PharmaSUG 2016 - Paper TT09

Formats and Informats – Concepts and Quick Reference Table

Emmy Pahmer, inVentiv Health, Montreal, Canada

ABSTRACT

Using formats and informats is very common in SAS® programming. They are used to read external data, to temporarily or permanently change how data are displayed, to categorize, or to look up related values as with a lookup table. This paper will look at how to create and use formats and informats in various contexts, and provide a quick-reference table with examples.

INTRODUCTION

SAS provides many useful formats and informats. The ones most used may depend on which field you work in. For example ISO date and time formats may be useful in clinical research, and financial formats may be more useful in banking or insurance. Users can define their own formats and informats as well. They can be created by coding directly in a program or by reading data from a control file.

Even experienced programmers may not be creating formats and informats in their everyday programming. Often they are created once and used on a long-term basis, or created from external files so programs don't need to be changed. So it's easy to forget how to code them.

Do you need a format or an informat? When do you need to use "value" or "invalue" in the PROC FORMAT statement? Do you need to define it as character (with "$") or not? Let's have a look.

See the quick reference table at the end of the paper for examples.

TO PERMANENTLY ASSOCIATE A FORMAT WITH A VARIABLE

A format may be permanently associated with a variable so that it is always displayed a certain way when viewing or outputting data.

```
data permfmt;
    date = '31dec1999'd;             *  Create numeric date value. ;
    *format date worddate.;          *  Associate format with either this ;
    attrib date format = worddate.;  *  or this. ;
    put '1. ' date= 8.;
    put '2. ' date=;
run;
```

Log:
```
1. date=14609    <- Numeric value of date, format 8. specified in the put statement
2. date=December 31, 1999   <- format already assigned, not specified in the put
                               statement.
```

Proc Contents:
```
#    Variable    Type    Len    Format
1    date        Num       8    WORDDATE.
```

Here's an example of using a custom format called AGECAT. This format applies an associated label: "kid" or "adult" to each value, thereby grouping them.

```
proc format;
    value agecat
    0-17 = 'kid'
    18-122 = 'adult';
run;
```

It is being permanently associated with the variable AGE and when a PROC FREQ is run on this variable, the formatted values are used.

```
data catwfmt;
    format Age agecat.;
    age = 1; output;
    age = 10; output;
    age = 20; output;
run;

proc freq data = catwfmt;
    table age / nopercent nocum ;
run;
```

```
 Age    Frequency
----------------------------
kid             2
adult           1
```

Output 1. PROC FREQ results with AGECAT format.

TO REMOVE A FORMAT

```
data catnofmt;
    set catwfmt;  ** from previous section **;
    format age;
run;

proc contents data = catnofmt; run;
```

```
#    Variable    Type    Len
1    Age         Num       8
```

Output 2. PROC CONTENTS with format removed.

THE DIFFERENCE BETWEEN A FORMAT AND AN INFORMAT

Formats are for controlling how data is displayed within SAS and how it is output. You are telling SAS “this is the way I want my data to appear”. Informats are used to identify how the input data appears, therefore how it should be read. You are telling SAS “this is what the incoming data looks like”.

CREATE BOTH INFORMATS AND FORMATS WITH PROC FORMAT

Both formats and informats are created with PROC FORMAT. PROC INFORMAT does not exist.

CREATING INFORMATS FOR READING CHARACTER VALUES

Use with the INPUT function.

Eg.

```
proc format;
    invalue gennum
    'F' = 1
    'M' = 2;
run;
```

- use INVALUE to create INFORMATS
- We know that the input is character. The character or numeric distinction is for the value we will be creating (value on right side of equal sign)
  - if the output value is to be character, create a character informat, use $
  - if the output value is to be numeric, create a numeric informat, no $

CREATING FORMATS FOR OUTPUTTING CHARACTER VALUES

Use with the PUT function.

Eg.

```
proc format;
    value rank
    1 = 'first'
    2 = 'second';
run;
```

- use VALUE to create FORMATS
- We know that the output is character. Character or numeric distinction is for the value we will apply the format to (value on left side of equal sign).
  - if the input value is character, then create a character format, use $
  - if the input value is numeric, create a numeric format, no $

PUT AND INPUT FUNCTIONS

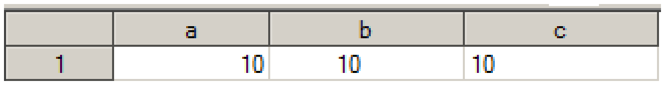
These functions use formats and informats.
Use INPUT to convert character values to numeric values or other character values
Use PUT to convert a numeric value to a character value.
Remember this: INPUT always reads characters and PUT always writes characters.

The INPUTN, INPUTC, PUTN and PUTC functions allow you to specify informats or formats at run time, so the second argument can be a variable that contains the format or informat you want to apply in that situation.

