Analysis for Beginners using TAMS Analzyer

Matthew Weinstein

1. What you need to use this documentation

At this point you should have read through document called Coding for Beginners. If you have, hopefully you know the difference between workbench windows and document windows, you know that I use project windows and workbench windows interchangeably and can now create documents, enter data, create data codes, and apply those codes to sections of your document.

We'll be using a modified version of the Old McD. from *Coding for Beginners* for this analysis (which is merely to introduce you to the basic procedures for analysis):

```
{!repeat verse}

{verse}1{/verse}
{setting>rural}Old Macdonald had a farm EIEIO
and on his farm he had a pig, EIEIO
with an {sound>pig}oink, oink{/sound>pig} here and an
{sound>pig}oink, oink{/sound>pig} there
here an oink, there an oink, everywhere an
{sound>pig}oink, oink{/sound>pig}
Old Macdonald had a farm EIEIO{/setting>rural}

{verse}2{/verse}
{setting>rural}Old Macdonald had a farm EIEIO
and on his farm he had a cat, EIEIO
with a {sound>cat}meow, meow{/sound>cat} here and a
meow, meow there
here a meow, there a meow, everywhere a
{sound>cat}meow, meow{/sound>cat}
Old Macdonald had a farm EIEIO{/setting>rural}
```

This version of our old data introduces some very “not for beginners” ideas. The first of which is the metatag. A metatag is an instruction to the tams program. It’s the way the researcher specifies how TAMS should treat the data. Metatags are always surrounded by braces “{“ and “}”, just like codes, but begin with an exclamation mark. Our new document begins with a critical metatag: repeat.

You should find the project all set up in the documentation folder.

A. Repeat metatag

The repeat metatag tells TA that the following list of codes are different than normal data codes. These codes are used to indicate information that gives context to the data codes rather than the data codes themselves. In interviews the speaker's name might be a good repeat code. In field notes the time index, or location, or observer might all be repeat codes. We are telling TAMS that when it returns information about our data we want it to include this other information (who is speaking, the time code for field notes, or in this example the verse #) with the relevant data. This ability to attach information to each other is one of the real strengths of TAMS, but you can see that it is not exactly coding for beginners. Note that the repeat values have to precede the data that they are coding (actually if you read the user guide you can structure a document so that you can assign the repeat codes at different points—but for unstructured documents what I have said is true: the repeat values need to be set before the data codes.)

Examples:

1.

{!repeat speaker}

{speaker}bob{/speaker}: {identity}this is me{/identity}
{speaker}mary{/speaker}: {identity>negation}no it's not{/identity>negation}

2.
{!repeat speaker, time}

{time}110{/time}
{speaker}bob{/speaker}: {identity}this is me{/identity}
{speaker}mary{/speaker}: {identity>negation}no it's not{/identity>negation}

{time}225{/time}
{speaker}bob{/speaker}: {identity}I can prove it{/identity}
{speaker}mary{/speaker}: ok do so

2. What is analysis?

Analysis is a process of finding out what information is present in your data and what that information means. Practically it is the process of taking codes and finding out what pattern they form and testing meanings for that pattern (through searching for negative instances, for instance). To find these patterns in TAMS you ask the program to turn your interviews into tables that you can use to count instances of particular codes (or collections of codes) etc. In essence, what TAMS is all about is taking those codes and turning your interviews, field notes, etc., into a database or spread sheet that you can browse, sort or further search in a variety of ways. It is this culling through the data which is the heart of analysis.

3. Searching, sorting and selecting: the grand plan

At the beginning of your analysis, after you have coded your documents, there are three essential operations you need to master: searching, selecting and sorting. That is what this tutorial will concentrate on. When you have comfort with these three operations you should go on to explore data sets, autosets, and set operations. But these are not for beginners.

4. Searching

Searching is the procedure by which you take your coded data and turn it into a nice table-like database/spreadsheet. In this tutorial we will use the workbench for searching; though each document window has an individual search tab that you can use to find information just in just that document. The steps for searching for information are easy enough: specify a search list (which documents should the program look through), specify search criteria (do you want to find just one code or a series of codes, or all your coded passages?) and press the search button. This will give you a "results window," i.e., a database/table of the passages that meet the criteria you established.

A. Creating a search list

First we have to indicate what documents our program should search through. Often we have lots of documents in a project that may represent searchable aspects of our work as well as other items that we do not want to search. To create a search list we use the bottom of the project window to move items from the file list (the list of all the documents in our project) to the search list (those that TAMS should search). If you want all of your documents brought over use the >> button. If you want to empty the search list of all documents use the << button. Otherwise click on one file in the file list (that 's the left hand list at the bottom) and click on > to add it to the search list. If you want to remove a file click on it in the search list and click the < button.

You can also change the order of files to be searched by using the ^ and v buttons on the left hand side of the screen.

