Paper 146-2022

Using SAS Formats

Tom Kari, Tom Kari Consulting Ltd.

ABSTRACT

Formats are a secret weapon in SAS, for the initiated. But there are a lot of moving parts...What does a format look like? Where are my formats stored? How can I replace a format? How can I decide which format will be used? Answers to these questions, and a few well-travelled tips for solving common programming problems with formats, will provide some additions to your toolkit that will let you get even more use out of this versatile tool.

INTRODUCTION

SAS is an amazing tool, and formats are a key part of what makes it so powerful. But, with power comes complexity, and this is certainly the case with formats. This paper will delve into how formats are managed and referenced in SAS, how you can figure out the structure of your format environment, and a few troubleshooting tips for solving problems.

I've also included a few of my favourite tips for using formats, to allow you to take full advantage of this powerful tool.

I'll be using the data in SASHELP for all of examples, so you can "steal" and use the code I'm presenting. The code should be ready to cut and paste.

This paper assumes that the reader is familiar with creating and using SAS formats, and would like to understand more about how they are managed and maintained.

WHERE DOES THIS APPLY?

As far as I know, the examples in this paper will apply to both desktop SAS installations, and those where a SAS client is being used with SAS on a server. The code and output were developed and tested using SAS OnDemand For Academics.

Everything in this paper should apply to SAS 9.4 environments. Most of this goes back a very long way, but depending on how old your SAS installation is, things may work differently.

I don’t have access to a Viya or CAS environment, so I have no information on whether any of these tips will work with them.

I use formats for my examples, but I believe they will apply to informats as well.

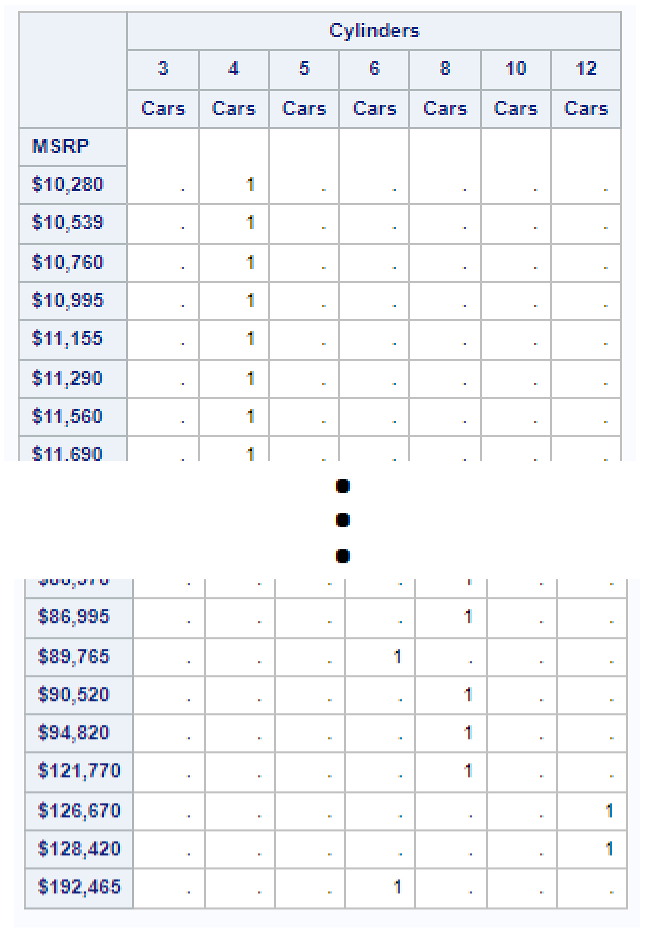
WHY ISN’T SAS USING MY FORMAT?

This scenario will look very familiar to anyone who has worked either in SAS administration or helping users to troubleshoot SAS problems.

A user would like to understand sashelp.cars better, so she runs a simple table against the data:

```
/* Tabulate price by cylinders */
proc tabulate data=sashelp.cars;
    class MSRP cylinders;
    keylabel n="Cars";
    table MSRP, cylinders;
run;
```

And of course, she doesn’t make any syntax mistakes, or forget any semicolons, so out comes 409 lines of statistical goodness...



