# 3

# CODING FOUNDATIONS

The excellence of the research rests in large part on the excellence of coding. (Strauss, 1987: 27)

# SECTION 3.1: INTRODUCTION TO CODING FOUNDATIONS

This chapter provides practical strategies that will help you identify meaningful labels for your codes and to begin coding your data. Coding can be labour-intensive but should not be approached as a hurdle you have to get through to eventually arrive at your analysis; it is part of an interpretive process that might shift over time and requires periodic use of other strategies. At first, your progress in working with the data might be slow. However, your project will gradually grow into a web of data, themes, and thinking, illuminating (and sometimes refining) your research question(s). As your ideas develop, coding the data is likely to become more efficient than it was when you started.

## In this chapter you will

- Learn how coding can support your analysis.
- Discover the specific strategies you can use to identify and name concepts and themes in your data.
- Learn some practical strategies for managing your coding process.
- Find out where and how NVivo stores your Nodes.
- Learn how to approach the basics of coding in NVivo and viewing what you coded (e.g., coding, uncoding, Node Descriptions, using Coding Stripes and Highlighting).
- Learn how to manage your Nodes (e.g., deleting, merging, customizing your Node list, exporting a list of Nodes).
- Learn how Memos and See Also Links can help you think about your coding process and initial interpretations.

## Other chapters with related materials

While you might find all the material related to your current needs in this chapter, you should also consider these other chapters that contain closely related information:

### Chapter 1: Where to begin?

In many ways, Memos and Links are companion tools to coding. Instead of pulling similar themes or concepts together through coding, Memos and Links help keep track of ongoing ideas and threads of connections in your data and this is often where a good portion of your analytical work can be developed and tracked. Be sure to use Memos and Links in conjunction with coding.

### Chapter 4: Advanced coding

After you master the basics of coding, you will move on to more advanced skills such as creating hierarchies, aggregating, and Auto Coding. You will also want to explore relationships among Nodes with visual Maps. You will likely want to create Node hierarchies while being mindful to use 'vista' Node structures and avoid 'viral' ones. Get acquainted with using Queries to aid coding by running a Word Frequency Query and a Text Search Query and consider turning these Query Results into Nodes.

## SECTION 3.2: CONCEPTUAL GROUNDING IN CODING

In this section of the chapter, we identify ways of thinking about your coding regardless of whether or not you use software. If you want to jump to the instructions on clicking, go to Section 3.3 (Coding in NVivo). The codes, themselves, are often referred to by qualitative researchers (regardless of whether they use software or not) as issues, themes, topics, etc., and although these labels carry nuances across different methodologies, we use them here as synonyms to help orient you to the process. Another term that resonates with most newcomers to this adventure is the verb, 'tagging'. While tagging or coding the data is often an important strategy for researchers, qualitative analysis is about working intensively with rich and often unstructured data with additional strategies that are related to – but often identified as distinct from – coding. These include annotating, associating, discussing, exploring, linking, memoing, organizing, reading, reflecting, suggesting, transforming, and visualizing.

### Goals for early work with data

As you begin coding, it is important to observe what is going on in the data while also paying attention to how you are reacting to the process of coding. By engaging in this self-observation you will become better at coding and you might notice that you encounter the following:

- *Excitement:* As you code data, even though you may have collected the information and transcribed it yourself, new codes and a new awareness of the relationship among your codes might emerge unexpectedly. As this happens, you will become more aware of what you can contribute to the field. Notice when this happens and write about it! (See memoing and linking in Chapter 1.)
- *Changing your mind:* Coding evolves as you handle the data over time. You might encounter compelling arguments or perspectives in the literature or among your colleagues that result in an alternative framework or definition of a concept. Notice when this happens and write about it!
- *Frustration:* You might change your mind once, and then change it back again; your timeline could get thrown by unforeseen circumstances; a brilliantly complicated coding structure that took time to develop might need to be simplified to better answer

your research questions. There are many things that can trigger frustration around the lack of predictable, fluid movement through the coding process. Notice when this happens and write about it!

- *Adjusting your gaze:* You might dive into nuances in the data to get close to the perspectives of your participants, but then occasionally see a need to step back and look at the bigger picture and how the data relates to a larger context. Moving back and forth between closeness and distance is an important part of making sense of the data and understanding your codes, although there are few prescriptions of when or how to do so. Notice when this happens and write about it!
- *Building ideas:* Strive, even from this early stage, to go beyond descriptive labelling and to think about codes beyond their relationship to a single File. Why is this information important? Where will these ideas take me? This will be reflected in the way you name codes. Notice when this happens and write about it!
- *Jumping to conclusions:* Early work with text and concepts is about laying the foundation and identifying key themes in the data. But, beware of making judgements or decisions too early. Constantly challenge your first ideas by looking for contrasts as well as comparisons, by purposively sampling diverse cases, or by reviewing what the various camps in the literature say on the topic. Notice when this happens and write about it!
- *Observing constellations of codes:* Right from the start, it is helpful to observe when you are applying multiple codes to the same content (e.g., the same portion of a transcript). Early in the coding process you should not assume these co-occurrences are correlations or indicate causation, but they could help you map constellations of concepts that facilitate your construction of theories later. Notice when this happens and write about it!

If you lay a sound foundation with your first File(s), then you will confidently move on to adopt further strategies for advancing your thinking about data, as outlined in the following chapters. For now, however, we will help guide you through the basic steps as you prepare to code.

## Selecting Files (we recommend starting with text documents)

If you are just beginning to gather data, selecting a text document to work on first is probably easy because you will have only one or two to choose from. If, however, you have already completed a number of interviews or have a variety of data items, then you might identify two Files:

- Choose one which you remember as being typical in some way, or which was contributed by someone who was representative in some way of a group or subgroup in the sample.
- Choose one which seemed to be particularly interesting or rich in its detail.

These first two Files can have a significant influence on the early stages of coding. When choosing a second one, therefore, you will benefit from focusing on one that contrasts in

some important way with the first in order to maximize the potential for variety in concepts (or in their forms of expression).

### Two, contrasting documents to code

In the *Researchers* Project, Pat chose *Frank* and *Elizabeth* as the first two documents to work through in detail, because, as academic researchers, they provided a strong contrast in terms of career development. *Elizabeth*'s path into a research career was characterized by digressions and serendipitous events, while *Frank*'s path was direct and purposeful.

When Pat first worked through *Elizabeth*'s document, the impact of her changing image of research was striking, and so she coded that and further documents for the way in which the speaker viewed research – only to find in later analyses that it had no particular significance for anyone else. It became more useful to see (and code) *Elizabeth*'s re-visioning of research as a turning point, rather than focusing on her image of research. The codes that dealt with images of research could then be retired.

Exactly how and what you code will most certainly depend on your choice of methodology, so it is important that you familiarize yourself with the literature surrounding your methodology, including the various camps and debates therein. Coding is also influenced by your community of practice (Lave & Wenger, 1991), your discipline, your participants, your personality, and any prior experience you have in handling qualitative data. We will not cover all of these nuances although we want you to be aware of them as we provide guidance in these early stages of coding. Furthermore, we advise you to consider writing about these influences on your analytical processes.

## Codes and coding

'A code is an abstract representation of an object or phenomenon' (Corbin & Strauss, 2008: 66) and it 'is most often a word or a short phrase that symbolically assigns a summative, salient, essence-capturing and/or evocative attribute for a portion of language-based or visual data' (Saldaña, 2013: 3; 2016: 4). More prosaically, Bernard and Ryan (2010) describe coding as a way of identifying themes in a text. Codes range from being purely descriptive (this event occurred in the *playground*) to more conceptual topics or themes (this is about *violence between children*) to more interpretive or analytical concepts (this is a reflection of *cultural stereotyping*) (Richards, 2009). Furthermore, all of these might be applied to the same text.

Field notes and verbatim transcripts reflect 'the undigested complexity of reality' (Patton, 2002: 463), needing coding to make sense of them, and to bring order out of chaos. Coding in qualitative research, in its simplest sense, is a way of indexing the data in order to facilitate later retrieval as singular concepts (e.g., *cultural stereotyping*) or constellations of concepts

(e.g., *cultural stereotyping*, *employee guidelines*, and *retention*). Naming a concept or topic aids organization of data and so assists analytic thinking. As data are seen afresh through the code rather than the original document, coding allows you to 'recontextualize' your data (Tesch, 1990), assisting you to move from individual document analysis to theorizing, all the while retaining access to the original material.

Because the codes we create are inextricably connected to our world-views, 'implicitly or explicitly, they embody the assumptions underlying the analysis' (MacQueen, McLellan-Lemal, Bartholow, & Milstein, 2008: 119). In other words, codes do not magically 'emerge' from the data, but are always generated by the researcher through a range of strategies, such as the constant comparative method (Glaser & Strauss, 1967). Some methodologies (e.g., autoethnography) push researchers to engage with and understand their personal roles in producing and making sense of data, while others (e.g., hypothesis testing) de-emphasize such an examination. Regardless, there can never be a complete separation between codes and researchers, as they are engaged in an iterative process of mutual construction.

## Approaches to coding

Coding is not a mechanical task. It requires continual assessment and rethinking and is part of analysis. It is a mistake to tenaciously push through coding in the hope that something interesting will tumble out of the Project when you are done. One way of understanding basic approaches to coding is to observe that there are 'splitters' – those who maximize differences between text passages, looking for fine-grained themes – and 'lumpers' – those who minimize them, looking for overarching themes (Bernard & Ryan, 2010), although most of us do some of each. A common approach is to start with general themes, then code in more detail (e.g., Coffey & Atkinson, 1996). However, those who employ grounded theory, phenomenology, or discourse analysis start more often with detailed analysis and work up to broader concepts.

### Sorting the laundry and sorting codes

Lynn Kemp, Director of the Translational Research and Social Innovation Group at Western Sydney University, tells students that choices about coding are like choices in sorting the laundry. Some hang the clothes just as they come out of the basket, and, when dry, throw them into piles for each person in the family, ready for further sorting and putting away – preferably by their owner! Others hang the clothes in clusters according to their owner, so they are already person-sorted as they come off the line, although pants and shirts might be mixed up and socks still need pairing. And yet others hang socks in pairs and place other similar things together, so they can be folded in batches as they come off the line. Ultimately, the wash is sorted, people get their clothes and (hopefully) all the socks have pairs. Similarly, whether you start big and then attend to detail, or start small and then combine or group, your coding will eventually reach the level required.

