SESUG Paper 178-2019

You Ought To Be In Pictures! Dressing up output with the PICTURE format

Jennifer Lindquist, Durham Veterans Affairs Health Care System

ABSTRACT

SAS®-provided and user-defined formats improve the appearance, interpretability, and readability of data. Using formats can eliminate the creation of additional 'parallel' text variables and inadvertent discrepancies between those 'parallel' character strings and the original numeric values. An under-utilized format option is the PICTURE format. A PICTURE format allows output to be stylized by instructing SAS to present numeric values in a specified pattern. The user can also include units and symbols (e.g. presenting a gain of 9.5 percent as +9.5%) and re-scaling by a multiplication factor is easy to do (e.g. displaying 15,000 as 15K). Also, rounding and truncation are a snap with a PICTURE format. All can be accomplished without creating additional variables or calculations! Familiar with Excel® formats? Almost all can be replicated with a SAS PICTURE format. Come explore how to dress up your output and get into PICTUREs!

INTRODUCTION

Formats are instructions to SAS on how to display the values of variables, allowing you to present the data differently without modifying the underlying values. Built-in, SAS-provided standard formats cover the basics, such as mmddyy10. for dates and dollar10.2 for currencies, but customizable user-defined formats can be created also. Formats come in two varieties - the more widely used VALUE format and the lesser utilized PICTURE format.

PICTURE FORMAT

The PICTURE format maintains the advantages of formats in general, such as improving the appearance of output, providing 'in-place' documentation, and reducing ambiguity. Formats offer an alternative to creating new/companion character variables to convey additional information:

```
x_num=78.2;
x_text= x_num|| 'degrees Fahrenheit';
        *which evaluates to '78.2 degrees Fahrenheit';
```

The shortcomings of duplicate text variables, like the pair above, includes additional time, programming effort, data step processing to create and maintain 'parallel' variables, and a potentially disastrous discrepancy arising if one variable is changed/updated but not the other. Formats eliminate the added expense and the possible disconnect of parallel text variables.

The more familiar user-defined VALUE format usually represents a variable's value as new character string.

```
PROC FORMAT;
      VALUE ToastyFormat
      LOW-<90= 'Below 90 degrees'
      90-120= '90 degrees or above';
RUN;
```

Original Values	ToastyFormat Values
23	Below 90 degrees
99.792	90 degrees or above
51.289	Below 90 degrees
115.2	90 degrees or above
-15.31	Below 90 degrees

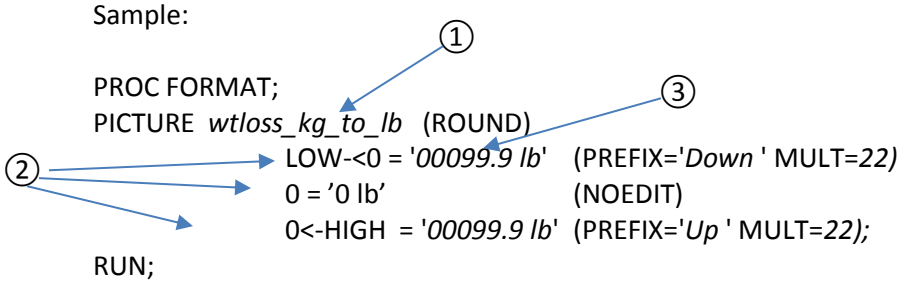
It is impossible to tell in the formatted output how much the actual value differs from the cut point of 90 degrees because the actual value is masked by the VALUE format. The PICTURE format is a set of instructions on how to display the value, preserving the value of the variable in the output. However, this highlights one of the restrictions of the PICTURE format – it can be used *only* with numeric variables. The PICTURE format allows you to add symbols, qualifiers, and comments to data output without modifying the numeric value. Once a PICTURE format is defined it can be used wherever formats are allowed, including in the PRINT, REPORT, TABULATE, SGPLOT, and FREQ procedures, to name a few.

