

Handling Qualitative Data

A Practical Guide

Second Edition

Lyn Richards

© 2009

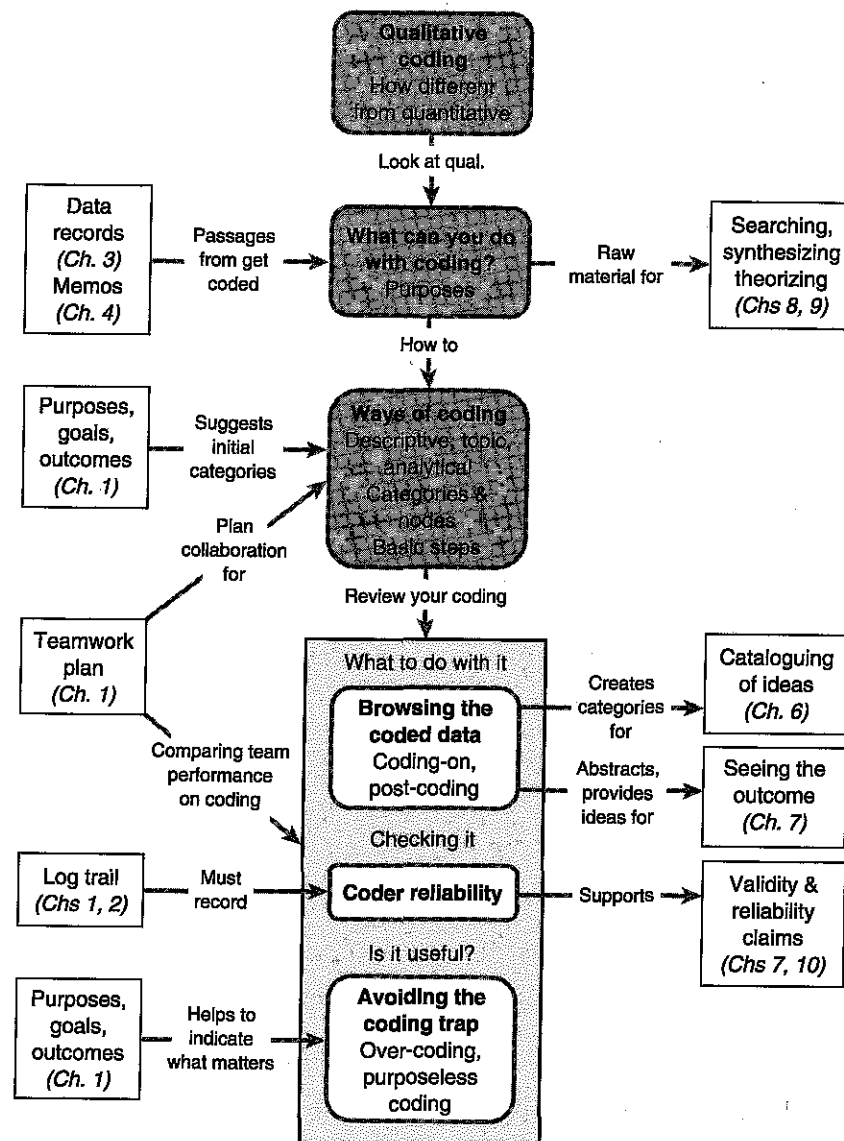
 **SAGE**

Los Angeles | London | New Delhi
Singapore | Washington DC

EARLIER
CHAPTERS

THIS
CHAPTER

LATER
CHAPTERS



FIVE

Coding

Most qualitative researchers code. Coding generates new ideas and gathers material by topic. This chapter tackles the coding task, the different sorts of coding and how to ensure that it does not become dominating and unproductive. The emphasis is on purposive coding, using the results of your coding to develop ideas and take enquiry further.

Reading and reflecting on data records, you see your project document by document. But your project requires you to see across the data, and above the individual documents, to themes and ideas. Usually it also requires you to gather all the data on a topic, to think about it and rethink it. To gather everything on a topic, you need to *code*.

In a study of isolated housing estates, you may have noticed that the theme of 'privacy' seems to occur in several data records, even when people are deeply committed to community. Move up from the data to the concept. You want to consider the concept, privacy; not the individual reports, but the varieties and patterns in all the discussions of privacy. This will help you reflect on the values and experiences of privacy and how they coexist with the apparently contradictory values of community.

To shift focus to the topic or concept, you need to access the data differently. You want to read and reflect on all extracts, from any of your records (or from just some selected ones), where the theme of privacy occurs. *Bringing those parts of the accounts together, you will be able to learn about the variety of responses or experiences, and reflect on how the emphasis on privacy impacts on goals of community and family.*


Almost all qualitative research involves some sort of coding (though different methods do it very differently). But that does not mean you need to spend your life coding. It's a first step to somewhere else.

Qualitative and quantitative coding

Most qualitative researchers code, but no qualitative research is only about coding.

