Converting Numeric Variables and Character Variables in SAS

Randall Reilly; Covance Clinical Pharmacology; Madison, WI

INTRODUCTION

This paper gives a general understanding of how to convert numeric and character SAS variables correctly.

CONVERTING NUMERIC VARIABLES TO CHARACTER VARIABLES

The simpliest way to convert numeric data to character data is using the PUT function. Converting numbers to characters is quite easy.

If you have only whole numbers use the following statement:

```
charvar = STRIP(PUT(numvar, 8.));
```

This will output the character values as whole numbers

Example:

FROM	TO
7	7
69	69
34	34
12	12

***

You should always format the new character variables (i.e. format charvar $8.) and you should always strip the new variable since numeric data is right justified and characters are left justified. If you don't you may run into the following:

format numvar 8. charvar $5.

If numvar is not justified you will, in most cases get no data populating charvar. In other cases you will only get partial data.

Example:

NUMERIC 8.	CHARACTER $5
Positions - 12345678	Positions - 12345xxx
567.4	56
16	null
801234569	801

***

If you have numeric data (subject numbers for example). If you want to convert to a character variable and still have the sort sequentially then you must zero fill the data.

```
charvar = STRIP(PUT(numvar, z2.));
```

Example:

FROM	TO

1	01
2	02
3	03
11	11
12	12
26	26

If you tried to convert the data using the first example:

```
charvar = STRIP(PUT(numvar, 8.));
```

The ouput will be a character variable, which will sort as (remember a character 1 in the first position sorts before a character 2):

Example:

FROM	TO
1	1
11	11
12	12
2	2
26	26
3	3

***

If you have a variety of numeric data depending on how you want the data stored in a character variables you can convert the numeric data a variety of ways:

1) if you just want to store the raw values you can use the BEST format.

```
charvar = STRIP(PUT(numvar, best32.));
```

This will store the data exactly as you see the numeric data

Example:

FROM	TO
7.1	7.1
23.87	23.87
12	12
697.45	697.45
998.9	998.9
53702.12456	53702.12456

2) if you want to add trailing zeroes to the data you use the following:

```
charvar = STRIP(PUT(numvar, 8.2));
```

This will store the data out to 2 decimal places

Example:

FROM	TO
7.1	7.10
23.87	23.87
12	12.00
697.45	697.45
998.9	998.90
53702.12456	53702.12

CONVERTING CHARACTER VARIABLES TO NUMERIC VARIABLES

The simpliest way to convert numeric data to character data is using the INPUT function.

If you have only numeric data as an integer in a character variable use the following statement:

```
numvar = INPUT(charvar, 8.);
```

Example:

FROM	TO
7	7
69	69
34	34
12	12

Remember: the numeric data is right justified and characters are left justified.

***

NOTE Do not ever use this statement to convert a character to a numeric:

```
 numvar = charvar * 1;
```

SAS will convert it but you will have to deal the FUZZ factor and other potential problems in your data.   If you want to know more about the FUZZ factor then consult a SAS manual – it can be a very difficult function to understand

You will get the following in the log:

```
NOTE: Character values have been converted to numeric values at the places given by:
(Line):(Column).
```

***

Because a numeric variable does not display trailing zeroes in the output.   The easiest thing to do when you don't know how the numeric data is stored in a character variable is:

```
numvar = INPUT(charvar, best32.);
```

Example:

FROM	TO
7.10	7.1
23.87	23.87
12.00	12
697.45	697.45

998.90	998.9
53702.12	53702.12

***

Because a character variable can hold both alpha and numeric data sometimes it make it difficult to convert the data. If you have any non-numeric character in the character variable the output will be missing and your log will contain a `NOTE: Invalid Data` message everytime you have a non-numeric character in your data. So you could get 10, 20, 50 or more notes in your log.

To solve this problem you can use the modifiers `?` and `??`. Both modifiers suppress the note message in your error log, but ?? modifier also resets the automatic error variable to 0, eliminating the error condition flagged because of the invalid data. In both cases the offending variable will still be set to missing, but our log has been cleaned up.

```
numvar = INPUT(charvar, ?? best32.);
```

Example:

FROM	TO
7.10	7.1
23.87	23.87
Sally Fisher	.
697.45	697.45
98.7C	.
53702.12	53702.12

CONVERTING VARIABLES USING PROC FORMAT

When converting variables to a new variables use the following table to figure out which INPUT or PUT statement you should use and how your PROC FORMAT statement should be written:

Converting:	INPUT/PUT statement	PROC FORMAT
Character to numeric	INPUT	INVALUE $fmt
Character to character	PUT	VALUE $fmt
Numeric to character	PUT	VALUE fmt
Numeric to numeric	INPUT	INVALUE fmt

CONVERTING A CHARACTER VARIABLE TO A NEW CHARACTER VARIABLE USING PROC FORMAT

You can also convert a character variable to a new character variable using PROC FORMAT. If you have the following case:

You want to change abbreviations to the full text.