SYNTAX

An abbreviated version of the syntax of the PICTURE format is shown here:

```
PROC FORMAT;
PICTURE  fmtname   <(ROUND)>
         Rangex = 'picturexpattern'
                                  (<PREFIX='prechar'>
                                   <MULT=scale>
                                   <NOEDIT>
                                   <FILL='fillchar'>);
```

Sample:




```
PROC FORMAT;
PICTURE  wtloss_kg_to_lb   (ROUND)
         LOW-<0 = '00099.9 lb'   (PREFIX='Down ' MULT=22)
         0 = '0 lb'              (NOEDIT)
         0<-HIGH  = '00099.9 lb' (PREFIX='Up ' MULT=22);
RUN;
```

① *Fmtname* gives the name of the PICTURE format and is the only required element. The name must be a valid SAS name with a maximum of 32 characters. It cannot have the same name as a SAS-provided format and, just like with user-defined VALUE formats, it cannot end in a number. When referring to a format, the name is followed by a period. However, the period is not used when defining the format.

② *Rangex* specifies individual values or intervals to which a picture pattern is applied. If a value is not covered in any of the ranges, then the value will appear either unformatted or as a series of asterisks depending on if the format width can accommodate the entire unformatted value. To help avoid missing values, three keywords are also available – LOW, HIGH, and OTHER. These special words

correspond to the lowest non-missing value, the highest value, and values not in any of the ranges. By convention, the OTHER category is usually the last range listed in a format definition.

③ *'Picturexpattern'* is limited to a maximum of 40 characters by default. It is a sequence of digit indicators called "selectors," text or directives in quotes which creates a template for displaying numbers. The digit indicators are numeric placeholders used to define a pattern. Although any digit (0 through 9) can be used, traditionally only 0 and 9 are used. If leading blanks are desired in the output, a 0 is utilized in the leftmost positions of the pattern; placing a 9 (or any digit other than 0) in these positions produces leading zeros in the output. A maximum of 16-digit indicators are allowed. Directives are special characters used in conjunction with user-defined date/time/datetime formats; these are beyond the scope of this paper but are well described in Carry Croghan's and Andrew Karp's papers listed in the references.

Picture format options are specified in parentheses.

ROUND rounds the data to the nearest integer before formatting. Note the ROUND option is placed before the range since it applies to the entire format. The other options are specific to the designated value or range of values.
CAUTION! – The picture pattern must be wide enough to accommodate an additional digit if needed when rounding up. The width of a picture is the width of the widest picture value and can be adjusted using the DEFAULT option or by adding additional digit selectors.

PREFIX instructs SAS to add characters designated by '*prechar*' to the beginning of the formatted value.
CAUTION! – If the picture pattern is not wide enough to accommodate both the value and the prefix, the format truncates or omits the prefix.

MULT (or *MULTIPLIER*) designates a number to multiply the value of the variable by before it is formatted.

NOEDIT allows the *picturexpattern* to be a text string containing numbers that will not be interpreted as digit selectors comparable to the user-defined VALUE format.

FILL tells SAS to left-pad the output with the character specified as '*fillchar*' when the length of the value is shorter than the maximum width of the pattern designated. The default fill character is a blank.

The *PREFIX*, *MULT, NOEDIT,* and *FILL* options may be defined differently on different ranges.

See SAS Help for additional options which are not covered in this paper: options to create user-defined date/time/datetime formats, specify a 3-digit separator other than the default comma, specify an alternative character for the decimal point, provide a fuzz factor for ranges, define multiple labels, and specify maximum, minimum and default lengths.

EXAMPLES

Now let's apply the syntax to some examples.

First create the data used in the examples:

```
DATA PicEx;
      INPUT  temperature 1-6 st $ 8-9  pop inches z  checks;
         DATALINES;
            42.31  FL 18801310 1.0 1234 1.23
            7.124  GA 9687653 5.2 789 57.89
            23     MD 5773552 8.6 23 950.01
            99.792 NC 9535483 9.2 85296 83.74
            51.289 SC 4625364 10.0 222222222 .
            115.2  VA 8001024 12    . .
            -15.31 WV 1852994 . . .
            163.99     . . . . .
            0          . . . . .
            ;
            RUN;
```

EXAMPLE 1 - DIGIT SELECTORS AND PREFIX:

Example 1 shows two formats to illustrate the use of the digit selectors, leading zeros, and the importance of the width of the pattern.

```
PROC FORMAT;
      PICTURE a_fmt     LOW - HIGH =  '99.9 degrees';
      PICTURE b_fmt     LOW - HIGH = '009.9 degrees';
      RUN;
```

The range LOW-HIGH instructs SAS to use the format for all non-missing values.