### Broad-brush or 'bucket' coding

This one is for lumpers! There is no need to treat coding as unchangeable and you will be able to code from broader themes to more refined subthemes. If you begin with the broad-brush approach, your initial coding task is to 'chunk' the text into more general topic areas, as a first step to seeing what is there, or to identify just those passages that will be relevant to your investigation. Beginning by coding to these broad themes does not mean they should be vague. Some researchers strategically begin here because they want to take one pass through the data without getting bogged down with a large volume of codes and they know in advance that they will be taking a second pass through each code to consider the work already done. During the second pass, researchers often take stock of the diversity of opinions in each code, the volume of data and the relative importance participants assign to them while simultaneously coding to more discrete subcodes. If you use this approach, notice that you are beginning with a deductive stance, because you hope that the answers to your research questions can be developed out of some fairly broad themes that you are applying to the data. In Chapter 11 we elaborate on the efficacy of the broad-brush approach for teams.

## Selecting from broad themes for greater focus

Lynn Kemp's doctoral study of the community service needs of people with spinal injuries employed broad-brush coding as an initial sorting strategy. In response to her question, 'You came home from hospital and …?' her interviewees talked extensively across all areas of their lives. In order to manage this large pool of data, Lynn coded her interviews first into very broad themes (e.g., *community services*, *employment*, *education*, *recreation*). She then coded on from the community services text (which was her immediate focus), capturing the detail of what people with spinal injury were seeking from life, what services were being offered, and how these supported or impeded their clients' capacity to fulfil their 'plan of life'. After recovering from the doctoral process, Lynn (or indeed, her students) could then focus attention on topics that were set aside, to engage in further analysis and reporting.

### Coding detail

And this one is for the splitters! For some methods, most notably grounded theory (Glaser & Strauss, 1967), your initial analysis will involve detailed, slow, reflective exploration of early texts – doing line-by-line coding, reading between the lines, identifying concepts, and thinking about the possible meanings as a way of breaking open the text. You record what you learn via your work with your codes, supplemented by the notes you write along the way. At the beginning of the analytical process, you will explore each word or phrase for meaning, perhaps considering theoretically the difference it might have made if an alternative word had been used or a contrasting situation described (Strauss, 1987). You will not continue to code at this level of intensity beyond the first few files, however, unless you come upon new ideas to explore or

contradictions to resolve. In practical terms, capturing the detail of the text does not mean you should segment it into tiny, meaningless chunks. Rather, the goal is to capture the finer nuances of meaning lying within the text, coding a long enough passage in each instance to provide sufficient context without clouding the integrity of the coded passage by inclusion of text with a different meaning (unless, of course, the whole passage contains contradictory messages).

## Implications and nuances within the interview

In the *Researchers* Project, *Frank* begins his response to an open question about his research journey as follows:

> My PhD was both a theoretical and empirical piece of work; I was using techniques which were novel at the time. (Interruption by secretary.) I was using novel mathematical dynamic techniques and theory and also I was testing these models out econometrically on cross-country data. I think it was a strong PhD, I had a strong supervisor, he was regarded as – it is fair to say he would have been in the top 5 in his area, say, in the world.

In the first place, *Frank* begins his response by focusing on his PhD experience. In broad terms, one could simply code this passage descriptively as being about the PhD or learning phase of becoming a researcher, and indeed, that is relevant as contextual (or structural) coding even if one is looking to capture detail in the text. But this passage is also saying considerably more than just that *Frank*'s research career included a PhD student phase. He implies that the PhD provided a strong foundation for his later career. Linked with that strong foundation are the kinds of work he did in his PhD and the role of his supervisor.

This text tells us also a great deal about *Frank* and his approach to research. His PhD was both theoretical and empirical (characteristics which are repeated in the next sentence). Not only does he have credibility in both these aspects of his discipline, but his work was novel. Here, he is both validating and emphasizing the strength of his foundational work. In describing his work as using novel mathematical techniques, he is also suggesting that he is making a mark on the development of the discipline – presaging a later, critical theme.

That his PhD was a 'piece of work' also suggests wholeness or completeness. So, research work can have the dimension of being partial, incomplete and ongoing, or of being finished and completed with loose ends tidied up. This, along with the dimension of research discoveries as occurring at a point in time versus occurring in incremental developments, could be interesting in relation to the nature of research activity (and being a researcher), but they are not necessarily relevant to the current question of how one becomes a researcher – and so they are noted but not coded.

*Frank*'s emphasis on the strength of his PhD (noting repeat use of 'strong') and the strength of his supervisor supports an idea also evident in the first sentence that the status of his work matters; It is important to him that his supervisor was at the top, and Frank was moving up there too, with leading-edge work. This theme of ambition (incorporating status) is also evident in several other passages, for example:

> I then teamed up quite soon with another 'young turk' … 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 just really went for it. … our two names became synonymous with this certain approach, and we went around Britain and into Europe giving research papers and getting ourselves published. … 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.

How important is this level of ambition, both from the point of view of the research question, and for detailed coding? At this early stage, the safe move is to create a Node for *ambition*: if no one else talks in these terms, then later it can be combined within a broader category of, say, *drive* or *commitment*.

Coding for the opening passages might therefore look as shown in Figure 3.1. Other ideas and reflections prompted by the passages were recorded in annotations and in the memo attached to *Frank*'s document.

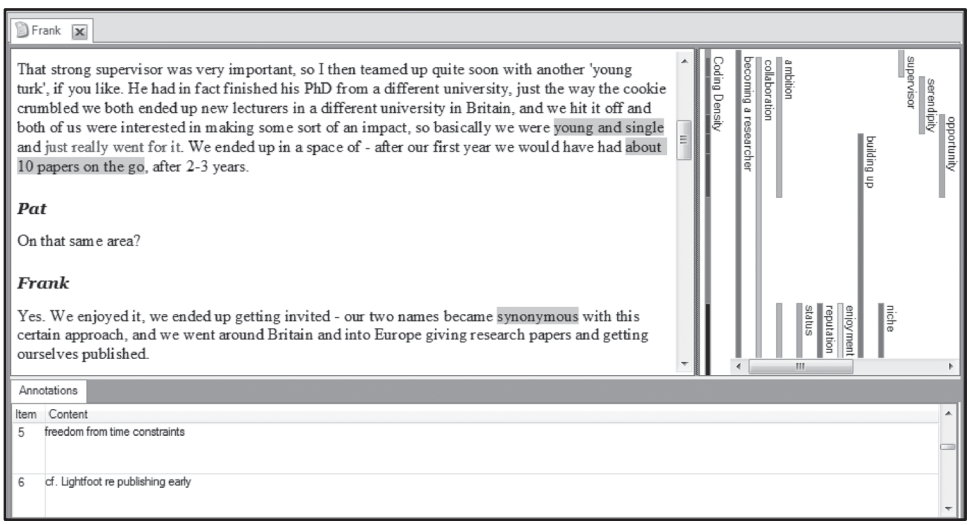


**Figure 3.1** *Frank*'s document showing application of multiple codes to each passage of text, and Annotations (see Chapter 1 for Annotations)

## Strategies for identifying and naming codes

The idea of coding a portion of text (or audio, video, images, etc.) to index its content sounds simple enough, and observing someone else's coding can make the task look deceptively easy. When you meet your own data, however, you find many things are going on at once: something is happening in a particular setting or at a particular time, particular people or groups are involved, perhaps their responses are based on their belief systems or cultural background, and there are consequences to be considered. Perhaps there is a twist to the way this experience, belief, or feeling is being reported that makes it just a bit different from other reports or makes it difficult to pin down. Narrative is inherently complex: a group can argue about

the content and meaning of just one paragraph for a very long time. In your NVivo Project, it tends to help when you code each passage to multiple codes that are all relevant (e.g., *Trust*, *Friends*, *Advice*), rather than create an overly nuanced code that attempts to capture all of them at once (e.g., *Trusted friends are sought for advice*). By using the former approach, you can later check whether this combination of codes forms a pattern, or how *Advice* AND *Trust* AND *Friends* occur in different (or similar) ways than *Advice* AND *Trust* AND *Family*.

What follows is a collection of ways in which codes might be identified and named, particularly those requiring more than a simple descriptive label. Further suggestions and extended examples of coding strategies can be found later in this chapter.

- *Look for repetitions and regularities:* People repeat ideas that are of significance for them (see the use of 'strong' by *Frank* in the example above), sometimes across a range of experiences. Identifying these repetitions can be the first step in understanding the contexts in which they are used.
- *Use questions of the text to generate codes:* Who, what, when, why, how, how much, what for, what if, or with what consequences? Asking these kinds of questions will help to ensure thoroughness of coding and to develop relational statements (do the actions or strategies change under different conditions and, if so, what are the implications?) and so will hugely benefit development of a theoretical model.
- *Compare and contrast passages of text:* Think about the ways in which they are both similar and different. This will help you go beyond simply deciding which chunks might go together: it will help you discern the dimensions within concepts, or perhaps to discern previously unobserved variables running through the text.

## Gender differences and the role of parents in relationship choices

In a study of parental and peer attachment among Malaysian adolescents, Noriah Mohd Ishak, in the Faculty of Education at the University of Kebangsaan, Malaysia, found boys and girls spoke in contrasting ways about their parents' role in their relationship choices, for example:

Boy: I trust my parents with their choice; I might have a girlfriend here in America, but the girl I finally marry will depend on my parents' choice. I think theirs is always the right choice.

Girl: I choose whom I will marry. My parents might have their own choice, but how can I trust their choice, because they live in a different generation than I do!

In relation to young people's acceptance of parental leadership, then, trust is a significant dimension. Other possibly relevant concepts arising from these comparative passages are parental adaptability, parental authority, cultural expectations, and cultural change. Also of interest in the boy's comment about having an American girlfriend is what it says about attitudes toward women and commitment in relationships.

- *Compare the data with hypothetical or extreme examples:* This helps to identify dimensions within concepts.

## Obsession with research

*As his career was unfolding,* Frank's *total focus was on research:*

> When you go out in the pub you are talking research, when you go to bed at night you are thinking it.

