TAMS How-To: Calculating Inter-rater Reliability with TAMS Analyzer
(Version 2.22 and later)

TA 2.22 added the ability to calculate inter-rater reliability (IRR) via Cohen's Kappa. To use this feature requires setting up your search list with some concocted files that are used as tests. Here's the basic idea. Two coders want to test their reliability. Someone sets up a test for them to apply codes to. The test consists of a file with different passages that they have to code. A copy of the file is provided to each "contestant," i.e., person checking his/her reliability. So to start with there are two identical files testA.txt and testB.txt for contestants A and B. The files have to be organized as such:

```
passage to code.
{!endsection}
passage to code
{!endsection}
passage to code
{!endsection}
.
.
.
passage to code
{!end}
```

Space doesn't matter. Typically you would probably leave lots of room between sections. You might want to number the passages. That won't effect what TA does. What does matter is the {!endsection} between passages and the {!end} at the bottom (with no !endsection right in front of it). The kappa analyzer does not recognize end of files as the same as {!end}, for instance. The last thing in each of the files must be an {!end}.

The contestants open their files and apply ONE code to each section. It actually doesn't matter where in the section the code is, but whatever the last code TA finds is the one attached to that section. Typically a finished file will look something like

```
{a}passage to code.{/a}
{!endsection}
{b}passage to code{/b}
{!endsection}
{c} passage to code {/c}
{!endsection}
.
.
.
{a}passage to code {/a}
{!end}
```

Though it could look like this, the effect would be the same:

```
{a}passage{/a} to code.
{!endsection}
{b}{/b}passage to code
{!endsection}
{c} passage to code {/c}
{!endsection}
.
.
.
passage to code {a}{/a}
{!end}
```

The kappa for a contestant coding these files would be the same. If the last version and this were tested for inter-rater reliability they would result in a 1.00; a perfect score! The score, not the position of the score, is being compared.

Now is the easy part. Move the two coded files to the search list. The search list must only have these two files in it, then pick **Coding->Report->Inter-rater reliability (kappa)** and voila: a report showing the coding frequency and kappa calculation for these two files.

Final note: I'm unconvinced that IRR tells much, though more positivistic types may like having this. One thing that I think makes a huge difference here is the pool of codes available. I would probably use only a small subset of the total number of codes in a project for testing IRR. I would set up the IRR test as its own project. To import codes from your main project, select the main project; pick **Project->Export codes to code file** and save that, bring it into (**Project->Add File**) the IRR project, select it on the file list and pick **Project->Import codes and defs from code file**. At this point you'll want to use the code browser to disable (uncheck the active box) codes that you do not want coders to have access to. You may also need to create parent codes. In my project I may have many 3 or 4 level codes with no overlord code really being in the project, e.g., I may have animal>type>goat, animal>type>sheep; for the purposes of calculating IRR I may need to get rid of all the subtypes and create one animal>type code representing all of them. **The code list is very important in the calculation: the kappa analyzer uses the code list to make the matrix, not the codes you've used in the files you're coding. So make sure that there is an "appropriate" codelist (appropriate = the codes you want your contestants to use + any others they should have to pick from).**

By the way, the EF column is the expected frequency for the number of agreements that would have been expected by chance for each coding category. It is a key number that differentiates Cohen's Kappa from a straight percent agreement.