The pattern for a_fmt includes the digit selectors 99.9. This instructs SAS to display the output with two digits to the left of the decimal and one digit to the right of the decimal and if any of the leading digits are missing, then show a 0 in that spot.

The pattern for b_fmt changes the digit selectors to 009.9. This allows for three digits to the left and one to the right of the decimal point. Since zeros are used in the hundreds and tens digit places, if those digits are missing, the display leaves those positions blank.

Remember to associate the format with the variable in the DATA step or output procedure. PICTURE formats can be associated with a variable any place built-in SAS or user-defined VALUE formats are used.

```
PROC PRINT DATA=PicEx noobs;
      VAR temperature;
      FORMAT temperature  a_fmt.;
      RUN;
```

CAUTION! – Both of the formats a_fmt and b_fmt have shortcomings which are addressed in the follow up examples. The red circled items dropped the leading digit and the blue circled items lost the negative sign.

Original Values	a_fmt	b_fmt
42.31	42.3 degrees	42.3 degrees
7.124	07.1 degrees	7.1 degrees
23	23.0 degrees	23.0 degrees
99.792	99.7 degrees	99.7 degrees
51.289	51.2 degrees	51.2 degrees
115.2	15.2 degrees	115.2 degrees
-15.31	15.3 degrees	15.3 degrees
163.99	63.9 degrees	163.9 degrees
0	00.0 degrees	0.0 degrees

Table 1: Output showing differences when patterns have a leading 9 vs. a leading 0, the result when the original number is longer than the digits allowed in the pattern, and the lack of a negative sign.

The text ' degrees' appears in the formatted output. You can add any text inside the quotes between or after the digit selectors. Use this feature to add units and symbols to self-document your data.

Since the *ROUND* option was not in the format definition, the original values are truncated to the number of decimal places indicated in the pattern – in this example to one decimal place.

Notice how the two formats differ in the output display of 7.124 highlighted in yellow. With the format a_fmt, 7.124 is displayed as “07.1 degrees” because in the picture pattern a 9 appears in the tens digit spot. This inserts a leading 0 whenever the original value is between -10 and 10. The 0 in the tens digit spot in the b_fmt picture pattern allows for a leading blank so 7.124 is displayed as “7.1 degrees”.

Both patterns indicate one decimal place be displayed. When the original value is a whole number the format adds in the .0 to complete the pattern.

Note the decimal point is between two 9s in the patterns. SAS does not recognize the decimal point if it is between two zeros in the pattern. If the decimal is placed between two 0s, the displayed value will have NO decimal point.

CAUTION! In the output from Example 1, format a_fmt displays 115.2 and 163.99 incorrectly as 15.2 and 63.9 (red circled items above). This is a common mistake. The pattern is 99.9 which allows only two digits to the left of the decimal point. When the pattern is too short, the formatted values are incorrect – only the rightmost digits are shown. To avoid this error, get acquainted with your data prior to

applying formats using PROC FREQ and PROC MEANS with *min* and *max* options. Also, you may want to include an extra leading 0 in the pattern.

In format b_fmt, the pattern 009.9 allows for three digits to the left of the decimal point so it displays 115.2 and 163.9 correctly.

PROBLEM! Neither format correctly displayed -15.31 (blue circled values above). Both print 15.3 without the negative sign. So, there is room for improvement.

What happens when you try to type the negative sign as the first character in the pattern? No error message shows in the log, but the number is displayed without the negative sign. So how do you properly display negative values? Let's make a modification to the PICTURE format.

EXAMPLE 1A

To show the negative sign (or any leading text), use the *PREFIX* option. The general form PREFIX='*prechar*' allows you to place text or symbols to the left of the formatted value.

