GEMS IN THE FORMAT PROCEDURE

Aileen L. Yam, G. H. Besselaar Associates, Princeton, NJ

ABSTRACT

This paper makes use of gems in the FORMAT procedure to obtain efficiency, precision and flexibility in programming. A real-world application in the pharmaceutical industry is provided to illustrate that creative use of the VALUE statement in the FORMAT procedure together with the ROUND function has the benefit of attaining efficiency and the precision in unit conversion that are otherwise obtained with many IF-THEN statements and clumsy computation. In version 6 of the SAS® System, the flexibility and efficiency of creating and maintaining user-defined formats are expanded by the CNTLIN option. The application is further enhanced with the CNTLIN option.

INTRODUCTION

How do you get the same results with less time and more flexibility?

This paper describes an application in the pharmaceutical industry of the resourceful features of the FORMAT procedure. First, a programming problem is defined. Then, two alternatives to address the same data management functions are presented: one is the conventional IF-THEN programming statements, the other is creative use of the VALUE statement in PROC FORMAT together with the ROUND function, as well as the application of the CNTLIN option in PROC FORMAT. The second approach requires less tedious work and gives more flexibility to the process of maintaining the codes.

PROGRAMMING PROBLEM

The following is a list of criteria to subset patients into two groups for subgroup analysis in an integrated hypertension study.

The list consists of heights in feet and inches and weights in pounds for male and female patients. For any given height, patients whose weights fall within the ones specified on the list are not considered to be overweight, while patients whose weights are beyond the limits specified on the list are considered overweight.

For a male patient to be considered overweight, the criteria are:

<table>
<thead>
<tr>
<th>Height (feet)</th>
<th>Weight (lbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5'2&quot;</td>
<td>&gt;=180 lbs</td>
</tr>
<tr>
<td>5'3&quot;</td>
<td>&gt;=184 lbs</td>
</tr>
<tr>
<td>5'4&quot;</td>
<td>&gt;=187 lbs</td>
</tr>
<tr>
<td>5'5&quot;</td>
<td>&gt;=192 lbs</td>
</tr>
<tr>
<td>5'6&quot;</td>
<td>&gt;=197 lbs</td>
</tr>
<tr>
<td>5'7&quot;</td>
<td>&gt;=202 lbs</td>
</tr>
<tr>
<td>5'8&quot;</td>
<td>&gt;=206 lbs</td>
</tr>
<tr>
<td>5'9&quot;</td>
<td>&gt;=211 lbs</td>
</tr>
<tr>
<td>5'10&quot;</td>
<td>&gt;=216 lbs</td>
</tr>
<tr>
<td>5'11&quot;</td>
<td>&gt;=221 lbs</td>
</tr>
<tr>
<td>6'0&quot;</td>
<td>&gt;=226 lbs</td>
</tr>
<tr>
<td>6'1&quot;</td>
<td>&gt;=230 lbs</td>
</tr>
<tr>
<td>6'2&quot;</td>
<td>&gt;=236 lbs</td>
</tr>
<tr>
<td>6'3&quot;</td>
<td>&gt;=242 lbs</td>
</tr>
<tr>
<td>6'4&quot;</td>
<td>&gt;=248 lbs</td>
</tr>
</tbody>
</table>
For a female patient to be considered overweight, the criteria are:

<table>
<thead>
<tr>
<th>Height (in feet and inches)</th>
<th>Weight (in pounds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4'10&quot;</td>
<td>&gt;=157 lbs</td>
</tr>
<tr>
<td>4'11&quot;</td>
<td>&gt;=160 lbs</td>
</tr>
<tr>
<td>5'0&quot;</td>
<td>&gt;=164 lbs</td>
</tr>
<tr>
<td>5'1&quot;</td>
<td>&gt;=168 lbs</td>
</tr>
<tr>
<td>5'2&quot;</td>
<td>&gt;=172 lbs</td>
</tr>
<tr>
<td>5'3&quot;</td>
<td>&gt;=176 lbs</td>
</tr>
<tr>
<td>5'4&quot;</td>
<td>&gt;=181 lbs</td>
</tr>
<tr>
<td>5'5&quot;</td>
<td>&gt;=186 lbs</td>
</tr>
<tr>
<td>5'6&quot;</td>
<td>&gt;=191 lbs</td>
</tr>
<tr>
<td>5'7&quot;</td>
<td>&gt;=196 lbs</td>
</tr>
<tr>
<td>5'8&quot;</td>
<td>&gt;=200 lbs</td>
</tr>
<tr>
<td>5'9&quot;</td>
<td>&gt;=204 lbs</td>
</tr>
<tr>
<td>5'10&quot;</td>
<td>&gt;=208 lbs</td>
</tr>
<tr>
<td>5'11&quot;</td>
<td>&gt;=211 lbs</td>
</tr>
<tr>
<td>6'0&quot;</td>
<td>&gt;=215 lbs</td>
</tr>
</tbody>
</table>

A CUMBERSOME SOLUTION

Intuitively, the problem seems to be fairly simple. All that one needs to do is to type in two sets of IF-THEN statements, one for male patients, the other for female patients. If the list of criteria is not too long, such as the one given above where there are 15 different height and weight criteria for male patients, and 15 for female patients, typing in 30 IF-THEN statements is a reasonable job.

