PharmaSUG 2014 - Paper CC20

BreakOnWord: A Macro for Partitioning Long Text Strings at Natural Breaks

Richard Addy, Rho, Chapel Hill, NC
Charity Quick, Rho, Chapel Hill, NC

ABSTRACT

Breaking long text strings into smaller strings can be tricky, if splitting the string needs to be based on something more than simply length. For example, in SDTM domains, character variables are limited to 200 characters. Excess text need to be placed into supplemental datasets (and the variables there are also limited to 200 characters) - but it's not sufficient to just break the text into 200-character chunks; the text needs to be split between words.

This paper presents the BreakOnWord macro, which breaks a long text variable into a set of smaller variables of a length specified by the user. The original text is partitioned text at natural breaks - spaces, as well as user-supplied character values. The macro will create as many variables as necessary, and the variables are named using a user-supplied prefix and an ascending suffix. The user has the option of naming the series of variables beginning with the prefix only (creating SDTM-friendly variable names: AETERM, AETERM1, AETERM2, for example).

BreakOnWord checks that the user inputs are valid (the specified data set exists, and the specified long text string is present in that data set) and that the variables to be created by the macro are not already present. The newly created variables are added to the input data set, or, optionally, the macro can output a new data set. BreakOnWord requires SAS® of 9.1 or higher.

INTRODUCTION

When designing and creating SDTM datasets, the maximum length of a character variable is 200 characters. Longer strings need to be broken into smaller strings, and there is a strong preference to use natural breaks when the smaller strings are created – i.e., not to just break the original string into 200 character sections. 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. While the macro was initially designed to handle long strings based on restrictions in SAS datasets, we decided to generalize it to handle strings with a maximum length provided by the user.

This paper introduces a macro that can be called in the body of a SAS program to modify a 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. The macro features parameters that allow the user to specify the name, label, and length of the output variables as well as any special characters (in addition to a space) that should be used to determine break points. As many new variables as are required to break the original string into smaller strings will be created in the output data set.

MACRO PARAMTERS

DSETIN	Name of input dataset. This is a REQUIRED parameter and there is no default
DSETOUT	Name of input dataset. This is a REQUIRED parameter and the default is &DSETIN
INSTRING	Name of long text string to parse. This is a REQUIRED parameter and there is no default
MAXLEN	Prefix to be used in newly created variables. This is a REQUIRED parameter and there is no default
OUTBASE	Prefix label for newly created variables. Parameter call should just use text. This is a REQUIRED parameter and there is no default.
OUTLABEL	Prefix label for newly created variables. Default is OUTBASE. Parameter call should use just text and no quotes
BLANKSTART	[YES/NO] YES: First created variable is named &OUTBASE and first created label is &OUTLABEL NO: First created variable is named &OUTBASE.1and first created label is &OUTLABEL.1 This is a REQUIRED parameter and the default is YES

DROPSPACE	[YES/NO] Drop the breaking character if that character is a space (i.e., trim the output vaiable. This is a REQUIRED parameter and the default is YES
BRKCHARS	List of characters to be included as break characters in addition to a space. This is an OPTIONAL parameter and there is no default. Single and double quotes cannot be included as an additional breaking character

Table 1**. BreakOnWord Parameters**

METHODOLOGY

BreakOnWord initially performs some checks on the user-supplied parameters – required parameters must be supplied, the data set specified must exist, the variable specified must exist within that data set, and quotation marks cannot be used as breaking characters.

Following that, the macro determines how many variables will need to be created. For each distinct value of the input string that is longer than the user-supplied maximum length (&MaxLen), the macro starts by copying the value into a temporary sting. BreakOnWord looks at character &MaxLen +1 – if it's a space, the first &MaxLen characters are taken as a chunk (i.e., the substring ends at the end of a word). If there are no breaking characters within the first &MaxLen characters, it's also taken as a chunk. Otherwise, the macro locates the position of the right most breaking character (a space, as well as any additional character supplied by the user), and takes everything up to that as a chunk. The process is repeated on the remaining characters in the string until fewer then &MaxLen characters remain. A running total of the chunks required is kept; if there are no values of the input string greater than &MaxLen, a single new variable will need to be created.