	Cylinders						
	3	4	5	6	8	10	12
	Cars	Cars	Cars	Cars	Cars	Cars	Cars
MSRP							
$10,280	.	1	.	.	.	.	.
$10,539	.	1	.	.	.	.	.
$10,760	.	1	.	.	.	.	.
$10,995	.	1	.	.	.	.	.
$11,155	.	1	.	.	.	.	.
$11,290	.	1	.	.	.	.	.
$11,560	.	1	.	.	.	.	.
$11,690	.	1	.	.	.	.	.

●

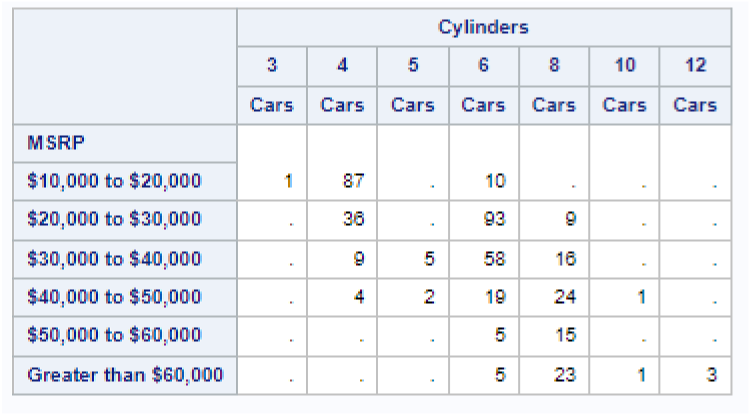
●

●

[illegible]	.	.	.	.	1	.	.
$86,995	.	.	.	.	1	.	.
$89,765	.	.	.	1	.	.	.
$90,520	.	.	.	.	1	.	.
$94,820	.	.	.	.	1	.	.
$121,770	.	.	.	.	1	.	.
$126,670	.	.	.	.	.	.	1
$128,420	.	.	.	.	.	.	1
$192,465	.	.	.	1	.	.	.

Figure 1. Unsummarized Price Report

Well. Sadly, this really doesn't advance the cause much. What she had in mind was something more like this...



	Cylinders						
	3	4	5	6	8	10	12
	Cars	Cars	Cars	Cars	Cars	Cars	Cars
MSRP							
$10,000 to $20,000	1	87	.	10	.	.	.
$20,000 to $30,000	.	36	.	93	9	.	.
$30,000 to $40,000	.	9	5	58	16	.	.
$40,000 to $50,000	.	4	2	19	24	1	.
$50,000 to $60,000	.	.	.	5	15	.	.
Greater than $60,000	.	.	.	5	23	1	3

Figure 2. Summarized Price Report

…and after some head scratching, she realizes that a format is just the thing. So she swings into action with proc format…

```
/* Set a format to group the MSRP values */
proc format library=CARlib;
   value MSRPfmt low -< 10000="Less than $10,000"
```

```
             10000 -< 20000="$10,000 to $20,000"
             20000 -< 30000="$20,000 to $30,000"
             30000 -< 40000="$30,000 to $40,000"
             40000 -< 50000="$40,000 to $50,000"
             50000 -< 60000="$50,000 to $60,000"
             60000 - high="Greater than $60,000";
run;
```

...and she reruns his code with a format statement:

```
/* Tabulate price by cylinders */
proc tabulate data=sashelp.cars;
    format MSRP MSRPfmt.;
    class MSRP cylinders;
    keylabel n="Cars";
    table MSRP, cylinders;
run;
```

But the outcome is the same as in Figure 1.

What went wrong?

At this point, your reaction might be to ask around your colleagues, or to post a question on the SAS Communities (communities.sas.com), and I'm certainly not going to try to discourage you from either of those options. But wouldn't it be more fun to diagnose it yourself? Then, YOU can be the person everyone comes to for help, and get sent to SAS conferences to share your wisdom!