Seeing this as a form of addiction prompted Pat to talk with a colleague who was studying youth gambling behaviour. Researchers and gamblers can both follow a passion, often (but not necessarily) to the detriment of their health and/or family relationships; both can engage in activities that benefit from careful strategic planning but which are also subject to luck or whim; both can be characterized by emotional highs and lows; both can experience problems with self-regulation. There were differences of opinion over whether addiction was necessarily 'bad'. As the project developed further, Pat saw addiction as one expression of a larger category of obsession, which she came to define as 'driven passion' to capture the dimensions of emotional engagement and the potentially blind (unregulated) commitment of the obsessed researcher.

- *A priori, or theoretically derived, codes:* These come from your prior reading and theoretical understanding to give you a start list of concepts to work with. Use, also, the categories embedded in your research questions. Don't let your a priori codes confine your thinking, however. Strauss (1987) and Patton (2002) each referred to these as sensitizing concepts, rather than as fixed ones.
- *In vivo, or indigenous codes:* These are different from a priori codes because they are derived directly from the data (Strauss, 1987). They capture an actual expression of an interviewee or label used by participants as the title for a code, but they can sometimes have the unfortunate problem of not being useful in exactly that form for expressing what is learned from other participants. Change the term to a more general construct as the project develops but keep a record (in the description or a note) about how the code arose in the first place.

## The 'hard labour' of going through change

When a member of a corporation that underwent a radical change programme talked about the hard labour of working through that change process, this expression was appreciated for its valuable imagery – of what goes on in a birthing process, of the punishing work of someone held in a prison – and hence became an in vivo code available for use in coding other text.

- *Explore discourse that reflects a particular construction of the topic, or of society:* In the *Researchers* data, you will find discourses of performance (being successful, competing, building a reputation), of play (following curious leads, puzzling, playing with new ways of doing things), and of romance (passion, sensuality, orgasmic) in the way participants talk about their research worlds. Coding these varying constructions provides insight into participants' motivations and approaches to navigating their research context. The field of discourse analysis encompasses a wide range of specialties and perspectives, each with implications for choice of codes. For example:
    - Critical discourse analysis: A focus on the reproduction of social power through language use (Fairclough, 2014) might result in the creation of codes for *formal or mature titles* (e.g., 'sir' or 'men') versus *informal or immature titles* (e.g., 'gals' or 'girls').
    - Rhetoric: Paying attention to the mechanisms and processes of persuasion (Anzaldúa, 2015) could entail codes that track the use of discursive strategies like 'definitively', 'certainly', 'obviously' versus 'perhaps', 'potentially', 'maybe'.

- *Narrative structure and mechanisms take account of how things were said as well as what was said:* You might note, for example:
    - transitions and turning points in the narrative, signifying a change of theme or a subject to be avoided (Duchinsky, 2013);
    - inconsistencies, endings, omissions, repetitions and silences (Poirier & Ayres, 1997);
    - repairs, hedging language, chronology (or lack thereof) (Fox, Hayashi, & Jasperson, 1996);
    - the use of metaphors and analogies (Lakoff & Johnson, 2003);
    - the sequenced, structural elements of a narrative (Elliott, 2005; Riessman, 2008); and use of articles or pronouns pointing to particularized or generalized referents, for example, 'the staff', 'my staff', or 'our staff'; indicating level and type of ownership or involvement (Morse, 1999).

## 'We' and 'ours' versus 'they' and 'them'

In an evaluation research project on a Violence Prevention Initiative, Kristi worked on a team where the final rounds of data collection and coding turned to a focus on how pronouns were used by community collaboration partners in 26 communities over the course of the Initiative. Prior rounds of analysis facilitated the development of codes that were descriptive (e.g., *special events* and *definition of violence*), thematic (e.g., *learning* and *conflict*), and interpretive (e.g., *critical events* and *cultural barriers*). However, the final round focused on an examination of the use of pronouns that were inclusive (e.g., *we* and *ours*) versus pronouns that indicated separations (e.g., *they* and *them*). The careful examination of pronouns in the context of the prior coding helped explore how well and in what ways the partners shifted to collaboration over time.

## Generating conceptual codes

> 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. … [T]he 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)

At first, you are not quite sure what is relevant, or how it will be relevant. When coding, it is very easy to be beguiled by fascinating things your participants have said and so to become sidetracked. To start, try the following three steps Lyn Richards (2009) developed for undergraduate teaching. They will help you move from 'seeing' to 'seeing as':

- *Identify:* What's interesting? Highlight the passage.
- *Ask:* Why is it interesting? This can generate a useful descriptive code or perhaps an interpretive code. It could also warrant a note to help remind you of the significance later.
- *Explain:* Why am I interested in that? This will 'lift you off the page' to generate a more abstract and generally applicable concept, which, if relevant to your project, will be very worthy of a code (and perhaps a memo).

That critical third question is giving you a way of generating concepts that will be useful across documents rather than for only one or two passages in a single document. These more general or abstract concepts are essential for moving from description to analysis and link the data to the broader field of knowledge. In a strategic sense, that third question also helps keep you on target, to keep the focus on issues relevant to your research questions.

### Developing more abstract categories

In a doctoral project on the role of support groups in assisting young people with a mental health problem, a participant reported moving from sheltered to independent accommodation. The student deemed this to be of interest and created the Node 'accommodation' to code it. Just one participant, however, had anything to say about accommodation, and accommodation was not an issue of concern for this project. (Had the project been about issues faced by young people with mental health problems, then of course accommodation could well have been a relevant code to make.) When the student was challenged about why the text was interesting, she said it was the evidence of improvement in mental health status indicated by the young person's change in accommodation. With the code name changed to reflect this more pertinent (and more interpretive) concept, she could use this code across other documents to code other indicators spoken of in the same way, such as gaining employment or repairing a relationship. If she wished to examine the nature of the evidence given to indicate improvement in mental health status, she could simply review the text coded at the Node.

You might expect to engage in periodic rereading of earlier material, when the salience of particular texts becomes more obvious. In any case, it remains likely that you will create codes which you later drop if you heed the advice of MacQueen et al. who say, 'Don't clutter the codebook with deadwood' (2008: 133). In addition, you will occasionally miss relevant passages, but that should not be a major concern. Firstly, important ideas will be repeated throughout the data; secondly, it is likely you will pick up on missed instances of something as you review other codes, or later as you are querying your data. Above all else, keep in mind that you can change your mind about relevant concepts and even about your research questions (depending on your methodology). Fortunately, NVivo accommodates such changes.

## Time and timing

### How long will it take?

People often ask, but it is very difficult to say how much time is needed for coding. Our best estimate is that, once you have an established coding system, you should allow at least 3 hours per hour of transcript – the actual amount will very much depend, however, on your research questions, methodological approach and the quality of the collected data. Experienced researchers (including Miles & Huberman, 1994) routinely recommend allowing a working period for analysis of data that is two to five times as long as the period taken to make arrangements and gather the data.

### When do I stop?

There are two ways you should consider walking away from the coding. The first pertains to a temporary but purposeful break that gives you time to rethink, regroup, and come back more focused. Kristi sometimes warns researchers that if they are becoming obsessed with the movement of the Scroll Bar down the right side of the screen and calculating the minutes left to code the File, it is likely time to stop. Lyn Richards commented in an online forum that: 'If you've been coding for more than two hours without stopping to write a memo, you've lost the plot.' She recommended regularly stepping out of coding, not only to write memos, but also to monitor and review the coding. Freely make new codes (rather than sweat over each one as you create it), but also periodically do a check to ensure the categories you are making are relevant. Occasionally review one or two codes in depth for a useful change of perspective on your data.

The second way to reconsider or step back from coding might be more permanent or final. If your project is one in which it is essential to code all materials in order to thoroughly test hunches/hypotheses or because counting the frequency of occurrence of codes (e.g., issues raised for a stratified sample) is part of the research strategy, you might need to persist until the task is completed. However, if you have worked sufficiently and thoroughly with enough texts to be able to generate convincing answers to your questions (or to develop your explanatory theory), it may be time to stop. Consider this if coding is becoming routine and fails to generate new themes or ideas. If you have additional data, consider reviewing these materials, just to check if any include comments or images which extend or contradict the model or

theory or explanation you have generated thus far. Alternatively, if you have additional data transcribed, then import them into a separate folder for *non-coded documents* and use Word Frequency or Text Search Queries to identify text that contains especially relevant terms from within those files (see Automatic coding in Chapter 4).

## SECTION 3.3: CODING IN NVIVO

NVivo has two predefined types of Codes: Nodes and Relationships (and a third, Sentiment, in NVivo Plus). In the *Navigation View* under the **Code** button you will also see Relationship Types. This is a classifying tool rather than a kind of Code and we discuss classifying tools in Chapters 5 and 6. The vast majority of NVivo users spend their time working in the Nodes area, and this will be our focus in the remainder of this chapter. Despite NVivo's flexibility in helping you apply the strategies from our earlier ideas about coding, knowing how to code in theory and making it work effectively and efficiently for your research can be two quite different things. In this section we turn to practical issues about how to apply the ideas above to the mechanics of the software, and we introduce a few tactical and practical issues that are related to coding in your NVivo Project.

### NVivo terms used in this section

Codes

Codebook

Coding Reference (or Reference)

Coding Stripes

Highlight

Node

Node Description

### Storing coding in Nodes

In many respects, coding in NVivo is much like the way some qualitative researchers use Post-it notes and whiteboards to develop ideas (although a digital Project is much more versatile than Post-it notes and highlighters). Because you will be doing this work in a digital Project, Files will not lose their coding when you move, split, or merge Nodes, nor will Files lose their coding if the text is modified (unless you delete all of the text instead of modifying part of it). The bottom line with coding in NVivo is that the software enables you to build on work already done and will allow you to change your mind about everything related to coding (the name of a Node, its Description, colour, location in the coding hierarchy, data coded [or uncoded], etc.).

In NVivo, you make a Node for each topic or concept to be stored, much like designating a hanging file for the cut-up photocopies in a manual system. As you code, the system interface will mimic the manual method (as if you were using paper and making copies to put them in a folder). Note the direction of the arrow in Figure 3.2.