ALIGNMENT OPTIONS, –L

There are alignment modifiers that can be used with the PUT function. '-L' aligns the output character value to the left. This makes it unnecessary to use the LEFT/TRIM/STRIP functions.

```
data algn;
    a = 10;
    b = put(a, best.);
    c = put(a, best. -L);
run;
```



	a	b	c
1	10	10	10

Output 3. PROC PRINT with and without alignment option. Alignment problems may not be easy to see when viewing PROC PRINT output or PUT statements written to the log.

FORMAT AND INFORMAT STORAGE

Format and informat catalogs may be created in the work library or stored in a permanent library. If you do not specify a library then they are created in the work directory and are temporary.

PROC FORMAT SYNTAX

A PROC FORMAT block may contain many value clauses, with a semi-colon after each group of values.

INPUT CONTROL DATA SETS

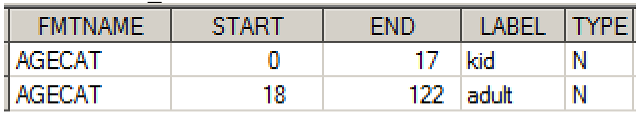
Rather than specifying them in the PROC FORMAT statement, the values may be supplied by an Input Control Data Set (CNTLIN).

Eg. Instead of :

```
proc format;
    value agecat
    0-17 = 'kid'
    18-122 = 'adult';
run;
```

Use a data set that already exists:

WORK.CNTL_AGECAT



FMTNAME	START	END	LABEL	TYPE
AGECAT	0	17	kid	N
AGECAT	18	122	adult	N

```
proc format cntlin = work.cntl_agecat;
run;
```

Control data sets must have particular variable names and formats. There may be several more variables; this only shows the bare minimum.

LOW, HIGH, <

The key word “LOW” can be used instead of the actual lowest value. And “HIGH” can be used instead of actual highest value. Use “<” to exclude the values in the range.

Eg.

```
proc format;
    value agecatc
    0-<18 = 'kid'          /* Up to but not including 18. */
    18- HIGH = 'adult';    /* Up to whatever the highest value might be. */
run;
```

OTHER IN THE PROC FORMAT VALUE STATEMENT

One may encounter unforeseen values. Using the OTHER key word will account for all values that were not previously specified.

```
proc format;
    value agecatd
    0-<18 = 'kid'  /* up to but not including 18 */
    18- HIGH = 'adult'
    other = '**check**';
run;
```

```
data ex9;
    age = -1;
    agecat = put(age, agecatd.);
    put age=  agecat=;
run;
```

Log:
age= -1   agecat= **check**

VIEWING THE FORMATS IN A CATALOG

```
proc format library = work fmtlib;
run;
```

This will output all the formats and informats in the catalog specified. You can select just certain formats with either the SELECT or EXCLUDE statements.

```
      FORMAT NAME: AGECAT   LENGTH:    5   NUMBER OF VALUES:    2
   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   5  FUZZ: STD

START            END              LABEL  (VER. V7|V8   28JUL2015:15:06:24)

               0               17 kid
              18              122 adult
```

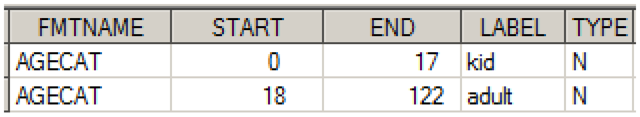
Output 4. Formats in the work library.

OUTPUT CONTROL DATA SETS

The information can also be output to an Output Control Data Set (CNTLOUT). Here we are selecting just the AGECAT format.

```
proc format library = work cntlout = work.cntl_agecat;
    select agecat;
run;
```

Here are some of the variables that are created:



FMTNAME	START	END	LABEL	TYPE
AGECAT	0	17	kid	N
AGECAT	18	122	adult	N

Output 5. Formats output to a control data set. Yes, this is the same as the input control data set above!

AUTOMATIC CONVERSIONS

If the variable you specify is not the right type, whenever possible, SAS will convert your values to suit the function being used.

```
data a;
    var1 = 1;
    var2 = input(var1, 8.);
run;
```

In this example, the input function normally requires a character variable. SAS is able to convert the value and puts this message in the log:

```
NOTE: Numeric values have been converted to character values at the places given by:
      (Line):(Column).
      4:18
```

```
   data b;
       var1 = 'b';
       var2 = input(var1, 8.);
   run;
```

If SAS cannot convert the value, a message like this will be generated.

```
NOTE: Invalid argument to function INPUT at line 17 column 12.
var1=b var2=. _ERROR_=1 _N_=1
NOTE: Mathematical operations could not be performed at the following places. The
results of the operations have been set to missing values.
      Each place is given by: (Number of times) at (Line):(Column).
      1 at 17:12
```

ERRORS

The FMTERR option controls whether or not you get an error if a format/informat is not found.
On: an error is generated.
Off (NOFMTERR): issues a note but no error, a default format is used.

OTHER

Several format-related topics are beyond the scope of this paper but a worth taking a look at as a next step.

- Searching different libraries
- Picture formats
- Nested formats
- Multi-Label formats

QUICK REFERENCE TABLE

The table below offers various examples of situations where formats and informats are created, and then used with INPUT/PUT statements or functions.