The format definition will need to be expanded to two ranges – one for the negative numbers, the other for the non-negative numbers. Note the pattern has four places before the decimal point for negative numbers and only three places preceding the decimal point for the positive numbers. The pattern needs to be wide enough to accommodate the negative sign.

Let's round the values instead of truncating.

```
PROC FORMAT;
   PICTURE c_fmt (ROUND)
      LOW-<0   =  '0009.9 degrees' ( PREFIX='-')
      0-HIGH =  '009.9 degrees';
      RUN;
```

Original Values	c_fmt
42.31	42.3 degrees
7.124	7.1 degrees
23	23.0 degrees
99.792	99.8 degrees
51.289	51.3 degrees
115.2	115.2 degrees
-15.31	-15.3 degrees
163.99	164.0 degrees
0	0.0 degrees

Table 2: Output showing the use of the *PREFIX* and *ROUND* options.

The incorrectly displayed values from Example 1 are fixed. Notice the 99.792, 51.289 and 163.99 are rounded up instead of being truncated. Not bad, gets the job done. Now let's add some extras.

EXAMPLE 1B

It is simple to include both + and – signs, just expand to three ranges, one for the negative numbers, one for the positive numbers, and one for zero. This time, instead of the text "degrees", let's use the degree symbol (°). The degree symbol is not on the standard American keyboard, but it is assigned to code 176 in the Expanded ASCII characters that SAS supports. First, we assign the macro variable `&deg_sym` with the ASCII code that corresponds to the º character. Then use the macro variable outside of the quotes in the PICTURE format.

The hottest and coldest weather temperatures ever recorded on earth are +135ºF and -130ºF, respectively. So, the range is modified with a warning if the data is outside the range or is a missing value. Many organizations use 999 to indicate a missing value, but if you put 999 inside of quotes in a PICTURE format, SAS would interpret the 999 as digit selectors. SAS addresses this issue with the *NOEDIT* option. *NOEDIT* allows you to incorporate text strings that include numbers similar to a VALUE format.

```
%let deg_sym=%sysfunc(byte(176));

PROC FORMAT;
   PICTURE d_fmt (ROUND)
      -130-<0   =  '0009.9' &deg_sym 'F'   (PREFIX='-')
      0='009.9' &deg_sym 'F'
      0<-135 ='009.9' &deg_sym 'F'  (PREFIX='+')
      OTHER = '999 or Out of Range - Check data'  (NOEDIT);
      RUN;
```

Original Values	d_fmt
42.31	+42.3 º F
7.124	+7.1 º F
23	+23.0 º F
99.792	+99.8 º F
51.289	+51.3 º F
115.2	+115.2 º F
-15.31	-15.3 º F
163.99	999 or Out of Range - Check data
0	0 º F

Table 3: Output showing the use of a macro variable in a PICTURE format, and the *NOEDIT* option

EXAMPLE 2 – MULTIPLIER:

EXAMPLE 2A

For the next example, consider US state census population numbers. The *MULT* option is useful when the data consists of very large or very small numbers. Displaying too many digits creates cluttered output, be it in a report, table, or graph. Eyes tend to glaze over when presented with an entire page of digits.

```
PROC FORMAT;
   PICTURE pop2010_
      LOW-HIGH='09.9 Million' (MULT=0.00001);
      RUN;
```

	Original data	pop2010_
Florida	18,801,310	18.8 Million
Georgia	9,687,653	9.6 Million
Maryland	5,773,552	5.7 Million
North Carolina	9,535,483	9.5 Million
South Carolina	4,625,364	4.6 Million
Virginia	8,001,024	8.0 Million
West Virginia	1,852,994	1.8 Million

Table 4: Output showing *MULT* option results.

The *MULT* option multiplies the value of the variable by the designated value and then applies the picture pattern. Always remember to check your results!

If you calculate by hand 18,801,310*0.00001 = 188.0131 – that does not match the result of 18.8. What gives?

SAS has a set algorithm when applying the PICTURE format. When the *MULT* option is present, it adjusts for the number of decimal places to the right of the decimal point in the pattern. In the population example there is one place to the right of the decimal in the pattern. The multiplication factor is then 0.00001*10^-1 so, for example, 18,801,310*0.00001*10^-1=18.80131, which is rounded to “18.8” and matches the first row of output in Table 4.