The term itself confuses. In common use, 'coding' refers to data *reduction* either by a system of symbols (as in Morse code which reduces everything to dots and

Table 5.1 Two very different modes of coding

	Quantitative	Qualitative
Place in research process	Normally a single stage between data collection and analysis	Occurs throughout project
Relation to categories	Applies predetermined categories	Generates categories
Relation to original data	Code applied summarizes or represents – and replaces – original data	Code retains copy of or pointer to original data, ensuring access
Flexible or inflexible?	Revisiting is often not possible, since original data not retained	Revisiting of coding to check development of categories
Changing coding categories during project?	Normally after piloting, no new categories will be added	New categories are generated until last stages of project
Reshaping categories?	Collapsing of codes to obtain a simpler picture	'Coding on' from coded material can create new categories/dimensions. Merging of categories takes place as common meanings emerge
 Team processes	Coding is a clerical task that can be severed from analysis	Coders are doing analytical work, involved in project interpretation

dashes) or by numbers (as in the coded boxes to tick on a questionnaire). Most researchers have done quantitative coding before they try qualitative coding, so it is worth starting out by clearing this confusion. In qualitative research you do usually have to do some of this sort of data reduction, to store information *describing* the attributes of an interviewee, for example (gender, age, ethnicity, etc.).

But whilst quantitative coding reduces data, *qualitative* coding is about data *retention* (see Table 5.1). The goal is to learn from the data, to keep revisiting data extracts until you see and understand patterns and explanations. So you need to retain the data records, or the relevant parts of them, until they are fully understood. Coding is not merely to label all the parts of documents about a topic, but rather to bring them together so they can be reviewed, and your thinking about the topic developed.

This sort of coding is more like the filing techniques by which we sort everyday information and ensure access to everything about a topic. Recipe clippings are most usefully gathered by food type. Go to 'cakes' and then 'chocolate cakes' and all the recipes will be there to review. In sophisticated (usually computer) systems they may be filed by more than one dimension. Ask for anything coded 'cakes' and 'chocolate' and you should get all the recipes for chocolate cakes. But

however you got there, you want the recipes, not merely the information that there are six. The point of this sort of coding is to *find again* the material coded. Coding allows you to return to the data you want to inspect, interrogate and interpret.

What can you do with coding?

Qualitative researchers code in order to get past the data record, to a category, and to work with all the data segments about the category. Coding aggregates them, so you can then work with them together, gaining a new view on the data. It's a first step to rethinking the data. When you make a final report on this project, you are aiming to do more than describe all the text coded at a category. That report will give the results of analysis deriving partly from your *work* with the coded data.

This chapter has a message: coding should always be for a purpose. It is never an end in itself.

Why would you want to work with that category? There are many ways that a researcher can use coding: that is, for all the purposes where it matters to have all the data about a category.

Each of the 'Methods in Practice' projects used coding. Go to the 'Working with Data' sections to compare how it was done and used.

PURPOSES OF QUALITATIVE CODING

There are very many purposes for qualitative coding, and most researchers use it for at least several of the following:

- to reflect on what the coded segments tell you about the category, and its meanings in the project;
- to ask questions about how the category relates to other ideas from the data, and construct theories about those relations;
- to gather all material about a case, from different sources, so you can apply the information about that person or site to everything from there, and compare cases on their attitudes, experiences, etc.;
- to make further, finer categories, from finding different dimensions in the data gathered by the first coding;
- to search for blends or combinations of categories, to find patterns in attitudes on this subject, for example by gender, or to compare text at different categories, seeing the category from a different viewpoint; and
- to compare how different researchers interpret data.

Some, or even all, of these processes may matter in your project. If so, as you start coding, reflect on what you want coding to do for you.

The purposes to which coding is to be put strongly influence the standards of coding and the style you will use. So ask what you want coding to do. Especially when you are working with software, searches of patterns of coding can be very rigorous, and their results will vary according to the amount of context coded. *When people talk about 'business', are values of altruism or social good ever mentioned?* If I always code the whole passage when these themes come up, a search for where they intersect will answer my question. But if I have merely coded the word 'business', it probably will find nothing.

Ways of coding in a qualitative project

I find it useful to distinguish three sorts of coding in qualitative research, since they require very different processes. The terms I use are 'descriptive', 'topic' and 'analytical' coding. (See *Readme First* (Richards and Morse, 2007), Ch. 6, for a fuller account of these and their place in different methods.)

Most studies use all three, and two are qualitative tools in the sense that they are interpretative processes. Researchers are much helped by seeing them as very different tasks, using different tools and for different purposes.

The first, descriptive coding, is more like quantitative coding (see Table 5.1 on page 94). It involves storing information about the cases being studied. Information usually applies to a document or a case, so there is not a process of selecting just the text to be interpreted – simply, the appropriate values of variables are stored at that case.

Then there are (roughly) two very different types of qualitative coding to gather all the text about a category. Topic coding is the hack work of the qualitative researcher, labelling text according to its subject. This labelling can often be automated with software.

Analytical coding, not surprisingly, is central to qualitative enquiry. This is the coding that leads to theory 'emergence' and theory affirmation. Don't think of automating it! But although the computer won't automate interpretation, it does give far more flexibility to this central interpretive task, helping you to read and think about the coded data, and keep coding.

