Paper 1436-2017

Writing SAS® Code on the Fly Using SAS Code as Character Variables

Peter Timusk, Statistics Canada

ABSTRACT

A SAS® program with the extension .SAS is simply a text file. This fact opens the door to many powerful results. You can read a typical SAS program into a SAS data set as a text file with a character variable, with one line of the program being one record in the data set. The program's code can be changed, and a new program can be written as a simple text file with a .SAS extension. This presentation shows an example of dynamically editing SAS code on the fly and generating statistics about SAS programs.

INTRODUCTION

The first step is importing a SAS program into a SAS data set consisting of only one character variable. A brief example is given of manipulating this character variable. We write out the resulting SAS dataset and it produces a new SAS program. At any step, something different could be accomplished depending on the application.

READ IN A SAS PROGRAM AS A TEXT FILE

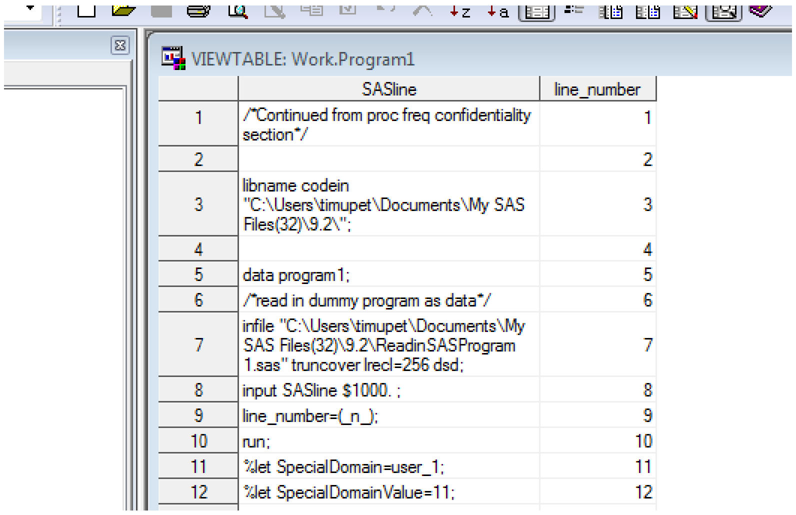
This code below reads in a SAS program which is just a text file. It stores the lines of SAS program code as values in a character variable 'SASline' in the dataset 'program1'. We add a simple line number variable using the (_n_) macro variable so that we can reference lines in the program if needed.

Here is the code:

```
DATA program1;
/*read in dummy program as data*/
            infile "C:\Users\ptimusk\Documents\My SAS
            Files(32)\9.2\Testreaddatastep.sas" truncover lrecl=256 dsd;
            input SASline $256. ;
            line_number=(_n_);
run;
```

RESULTS OF PROC IMPORT

Display 1 below is a view of the dataset that results when a SAS program is read in as a text file.



	SASline	line_number
1	/*Continued from proc freq confidentiality section*/	1
2		2
3	libname codein "C:\Users\timupet\Documents\My SAS Files(32)\9.2\";	3
4		4
5	data program1;	5
6	/*read in dummy program as data*/	6
7	infile "C:\Users\timupet\Documents\My SAS Files(32)\9.2\ReadinSASProgram 1.sas" truncover lrecl=256 dsd;	7
8	input SASline $1000. ;	8
9	line_number=(_n_);	9
10	run;	10
11	%let SpecialDomain=user_1;	11
12	%let SpecialDomainValue=11;	12

Display 1. SAS dataset containing SAS program code as values in a character variable.

USE THE DATASET OF SAS CODE LINES TO COMPLETE VARIOUS TASKS

COUNT HOW MANY DATA STEPS OR PROCEDURE STEPS ARE USED IN PROGRAM

With the SAS program now accessible as a character variable, we can search for every line that contains the word ‘DATA’ for example and count the size of this file. This may, depending on the programming style or keyword use in your SAS programs accurately indicate the number of new datasets created. The number of datasets is possibly one statistic about this program. One can, of course, be careful not to count null datasets.

This code creates a dataset with only the program lines that declare the beginning of data steps:

```
   DATA program1_datasets_list;
   Set program1;
   /*select only those lines that start data steps*/
      where upcase(SASline) contains 'DATA ' and  upcase(SASline) not
      contains '_NULL_';
run;
```

LIST ALL DATSETS CREATED IN THE PROGRAM

With this dataset listing all SAS code lines that start data steps, one can then parse out the dataset names. Here knowledge of the style of the ‘DATA’ line code will help find the correct names.