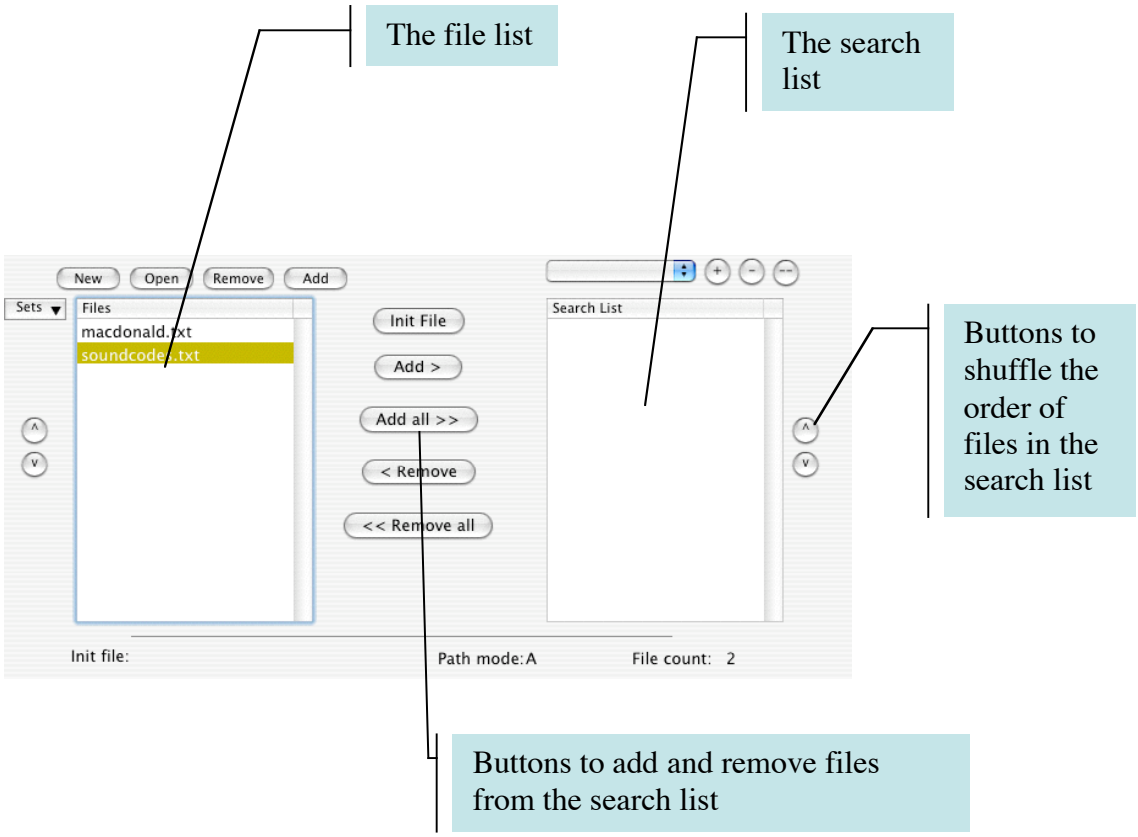


Figure 1: assembling a search list on the workbench.

For our project (included with this documentation, we have two files in the file list. We want only the file called “oldMacD.rtf” in the search list, so click on it on the left side of the project window and press the button labeled “Add >” to move it to the search list.

B. Executing a search

We have our search list now we need to tell tams what to search for. This we do in the top half of the workbench window. Our first search will be what is called an “unlimited” search, meaning we’re not putting any limits on what data is returned: we want it all. This will demonstrate how TAMS turns raw stuff (interviews) into tables. To do an unlimited search make sure that the field marked search is empty, pop-up menu underneath the check boxes says “Simple” and leave the “raw” box checked and the other two unchecked and press the button marked Search. Voila!

