that it can be used for those later comparisons.

Demographic information is the most obvious kind of data to record: things like gender, location, role, age. If you are working in a project where survey data are available for the same participants, you might also want to record responses they made to categorical questions ('Did you experience discrimination often, sometimes, or never?') or perhaps even scaled information ('On a 10-point scale, what was your level of pain today?') to consider along with the qualitative data. What you record will be governed by two factors: what information is relevant to the issues your project is addressing; and your capacity to use the data for analysis – the latter being governed largely by whether you are working manually or using a computer.

Those working with paper records will need to ensure, if they cut them up, that they have a labelling system for the sorted segments that allows them to retrieve relevant demographic data. For those who are recording text data into columns or fields in a spreadsheet or database, then additional columns or fields can be set up within those programs to record associated data. Or, if you are using qualitative software, most current packages provide for recording this kind of classificatory information (typically referred to as 'attributes') so that it is linked to the sources and/or cases, but stored separately from the coding system.[6] This is because such information is relevant to the whole source or case, whereas coding is usually applied to selected segments of data only.

Sometimes you are unaware of just what variables to apply until you are working through the data source, or you might decide during the coding process that you now wish to classify people on another variable or attribute, such as a dominant attitude, or the presence or absence of a particular experience in their lives. This is usually possible, but the thing to remember, if you are doing so, is that the classification you use has to 'fit' *all* of the data for that source or case – otherwise it is better to use coding, applied to the relevant passages only.

Methods decisions in coding

Some of you will have generated some codes already, based on the suggestions in the previous chapter. Maybe you have also explored some techniques for coding. Those experiences provide a good background against which to make some decisions about the way you are going to approach coding for your current project.

Keeping your purpose in focus

The first and most important thing to consider is your purpose in coding. Before you start, remind yourself of the aims and objectives for your project and the questions your research is designed to answer, as these will critically influence what you will look for in coding your data. Then, as your list of codes grows, periodically review them in the light of your questions. Will they give you access to the kind of data that will allow you to answer them?

---

[6]Many of these software programs allow, also, for this kind of data to be imported directly from a spreadsheet.

Tips for coders:

- Paste a copy of the questions for your project on the wall near your workspace.
- Challenge each code you create. Ask: why am I interested in that?[7]
- Have a trial run to see if you can answer your questions from the codes you have created. Are there any gaps that need filling?
- 'Park' data and ideas that are interesting but not immediately relevant (use a single code). Ignore or remove any that are neither interesting nor relevant.

Breadth versus detail in coding

Will you start generating codes by marking or sorting data into broad categories or by working line-by-line or even phrase-by-phrase? Bernard and Ryan (2010) evocatively characterised different approaches to the task of coding as lumping and splitting, with 'splitters' maximising differences between text passages as they look for fine-grained themes or categories, while 'lumpers' ignore the finer nuances in the text as they look for overarching themes or areas of discussion.

In practice, most people will engage in a combination of both splitting and lumping, if not at the start, then certainly during the course of their project. For example, you might:

- Use broad codes indicating areas to revisit if it is too early to comprehend detail in those areas.
- Focus on the detail of a few rich paragraphs if you are feeling lost in the whole. Qualitative analysis is designed to be flexible, and most people revisit their early work in any case.

The important thing is to start as soon as you have data. Don't delay because you're feeling uncertain (you can make changes later), and never allow documents to build up unprocessed.

'Chunking', fracturing, or slicing text

Unless you are simply indexing original sources, coding inevitably involves segmenting text or other source material, in so far as the code is applied to sections of the text only. Segmenting may occur as a preliminary step to coding and analysis, or during the process of coding, depending on the method you are using to apply codes. Some codes will be applied to longer 'chunks', but most will be used with shorter passages. Breaking the text into short paragraphs or even smaller

[7]For those using a grounded theory methodology, this question should not be asked too early (e.g., in the first two or three sources), assuming that the data are more or less relevant to the topic of investigation. This is so that you do not pre-empt possible new ways of thinking about the issues raised.

passages can be helpful in clarifying content and meaning in the text, especially in dealing with long, possibly poorly structured turns of conversation from a talkative participant. Each of these passages can then be coded with one or more relevant codes.

