

Creepy Characters

Elizabeth Axelrod, Abt Associates Inc., Cambridge, MA



ABSTRACT

SAS® formats are extremely useful for categorizing and grouping ranges of values, but there are situations that can yield unexpected results. This paper illustrates several very creepy results, and cautions the user when identifying character ranges.

INTRODUCTION

Using ranges in character formats can be useful, but to get the desired results, you've got to think like your computer. I was reminded of this when I used a character format to identify a range of 5-digit diagnosis codes. Although most diagnosis codes are comprised exclusively of numbers, some contain characters, and because of this, they are defined as character variables. Naively, I wrote a character format to identify this range of values:

```
proc format;
  value $dx
    '23500'-'31699'='23500-31699'
    other=other;
run;
```

My sample data (Fig. 1) contains a variable called DIAG.

DIAG
23500
24567
30000
31699
3
316
235

Fig. 1

Here is the result when I apply the \$DX format to DIAG:

DIAG	Formatted Value
23500	23500-31699
24567	23500-31699
30000	23500-31699
31699	23500-31699
3	23500-31699
316	23500-31699
235	OTHER

Fig. 2

THE 1ST CREEPY THING

Most of the diagnosis codes (Fig. 2) are handled as we would expect – values that *seem* to be within the defined range – like '24567' – do, in fact, get formatted as we expect. But why did the '3' get trapped in the defined range? And why did the '316' get trapped in the range, but the '235' did not?

The answer lies in the twin facts that a) all the values are padded with blanks to 5 places, and b) the code for a blank is lower than all the letters and numbers. That is, if you were to sort a list of alphanumeric characters, blanks would always float to the top of the list. So '316 ' (316-blank-blank) gets trapped in our defined range ('23500'-'31699') because '316-blank-blank' is

less than '31699' (but greater than '23500'), and thus falls within our range. But '235 ' does *not* fall within the defined range for the very same reason – '235-blank-blank' is less than '23500', the lower bound of the range.

Okay, so you're thinking that it might have made sense to first convert DIAG to a numeric variable, and then apply a *numeric* format to identify my desired ranges. That might work in this instance, but only because the specified range in my format contains exclusively *numeric* values. What happens when we introduce alpha characters into the mix? We find ourselves in the company of *really* creepy things.

Take a look at this format:

```
proc format;
  value $mixed
    '01'-'99'='01-99'
    'AA'-'ZZ'='AA-ZZ'
    other    ='other'
  ;
run;
```

Our intuition might tell us that only *numeric* values should get trapped in the '01'-'99' range, and that only *alpha* characters will get trapped in the 'AA'-'ZZ' range. And what about values like 'A1' or '1A', which contain *mixed* alpha-numerics? In which range will they fall? You may be horrified to learn that the answer is: *It depends!*

THE 2ND CREEPY THING

The sort order of alphanumerics is *different* in ASCII and EBCDIC collating sequences. So if you write and test code on your PC, and then run that same code on your mainframe, you could get *different* results, even using the same code and the same data.

Look what happens when we apply the \$mixed format to data on a PC, which uses the ASCII collating sequence, and to the same data on a mainframe, which uses the EBCDIC sequence.

VAR	ASCII	EBCDIC	
1A	01-99	01-99	
A1	other	AA-ZZ	Yikes!
Z	AA-ZZ	AA-ZZ	
ZZ	AA-ZZ	AA-ZZ	
A	other	other	
AA	AA-ZZ	AA-ZZ	
9a	other	01-99	Yikes!
a9	other	other	

Fig. 3

Figure 3 illustrates that the two systems yield different results. It doesn't get much creepier than this!

Figure 4 shows the collating sequence of alphanumeric characters in ASCII and EBCDIC. Note that blanks float to the very top in both sequences. But in the ASCII collating sequence, numbers come before letters, and capital letters come before lower-case letters. Not so with EBCDIC, where lower-case letters come before caps, which come before numbers.

ASCII	EBCDIC	
		← blank
0	a	
1	b	
2	c	
3	A	
A	B	
B	C	
C	0	
a	1	
b	2	
c	3	

Fig. 4

SOLUTION

What's a programmer to do? The first thing is to familiarize yourself with the collating sequence of the system you are using. Beyond that, it really does depend on your specific situation – the incoming data, and the kind of ranges you are identifying.

In the very first example where I was trying to categorize a strictly numeric range of diagnosis codes I could have used some functions to help me first weed out some codes that I didn't want getting trapped in my specified range, such as:

- the compress function to eliminate embedded spaces,
- the length function to help me examine only codes with valid lengths,
- the verify function to include only codes that are strictly numeric.

I could also have defined several formats – a separate one for each valid length, and then apply the specific length format to codes of that length.

In the second example, if the range of values I am trying to identify is reasonably short, I could unravel the range into discrete values, and eliminate the problem entirely:

```
proc format;
  value $discrete
    'A1',
    'A2',
    'A3',
    'A4' = 'GOOD'
    OTHER = 'BAD '
run;
```

Similarly, I could use the IN operator:

```
if VAR IN('A1','A2','A3','A4') then ...
```

These are just a few paths you might take, and are not meant to provide you with an exhaustive set of tools or solutions to this sometimes complicated problem. Whatever approach you ultimately use, you would be wise to generate some test data that will thoroughly exercise your code, and prove to yourself that you are trapping only your intended values.

CONCLUSION

Using character ranges in formats can yield unexpected results, unless you are familiar with the sort order that your system applies to alphanumeric characters.

CONTACT INFORMATION

Elizabeth Axelrod
 Abt Associates Inc.
 55 Wheeler Street
 Cambridge, MA 02138
 (617) 349-2458
 elizabeth_axelrod@abtassoc.com

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.