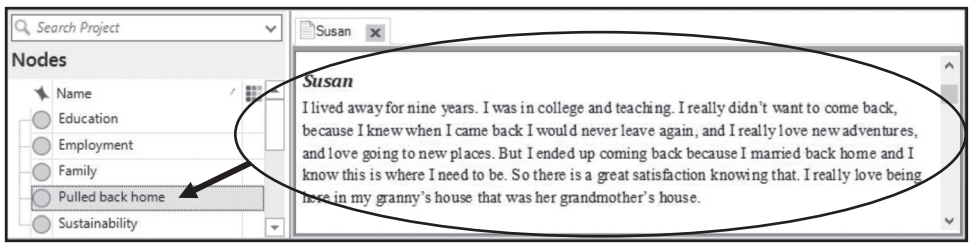


**Figure 3.2** NVivo mimicking the conventional strategy of coding: making copies and placing them in categories/concepts

What NVivo keeps in a Node, however, are not actual segments of data, but pointers (references) to the exact location of the text (or audio, video, etc.) you coded in the File. Note the more accurate depiction of the direction of the arrow in Figure 3.3.

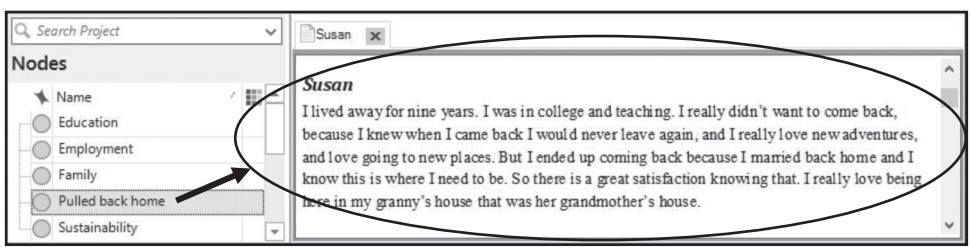


**Figure 3.3** NVivo actually tagging the text with the Node

Coded passages are never copied, cut, or physically moved into the Nodes. Think of a Node as though the only thing it contains is a command that says, 'go get it!'; when you open a Node you are looking at the original text in the File. These coded passages can be short or long, according to your assessment of a meaningful range. Regardless of their size, in NVivo these passages are called References (also sometimes Coding References). A Node can point to as many References across your documents as you would like, and more than one Node can point to the same Reference, since some passages will convey multiple meanings. Because the Node is a pointer to the Reference:

- the File always remains intact;
- information about the File and location of a quote is always preserved;
- it is always possible to view the Coding Reference in its original context (because the Node is pointing to the File);

- changes to the File, such as edits, are immediately reflected in the Nodes (because the Nodes are just pointing to the File); and
- text (audio, video, pictures, etc.) can be coded at multiple Nodes (which will become especially important later when we show you how the Queries in NVivo can help you find intersections [and other patterns] among Nodes).

At first, you will probably use Nodes that do not presume any relationships or connections to each other. They serve simply as tags for data about ideas you want to track. At this stage we recommend you focus on creating Nodes without trying to sort them into different levels – it is easy to reorganize them later when you have a better understanding of the data. In the meantime, they will be listed alphabetically by default (and we will show you how to custom sort your list of Nodes later).

## Creating and describing Nodes

If you do not yet have any Files in your Project (or have not even created a Project yet), see Chapter 1 (pp. 18–19) for instructions. You can create some Nodes in advance if you are using an existing framework or theory or if you are so familiar with your data that you already know some of the Nodes you want to create. (You can use many of these strategies for creating a Node while you read the data as well.) Remember each Node should only refer to one theme or concept. If you need to capture two or more elements of what is happening, use two or more Nodes to do so, applied to the same text (and NVivo will help you find the intersections later).

To follow the subsequent instructions, remember the four main areas on your screen (*Ribbon*, *Navigation View*, *List View*, *Detail View*) from Chapter 1 (Windows: Figure 1.1; Mac: Figure 1.2), as each instruction begins with one of these four locations.

## Guided tour of Nodes and coding

One of the challenges in learning NVivo is there is more than one way of achieving most tasks like creating a Node or adding coding, and it takes experience to know which most suits your purpose at a particular time. We will begin with a guided tour with some activities within a File as you code and then we will walk you through some activities from inside a Node. After you are familiarized with these most common pathways through the Project, we provide alternative ways to create Nodes, code data and uncode data. After this tour, you might jump to Chapters 4 and 7 if you are interested in the ways NVivo can use Queries and Auto Coding to create Nodes, but if you intend to use these tools, you will first need to understand the basic skills we will teach you in this chapter.

### Node Names and Descriptions

The two main Properties of a Node are the Name and the Node Description. Node Descriptions serve as accessible reminders about the basic meaning of the Node and they are very useful when:

- there are multiple people working on the same project;
- other responsibilities have taken you away from this project for a time;
- you need to investigate if you have multiple Nodes with similar meanings (in which case you can combine the Nodes later).

These Descriptions essentially serve as your Codebook, can be modified as your data informs your Description, and can be exported later as a reference or appendix.

## 3.a.

### Getting ready to code

⇒ *Navigation View*: **Data** › **Files**.
⇒ *List View*: **Double-click** on the **File** so it opens.
⇒ *Navigation View*: **Codes** › **Nodes**.

### Creating a Node from the *Ribbon*

⇒ *Ribbon*: **Create** › **Node** › (Figure 3.4) provide a **Name** and **Description**.
⇒ We recommend skipping the Nickname and you can always add it later.
⇒ We discuss Aggregate in Chapter 4 (pp. 120–122).
⇒ If you wish, you can assign a colour to the Node in order to sort Nodes or view these colours in the Coding Stripes (both of which we discuss later).
⇒ **OK** / **Done.**

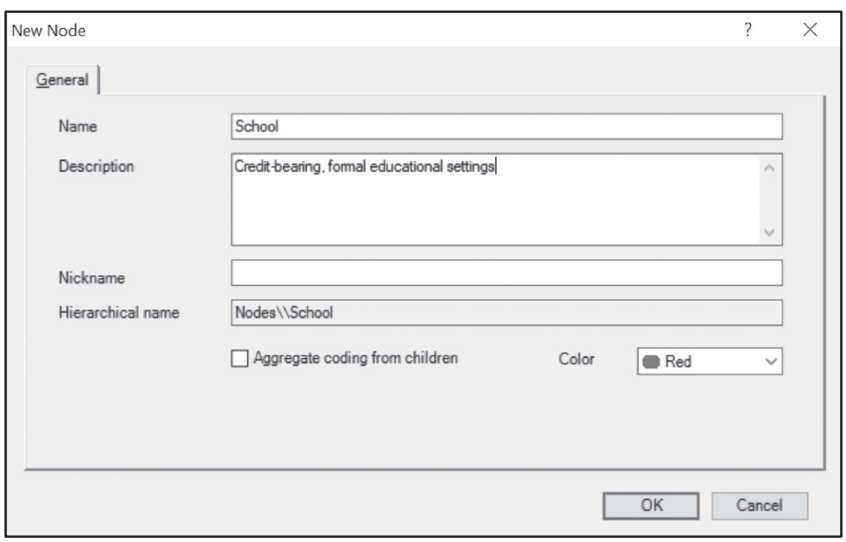


**Figure 3.4** The Node Properties window

### Creating a Node with a Right-click

Mac users will find that this option only works until the *List View* fills with Nodes. At that point you should use another tactic.

⇒ *List View*: In the white space (below the Node you just created) **Right-click** › **New Node / New top level Node** › provide a **Name** and a **Description**.

*Fill in additional information as described in 'Creating a Node from the* Ribbon.'

## Coding to your existing Nodes

⇒ *Detail View*: Select the text to be coded › Drag selected text to a **Node** (Figure 3.5).
⇒ A brief pop-up message in the *Navigation View* (next to Nodes) confirms your action.

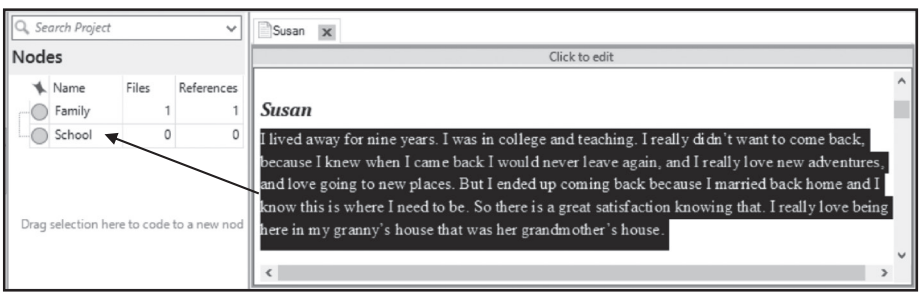


**Figure 3.5** Coding to an existing Node via drag-and-drop

## Creating a Node and simultaneously coding (Windows only)

⇒ *Detail View*: Highlight the text you want to code › Drag the text to the empty space in the *List View* that says, 'Drag selection here to code to a new node' (Figure 3.6) › **Name** the Node, provide a **Description** › **OK**.

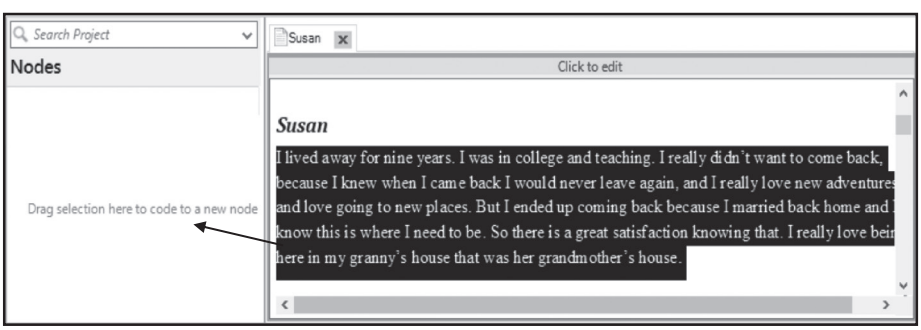


**Figure 3.6** Creating a Node and coding at the same time

## Seeing what you coded

⇒ *List View*: **Double-click** on the **Node** and the **Reference** will be displayed in *Detail View*.
⇒ *Detail View*: Each time you code, additional References (across Files) will stack in this window (alphabetically by Folder name and then File name).

*(Continued)*

(Continued)

⇒ Select the **X** on the tab you just opened to close it (Figure 3.7) but leave the File open.
⇒ *Navigation View:* Open Items list (Chapter 1, Figure 1.8) › Select the **X** next to the Node to close it.

