ABSTRACT:

Management Assistance Corporation of America (MACA), in contract work for the United States Army Air Defense Artillery Board at Ft. Bliss, Texas, has been using SAS software to automate the scoring of air defense weapon tests. Weapons testing, and the scoring of such testing, operates under a highly complex set of rules set down by a myriad of Army agencies. In addition, testing agencies are under heavily enforced deadlines to produce reliable scored results for each weapon test. Development of automated scoring techniques has often been attempted in more traditional languages, but these had proven to have excessive development time and insufficient flexibility for use in a tightly regulated testing schedule.

To resolve these problems, MACA and the Air Defense Artillery Board used the FORMAT procedure to turn long and complex sets of rules into simple table lookups.

BACKGROUND:

Air defense weapons tests are usually divided into a series of missions, or trials. These trials are periods of time in which a number of potential target aircraft are flown into the weapon's advertised effective range. These trials are further subdivided into presentations, or passes, which consist of the engagement opportunity between the weapon system against each individual aircraft.

Traditional methodology consists of in-depth analysis of each presentation against a weapon system, with results for each becoming a record or observation in a data base, followed by extensive "roll-up" analysis of the entire series of test presentations. Information expected at the end of the test consists around "red attrition" (how many of the enemy were destroyed), "blue survivability" (how often would the weapon system, and its defended assets, survive attack), reliability and maintenance needs, and what man-to-machine interface problems exist, and much more. All results are further subdivided into how well the system performed under various types of conditions such as smoke, bad weather, electronic countermeasures, etc.

As might be expected, SAS software works very well in the production and analysis of these statistics, and its use in weapons testing environments has been proven effective by its successes at the Army Air Defense Artillery Board. One problem which still remained, however, was the automated scoring of each engagement.

GOAL:

During the analysis of each engagement, it was desirable to have a single code, from which it would be possible to describe the entire series of occurrences within an engagement. While it was always possible, using a series of if-then logical statements, to retrieve like-circumstanced data from the database, it was also cumbersome. In order to circumvent this, the practice was to have the analyst who had examined a particular presentation assign a result code to it, known as its "score." Thus a code of "bc", for instance, might mean:

"system detected target within required range using radar, tracked him to within engagement range, correctly identified him as hostile, tracked with infrared, and fired while the target was inbound, but intercept occurred after the loss of a defended asset."

The trouble with analyst assignment of these codes was that a certain amount of interpretation invariably crept in, and some mistakes were made. It was necessary to automate this process to avoid inconsistency.

First approaches to doing this automation were fairly simplistic, and consisted of long series of what inevitably resolved into if-then-else logic. The trouble with these was that they were difficult to write cohesively, branching was cumbersome and hard to follow, and any modification of the scoring rules often would have the effect of causing a complete restructuring of the logic flow. We decided to use a table-lookup methodology, but again, we were stymied by how cumbersome this could become using normal methods as soon as the number of variables contributing to the score grew beyond a dozen or so. Finally, we settled on a methodology that centered around the use of the FORMAT procedure and the PUT function.

METHOD:

The basic method consists of these steps:
1. Collapse all answers to analytical questions into short, one or two byte variables.
2. Construct a truth table or "scoring matrix" to reflect the scoring rules. (If data already exists, this may be an automated procedure.)
3. Construct a "mega" variable consisting of the concatenated variables created in (1) and organized in (2).
4. Create a format using the FORMAT procedure, which translates the "mega" variable into a score.
5. Use the PUT function to call this format in a data step.
A SIMPLE EXAMPLE:

An analyst, when examining a presentation, is answering the following questions:

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>ANSWER</th>
<th>VARIABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Was target detected?</td>
<td>Y</td>
<td>tgtdet=Y</td>
</tr>
<tr>
<td>Target identification?</td>
<td>H</td>
<td>id=H</td>
</tr>
<tr>
<td>Man-machine interface rating?</td>
<td>B</td>
<td>mir = B</td>
</tr>
<tr>
<td>Was trigger pulled?</td>
<td>Y</td>
<td>tp = Y</td>
</tr>
<tr>
<td>Was target intercepted?</td>
<td>Y</td>
<td>intcpt=Y</td>
</tr>
</tbody>
</table>

A proc FORMAT has run prior to this point, which looks about like the one below:

PROCFORMAT;
VALUE $MEGACHK
'YHANN'= '4D'
'YTNNN'= '7F'
'YHAYY'= 'IA'
'YBYBY'= 'IA'

"OTHER='ERROR';

A data step is then run which contains code like that below:

DATA SAMPLE;
SET . . .;
MEGAVAR=
COMPRSS(TGTDET)[ID][MIR][TP][INTCPT];
RESULT=PUT(MEGAVAR,$MEGACHK.);
more code . . ;

The value of megavar in our example would be 'YBYBY' which the format called $MEGACHK resolves into 'LA'. Scoring of this presentation is achieved.

