PharmaSUG AD08

Maximize the power of %SCAN using WORDSCAN utility

Priya Saradha, Edison, NJ

This paper demonstrates the power of %SCAN macro function using the user-defined utility, WORDSCAN, and illustrates the most effective way to pass input parameters for any acro. WORDSCAN is a powerful utility macro which splits a given string into new macro variables and counts the total number of individual words. The utility's key features are, any character can be used as a delimiter in the input string and any character (bound by the rules of SAS variable naming conventions) can be used as a prefix for the new macro variable names. Using the %SCAN macro function, the utility searches through the input string for the specified delimiter and identifies the words to be created as new macro variables. Each word, thus identified, is stored in separate macro variables. The macro variables are named by the utility using the prefix, provided as an input, followed by a sequence number. This naming convention helps the user to identify the order in which each word appears in the given string. Along with the new macro variables, the utility also outputs the number of words read from the input string into a macro variable. Thus, WORDSCAN can increase the robustness and efficiency of user-defined macros using the power of %SCAN macro function.

Introduction

In this day and age of data standardization, macros have become a necessity for the programming community. Almost every one of us have written macros for various purposes. While defining macros, it is a common practice to use mutiple input parameters to get user input for a specific operation. For instance, it is a general tendency to define 3 input parameters to accept baseline, actual and change variables as input for a macro which calculates statistical summary for continuous variables. Similarly, macro programmers tend to define multiple input parameters to accept variables names and their formats as user input. In both the situations, the macro programmers can reduce the number of input parameters by half. This will increase the efficiency in handling input parameters and decrease ambiguity for the macro user. WORDSCAN macro is created for the purpose of reducing the number of input parameters defined in a macro. The idea of grouping input parameters will add clarity to the macro call and increase the user-friendliness of any macro.

%SCAN macro function

The %SCAN and %QSCAN functions both search a text string for the nth word and return its value. The delimiters, if specified, function the same way as the delimiters in SCAN function in a DATA step. For an ASCII system the delimiters include blank *. < (+ | & ! $ *) ; ^ - / , % > \ )* (for EBCDIC the ¬ is substituted for the ^). %QSCAN removes the significance of all special characters in the returned value.

Syntax

%SCAN(variable,n[,delimiters])

Example

The macro variable &X below can be broken up using the %SCAN function.

```
%LET X=XYZ.ABC/XYY;
%LET WORD=%SCAN(&X,3);
%LET PART=%SCAN(&X,1,Z);
%PUT WORD IS &WORD AND PART IS &PART;
```

%PUT statement returns the following: `WORD IS XYY AND PART IS XY`

Using this functionality of %SCAN macro function, the utility WORDSCAN reads through the given input string to split into multiple words or segments.

WORDSCAN macro utility

Syntax and code :

```
%macro wordscan(string=,prefix=,delim=%str( ));

   /****** Count the # of words in the string ******/
  %global &prefix.c wordx;

    %let wordx=1;
    %let word=%scan(&string,%eval(&wordx),&delim);
    %global &prefix&wordx;
    %let &prefix&wordx=&word;

  %do %while(&word ne %str());
    %let wordx=%eval(&wordx+1);
    %let word=%scan(&string,%eval(&wordx),&delim);
    %global &prefix&wordx;
    %let &prefix&wordx=&word;
  %end;

  %let &prefix.c = %eval(&wordx-1);

%mend wordscan;
```

In the above macro definition, the first step is to scan the given STRING and count the number of words in the string identified by the delimiter (DELIM). The second step is to extract the individual words from the string based on the delimiter. Simultaneously, the individual words are output as macro variables along with the total count.

Usage: `%wordscan(string=%str(base,aval,chg),prefix=astat,delim=%str(,));`

This sample call stores the total number of words into *&astatc* and the creates *&astat1* through *&astat3* to store the individual words from the macro. A *%put ASTAT1=&astat1 ASTAT2=&astat2 ASTAT3=&astat3 ASTATCOUNT=&astatc;* statement will produce the output as *ASTAT1=base ASTAT2=aval ASTAT3=chg ASTATCOUNT=3*.

Applications of %WORDSCAN macro utility

In the real world, there are many situations where %WORDSCAN utility can be used. There are two such situations presented below. The first example is of moderate complexity and demonstrates the usage of WORDSCAN to calculate descriptive statistics. The second example, which is fairly easy, demonstrates the scenario of assigning attributes to the variables in a dataset.

Calculating statistics for continuous and discrete variables

Every SAS user has written programs to output descriptive statistics and almost everyone has created a macro for the same purpose. When creating such macros, the macro developer requires more than one parameter to input the analysis variables, the subset conditions for each analysis variable and also the statistics requested for each variable. The macro call below uses only one macro parameter (ANALYSIS) to receive all the said input values.

```
%stats(indst=adsl, byvars=trt01an,
        analysis=%nrstr(age[n=agen|mean=meanage]where age ne .,
                          race[npct]where race ne '',
                          heightb[mean=htmean|n=htn]where heightb ne .,
                          weightb[mean=wtmean|n=wtn]where weightb ne .));
```

The macro variable ANALYSIS uses 3 different delimiters to identify each component, viz., “[“, “]” and “,” to get the analysis variables, list of statistics, and the subset condition. It also uses “|” to separate the individual statistics requested. The macro code presented below shows how %WORDSCAN is used to separate out the components and use them as needed to compute the requested descriptive statistics and output results into a dataset.