INTO THE DEPTHS OF FORMATS

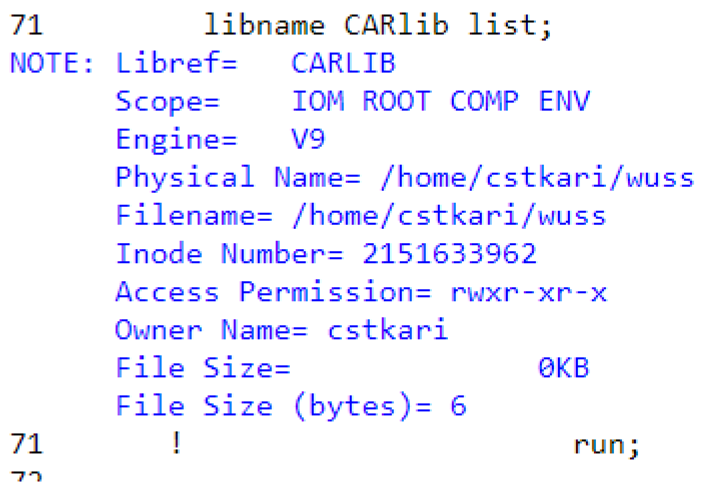
So, how can we take an evidence-based approach to this problem? We know two things: our format is named "MSRPfmt", and it's in SAS library CARlib. This leads to our first question; does CARlib exist? This question is answered with the libname statement. One option is to look at ALL of the SAS libraries that are have defined with:

```
libname _all_ list; run;
```

Another option is to look at the characteristics of a single library. For CARlib we would use:

```
libname CARlib list;
run;
```

which returns this in the log:



```
71          libname CARlib list;
NOTE: Libref=   CARLIB
      Scope=    IOM ROOT COMP ENV
      Engine=   V9
      Physical Name= /home/cstkari/wuss
      Filename= /home/cstkari/wuss
      Inode Number= 2151633962
      Access Permission= rwxr-xr-x
      Owner Name= cstkari
      File Size=              0KB
      File Size (bytes)= 6
71        !                     run;
```

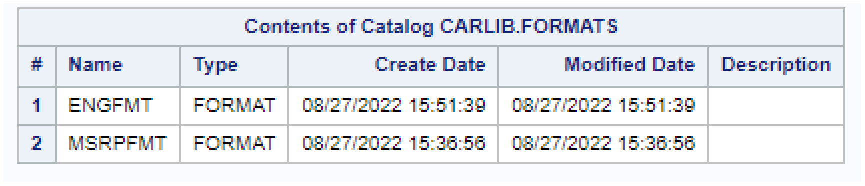
Figure 3: Using Libname to Check a Library

So we can see that CARlib is defined, and it is a SAS V9 library. That looks fine.

On to question number 2; could it be our format isn't in the library? A quick submission of proc catalog

```
proc catalog catalog=CARlib.formats;
    contents;
quit;
```

results in:



Contents of Catalog CARLIB.FORMATS					
#	Name	Type	Create Date	Modified Date	Description
1	ENGFMT	FORMAT	08/27/2022 15:51:39	08/27/2022 15:51:39	
2	MSRPFMT	FORMAT	08/27/2022 15:36:56	08/27/2022 15:36:56	

Figure 4: Using Proc Catalog to List Formats

Well, we've narrowed it down: (1) The SAS library exists, and has been assigned. (2) The format has been created, and is in the format catalog. What's left?

The last link in the chain is to examine where SAS looks for formats. When you reference a format, SAS will look in the following places:

SAS internal formats: These are the formats that SAS has created and provided with SAS, such as $upcase, datetime, and z. These are contained within the SAS system, aren't accessible, and you can't create a format with the same name (e.g. "ERROR: Format Z could not be written because it has the same name as a SAS-supplied format"). Therefore, don't worry about them. If you ever use one, and SAS can't find it, there's a major problem!