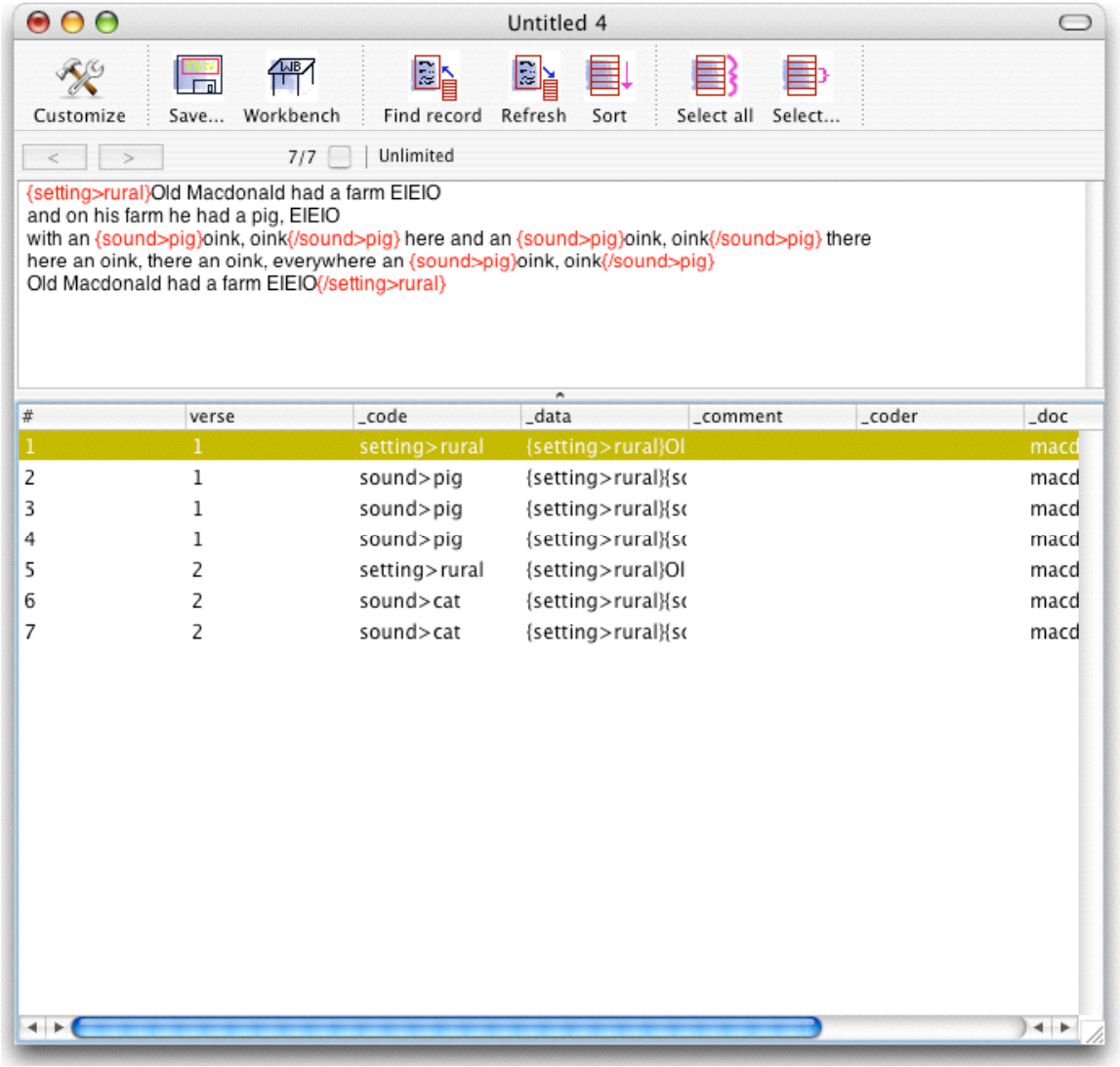


Figure 2. A results window

What you get is a results window: This is the window that lets YOU analyze your data. There are a lot of parts of a result window, so briefly:

1. There is a panel of buttons that you can configure and by default includes buttons for taking you back to your project window (Workbench), back to your document windows (find record), and for refreshing this window if one of your documents changes.

2. Then is a status panel which gives you information about your results. It tells you how many records are showing, a check box which tells you whether any of your documents have changed (which means you should hit the refresh button) and a description of the data you're seeing. In this case it tells us that we have done an unlimited search.

3. Below the status panel is a large pane that shows us the data of the selected record. To select a different record, click on a different row of the table below. This is the browser pane.

4. Finally, occupying the whole bottom of the window, there is the table of records: one row represents one passage of text that met the criteria we asked for.

A simple, unlimited search returns a row for every coded passage in our data. If a section of text has two codes: {code1}{code2}This is my text{/code2}{/code1}, for instance, that text will appear in two different rows in the table of records, one for code1 and one for code2 (yes, this might mean that there is redundancy. A non-simple search removes the redundancy and returns one row for every stretch of text surrounded by codes (no how many codes are embedded in that text, but you lose information that way).

By clicking on one of the rows you can use your up and down arrow to browse through your data, examining the passages in the browser pane. If you want to look at one row of the data in its original context click the "Find record" button.

Finally, it is worth examining the columns given to you in the table of records. First is a simple numbering of the shown records. Next will appear all of your universal and repeat values. Here we can see which verse each coded passage came from. If we had an interview we could see who was speaking. Next comes the code of the passage, followed by the data, followed by comments we've attached to the codes (another non-beginner function), the name of the coder if you are using multiple coders, the name of the document, and finally, the number of characters into the document that the passage started. The columns that always appear start with the "_" (underscore) character.