/********** Macro to calculate descriptive statistics for continuous and discrete variables **********/

```
%macro stats(indst=, byvars=, analysis=);

%wordscan(string=&analysis,prefix=anlv,delim=%str(,[]));

%do j = 2 %to &anlvc %by 3;
  %wordscan(string=%str(&&anlv&j),prefix=anls&j,delim=%str(|));
%end;

%if &byvars ne %str() %then %do;
  proc sort data= &indst out=_tmpdst;
    by &byvars;
  run;
%end;
%else %do;
  data _tmpdst;
    set &indst;
  run;
%end;

%do i = 1 %to &anlvc %by 3;
  %let stat = %eval(&i+1);
  %let subset = %eval(&i+2);

  %if &&anlv&stat ne npct %then %do;
    proc univariate data = _tmpdst noprint;
      %if &byvars ne %str() %then %do; by &byvars; %end;
      var &&anlv&i;
      output out = stat_&&anlv&i
                     %do j = 1 %to %eval(&&anls&stat.c);
                        &&&anls&stat.&j
                     %end;
    ;
    &&anlv&subset;
    run;
  %end;
  %else %do;
  proc freq data=_tmpdst noprint;
    %if &byvars ne %str() %then %do; by &byvars; %end;
    tables &&anlv&i / out=_tmpfrq(rename=(count=n&&anlv&i) drop=percent);
    run;

    %if &byvars ne %str() %then %do;
    proc sort data=_tmpfrq;
      by &byvars;
      run;

    data stat_&&anlv&i(drop=bign n&&anlv&i);
      merge _tmpfrq bign;
      by &byvars;
      &&anlv&stat = compress(put(n&&anlv&i,8.))||
                   ' ('||compress(put(round(n&&anlv&i/bign*100,0.1),8.1))||')';
      run;
    %end;
    %else %do;
    data stat_&&anlv&i;
      set _tmpfrq;
      run;
    %end;
  %end;
```

```
%end;  /*** End of DO Loop ***/
%mend stats;
/************ Demonstration of %stats macro using ADSL dataset ***********/
proc sql;
  create table bign as
        select trt01an, count(usubjid) as bign from adam.adsl group by trt01an;
quit;

%stats(indst=adsl, byvars=trt01an,
        analysis=%nrstr(age[n=agen|mean=meanage]where age ne .,
                  race[npct]where race ne '',
                  heightb[mean=htmean|n=htn]where heightb ne .,
                  weightb[mean=wtmean|n=wtn]where weightb ne .)
      );
```

Thus, any seasoned macro developer can increase the user-friendliness of their macro by using the %WORDSCAN utility.

Assigning attributes for dataset variables

The next example discussed in this paper is for assigning attributes to dataset variables. Due to SDTM and ADaM standards, programmers have to assign attributes to the derived variables. This process is done for each trial. Many groups achieve this task by reading excel spreadsheets into macros which assign attributes to the dataset variables and many write individual attrib statements in their program. There are two variations presented here to demonstrate the application of %WORDSCAN utility to assign attributes for dataset variables. The first variation (ATTRIB1 macro) accepts the variable names, formats, and labels as input in a single macro parameter. The second variation (ATTRIB2 macro) accepts all the three components in 3 separate input parameters. Both the examples present the versatility of using %WORDSCAN in user programs and macros.

Variation # 1:

```
   /****** Macro variation 1 to assign attributes to dataset variables ******/
  %macro attrib1;
        %wordscan(string=&atrbvars,prefix=attrib,delim=%str(,|));

        %if &outdst eq %str() %then %do; &indst %end;
        data %if &outdst eq %str() %then %do; &indst %end; %else %do; &outdst %end;;
         attrib
           %do i = 1 %to &attribc %by 3;
              %let format = %eval(&i+1);
              %let label = %eval(&i+2);
                 &&attrib&i format=&&attrib&format... label="&&attrib&label"
            %end;
          ;
          set &indst;
        run;

  %mend attrib1;

  %attrib1(indst=intest,
           atrbvars=%str(studyid|$20|Study identifier,
                        usubjid|$20|Subject identifier),
           outdst=outtest
        );
```

This example presents the usage of 2 delimiters "," and "|" which are used to extract individual words from ATRBVARS parameter.

Variation # 2:

```
  %macro attrib2(indst=,atrbvars=,fmtvals=,lblvals=,outdst=);

        %wordscan(string=&atrbvars,prefix=dsvar,delim=%str(,));
        %wordscan(string=&fmtvals,prefix=varfmt,delim=%str(,));
```

```
       %wordscan(string=&lblvals,prefix=varlbl,delim=%str(,));

       %if &outdst eq %str() %then %do; &indst %end;
       data %if &outdst eq %str() %then %do; &indst %end; %else %do; &outdst %end;;
         attrib
             %do i = 1 %to &dsvarc %by 3;
               &&dsvar&i format=&&varfmt&i... label="&&varlbl&i"
             %end;
         ;
         set &indst;
       run;

 %mend attrib2;

 %attrib2( indst=intest,
           atrbvars=%str(studyid,usubjid),
           fmtvals=%str($20,$20),
           lblvals=%str(Study Number, Subject Number),
           outdst=outtest
                );
```

This example presents the usage of 1 delimiter, which is used to extract individual words from ATRBVARS parameter.

Conclusion

WORDSCAN is a versatile tool which can be used for building macros of any complexity and can also be used in individual programs for any specific purpose. This utility will certainly increase the user-friendliness and efficiency of any macro by reducing the number of input parameters.

References

%StrSrch – A Recursive SAS® Tool to Find and Replace Any String in Text Files, Wenyu Hu and Liping Zhang, Merck Research Labs, Merck & Co., Inc., West Point, PA

Acknowledgements

I thank Ju Zhang, VP, Tech Data Services for his support and encouragement. I am also thankful to my friends helped me write this paper.

Trademarks

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

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:
Priya Saradha
(409)-504-6070
priya.saradha@gmail.com