EXAMPLE 2B

Let's look at an example using the MULT option to convert inches to centimeters. The standard conversion is 1 inch=2.54 cm. In the example below the pattern has two decimal places to the right of the decimal point. Since the pattern introduces a factor of 10^-2, the *MULT* scaler is listed as 254, resulting in 254*10^-2=2.54.

```
PROC FORMAT;
   PICTURE inchToCM_fmt (ROUND)
       LOW-HIGH = '0009.99 cm' (MULT=254.0);
   RUN;
```

Original data	inchToCM_fmt.
1.00	2.54 cm
5.2	13.21 cm
8.6	21.84 cm
9.2	23.37 cm
10.0	25.40 cm
12	30.48 cm

Table 5: Output showing *MULT* option converting inches to centimeters

EXAMPLE 3 - FILL

EXAMPLE 3A

This example demonstrates the *FILL* option. On occasion, you may need to standardize the width of a column of numbers to tidy up/'square up' the display. One way to accomplish this would be:

```
PROC FORMAT;
   PICTURE hash_fmt LOW-HIGH ='000009999'  (FILL='#');
   RUN;
```

Original Values (left justified)	Original Values (right justified)	Original Values (center justified)	hash_fmt
1234	1234	1234	#####1234
789	789	789	#####0789
23	23	23	#####0023
85296	85296	85296	####85296
222222222	222222222	222222222	222222222

Table 6: Output showing *FILL* option

EXAMPLE 3B

Do you use on-line banking? Banks issue automated checks but of course the amounts will be different, the mortgage payment is much larger than the electric bill. When you examine those checks they appear with leading asterisks. A PICTURE format can reproduce this type of output.

```
PROC FORMAT;
      PICTURE check_fmt
      LOW-HIGH ='000,009.99' (PREFIX='$' FILL='*');
   RUN;
```

Original data	Check_fmt.
1.23	*****$1.23
57.89	****$57.89
950.01	***$950.01
83.74	****$83.74

Table 7: Output showing *FILL* option for automated checks

ADDITIONAL EXAMPLES

The above examples explain the basic mechanics of building a PICTURE format.

Next, I would like to share some ideas where you may want to use a PICTURE format.

Scientific journals have specific style requirements, especially for data presented in tables. If sample size for each characteristic is required to be of the form (n=xx) then using the following format will do the trick.

```
PICTURE sampnum LOW-HIGH =  '09)'    ( PREFIX='(n=' ) ;
```

You can create a PICTURE format which will 'automatically' insert asterisks to indicate a significant p value.

```
PICTURE  pval
           LOW-<0.01 = '<0.01 *' (NOEDIT)
           0.01-<.05 = '<0.05 *' (NOEDIT)
           0.05-HIGH = '9.0009';
```

Bank statements and credit card bills often print only the last 4 digits of an account number. Instead of creating a second variable consisting of just the last 4 digits, only indicate 4 digits in the pattern. Use the *PREFIX* option to mask the other digits .

```
PICTURE lastfour LOW-HIGH = '9999' (PREFIX='xxx-xx-');
```

Anytime you want to emphasize units, use a PICTURE format to display the units. Say the measurement is in kilograms instead of pounds: this format will display the weight in kilograms.

```
PICTURE kgfmt LOW-HIGH  = '00009 kg';
```

The following additional examples of the PICTURE format mimic some of the available EXCEL® formats (or use SAS provided formats if available!).

Suppose you have a variable in the dataset is already a percent, x=25 (not 0.25)and you want to add a percent sign. Using the SAS provided format percent7.0 would display 2500%. The format 'converts' the number to a percent by multiplying by 100. Using the following PICTURE format allows you to just add the percent sign.

```
PICTURE addpct    LOW-<0 ='00009 %'    (PREFIX='-')
                  0-HIGH ='0009 %';
```

In financial settings negative numbers are often presented in parentheses instead of with the negative sign.

```
PICTURE negparen LOW-<0 ='00009.99)'    (  PREFIX='('   );
```

Adding ASCII characters expands possibilities. The currency symbols for the Euro (128, €), British sterling pound (163, £), and Chinese yuan (165, ¥) are all available. The plus/minus sign (177, ±), trademark (153, ™), copyright (169, ©) and US registered trademark (174, ®) symbols are options. See the Appendix for a complete listing of the expanded ASCII codes.