A passage of text will quite normally require all three types of coding. *A man interviewed is discussing the need for community action in the local council elections, in which a schoolteacher is a candidate. This man says that he never listens to gossip about the schoolteacher; it's women's stuff. But he does worry that she is standing for local council, when she is obviously not a responsible person.*

- Descriptive coding: Firstly, store the information about the speaker, perhaps about three attributes: gender, age and job (*male, 45 yrs and tradesman*).
- Topic coding: Now, what topics are being discussed in this passage?

The need for community action and the schoolteacher; perhaps too we need to code for her multiple roles.

In two ways, the coding has described the passage: what sort of person offered these ideas, and what they were about.

- Analytical coding: Now, what's going on in the statement about the schoolteacher? Several themes there that need noting, about patriarchal assumptions, the credibility of 'gossip', the informal networks of women, the authority of the schoolteacher and the interplay of interpersonal and political relations.

The one passage may now be coded 11 times, because it's about 11 different aspects of the project. This is not a problem if you are working in a computer rather than a filing cabinet. In the computer, these codings will be stored differently (descriptive codes are stored as attributes, in spreadsheet-like displays). As such coding progresses, we are able to ask useful questions – *Did men always deny they gossiped? Are the negative attitudes to the schoolteacher coming mainly from the over-40s? And how do they relate to attitudes to community action?*

Qualitative coding and computers

Qualitative records, in the days before computers, were coded much as the records in a home filing cabinet are coded.

Descriptive coding was done by storing face sheets of data about the respondent or the institution being studied. (Or worse, it was done by punching holes in cards and sorting descriptive categories with knitting needles!) This meant that information was often inaccessible when you wanted to ask a question like 'What do the women say about privacy?'. Computers changed the use of such coding totally since they can store data about a case very efficiently and give access to it at all stages in a project. Descriptive coding can usually be done by importing the data in tables, rather than selecting and coding text, or by working in a table of cases and attributes, selecting the value for a cell, as you would work in a spreadsheet.

Before computers, both types of qualitative coding (topic coding and analytical coding) were clerical challenges. Since the goal was that the researcher should be able to retain a copy of or pointer to original data, a lot of paper was required to ensure access to all the data about the topic or concept! *If a passage was about privacy, you made a file for 'Privacy', copied the data and placed the relevant segments in that file. If it was also about community, you would copy those segments again and place in a file titled 'Community'. Then, when you wanted to write about privacy, you took out that file and spread the cuttings over the floor.* The task was boring, time consuming, and not very rigorous, since dogs and babies

were likely to mix with the precious paper segments. But most importantly, it worked only when you restricted your goal to getting back everything about one topic. It failed when you want to ask just the sorts of questions that qualitative research is about. *When people talk about privacy, how do they reconcile it with values of community?*

Even from that simple example, it is evident you will be able to do more with the coding stored in your computer project than you could with any number of cardboard folders. The computer stores information, not paper. Each abstract idea you have – the category – is accessible at any time. The computer can go to all these places or combinations of them, for information and exploration of patterns.

Categories and where you put them When coding on paper, most researchers talked about the *categories* or *codes* they were creating and they filed in folders all the material *coded* at these categories. On a computer, you can much more easily treat these categories as objects, move and arrange them and explore the relationships of the data coded in more than one of them. You also can manage categories much more flexibly than in a filing cabinet, because software can store their positions in an index system (see next chapter).

In information systems the term for a place in an index or a network is a *node*. The term is used across a wide range of areas (botany, engineering, geology) to represent places that are related. The orchardist prunes fruit trees above the node. New growth comes from the node. (So too in qualitative research.)

The term node is used here to refer to the place where the software stores a category. There is more about nodes and node systems in the next chapter. You create them to hold categories and store coding, and you can handle them flexibly and ask questions about them, since the software will be able to check at any time where they are and what coding you have done at them. They need not hold coding; for example, you might store anticipated categories at nodes in your system, ones that you think might be important later. Or you might create them to store ideas and memos about the goals of the project (Chapter 1) or the quality of the data (Chapter 2), or topics to be considered in another stage of the project (Chapter 3).

You will find that terms associated with coding and processes used differ among qualitative software packages. It is important to check that you understand how coding is stored in your software, and what can be done with the coded material and the categories.

For very different detailed accounts of the processes of coding on computer software, compare the **Wedding Work** project and the **Harassment Complaints** project.



BASIC STEPS OF QUALITATIVE CODING

There is no mystery in doing qualitative coding:

- To store descriptive coding (information about characteristics or attributes) is simple; either one at a time, or by table import, you allocate to each case the appropriate value of each attribute.
- To code something qualitatively, *retaining* the relevant data, you do three things:
 1. Select the material of interest and decide what it is *about*.
 2. Create or find the appropriate *category*. In a manual system the category went into a cardboard folder labelled with its name. In your computer project the category, its description and any coding will be stored at a node.
 3. Now, code! Working on paper, you put the relevant material or a copy of it in the folder, or mark it in some way on the paper. Working on computer, depending on the software you are using, you copy the extract to the code, or place at the code a pointer to the relevant material.

