PharmaSUG 2021 - Paper QT-069

Writing Out Your SAS® Program without Manually Writing it Out-Let Macros Do the Busy Work

Mindy Wang, Paradyme Management

ABSTRACT

Problem happened when we were asked to write out data to ASCII files using SAS datasets. The format is in Excel spread sheet, but there are hundreds of variables. It is tedious to write the program all by hand. By using macro variables, there is an easier way for SAS to write out the text string for you, rather than manually keying in the details. It only takes a few second to run the program and the program can be easily applied to other datasets, as long as we have the formats the Excel or in SAS metadata. With the same logic, you can also modify the code for other repetitive tasks. For the SAS users who are thinking about using macros, but are kind of intimidated to use macros, this will be the presentation to get you into the door of starting to use macros. Once you learn the concept, the possibility is endless. This presentation will make your life a lot easier. You will have less busy work and less headache.

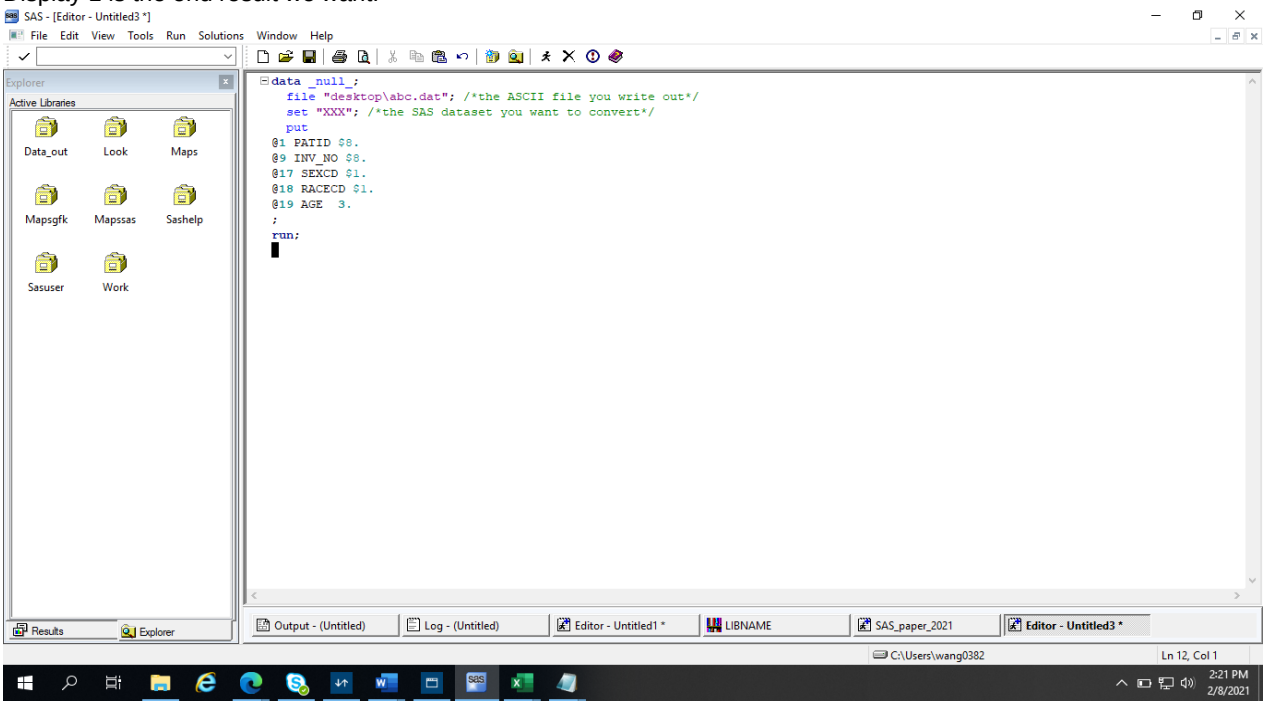
INTRODUCTION

This paper is intended for people who has some basic SAS experience and heard about SAS Mcaros, but feel intimidated to try SAS Macros yet. In clinical trials and many other research fields, repetitive statements happen very often. This paper illustrates how to deal with repetitive task without manually keying in the code. Instead, Macro variables are used. Using a simple macro program to make the task more efficient without a lot of copying and pasting. Furthermore, once you learn this program, you can improve the program and to make the program more adaptable to other projects as well. Using variable-dependent macros minimizes tedious typing, eliminate possible human errors, and ensure the accuracy of code.

THE END RESULT WE WANT

The end result we want is a piece of SAS code as following.