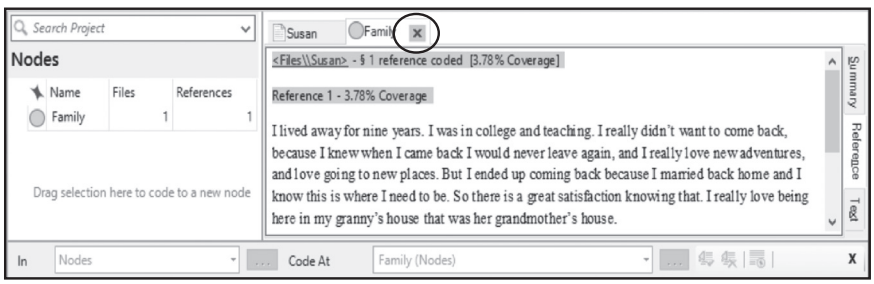


**Figure 3.7** Coding for *Family* displayed by the Node

## Changing a Node Description (or name, colour, etc.)

⇒ *List View*: **Right-click** on the **Node** › **Node Properties** / **Get Info** › change the Description (or colour, etc.) › **OK** / **Done**.

## Uncoding from the File

⇒ *Detail View*: Select text in the **File** to be uncoded › **Right-click** and select one of the following two options:
- **Uncode** / **Uncode Selection** › **At Existing Nodes or Cases** › tick the desired boxes › **OK** / **Select;** or
- **Uncode from Recent Nodes** › select the desired **Node** (Figure 3.8).

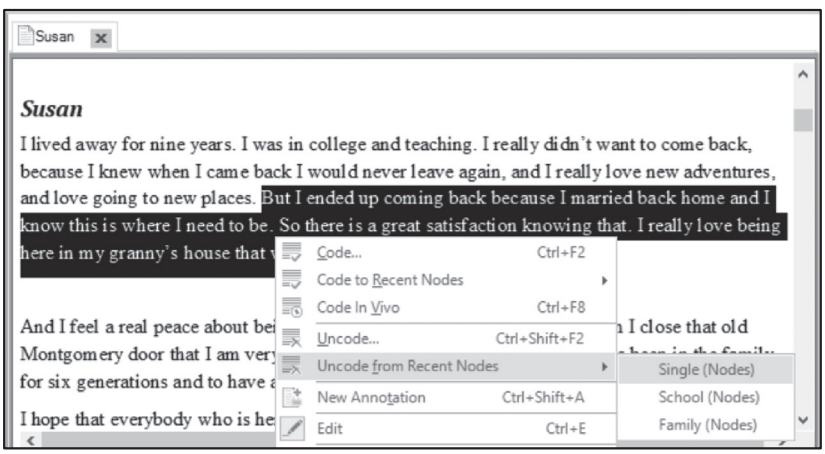


**Figure 3.8** Uncoding from the File with a right-click

## Re-examining your coding

As you proceed with your coding, you might want to reconsider the data you coded or to look at it in a slightly different way for a variety of reasons:

- You are still getting used to the software and want some reassurance your coding is really happening.
- You have been interrupted in your work and want to check where you stopped.
- You want to check if you picked up all important aspects for this passage.
- You want to visually show your coding on a passage to a colleague, teammate, or instructor.

You can achieve these various goals in several ways while you are working with your Files.

## 3.b.

### Opening and managing the Coding Stripes on a File

⇒ *Detail View*: Click anywhere in the **File** so NVivo knows you are active in the *Detail View*.
⇒ *Ribbon*: **Document** / **View** › **Coding Stripes** › **Recent Coding** / **Nodes Recently Coding**.
⇒ As you scroll up or down with the **Scroll Bar** (between the text and the stripes), you see the stripes are anchored adjacent to the text you coded (Figure 3.9).

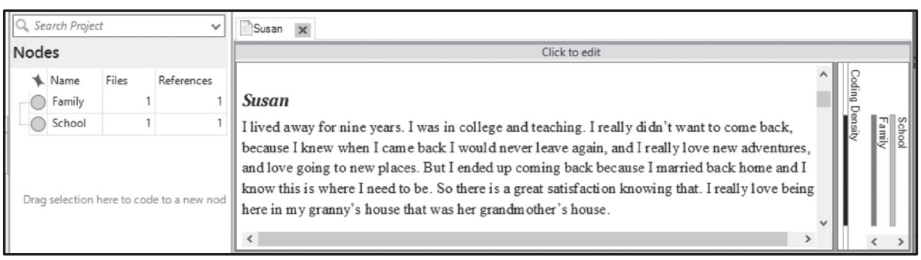


**Figure 3.9** Viewing Coding Stripes in a File

### Turning on the colours you assigned to the Nodes (if you did!)

⇒ *Ribbon*: **Document** / **View** › **Coding Stripes** › **Item Colours** / **Use Item Colours**.

### Coding Density

⇒ With your cursor/pointer, hover over the vertical black/grey/white **Coding Density** stripe to the right of your text to see the associated Nodes (Figure 3.9).
  - The darker the stripe the more Nodes you have applied.

*(Continued)*

(Continued)

⇒ **To modify the default of seeing only seven Coding Stripes**
- *Ribbon*: **Document** / **View** › **Coding Stripes** › **Number of stripes** (up to 200) › **OK**.

*Viewing coding with the Highlight tool*

⇒ **Single-click** (not twice) on any stripe to turn on Highlighting for that Node in the File.
⇒ To turn the Highlight off go to the *Ribbon*: **Document** / **View** › **Highlight** › **None**.

*Uncoding with a stripe*

⇒ **Right-click** on a stripe to **Uncode** (and see additional options).

---

You have completed your basic, guided tour of coding from inside of the File and examining the work you have done. As you continue to work with data, however, you will occasionally want to open a Node instead of a File, in order to see things from a different perspective while asking yourself some of the following questions:

- What was said before or after this Reference?
- What if I want to code this same Reference somewhere else?
- Is this Node too general or too specific?
- Has my understanding of this concept shifted?
- Did I code too much or too little context?
- How do I uncode some or all of a Reference while I review a Node?
- To what other Nodes is this data coded?

Some of the tools we will now explore are similar to the ones you used in the File and some of them are new.

## 3.c.

### Expanding Coding Context

⇒ *List View*: **Double-click** on one of your **Nodes** so it opens › **Highlight** a portion of a Reference (or multiple references) **Right-click** › **Coding Context** › **Narrow** (to get 5 words on either side as in Figure 3.10).
- **Broad** provides the context of the entire paragraph.
- **Custom** allows you to choose a number of words on either side.

⇒ On your keyboard select **Ctrl** / **Cmd** + **A** to select all of the References and then expand the Coding Context of all simultaneously by following the steps above.
⇒ Mac users might need to go to the bottom of the *Detail View* and drag up to select everything.

⇒ If seeing it as temporary context is insufficient and you want to retain some or all of the context around the coded portion, then select and code text in your usual way (e.g., drag-and-drop the additional context into the Node). It will show as coded text in the Node next time you open it.

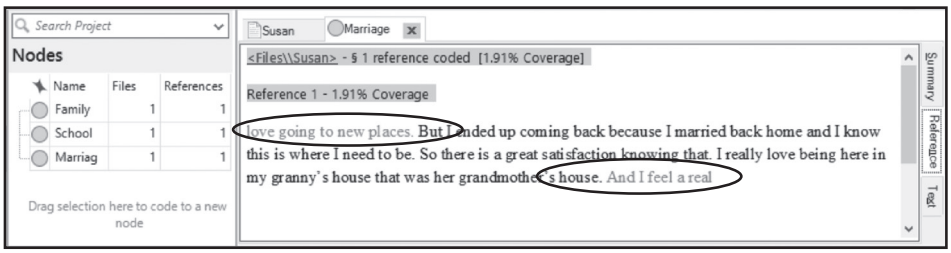


**Figure 3.10** Viewing narrow coding context in a Node

## Coding from a Node to another Node

⇒ *Detail View*: Select the text in a **Node** › Drag it to the *List View* to another **Node**.
- This does not uncode the data, it simply adds more coding to the File in the Project.

## Coding Stripes in a Node

⇒ *Detail View*: Click on one of the **References** (so NVivo knows you are in this window).
⇒ *Ribbon*: **Node** / **View** › **Coding Stripes** › **Most Coding** / **Nodes Most Coding** (Figure 3.11).

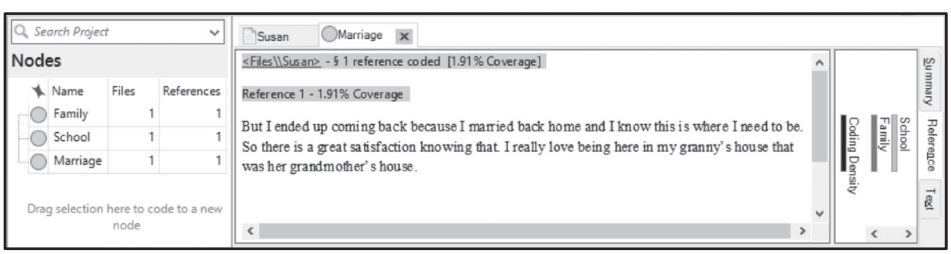


**Figure 3.11** Viewing Coding Stripes in a Node

## Highlight in a Node

⇒ **Single-click** (not twice) on any Coding Stripe to turn on Highlighting for that other Node.
⇒ To turn the Highlight off go to the *Ribbon:* **Node** / **View** › **Highlight** › **None**.

*(Continued)*

(Continued)

### Uncoding from the Node

⇒ *Detail View*: Select text in the **Node** to be uncoded (code the text where it belongs before you uncode it, because it is about to disappear from the screen) › **Right-click** › **Uncode from This Node** › **OK** / **Uncode Selection** › **At This Node**.
⇒ As an alternative use **Uncode** / **Uncode Selection** › **At Existing Nodes or Cases** › tick the desired boxes › **OK** / **Select**.

### View the context in the File

⇒ To see the data in its full, original context, **Right-click** on a **Reference** › **Open Referenced File** (Figure 3.12).
⇒ Turn Highlight off in the *Ribbon*: **File** / **View** › **Highlight** › **None**.

