Grouping in TAMS Analyzer

Analysis in qualitative research ultimately requires some means of grouping codes together into other categories. These category schemes are complicated in that they may represent either a many-to-one relationship between codes and categories or many-to-many relationships. TAMS Analyzer provides mechanisms for both. TAMS Analyzer provides five different mechanisms for creating categories, and each has its own advantages and disadvantages. These categorizing schemes can all be used singly or in combination.

1. Adding a root code. If you are grouping codes into a single category scheme in which each code will only be in a single category you can add a root code representing the category in which the rest of the code belongs. For example given three codes aa, aa>bb, and cc; you might want to group aa>bb and cc into a group X and aa into a group Y. By adding a code at the root, so that these become X>aa>bb, X>cc and Y>aa. This is done with Results->Recode->Add root code (and reversed with Results->Recode->Delete root codes). The advantage of this that you can use the full power of TAMS Analyzer's ability to search for and select codes. This includes the data summary feature, The other advantage is that once a code is attached to a root code, new passages will automatically be put into the category as you are coding (since the category is part of the code).

   The disadvantages of this approach are that codes cannot be put in more than one category or category system using this technique (though other techniques can be used as well). Furthermore, as in the example I gave, existing groups (e.g., aa in the example) may be broken up. This can somewhat be compensated for through the use of ">" in searches (looking for >aa will look for aa at all levels of the code) and regex selections.

2. Adding another layer of codes. Consider two codes a>b and c which you want to put in category m. You could simply wrap each passage coded with a>b and c with a new code m. You would simply mark the instances you are interested in and use the "Results->Recode->Add code" menu item. You must make sure that you check for nested, however since there will very likely be some passages that are coded with both a>b and c and will thus have been put in category m two times. The real advantage of this mechanism is that you can use data summaries and other reporting mechanisms that are set up for codes. For the subsequent methods I describe, you have to use data comparison to manually get measures of frequency. This is not too bad, but there are levels of

   On the other hand, consider what happens if, upon examination you decide to mark a few additional passages with a>b and c, after you have used the "Add code" command. These new passages will not be automatically coded with m. You have to somehow add them yourself to the category. While not onerous it is a conscious effort (I would suggest an unlimited search, selecting for a>b—and then c and then the other codes in the category; then removing from the selection any data with {m} in it; then, “add code” code m to what remains—everything but the last step could be automated with an auto set). Furthermore, it is one that you will have to do regularly, if you continue coding.

3. Code sets. In the project window there exists a tab called tags and sets. Within this tab there are other tabs to further facilitate activities for labeling codes and files; one of these tabs is marked “Codes” which is for creating and managing code sets. With code sets you can create groups of codes that can be named and act as a single category. These are very easy to assemble. Once assembled, you can look specifically for passages coded by that code set (load the code set and use the “Add current code set” workbench button to add the code set to the search criteria), and you can filter any results window by a code set (you can filter code sets in or out of the current selection). This gives a very handy way of viewing in a moment the data that falls in a particular code set category. You can also group codes by code sets to give you counts and produce “graphs” of the relation between your variables and code sets. Furthermore, passages coded after the creation of the set will automatically be returned when you ask for the set, since it is keeping track of codes not coded passages of text, unlike category schemes 1 and 2.

   While the summary statistics are not as complete as with other methods of creating categories—you can not use the data summary table, TA can produce counts using the data comparison tables, including counts cross tabulated with other variables. **In the present version of TA this is probably your best, most flexible, most easy to use, scheme for creating categories of codes**.

4. Autosets. Automatic sets, aka macros, provide a very powerful way of gathering data together that matches particular criteria. Furthermore, these criteria can be more complex than particular codes, e.g., it could be data that contains a particular word, or passages with a certain code but not containing a particular word, or having this particular code but not that one. Auto sets can be created that are “local” to a particular search or that are project wide, and thus available to all searches. Finally, autosets themselves can be subject to complicated comparisons through the use of “Set operations” (On the Results->Result sets menu). Here overlaps, exclusive or’s and unions can be created.

   On the other hand, the data summary mechanism does not work with auto sets. The best that can be done is what I described in the second paragraph of “Code sets.”

5. Named sets. Named sets offers a way to gather together specific results records rather than codes or some other larger category of data. These groups of records can be highly complicated, with connections only apparent to the analyst. Furthermore, these sets can be compared and studied using the Results->Result sets->Set operations dialogue. Finally, they are easy to create: once the data you

have is showing, you simply name the visible data with Results->Result sets->Create named set menu item.

On the other hand, named sets do not persist if the data is refreshed; they are not project wide; they do not automatically include new coded data; and they cannot be counted in data summaries. Their use, therefore, is important (given that they alone can group any data records) but limited.

Category method	Project Wide	Self-updating*	Countable	Easy to use	Advantages	Disadvantages
Category as root of codes	Yes	Yes	Yes, in DST and DCT**	Yes	Easy to use, supported through recode additions in 2.45. Once set up passages are categorized as coded. Can be analyzed with data summaries.	A code can only be put in a single category with this scheme.
Category as a layer of Codes	Yes	No	Yes in DST and DCT	Initially yes, subsequently no	Can be subject to analysis like any other code	Difficult to apply subsequently to coded passages en masse. Can lead to nested code problems
Code sets	Yes	Yes	DCT only	Yes	Integrated across project; easy to use	Can't be counted in a summary. Criteria is defined only by the _code column.
Auto sets	Optional	Yes	No	No	Most flexible way of winnowing data: no limits on which columns or how data is compared	Can be difficult to create and edit Can't be counted in summary reports.
Named sets	No	No	No	Yes	Can represent very complex sets of data. Very easy to create	Not persistent between refreshes, not project wide. Useful for quick investigations. Can't be counted in DST or DCT

*Self-updating in this context means that if you have a code a>b in a category m, then all new instances of passages coded a>b will also appear as category m. If a method of categorizing is not self-updating, you will need to manually add the new coded passage to category m.
**DST = Data summary table; DCT = Data comparison table