Display 1 is the end result we want.



Display 1. The End Result of What We Want

Following is the code to write out the ASCII file manually:

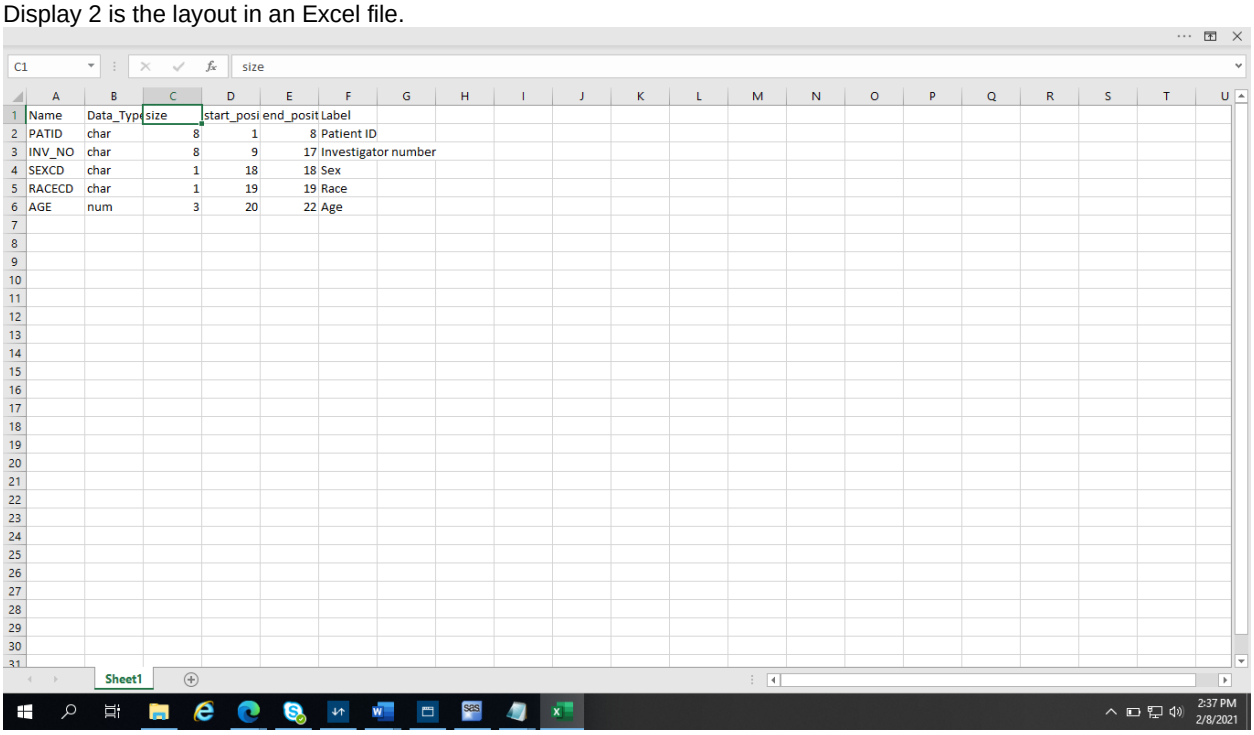
```
data _null_;
  file "desktop\abc.dat"; /*the ASCII file you write out*/
  set "XXX"; /*the SAS dataset you want to convert*/
  put
@1 PATID $8.
@9 INV_NO $8.
@17 SEXCD $1.
@18 RACECD $1.
@19 AGE  3.
;
run;
```

For illustration purpose, I use very few variables here. This may not look like much when the number of variables is small, but when the number of variables is in the hundreds or thousands, the task can be very tedious. Following will do the trick without manually writing the program out.

CONVERT THE LAYOUT FORM EXCEL TO SAS

The layout is in Excel. Following is the screen shot of the layout in the Excel file.

Display 2 is the layout in an Excel file.



Display 2. The Layout in An Excel File

First of all, we need to convert the layout from Excel to SAS.