Coding is often seen as 'fracturing' the text (e.g., Strauss, 1987), with attention then being paid to each separate fragment.[8] This metaphor, with its strong connotation of breaking something into small pieces, suggests a consequent loss of connection between the fragments of data and hence connections to the immediate and larger context of the surrounding text. This has led to criticism of the practice of microanalysis (Hodgkinson, 2008). Fracturing of data into very small segments can be a problem with manual methods where copies of sources are cut up into pieces for sorting into piles. It was seen as a problem also in the early days of qualitative computing when, on retrieval, the coded segment was visually separated from the surrounding text. It is less of a problem with current software, in that the immediate and larger contexts of any coded segment are each available with a click of the mouse (cf. Figure 5.4), although the initial retrieval may still be of the decontextualised segment.

Apply all relevant codes to the whole 'meaning unit' (a sentence, a paragraph, several paragraphs), even if each doesn't apply to the whole passage. Codes will overlap, while the specific text that gave rise to the code is still readily apparent within the passage. Segmenting text, then, might be thought about more as slicing or layering rather than fracturing, and 'disconnect' issues with fracturing can largely be overcome. In slicing the text, multiple codes are used to capture the who, what, when, where, and how of what is being reported in a passage of text (Figure 5.5).

Using multiple overlapping codes on text has the advantage that patterns of connection between codes can be readily explored. Codes applied to the same or overlapping passages of text can be presumed to have some relationship; codes applied to adjacent passages of text may or may not have any association. As you code your way through data sources, patterns of association between codes will become apparent. To facilitate later analyses, it is important to note these associations in memos as you become aware of them, or if you are working on paper, then you need also to note the document, page, and line numbers where the associations were seen, so that they can be located again when needed for further analysis or as evidence for the pattern.

[8]It is important not to confuse coding detail (identifying specific subcategories of content in the text) with coding detailed (small) segments of data. They have different meanings, so while the two often go together, they do not necessarily do so.

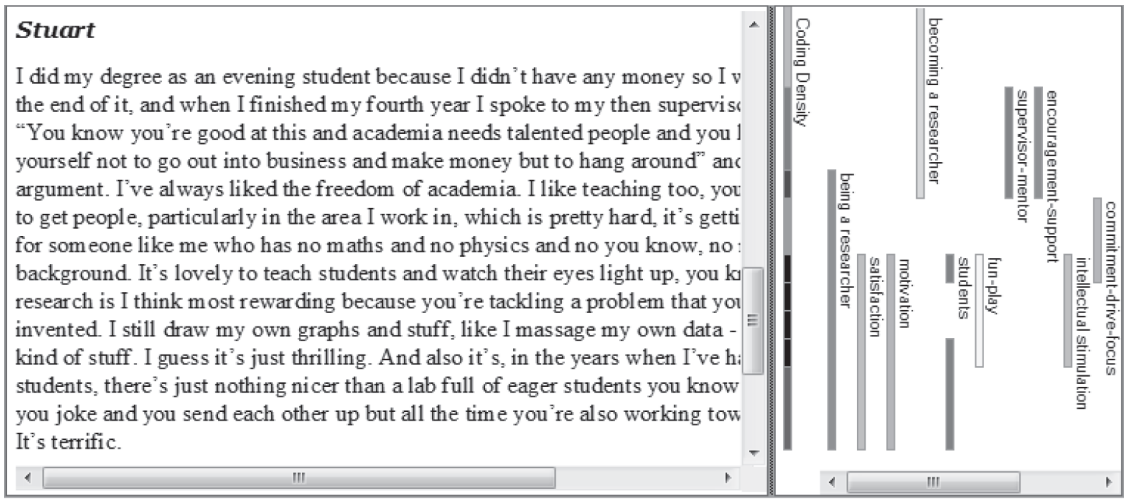

Figure 5.5 Coding stripes showing multiple overlapping codes applied to adjacent text in NVivo

Tips for coders:

- Apply codes to *meaning units* within the source. These might vary in size, depending on your current purpose, as well as the nature of the material being coded.
- Apply as many codes to a meaning unit as is necessary to capture all the dimensions of what is going on in that passage.