Now, consider what coding your project needs.

Descriptive coding

Descriptive coding is the sort of coding occurring in quantitative studies – storage of information that *describes* a case.

Every qualitative project requires this sort of information. Where cases are being studied – cases of interviewees, cases of schools or businesses, for example – there will be information about the *attributes* of these cases (the person's gender, the school's size and so on).

If you were working on paper, you would probably save a face sheet of information about each case, or a table summarizing them all. In software, similarly, storing information about a case's attributes is handled in a separate system from coding text. Storing and using these data is both easy and highly productive given computer software. If you have a table of data about your cases from a spreadsheet or statistics package, it can be imported directly to your qualitative software, creating the attributes of your cases such as their gender and marital status. Now you can ask questions about the sample using that information, and split or sort the data by its attributes.

The challenge is usually not whether to store such information, but how much to store. The amount of such information available may be overwhelming. For example, in a mixed methods study, cases of interviewees from a survey may

be selected for in-depth interviewing. From the survey, you have data on perhaps 50 variables, but it is highly unlikely that all these are relevant to your qualitative questions. As in many areas, software compounds the challenge, since it removes barriers.



If you are working in a team, you will find that descriptive coding is the only sort of coding easily handled without collaboration and communication. The task is basically storing information that is (presumably) reliably recorded. For topic or analytical coding, collaboration is essential.

STORING ATTRIBUTES

- Plan to collect all information about cases that you'll need to ask the project's questions. To know what you'll need, reflect on what you want to ask. Your research design should indicate what information is required.
- Keep this information systematically, preferably in table form. This will remind you of all the data needed for each case, and as you fill in each row of the table, you'll notice the patterns of characteristics your sample is building up.
- Don't import attributes you don't want. Excess information makes it hard to see the data clearly. It can always be imported later (so long as you heed the next point!).
- Don't discard information that you may want. Find a way to store it so it doesn't confuse and clutter the data you are working with. Information about attributes that may be needed later can be stored in a separate spreadsheet or table, and imported later if relevant.
- Storing information can be boring work, so aim to automate it. If you are using software, you can do this by preparing and importing tables.

Topic coding

Topic coding is my term for coding that merely allocates passages to topics. It usually involves little interpretation. You are putting the data 'where they belong' – a sort of data disposal. *'This is about the local schoolteacher', 'This is about neighbouring', and so on.* This sort of coding is almost always easy to do and almost always boring. It is, of course, also necessary. *If the role of the local schoolteacher is relevant to your community study, it is obviously important to get in one place everything said about the role and its occupant. Only then will you be able to get an overview of the range of interpretations of her role, and write about the ways her authority was challenged. Moreover, coding everything about the schoolteacher in one place is necessary if you wish to ask whether, when there was gossip about the schoolteacher, it came from her neighbours, the parent group or her colleagues.*

Often, topic coding dominates early in a project, because it requires relatively little understanding of the situation. *You may be very puzzled to understand the complex relations of privacy and community, but you do know if they are talking about the schoolteacher.* The topic coding may also be a first step to more interpretive work. *Having coded everything that is about the schoolteacher, you can review that material and from it develop the analytical categories.*

Such coding is relatively unchallenging, and this may be a problem. The 'coding trap' (see below) is almost always about topic coding.

TOPIC CODING

1. Plan what topics you need to gather data on in order to ask the project's questions. Just as for attributes, to find out what you'll need, reflect on what you want to ask:
 - Your research design should indicate what information is required. A great way to start is to put your research design into your project on computer and code it, making the nodes as they are indicated in the design.
2. Record these as a starter list of categories, or better still, create them in your project. If they can be organized, in category/subcategory trees, do it. Critique this starter list, checking for relevance and overlap of categories.
3. As you read, ask always 'What is this about?'
 - If the answer is a topic for your project, select the text about it.
 - If you have that topic already as a category, select it and code.
 - If the topic has not been recorded as a category, create the category, and code. As a new topic appears, consider whether it deserves a place in your project. Why will you want all the data about this topic coded at one place?
4. Automate topic coding either by section coding (if topics have been set out as sections in the document), or by text search (if a word or phrase identifies this topic).
5. If you can't automate it, protect your project against boredom:
 - The topic coding task can be given to a competent assistant (it's a good training ground for software competence). Better, combine it with analytical coding. Keep visiting the topic. This is getting analytic!

Now you have a lot of material about the schoolteacher: browse it and you will be struck by the very different ways she is seen and the different responses to her efforts for the community.

A note on autocoding Using software, it is possible to do rapid coding to identify the key topics or issues in the record immediately. This is done either by identifying the section of the document about the topic, or by finding specific words in the text. (Recall the warning in Chapter 1 that to do autocoding you need to know how to format a document so the relevant text will be found and the relevant context can be coded.)

It's important to be clear that such 'autocoding' is no substitute for your interpretation!