* Run statements were removed to save space.
** Semi-colons were removed at end of the Datalines statements to save space.
Examples 6 and 7 show two ways of doing the same thing.

Formats and Informats – Quick Reference Table

	Action	In put	Out put	Example	What to use	How to create	Example*	How to use	Example**	Example Result
1	Read character values, save as numeric	char	num	F -> 1 M -> 2	informat	proc format with invalue	`proc format; invalue gennum 'F' = 1 'M' = 2;`	with input statement	`input genn gennum.; datalines; F M`	`genn 1 2`
2	Read character dates, save as numeric	char	num	31/01/1960 -> 30	informat	N/A	`N/A SAS informat`	with input statement	`input daten ddmmyy10.; datalines; 31/01/1960;`	`daten 30`
3	Read numeric values, save as character	num	char	1 -> F 2 -> M	informat	proc format with invalue	`proc format; invalue $gen 1 = 'F' 2 = 'M';`	with input statement	`input genc $gen.; datalines; 1 2`	`genc F M`
4	Create numeric values from character values	char	num	F -> 1 M -> 2	informat	proc format with invalue	`proc format; invalue gennum 'F' = 1 'M' = 2;`	with input function	`genc = 'F'; genn = input(genc, gennum.); put genc= genn=;`	`genc=F genn=1`
5	Create numeric values from character values	char	num	"1.23" -> 1.23	informat	N/A	`N/A, SAS format`	with input function	`numc = '1.23'; numn = input(numc, best.); put numn=;`	`numn=1.23`
6	Create different character values	char	char	A -> first B -> second	informat	proc format with invalue, $	`proc format; invalue $newchar 'A' = 'first' 'B' = 'second';`	with input function	`c1 = 'A'; c2 = input(c1, $newchar.); put c1= c2=;`	`c1=A c2=first`
7	Create different character values	char	char	A -> first B -> second	format	proc format with value, $	`proc format; value $newcharb 'A' = 'first' 'B' = 'second';`	with put function	`c1 = 'A'; c2 = put(c1, $newcharb.); put c1= c2=;`	`c1=A c2=first`
8	Display char values as different char values, value not changed	char	char	A -> first B -> second	format	proc format with value, $	`proc format; value $newcharb 'A' = 'first' 'B' = 'second';`	specify display format	`c1 = 'A'; put c1=   $newchar.;`	`c1=first`
9	Create character values from numeric values	num	char	1 -> first 2 ->second	format	proc format with value	`proc format; value rank 1 = 'first' 2 = 'second';`	with put function	`n = 1; c = put(n, rank.); put n= c=;`	`n=1 c=first`
10	Display formatted value as character without changing original value	num	char	1 -> first 2 ->second	format	proc format with value	`proc format; value rank 1 = 'first' 2 = 'second';`	specify display format	`num = 1; put num=  rank.;`	`num=first`
11	Display formatted value as character without changing original value	num	char	1.23 -> $1.23	format	N/A	`N/A SAS format`	specify display format	`num = 1.23; put num= dollar5.2;`	`num=$1.23`
12	Display numeric date as character without changing original value	num	char	10 -> 11JAN1960	format	N/A	`N/A SAS format`	specify display format	`daten = '11JAN1960'd; put daten=  ; put daten=  date9.;`	`daten=10 daten=11JAN1960`
13	Create character values from numeric ranges	num	char	0-17 -> kid 18-122 -> adult	format	proc format with value	`proc format; value agecat 0-17 = 'kid' 18-122 = 'adult';`	with put function	`age = 10; age_group = put(age, agecat.); put age= / age_group=;`	`age=10 age_group=kid`
14	Display group value as character without changing original value	num	char	0-17 -> kid 18-122 -> adult	format	proc format with value	`proc format; value agecat 0-17 = 'kid' 18-122 = 'adult';`	specify display format	`age = 10; put age=   agecat.;`	`age=kid`

CONCLUSION

There are many situations where formats and informats are extremely useful. This overview of creating and using them with contextual examples should provide a good basis of understanding.

REFERENCES

Formats, Informats and How to Program with Them, Ian Whitlock, Independent SAS Consultant, Kennett Square, PA
http://www.lexjansen.com/nesug/nesug07/ff/ff19.pdf

PROC FORMAT in Action, Jack N Shoemaker, ThotWave Technologies, Cary, NC
http://www2.sas.com/proceedings/sugi27/p056-27.pdf

SAS(R) 9.2 Language Reference: Dictionary, Fourth Edition, PUT Function
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000199354.htm

SAS(R) 9.2 Language Reference: Dictionary, Fourth Edition, INPUT Function
https://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000180357.htm

The Power of PROC FORMAT; Updated for SAS ®9 Jonas V. Bilenas, JP Morgan Chase, Wilmington, DE
http://www.lexjansen.com/nesug/nesug05/pm/pm6.pdf

ACKNOWLEDGMENTS

Thanks to Bruce Gilsen for reviewing this paper.

CONTACT INFORMATION

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

Name: Emmy Pahmer
Enterprise: inVentiv Health
E-mail: emmy.pahmer@inventivhealth.com