

Paper CC-019

Formats - Let the Data Do It
Janet Stuelpner, Left Hand Computing, Inc.

ABSTRACT

Would you like your formats to be dynamic? Do you want the data to create the format for you? Using the CNTLIN feature of PROC FORMAT, you can do just that. It is quick and easy and you don't need a whole bunch of information in your dataset to create the format. This presentation will show you just how much data you need and the details of how to do it.

INTRODUCTION

The data has been entered into a dataset. It has been reviewed and decided that this data would make a great format. You wonder what information is needed to create that format. One way to tell what kind of information that you have available to you in a format is to create a dataset from an existing format. That may give you a hint as to the kind you can put into the format. It will also tell you the variables that are needed for the format. Once you have the information, you can now create your dataset and create your format.

GETTING THE INFORMATION

The example for this paper is a short format as can be seen below. Many of the formats that we write are much longer. This short example will provide a great deal of information to explain how this all works. Let's first see what kinds of information you can retrieve from the format. Let's use a very simple format.

0-2	Baby
3-12	Child
13-19	Teen
20-64	Adult
65-90	Senior

In this format, there are categories for groups of ages of people. As you can see, there is a definite beginning and end to each range. The name of the format will be 'PERSON'. This will be defined as a numeric format with a range of values. If the code needed to be written for this format it would look like this:

```
proc format library=work;
  value person 0-2      =' Baby'
              3-12     =' Child'
              13-19    =' Teen'
              20-64    =' Adult'
              65-90    =' Senior' ;
run;
```

In order to look at the kinds of information that you can place in a dataset, it would be nice to see the information that is provided to you if you take a format that exists and retrieve its contents into a dataset. The code is very simple.

```
proc format library=work cntlout=person;
  select person;
run;
```

Here is just a small selection of variables that are available in the dataset PERSON that is created from the code above.

FMTNAME	START	END	LABEL	TYPE	LENGTH	SEXCL	EEXCL	HLO
PERSON	0	2	Baby	N	6	N	N	
PERSON	3	12	Child	N	6	N	N	
PERSON	13	19	Teen	N	6	N	N	
PERSON	20	64	Adult	N	6	N	N	
PERSON	65	90	Senior	N	6	N	N	

Let's see what the minimum amount of information is needed to create the format in the dataset.

THE MINIMUM

The first piece of information that we need is the name of the format. In this case, as was stated above, the format is named 'PERSON'. We will need to put that value into a variable called FMTNAME. If this format is a character format, the name of the format needs to start with a dollar sign (\$). Since we are creating a numeric format, the \$ is not necessary. The next item to consider is the range of values. The beginning of the range must be a variable called START and the end of the range must be a variable called END. One thing to note is that even though this is a numeric format, the variable type for START and END can be character. If the value for the format is not a range, then the value of the variable START will be the same value in the variable END. Now, the only thing left is the coded value that will be written in the output if the format is used. The variable for that must be called LABEL. Believe it or not, that is all that you need. For the format in our example, we would only need the first four columns in the table above.

To create the format from the dataset PERSON, the code would simply look like this:

```
proc format library=work cntlin=person; run;
```

THE ENHANCEMENTS

There are many more variables that can be added to the dataset. Rather than specify what type of format we are defining by using the \$ in the name of the format, a variable named TYPE can be added with a value of N for numeric or C for character. There are many other values to create informats and pictures, but that is beyond the scope of this paper. The variable HLO can be added to the dataset to denote a value of HIGH, LOW or OTHER. For example, if we wanted to leave the upper limit for SENIOR to be open-ended (more and more people are living to an age of 100 or more), the END value for SENIOR could be missing and the value in HLO would be HIGH. If we wanted to put in an error condition (e.g., a typo is made and negative numbers are values in the dataset) then we can add a row in the dataset where HLO has a value of O and the LABEL='Wrong Value'. This would be equivalent to the PROC FORMAT code below

```
.
proc format library=work;
  value person 0-2      ='Baby'
              3-12     ='Child'
              13-19    ='Teen'
              20-64    ='Adult'
              65-HIGH  ='Senior'
              Other    ='Wrong Value';
run;
```

The dataset that would be needed would look like this:

FMTNAME	START	END	LABEL	TYPE	HLO
PERSON	0	2	Baby	N	
PERSON	3	12	Child	N	
PERSON	13	19	Teen	N	
PERSON	20	64	Adult	N	
PERSON	65	HIGH	Senior	N	H
PERSON	OTHER	OTHER	Wrong Value	N	O

There are variables SEXCL and EEXCL (which have a value of 'Y' for yes or 'N' for no) that allow us to specify that the range does not include the endpoint. For example, a baby could be up to but not including the age of two and the variable EEXCL would equal Y. It actually looks a bit strange as it looks as if there are overlapping values. In this case we are saying that a BABY is up to but not including two. Once the person has a birthday at the age of two, they are then considered to be a CHILD. The typical code to create this would look like this:

```
proc format library=work;
    value person 0-<2    ='Baby'
                2-12    ='Child'
                13-19   ='Teen'
                20-64   ='Adult'
                65-HIGH ='Senior'
                Other    ='Wrong Value' ;
run;
```

The dataset that would be needed to imitate the code above would look like this:

FMTNAME	START	END	LABEL	TYPE	SEXCL	EEXCL	HLO
PERSON	0	2	Baby	N	N	Y	
PERSON	2	12	Child	N	N	N	
PERSON	13	19	Teen	N	N	N	
PERSON	20	64	Adult	N	N	N	
PERSON	65	HIGH	Senior	N	N	N	H
PERSON	OTHER	OTHER	Wrong Value	N	N	N	O

These are just a few of the most used variables that can be included in a dataset that is used to create a format. Many others exist and using the CNTLOUT feature of a PROC FORMAT as shown above, will help us to see the others. Placing the output from the format into a dataset will allow us to see other options that can be used in a dataset when trying to create a format.

CONCLUSION

Read the data, create the format, apply the format to output the results. This is a terrific way to create a format from your data and then use the format. It has many applications for its use. Some of these include: enhancing the format, changing the format as the data itself changes, adding extra conditions, adding exclusions. One asset of using this method is cutting down on the amount of typing required to create a format. The less typing that you need to do, the fewer errors that are possible. Rather than

having to keep lots of code around that will create a format and then finding the errors or places where changes need to be made, the data itself is used to derive the format. Whether it is a pharmaceutical, financial, educational or government application, formats can be created using the data. What could be better than that!

REFERENCES

SAS® Procedures Guide, Version 8, SAS Publishing, 1999

ACKNOWLEDGMENTS

I would like to thank my husband Bob for his unending support and tireless proofreading comments. Without his care, fortitude and encouragement, I could not have successfully completed my paper.

AUTHOR CONTACT

Janet Stuelpner
Left Hand Computing, Inc.
326 Old Norwalk Road
New Canaan, CT 06840
(203) 801-4373 voice
(203) 801-0043 fax
jstuelpner@usa.net

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.