Coding different types of data

Qualitative data take many forms, with implications for coding practices. Data vary in their degree of structure; how openly ideas, experiences, and attitudes are recounted or discussed; whether they are descriptive or emotive; the style of language used; and so on.

Interview transcripts are typically complex in terms of the number of cross-cutting contexts, ideas, and concepts present in any one passage, as people express and respond to whatever thoughts are prompted by an interviewer's questions or comments. Consequently, coding of these is multifaceted in order to capture descriptive context and experience as well as interpreted rationales and responses, and memo writing associated with coding is likely to be extensive, focusing on both cases and concepts.

In contrast to coding an interview where you are focused directly on what is said by the interviewee in response to your questions, to effectively code *focus group data* you will pay more attention to the interaction within the group. View any individual's comments in the light of that interaction.

Field notes differ from interview transcripts in having been written, in most cases, by the person who is coding them. Like memos, field notes are more likely to be written around particular incidents or observations or topics, making coding more straightforward. Coding of field notes is likely to be more descriptive than coding of interview data, particularly if the observer's reflections are stored separately from his or her observations. Where events being observed have become familiar to the observer/coder, Charmaz (2006) suggests using an approach which compares one incident with another, rather than routinely working through the detail of each incident, as a way of 'breaking through' the ordinariness of each one.

Short responses to open-ended questions lack elaboration, particularly those provided in response to a self-completed questionnaire, and so they can leave open as many questions as they answer (some would question that these should be considered as qualitative data). Often these are suitable for semi-automated coding procedures, such as word searches, or they can be readily categorised – if they are coded at all. For interactive coding of short-answer responses, depending on the nature of the questions and to what extent the responses to different questions are interrelated, it can be more helpful (and efficient) to focus in turn on the issues for the study by coding all responses to each question, question-by-question, rather than the more usual procedure of working through all questions for each case, case-by-case. (This depends on how necessary it is to see each case as a whole.) Memo writing, in this type of project, is likely to be associated with important codes or with the project as a whole, rather than with individual cases.

Streaming data (audio, video) pose a mechanical difficulty for coding of content. By the time you realise you want to code a specific passage in a particular way, you have missed the marker (usually a time stamp) for the beginning of the passage. You will therefore need to be very familiar with the content of the source so that you know what is coming, have detailed notes to assist, or be prepared to do a lot of backtracking as you work through, all of which make the task very laborious and slow. Retrieval is also much more time consuming than for written text. If the reason for working with video is to capture non-verbal cues, then a parallel transcript or summary of the spoken words would be very helpful.

Epistemological frames, theoretical perspectives, and coding

Epistemological frameworks for understanding the social world influence the questions being asked, and consequently the ideas about what to code and how to code (or whether to code at all). Some will see and code only those things that have been directly observed and reported; others will seek to understand and code the speaker's interpretation or construction of their experience; and others

again, their own interpretation or construction of what was said and done by the participant.

Different theoretical approaches to understanding the social world, similarly, lead to different questions being asked, and different concerns in coding. Those whose focus is on larger issues such as the causes, expressions, and impacts of power and domination throughout society will identify different features in their data from others who attend to the structure of a conversation. Cooper's (2008) description of how different social theorists would attend to quite different features in studying a door (Box 2.2) suggests that each would code using different labels, and would read different implications from those codes. These orientations influence decisions about coding right from the start of the process, even for something as apparently straightforward as 'clerical' and descriptive coding: do we pay attention to a person's gender? Their sexuality? Whether this event occurred before or after a change was introduced into office procedures? The presence or proximity and size of the photograph of the child?