Now we can sort the data just by clicking on a header and pressing the sort button. This is not the best way to sort information though, a subject we will address in a future section.

C. Searching for specifics

There is a lot that can be done in unlimited searches, and more often than not that's what I use to explore my data. Often however, the researcher wants to see only a single code or a few codes. While that is possible to do in an unlimited search (see the section on "selecting data") it's easier just to search specifically for the codes of interest.

The easiest way to search for a specific code is to simply double-click the code from the list of codes on the workbench. So if we want to see all of the sections of text coded as sound>pig we could double click sound>pig from the list and hit the search button. Then only the sound pig records would appear in the results window. To clear the search field click the button called "Clear" on the work bench.

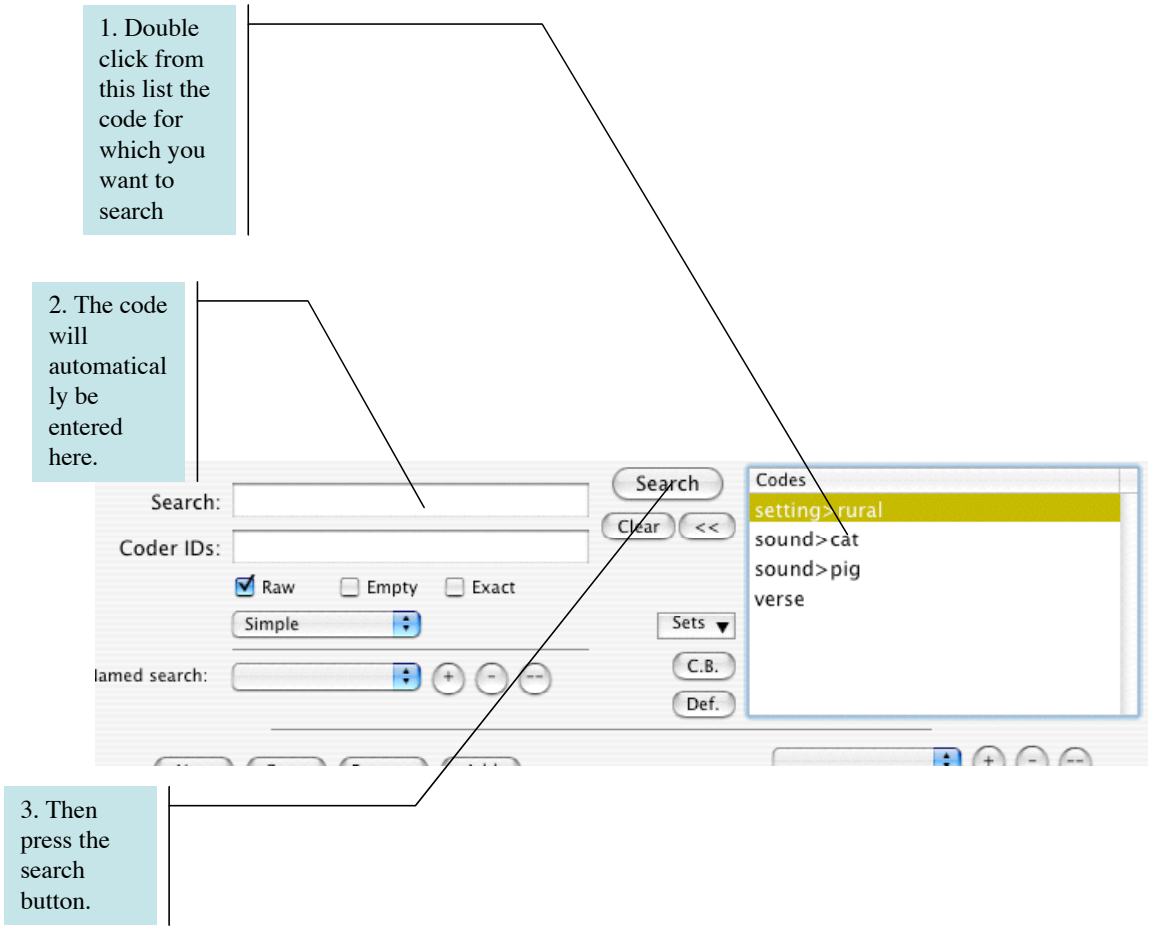


Figure 3. Searching for a single code

You could have also just typed the name of the code into the box labeled “Search:”

What If you were looking for either sound>cat or sound>pig? Simple:

1. Make sure that the Search: box is empty by pressing the “Clear” button.
2. Double click sound>cat off of the list
3. Type in a comma and a space after sound>cat in the Search: field box.
4. Double click sound>pig.
5. Press the search button

Alternatively you could have typed in “sound>cat, sound>pig” into the search box with the keyboard, and then pressed the search button. You could keep adding commas and different codes. If you wanted to search for three codes you could type “sound>cat, sound>pig, setting>rural”, for instance.

