PharmaSUG 2013 - Paper CC29

Handling Dynamic Variable Types in SAS®

Venkat Lajapathirajan, PPD Inc., Hamilton, NJ

ABSTRACT

The variable type plays a vital role in SAS® datasets; it can be either numeric or character. If type changes on each data transfer, programmers will need to revise analysis dataset specifications and programs multiple times and it reducing efficiency, burning additional hours and affecting gross profit. This paper shows how to handle dynamic variable types through the macro %ConvertType – this macro converts the specified input variable(s) to character type and the revised variable(s) can be referred in further developments.

INTRODUCTION

If the entered data contains numbers only, then variable type would be numeric and we receive data in the same type for analysis dataset specification and program creation. I.e., numeric type will be referenced in these processes. In future transfers if the entered data contains both character and numbers then variable type changes to character and length gets assigned according to maximum length of the raw data, so the programmers will need to revise attributes in both specifications and programs throughout the study.

The macro %ConvertType uses simple PUT function to convert variable(s) to character type; we simply need to specify the raw dataset(s) and the respective variable(s) that needs to be processed. We could combine all macro calls into a single import program and run it for each data transfers; this will take care of the type conversions automatically and ignore redundant activities.

THE CONVERTTYPE MACRO

SYNTAX

```
%ConvertType(   DsetIn=,
                DsetOut=&DsetIn,
                ConvertVars=,
                Original_Char_VLength=Keep);
```

ARGUMENTS

DsetIn: Specify input dataset name.

DsetOut: Specify output dataset name, if missing defaults to “DsetIn”.

ConvertVars: Specify variable(s) that need to be converted to character type.

Original_Char_VLength: If you want to keep raw length of the character variable(s) then specify “Keep” else “Drop”, if missing defaults to “Keep”.

The macro does the following progressions:

- Count number of variables passed in the ConvertVars macro parameter and repeat below steps for all the variables.
- Read variable type and label from input dataset.
- If input variable type is numeric, use raw format and convert to character, the length and format of the derived variable will be $200.
- If input variable type is character and Original_Char_VLength is “Keep”, use raw character format and convert to character, the length and format will be same as raw.
- If input variable type is character and Original_Char_VLength is “Drop”, use raw character format and convert to character, the length and format of the derived variable will be $200, sometimes it might truncate data values so compare both raw and derived values and trigger warning messages in log if any differences are found.
- Attach labels to derived variables as in raw.
- Output the revised dataset with both raw and derived variables, the derived variables can be found with suffix of “_d”. E.g.: result_d.

LET'S DISCUSS WITH EXAMPLES

EXAMPLE 1: All the Laboratory parameters are being captured in the same LAB database, the initial raw dataset has numeric results only so the result variable LBORRES is numeric type, see Outputs 1 and 2 for data and description respectively.

	SUBJID	VISIT	LBTEST	LBORRES
1	0001	Day 5	APTT	79.5
2	0001	Day 5	Fibrinogen	283
3	0001	Day 5	Hematocrit	47.9
4	0001	Day 5	Hemoglobin	17.6
5	0001	Day 5	MCH	19.1

Output 1: Initial Lab data transfer

Variable	Type	Length	Format	Informat	Label
SUBJID	Character	200	$200.		Subject ID
VISIT	Character	200	$200.		Visit Name
LBTEST	Character	200	$200.		Lab Test Parameter
LBORRES	Numeric	8			Lab Result

Output 2: Numeric type result variable

```
%ConvertType(DsetIn                 = indata.lab,
             DsetOut                = lab,
             ConvertVars            = lborres,
             Original_Char_VLength  = drop);
```

The ConvertType macro has outputted the processed LAB dataset as in Output 3 below; it contains both raw and derived variables, as mentioned before the derived variables always named with suffix of "_d".

	SUBJID	VISIT	LBTEST	LBORRES	lborres_d
1	0001	Day 5	APTT	79.5	79.5
2	0001	Day 5	Fibrinogen	283	283
3	0001	Day 5	Hematocrit	47.9	47.9
4	0001	Day 5	Hemoglobin	17.6	17.6
5	0001	Day 5	MCH	19.1	19.1

Output 3: Processed LAB dataset

The second Lab data transfer has both numeric and character results as in Output 4.

	SUBJID	VISIT	LBTEST	LBORRES
1	0001	Day 5	APTT	79.5
2	0001	Day 5	Fibrinogen	283
3	0001	Day 5	Hematocrit	47.9
4	0001	Day 5	Hemoglobin	17.6
5	0001	Day 5	MCH	19.1
6	0001	Day 5	MCHC	36.7
7	0001	Day 5	MCV	52
8	0001	Day 5	RBC Morphology	Normal
9	0001	Day 5	Platelet Count	165
10	0001	Day 5	Protime	11.0

Output 4: Second LAB data transfer

Now the type of variable LBORRES has changed to character.