Once the macro knows how many variables will need to be created, it goes about actually creating them. The macro checks to see if the variables it will create already exist in the input dataset – if they do, and are numeric, the macro stops. If one or more variables already exist, and are shorter than &MaxLen, a warning is generated. Using the same logic as before, BreakOnWord breaks down each value of the input string into smaller chunks, and each chunk is assigned to a new variable. Variables are named based on the user supplied prefix – the user can chose to name the variables according to the SDTM standard (first new variable is unnumbered, additional new variables are numbered sequentially) or by numbering all new variables sequentially. If the user wishes, a label is applied to the new variables. The new variables are trimmed of leading and trailing spaces by the DROPSPACE parameter default unless otherwise specified by the user. If none of the newly created variables already exist in the input data set, they will be set to length &MaxLen.

BREAKONWORD

```
%macro BreakOnWord( dsetin=, dsetout=, instring=, maxlen=200, outbase=, outlabel=,
                    blankstart=YES, dropspace=YES, brkchars=);
   %* local macro vars;
   %local GotData dsid GotVar rc PosTrunc FoundNum BrkChars2 Varlist Lbllist
          Varlist2 Lblsttm Continue Lname Mname PrevDef NewVar;

   %* Required parameters are present;
   %if %nrbquote(&dsetin) = %then %do;
      %put REQUIRED PARAMETER dsetin DOES NOT EXIST. BREAKONWORD WILL STOP;
      %goto exit;
   %end;
   %if %nrbquote(&instring) = %then %do;
      %put REQUIRED PARAMETER instring DOES NOT EXIST. BREAKONWORD WILL STOP;
      %goto exit;
   %end;
   %if %nrbquote(&maxlen) = %then %do;
      %put REQUIRED PARAMETER maxlen DOES NOT EXIST. BREAKONWORD WILL STOP;
      %goto exit;
   %end;
   %if %nrbquote(&outbase) = %then %do;
      %put REQUIRED PARAMETER outbase DOES NOT EXIST. BREAKONWORD WILL STOP;
      %goto exit;
   %end;
   %if %nrbquote(&blankstart) = %then %do;
      %put REQUIRED PARAMETER blankstart DOES NOT EXIST. BREAKONWORD WILL STOP;
      %goto exit;
```

```
%end;
%if %nrbquote(&dropspace) = %then %do;
  %put REQUIRED PARAMETER dropspace DOES NOT EXIST. BREAKONWORD WILL STOP;
  %goto exit;
%end;

%* Input dataset must exist;
%if &dsetin= %then %let GotData = NO ;
%else %do ;
   %if %sysfunc(exist(&dsetin)) %then %let GotData = YES;
   %else %let GotData = NO ;
%end ;
%if &GotData=NO %then %do ;
   %put THE INPUT DATA SET &dsetin DOES NOT EXIST. BREAKONWORD WILL STOP;
   %goto exit;
%end;

%* Instring must be a variable in the input dataset;
%let dsid = %sysfunc(open(&dsetin));
%if (&dsid) %then %do;
   %if %sysfunc(varnum(&dsid,&instring)) %then %let GotVar = YES ;
   %else %let GotVar = NO ;
   %let rc = %sysfunc(close(&dsid));
%end;
%if &GotVar = NO %then %do;
  %put INPUT VARIABLE &instring DOES NOT EXIST. BREAKONWORD WILL STOP;
  %goto exit;
%end;

%* Additional break characters can not contain quotes;
%let continue=YES;
data _null_;
   if index("&brkchars", "'") > 0 then do;
      call symputx('continue', 'NO');
   end;
   if index("&brkchars", '"') > 0 then do;
      call symputx('continue', 'NO');
   end;
run;
%if &continue=NO %then %do;
   %put ADDITONAL BREAKING CHARACTERS CANNOT INCLUDE QUOTATION MARKS;
   %goto exit;
%end;

%* Handle some defaults;
%if &dsetout = %str() %then %do;
   %let dsetout = &dsetin;
%end;
%let blankstart = %upcase(&blankstart);
%let dropspace  = %upcase(&dropspace);

%* Create a single-quoted, comma separated list of a space and any additional;
%* breaking characters provided;
%let brkchars2 = ' ';
%if &brkchars ne %then %do;
    data _null_;
        brkchars2 = "' '"
           %do j = 1 %to %length(&brkchars);
                    ||  ", '%substr(&brkchars,&j,1)'"
           %end;    ;
        call symputx( 'brkchars2', brkchars2);
   run;
%end;
```

```
%* Find records where text variable is longer than the length of the output;
%* variables;
proc sql;
  create table __maxlen as
   select distinct &instring, length(&instring) as maxval
   from &dsetin
   order by maxval
   ;
quit;

%* see if there are any values that will need to  be broken apart;
%let continue = NO;
data _null_;
   set __maxlen;
   call symputx( 'maxval',  strip(put(maxval, best.)));
   if maxval <= &maxlen then do;
      call symputx('continue', 'NO');
   end;
   else do;
      call symputx('continue', 'YES');
      if _N_ = 1 then newvarmax = 0;
      retain newvarmax;
      %* see how many variables will be needed to handle current string;
      _instring = trim(&instring);
      newvars = 0;
      done = 0;
      length _chopped $&maxlen;
      do while(not done);
         newvars = newvars + 1;
         if length(_instring) <= &maxlen then done = 1;
         else do;
            _chopped = substr(_instring,1,&maxlen);
           %* this segment ends on a word  - only look for a space;
            if substrn(_instring, &maxlen + 1, 1) = ' ' then _notch = &maxlen;
            %* no place to split, so just take the whole segment;
            else if indexc(_chopped, &brkchars2) = 0    then _notch = &maxlen;
            else do;
              %* find right-most instance of a breaking character;
               _notch = &maxlen - indexc(reverse(_chopped), &brkchars2) +1;
            end;
            _instring = substrn(_instring, _notch + 1);
         end;
      end;
      if newvars > newvarmax then newvarmax = newvars;
      call symputx( 'newvarn', strip (put( newvarmax, best.)));
   end;
run;

  %* if no outlabel parameter specified, use outbase;
  %if &outlabel = %then %let outlabel = &outbase;
  %else %let outlabel = &outlabel;

%if &continue = NO %then %do;
   %put NOTE: Maximum text length (&maxval characters) in &dsetin is less than;