The codes sound>pig and sound>cat are part of a family of codes called **sound**. If we just look for sound we will get everything that is in the sound family including anything labeled sound, sound>pig, sound>cat, sound>cat>persian, etc. If I only want to look for those things that are sound and not sound>cat, etc., I have to click the “Exact” box under the search. Otherwise TAMS assumes I want the whole family.

Finally, and this is almost a beyond beginner item. If I want passages that have two codes, i.e., the intersection of coded passages, I would need to use a plus sign between the codes. In other words if I want to see the part of my document that is sound>pig and setting>rural at the same time I would type "sound>pig +setting>rural" into the search field.

If you feel comfortable with all of this you can learn a few other tricks to searching by clicking the search tab on a document window and looking at the Help box.

5. Sorting

Finding information is the first part of analysis. We're half done with the topic of finding information, but before continuing I want to offer as an interlude some notes on arranging information. This is the art of sorting, and TAMS can do very complex sorts. While beyond this tutorial, TAMS can also remember sort orders and use them in macros that are very powerful for finding key information across searches.

I've already noted that a simple way to sort is to click on a column in a results window and hit the sort button. The problem with this method of sorting is you have no control over the type of sorting that the program is doing. The more powerful sorting tools are under the "Results->Sort up" and "Results->Sort down" menus. These allow you to control the direction of the sort (Should A or Z be first? If A is first you are sorting up; if Z is first you are sorting down) and allow you to nest sorts.

For example you may put three or four documents in your search list and then want to see how many documents use each code. So you want to sort by the codes and then sort within each code by document. Specifically you would click on the column with the header _code and then pick "Results->Sort Up->alpha" (i.e., alphabetically) Then you would click on the header of the column labeled _doc and pick "Results->Sort up->alpha within" The word **within** means keep the first column sorted appropriately, but rearrange items that match in the first column according to the currently selected column." You could sort by any number of columns **within** each other. If you want to start over, just sort NOT within the others. Just pick "Results->Sort up->alpha" or whatever item is appropriate.

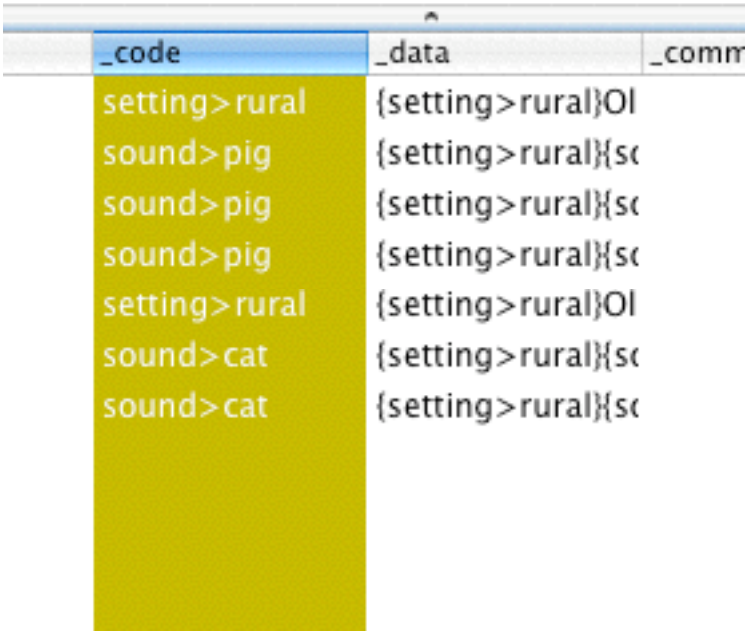


Figure 4. Column _code is now selected for searching or selecting

TAMS can sort alphabetically, numerically either by integer or real (floating point) comparison, by date (though you must pick the format first by picking it from the Results->Sort options…->Date format menu item first—it's best if there's only one date format in a project), and by code. This latter type of sort is really not part of a beginners tutorial, but just so that you know, if I set the code level to 1 (by picking the "Results->Sort options->Code level" dialogue) sound>cat and sound>dog will be taken to be identical for the purposes of sorting since they match at the FIRST (this is the meaning of code level 1) level of codes. Picking code level 2 means both level 1 and 2 must match. Ok, that's for another lesson.

Now you can organize your data as well as find it!

6. Selecting

The third type of operation you need to know about is selecting. Selecting can be thought of as finding information inside of a search. It's a way of searching searches. Remember that a results window is a fairly complete database, and a good database should be searchable. The terminology is confusing because selecting also refers to the way that we pick things on the screen with our mouse: select column _code, select the first field, etc. I'll try to use "pick" instead of select for the latter type of operation.