- The software is not reflecting on the meaning of your text. It is searching mechanically for words that occur, and coding mechanically the slab of text you specify. Never assume that a 'find' implies meaning: it indicates merely the presence of specified characters.
- It will miss anything you don't tell it to look for.
'Schoolteacher' will not find references to the teacher by her name. You can of course nominate alternative strings to search for, but did you remember to search for 'that bitch'?
- Autocoding will inevitably make some weird errors, so you must always check the results. If in doubt, check back to the context. And always keep a note that this coding was done mechanically.
- You have a choice of ways to clean up the errors: review each find as it's made or (usually better) browse all the finds, in their context, and think about them. Always do one, and sometimes both. Never leave autocoding without reviewing – later you will be tempted to think the coding was reliable.
- Autocoding will always save huge amounts of time, so it is tempting to over-code, and the resulting topics may be far too numerous and confusing.
- One very helpful role for autocoding is to get everything about a topic in one place in order to explore and establish the more subtle categories you should code at. In other words, autocoding can be used as a first step to analytical coding. (See below for a discussion of coding on from the first coding.)



Examples of the use of autocoding with computer software are in the Harassment Complaints project.

Analytical coding

The three sorts of coding are not always clearly different, and to some degree analysis, of course, is involved in all three. But it is worth distinguishing coding that requires interpretation from descriptive and topic coding (both of which are fairly matter-of-fact processes, even to researchers who dispute the existence of facts!). I use the term 'analytical coding' here to refer to coding that comes from interpretation and reflection on meaning.

Of the three sorts of coding, it is the hardest and also the most rewarding. Rather than just store information or name the topic of the text, you are considering the meanings in context, and creating categories that express new

ideas about the data, coding to gather and reflect on all the data related to them. This is qualitative research!

Qualitative research is not a task to be hurried. The goal is careful interrogation of the data. What is a particular passage about? What category or categories will properly represent that passage? What context should be coded there? Well-handled, analytical coding is a prime way of creating conceptual categories and gathering the data needed to explore them. Coding is a first step to opening up meaning.

What meaning is sought, of course, will depend on the project's question and method. In some approaches to text, coding may be used to mark and return to features in the language (for example, ideological assumptions) or points in the narrative or conversational structure of the text (contradictions, omissions, turning points). In others, it will be a way of creating concepts. In any of these cases, the act of coding has gathered the material that brought the idea and put a pointer to it, so the researcher can return there to think some more.

Analytical coding follows most easily from the quick steps to 'taking off' from the data suggested in the previous chapter. In a series of questions, you moved from some detail of the document ('That's interesting') to comment about it ('Why is it interesting?') and then to the abstraction ('Why am I interested in that?'). The answer to that latest question was an analytical category. Annotations and memos were used to record the insights. Now, replay those steps for one more time, this time thinking coding.

ANALYTICAL CODING – TAKING OFF AND CODING IT

1. As you read, if a passage is interesting, select it ('That's interesting').
Perhaps a hostile comment was made about the nosiness of neighbours. Ask 'Why is it interesting?' It seems odd in context, as the interviewee had been very positive about community. Make a note.
2. Now, you have a passage selected: where do you code? Step up to the abstraction: 'Why am I interested in that?' This is a very different question. The answer is a category, and a place you will want to code data. Make the category, and carefully name it. (Naming is an analytical process in itself. You may also want to describe the category you have created.)
New category for 'tension between ideals and reality', and another, since it seems also necessary, for 'ambivalence re neighbouring'.
3. Code the selected text at the category. (Software will place at the node a reference to just that text.) Before you move on, you might use the techniques

(Continued)

(Continued)

described in Chapter 4 to reflect 'up' from the text. If these take flight, write a memo about that category.

4. Now later in that document, or in another, when you again hear the ambivalent messages about nosy neighbours but how needed neighbour help is, you know you are interested in that. Select the text, find the category you created, and code.

Analytical coding rapidly becomes a very smooth and exciting process of identifying text to be coded, creating categories or finding the categories already created and coding at them. The main challenges are to keep your thinking 'up' at this abstract level, and to keep generating ideas and questions. Especially if you are tired, it's easy to drop back down into topic coding, merely saying what the data are about. Be aware of this pattern – time to walk the dog.

A note about in vivo coding One valuable technique is to look for 'in vivo' categories. This term (from grounded theory) refers to categories well named by words people themselves use. Several software products allow you to select words and code *in vivo*, naming the category with the selection.

Often, by focusing on a phrase or expression that is surprising or recurring, you will find you are helped in taking off from the data again, thinking about themes, rather than merely noting the topic discussed. Understanding grows as you bring together instances of a 'sound' in the data.

An example from my real project on community and privacy was an *in vivo* category 'not in your pocket'. Definitions of good neighbours, in the setting I was studying, often contained that phrase, or other phrases asserting what good neighbours should *not* be in (your lap, your house, your kitchen or your hair)! As these *in vivo* categories built up, they helped me reflect on the sense of physical invasion and danger that was associated with neighbouring (Richards, 1990).