Variable	Type	Length	Format	Informat	Label
SUBJID	Character	200	$200.		Subject ID
VISIT	Character	200	$200.		Visit Name
LBTEST	Character	200	$200.		Lab Test Parameter
LBORRES	Character	200	$200.		Lab Result

Output 5: Character type result variable

At this point, if we do not have the **%ConvertType** macro we will have to revise specifications and programs and revalidate them, but now the job becomes quite easy, just run the macro call again it will create processed dataset as in Output 6.

	SUBJID	VISIT	LBTEST	LBORRES	lborres_d
1	0001	Day 5	APTT	79.5	79.5
2	0001	Day 5	Fibrinogen	283	283
3	0001	Day 5	Hematocrit	47.9	47.9
4	0001	Day 5	Hemoglobin	17.6	17.6
5	0001	Day 5	MCH	19.1	19.1
6	0001	Day 5	MCHC	36.7	36.7
7	0001	Day 5	MCV	52	52
8	0001	Day 5	RBC Morphology	Normal	Normal
9	0001	Day 5	Platelet Count	165	165
10	0001	Day 5	Protime	11.0	11.0

Output 6: Processed LAB dataset

The macro has assigned raw label to the derived variable as in Output 7, so the same can be referred in specifications and programming.

Variable	Type	Length	Format	Informat	Label
SUBJID	Character	200	$200.		Subject ID
VISIT	Character	200	$200.		Visit Name
LBTEST	Character	200	$200.		Lab Test Parameter
LBORRES	Character	200	$200.		Lab Result
lborres_d	Character	200	$200.		Lab Result

Output 7: Description of processed LAB dataset

EXAMPLE 2: Consider a Test dataset with integer and decimal values populated.

	i_num	i_char
1	ONE	1
2	12	12
3	123	123
4	1234	1234
5	12345	12345
6	1.1	1.1
7	12.12	12.12
8	123.01	123.01
9	1234.1	1234.1
10	12345	12345.0
11		1
12	0.1	0.1

Output 8: Test dataset

The variable I_NUM has been attached with a numeric format and I_CHAR is a regular character type variable.

Variable	Type	Length	Format	Informat	Label
i_num	Numeric	8	NUMFMT8.		Numeric Variable
i_char	Character	550			Character Variable

Output 9: Test dataset description

The macro converts raw variables to character type using raw format,

```
%ConvertType(DsetIn          = test,
             DsetOut         = test_revised,
             ConvertVars     = i_num i_char);
```

It outputs dataset as below, the variables i_num_d and i_char_d are derived from i_num and i_char respectively.

	i_num	i_char	i_num_d	i_char_d
1	ONE	1	ONE	1
2	12	12	12	12
3	123	123	123	123
4	1234	1234	1234	1234
5	12345	12345	12345	12345
6	1.1	1.1	1.1	1.1
7	12.12	12.12	12.12	12.12
8	123.01	123.01	123.01	123.01
9	1234.1	1234.1	1234.1	1234.1
10	12345	12345.0	12345	12345.0
11		1		1
12	0.1	0.1	0.1	0.1

Output 10: Processed Test dataset

EXAMPLE 3: The below dataset has a character variable with the length of $606.

	Char_var
1	If you define a character variable and assign the result of a numeric expression to it, SAS tries to convert the numeric result of the expression to a character value using the BESTw. format, where w is the width of the character variable and has a maximum value of 32. SAS then tries to execute the statement. If the character variable you use is not long enough to contain a character representation of the number, SAS prints a note to the log and assigns the character variable asterisks. If the value is too small, SAS provides no error message and assigns the character variable the character zero (0)