Let's take as an example of finding how many sound>cat codes there are out of the total number of coded passages. To start with, do an unlimited search: Go to your workbench and press the clear button and then the search button. You should now have a result window with one line for every coded passage in the document: one line per code.

We can see that we coded 7 passages not counting our repeat codes by looking right under the buttons on the results window:

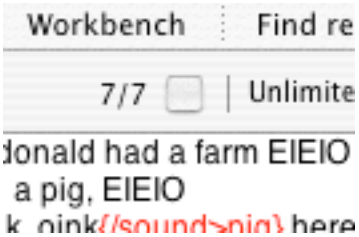


Figure 5. We found 7 coded passages (and all 7 are showing)

Let's find the sound>cat codes. Pick the _code column by clicking on its header. Then pick the Results->Select… menu option:

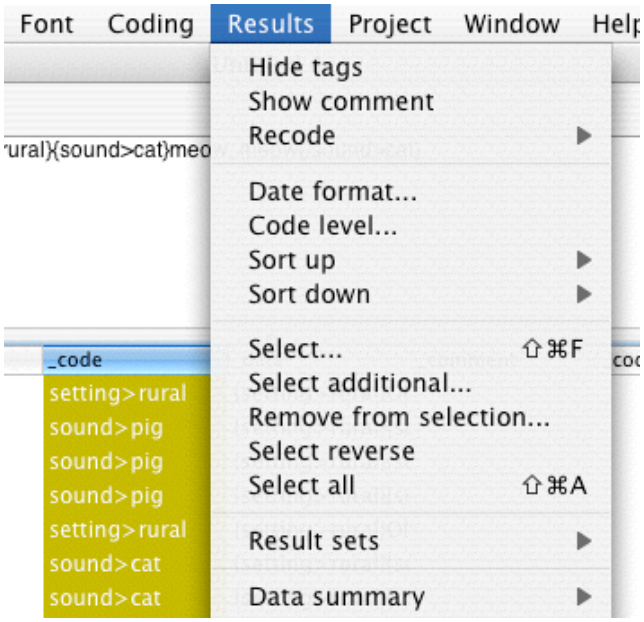


Figure 6. Picking Select…

This provides a dialogue box into which you can type the phrase you are looking for: sound>cat.

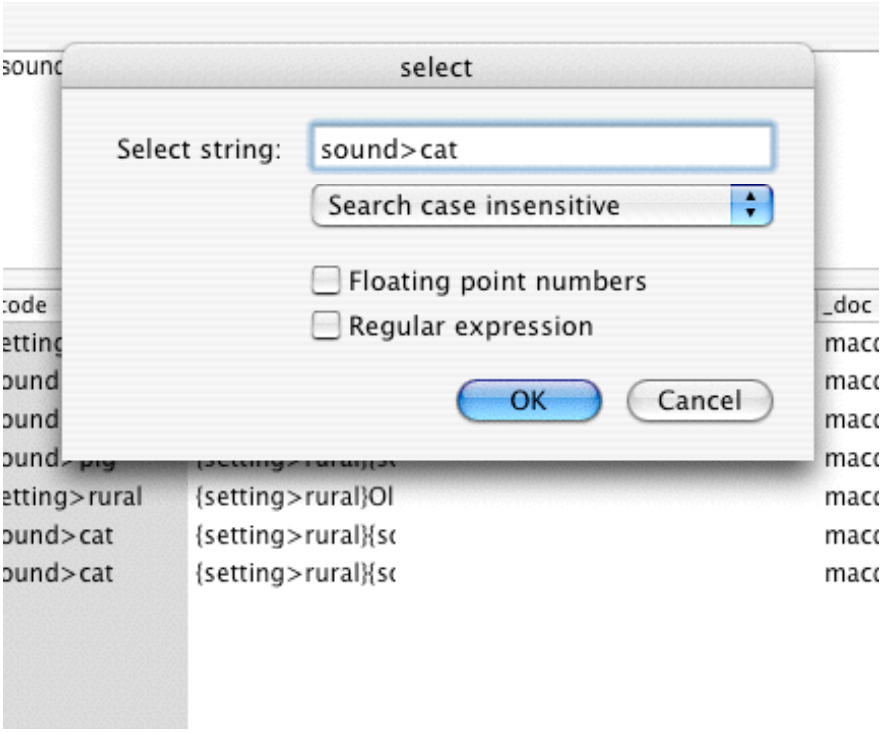


Figure 7. Selecting “sound>cat”

After clicking OK only those records that have sound>cat will be showing!