1) You will need a PROC FORMAT statement

2) You need a VALUE statement along with a name for your format ($bdsysf). When converting from character to character the format name needs to start with a $. Also close the VALUE statement with a semicolon (;). It is usually best to keep the format name less than or equal to 8 characters including the “$”.

```
proc format;
  value $bdsysf  'BODY' = 'BODY AS A WHOLE'
                 'CV'   = 'CARDIOVASCULAR'
                 'DIG'  = 'DIGESTIVE'
                 'ENDO' = 'ENDOCRINE'
                 'HAL'  = 'HEMIC AND LYMPHATIC'
                 'MAN'  = 'METABOLIC AND NUTRITIONAL DISORDERS'
                 'MS'   = 'MUSCULOSKELETAL'
                 'NER'  = 'NERVOUS'
                 'RES'  = 'RESPIRATORY'
                 'SKIN' = 'SKIN AND APPENDAGES'
                 'SS'   = 'SPECIAL SENSES'
                 'UG'   = 'UROGENITAL'
                 Other  = '**ERROR**'
                 ;
run;
```

3) The following statement will convert the below:

`aesoctxt = STRIP(PUT(STRIP(aesoc), $ bdsysf.);` (Remember to format the new variable `aesoctxt` in your data step).

Example:

FROM	TO
UG	UROGENITAL
NER	NERVOUS
SSS	**ERROR**
DIG	DIGESTIVE

***

The statement

`Other  = '**ERROR**'`

will convert anything that doesn’t match any previous values to `'**ERROR**'` since it is a special value. If you don’t use this statement then any unformatted value will come back unchanged. Special values also include LOW and HIGH.

Example:

FROM	TO
UG	UROGENITAL
NER	NERVOUS
SSS	SSS
DIG	DIGESTIVE

Notice that the SSS did not convert to a long name because it isn't in the PROC FORMAT.

****

The DEFAULT option: When using PROC FORMAT the maximum length of the output value depends on the length of the first formatted value. In the above case the first formatted value is `'BODY AS A WHOLE'` which is 15 character long. This means that anything longer than 15 characters will be output as 15 characters, so `'METABOLIC AND NUTRITIONAL DISORDERS'` will be output as `'METABOLIC AND NU'`.

The solution is to use the DEFAULT=length option. The statement `value $bdsysf` should be written as `value $bdysysf (DEFAULT=50).`

***

In addition your PROC FORMAT statement can use ranges or lists.:

```
proc format;
 value $sigfmt
                'H'   = 'High'
                'L'   = 'Low'
                'U'   = 'Unknown'
                '*' , 'A'  = 'Abnormal'
                'N'   = 'Normal'
               ;
run;

proc format;
 value $regfmt
                'A'-'F'   = 'Monday'
                'G'-'L'   = 'Tuesday'
                'M'-'R'   = 'Wednesday'
                 'S'-'Z'   = 'Thursday'
               ;
run;
```

CONVERTING A CHARACTER VARIABLE TO A NUMERIC VARIABLE USING PROC FORMAT

Converting a character variable to a numeric variable is similar to the above example except you use an INPUT statement instead of a PUT statement. In your PROC FORMAT you use INVALUE instead of VALUE.

```
proc format;
invalue $rtdfmt
              'NOT RELATED'  = 1
              'POSSIBLE'   = 2
              'PROBABLE'   = 3
              'DEFINITE'   = 4
               'UNKNOWN'   = 5
```

```
                Other       = 9999
               ;
run;
aertdnum = INPUT(STRIP(UPCASE(aertd)), $rtdfmt.);
```

***

A few useful options that the INVALUE statement has that the VALUE statement does not are the options JUST and UPCASE.

If you rewrite the statement:

`invalue rtdfmt` as `invalue rtdfmt (JUST UPCASE)`

You can write the conversion statement as

```
aertdnum = INPUT(aertd, $rtdfmt.);
```

since the variable aertd is stripped and upcased by the options JUST and UPCASE.