Revisiting the coded data

Coding in qualitative research is never merely a way of getting everything about a topic in one place so you can count it or summarize it. Of course you may wish to do that, but if that is all you want to do, why have you gone to the trouble to retain the detail of the data until now? Sufficient information to support the count or the summary would have come from ticked boxes in

a questionnaire. If you have coded qualitatively, retaining the text, it must mean there is more to be learned from reviewing it in this new context.

Whether coding has been only according to topic, or more analytical, there will always be surprises in reviewing coded data. Often the most extraordinary discoveries come from browsing together all of the segments you coded at a category.

Don't wait to review the coded data. From the start of coding, visit the categories and think about the data coded there. You literally see the segments anew. At any time when you want to think about the theme, go to the category and read, think, reflect. Coding has given you an opportunity to review the data, getting closer to the words of the text and revisiting their context.

WORKING WITH THE CODED DATA

Find how to see together all the passages coded at a category. Read them as though they are a different sort of document – about the category:

1. Look for differences, especially differences that surprise you. Can you explain why two people you had seen as similar have such different views on this topic?
2. Now look for similarities that surprise.
3. Scan the material. Are you happy with your coding? Anything here that seems not to belong? If so, find how you can 'uncode' that passage from the category.

Assess context. Perhaps a segment doesn't make sense because you coded only a narrow context? Go back to find the context and read it in the original, or expand the coding to give the appropriate context.

The words spoken about the schoolteacher sound very angry – but was the anger in response to an earlier question?

4. On the basis of the data coded here, reflect on the category – what have you learned about it? (Record these thoughts in a memo.)

Seeing together everything about the same topic may be the critical step to understanding it. You can of course do this with paper files which store all the copied segments about a category, but the ability to do it on computer brought an extraordinary change in method. Because those segments are not cut out from context, you can work 'live' with the data about a category. Software can take you back there, so you can code the record differently, as your ideas develop.

If you have previously coded on paper, there are two very significant differences in this way of browsing the coded text. Both will support new ways of working:

- Coding on computer can be very flexible. One major advantage for novices is that they can shed the fear of not immediately 'getting it right'. Do first-pass coding, into very general, broad-brush categories: 'privacy stuff' or 'community stuff'. (My personal indicator of a broad-brush category is that its name includes 'stuff'!) As you become more confident of the meanings in the data, return to browse the broad-brush categories and recode the data to finer categories with more definitive names.
- Coding on computer need not decontextualize. For some purposes, you want to have just exactly the text coded at a category (for example, if you are searching for anything coded at 'privacy' and also at 'loneliness'; see Chapter 8 for such processes). But for other purposes, you want generous context, to see what was said before, or what another group member had said about this. The browsing process can go out to the wider context as needed. Note that this means you don't have to make your coding over-generous of context.

Coding on to develop your ideas

For any sort of coding, but particularly analytical coding, you will want to reconsider the category as the material builds up.

Revisiting may be simply for housework. The main reason for revisiting descriptive coding is if you decide your early categories were too coarse or too fine (we needed age in months, not in age groups because these were too coarse). Topic coding can also be adjusted similarly. *You may need to separate out the material on the schoolteacher's work and her personality, or correct your coding – the 'bitch' referred to was the postmistress.*

But revisiting text coded at a category, and reviewing coding, can be highly analytical. Your understanding of the categories and concepts emerging as you code will develop and change as you review the data. (It was for that purpose that you retained the data.) Analytical coding gathers material that should be rethought and reviewed during the project. *Data coded at 'privacy' will need recoding when you discover there are several different meanings of privacy in this community.* Now you will need to code on from the first coding, to create and reflect on new categories.

CODING ON

- Browse all the data coded at a category, and reflect rather as you would on the text of a document. What new meanings can you see?
- When a new category occurs to you, create it in your coding system and code the text, just as you would if you were in the document.
- Now back to the originally coded data – should these segments all be in the same category? Are you discovering new dimensions of that category? If so, record these as new subcategories, coding the material that belongs in them on from the original coding.

This process of revising coding, and *coding on* to newly discovered categories, makes coding a process of discovery rather than merely description. Work this way, and coding is never an end in itself, just a first step to thinking through the material.

If you are working with software, the ability to code on from a first stage of coding may radically change the way you code. There is no need to 'get it right' first time. Coding with a broad brush, you can gather everything about a general topic in one place. Then, as the subtler meanings of your data emerge, revisit and code on from that broad category to codes reflecting finer dimensions.

The use of coding on in revisiting a previously coded data set is described in the Harassment Complaints project.



Post-coding for data reduction

In most qualitative projects, the goal of coding is to gather together everything about a topic or an analytical concept, in order to review and refine thinking about this category. You go on from the category to finer dimensions. *There are several different meanings of privacy in this community; some of this text is about what for now I'll term 'haven home', some about 'personal life privacy' and some about 'anonymity'. Not all will conflict with ideals of community co-operation.* In browsing and working with the coded data, the researcher will expand their ideas and create new categories.

But the same tools, for coding and 'live' browsing of coded data, can be used in large projects for data reduction. If your challenge is a very large number of open-ended responses to a survey, the goal may be 'post-coding' them. This involves deriving from the answers a group of categories that sufficiently represent the range, then reviewing each answer and coding to the appropriate category. The results of this post-coding exercise can be simple counts, or information about the coding of cases, exported to your statistics package if appropriate.