However, the criteria of height are in feet and inches and the criteria of weight are in pounds, while the data for subgroup analysis have the respective measurements in centimeters and kilograms. The units in feet have to be converted into inches, and the inches are added up to be converted into centimeters. The units in pounds have to be converted into kilograms. Sixty different numbers with decimals to be entered manually into 30 IF-THEN statements can be troublesome.

A closer look at the data shows that 60 numbers are not enough. Instead, 90 different numbers with decimals are needed to be entered by hand into 30 IF-THEN statements, because heights have to be entered as ranges rather than as single numbers to preserve accuracy after unit conversion.

An even closer look at the data shows that a direct conversion of height from inches into centimeters to obtain the ranges of height can create precision problems. For example, in setting up an IF-THEN condition for 5'4", to state that if a patient is taller than 5'3" and shorter than or equal to 5'4" can put the patient into the wrong weight group. The reason is that 5'4" actually means between 5'3.5" and 5'4.4", whereas taller than 5'3" and shorter than or equal to 5'4" means between 5'3.1" and 5'4.0". Since centimeter has finer gradations than inch, the variations even in decimals can create precision problems after conversion. Therefore, instead of getting the ranges of height directly from the criteria list, all the height criteria have to be put into a lower limit and an upper limit before the conversion. Height is first separated into two components, $X_1$ for feet and $X_2$ for inches. $X_1$ is multiplied by 12 to obtain inches, and the inches are added up to be converted into centimeters. The units in pounds have to be converted into kilograms. Sixty different numbers with decimals to be entered manually into 30 IF-THEN statements can be troublesome.

To outline the steps programmatically:

1. WEIGHT CONVERSION—convert male and female weights given in the criteria from pounds into kilograms.

   \[
   \text{MWTKG} = \text{MWTLB} \cdot 0.453; \\
   \text{FWTKG} = \text{FWTLB} \cdot 0.453; \\
   \]

2. HEIGHT CONVERSION—set up the lower limit and the upper limit heights from the criteria for male and female patients and convert the ranges of height from inches into centimeters.

   \[
   H_{L} = [X_{1} \cdot 12 + (X_{2} - 0.5)] \times 2.54; \\
   H_{U} = [X_{1} \cdot 12 + (X_{2} + 0.4)] \times 2.54; \\
   \]
3. ESTABLISHING SUBGROUPS—enter the converted weight and height values obtained from the previous two steps into IF-THEN statements.

\[
\text{IF SEX=‘MALE’ THEN DO;}
\]
\[
\begin{align*}
&\text{IF 156.210<HT<158.46 AND} \\
&\text{WT>81.540 THEN OVERWT=1;} \\
&\text{(IF-THEN statements for all different criteria for male patients)}
\end{align*}
\]
\[
\text{ELSE OVERWT=0; END;}
\]
\[
\text{ELSE DO;}
\]
\[
\begin{align*}
&\text{IF 146.050<HT<148.336 AND} \\
&\text{WT>71.121 THEN OVERWT=1;} \\
&\text{(IF-THEN statements for all different criteria for female patients)}
\end{align*}
\]
\[
\text{ELSE OVERWT=0; END;}
\]

After 90 different numbers with decimals are meticulously typed in and verified, the subgroups are defined. But what if the criteria for defining the subgroups change and another programmer has to maintain the program while the programmer who wrote the program is away on vacation? Detailed documentation is needed on the logic and the method for creating the height conversion ranges and weight conversion numbers. The programmer responsible for maintaining the program has to read the document, restart the process of typing and verifying all the new numbers.

There are times when lengthy and straightforward solutions like the one above are fine. But there are also times when such solutions entail a good deal of detailed and error-prone work, and more elegant solutions are preferable.

A REFINED SOLUTION

The key is to find a way not only to come up with the correct height conversion but also to automate the process.

Since the VALUE statement of PROC FORMAT allows the specification of formats for different values, the first step is to simplify the height values to range from LOW-1 to 15-HIGH for both male and female patients, instead of from 5'2" to 6'4" for male patients and from 4'10" to 6'0" for female patients, then put the new height ranges on the left side of the equal sign of the VALUE statement, and put the converted weight values on the right side of the equal sign to associate weight values with height specifications. Height values are simplified into the same ranges for both male and female patients for two reasons: one is to obtain the correct conversion, the other is so that macro calls can be set up to invoke the same programming statements for both male and female patients.