CONVERTING A NUMERIC VARIABLE TO A CHARACTER VARIABLE USING PROC FORMAT

Converting a numeric variable to a character variable is almost the same as converting a character variable to a new character variable. The only exception is that the value name (`lbcat` below) doesn't start with a $ (i.e. `$lbcat`) and the unformatted values are not in quotes.

```
proc format;
value lbcat (DEFAULT =100)
            1 = 'Chemistry: Metabolic Substrates'
            2 = 'Chemistry: Hepatic Status Liver Function Tests'
            3 = 'Chemistry: Renal Status Serum Metabolites'
            4 = 'Chemistry: Serum Electrolytes'
            5 = 'Chemistry: Other'
            6 = 'Hematology: CBC'
            7 = 'Hematology: Differential Counts'
            8 = 'Hematology: Coagulation'
            9 = 'Urinalysis: General'
            10 = 'Urinalysis: Dipstick'
            11 = 'Urinalysis: Microscopics'
            12 = 'Urinalysis: Other'
            ;
run;
```

lbcatname = STRIP(PUT(lbcatnum, lbcat.);

***

Again, remember to include a format statement in your data step (i.e. format lbcatname $100.) and use the DEFAULT option rather than trusting that SAS will create everything correctly for you.

***

You can also create a variable by checking for numeric ranges and output the result depending on which group the data falls in. This can be useful when trying to group data in different groups so that you analyze.

```
proc format;
value crclfmt (default=10)
    LOW- <30      = ‘Mild’
     30 - 60      = ‘Moderate’
     >60 - HIGH   = ‘Severe’
       ;
run;
```

CONVERTING A NUMERIC VARIABLE TO A NEW NUMERIC VARIABLE USING PROC FORMAT

Converting a numeric variable to a new numeric variable is almost the same as converting a character variable to a numeric variable. The only exception is that the value name (tptnumf below) doesn't start with a $ (i.e. $ tptnumf) and the unformatted values are not in quotes.

```
proc format;
invalue tptnumdc
                    1=     0
                    2=     0.5
                    3=     1
                    5=     2
                    6=     4
                    7=     6
                    8=     8
                    9=     10
                    10=    12
                    11=    18
                    12=    24
                    ;
run;
```

This statement will convert a numeric to a numeric.

tpthour = INPUT(tptnum, tptnudc.);

You will get the following NOTE in your log, but it will not affect the data:

```
NOTE: Numeric values have been converted to character values at the places given by:
```

OTHER TYPES OF CONVERSIONS USING PROC FORMAT

You can also use a PICTURE statement in PROC FORMAT to create or present data a certain way. The PICTURE statement creates formats for numeric variables only.

The first examples are converting SAS DATE and TIME variables. Remember a SAS DATE and TIME variable is considered numeric.

```
proc format;
  picture slashdate (default=11)
  other='%0d/%0b/%Y'  (datatype=date);
run;
```

SLASHDATE will convert a DATEx variable to DD/MON/YYYY format, which is not available in SAS.

```
format newdate $20.;
newdate = STRIP(PUT(datevar, slashdate.));
```

Example:

FROM	TO
24SEP2009	24/SEP/2009
01JAN2010	01/JAN/2010
06APR1962	06/APR/1962

```
***
proc format;
  picture tmfive(default=5)
  other='%0H:%0M' (datatype=time);
run;
```

The picture TMFIVE will give a character time with leading zeroes. In this case the format is taking a SAS time which when otherwise converting to a character variable using TIME5 will not display leading zeroes, so 08:34 becomes 8:34. The following PICTURE will format with leading zeroes based on the special characters used.

```
newtime = STRIP(PUT(timevar, tmfive.));
```

Example:

FROM	TO
8:34	08:34
12:00	12:00
1:59	01:59

There are a handful of permitted directives in SAS that allow you to customize a user-defined format. If you wanted to know the day of year you could use "%j" which would give you a value of 1–366.

***

Converting true numeric data you have the option of three types of picture permitted directives:

0 = print the digit suppressing the leading zeroes

1 to 9 = print the digit with the leading zeroes

Others = print the character unchanged.

***

If you a hand a list of social security numbers in a numeric variable (without the dashes) you could write the format:

```
proc format;
  picture ssfmt
  other= '999-99-9999';
run;
```

newssnum = strip(put(ssnum, ssfmt.);

Example:

FROM	TO
123456789	123-45-6789
98765432	098-76-5432
456321789	456-32-1789
8888	000-00-8888

If you were to change the picture from 9's to 0's which suppress the leading zeroes then you would have:

```
proc format;
  picture ssfmt
  other= '000-00-0000';
run;
```

newssnum = strip(put(ssnum, ssfmt.);

Example:

FROM	TO
123456789	123-45-6789
98765432	987-65-432
456321789	456-32-1789
8888	888-8

There are many options you can use with the PICTURE statement such as FILL and PREFIX. Consult a SAS manual to see the complete list and how to use them.

REFERENCES

SAS Procedures Guide by SAS Institute by SAS Institute, Inc.

SAS Language: Reference by SAS Institute by SAS Institute, Inc.

CONTACT INFORMATION

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

Randall Reilly
Covance Clinical Pharmacology Services
Senior Programmer - Technical Services
3402 Kinsman Boulevard
Madison, WI 53704
Work Phone: 608-310-8289
Work E-mail: randall.reilly@covance.com