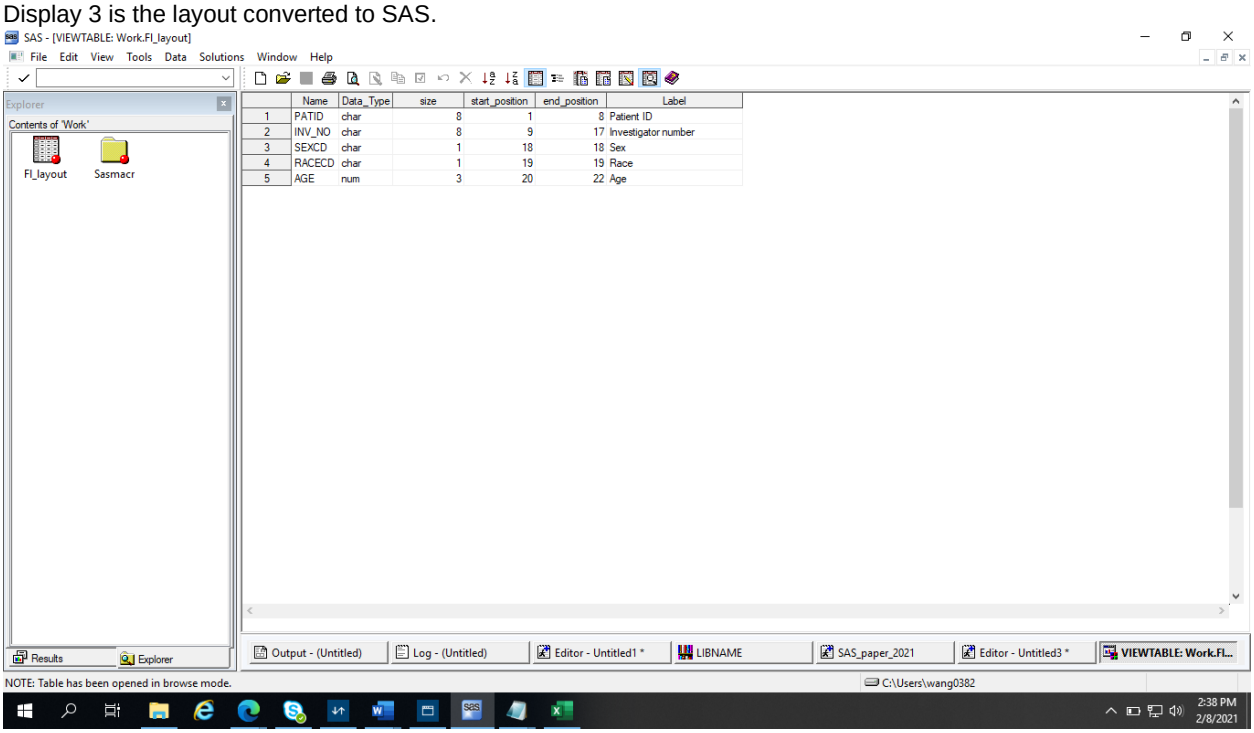
Following is the code to bring the layout from the Excel file to SAS:

```
/*convert the layout from Excel to SAS*/

proc import out=fl_layout
   datafile="XXX\file_layout.xlsx"
       dbms=xlsx replace;
       getnames=yes;
run;
```

Display 3 shows the layout in SAS.

Display 3 is the layout converted to SAS.



Display 3. The Layout Converted to SAS

SET UP MACRO VARIABLES

Then we set up all the macro variables that we will be using in the next step.

Following is the code to set up the macro variables to use in the next step:

```
/*Set up the macro variables to use in the next step*/
data _null_;
   set fl_layout(where=(name ne '' and Data_Type ne ' ' and size ne .))
   end=eof;
     n=put(_n_,3.); /*if the number of variables is in the hundreds*/
      /*if the number of variables is in the thousands, set to 4.*/
      /*need to set the number big enough to accommodate the number of
      variables*/

     if _n_=1 then bcounter=1;/*set the counter on the first record*/
     pre=' ';/*initiate the prefix to blank*/

/*if the Data_Type is character, set pre to $*/
     if Data_Type in ('CHAR','char') then pre='$' ;

     call symputx('var'||left(n), name);
     call symputx('start'||left(n), bcounter);
     call symputx('pre'||left(n), pre);
     call symputx('size'||left(n), size);
     bcounter+size;

/*get the number of variable at the end*/
     if eof then call symputx('n',n);
```

