Search and Replace in SAS® Data Sets thru GUI

Edmond Cheng, Bureau of Labor Statistics, Washington, DC

ABSTRACT

In managing data with SAS®/BASE software, performing a search and replace is not a straight forward task (unlike when working within a word processor or spreadsheet program). Because time and productivity are valuable, a utility to search and replace data values would enable users to easily make repetitive modifications while working with SAS data sets.

This paper introduces a SAS program written in SAS/BASE, with a simple search and replace function through the use of a graphical user interface (GUI) in the SAS Display Manager environment. This program presents an intuitive way of searching through the contents of a SAS data set for specific character or numeric string patterns, with the option of replacing one string with another. In addition, the program also allows for various common search features such as 'column selection', ‘work copy’, and ‘print result’. Using the GUI input window, the variables entered by the user will be transferred directly into the macro parameter, without disrupting code internally stored. This program should prove useful by facilitating edits without additional programming; it also presents an interactive way in working with SAS/BASE.

INTRODUCTION

Database management requires changes and updates through time, to reflect accurate and timely representation of the business or purpose for which the database is intended. SAS/BASE software works well with processing of large data with a simple yet powerful programming language. Tasks such as adding records, updating information, or updating, require users to create a program code to perform specific task. Although this is the usual method, however, users might always like to have a handy application able them to perform repetitive tasks.

In this paper, the presented program is a quick approach in building a simple application for performing routine searching and replacing text with the use of Base SAS code. The TRANWRD function is used to control the search and replace functionality. TRANWRD converts one word to another within a string variable.

The graphical user interface is invoked thru the use of %WINDOW and %DISPLAY commands. The combined use of the two display environment commands provides a means of communication between the user and programming in a graphical format. By allowing the defined field in %WINDOW to accept user input and save the input as macro variable, the interaction creates a more intuitive way for users to store search and replace variables without dealing with programming code.

HOW THIS WORKS

To understand what is happening behind the application, the components are discussed below:

The TRANWRD FUNCTION - The TRANWRD function executes the searching and replacement functions in this application. The function replaces an entire word or phase within a string with another word. Think of the replace function in a word processing program. The TRANWRD not only replaces a single word in a text string, yet it also performs replacement of phases. The syntax is as follows:

TRANWRD (*source, target, replacement*)

Fig 1. Example

```
data _null_;
  color = 'red orange green' ;
  put "Before TRANWRD:  " color;
run;
```

Result

```
Before TRANWRD: red orange green
```

Fig 2. Example

```
data _null_;
  color = 'red orange green' ;
  color = TRANWRD(color,'orange','blue');
  put "After TRANWRD:  " color;
run;
```

Result

```
After TRANWRD: red blue green
```

Example, the variable COLOR contains the value ‘red blue green’ in the character string. By using the TRANWRD function, the argument *source* declares input variable to be ‘color’. Next, the argument *target* is the string of characters to search for, and while *replacement* is obviously the string of characters to be replaced. With three arguments, the TRANWRD function search for the string ‘orange’ within the source variable ‘color’, and then replaces it with another string ‘blue’.

Beyond the function TRANWRD, SAS offers various kind of character manipulation functions, such as SUBSTR, TRANSLATE, and PREL expression. While there hold advantages and disadvantages of each function; the TRANSWRD function is found to be the most applicable and simple method for the particular need in search and replace application. If one is interested in replacing individual characters instead, the more useful function is TRANSLATE.

The GRAPHICAL USER INTERFACE (GUI) DEFINITION – To create a user friendly and flexible environment for data input; the application combines the use %WINDOW and %DISPLAY commands. Both of these work well within Base SAS which requires nothing more than the arguments within the function. This allows for building a quick and easy interface for data entry, without dealing with additional programming language.

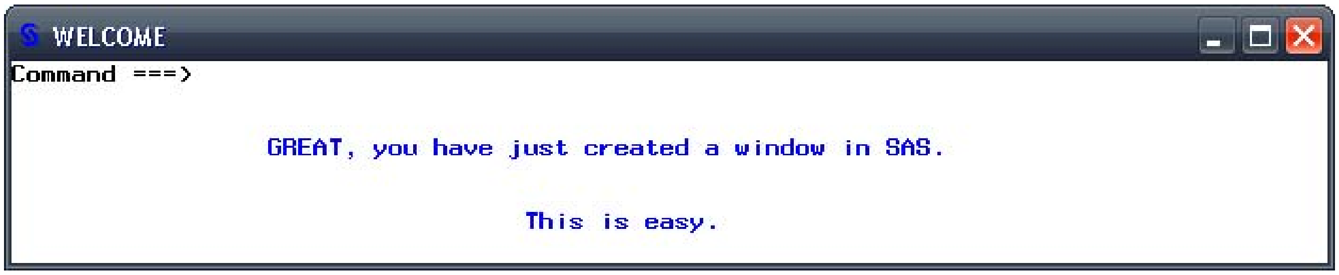
%WINDOW *window <window-options group-definitions> field-definition(s);*