Attending to the detail of our texts or other sources will facilitate openness to what those texts are saying, but reflexive awareness of the ways in which prior frameworks and pre-existing knowledge might influence interpretation and understanding is called for and their impact should be noted in memos as coding proceeds.

Methodological purposes and coding

Because each methodological approach seeks to meet different purposes and has different emphases, what is coded and how the coding is applied will vary. Is the emphasis on process, or structure, or emotion? Does the coder work holistically or focus on detail? Are the initial codes they create descriptive or interpretive? Examples of how different methodological approaches might impact on the coding of one interview are illustrated in Box 5.4.

Box 5.4

Different methodological emphases in coding a sample of text

Different methodological approaches to text can be illustrated from Shane's account of how he came to be interested in research.

- The grounded theorist, with an emphasis on *Shane's perspective* on the *process* of his becoming interested, would note his desire and capacity to pinpoint when this occurred; the circumstances, process, and consequences of his being encouraged and praised by others and the important role that encouragement played; the pleasure he gained from his engagement with the task; and

(Continued)

(Continued)

at a later stage of interpretation, perhaps the match between the tasks he undertook and his personality.

- An interpretive phenomenologist, concerned to identify the essential elements of the *experience* of becoming and being a researcher, might focus on Shane's passion, the pleasure of working in intimate contact with data; the satisfaction that comes from carefully completing a task and creating a product that can be held and admired; and the warmth of receiving interest and praise from others.
- Someone interested in discourse analysis would focus on the *language* of passion and romance in Shane's telling of his experience, and what that conveys about the meaning of that experience for him. It is a language which contrasts strongly with the language of performance, of duty, and of play that is found in other transcripts.

Issues of validity and reliability in coding

How you perceive, interpret, and code your data will be impacted intellectually and practically by:

- the perspectives learned through your disciplinary training;
- the focus of your study, the questions you are asking, and related or consequent choices regarding epistemology and methodology;
- the extent to which you have already explored these questions in previous studies or in the theoretical and research literature;
- your own level of involvement in data collection and preparation, and consequent familiarity with their contents;
- your mood at the time, the degree of pressure you are under to finish, and other personal factors.

How credible is my interpretation?

All of our observations, of numbers and visual images as well as text, involve interpretation – by us, *and* by the participants who provided the data. They represent one view among many. By making your perspective and your questions explicit, you can increase the consistency of your coding, and assist the reader to understand how and why you interpreted the data in a particular way. The issue is not whether you have discovered or developed a 'true' understanding of the data, but whether, in the eyes of the coder and those evaluating the coding, the interpretation makes sense *given the conceptual framework of the coder* (Kvale, 1996; see Box 5.5).

Box 5.5

Hamlet's interview

Kvale (1996: 151–3) uses Hamlet's interview of Polonius to illustrate how interview technique, the responses given, and interpretation of those responses are influenced by the purposes of both interviewer and interviewee. The interview is very short, comprising just three questions:

Hamlet: Do you see yonder cloud that's almost in shape of a camel?
Polonius: By th' mass, and 'tis like a camel indeed.
Hamlet: Methinks it is like a weasel.
Polonius: It is back'd like a weasel.
Hamlet: Or like a whale?
Polonius: Very like a whale.
Hamlet: … (Aside) They fool me to the top of my bent.
(*Hamlet*, Act III, Scene 2)

Kvale observes that this appears, at first, to be use of an unreliable interview technique, with its three leading questions leading to entirely different answers that tell us nothing of the shape of the cloud. But is the cloud the issue, or is it rather the personality of Polonius, and his trustworthiness? Now, Hamlet's sophisticated indirect interview technique provides 'reliable, thrice-checked knowledge about Polonius as an unreliable person' (1996: 151–2). This is confirmed in Hamlet's aside at the end of the interview.

But then, if the interview is considered in the context of a royal court where a prince is questioning a courtier, it demonstrates the potency of power relations. Polonius was well versed in indirect interview techniques (drawing on evidence from an earlier scene), so perhaps he is simply playing up to Hamlet as a courtier?