Not all situations need a PICTURE format and SAS has a large library of built-in formats at your disposal. However, employing user-defined PICTURE formats allow you to create the custom format you need.

CONCLUSION

Use the PICTURE format to make your output Academy Award worthy! The PICTURE format allows you to create customized templates to dress up your output. With even just the basics, you can improve in-place documentation by adding units, symbols, and text while preserving the numeric value of a variable in the formatted output. Using the PICTURE format can make numbers easier to read and understand. To improve your reports, journal articles, and output for consumption, you ought to be in pictures – PICTURE formats that is!

APPENDIX

ASCII Table

```
DATA GenerateASCIIChar;
DO Code=0 TO 255;
   ASCIIChar=byte(Code);
OUTPUT;
END;
RUN;
```

```
PROC PRINT DATA=GenerateASCIIChar noobs;
RUN;
```

Code	ASCIIChar
0-32	Control codes to control peripheral devices such as printers
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	•
128	€
129	•
130	‚
131	ƒ
132	„
133	…
134	†
135	‡
136	ˆ
137	‰
138	Š
139	‹
140	Œ
141	•
142	Ž
143	•
144	•
145	‘
146	’
147	“
148	”
149	•
150	–
151	—
152	˜
153	™
154	š
155	›
156	œ
157	•
158	ž
159	Ÿ
160	
161	¡
162	¢
163	£
164	¤
165	¥
166	¦
167	§
168	¨
169	©
170	ª
171	«
172	¬
173	-
174	®
175	¯
176	°
177	±
178	²
179	³
180	´
181	µ
182	¶
183	·
184	¸
185	¹
186	º
187	»
188	¼
189	½
190	¾
191	¿
192	À
193	Á
194	Â
195	Ã
196	Ä
197	Å
198	Æ
199	Ç
200	È
201	É
202	Ê
203	Ë
204	Ì
205	Í
206	Î
207	Ï
208	Ð
209	Ñ
210	Ò
211	Ó
212	Ô
213	Õ
214	Ö
215	×
216	Ø
217	Ù
218	Ú
219	Û
220	Ü
221	Ý
222	Þ
223	ß
224	à
225	á
226	â
227	ã
228	ä
229	å
230	æ
231	ç
232	è
233	é
234	ê
235	ë
236	ì
237	í
238	î
239	ï
240	ð
241	ñ
242	ò
243	ó
244	ô
245	õ
246	ö
247	÷
248	ø
249	ù
250	ú
251	û
252	ü
253	ý
254	þ
255	ÿ

REFERENCES

Croghan, Cary 2004. “PICTURE Perfect: In depth look at the PICTURE format” Proceedings of SouthEast SAS Users Group (SESUG) 2004. Nashville, TN Available at http://analytics.ncsu.edu/sesug/2004/TU03-Croghan.pdf.

Dilorio, Frank C 1991 SAS Applications Programming: A Gentle Introduction (Duxbury Series in Statistics & Decision Sciences).

Karp, Andrew 2006 “Getting in to the Picture (Format)”. Proceedings of SAS Users Group International (SUGI) 31. San Francisco, CA. Available at http://www2.sas.com/proceedings/sugi31/243-31.pdf.

Williams, Christianna 2015 'SAS Formats Top Ten” Proceedings of SAS Global Forum 2015 Available at https://support.sas.com/resources/papers/proceedings15/3155-2015.pdf.

SAS Documentation

https://v8doc.sas.com/sashtml/proc/zpicture.htm.

ACKNOWLEDGMENTS

I would like to acknowledge my co-workers in the BioStat unit of the Health Services Research and Development department at the Durham Veterans Affairs for their support and feedback.

CONTACT INFORMATION

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

Jennifer Lindquist
Durham Veterans Affairs Health Care System
Jennifer.Lindquist@va.gov