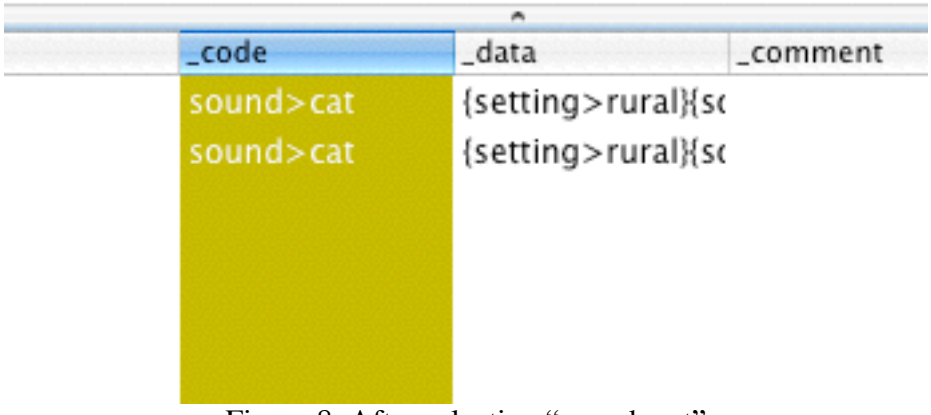


Figure 8. After selecting “sound>cat”

Now only two records are showing. And the count indicator under the buttons shows us that 2 out of 7 total records are sound>cat.

To see all of the records again pick “Results->Select all”

That's the heart of selecting. We could modify this in a whole variety of ways mostly shown in the menu displayed in Fig. 6. We could switch the records that are showing and the ones not showing (so only records that were not sound>cat were visible), by picking "Results->select reverse", we could select additional records from the total pool of found records by picking a column and picking "Results->Select additional." We could keep whittling down the records showing by doing more picking more columns and doing "Results->Select…" again. Finally we could whittle down our findings by picking a column and picking "Results->Remove from selection…" By combining these and using different columns: your _code column, your repeat columns, and your _data column you can find many patterns inside your data.

7. A more complicated example… Less for beginners

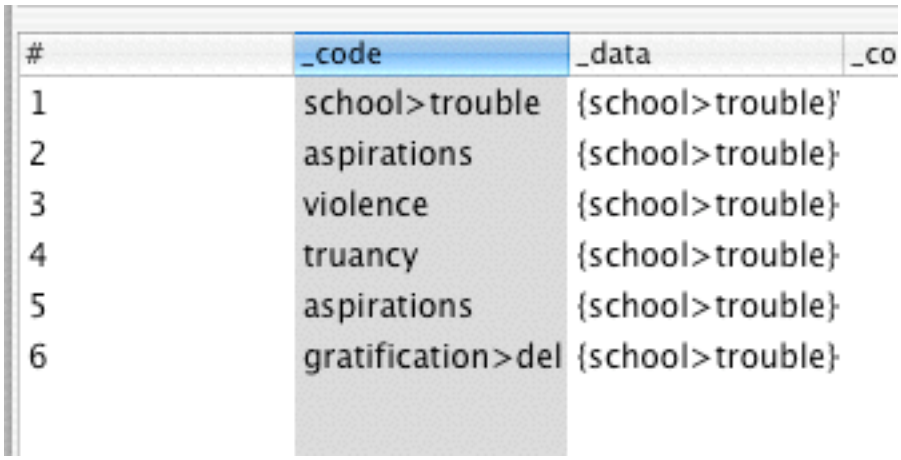
Before concluding this tutorial I want to work through a more advanced example of selecting. This is an alternative way of finding the intersection of coded passages by using Result->Select… It also shows some of the issues that can result from such searches. For this example I'm going to use this small passage:

```
R:  Well, my high school was known as a trouble
school. There were a lot of fights, and kids, uhm
wandering around, and most of us worked in factories
on the [city's] east side. Most of us partied rather
than worked.
```

We'll code this as follows:

```
R:  {school>trouble}Well, my high school was known as
    a trouble school. {aspirations}We weren't going
    anywhere. {/aspirations}{violence}There were a
    lot of fights{/violence}, and {truancy}kids, uhm
    wandering around{/truancy}, and {aspirations}most
    of us worked in factories on the [city's] east
    side{/aspirations}. {gratification>delayed}Most
    of us partied rather than
    worked.{/gratification>delayed}{/school>trouble}
```

Now let's imagine that we are searching for the intersection of discourses we've labeled "school>trouble" and "aspirations." Maybe we want to see how high school image and occupational aspirations correlate. One way to find these intersections is through searching for "school>trouble+aspirations." You can also find this from an unlimited search, however. Reasons for doing so include the ability to do "set mathematics" i.e., the ability to look at how groups of coded passages intersect with each other. To do this we're going to work from an unlimited search; that means we hit the search button with no criteria filled in.



#	_code	_data	_co
1	school>trouble	{school>trouble}	
2	aspirations	{school>trouble}	
3	violence	{school>trouble}	
4	truancy	{school>trouble}	
5	aspirations	{school>trouble}	
6	gratification>del	{school>trouble}	

Figure 9. Picking _code

Once we have done a search we are going to look for records that have both. As a first cut, let's work with just those records that have "school>trouble" Select the _code column by clicking it's header, and pick Results->Select… Fill in "school>trouble" and hit "ok."