Finally, Kvale notes, the central theme of the play is 'a preoccupation with the frail nature of reality'; thus his interview may be seen as reflecting 'pervasive doubt about the appearance of the world, including the shape of a cloud and the personalities of fellow players' (1996: 153).

What if I get it wrong?

In discussing open coding and the anxiety that novices have about identifying the intended meaning of an interviewee's words, Strauss observed:

> The aim of the coding is to *open up* the inquiry. Every interpretation at this point is tentative. In a genuine sense, the analyst is not primarily concerned with this particular document, but for what it can do to further the next steps of the inquiry. Whatever is wrong in interpreting those lines and words will eventually be cancelled out through later steps of the inquiry. Concepts will then work or not work, distinctions will be useful or not useful – or modified, and so forth. (1987: 29)

Coding involves regular review and revision of concepts. If you are unsure about whether to apply a particular code you might review what text you have already

coded there, to check if it matches. When you start to organise, group, or synthesise codes, you will inevitably review and revise what you have coded. These are opportunities to 'get it right' and to recode and/or remove irrelevant material. Finding missing gems is a little harder. I sometimes locate these also during the process of reviewing a code, when I check what else a particular passage is coded at and discover via the coding stripes that a concept present in the text is missing from that list (this would be difficult to do manually). I have also found missing codes during analysis, particularly when looking for exceptions to an association. For example, in my original coding of Stuart's document in the Researchers project, I had missed coding his passage where he talks about playing with graphs and massaging data as being about intellectual stimulation, as it should have been. This became evident when I was looking to see whether one could experience enjoyment or satisfaction in doing research without talking at all about being intellectually stimulated by it, and this passage came up. I added the additional code to the passage, and dismissed it as a valid result for this query – problem quickly solved!

Multiple coders and coding reliability

There is often an expectation in qualitative circles that coding should be checked for reliability. Coding for reliability usually means that a second person is asked to code a sample of material, to see if they arrive at the same codes for each passage as did the first person, or a sample might be recoded at a later stage to see if the coder has remained consistent in applying codes throughout the project. These kinds of checks are often seen also as contributing to the validity of the conclusions drawn from the codes. These practices derive from coding in surveys and other quantitative work designed for measurement or statistical analysis, where codes become separated from data and so clear definition and consistent application of a code is critical to being able to interpret statistical results.

In qualitative work, however, 'there is no single set of categories waiting to be discovered. There are as many ways of "seeing" the data as one can invent' (Dey, 1993: 110–11). Most samples of qualitative data have multiple stories to tell, and each person coming to the data brings with them their own purposes, perspectives, experiences, and knowledge. As indicated earlier, each of these influences what is seen in the data, their interpretation of what is seen, and hence what categories are developed from the data.

It is *not* reasonable to expect that two people coming to a sample of data, only for the purpose of coding, will code it using the same categories and in the same way, unless the second person is drilled in the purpose and framework for the project, and given a tightly defined set of codes to work with, with strict instructions and training in how to apply them. (I have been known to refer to this as the 'trained monkey' approach to coding reliability.) It *is* reasonable to expect some

consistency in coding done by the same person across a whole project, and that any early vagueness in codes has been clarified and 'tidied up' by the time final analyses are undertaken.[9] Additionally, the categories developed should make sense (in the context of the project's purpose, etc.) to a second observer and potentially to those who provided the data, and it should be evident that the data have been appropriately fitted to the categories (Guba, 1978, reported in Patton, 2002).

What is more likely to convince an audience of the reliability and validity of your conclusions than an artificially created measure of reliability is the strength of your argument and the clarity and comprehensiveness of your evidence. What was it that convinced you of the conclusion you have reached? To take your audience through that process or argument as a means of convincing them, you will need to have recorded detailed memos of the process you went through, and you will need to be able to readily locate relevant supporting evidence from your data (more on this in later chapters). The implication, for now, is that reliability testing for coding does not, in itself, add to the strength of your conclusions. Rather, you need to keep a clear record (audit trail) of your coding decisions, linked to a strong evidentiary database (Yin, 2003).