The criteria of height make the process of simplification easy because the interval between each height value, from 5'2" to 6'4" for male patients and from 4'10" to 6'0" for female patients, is equal, that is, each height value differs from the next height value by one inch. Recall that the height data for subgroup analysis are in centimeters. For all male patients, subtract 154.94, i.e. 5'1", from their actual height values, divide the results of the subtraction by 2.54, and round off the results to the nearest integer. Similarly, for all female patients, subtract 144.78, i.e. 4'9", from their actual height values, divide the results of the subtraction by 2.54, and round off the results to the nearest integer. Consequently, the male and female height data are transformed to range from less than or equal to 1 to equal to or higher than 15. This method of transformation works out very well for centimeter-to-inch conversion. Precision is retained after
conversion by making good use of the ROUND function. After the height values are subtracted and converted, the built-in mechanism of the ROUND function automatically puts the patients into the correct categories. The programming statements are:

```
IF SEX='MALE' THEN C_HT=ROUND((HT-154.94)/2.54);
ELSE C_HT=ROUND((HT-144.78)/2.54);
```

The FORMAT procedure can be set up as follows:

```
PROC FORMAT;
VALUE MWT
LOW-1 = '81.540'
  2 = '83.352'
 .
15-HIGH = '112.344';
VALUE FWT
LOW-1 = '71.121'
  2 = '72.480'
 .
15-HIGH = '97.395';
```

Once the formats are set up, only two IF-THEN statements remain to complete the program. In the IF-THEN conditions, use the PUT statement to associate patients' heights with weight criteria, and compare the criteria with their actual weights to determine subgroups:

```
IF SEX='MALE' THEN DO;
  IF WT>=PUT(C_HT,MWT.) THEN OVERWT=1; ELSE OVERWT=0;
END;
ELSE DO;
  IF WT>=PUT(C_HT,FWT.) THEN OVERWT=1; ELSE OVERWT=0;
END;
```

By creatively fitting the data into the requirements of the VALUE statements, there is no longer a need to deal with the upper limit or lower limit height conversion values. Only 30 numbers are needed to be entered into the FORMAT procedure, and the number of IF-THEN statements is reduced from 30 to two, while the precision problem is taken care of mechanically. With version 5 of the SAS® System, that is as much as one can do to reduce the intricacy and the size of the program.

### ENHANCEMENT WITH VERSION 6 SAS SOFTWARE

With version 6 of the SAS System, the program can be further automated with the CNTLIN option in PROC FORMAT. The CNTLIN option creates formats from a control data set. A control data set can be created from either an existing SAS data set or from a DATA step as long as the data are made to conform to the control data set structure. In other words, the CNTLIN option saves the trouble of having to manually enter the values and label specifications in the VALUE statement of PROC FORMAT. The control data set serves as input for the values and label specifications.

The weight values specified in the criteria for male and female patients are in pounds. The values are converted into kilograms by multiplying by .453.

To create a control data set for the height and weight formats, the following variables are needed:

a. FMTNAME—a format name.
b. START—starting values for the ranges.
c. END—ending values for the ranges.
d. LABEL—the formatted specifications associated with the ranges.
e. HLO—if there is no special value for a range, HLO is blank; however, if there is a special value for a range such as LOW, HIGH, LOW-HIGH, or OTHER, the special value is defined in HLO.
An additional variable, TYPE, is usually needed in setting up a control data set. TYPE specifies format type; it can be character, numeric, picture, format or informat. TYPE can be omitted if the format is numeric, because the default type is numeric.

The following statements create the format MWT for male patients from the control data set MWTFILE, as well as the format FWT for female patients from the control data set FWTFILE.

```python
%MACRO WTFILE
  (CNTFILE~,FMTVAR~,WTVAR~);
DATA &CNTFILE (KEEP=FMTNAME START END &WITAR LABEL HLO);
SET PERM.CRITERIA END~EOF;
  RETAIN FMTNAME "&FMTVAR~";
  LENGTH START END $4;
  LABEL=PUT(&WTVAR,8.3);
  IF _N_=1 THEN DO;
    START=LOW; END=PUT(_N_,A);
    HLO='L';
  END;
  ELSE IF EOF THEN DO;
    START=PUT(_N_,A); END=HIGH;
    HLO='H';
  END;
  ELSE DO;
    START=PUT(_N_,A); END=PUT(_N_,A);
    HLO=' ';
  END;
PROC PRINT DATA=&CNTFILE;
  TITLE "CONTENTS OF &CNTFILE";
PROC FORMAT LIBRARY=LlBRARY
  CNTLIN=&CNTFILE;
%MEND WTFILE;
%WTFILE(CNTFILE=MWTFILE,FMTVAR =MWT,WTVAR=MWTKG);
%WTFILE(CNTFILE=FWTFILE,FMTVAR =FWT,WTVAR=FWTKG);
```

The RETAIN statement initializes and assigns the variable FMTNAME a unique format name. The format name remains the same throughout the DATA step. The LENGTH statement sets the lengths for the variables START and END, and establishes START and END as character variables because of the special range values, LOW and HIGH, in the first and last ranges. The LABEL= statement assigns formatted values for the variable LABEL.

