ABSTRACT
In SDTM domains, all character variables are limited to a maximum of 200 characters due to FDA requiring datasets in SAS v5 transport format. Text more than 200 characters long should be stored as a record in the SUPP--dataset. But Comments (CO) and Trial Summary (TS) domains are allowed to add variables for the purpose of handling text exceeding 200 characters. To improve readability the text should be split between words not just broken the text into 200-character - i.e., when text is longer than 200 characters in CO domain, additional variables COVAL- COVALn will be derived. The first 200 characters of the comment will be in COVAL, the next 200 in COVAL1, and additional text stored as needed to COVALn.

This paper presents the AddVar macro, which splits a long text variable into a set of smaller variables without truncating an intact word and automatically generate COVAL- COVALn in CO domain or TSVAL- TSVALn in TS domain.

AddVar checks that the user inputs are valid and that the specified split character does not already exist in the input variable. The newly created variables are added to the output data set.

INTRODUCTION
When creating SDTM datasets, the maximum length of a character variable is 200 characters. Text over 200 characters needs to be split into smaller strings without truncating an intact word not just be broken the original string into 200 character chunks. All long text strings in a SDTM conversion should be handled in this fashion, so it seemed appropriate to create a macro to handle the work.

This paper introduces a macro that can be called in the body of a SAS program to add variables for dataset that contains one or more lengthy variables. The macro should be called once for each variable to be split, and should be called in open code, outside of a DATA step.

MACRO PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN_DATA</td>
<td>Name of input dataset. This is a REQUIRED parameter and there is no default</td>
</tr>
<tr>
<td>IN_VAR</td>
<td>Name of input variable. This is a REQUIRED parameter and there is no default</td>
</tr>
<tr>
<td>SPLITCHAR</td>
<td>Split character. Space, forward slash [/], backslash [/] cannot be used as a split character. This is a REQUIRED parameter and the default is ~.</td>
</tr>
<tr>
<td>MAXLEN</td>
<td>Maximum length of a split part. This is a REQUIRED parameter and the default is 200</td>
</tr>
<tr>
<td>OUT_DATA</td>
<td>Name of output dataset. This is a REQUIRED parameter and there is no default</td>
</tr>
<tr>
<td>OUT_PRE</td>
<td>Prefix label for newly created variables. Parameter call should just use text. This is a REQUIRED parameter and there is no default.</td>
</tr>
</tbody>
</table>

Table 1. AddVar Parameters

METHODOLOGY
AddVar initially performs some checks on the user-supplied parameters –forward slash [/], backslash [/] cannot be used as a split character and the specified split character does not already exist in the input variable.

Following that, the macro checks to see if the length of the input variable is shorter than &MAXLEN – if it is, the macro redefines the length of the variable COVAL (200). If it isn’t, the macro uses the PRXCHANGE function to insert split character in the input variable according to the maximum length of a split part (&MAXLEN) and uses the COUNT function to determine how many variables will need to be created.

Syntax: PRXCHANGE(regular-expression-id|perl-regular-expression, times, source). Example:

```
VAR=prxchange('s/./{1,200})((\s|\$)/\1~/', -1, cats(VAR));
```

Regular expression visualization:
AddVar looks at character &MAXLEN +1 – if it’s a space, the split character is inserting. Otherwise, the macro locates the position of the right most breaking character and inserts a split character. The process is repeated on the remaining characters in the string until the end of the input variable.

And finally AddVar extracts individual words from the input variable based on the delimiter (&SPLITCHAR), and each chunk is assigned to a new variable. Variables are named based on the user supplied prefix (&OUT_PRE) – according to the SDTM standard (first new variable is unnumbered, additional new variables are numbered sequentially).

**MACRO SOURCE CODE**

```sas
%macro AddVar(in_data=, in_var=, splitchar=~, maxlen=200, out_data=, out_pre=);

data _null_
   call symput("ERR", "ERR"||"OR:"
run;

/*Checks*/
%if "&splitchar" = "/ or "&splitchar" = "/
   %then %do;
   %put &err / CAN NOT BE USED AS SPLIT CHARACTER. MACRO TERMINATING.;
   %goto exit;
%end;

/*Check if a split character in the input variable*/
%let flag=0;

data _null_
   set &in_data;
   if "&splitchar" not in ("", "/", "\") then do;
    if prxmatch("/\&splitchar/", &in_var) then call symputx('flag', 1);
   end;
run;
%if &flag=1 %then %do;
   %put &err A SPLIT CHARACTER(&splitchar) WAS FOUND IN VARIABLE &in_var.. AddVar TERMINATING;
   %goto exit;
%end;

/*Flag dataset*/
proc sql noprint;
   select distinct max(length(&in_var)) into :lngth
   from &in_data ;
quit;
%if &lngth > &maxlen %then %do;
```

**Figure 1. Regular Expression Visualization**

The Regular Expression Visualization shows that AddVar looks at character &MAXLEN +1 – if it’s a space, the split character is inserting. Otherwise, the macro locates the position of the right most breaking character and inserts a split character. The process is repeated on the remaining characters in the string until the end of the input variable.