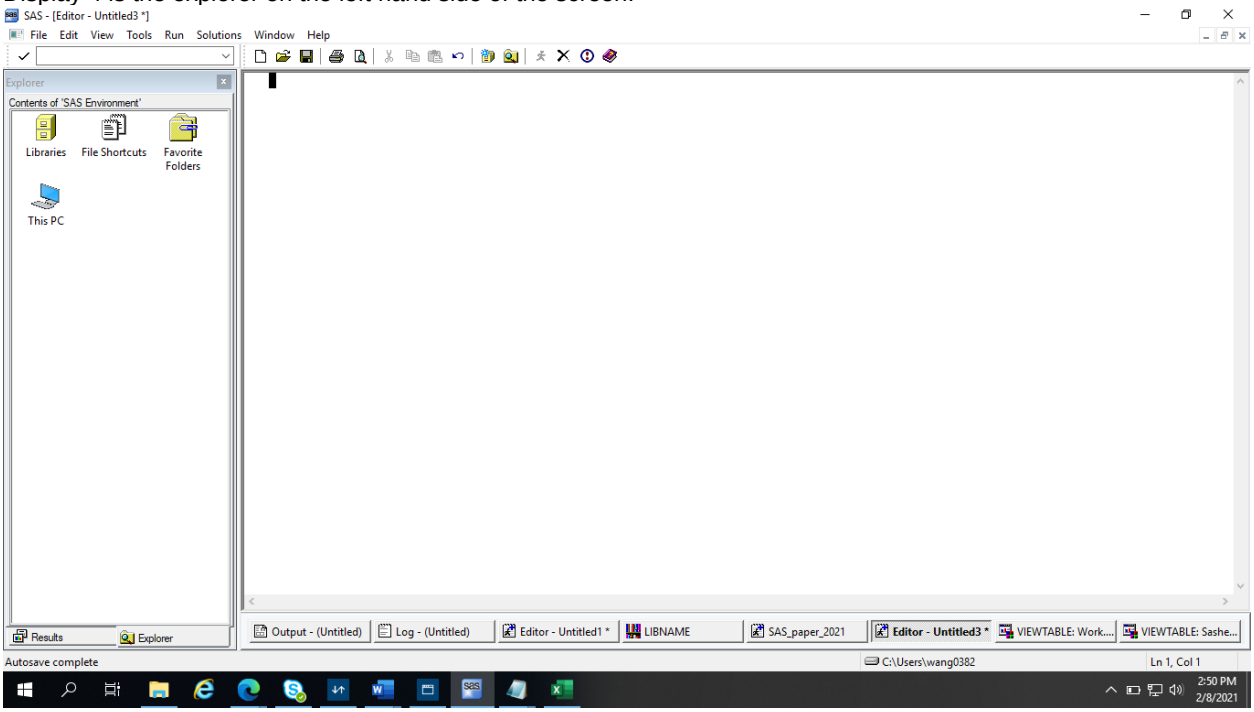
```
run;
```

Please note that in the above code, we first set up n=put(_n_,3.), since the number of variable are in the hundreds. If the number of variables is in the thousands then we would set it to n=put(_n_, 4.) to allow it hold 4-digit numbers. Then we set up the counter for the number of variables in the file "if _n_=1 then bcounter=1" and later we set "bcounter+size;" to calculate the number of variables in the dataset. At the end of file, we have the number of variables and store the macro variable n using call symputx statement (if eof then call symputx('n',n);).

LOOK AT SAS META DATA

After we set up the macro variables, SAS automatically stored the macros variables created in Sashelp.Vmacro. In the Explorer on the left hand side of the screen, we click on Libraries.

Display 4 is the explorer on the left hand side of the screen.