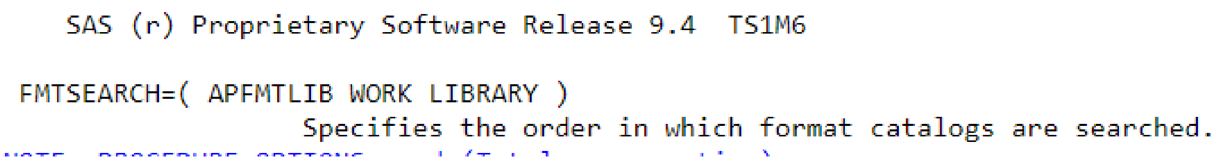
User defined formats: These are the formats that we create and use to make our data more usable. These are created using proc format, and stored in special SAS datasets called "catalogues", which exist within SAS libraries. Generally, when you want to reference a user-defined format, you need the library name and the format name[1].

The connection between creating a user defined format in a catalog and SAS finding and using it is the FMTSEARCH option. This option determines which libraries will be searched for formats, and in what order.

The current value of the option can be determined, as with any other option, using proc options. In the example that we've been exploring, submitting

```
proc options option=fmtsearch;
run;
```

Results in this log entry:



```
    SAS (r) Proprietary Software Release 9.4  TS1M6

FMTSEARCH=( APFMTLIB WORK LIBRARY )
                    Specifies the order in which format catalogs are searched.
```

[1] Of course it's never quite that straightforward in SAS. The default is that formats are stored in a catalog named "FORMATS" within the library. For advanced applications, the option exists to use a different catalog name, and to use LOCALE information to point at different catalogs. These topics are beyond the scope of this paper. When I search the Internet on "SAS 9.4 fmtsearch" the first hit describes the capabilities.

Figure 5. Format Library Concatenation

This means that

In this case, SAS will search for the MSRPfmt format (or any other user defined format for that matter) first in the library APFMTLIB, if it's not there it searches WORK, and finally if it's not in the first two it searches LIBRARY. In my case, neither APFMTLIB nor LIBRARY are assigned in my SAS session, but that doesn't cause any difficulties.

And now we have it! You'll recall that when we created the MSRPfmt, we wanted it retained so we created it in the CARlib SAS library (proc format library=CARlib;). So even though the library is there, and the format resides in it, by default SAS isn't looking for formats in that library.

Easy-peasy! There are a lot of possibilities with the FMTSEARCH option, but for now we'll use the easiest one. Submitting:

```
options fmtsearch=(CARlib);
run;
```

will add the CARlib library into the format search list, and the MSRPfmt format will be found.

One quick, important, note. If your format exists in any form in any of the libraries before your library in the concatenation, that is the format that will be used. Using these tools, you can ensure that it isn't in any other library. There are additional options with fmtsearch, which can be used to manage the order of the libraries.

From this section, you have the tools to:

- Determine whether a library exists, and what kind of library it is;
- Establish whether a particular format is in the library;
- Ensure that the library name is in the list of libraries that will be searched by SAS when a user defined format is referenced.

In the next sections, we'll discuss a number of topics relating to formats that I find useful.

USEFUL INFORMATION ABOUT FORMATS

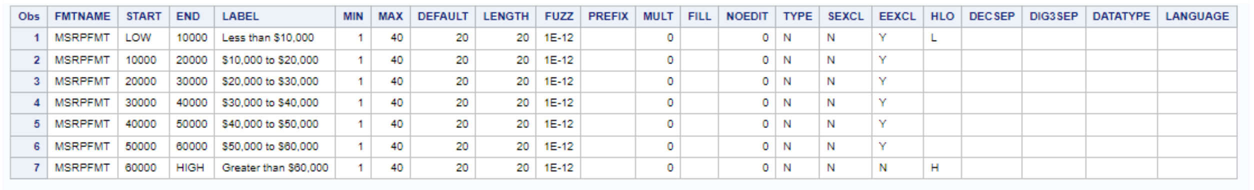
WHAT'S IN A FORMAT?

The last thing we might need to examine about a format is what's actually in the format. Fortunately, this is simple, and SAS also provides a tool to create formats from data in an automated manner.

