Checking Multiple Values for Inequality

Stephen R. Schulz, Consultant, San Francisco, CA

Abstract

Consider the case where there are more than two potential sources for the value of a field. For example, there are three files contributing potential values for a field and each may or may not have missing values. The code to check for missing values and disagreement among non-missing values can get complicated. This paper offers a simple solution to this problem if the fields are numeric and a possible line of attack for character fields.

The Problem

There are three different files (actually forms coming from different departments) with fields that may or may not provide a value for a unit of measure code (UOM). If we assume the data are all correct, we can obtain the value simply with the following statement:

```
UOM = max (of UOMa, UOMb, UOMc);
```

This easily solves the problem of checking each field to see if it is non-missing and then assigning one of the non-missing values to the output field.

Brute Force Solution

Unfortunately, the incoming fields may disagree. The code to check for disagreements among non-missing fields makes the code somewhat more complex.

```
if UOMa ne . then do;
  if UOMb ne . then
    if UOMa ne UOMb then do;
      error processing;
    end;
  if UOMc ne . then
     if UOMa ne UOMc then do;
      error processing;
     end;
end;
if UOMb ne . and UOMc ne . then
  if UOMb ne UOMc then do;
      error processing;
  end;
```

The problem is that we don't want missing values to be included in the comparisons.

An Easier Solution

There is a much easier way to check for equality, however, using the STD function to compute the standard deviation of the values.

```
if std (UOMa UOMb UOMc) then do;
      error processing;
end;
```

Generally speaking, there are four cases: For 3 of these the STD function returns either 0 or missing. 1) If all values are missing or 2) if only 1 value is non-missing, no standard deviation can be computed and so missing is returned. 3) If there are 2 or 3 non-missing values and they are all equal, 0 is returned. 4) If there are 2 or 3 values but they do not all equal each other a non-zero value is returned. Since in SAS® 0 and missing are Boolean false values and all non-zero numbers are true, we have our solution. After all a standard deviation is telling us how much variation there is among a set of numbers. Note that this technique works just as well for any number of fields. The brute force method is a little long for 3 fields. For more than 3 it gets much longer.

Processing

So, what do we do with this information now that we have it. In our case, we are building a summary file every night from the same set of source files, so error processing consists only of printing an exception report showing all three values. It is then up to a human being to determine which files are in error and fix them. There is no particular reason for us to believe one input file more than any other, so using the max of the 3 values works for us as the value to place in our summary file. Once the input files are corrected, the summary file is corrected by the next nightly run.

What about character strings?

This algorithm can be extended to character strings if the character strings can be encoded to a number that is unique for each possible value of the string. A simple macro named ENCODE to do this is displayed in Figure 1. This routine is very simple. It only handles blanks and uppercase alphabetic, but it could easily be extended to include more characters if desired. A more dangerous weakness is that there is no error handling to check for characters not in the translate table, but again this would be a simple fix using the verify function. It is also limited to strings of no more than 8 bytes of length. This is because each character is being recoded to 2 digits in a number and there is a limit to the maximum precision of numbers that can be stored in SAS. This maximum varies by platform On our system, sixteen digits is the most that can be precisely stored. Probably there are many more sophisticated methods to use for the encoding process, but this one will

suffice for an example. The code to use it could look like this:

```
..%encode (string1)
..num1 = __num__;
..%encode (string2)
..num2 = __num__;
..%encode (string3);
..num3 = __num__;
..if std (of num1 - num3) then do;
..    error handling;
..end;
```

Another Solution

Like most other problems in SAS, there are many ways to attack this problem. One that is almost as simple as the one proposed is to use the MAX and MIN functions.

```
if min(UOMa UOMb UOMc) ne
    max(UOMa UOMb UOMc) then do;
  error processing;
end;
```

This method has all the advantages handling missing values as the original solution and may be more intuitive to some users.

Conclusion

Given that a standard deviation is a measure of the variation in a set of numbers, the STD function is the ideal tool to accomplish this task simply and elegantly. There is much less possibility of making a coding error using this tool than the *brute force* method.

Steve Schulz
3 Burnett N, #1
San Francisco, CA 94131

(415) 282-8373
Internet: srs2@pge.com

SAS is a registered trademark of SAS Institute Inc.
© indicates USA registration.

Figure 1. The ENCODE MACRO

```
%* Data step variables used only in this MACRO are given names preceded
%* and followed by 2 underscores to prevent conflict with program
%* variables and drop them;

%macro encode (string);

     length __str__ $16;
     drop __str__ __num__ __i__ __p__;

     do __i__ = 1 to length (&string);
        __p__ = (__i__ * 2) - 1;
        substr (__str__, __p__, 1) =
          translate (substr (&string, __i__, 1),
                             '00000000011111111122222222',
                             ' ABCDEFGHIJKLMNOPQRSTUVWXYZ');
        substr (__str__, __p__+1, 1) =
        translate (substr (&string, __i__, 1),
                           '12345678901234567890123456',
                           ' ABCDEFGHIJKLMNOPQRSTUVWXYZ');
     end;
     __num__ = input (__str__, 16.);
%mend encode;
```