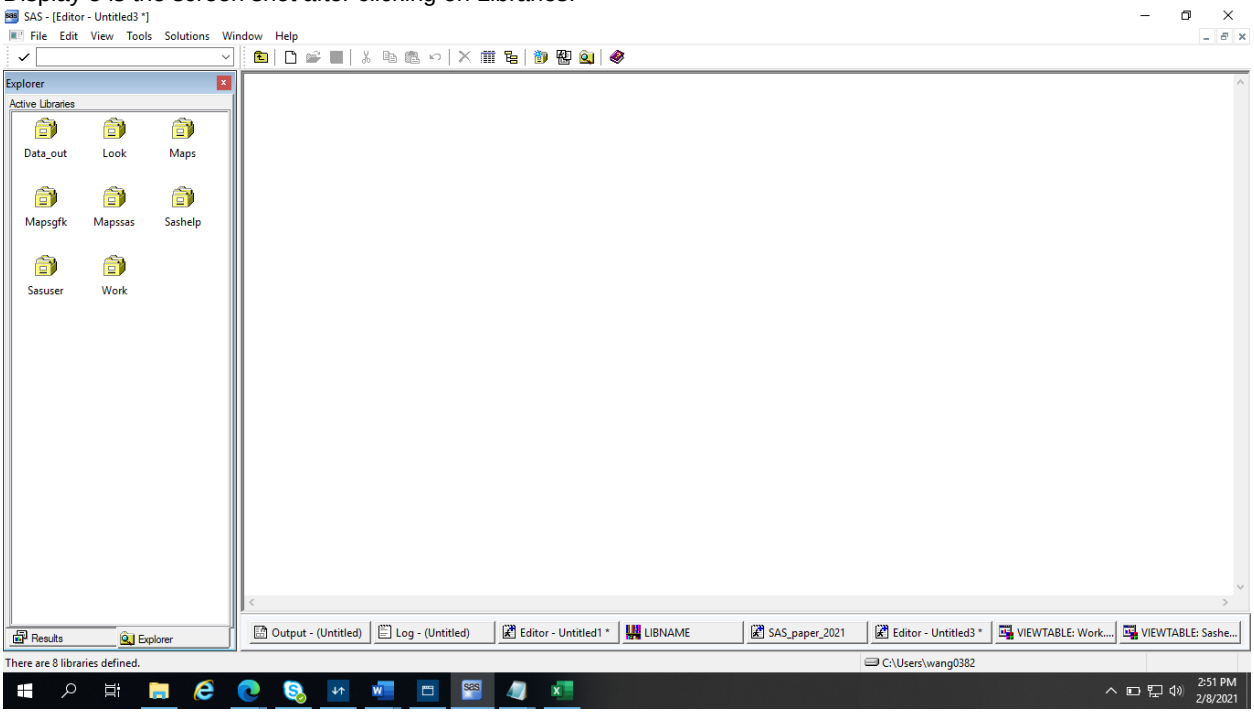


Display 4. The Screen Shot of the Explorer on the Left Hand Side

Then we click on Libraries.

Display 5 shows the screen shot after we click in Libraries.

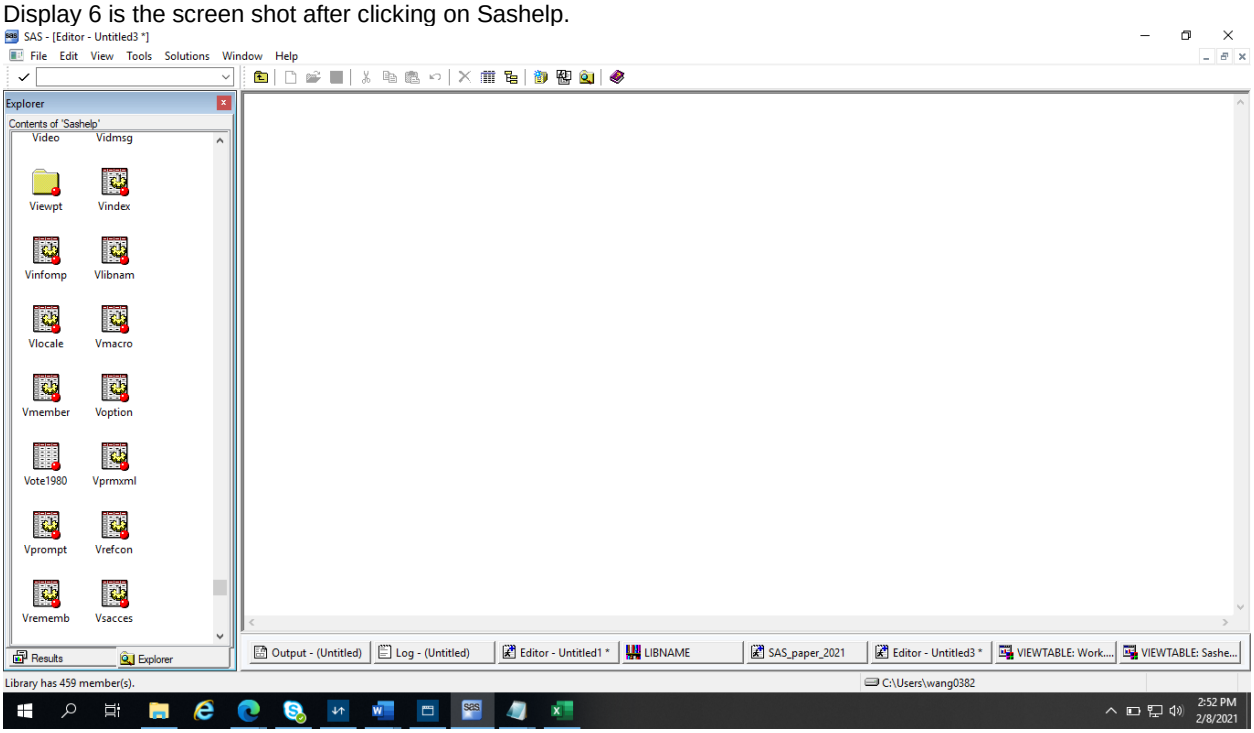
Display 5 is the screen shot after clicking on Libraries.