%DISPLAY *window <.group> <NOINPUT > <BLANK > <BELL > <DELETE>;*

Fig 3.
Example

```
%window welcome color=white
           #3 @18 'GREAT, you have just created a window in SAS.'
              color=blue
           #6 @35 'This is easy.' color=blue;
%display welcome ;
```

Result



```
WELCOME
Command ===>

                GREAT, you have just created a window in SAS.

                                 This is easy.
```

Let’s take a look at the syntax for %WINDOW and %DISPLAY. Inserting the %WINDOW invoke the program to execute the window macro, *window* is the name given to this particular window, as in a filename for a file. *Windows-options* are extra features applied to the window as a whole. In the example, ‘color=white’ was chosen to turn the background of the window as white. *Group definition* is the name of group specified for the fields defined. The window can display different groups of fields collectively or individually, it is up to the user to determine if all or selective fields are display in the window at a given time. The example, we choose to leave it blank, so that all fields are displaying in one window. *Field-definitions* defines the string or macro variable to be displayed in the window. The field definitions area serves as building elements of the window. This is where we can specify the contents of information to be displayed by rows and columns.

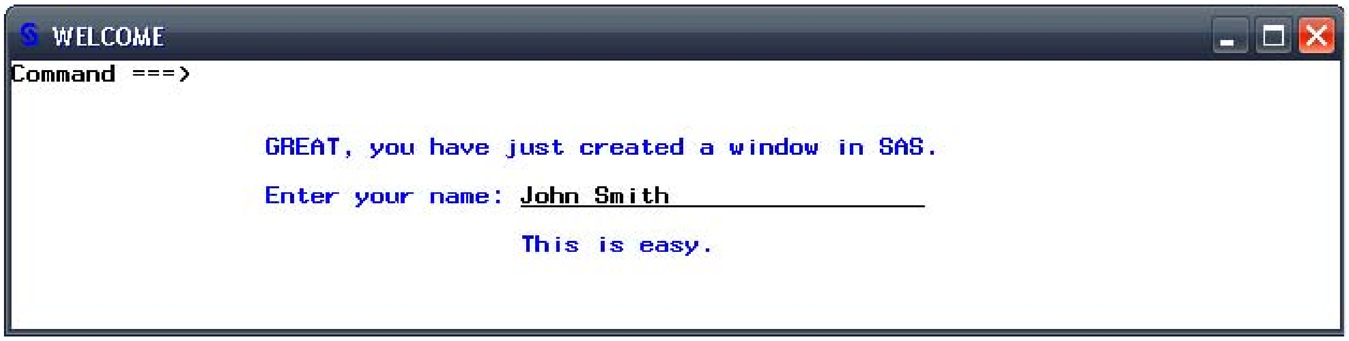
The %DISPLAY is used to display the fields from window created. Only one *group* of field is displayed for one %DISPLAY call. Since our example contains a single default window, the %DISPLAY calling for *window* displays all fields available. Additional *options* for each field increase flexibility and control. The example calls the display to bring up the *window* WELCOME, which shows a short message is being relayed to the user.

To enable user data input from a window, a macro variable can be used to store user entries as text characters. The results would then be saved into the specified macro variable and available in other part of the program. Think of this as the pop up dialog window as in your favorite regular word processing program.

Fig 4.
Example

```
%let name=;
%window welcome color=white
     #3 @18 'GREAT, you have just created a window in SAS.'
             color=blue
     #5 @18 'Enter your name:' color=blue
     #7 @35 'This is easy.' color=blue
     #5 @35 name 25 attr=underline;
%display welcome delete blank ;
%put You have entered the name: ;
%put &name;
```

Result



```
Command ===>

                GREAT, you have just created a window in SAS.

                Enter your name: John Smith

                                 This is easy.
```

A new macro variable ‘name’ has been declared with a length of 25 characters and underlining attribute. For each time when the %DISPLAY window is called, the blinking cursor will pause at the underlined entry waiting for an input from the user. In this scenario, ‘John Smith’ was entered and thereafter saved into the macro variable ‘name’, all these without having to go thru program code nor modifying the call each time.