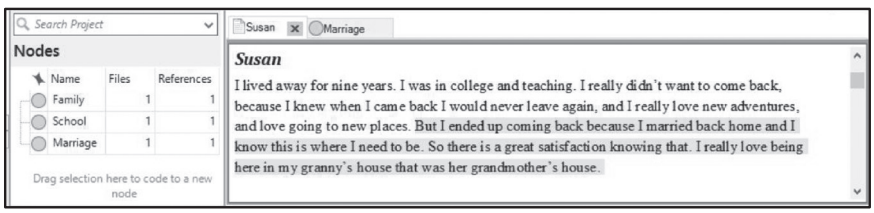


**Figure 3.12** Jumping from a Reference in a Node to the File

We began this basic, guided tour in one of your Files, where we showed you how to create Nodes, code, and visualize your coding with Coding Stripes and Highlight. Then we turned to one of your Nodes to play with similar tools for coding, uncoding, and visualizing your work. We just ended back where we started, in one of the Files. You can now repeat the steps until you feel familiar with the basic tactics, explore some of the alternative ways to accomplish the same outcomes that we describe next or skip to the subsequent section on Miscellaneous Node management.

## Alternative tactics for creating Nodes, Coding, and Uncoding

As with most software, many keyboard shortcuts are provided as alternatives to working with the *Ribbon* or a Right-click. You will find these while you work by noticing the additional symbols next to the names of the actions. For instance, instead of going to the *Ribbon:* **Home** > **Copy** / *Menu bar:* **Edit > Copy**, you can use **Ctrl** / **Cmd** + **C** (Figure 3.13).

A comprehensive list of keyboard shortcuts can be found in the NVivo Help files.

NVivo also provides other options in addition to keyboard shortcuts that you might prefer because they seem easier to you or because they are especially helpful in particular circumstances. Examples include:


**Figure 3.13** Shortcut keys to Copy in Windows (left) and Mac (right)

- The creation and naming of Nodes directly from participant speech (in vivo coding).
- Rapid review of a File to code additional text to a single Node that you recently added to your list of Nodes.
- Uncoding from multiple Nodes simultaneously.

These moves can be achieved by using the *Ribbon,* a **right-click** on selected text, or the Windows Quick Coding bar (Figure 3.14) or the Mac Coding Panel (Figure 3.15).

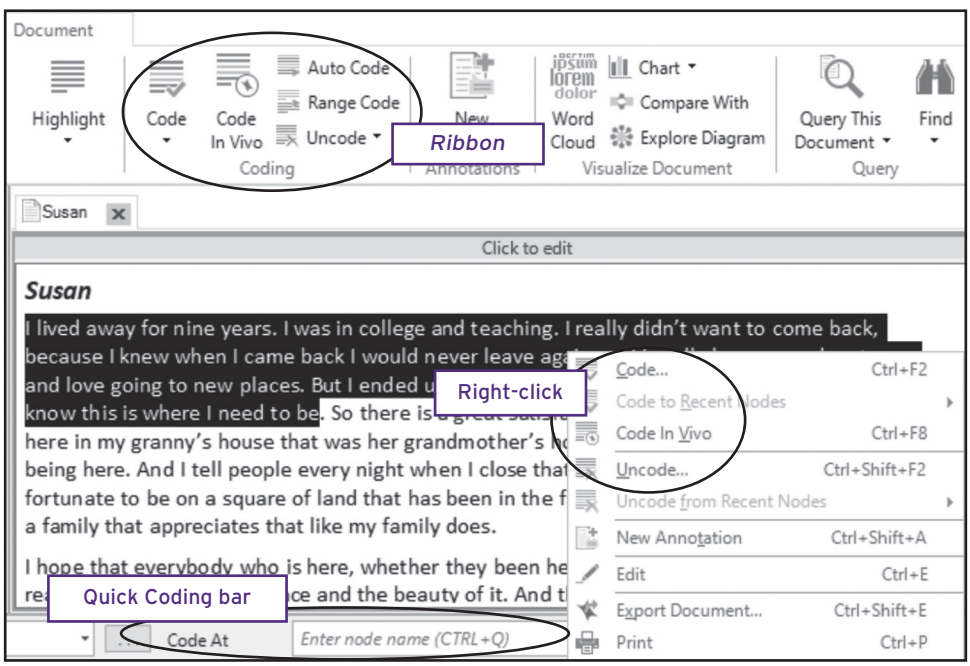


**Figure 3.14** Coding and uncoding alternatives: *Ribbon*, right-click, Quick Coding bar (Windows)

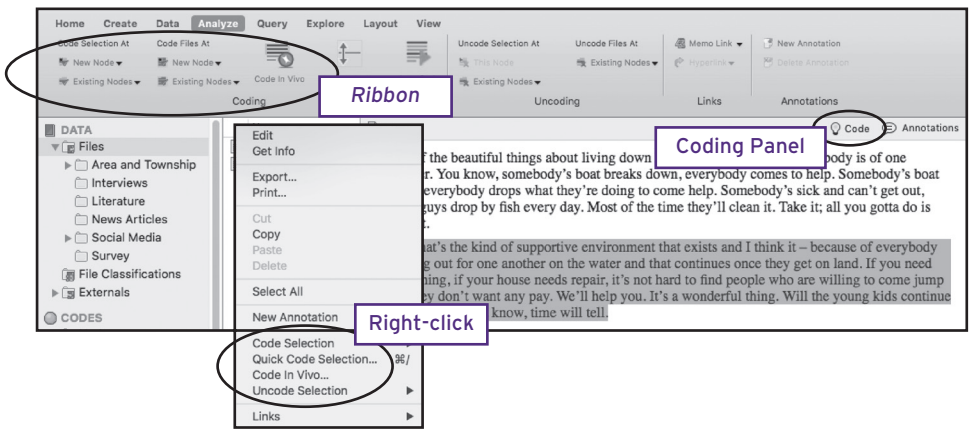


**Figure 3.15** Coding and uncoding alternatives: *Ribbon*, right-click, Coding Panel (Mac)

Figure 3.14 and 3.15 might be sufficient for you to explore the options, but if you would like additional details about using these alternative tactics for coding and uncoding, see the online resources (https://study.sagepub.com/jackson3e) to view the 'Coding and uncoding alternatives' for Windows and Mac.

## Miscellaneous Node management

The bottom line about your Nodes and coding is that you are never stuck with a prior decision; this includes the ability to rename, uncode, recode, change the Node Description, adjust the Node colour, etc. In addition, there are a few additional tactics for managing your Nodes that you will want to understand before we describe some of the more complex aspects of managing your Node structure in Chapter 4. These tactics are deleting Nodes, merging two or more Nodes together, and customizing your Node list.

## 3.d.

### Moving, Deleting, and Merging Nodes

#### *Moving Nodes*

⇒ *List View*: Select the Node to move › Drag it to the destination (e.g., under another Node, creating 'parent' and 'child' Nodes. See Chapter 4, pp. 118–121).

- To move the Node to the top level of Nodes: **Drag** the Node to the *Navigation View*: **Nodes**.

#### *Deleting Nodes*

⇒ *Navigation View*: **Codes** › **Nodes**.
⇒ *List View*: Select the **Node** to remove from your Project › **Right-click** › **Delete** (or click **Delete** on your keyboard) › **Yes** / **Delete**. (This also deletes all coding for that Node.)

*Merging Nodes*

⇒ *Navigation View*: **Codes** › **Nodes**.
⇒ *List View*: Select a **Node** › **Right-click** › **Cut** / **Copy** › select the other **Node** › **Right-click** › **Merge into Selected Node** › **OK** / **Merge** (Figure 3.16).
⇒ *List View:* **Right-click** *on original Node* › **Delete** › **Delete**.

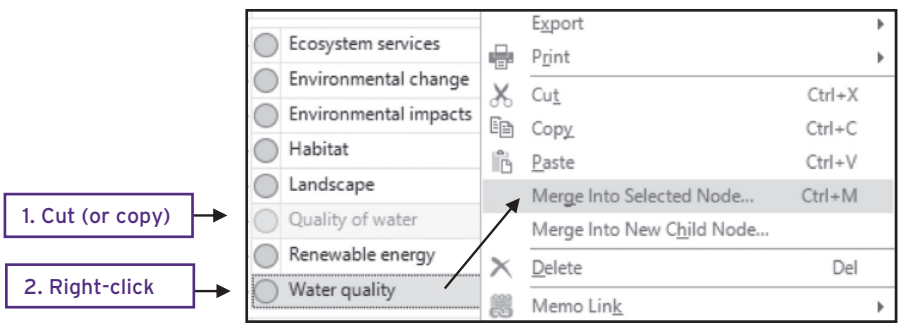


**Figure 3.16** Merging Nodes

## Customizing your display and listing Nodes

You can customize the display in the *List View* to show or hide information such as the Description and how many Sources or References are coded to the Node. This list can be customized independently in each of two contexts: (1) when the *Detail View* is closed (resulting a much wider *List View*, which makes it more practical to see additional information); and (2) when the *Detail View* is open (resulting in a narrower *List View*). Furthermore, once you set this customization, you can export the displayed information into a text, Excel or pdf file for your own reference, to bring to a data meeting, or to use in a report/publication.

## 3.e.

### Customize your display (Windows only)

⇒ If you have anything open in the *Detail View*, close all items by clicking the **X** in each tab (or on any tab **Right-click** › **Close All**).
⇒ *List View*: **Single-click** on any **Node** (not twice, or it will open). This ensures your cursor is active in the *List View*.
⇒ *Ribbon*: **Home** › ***List View* icon** (in the **Workspace group** on the right side of the *Ribbon*) › **Customize**.
⇒ Use the arrows to move items left or right between **Available columns** and **Selected columns** to suit your preferences (Figure 3.17).
  ○ We recommend that you begin with **Description**, **Files**, and **References** in the **Selected Columns** window.

*(Continued)*

(Continued)

⇒ Use the **arrows** on the right to rearrange the order of these items.
⇒ **OK**.