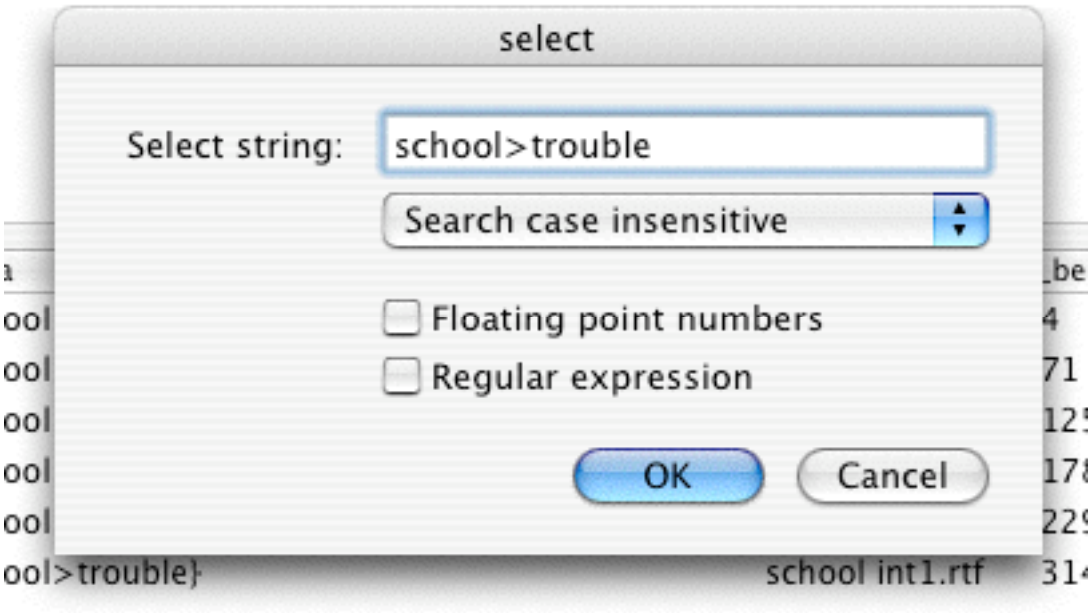


Figure 10. Selecting _code

Now we need to find the records that are left that have "aspirations" (in our example there is only one record to search, but that just points to the artifice of the example). Pick the _data column this time. Then pick Results->Select and enter "{aspirations". As the search string: voila: what you see are records that have both codes.

WARNING: In our example we would get a different number of records if we first searched for "aspirations" in the _code column and then "{school>trouble" in the _data column. We would eventually see the same text as we browsed through the records, but you should know that the number of records would be different. Finding how many passages coded with A have code B turns out not to be the same question as how many passages coded with B also have code A. Why? The first time we would get the passage marked as orange as the result of the first and second search:

```
R:  {school>trouble}Well, my highschool was known as a
     trouble school. {aspirations}We weren't going
     anywhere. {/aspirations}{violence}There were a
     lot of fights{/violence}, and {truancy}kids, uhm
     wandering around{/truancy}, and {aspirations}most
     of us worked in factories on the [city's] east
     side{/aspirations}. {gratification>delayed}Most
     of us partied rather than
     worked.{/gratification>delayed}{/school>trouble}
```

The second time we would get the following two orange passages returned separately.

```
R:  {school>trouble}Well, my highschool was known as a
     trouble school. {aspirations}We weren't going
     anywhere. {/aspirations}{violence}There were a
     lot of fights{/violence}, and {truancy}kids, uhm
     wandering around{/truancy}, and {aspirations}most
     of us worked in factories on the [city's] east
     side{/aspirations}. {gratification>delayed}Most
     of us partied rather than
     worked.{/gratification>delayed}{/school>trouble}
```

If you want a count of intersections of the two codes rather than a count of how many of coded passage X also has some of Y, you should do a search from the workbench with the + sign: aspirations+school>trouble.

8. Concluding remarks

At this point you have some idea of how to search, sort, and select. You should also be able to designate some codes as repeat codes to identify contextual information. From here some simple additional things to learn include personalizing the toolbar on results windows, adding and viewing comments in results windows, and exporting data to the clipboard so that it can printed in Microsoft Word, Appleworks, or the word processor of your choice. An even more advanced step would include learning about "reanalysis," i.e., transforming your data files based on your results files. This includes recoding your files (actually changing the code of a given passage) and adding codes around given passages. Finally, there is a lot to be explored in terms of reporting, including simple reports (counts of codes) and complex reports (data summaries and dot graphs).These are all

described in the user guide. Have fun exploring this complex but powerful software program.