PUTTING ALL TOGETHER

By combining the TRANWRD function to perform search and replacing character string function into %WINDOW user interface, it creates an application out of simple SAS code. The application starts out allowing allow user to input specific file information, enter the search and replace text string, define columns (variables), and then touch up with a few options to increase flexibility. The features are pretty self-explanatory, so that different users can interpret the interface in the same manner. Hence, the application is created with user friendliness and reusability in mind.

The program for this application is as follows:

Fig 5.
Program

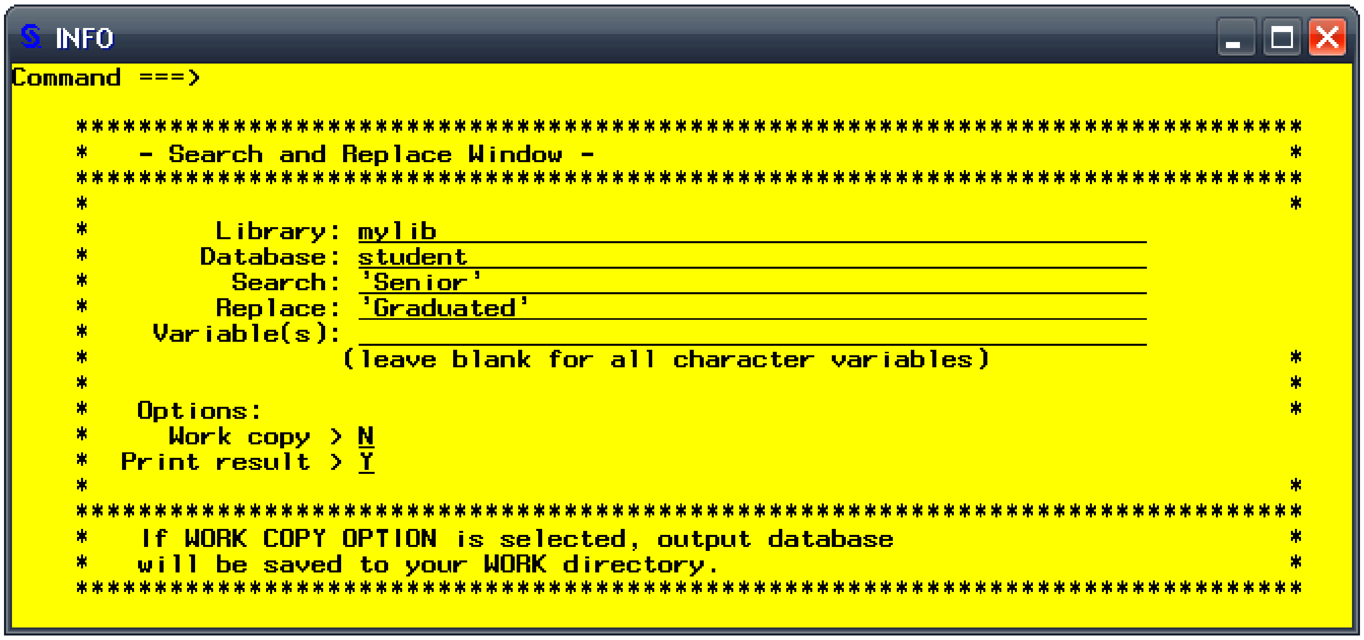
```
/*** Start of program ***/

/* Declaration */
%let libname=;
%let dbname=;
%let search='';
%let replace='';
```

```
%let columnlist=;
%let workcopy=;
%let printlist=;

/* GUI */
%macro inputgui;
  %window info color=yellow
     group=chooseit
     #2 @5  '***************************************************************************'
     #3 @5  '*   - Search and Replace Window -                                         *'
     #4 @5  '***************************************************************************'
     #5 @5  '*                                                                         *'
     #6 @5  '*        Library:'
     #7 @5  '*       Database:'
     #8 @5  '*         Search:'
     #9 @5  '*        Replace:'
     #10 @5 '*    Variable(s):'
     #11 @5 '*                (leave blank for all character variables)                *'
     #12 @5 '*                                                                         *'
     #13 @5 '*   Options:                                                              *'
     #14 @5 '*     Work copy >'
     #15 @5 '*  Print result >'
     #16 @5 '*                                                                         *'
     #17 @5 '***************************************************************************'
     #18 @5 '*   If WORK COPY OPTION is selected, output database                      *'
     #19 @5 '*   will be saved to your WORK directory.                                 *'
     #20 @5 '***************************************************************************'

     #6 @23  libname 50 attr=underline persist=no required=yes
     #7 @23  dbname 50 attr=underline persist=no required=yes
     #8 @23  search 50 attr=underline persist=no required=no
     #9 @23  replace 50 attr=underline persist=no required=no
     #10 @23 columnlist 50 attr=underline persist=no required=no
     #14 @23 workcopy 1 attr=underline persist=no required=no
     #15 @23 printlist 1 attr=underline persist=no required=no ;
  %display info.chooseit delete blank ;
%mend;
%inputgui

/* Options list */
%macro checklist;
  %global columnlist replacefile;
  %if &columnlist eq %then
    %let columnlist=_character_;
  %if &workcopy eq or &workcopy eq N or &workcopy eq n %then
    %let replacefile=&libname..&dbname;
  %else %let replacefile=work.&dbname;
%mend checklist;
%checklist

/* Search and Replace function */
data &replacefile ;
  set &libname..&dbname ;
  array varlist{*} &columnlist;
  do i = 1 to dim(varlist);

    varlist(i)=tranwrd(varlist(i), &search, &replace);
  end;
  drop i ;
run;

/* Options - print result */
%macro printlist;
%if &printlist ne and &printlist ne N and &printlist ne n %then
  %do;
    title "&dbname data set (AFTER)";
    proc print data=&replacefile;
```

```
      var &columnlist;
       run;
     %end;
   %mend printlist;
   %printlist

   /*** End of program ***/
```