   %put length of string to be created (&maxlen characters);
   %if &blankstart = YES %then %do;
      %let newvar = %upcase(&outbase);
      %let newlbl = %upcase(&outlabel);
   %end;
   %else %do;
      %let newvar = %upcase(&outbase.1);
      %let newlbl = %upcase(&outlabel.1);
```

```
   %end;
   data &dsetout;
      set &dsetin;
      label &newvar=%upcase(&newlbl);
      &newvar = &instring;
   run;
%end;
%else %do;
   %* Maximum length of string is greater than maximum length of text string to;
   %* be created - at least 2 variables will be created;

   %* generate a list of variables needed;
   %if &blankstart = YES %then %do;
      %let varlist = &outbase;
      %let lbllist = &outlabel;
      %let lblsttm = &outbase=&outlabel;
      %do j = 1 %to &newvarn-1;
         %let varlist = &varlist &outbase.&j;
      %let lbllist = &lbllist &outlabel.&j;
      %let lblsttm = &lblsttm &outbase.&j=&outlabel.&j;
      %end;
   %end;
   %else %do;
      %let varlist = ;
      %let lbllist = ;
      %let lblsttm = ;
      %do j = 1 %to &newvarn;
         %let varlist = &varlist &outbase.&j;
       %let lbllist = &lbllist &outlabel.&j;
       %let lblsttm = &lblsttm &outbase.&j=&outlabel.&j;
      %end;
   %end;

   %put BREAKONWORD: &newvarn (&varlist) variables will be needed ;

   %* see if input dataset is a work dataset or not.  ;
   data _null_;

      dsetin = upcase("&dsetin");
      if index(dsetin, '.') = 0 then do;
         call symputx('lname', 'WORK');
         call symputx('mname', dsetin);
      end;
      else do;
         call symputx('lname', substr(dsetin, 1, index(dsetin, '.')-1));
         call symputx('mname', substr(dsetin, index(dsetin, '.')+1));
      end;

      * placeholder;
      %* while we are here, generate a quoted list of variable names;
      varlist2 =  "'%scan(&varlist,1,%str( ))'"
                     %do j = 2 %to &newvarn;
                       ||  ", '%scan(&varlist,&j, %str( ))'"
                     %end;
                  ;
      call symputx( 'varlist2', varlist2);

   run;

   %* Check to see if variables have been previously defined.  If not, will;
   %* need to define them.  Throw a warning;
   %* if variable is previously defined and length < &maxlength, and error;
   %* out if its numeric;
```

```
%* Assume none of the new variables were previously defined;
%let PrevDef  = NO;
%* Assume none of the new variables were previously defined numeric;
%let FoundNum = NO;
%* Assume none of the new variables were previously defined as short;
%let PosTrunc = NO;

data _null_;
   set sashelp.vcolumn
    (where=(upcase(libname)="&lname" and upcase(memname) = "&mname"));
   name = upcase(name);
   if name in (%upcase(&varlist2)) then do;
      call symputx( 'PrevDef', 'YES');
      if length < &maxlen then do;
         call symputx( 'PosTrunc', 'YES');
      end;
      if type = 'num' then do;
         call symputx( 'FoundNum', 'YES');
      end;
   end;
run;

%if &FoundNum = YES %then %do;
   %* Variable needed by the macro has been defined, and it is a number.;
   %put ONE OR MORE VARIABLES NEEDED BY THIS MACRO (&VARLIST) IS DEFINED AS;
   %put NUMERIC IN THE INPUT DATASET. BREAKONWORD WILL STOP EXECUTING;
   %goto exit;
%end;
%else %do;
   %if &PosTrunc = YES %then %do;
      put One or more variables needed for this macro (&VARLIST) has been;
      %put previously defined as character, with a length less than the;
      %put maximum length defined for the macro (&maxlen).  The macro will;
      %put continue, but be aware that values may be truncated.;
   %end;
%end;

data &dsetout;
   set &dsetin;
   label &lblsttm;

   %if &prevdef = NO %then %do;
      length &varlist $&maxlen;
   %end;
   array bits (&newvarn) &varlist.;

   %* use similar logic as above, but actually populate variables;
   length _chopped $&maxlen;
   if length(&instring) < &maxlen then do;
      bits(1) = &instring;
   end;
   else do;
      _instring = trim(&instring);
      _newvars = 0;
      _done = 0;
      length _chopped $&maxlen;
      do while(not _done);
         _newvars = _newvars + 1;
         if length(_instring) <= &maxlen then do;
            %* leftover does not exceed max length;
            bits(_newvars) = _instring;
            _done = 1;
```

```
                end;
                else do;
                   %* Find notch - where to split the segment;
                   _chopped = substr(_instring,1,&maxlen);

                   * this segment ends on a word - only look for a space;
                  if substrn(_instring, &maxlen + 1, 1) = ' ' then _notch = &maxlen;
                  %* no place to split, so just take the whole segment;
                  else if indexc(_chopped, &brkchars2) = 0    then _notch = &maxlen;
                   %* find right-most instance of a breaking character;
                  else _notch = &maxlen - indexc(reverse(_chopped), &brkchars2)+ 1;

                   %* assign current segment;
                   bits(_newvars) = substrn(_instring,1, _notch);

                   %* trim if necessary;
                   %if &dropspace = YES %then %do;
                      bits(_newvars) = strip(bits(_newvars));
                   %end;

                   %* Set up for next pass through loop;
                   _instring = substrn(_instring, _notch+1);
                end; %*  length(_instring) > &maxlen;
             end; %* else do while not done;
          end; %* length(&instring) > &maxlen;

          drop _newvars _done _instring _chopped _notch;
       run;

       %* Clean up;
       proc datasets lib=work nolist;
           delete __maxlen __maxlen2;
       quit;
       run;

    %end;

  %* if exit conditions exist, program will skip to this point;
  %exit:

%mend BreakOnWord;
```