Determining inter- and intra-coder agreement on coding *is* relevant in the context of a team and/or longitudinal project, where consistency between coders is clearly necessary. The real value of the exercise in producing a measure, though, is not the measure itself, but what can be learned in the process. Having additional people code the same document, especially while the codes are being developed, and then comparing that coding promotes some very worthwhile discussions about how the coding is being approached, based on the differences observed, leading to agreement about what is important (hopefully!) and clear definitions for categories. For a team project, this needs to occur relatively early in the coding process, and again at regular intervals throughout, to keep all coders 'on the same page'. For your individual project, comparing your coding on a sample of data with that done by someone else (with some guidance as to the purpose and orientation of the project) similarly can lead to valuable discussions (as with brainstorming more generally) about how the coding is proceeding; it's just not particularly useful in terms of 'reliability'.

For those working on computer, several qualitative software packages now support checking for inter-coder agreement by, first of all, tracking the contribution of each user to the project, including application of codes, and then showing comparison data where two have coded the same text. Figure 5.6 shows the output from NVivo. The main results pane shows measures of agreement (various percentages and the kappa statistic, calculated at the level of characters coded) for two researchers regarding the application of each code within one source. Double-clicking on any line in the results pane opens the actual text, with

[9]This is most easily achieved by reviewing the text stored under each code for consistency.

coding stripes showing lines of text where each coder applied that particular code (with a further option of showing exact characters), thereby providing a useful basis for discussion about how coding is being applied.

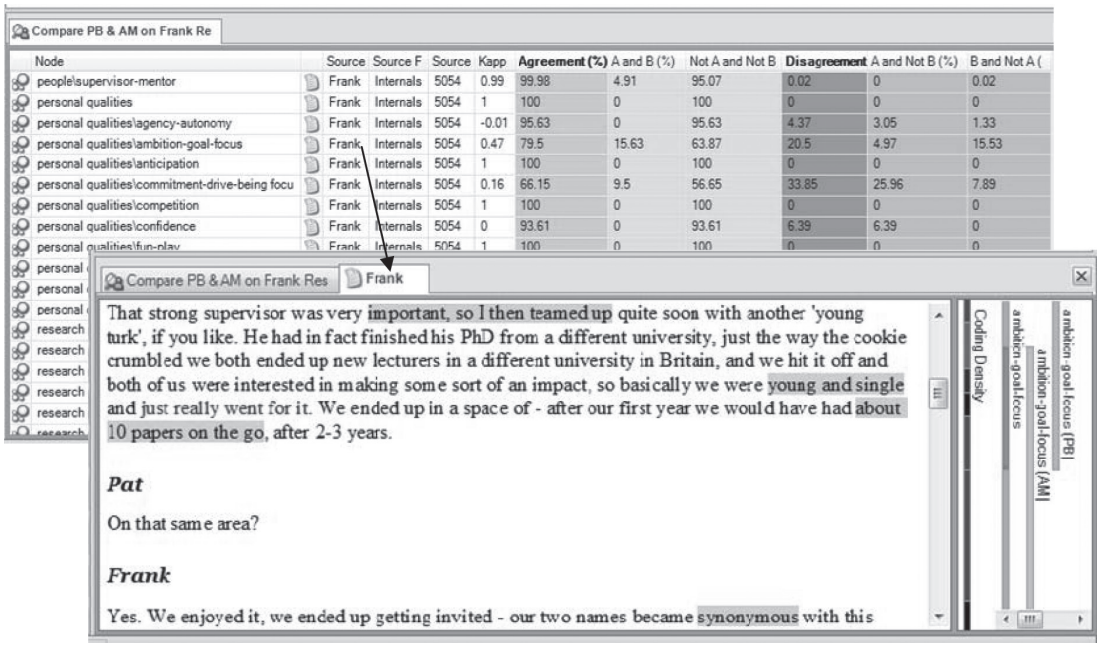

Figure 5.6 Results from checking inter-coder agreement in NVivo

Managing the process of coding

When am I done?

