

Paper 068-2007

Creating a Format from Raw Data or a SAS® Dataset

Wendi L. Wright

ABSTRACT

This introductory level presentation demonstrates the creation of a format from raw data or a SAS dataset using the CNTLIN= option in PROC FORMAT. In order to create a format from a SAS dataset, the SAS dataset must have a few required variables. This paper will explain how to construct this dataset and how to create character and numeric formats or informats within a dataset. Also, the CNTLOUT= option will be used in order to demonstrate how SAS stores formats internally. This knowledge can be used to help construct even more sophisticated formats using the CNTLIN option.

THE PROC FORMAT CNTLIN= OPTION

The simplest way to create a format from a dataset is to use the CNTLIN= option in PROC FORMAT.

```
PROC FORMAT CNTLIN=fmtdataset;
RUN;
```

REQUIRED VARIABLES IN THE FORMAT DATASET (FMTNAME, START, AND LABEL)

There are three variables that are required in the formatting dataset. They are:

Variable	Used for
FMTNAME	The format name
START	The left side of the formatting = sign (assumed character) – must be unique unless defining a Multi-Label format
LABEL	The right side of the formatting = sign

These three variables in a dataset will define a format that can also be defined using more traditional code of:

```
PROC FORMAT;
  VALUE $fmtname
    'Start' = 'label'
    ...etc....
  ;
```

Below is an example of some simple code that reads in a data file with two columns, state code and full state name. We will use this data file to create a format called \$state in SAS.

State Code	State Name
PA	Pennsylvania
NJ	New Jersey
etc.	etc.

```
DATA fmt;
    Infile in1;
    Retain fmtname '$state';
    Length start $2 label $50;
    Input start $ label $;
RUN;
PROC FORMAT CNTLIN=fmt;
RUN;
```

ASSIGNING THE SAME LABEL TO MULTIPLE START VALUES

If you have multiple 'START' values that are all assigned to the same 'LABEL', you can treat this as a regular format by listing each start value as its own observation with the same label. SAS allows label values to be repeated across observations. Here is an example:

State Code	Region
PA	NorthEast
NJ	NorthEast
NY	NorthEast
CA	West
WA	West
etc.	etc.

```
DATA fmt;
    Infile in1;
    Retain fmtname '$reg';
    Length start $2 label $50;
    Input start $ label $;
RUN;
PROC FORMAT CNTLIN=fmt;
RUN;
```

CHANGING THE TYPE OF FORMAT (THE 'TYPE' VARIABLE)

There is a fourth variable, called 'TYPE', that can be used in your formatting dataset to modify the type of format you want to create. Type can have the following values with the following meanings:

Value	Means to Create
C	Character Format
N	Numeric Format
J	Character Informat
I	Numeric Informat
P	Picture Format

In the \$reg. example above, we created a character format and SAS automatically assigned the Type Variable = 'C'. SAS assigned a 'C' because our format name started with a '\$'. If the name had not started with a '\$', the default type assigned would have been 'N'.

Let's have a short discussion about the difference between formats and informats vs. character and numeric (in)formats. If you are creating a format (vs. an informat) then the right side of the equal sign (or the 'LABEL' variable) will be character. If you are creating an informat, then the 'LABEL' variable will be numeric. If you are creating a character format or informat, then the format name must have a '\$' prefix and the 'START' variable will be character. If you are creating a numeric format or informat, then the format name will not have the '\$' prefix and the 'START' variable will be numeric.

You can explicitly assign the type in your program. The most efficient way to do this is to use the retain statement. For example, let's assume the state code is numeric instead of character. We will create a format name of 'STATE' without the '\$' prefix. We can let SAS assign the type value of 'N' automatically, or we can assign it ourselves as shown below.