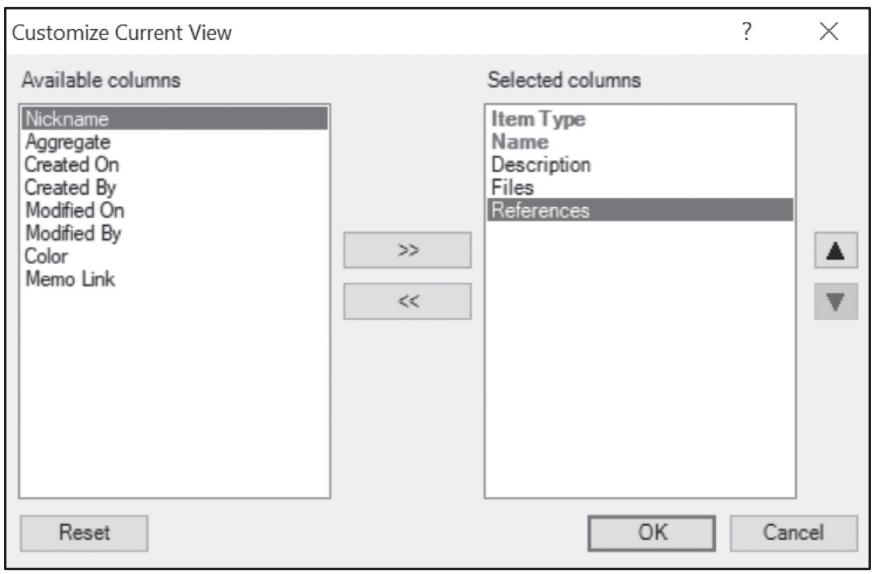


**Figure 3.17** Window to Customize Current View in the *List View*

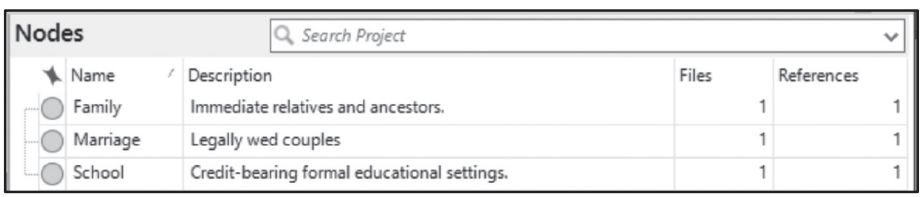


**Figure 3.18** Customized Node *List View*

To customize the narrower list that appears in *List View* when items are open in the *Detail View*, simply click on any Node in the *List View* and follow the same steps to customize the view. Here we recommend not having any additional data showing.

## How many Nodes should I create?

There is no set standard on 'too many' or 'too few' Nodes. The Nodes should help you generate, organize, and reconsider the concepts in your data. Having too many or too few can prevent this and you should use your judgement (and guidance from experienced qualitative researchers) to help you land on the sweet spot about the number of Nodes in your system, given your research design.

## Is it Node worthy?

New Nodes proliferate early in the project, but as you work through further documents, you are more likely to add coding to existing Nodes. If you keep making a new Node for each passage, then it might be time to rethink. Very specific categories tend to be of little value for analysis. Check after a few documents to identify those Nodes that code only one passage.

### 3.f.

⇒ *Navigation View*: **Codes** > **Nodes**.
⇒ *Ribbon:* **View > *Detail View* > At Bottom.**

Then challenge each of these with Lyn Richards' question 'Why am I interested in that?' to identify a broader concept or purpose for this Node. The discipline of writing a Description for each Node in the Node Properties dialogue can also help sort out what they are about, and which you should keep, delete, or modify (*List View*: **Right-click** on a Node > **Node Properties** / **Get Info**).

## Do I code the question?

When a conversation is being held about a particular topic, there is often an exchange between the participant and the interviewer while on the same topic area, such that you will want to code more than one paragraph or speaking turn. When you do so, include the interviewer's intervening text as well as the participant's text, for methodological and practical reasons:

- It is helpful to know how the interviewer was prompting or responding to what the participant was saying.
- Every time the coded passage is broken by a non-coded interviewer's turn, then the parts will be displayed as separate References and will consequently break up the text when the node is being reviewed.

Where the passage you are coding is not interrupted by the interviewer you might not need to include the question – unless it is needed to understand the participant's response (e.g., when interviewing someone who can't say more than 'yeah' or 'nah').

## How much surrounding context should I select?

Coding can be applied to a word, phrase, sentence, paragraph, long passage, or a whole document. The length of the passage to be coded (which turns it into a Reference) is dependent

on context and analytic purpose. You need to include a sufficient length of passage for the coded segment to make sense when you retrieve it, but at the same time, avoid coding long, drawn-out, vaguely relevant passages that are going to clutter your reading when you review the text at the Node. This is about balancing specificity and context. It helps, too, to have a basic understanding of how the reporting and Query functions of the software work (see Chapter 7 for an overview of these). As a general guideline, we advise that you tend to over-code for context rather than under-code for context. Coding very short segments could:

- make the retrievals hard to interpret, because they are too disconnected from the original context;
- increase the time it takes to code because so many more passages are chosen during the process;
- limit the utility of Queries, particularly those that are looking for intersecting content in a File.
- If you want to find whether concepts such as *Financial resources*, *Material resources*, or *Emotional resources* (for instance) are used to manage challenges like *Language barriers*, *Transportation access*, and *Peer group upheaval* (for instance) then be sure to code the passage at the resource AND at the challenge. This will allow you to use a Matrix Coding Query (Chapter 6, pp. 186–189; Chapter 7, pp. 208–209) to discover patterns of association between resources and challenges (Table 3.1). In addition to counting the intersecting data, you can also access the qualitative material by double-clicking on the cell.

**Table 3.1** Simulation of Matrix Coding Query output

| | Financial resources | Material resources | Emotional resources |
|---|---|---|---|
| Language barriers | 45 | 18 | 22 |
| Transportation access | 55 | 33 | 12 |
| Peer group upheaval | 52 | 12 | 67 |

## Printing and exporting Node lists and Codebooks

Periodically, you will want to move data or other information about the Project into another file, such as a report or a dissertation. At this juncture, your list of Nodes, their Descriptions and the number of Files and References would be helpful to print, export, or generate as an NVivo formatted Codebook. These three options provide a slightly different output, so experiment with them (and with the *Detail View* open as well as closed). Remember that if you want to create a list of your Nodes with descriptions (Figure 3.18), customize your *List View* to include Descriptions (Windows only).

## 3.g.

### Print, Export as a list, or Export as a Codebook

*Print*

⇒ *List View*: Select a **Node**.
- *Ribbon:* **Share** › **Print List.**
- *Menu bar:* **File** › **Print List**.

*Export*

⇒ *List View*: Select a **Node**.
- *Ribbon:* **Share** › **Export.**
- *Ribbon:* **Data** › **Export List.**

⇒ Choose a location, provide a Name, and file type for your export › **Save** / **OK**.

*Codebook*

⇒ *Ribbon*: **Share** › **Export Codebook** / **Data** › **Codebook** › **Select**
⇒ Choose a location, provide a Name, and File type / format for your export › **Save** / **OK**.

## Moving on

After working your way through a few Files, you will have built up quite a list of Nodes. In the next chapter, we start by thinking about how these might be organized to better reflect the structure of your data, and to make your coding system and the coding process more manageable. In the meantime, you might consider visualizing the coding you have applied to a Node or a File. There are several ways to accomplish this, but we provide one of each for you. You might also consider running a Coding Query to examine File contents where two Nodes intersect.

### Visualizing your coding with a Node Chart

## 3.h.

### Generating a Node Chart

You can instantly produce a chart that represents the Files coded to a Node (see example in Figure 3.19).

⇒ *Navigation View*: **Codes** › **Nodes**.
⇒ *List View*: **Select** any Node that is coded by multiple Files.

*(Continued)*

(Continued)

⇒ **Right-click** › **Visualize** › **Chart Node Coding** / **Chart Selected Node**.
⇒ *Ribbon* / **Data Tab and Format tab**
- See additional options to change axes, values, labels, etc.

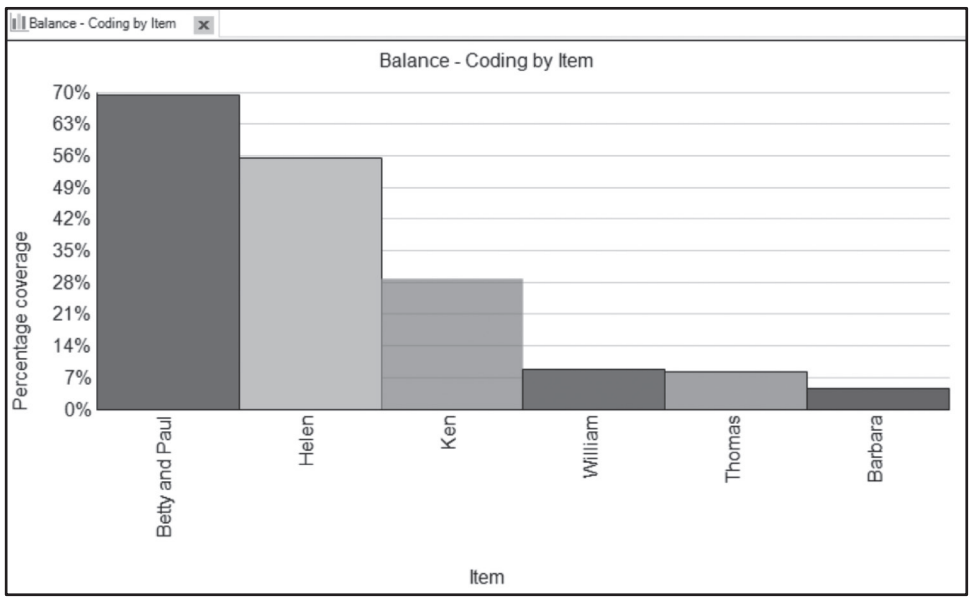


**Figure 3.19** Node Chart of Files coded to *Balance* based on the percentage coverage

## Visualizing your coding with a Comparison Diagram

Comparison Diagrams help you examine the way two items in the Project have shared and/or unique connections with other items in the Project.

### Farmers who adopted two strategies or only one

Philip Thomas, in a study for the Cooperative Research Centre for Sheep Industry Innovation at the University of New England in regional New South Wales, wanted to know if the farmers who adopted one kind of innovative animal husbandry practice also adopted another. Of the 48 farmers in the sample, 29 adopted the practice of scanning their ewes each season to detect those which were not pregnant, pregnant, or pregnant with multiples, so they could cull or regulate nutrition accordingly. Additionally, 22 were employing measures to reduce resistance to drenches used for worm control. Figure 3.20 shows the kind of display that can be generated from his data, to assist in answering his question about adoption of innovation.