Codes always need revising as the work proceeds. Some will be changed, others deleted, new ones emerge. Generally speaking, coding and recoding are finished when all of the incidents can be readily classified, categories are 'saturated' and sufficient numbers of regularities emerge (Miles & Huberman, 1994). But what does saturation mean, in practice? It is commonly thought of as being when no new categories are 'emerging', that is, being found in the data (as seen in Box 2.8). Corbin and Strauss (2008) argue that, more than that, each category must be fully developed and described, with variations in each identified and preferably related to other concepts, before saturation is considered to have been reached and sampling (to extend categories and theoretical understanding) can cease. This could mean, of course, that one never finishes!

Once a category has been elaborated and refined, the question arises as to whether to code every instance where there is a reference to it in the remaining data. A related issue is whether to code every aspect of any category that is evident in any particular account, or just those that appear to be important or meaningful. I have seen gross examples of overcoding in some novices' projects using computer software, where every vague reference to a concept has been captured, often with extensive context, with the consequence that the meaning of the concept becomes clouded, and the patterns of association with other codes are more difficult to see.[10] (Perhaps this is not an issue with manual coding, where applying and using a code are more laborious.) Once again, how much to code is a question of methodology and purpose. If the goal of category creation is to reach a point where the category can be comprehensively described, then once no new information is forthcoming, there is little point in continuing and thereby adding redundant material. If the goal, however, is to go beyond description, and to identify the extent to which this category exists across the sample, or the extent to which it is associated with other particular categories (and when it isn't), and how it varies in relation to those, then it is useful to continue to code instances of a category for as long as data are being gathered and coded, that is, until sufficiency for the project as a whole is reached. This is implying that, when it comes time for analysis, it will be necessary not just to be able to identify associations and variations, but to be able to produce comprehensive evidence for those associations – and that, too, is an issue of methodology.

In some situations, especially where you are having to manage large volumes of textual sources, once categories are well established along with their variations and conditions, then techniques for rapid perusal of remaining recordings or texts can be applied appropriately. These techniques include listening carefully to non-transcribed audio for confirmation of or variations on what has already been established; or word searches through computer-based transcripts or documents to identify potentially relevant text. The passages so identified can then be perused in more detail.

Staying sane!

While novices working on computer may be at greater (though not exclusive) risk of becoming caught in an interminable coding trap, those working manually more often become overwhelmed with the volume and complexity of their data. Neither is a happy situation. If you are simply recording or organising rather than engaging in interpretive thinking, coding has the potential to become a headache, rather

[10]MAXQDA's solution is to allow each instance that a code is applied to be weighted, using a sliding scale of 0–100.

than a pathway to enlightenment. Helen Marshall noted that the source of this problem was coding rather than computers, quoting a contributor to an email discussion on the issue: 'I am a poor veteran of both methods and they both WRECK MY HEAD ... When I used paper and scissors I was constantly chasing scraps of paper – now I am a zombie in front of a confuser' (2002: 62, emphasis in original).

Coding must never become an end in itself. Practical strategies to avoid being either caught on a treadmill or lost in a pool of disconnected ideas include:

- Ensure you have efficient clerical systems for keeping track of relevant data and the ideas flowing from those.
- Automate whatever processes you can (such as routine coding of demographic data, or what structured question is being responded to), giving yourself more time for interpretive activity.
- Switch between coding and other tasks, such as reading and reflecting on a fresh item of data; reviewing the coding you have already done for one category; or checking what happens when you run a query using the data you have already coded.
- Regularly stop to write memos or draw models to capture ideas coming from the data. Lyn Richards' contribution to that same email discussion was that if you are coding for more than an hour or two without stopping to write a memo, you have 'lost the plot'.
- Focus on what you are wanting to achieve with your coding. You do not have to code every item of data, and you do not have to create a code for every idea contained within it! Remember to regularly ask the 'Why am I interested in that?' question.
- Once a category is taking shape, a process of refinement is taking place; so my recommendation, then, is to code only if the data provide a clear instance of the category, rather than a vague allusion to it. A helpful question to ask is whether you would want to find this passage of text each time you need to retrieve and review anything you know about this category – a consideration that is especially relevant if you already have a large amount of data relating to that category.
- Take regular time out from your data, walk right away from them (cleaning or weeding suddenly seem like remarkably attractive options!); give your brain a chance to move out of the rut and make fresh connections, and your unconscious creativity a chance to work.
- Take time also for discussions with colleagues about what you are doing. Research productivity is a direct consequence of doing so (Pelz & Andrews, 1976).