State Code	State Name
01	Pennsylvania
02	New Jersey
Etc.	etc.

```
DATA fmt;
    Infile in1;
    Retain fmtname 'state' type 'N';
    Length start $2 label $50;
    Input start $ label $;
RUN;
PROC FORMAT CNTLIN=fmt;
RUN;
```

USING A SAS DATASET RATHER THAN RAW DATA

Here is an interesting problem I was recently asked to help resolve. The dataset (named 'Original') included a variable called 'Code'. Code has a length of 30 and approximately 1000 unique values. The client wanted to assign a different groupcode (numeric number) for each value. The data had been pre-read and was already in a SAS dataset. Note: this is only one of a number of ways this could have been solved. There are three steps needed:

1. Create a second SAS dataset that holds one observation for each unique value of code.


```
PROC FREQ data=original;
    TABLES code / noprint out = uniqueValues;
RUN;
```
2. Use this new SAS dataset to set up the required variables needed for a format and create the format.


```
DATA fmtDataset;
    SET uniqueValues;
    RETAIN fmtname '$fmtcode' type 'C';
    RENAME Code = Start;
    LABEL = put(_n_,4.);
RUN;
PROC FORMAT CNTLIN=fmtDataset;
RUN;
```

3. Apply the newly created format to the original dataset.

```
DATA original;
    SET original;
    GroupCode = put (Code, $fmtcode.);
RUN;
```

IF THE START VALUE DEPENDS ON MORE THAN ONE VARIABLE

If the start value depends on more than one variable and/or you need to include a mix of character and numeric variables in determining what label should be applied, you can use the SAS concatenation. The example below uses SAS's concatenation feature, the double bar '||'. The objective is to apply a Scaled Score for each student who took each test. In this case we want the result to be numeric so we are going to create an informat. Note how the concatenation contains not just character variables, but also includes a transformation of a numeric to a character variable.

The dataset has the following variables:

Test (character)	Grade (character)	RawScore (numeric)	ScaledScore (numeric)
Read	04	1	200
Read	04	1.5	201
Read	04	2.0	202
...
Math	12	74	798
Math	12	74.5	799
Math	12	75	800

```
DATA fmtct;
    SET ctsem;
    RETAIN fmtname '$applyss' type 'J';
    LENGTH fmtname $8 start $10;

    START=test || grade || put(RawScore, 4.1 );
    RENAME ScaledScore=LABEL;
RUN;
PROC FORMAT CNTLIN=FMCT;
RUN;
```

CREATING A RANGE ON THE LEFT SIDE OF THE FORMAT (THE 'END' VARIABLE)

There are times when it is not convenient to list each unique starting value as a separate observation. SAS allows you to create a format with both a start and ending range on the left side of the equal sign to specify a range of values that all get formatted to the same value. To do this using a SAS the CNTLIN option, you need to add another variable to the dataset, a variable called 'END'. Note that the start and end are numeric, so the format will be a numeric format (no '\$' in front of the format name). The following is a simplified example that shows how this works.

Low	High	Rating
0	10	A
11	20	B
21	30	C
etc.	etc.	etc.

```
DATA fmt;
    Infile in1;
    Retain fmtname 'Rate';
    Length label $50;
    Input start end label $;
RUN;
PROC FORMAT CNTLIN=fmt;
RUN;
```

EXCLUDING EITHER THE LEFT OR RIGHT SIDE OF THE RANGE (EEXCL AND SEXCL)

What if your ranges have overlapping starting and ending values? For example if one range is 0-10 and the next is 10-20. For those of you who have used format in the past, you know that SAS will not allow two ranges to include the same number (unless you are using the MLF option). You must specify which format the 10 belongs to (or is excluded from). In a regular SAS format you specify it like this:

```
PROC FORMAT;
    VALUE fmtname
        0 - < 10 = 'A'
        10 - < 20 = 'B'
        Etc.;
RUN;
```

However, when you are constructing a CNTLIN dataset, you will need to use either the EEXCL or the SEXCL variables to tell SAS which side of the range the number is excluded from.