Display 5. The Screen Shot of the Explorer after Clicking on Libraries

Then click on Sashelp, and then scroll down to find Vmacro and click on Vmacro.

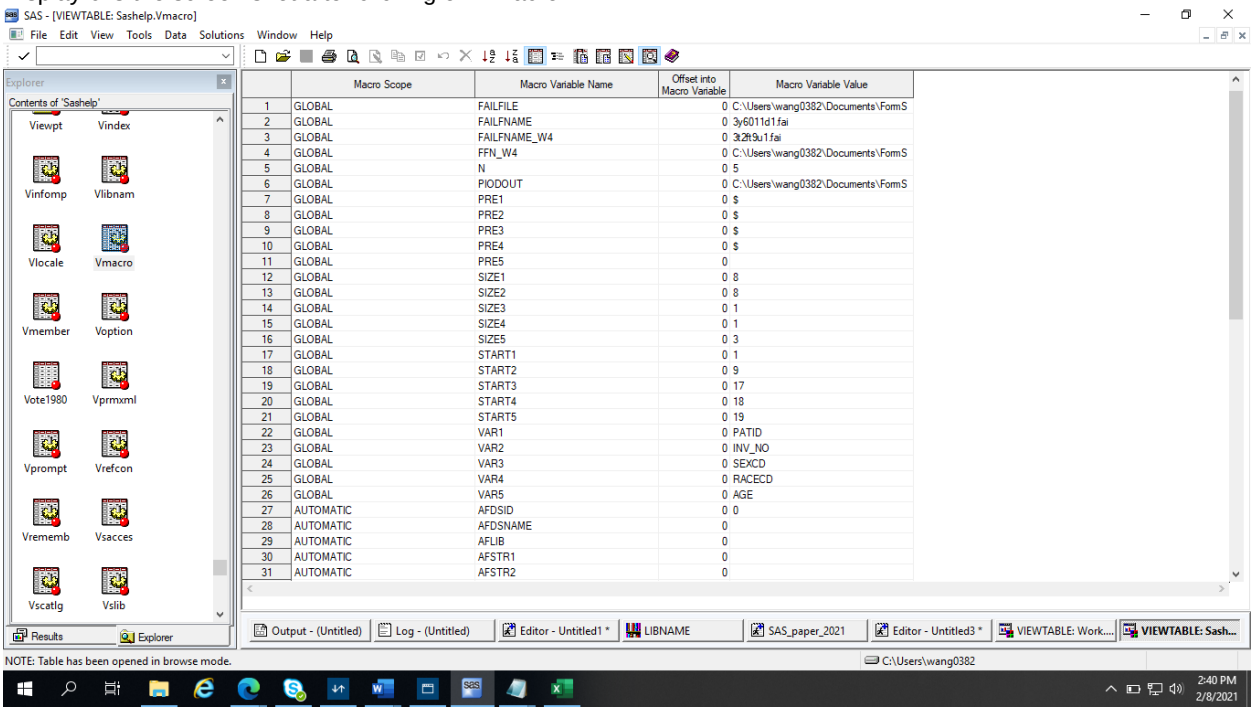
Display 6 is the screen shot after clicking on Sashelp.



Display 5. The Screen Shot of the Explorer after Clicking on Sashelp

Following is a view from Sashelp.Vmacro on the stored macro variables from the above data step.

Display 6 is the screen shot after clicking on Vmacro.



Display 6. The Screen Shot of the Explorer after Clicking on Vmacro

We can see from Display 6 that Macro variable VAR1 is PATID, VAR2 is INV_NO, VAR4 is SEXCE, and VAR5 is AGE. We can also see that Macro variable START1 is 1, START2 is 9, START3 is 17, START4 is 18, and START5 is 19, which represent the starting positions of the variables respectively in the ASCII file that we want to write out. We also can see that Macro variable SIZE1 is 8, SIZE2 is 8, SIZE3 is 1, SIZE4 is 1, and SIZE5 is 3, which represent the sizes of the variables respectively that we want to write out. All the macro variables are stored in Vmacro as expected.

WRITE OUT THE TEXT STRING TO NOTEPAD

The next step, we can write out the text string using the macro program below. We can exam the text string to see if it is what we expected.