A final reminder: codes and coding in context

Remember that codes and coding are only one part of the process of qualitative analysis; they cannot exist in isolation. Codes will be used to link or test for associated ideas; they will need to become connected in various ways in order to build a metanarrative, a meaningful word picture, or a theory. Strategies for exploring and establishing or testing connections across data and links between codes will be covered in later chapters. Your more immediate tasks, however, are to determine what codes to use, and then how to create some order and structure in your coding system. These are the subjects of the next chapter.

Writing about coding processes

In the methods chapter

- Explain the approach you have taken to coding over the course of the project. For example, were the codes you used determined from your literature and theory, or did you create them from the data as you worked? Were they driven by any guiding philosophy or methodology (how)? Did you code in isolation, as part of a team, or were there regular discussions with others to provide some checks and balances?
- Provide an example that shows how a code originated and was developed and/or revised through the project.
- Describe the methods you used to record codes and to retrieve coded material, and also those for recording and making use of associated demographic or categorical data.
- Reflect on how or why your coding should be considered appropriate for your purpose.

Exercises

Exercise 1: Preliminary coding

- Read through Frank's account of his early career (Box 5.6), once he completed his PhD and began working with another new graduate – or select a magazine or newspaper article or a brief account from your own data. Underline significant words and phrases and use the margins to note the kinds of things you might want to index or code in that account. As an experiment, try working from the perspective of two (or more) of the following:
    - a grounded theorist who wants to understand the process of becoming established as a researcher;
    - a phenomenologist focusing on early career experience;
    - an ethnographer considering culturally based behaviours among younger, research-oriented academics; or
    - a narrative or discourse researcher looking at the structure or language of an account.
- Discuss your thoughts about codes and the notes you made with a colleague.

Box 5.6

Frank's early career development

Frank: I then teamed up quite soon with another 'young turk', if you like. He had in fact finished his PhD from a different university, just the way the cookie crumbled, we both ended up new lecturers in a different university in Britain, and we hit it off and both of us were interested in making some sort of an impact, so basically we were young and single and

(Continued)

(Continued)

just really went for it. We ended up in a space of – after our first year we would have had about 10 papers on the go, after two to three years.

Pat: On that same area?

Frank: Yes. We enjoyed it, we ended up getting invited – our two names became synonymous with this certain approach, and we went around Britain and into Europe giving research papers and getting ourselves published.

Pat: So you've got a reputation?

Frank: Yes, it wasn't a – you've got to keep that going for basically 10 years before you build up a reputation. Those in the know in the area knew that we were up and coming and we knew that they knew that we were up and coming, because they'd start writing to us and asking us. We basically knew that it was us carving out a niche for ourselves in the research part of the profession and that was our key to success so we kept at it. Eventually we both went apart and our research paths have diverged.

Further reading

Saldaña (2009: Chapter 1) introduces his coding manual with an overview of codes and coding.

Coffey and Atkinson (1996: Chapter 2) provide a rationale for and approaches to coding with examples from anthropology.

Turner (1981) usefully describes the detail of his methods for using an index card system in his evolving approach to coding field notes in a grounded theory study.

Bernard and Ryan (2010: Chapter 4) describe, with examples, codebooks and manual coding processes.

Morse (1997), in a delightfully titled editorial, discusses how coding reliability processes can render interpretation of data 'perfectly healthy, but dead'.