Variable	Meaning
EEXCL	If 'Y' then the range's ending value is excluded from this range If 'N' then the range's ending value is not excluded
SEXCL	If 'Y' then the range's starting value is excluded from the range If 'N' then the range's starting value is not excluded

To create a dataset that will do the same as the format code above, the following code can be used. Note the sample input dataset has changed slightly from the prior example so that the end and the beginning of the ranges are the same.

Low	High	Rating
0	10	A
10	20	B
20	30	C
etc.	etc.	etc.

```
DATA fmt;
    Infile in1;
    Retain fmtname 'Rate' EEXCL 'Y';
    Length label $50;
    Input start end label $;
RUN;
PROC FORMAT CNTLIN=fmt;
RUN;
```

USING THE 'LOW' AND 'HIGH' VALUES IN YOUR FORMAT DATASET (HLO=)

What if you want to specify a starting value of 'LOW' or an ending value of 'HIGH' in your format? In this case, you cannot just put 'LOW' as the value of the 'START' variable or 'HIGH' as the value of the 'END' variable. Instead, you need to use a different variable called 'HLO'. Below are three of the values that HLO can have. There are several others.

HLO Value	Meaning
H	Range's ending value is HIGH (Value in END will be ignored)
L	Range's starting value is LOW (Value in START will be ignored)
O	Range is Other (both Start and End values are ignored)

HLO can be set to 'O' if you want to include a format range of 'Other' in your CNTLIN dataset. For example, if the client requested all values other than what is in the given table are to be assigned a '9'. This example assumes that the lowest rating is the first observation and the highest is the last observation (pre-sorted dataset).

Note the use of the infile option of 'END=LAST' to mark which observation is the last in the input file.

Low	High	Rating
Lowest	10	A
10	20	B
20	30	C
etc.	etc.	etc.
999	High	Z

```

DATA fmt;
  Infile in1 end=last;
  Retain fmtname 'Rate' EEXCL 'Y';
  Length label $50;
  Input start end label $;

  *** For the first observation, assign the 'LOW' keyword;
  If _n_ eq 1 then HLO='L';

  If last then do;
    *** For the last observation, assign the 'HIGH' keyword and output;
    HLO='H';
    Output;
    *** After outputting the last observation, output an additional
        observation so that all other values are assigned '9';
    HLO='O';
    Label='9';
    Output;
  End;

  Else output;
RUN;
PROC FORMAT CNTLIN=fmt;
RUN;

```

HOW SAS STORES FORMATS INTERNALLY

To see what SAS has stored in the format library, you can use the following code:

```
PROC FORMAT CNTLOUT=fmtout;
RUN;
PROC PRINT DATA=fmtout;
RUN;
```

Output from this will look like this:

```

          F M T N A M E           S T A R T           E N D           L A B E L           D E L I M I T E R           P R E F I X           N O T E           S E E K I N G           D I T A G A T I O N
          RATE LOW           10           20           990 HIGH           **OTHER**           10 A 1 40 1 1           1E-12           0           0 N N Y L
          RATE           20           30 C 1 40 1 1           1E-12           0           0 N N Y
          RATE           990 HIGH           Z 1 40 1 1           1E-12           0           0 N N Y H
          RATE **OTHER**           **OTHER**           9 1 40 1 1           1E-12           0           0 N N N O

```

Note how the Low, High and Other values are represented depending on the value of the HLO variable. Also note the value of the EEXCL variable that was set in the retain statement.

CONCLUSION

This has only been an introduction of the many and varied ways to use a formatting dataset with the CNTLIN= option to create formats. More information about the use of the CNTLIN= option can be found in the SAS online documentation for PROC FORMAT. In the documentation are descriptions of the Input and Output Control Datasets. Note that the output control dataset description is more complete and the author has found the description of the output control dataset to be of much use when deciding how to construct the input control dataset.

CONTACT INFORMATION

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

Wendi L. Wright
 1351 Fishing Creek Valley Rd
 Harrisburg, PA 17109
 Phone: (717) 513-0027
 E-mail: sleepypuppyfeet@earthlink.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.