To build the value ranges for the format, three conditions are needed. The first condition where _N_=1 creates the first range. The first range is from LOW to 1. The START value is set to LOW, the END value is set to _N_. The automatic variable _N_ is used because the range value corresponds to the observation number in the criteria. Since the special range value LOW is assigned to the variable START, the HLO variable needs to be set to L to indicate the use of the special range value. The second condition, IF EOF, sets up the last range to be from 15 to HIGH. The START value is set to _N_ which corresponds to the observation number in the criteria. The END value is set to HIGH. The HLO variable is set to H to indicate the use of the special range value HIGH. The third condition is to create the rest of the ranges between the first and the last ones. The remaining ranges are 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14. Since those range values also correspond to the observation numbers in the criteria and the starting range values are the same as the ending range values, _N_ is used for both START and END.

The PRINT procedure prints out the contents of the control data sets.

The FORMAT procedure builds formats from control data sets.

The PRINT procedure helps to verify the control data sets before the formats are created. To verify the formats after they have been created, there is an option in PROC FORMAT. The option is FMTLIB:

```python
PROC FORMAT LIBRARY=LlBRARY FMTLIB;
  SELECT MWT FWT;
```
The SELECT statement can be omitted if all the formats in a format library are to be printed out for verification.

To outline the steps programmatically:

1. **WEIGHT CONVERSION**—convert male and female weights specified in the criteria from pounds into kilograms.
   
   \[
   \begin{align*}
   MWTKG &= MWTLB \times 0.453; \\
   FWTKG &= FWTLB \times 0.453; 
   \end{align*}
   \]

2. **CNTLIN OPTION**—create formats with control data sets from the criteria as exemplified in the macro WTFILE above.

3. **ESTABLISHING SUBGROUPS**—convert height data with the ROUND function, associate the converted height with the converted weight criteria, and compare the criteria with actual weight data.
   
   \[
   \begin{align*}
   &\text{IF } \text{SEX} = '\text{MALE}' \text{ THEN } \\
   &\quad \text{CHT} = \text{ROUND}((\text{HT} - 154.94)/2.54); \\
   &\text{ELSE } \text{C}_\text{HT} = \text{ROUND}((\text{HT} - 144.78)/2.54); \\
   \\
   &\text{IF } \text{SEX} = '\text{MALE}' \text{ THEN DO;} \\
   &\quad \text{IF WTW} = '\text{PUT(C,HT,MWT.)} \text{ THEN} \\
   &\quad \quad \text{OVERWT} = 1; \text{ ELSE } \text{OVERWT} = 0; \\
   &\quad \text{END;} \\
   &\text{ELSE DO;} \\
   &\quad \text{IF WTW} = '\text{PUT(C,HT,FWT.)} \text{ THEN} \\
   &\quad \quad \text{OVERWT} = 1; \text{ ELSE } \text{OVERWT} = 0; \\
   &\quad \text{END};
   \end{align*}
   \]

The same question is asked: what if the criteria for defining the subgroups change and another programmer has to maintain the program while the programmer who wrote the program is away on vacation? The answer is simple: the maintenance process is automatic. The reason is that the height criteria are not likely to change; height values always increment by one. The changes in weight criteria will be input directly from a data set, and the CNTLIN option in PROC FORMAT builds formats from the control data sets with programming statements already written. Therefore, the programmer responsible for maintaining the program simply has to rerun the program to get the new results without having to understand the whole process.

**SUMMARY**

Gems in the FORMAT procedure refer to the use of the VALUE statement and the CNTLIN option in PROC FORMAT, in combination with the ROUND function. This paper shows that with the gems in the FORMAT procedure, a lot of cumbersome and error-prone work is saved in the initial entering of numbers into the lengthy list of IF-THEN statements, the verification of numbers, the documentation of programs, the process of having to enter and verify the numbers again whenever data are updated. Program creation becomes simplified and yet accurate, and program maintenance becomes automatic, which are translated into fewer human and computer resources.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

For additional information, contact:

**Aileen L. Yam**

G. H. Besselaar Associates

103 College Road East

Princeton, NJ 08540-6681

Tel.: (609) 452-8550