NOTES

The method BreakOnWord uses to determine how many variables will need to be created is complete, but inefficient. For specific situations (large data sets, cases where &MaxLen is large, etc.), the macro could be made faster by reducing the number of distinct values it looks at when determining how many variables will be needed. However, we felt that for general use, a robust, albeit slower, method of determining the number of new variables was appropriate.

By requiring BreakOnWord to be called outside of a DATA step, the macro is able to perform a few tasks more easily: checking to see if the variables that will be created are already present in the input data set, for example.

If the output variable is the same as the input variable (For example, if INSTRING=AETERM, OUTBASE=AETERM, BLANKSTART=NO), the input variable will have its values truncated appropriately, but its length will not be reset.

CONCLUSION

BreakOnWord is a useful macro for breaking apart long string variables into smaller strings at user-specified break points. The user does not need to know how many variables will be needed – the macro will determine that for them. BreakOnWord emphasizes robust functionality and features flexible options for variable naming, break character choices, labels, and string trimming. 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.

CONTACT INFORMATION

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

Richard Addy
Rho
6330 Quadrangle Drive
Chapel Hill, NC 27517
919-595-6579 (w)
919-408-0999 (f)
richard_addy@rhoworld.com
www.rhoworld.com
https://twitter.com/rhoworld

Charity Quick
Rho
6330 Quadrangle Drive
Chapel Hill, NC 27517
919-595-6594 (w)
919-408-0999 (f)
richard_addy@rhoworld.com
www.rhoworld.com
https://twitter.com/rhoworld