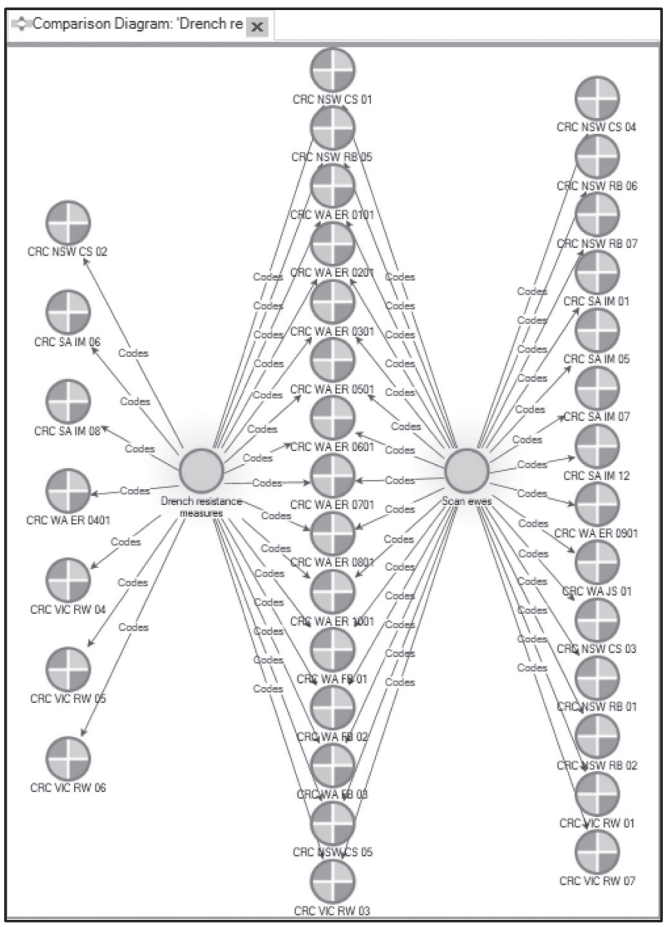


**Figure 3.20** Comparison diagram showing which farmers adopted one or other or both innovative practices

## 3.i.

### Generating a Comparison Diagram

You can generate a Comparison Diagram on the Files coded to one or more of your Nodes.

### Create a Comparison Diagram

- ⇒ *Navigation View*: **Codes** › **Nodes**.
- ⇒ *List View*: **Control-click / Cmd-click** to select the two Nodes to be compared **Right-click** › **Visualize** › **Compare / Compare Selected Items.**
- ⇒ OR: Select one Node › **Right-click** › **Visualize** › **Compare With ...**
- ⇒ *Ribbon*: **Comparison Diagram / Display Tab** › Use checkboxes to modify what is included in the diagram, e.g., you are unlikely to want both Files and Cases showing.
- ⇒ *Detail View*: **Right-click** › **Properties / Get Info** on any item in the diagram to see its Properties.

If you have not already, you might also find it helpful to write a brief case vignette or a summary of what you learned from the File about the main issues of interest in your study in the linked Memo, potentially also using See Also Links (Chapter 1, pp. 30–33). You can also right-click anywhere in the Chart or Comparison Diagram, copy, and then paste it into your linked Memo.

## Coding Query

You will learn more about the Coding Query in Chapter 7 (pp. 209–210) but for now you should experiment with a simple example of the intersection of two Nodes. Before running this Query, you might check that you know you have at least one intersecting Reference in the two Nodes.

## 3.j.

### Using the Highlight to check for intersecting content

⇒ *Navigation View*: **Codes** › **Nodes**.
⇒ *List View*: **Double-click** on the first Node to open it.
⇒ *Ribbon*: **Node** / **View** › **Highlight** › **Coding for Selected Items** › **tick the box** next to the second Node › **OK** / **Select**.

Yellow highlight indicates content coded at both Nodes. When you run the Coding Query, you will see the same content.

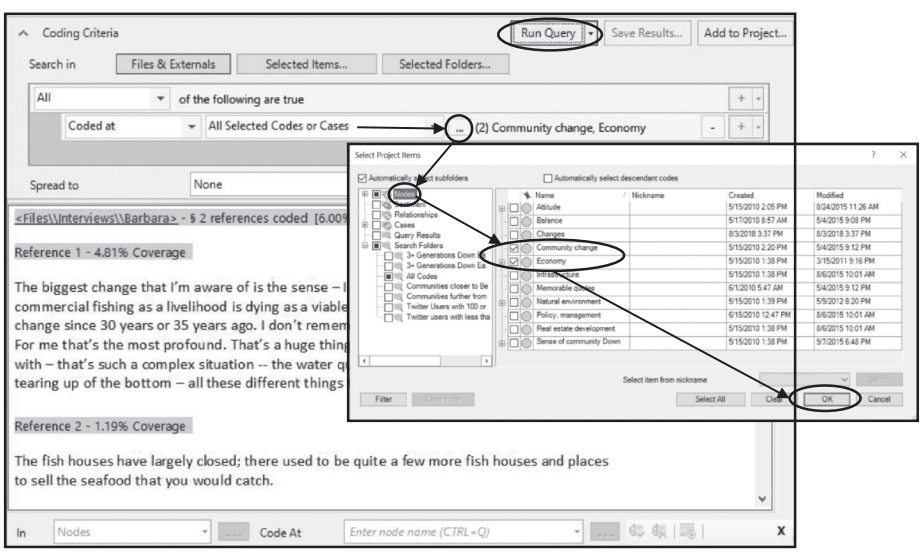


**Figure 3.21** Coding Query in the *Environmental Change* Project on the Nodes *Community change* and *Economy*

### Running a Coding Query

⇒ *Ribbon*: **Explore** / **Query** › **Coding**.
⇒ Select the '...' next to **All Selected Nodes or Cases** (Figure 3.21) / Select the arrow next to **all of these nodes** › use the left side of the window to click once on the word **Nodes** (not the tick box) › use the right side of the window to **tick the box** next to each of the two Nodes › **OK** / **Select**.
⇒ **Run Query** › the bottom half of the window displays the intersecting content.

For additional information about the options in this Query, see Chapter 7 (pp. 209–210).

## SECTION 3.4: CHAPTER 3 TAKEAWAYS

### Key points from this chapter

- Coding is an inherently flexible process and highly dependent on the context, research questions, methodologies, and types of data engaged in the process.
- You can change every decision you make about your coding in NVivo (Node names, descriptions, locations in the hierarchy, coded references, colours, etc.)
- A Node is simply a pointer or a link to the File. The only thing in a Node is a command that says, 'go get it!' When you code you establish these pointers and when you delete you dissolve these pointers.
- You will likely need to move beyond simple, descriptive coding to generate interesting interpretations from the data. Play around with the many strategies for constructing meaning through coding. Remember to challenge your initial interpretations of the data and take alternative views before drawing conclusions.
- Write, write, write. Use your Memos and See Also Links.

## Tips, challenges, and warnings

### Nodes and Coding

- If this is your first time coding, you will benefit from reading more detailed literature about the process from various perspectives and/or getting guidance from an expert.
- When you are struggling with conceptualization (and if you aren't part of a team project), connect with a group of other researchers undertaking qualitative work – the discipline area is irrelevant – and meet with them on an occasional basis. Each takes a turn to bring a short sample of data (maximum one page, de-identified) as the basis for discussion in that session.

- Spending too much time calculating how long it will take you to code or watching the scroll bar while you code to see how much is left might mean you are focusing more on time and mechanics than data.
- Chaos may reign periodically. You might occasionally need to force yourself to stop and rethink things and this can be done inside or outside of the software.
- If you are new to coding, you will feel a strong need to create the 'right' Nodes. There is no one, right approach and you might struggle to find the internal logic that makes the most sense to you as a researcher in this context (look to your theory and methods for guidance; seek advice from an experienced qualitative researcher).
- Keep Files in 'read only' mode while coding, or you might accidentally rearrange the text.
- By default, NVivo codes the entire word, so you do not need to precisely grab the first and last character of a Reference (you can change this default in the application options).
- Triple-click to quickly select an entire paragraph for coding.
- If you find you want to add further coding on either side of the passage you just coded, select overlapping text to ensure the passages connect and are combined as one passage. Text coded more than once with the same Node will show up only once, because the Node is simply pointing to the text (and that text is unique in the transcript).

## Visualizing with Coding Stripes and Highlighting

- If you tend to jump around in the document while you code, turn on Highlighting for all of your conceptual/thematic Nodes to easily see what you have already coded.
- Quickly find where you left off the last time you were coding by turning on Coding Stripes for Nodes Recently Coding.
- Turn on Coding Stripes within a Node to see what else is going on to help you think.

## Tracking and writing

- The description field of your Nodes is one place to write. While the goal is to end up with concise descriptions, you can add brief quotes or examples to the descriptions as long as you keep within the limit of 512 characters, including spaces.
- If you want to track the way your Nodes and Node Descriptions change over time, do so in a Memo. You can also create backup copies at key junctures to see Node structures and descriptions from prior phases.
- Use Annotations to track ideas as you code a document and then decide what to write up more formally in a Memo when you are done coding that document.
- If you are flooded with ideas while you code and fear you will have hundreds of Nodes, start by using Memos and See Also Links. Wait until you know which concepts are 'node worthy' (Richards, 2009).

## Videos and online resources (https://study.sagepub.com/jackson3e)

### Videos

- 'Coding and uncoding alternatives' (Windows, Mac)

## Help files

- *File*: Help.
- *Menu bar*: **Help > NVivo Help.**

Search for any of the following:

- Charts
- Code files and manage codes
- Coding
- Comparison diagrams
- Export nodes
- Reorder and organize nodes
- Use coding stripes to explore coding

## Practice questions

1 Select a 3–5 page section of one of your transcripts and then choose four of the eight strategies we suggest for identifying and naming nodes (p. 72). Code the pages from beginning to end using strategy 1; then again for strategy 2, etc. Write a Memo in your Project about how these strategies help you see different (or similar) things in the data and how they help you identify/create different (or similar) Nodes. Which of them might be most appropriate and fruitful as you continue coding?

2 Do you think you are more of a 'splitter' or a 'lumper' when it comes to coding? Why? How do you think this will be reflected in your use of Nodes?

3 Write a reflexive journal/Memo in NVivo about how it felt to do your first round of coding in NVivo (whether you have experience analysing qualitative data or not). What did you like? What was annoying? In what ways might NVivo facilitate your analysis? In what ways might it get in the way?