Question 6 in a survey of this community asks each respondent, 'What does privacy mean to you?' Answers are typed up and coded automatically, descriptively and by topic. (Characteristics of the respondent are imported from the statistics package, and the topics are identified by question headings, so can be section coded.) Now go to the new category, 'Q. 6. Privacy', and browse all the answers to that question. Make a new category each time you see a new meaning of privacy. Rapidly you will arrive at a satisfactory group of new nodes. Importantly, the text for each of the newly discovered meanings of privacy is available, to inform debate about the category or illustrate it with quotations.



Coder reliability in qualitative research

Reliability in qualitative research is a contentious topic, discussed in more detail in Chapter 7. Qualitative methods are all about interpretation and individual agency. Field researchers claim credibility for research in very different ways and with different standards from those used in a randomized control trial.

But being reliable (to use the adjective) beats being unreliable. If a category is used in different ways, you will be unable to *rely* on it to bring you all the relevant data. Hence, you may wish to ensure that you yourself are reliably interpreting a code the same way across time, or that you can *rely* on your colleagues to use it the same way.



The need to check consistency is of course greater in larger projects, for many reasons. When data records are bulky, and project timelines longer, researchers lose recall of earlier coding or ability to review colleagues' coding. And there is more emphasis on consistency for two reasons. Practically the coding is more likely to be output directly to another stage of analysis (for example, statistical) rather than used to revisit and review coded material. Such projects are likely to spend less time browsing coded material (and thereby confronting inconsistencies). And politically, these projects are likely to be in research contexts where the standards of quantitative research prevail and suspicion is cast on qualitative processes that cannot show reliability checks. (Don't get aggressive about it: qualitative researchers can do this!)

Warning! Do the tests but interpret them carefully. You would not expect consistency of qualitative coding, over time or between colleagues, and you may not desire it! Your understanding of the categories has of course changed over time – that's built into the method. (Your coding categories are emerging as you learn from the data.) And your colleagues probably see the project and data differently from you. The research design may even specify that two researchers will come from different disciplines to give (*reliably!*) different views. So we will not be concerned when we find inconsistency. We do, however, need to know about it, discuss it, to place any coding-dependent analysis in that context, and document differences.

Coder consistency tests usually apply to combinations of topic and analytical coding. Descriptive coding is not interpretative, but requires only accurate entry of information about a case.

DOING CODER CONSISTENCY TESTS

In this context, reliability means consistency, and it is easy to discover inconsistencies in coding over time or between colleagues. The best method is to compare two codings of the same document:

- For consistency over time, code a clean version of a document you coded earlier. Don't cheat and check the earlier coding! It is important that you discover how different your coding will be now.
- For consistency between colleagues, each colleague codes a duplicate of the same document.

Now for either purpose, print out the display of that coding and compare. (Or use your software to make a report on the differences.) You are interested in the following differences:

- What categories are used by one but not by the other?
- What differences are there in the segments selected for coding at each category?
- Are the styles of coding different? In particular, is one version of the document more richly coded (at more categories) and does one coder select substantially larger passages for coding?

Coding consistency is a different challenge if you are working in a team. If you and your colleague are coding at the same category, you do need to know that you are using it similarly.

Visit the **Youth Offenders** project for the saga of a project with very many coders in many sites.

Avoiding the coding trap

Coding, especially on computers, is terribly easy! It is easy to do ever-increasing coding – especially descriptive or topic coding – if you are unsure what else you can do and the theory does not seem to be emerging from your data. Sadly, this is the best way of stopping emerging ideas.

Coding was always a trap to researchers, but its danger is far greater with software. This is ironically because of one of the great achievements of qualitative computing: that at last, as the method demands, there are effectively no restrictions to the number of categories you can code data at, or the number of times you can code very rich data. So coding can be a way of never finishing your project. The problems are worst when researchers become over-zealous about coding everything, and making as many categories as possible. This sort of coding fetishism can delay or even destroy a project.

There is a particular version of the coding trap for teams: in order to keep workload steady and output guaranteed, topic coding takes over. When collaborative construction of the meanings of codes becomes a challenge, it is easy

to keep topic coding. *It's about the schoolteacher, and we'll worry later about the more subtle meanings.* The only answer is meetings and discussions about what you are coding for. Meet often, especially when coding has become routine.

To avoid the trap, insist that all coding must have a purpose. If it has a purpose it will be interesting. If it's boring, stop and do something else.