To see the contents of a format, use proc format with the cntlout option, like this:

```
proc format library=CARlib cntlout=work.MSRPfmt_desc;
    select MSRPfmt;
run;
```

SAS will create an image of the format in the SAS dataset that you specify, like this:



Obs	FMTNAME	START	END	LABEL	MIN	MAX	DEFAULT	LENGTH	FUZZ	PREFIX	MULT	FILL	NOEDIT	TYPE	SEXCL	EEXCL	HLO	DECSEP	DIG3SEP	DATATYPE	LANGUAGE
1	MSRPFMT	LOW	10000	Less than $10,000	1	40	20	20	1E-12		0		0	N	N	Y	L				
2	MSRPFMT	10000	20000	$10,000 to $20,000	1	40	20	20	1E-12		0		0	N	N	Y					
3	MSRPFMT	20000	30000	$20,000 to $30,000	1	40	20	20	1E-12		0		0	N	N	Y					
4	MSRPFMT	30000	40000	$30,000 to $40,000	1	40	20	20	1E-12		0		0	N	N	Y					
5	MSRPFMT	40000	50000	$40,000 to $50,000	1	40	20	20	1E-12		0		0	N	N	Y					
6	MSRPFMT	50000	60000	$50,000 to $60,000	1	40	20	20	1E-12		0		0	N	N	Y					
7	MSRPFMT	60000	HIGH	Greater than $60,000	1	40	20	20	1E-12		0		0	N	N	N	H				

Figure 6. SAS Cntlout Dataset

To create a format, possibly from a spreadsheet or the results of a SQL query, the cntlin option is used. Suppose I have a SQL result of the codes and labels for Canadian provinces in a SAS dataset called work.prov that looks like this:

Province_Code	Province_Name
10	Newfoundland and Labrador
11	Prince Edward Island
12	Nova Scotia
13	New Brunswick
24	Quebec
35	Ontario
46	Manitoba
47	Saskatchewan
48	Alberta
59	British Columbia
60	Yukon
61	Northwest Territories
62	Nunavut

Table 1. Canadian Provinces and Territories

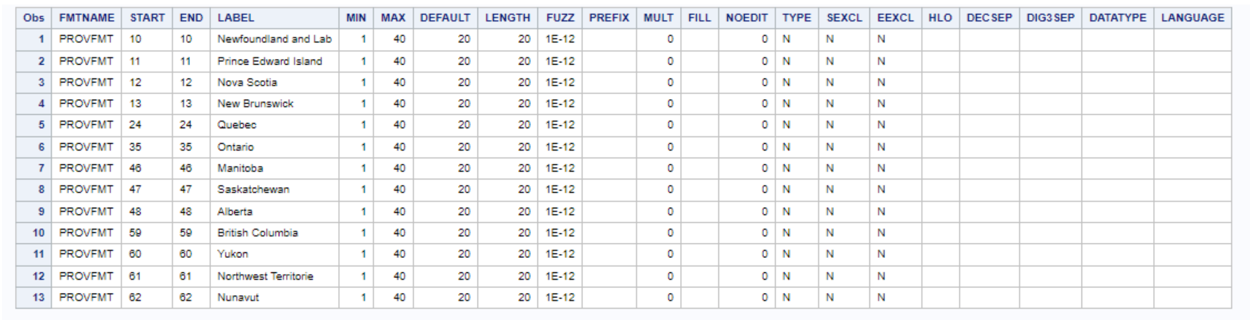
I could do all of the text editing to turn it into the correct appearance for proc format, but it's much easier than that. If you supply a SAS dataset with the columns you want, it will create the format from the dataset. The dataset, however, must have a very specific format. The following step will reformat the dataset as required:

```
data work.prov_fmt;
    set work.prov(rename=(Province_Code=start Province_Name=label));
    fmtname="PROVfmt";
    type="N";
run;
```

And then you run proc format with the cntlin option:

```
proc format library=CARlib cntlin=work.prov_fmt;
run;
```

Results in this format:



Obs	FMTNAME	START	END	LABEL	MIN	MAX	DEFAULT	LENGTH	FUZZ	PREFIX	MULT	FILL	NOEDIT	TYPE	SEXCL	EEXCL	HLO	DECSEP	DIG3SEP	DATATYPE	LANGUAGE
1	PROVFMT	10	10	Newfoundland and Lab	1	40	20	20	1E-12		0		0	N	N	N					
2	PROVFMT	11	11	Prince Edward Island	1	40	20	20	1E-12		0		0	N	N	N					
3	PROVFMT	12	12	Nova Scotia	1	40	20	20	1E-12		0		0	N	N	N					
4	PROVFMT	13	13	New Brunswick	1	40	20	20	1E-12		0		0	N	N	N					
5	PROVFMT	24	24	Quebec	1	40	20	20	1E-12		0		0	N	N	N					
6	PROVFMT	35	35	Ontario	1	40	20	20	1E-12		0		0	N	N	N					
7	PROVFMT	46	46	Manitoba	1	40	20	20	1E-12		0		0	N	N	N					
8	PROVFMT	47	47	Saskatchewan	1	40	20	20	1E-12		0		0	N	N	N					
9	PROVFMT	48	48	Alberta	1	40	20	20	1E-12		0		0	N	N	N					
10	PROVFMT	59	59	British Columbia	1	40	20	20	1E-12		0		0	N	N	N					
11	PROVFMT	60	60	Yukon	1	40	20	20	1E-12		0		0	N	N	N					
12	PROVFMT	61	61	Northwest Territorie	1	40	20	20	1E-12		0		0	N	N	N					
13	PROVFMT	62	62	Nunavut	1	40	20	20	1E-12		0		0	N	N	N					

Figure 7. SAS Format Created From Dataset

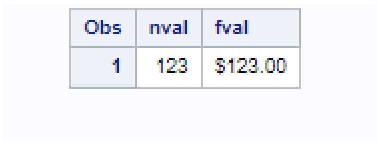
Note that you can create an enormous number of formats in one run. I used this when I was converting from an older system to SAS, and we processed tens of thousands of lines of format data.

THE PUT AND INPUT FUNCTIONS

The put function returns a character variable that represents the formatted value of the variable that is passed to it. For example, if I run

```
data xmp;
    nval=123;
    fval=put(nval, dollar7.2);
    output;
run;
```

My resulting dataset looks like this:



Obs	nval	fval
1	123	$123.00

Figure 8. Results of Using "put"

Input works the other way around, turning a formatted value to an unformatted one.

To really show off, use put and input in the same statement. This example converts a hexadecimal character value to the equivalent number, and then converts that number to a date character string. Running

   var1 = put(input("595A", hex4.), weekdate.);

my resulting value will be

   Wednesday, August 17, 2022.

UNATTACHING A VARIABLE FROM A FORMAT

If a variable in a dataset is associated with a format, there are many ways to remove the association. I like using proc datasets to do this, as there's no need to read the data. Only the dataset metadata is modified. The following code demonstrates removing a format:

```
data work.xmp1;
    var1=50;
    format var1 dollar5.;
    output;
run;

proc print data=work.xmp1;
run;

proc datasets lib=work nolist;
    modify xmp1;
    format var1;
quit;

proc print data=work.xmp1;
run;
```

AND FINALLY

Here are a few pointers that are too short to go into their own section.

- The process of finding a label for a format value is incredibly efficient in SAS. I have never found that this delays processing.

- SAS will let you create formats with more that 15 significant digits. But even though you appear to get results, the numeric variables will still have a maximum of 15 significant digits. Searching on “numerical accuracy in sas software” will provide more information.
- The fmterr option will toggle whether SAS throws an error if a variable is associated with a format that can’t be found at processing time.
- SAS Enterprise Guide (version 7.15 and later) contains a task that allows you to view catalogs and formats in your SAS session

CONCLUSION

I hope that you find these tips helpful. Without them, formats might appear to be too complicated to use. With the needed information, there is an amazing amount you can do with formats!

RECOMMENDED READING

Base SAS® Procedures Guide

SAS® For Dummies®

CONTACT INFORMATION

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

Tom Kari
1-613-899-6359
tom.kari.consulting@bell.net