Here we assume no options on the data step ‘DATA’ line and assume the name of the dataset occurs between the end of the word ‘DATA’ and the first semicolon:

```
   DATA program1_datasets_names;
   Set program1_datasets_list;
      length datasetnames $32.;
      datasetnames=compress(
                                substr(SASline,(
                                      prxmatch('/DATA/',upcase(SASline))
                                +4))
                             ,';');
   run;
```

WRITE A NEW SAS PROGRAM

CREATE A NEW PROGRAM BASED ON SOMETHING IN THE OLD PROGRAM: WRITE A NEW PROGRAM TO DO SOMETHING NEW

This is a section of code that writes a character variable to a new program. Here we use parts of the old program; namely, the header of the program. We have rewritten code contained in variables in the colvars dataset which is completed in the first and third data step. This colvars dataset contains the body of the data step we are rewriting. Some macro variables are not fully shown. This new code is all placed in the new ‘SASline’ variable. We add a ‘DATA’ statement and ‘SET’ statement from the header file and a ‘RUN’ statement. We put all this to a file as text and the text file is given a .SAS extension. Since a SAS program is just a text file, this means we are writing a useable SAS program.

Here are these series of steps:

```
   DATA colvars;
   Set newfile1;
   format ifstatement $3.;
```

```
ifstatement="IF ";

Do i = 1 to &colnum.;
   if i = 1 then ivar1=" i"||
   left(trim(substr(surveyvar1,2,6)));
format conditionstatement1 $9.;
   if i= 1 then conditionstatement1 =" NE '' ";
format orstatement1 $4.;
   if i >=2 and surveyvar2 ne ""  then orstatement1 =" or "; else
   orstatement1 ="";
   if i = 2 and surveyvarnum >= 2 then ivar2="
   i"||left(trim(substr(surveyvar2,2,6)));
format conditionstatement2 $9.;
   if i= 2 and surveyvar2 ne "" then conditionstatement2 =" NE '' ";else
   conditionstatement2 ="";
end;
   thenstatement = " then icol_"||
   left(trim(column1))||
   " =1; else     icol_"||
   left(trim(column1))||
   " = 0;";
keep ifstatement ivar1--ivar&colnum.
   conditionstatement1 conditionstatement2
   orstatement1
   thenstatement
   surveyvarnum  column1;
   if surveyvarnum = . then delete;
   if column1 ne '' then output colvars;
run;
%let beginfromsetdataset2linenumber=%eval(&BeginDataStep2linenumber.+1);

DATA headerandbegindatastep1;
Set program1 (obs=&beginfromsetdataset2linenumber.);
 drop line_number;
run;

DATA datastepcontents;
Set colvars;
format SASline $256.;
   SASline=right(ifstatement)||
   ""||
   trim(left(ivar1))||
   ""||
   conditionstatement1||
   orstatement1||
   left(trim(ivar2))||
   conditionstatement2||
   thenstatement;
keep sasline;
run;

DATA endnewdatastep;
format sasline $256.;
sasline=" run;";
run;
```

```
DATA newprogram;
Set headerandbegindatastep1 datastepcontents endnewdatastep;
run;

   filename newprog "C:\Users\timupet\Documents\My SAS
   Files(32)\9.2\newprogram.sas" LRECL=256;

DATA _null_;
Set newprogram;
   file newprog;
   put sasline 1-256  ;
run;
```

CONCLUSION

This Quick Tip paper is meant to show basic techniques for manipulating SAS code. Similar results could be achieved with macro variables for some cases. The author has also written SAS code on the fly using macro variables to insert SAS code as text. The present example shows a way to allow one's character data handling skills to be applied to write SAS programs.

REFERENCES

Lal, Rajesh. Ranga, Raghavender. 2012. "How Readable and Comprehensible Is a SAS® Program? A Programmatic Approach to Getting an Insight into a Program." *Proceedings of the SAS Global 2012 Conference*, Orlando, FL. SAS Institute. Available at http://support.sas.com/resources/papers/proceedings12/001-2012.pdf

ACKNOWLEDGMENTS

The author would like to thank his employer for the support in writing this Quick Tip and the global SAS user community for the environment where it is possible to express these ideas and concepts.

RECOMMENDED READING

- *The Little SAS® Book*

CONTACT INFORMATION

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

Peter Timusk
Statistics Canada
peter.timusk@canada.ca