Following is the code to write out the text string to notepad:

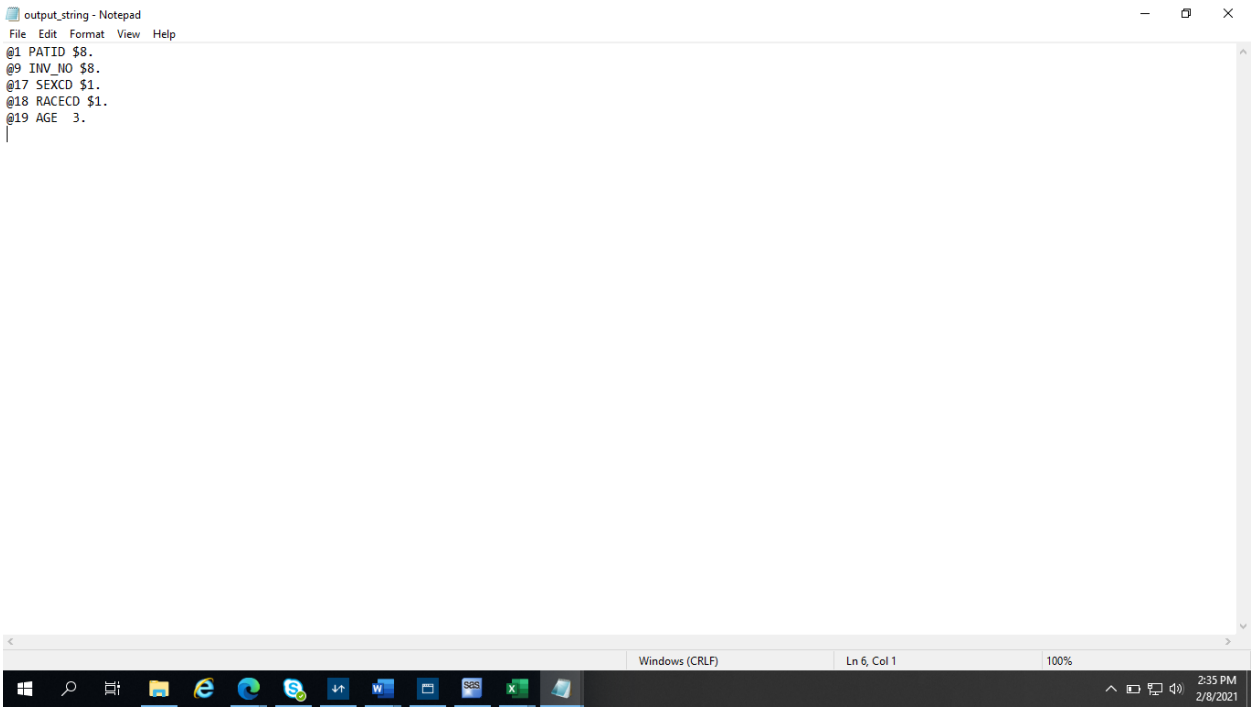
```
   /*write the string using macro*/
   %macro string;
      %do i=1 %to &n;
         stri="@&&start&i."||' '||"&&var&i. "||"&&pre&i"||"&&size&i."||"."

   ;
         put stri;
      %end;
   %mend string;

   /*The following is optional if you want to take a look the string that has
been written out by the macro*/
   data _null_;
    length stri $250;
      file 'desktop\output_string.txt';
      %string
   run;
```

The macro program string comprises the text string by concatenating the starting position, variable name, prefix(if it is character, there is $ as prefix), and variable size. Then the data _null_ step writes out the text string to notepad, so we can take a look to make sure its accuracy. Following is the screen shot when we open the text file.