Upon execution, figure 6 should be the interface displayed under SAS window. The interface provides an organized mean of allow user to enter input without going thru programming code. Texts entered in each individual field are passed on to a pre-defined macro variable.

Fig 6.
Window



```
Command ===>

    ******************************************************************************
    *    - Search and Replace Window -                                           *
    ******************************************************************************
    *                                                                            *
    *         Library: mylib
    *        Database: student
    *          Search: 'Senior'
    *         Replace: 'Graduated'
    *     Variable(s):
    *                 (leave blank for all character variables)                  *
    *                                                                            *
    *    Options:                                                                *
    *      Work copy > N
    *   Print result > Y
    *                                                                            *
    ******************************************************************************
    *    If WORK COPY OPTION is selected, output database                        *
    *    will be saved to your WORK directory.                                   *
    ******************************************************************************
```

To demonstrate, let's say a high school administrative office is getting ready to update the enrollment status of the school student information database. One of the task responsible to change the enrollment status of the graduated senior students from 'Senior' to 'Graduated' to reflect their status in the school database 'STUDENT', which happens to be SAS data set.

Fig 7.
STUDENT data set (BEFORE)

```
Obs    student    class

1      Adam       Freshman
2      Clark      Junior
3      Mandy      Senior
4      Smith      Sophomore
5      Travis     Senior
```

Fig 8.
STUDENT data set (AFTER)

```
Obs    student    class

1      Adam       Freshman
2      Clark      Junior
3      Mandy      Graduated
4      Smith      Sophomore
5      Travis     Graduated
```

Perhaps in this case, maybe the administrative person does not work SAS daily, which makes the graphical user interface to be very beneficial. Now, he or she would only need to enter the parameters within the GUI window as in regular database entry program, perform exactly what the user want the program to do. Hence, the task of updating the student database has become more effective and easily repeatable. A simple application has just been created.

CONCLUSION

Through the use of a simple combination of two simple SAS tools, TRANWRD and %WINDOW, we created a quick and easy means of searching and replacing text string in s SAS data set. By adding user defined variables, modifiable variables, and extra options, the program adds a layer of flexibility making it much more user friendly.

Putting all these features together, we have built an application within an application; a tool was created to perform specific tasks. At the same time, the application now enables users to perform the same task, with less coding. It is hoped that this paper introduces and inspires other ideas on developing general, simple applications that can be designed with creativity and user friendliness in the development process.

REFERENCES

Cody, Ron, 1997. "Having a Ball with Strings: SAS Character Functions." *Proceedings of the Twentieth-seventh Annual SAS Users Group International Conference,* San Diego, CA, 375-382.

SAS Institute Inc. (2000), *SAS® Language Reference: Dictionary, Version 8.* Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Thanks Mr. Bryan Beverly, BAE Systems, for guidance and support of this paper.

CONTACT INFORMATION

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

Edmond Cheng
U.S. Bureau of Labor Statistics
2 Massachusetts Ave., NE
Washington, DC 20212-0001
(202) 691-5458
cheng_e@bls.gov