Of course, real life examples are much more complex, but the procedure is the same.

A MORE COMPLICATED EXAMPLE:

One of the limitations of the FORMAT procedure is that there is only room for sixteen characters on the left side of the equal sign. Although this might seem to be a problem for our use (we will certainly encounter rules based upon more than sixteen variables), it is easily overcome by branching techniques whereby a single letter within your final mega variable is representative of up to sixteen or more previously collapsed variables.

Let us assume that four or five questions on an analyst’s worksheet have to do with the man-machine interface.

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>ANSWER</th>
<th>VARIABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Could the soldier see the target on his radar?</td>
<td>Y</td>
<td>rdrtgt=Y</td>
</tr>
<tr>
<td>Did the radar officer properly alert the gunner?</td>
<td>Y</td>
<td>alert =Y</td>
</tr>
</tbody>
</table>

Was there any trouble reaching the controls? N  contl=N
Did the gunner acknowledge target as hostile? Y  tgtatn=Y
Were all the soldiers happy campers? N  happy=Y

Now our Proc FORMAT looks like this:

PROCFORMAT:*an intermediate format is created;
VALUE $MINIClk
'YYPY'= 'A'
'YTNN'= 'B'
'YTNN'= 'C'
'YTNN'= 'D'
'OTHER='ERROR';

which will be used to feed this one.

VALUE $MEGACHK
'YHANN'= '4D'
'YTNN'= '7F'
'YHAYY'= 'IA'
'YBYBY'= 'LA'

The variable minimeg has the value in our example of 'YYPY' which resolves into a 'B' in the variable MIR. This then becomes 'YYBY' in the mega variable megavar, which resolves into 'LA'. Note that there can be more than one way to arrive at a result on the right side of the equal sign in our formats.

As you can see, a long cascade of such formats would allow you to accurately describe an incredible variety of situations with very few characters. Although not shown here, numeric range data is easily converted into character data by the FORMAT procedure, and is thus easily assimilated into this process.

PRECAUTIONARY NOTE:

It is necessary to insure that any variables feeding the mega variables have some value assigned to them if the compress function is used. This is because the compress function will eliminate blanks from your variable.

SOME FEATURES OF THIS METHODOLOGY:

1) The programs run very quickly.
A) Proc FORMAT is one of the fastest ways to do table lookup within SAS software.
B) This becomes even faster if the formats are saved permanently.

2) Development of the program is very fast.
A) If you already have some data, you can use it to help write its own "proc format."
B) No more long strings of if-then-else logic to wade through.

3) It's easy to modify, without messing up the rest of the program.
A) Additional rules are simply added.

4) It eliminates possible ambiguities.
A) While Proc FORMAT will allow duplicate values to the right of the equal sign, duplicates on the left, which indicate ambiguities, are not tolerated, and the format will not be output.
B) This is one of the most common errors in long series of if-then-else logic — allowing a set of "if" conditions to be duplicated with a different set of "then" parameters overwriting an earlier set.

5) It allows the inclusion of these formats in other SAS procedures and products, such as FSCALC and SAS/AF, allowing "instant" results as input is added.

6) Unpredicted combinations of input are flagged as errors for later examination.

ACKNOWLEDGEMENTS:

SAS, SAS/FSIP and SAS/AF are registered trademarks of SAS Institute, Inc., Cary, NC, USA.

I would like to thank James Wilcott, Rafael Ramirez, and Jeanette Lewis for their critiques of this paper.

This paper does not necessarily reflect the views of the U.S. Army Air Defense Artillery Board. Therefore, no official endorsement should be inferred.

CONTACT AUTHOR:

Tony N. Lewis
Management Assistance Corporation
of America (MACA)
8600 Boeing Drive
1 MACA Plaza
El Paso, Texas 79925
(915) 778-1359