AVOIDING THE CODING TRAP

Practical routines that work to avoid the coding trap are the following:

- Do all descriptive coding (storing information about attributes) efficiently and where possible automatically.
- Watch the balance between topic and analytical coding. The first will dominate when you are uncertain about what is to be done, or prevented from progressing. Keep asking not just 'Where does this text go?' but 'Why am I interested in that?'. If your project has no interest in it, don't code it.
- Especially early in a project, during coding, be generous with your categories. It is very easy with software to revise and remove categories that prove extraneous, so don't spend time worrying if a new meaning deserves a new category; just do it. (If it is node-worthy, make a node for it.)
- Periodically monitor what you are coding, where and why. Review and be stern about proliferating categories. On review, any new category has to promise something. (What might you use it for?) There is no point in coding a category if you are not going to use it for discovery that's relevant to your project.
- Always combine coding with thinking about the categories you are creating. Frequently drop out of the document you are coding to visit a category you coded it at. What else is coded there? Does it now seem to have several dimensions? Reflect on the different meanings in the coded material.
- If it's boring, stop! Take time out, do something entirely different. An excellent step aside from coding is to assess the categories being created and do some thinking about them.

Establishing your personal data processing style

Qualitative research is intensely personal; agency and ownership of the data are critical factors in the project. Establishing a data processing style that you like is essential – as is reworking it so new ways of relating to data keep it alive.



Celebrate variety in coding styles within your team. Swap tricks and techniques and write a paper about the different ways you code, and the different

sorts of insights and discoveries achieved by different approaches. This will assist you to write the coder reliability paper in the proper context!

SETTING A STYLE THAT IS YOURS

1. Start coding data records as they come in and keep coding – never allow them to build up unprocessed. This is a hard job, so treat it as a challenge, a crucial step in your project. Carefully assess your access paths to all data that may matter.
2. Don't delay because you are uncertain of the coding needed: as a first pass through the data, make broad categories and do 'broad-brush' coding, especially early in the project, to indicate areas of text you wish to revisit. Remember, you can go back for coding on.
3. Browse categories frequently, to explore the variety or similarities of the material being gathered, and reflect on the idea or concept the category represents – and write about it.
4. Watch the categories you are making, seeing them as a continually changing representation of what you are finding in your material.
5. Aim for a system that is yours. If you're convinced some 'expert' has a perfect system, deliberately critique that method. Don't be afraid of trying new ways of seeing new meanings in your data. Design and celebrate tricks and gimmicks of your own that work to alert you to interesting things and spark new views on previous data.
6. Get quick and efficient at data disposal tasks – descriptive and topic coding – so they don't become onerous. If it can be properly automated, automate it.
7. Find a qualitative colleague or group of colleagues: if there isn't one, start one. Coding discussions in groups are highly productive of ideas and exploratory concepts; you will never see your data the same way after sharing a transcript with a group and discussing it. If you have no real group, find a virtual one, a supportive list serve or discussion group.

Writing about coding

Log changes in your coding method and codes as part of your continuing report on the story of the project.


- A new category that has any analytical momentum requires a memo.
- A shift in the meaning of the category requires thought, and noting, especially in teams.
- The building up of data at some but not other categories requires reflection.



As with previous advice about project logging, the basic requirements are that you note a new step and why it was taken, what were the alternatives and what are the implications for the growing project. Don't shirk this logging. You will find that when coding gets underway, in any method, it has a momentum of its own. (Strauss remarked to me in 1992: 'We create categories like God creates raindrops!')

To do

For these exercises, choose just one document in your project. Work with it on paper or follow the tutorial for this chapter on the website www.sagepub.co.uk/richards to conduct the following exercises on computer:

1. Do some of each sort of coding (pp. 96–97) for the data you have so far in your project:
 - Descriptive coding: Enter attributes and assign the relevant values to the cases you are currently considering.
 - Topic coding: List what you expect will be your main topics and code the relevant data at them.
 - Analytical coding: Make categories for the new ideas or concepts that you find in the data. For each new category you create, write a description and a memo.
 2. If you are working on computer, find how to automate some non-analytical coding:
 - Do descriptive coding by table import.
 - Do automatic coding of questions in your interviews.
 - Use text search to gather material on a topic.
 3. Compare your coding with a colleague's coding of the same document. Discuss categories you have used differently, and the differences between your styles of coding.
- 
- If you are working in a team, extend this exercise to discussion of ways to maximize benefit from coding different styles and manage the differences.
4. Choose one analytical category, at which you have done some coding. Browse the data you have coded at this category (pp. 104–07):
 - remove inappropriate coding;
 - revisit the context of coded passages; and
 - code on to make new categories.

Suggestions for further reading

The processes of coding are covered in *Readme First* (Richards and Morse, 2007), Part 2. For a wider-ranging and widely referenced discussion of what we do when we code see Chapter 4, 'Working in Analytic Mode', in Ely et al.

(1997). Few texts explain how to code using manual methods, the notable exceptions being Coffey and Atkinson (1996), Miles and Huberman (1994) and the texts of grounded theory (see previous chapter). See also Flick (1998), Mason (2002) and Seale (1999). Detailed discussion of coding on computer is in Lewins and Silver (2007). For extended coverage of coding using particular software, see Auerbach and Silverstein (2003), Bazeley (2007) and Corbin (2007). A web search for 'qualitative coding' will find instructions and discussions for particular projects, often giving vivid insight into the processes involved.

Coding of qualitative data that are to be combined with quantitative analysis requires attention to compatibility issues: see Bazeley (2002).