And finally AddVar extracts individual words from the input variable based on the delimiter (&SPLITCHAR), and each chunk is assigned to a new variable. Variables are named based on the user supplied prefix (&OUT_PRE) – according to the SDTM standard (first new variable is unnumbered, additional new variables are numbered sequentially).
/*Insert split character*/
data &in_data;
   set &in_data;
   length _&in_var._ $32767;
   _&in_var._=prxchange("s/\d+/\1&splitchar/", -1, cats(&in_var));
drop &in_var;
run;

/*Number of variables*/
proc sql noprint;
   select cats(max(count(_&in_var._, ",", ","))) into :varn
        from &in_data
        ;
quit;
data &out_data;
   set &in_data;
   array vlst{*} $200 &out_pre. &out_pre.1 - &out_pre.&varn;
   do i=1 to %eval(&varn+1);
      vlst(i)=scan(_&in_var._, i, ",", ",");
   end;
drop _&in_var._ i;
run;
%end;
%else %do;
data &out_data;
   set &in_data(rename=COVAL=_COVAL_);
   length COVAL $200;
   COVAL=_COVAL_;
   drop _COVAL_;
run;
%end;

/*if exit conditions exist, program will jump to this point*/
%exit:
%mend AddVar;

TEST EXAMPLE
/*To mute "WARNING: The quoted string currently being processed has become more than
262 characters long. You may have unbalanced quotation marks."*/
options NOQUOTELENMAX;

/*Test data*/
data test;
   length COVAL $500.;
   COVAL="COMPARED TO PREVIOUS ECG, SIGNIFICANT CHANGES HAVE OCCURRED DUE TO
   CRITERIA FOR POSSIBLE ANTERIOR INFARCT GONE";
   output;
   COVAL="COMPARED TO BASELINE ECG, SIGNIFICANT CHANGES HAVE OCCURRED DUE TO NON-
   SPECIFIC INTRA-VENTRICULAR CONDUCTION DELAY IS SEEN COMPARED TO PREVIOUS ECG,
   SIGNIFICANT CHANGES HAVE OCCURRED DUE TO NON-SPECIFIC INTRA-VENTRICULAR CONDUCTION
   DELAY IS SEEN";
   output;
   COVAL="POSSIBLE WOLFF-PARKINSON-WHITE COMPARED TO BASELINE ECG, SIGNIFICANT
   CHANGES HAVE OCCURRED DUE TO POSSIBLE WOLFF-PARKINSON-WHITE IS SEEN COMPARED TO
   PREVIOUS ECG, SIGNIFICANT CHANGES HAVE OCCURRED DUE TO POSSIBLE WOLFF-PARKINSON-WHITE
   IS SEEN";
   output;
   COVAL="COMPARED TO BASELINE ECG, SIGNIFICANT CHANGES HAVE OCCURRED DUE TO AGE
   UNDETERMINED, SEPTAL MI IS SEEN COMPARED TO PREVIOUS ECG, SIGNIFICANT CHANGES HAVE
   OCCURRED DUE TO AGE UNDETERMINED, SEPTAL MI IS SEEN";

CONCLUSION

The macro described in this paper is a very useful tool which can be used to derive COVAL-COVALn or TSVAL1 - TSVALn. In addition, the macro will perform automatically, regardless of length of input variable. So the users do not need to know how many variables will be added – the macro will determine that for them. While the macro was developed in response to specific needs found in SDTM conversions, it can be used for other cases where longer strings need to be broken apart. And the usage of PRXCHANGE function in this macro also can be used to align a column in TXT/LST/PDF output.

REFERENCE


**ACKNOWLEDGMENTS**

I would like to express my deep gratitude to my manager Shenglin Zhang for his support. I’d also like to thank my colleague Susie Yu for helpful discussion and proof reading.

**CONTACT INFORMATION**

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

- Name: Xianhua Zeng
- Enterprise: PAREXEL International
- Address: 20F, Taiping Finance Tower, No. 488, Middle YinCheng Road, Pudong
- City, State ZIP: Shanghai, 200120
- Work Phone: +86 21-51118305
- Fax: +86 21 61609196
- E-mail: huazizeng@gmail.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.