Output 11: Character variable

This macro call drops raw length and assigns $200 to the derived variable, since the length is lesser than raw it truncates data value and triggers warning message in the SAS® log as in Display 1.

```
%ConvertType(DsetIn                = char_test,
             DsetOut               = char_revised,
             ConvertVars           = char_var,
             Original_Char_VLength = drop);
```

```
WARNING: Derived variable is not matching with raw, suggest to keep original attributes
Raw variable
Char_var=If you define a character variable and assign the result of a numeric expression to it, S
AS tries to convert the numeric result of the expression to a character value using the BESTw. for
mat, where w is the width of the character variable and has a maximum value of 32. SAS then tries
to execute the statement. If the character variable you use is not long enough to contain a charac
ter representation of the number, SAS prints a note to the log and assigns the character variable
asterisks. If the value is too small, SAS provides no error message and assigns the character vari
able the character zero (0)
Derived variable
char_var_d=If you define a character variable and assign the result of a numeric expression to it,
 SAS tries to convert the numeric result of the expression to a character value using the BESTw. f
ormat, where w
NOTE: There were 1 observations read from the data set WORK.CHAR_TEST.
NOTE: The data set WORK.CHAR_REVISED has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Display 1: LOG message

CONCLUSION

This macro uses the simple type conversion techniques but it bypasses lot of rework and increases productivity in the dynamic database platform.

REFERENCE

http://support.sas.com/documentation/onlinedoc/91pdf/index.html

ACKNOWLEDGMENTS

I would to thank all the reviewers for your valuable comments and suggestions.

CONTACT INFORMATION

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

Name: Venkat Lajapathirajan
Enterprise: PPD Inc.,
Address: The Neuman Building, 3575 Quakerbridge Rd., Suite 201
City, State ZIP: Hamilton, NJ 08619-1205
Work Phone: 609-528-8113
E-mail: venkateshwaran.lajapathirajan@ppdi.com, vlajapathirajan@gmail.com

APPENDIX: The CONVERTTYPE MACRO

```
%macro ConvertType(DsetIn =,
                   DsetOut=&DsetIn,
                   ConvertVars=,
                   Original_Char_VLength=Keep);

  OPTIONS MISSING = "";

  DATA _null_;

       %* Count number of variables in the ConvertVars parameter ;
       CALL SYMPUTX("novars", count("&ConvertVars", "") + 1, "G");
  RUN;

  %PUT Number of variables count is &novars;

  %GLOBAL vars newvar;
  %LET vars = 1;

  %* Find out datatypes and labels ;
  %DO %WHILE(&vars lt (&novars + 1));
       %LET newvar = %SCAN(&ConvertVars, &vars);
       %PUT Processing variable is &newvar;

       %LET dsid=%SYSFUNC(open(&DsetIn,i));

       %GLOBAL &newvar.v &newvar.l;

       %DO i = 1 %TO %SYSFUNC(ATTRN(&dsid,nvars));
       %IF %UPCASE(&newvar) = %UPCASE(%SYSFUNC(VARNAME(&dsid,&i)))
              %THEN %DO;
               %LET &newvar.v = %SYSFUNC(VARTYPE(&dsid,&i));
                %LET &newvar.l = %SYSFUNC(VARLABEL(&dsid,&i));
               %GOTO _over_;
             %END;
       %END;
       %_over_: %LET rc=%SYSFUNC(CLOSE(&dsid));

       %PUT NOTE: &newvar datatype is &&&newvar.v and label is &&&newvar.l;
       %LET vars = %EVAL(&vars +1);
       %PUT Updated vars loop count &vars;
  %END;

  %* Derive variables for further purpose ;
  DATA &DsetOut;
       SET &DsetIn;

       %LET vars = 1;
       %DO %WHILE(&vars lt (&novars + 1));
       %LET newvar = %SCAN(&ConvertVars, &vars);

       %* If Numeric then derive a character variable ;
       %* If Character then keep data as is ;
       %IF "&&&newvar.v" eq "N" %THEN %DO;
                    IF &newvar ne .
                   THEN &newvar._d = STRIP(PUTN(&newvar, VFORMAT(&newvar)));
             %END;
             %ELSE %IF "&&&newvar.v" eq "C" %THEN %DO;
                    %IF %UPCASE(%SUBSTR(&Original_Char_VLength, 1, 1)) = K %THEN %DO;
                        IF &newvar ne "" THEN
                           &newvar._d = STRIP(PUTC(&newvar, VFORMAT(&newvar)));
                    %END;
```

```
                    %ELSE %DO;
                           LENGTH &newvar._d $200;
                           IF &newvar ne "" THEN DO;
                            &newvar._d = STRIP(PUTC(&newvar, VFORMAT(&newvar)));
                           IF &newvar._d ne &newvar tHEN DO;
PUT "WARNING: Derived variable is not matching with raw, suggest to keep original
attributes";
PUT "Raw variable " &newvar=;
PUT "Derived variable " &newvar._d=;
                            END;
                           END;
                    %END;
             %END;

             %* Assign label only if available in raw ;
             %IF "&&&newvar.l" ne "" %THEN %DO;
                    LABEL &newvar._d = "&&&newvar.l";
             %END;
       %LET vars = %EVAL(&vars +1);
       %END;
  RUN;

%mend ConvertType;
```