

Your Friend – Custom Formats

Ron Richardson, Regions Bank, Birmingham, AL

ABSTRACT

The SAS® programmer's life is a challenging one. Having the right tools to deal with these challenges is often just a matter of knowing what's available. Of the many tools SAS offers, the "custom format" is both powerful and flexible. This paper will demonstrate some custom format basics and illustrate some interesting ways to use them to tackle your programming challenges.

INTRODUCTION

The purpose of this paper is to discuss some different ways to leverage custom formats in your daily work. We will look at two ways of creating a custom format; the list and the data driven. Also, we will discuss methods to deploy your own custom formats in your datasteps. Lastly we will examine the improvements in performance gained from using different techniques.

CREATING THE CUSTOM FORMAT

There are at least two ways to create the custom format. We'll call them the "List" and the "Data" types. In a list type format, the values driving the format are listed in the code like below:

```
proc format lib=PermLib2 ;
  value FICO_example
    low - 0 = 'Unknown'
    0< - 639 = '001-639'
    639< - 659 = '640-659'
    659< - 679 = '660-679'
    679< - 699 = '680-699'
    699< - 719 = '700-719'
    719< - 739 = '720-739'
    739< - 759 = '740-759'
    759< - 779 = '760-779'
    779< - 799 = '780-799'
    799< - 999 = '800-999'
  other = 'Unknown';
run;
```

This is a format to group numerical data into ranges useful for coarse classing data. You'll notice here we are converting number variables into character data.

Frequently the custom format will be stored in a permanent library where it can be used multiple times by various programs. As a reminder of what the format does and how it is used, include the following code to add an explanatory description that will display in the SAS explorer view of the formats.

```
proc catalog catalog = PermLib2.formats ;
  modify FICO_Example.format
  (description='Groups FICO scores into meaningful ranges');
quit;
```

When using custom formats stored in a permanent library, you must instruct the SAS session on where to find your format. This is done with the following entry in an options statement:

```
Options fmtsearch=(PermLib1 PermLib2);
```

Notice how two libraries are referenced in the search path with a space inserted between.

In the Data format type, the values are stored in a control dataset. This example below shows how to build a dataset that can drive a custom format:

```
data work.CustomFormatData;
  set LibRef.LocationList;
  retain fmtname "$ValidLocations";
  label = "KEEP"; /* note: the label is a selection key value */
  start = LocationCode;
run;
```

This format is designed to be used as part of a selection criteria. These few lines are all that is required to convert the data into a custom format:

```
proc format library = work
             cntlin = work.CustomFormatData;
run;
```

Another feature of the data driven custom format handles the "out-of-range" observations. By adding a few features to the dataset above, we can group all the unknown locations together for ease of analysis. Of course a simple label of "KEEP" isn't very helpful. So in this example, the label is the value in the variable LocationName.

```
data work.CustomFormatData;
  set LibRef.LocationList (end=LastObs);
  retain fmtname "$LocationNames";

  /* note: here we use the variable as the return value for the format */
  label = LocationName;
  start = LocationCode;
  output;
  if LastObs then do;
    HLO = "0";
    Label = "Unknown";
    Output;
  end;
run;
```

This format is designed to be used like a look-up table.

DEPLOYING THE CUSTOM FORMAT

With your new list or data driven custom format in hand, the next step is to leverage this technique in your code. Of all the ways to use a format in SAS code, these two are favorites because of their speed and transparency. The first method, translating an existing variable into a new value (i.e. like a lookup table) is useful for coarse classing data and examining outliers.

```
data work.OutPutData;
  set LibRef.InputData;

  GroupFICO = put(FICO,FICO_Example.);

run;
```

Simply sorting the dataset by the new "GroupFICO" variable will bring all the unexpected observations together where GroupFICO = "Unknown". This is very useful when dealing with dirty data where having all the observations with issues grouped together can aid in discovering commonalities leading to issue resolution. That is, grouping all the observations with Unknown FICO scores might show they all came from the same location (where data entry had been entrusted to the new employee).

The second method illustrated is to use the custom format as part of your selection criteria. You might think of the custom format as either a selection or a suppression list. First as a selection criteria.

```
data work.SubsetData;  
  set LibRef.InputData;  
  
  where put(LocationCode,$ValidLocations.) eq "KEEP";  
  
run;
```

In the example above, we want only the observations with a valid location code. Suppose you want to examine only the records without a valid location code. In this case, the selection criteria might read:

```
where put(LocationCode,$ValidLocations.) ne "KEEP";
```

Another use for a custom format in a selection criteria, is the suppression list. In this case, you would have a list of customers and accounts that have opted out of marketing campaigns. The format would have a HLO feature but the label there might be a null or a keyword like "EXCLUDE". In this example you might have a selection criteria like:

```
where put(CustomerAccount,$CustomerOptOut.) eq "EXCLUDE";
```

As you can tell, there are many ways to work with the custom format to achieve the desired result.

The custom format techniques above are valuable not only for the clarity in which the code reads, but also for the speed in which it executes. The binary search SAS uses on the data and the format result in a high level of performance.

CONCLUSION

The custom format is a very flexible and powerful tool in the SAS programmer's arsenal. It can serve as a permanent and easily utilized look-up table. You can use it as a selection or suppression criteria. There is value added in the speed in which it executes. With a little creativity, you can combine techniques like concatenation of key variables to extend its usefulness. Lastly it is possible to "chain" custom formats in a "PUT" statement to perform multiple conversions in a single statement. For all these reasons, I highly recommend you keep the custom format in the top drawer of your SAS tool chest.

REFERENCES

SAS programming III: Advanced Techniques and Efficiencies. Copyright 2006 by SAS Institute, Cary, NC 27513.

ACKNOWLEDGMENTS

Brian Buzbee and Jim Snider provided inspiration for this paper. The fine people at SAS technical support were invaluable in translating problems into solutions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ron Richardson
Regions Bank
2050 parkway Office Circle
Hoover, AL 35244
Work Phone:
E-mail: ron.richardson@regions.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.