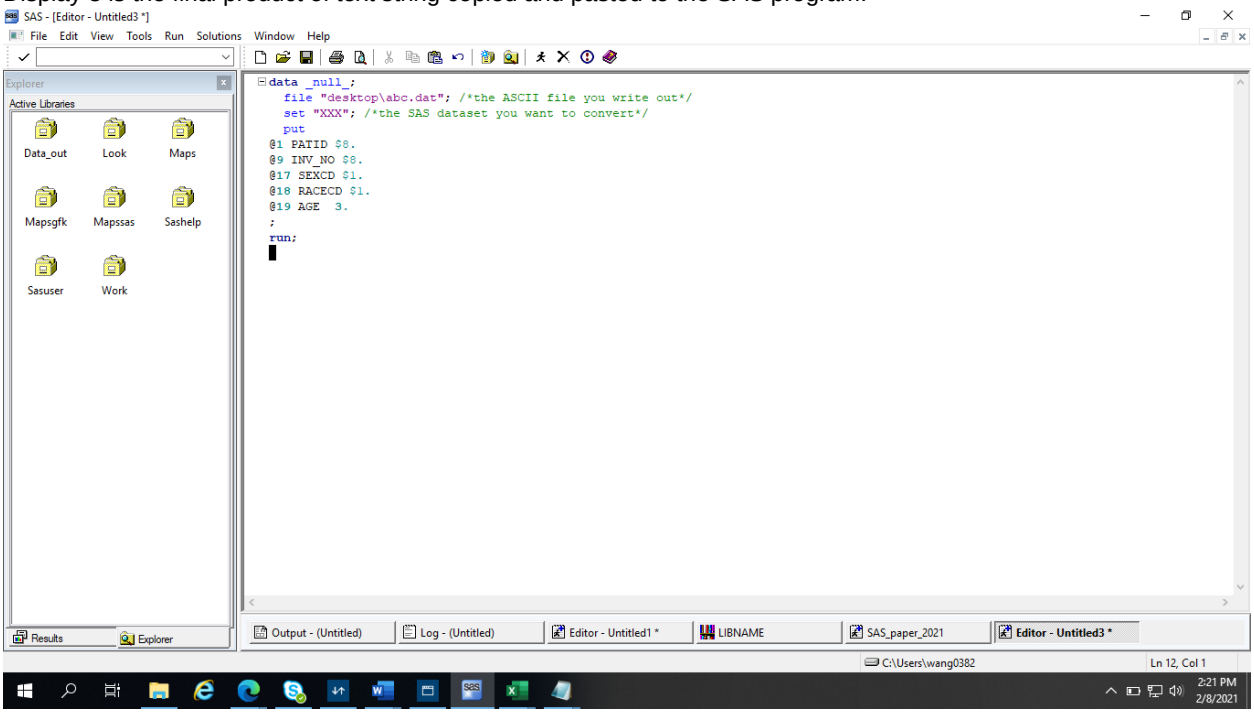
Display 7 is the screen shot of the text string written out to notepad.



Display 7. The Screen Shot of the Text String Written Out to Notepad

After we look at the string and make sure everything is as expected, we can copy the text string and insert it to SAS program. Now we just wrote the SAS code, without having to manually key in the code. When the variables are many, this approach really saves a lot of time as well as possible human errors introduced to the SAS program by keying in the statements manually.

Display 8 is the final product of text string copied and pasted to the SAS program.



Display 8. The Final Product of Text String Copied and Pasted to the SAS Program

For beginners of macro, it is always a good practice to look at the text string to make sure that they are as expected. Once you are familiar with macro variables and macro programs, the data _null_ part of writing out the text string to notepad can be used just as a debugging tool, or can be eliminated all together.

CALL THE MACRO DIRECTLY IN THE DATA STEP

Another alternative to writing the text string out, and then copy and paste the text string to the SAS program, we can just call the macro directly in the data step.

Following is the code to write out the text string to without writing out to notepad:

```
   /*If you are comfortable not see the string first, the following is to
write the string using macro in the data step*/
   data _null_;
     file "desktop\abc.dat"; /*the ASCII file you write out*/
     set "XXX"; /*the SAS dataset you want to convert*/
     put
     %string
     ;
   run;
```

Please note in the above code, we invoke %string, therefore the text string that generated by the string macro program will be inserted here, just as we copy and paste from notepad without needing to copy and paste. The above code will do the same just as you see in Display 8.

CONCLUSION

Once you have this one program done, applying this technique to other situations and other tasks is easy. I hope this paper encourage you to start using Macros to reduce doing busy work manually. This program can be modified to do other repetitive tasks. For example, if you are asked to do something on

different clinical sites, you can modify on this code and assign macro variables on the sites (SITE1, SITE2, SITE3, etc.) instead of assigning macro variables on variable names. Not only this small program has minimized typing, but it is also adaptable to other projects. Oftentimes, we have time constraint to get thing done at very tight timeline, copy and paste method is a sure thing we can get it done. However, we should always strive to rewrite the program to make it easier to adapt to new situation afterward. Even though at first it takes time to develop; In the long run, the variable dependent macro really saves time.

ACKNOWLEDGMENTS

Many thanks to Kevin Chung for introducing me to variable-dependent macro.

RECOMMENDED READING

- *SAS® Macro Programming Made Easy by Michele Burlew*

CONTACT INFORMATION

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

Mindy Wang
240-760-9988
ymindywang@yahoo.com