Paper 353-2013

SASY Codes for Lazy People

Prashanthi Selvakumar

ABSTRACT

"I choose a lazy person to do a hard job, because a lazy person will find an easy way to do it." - Bill Gates. Everyone wants to save time. While hard work is useful, smart work is a pre requisite. Are you tired of typing codes? Read this paper, it gives you the ways to shorten your codes. The topics discussed in this paper include, array, do loops, macros, functions. It also discusses the procedures and data steps where macros can save your time. The other techniques like, combining the macros while creating a html, pdf, rtf output, to produce professional reports. The possible ways of saving time in programming are addressed in this paper.

INTRODUCTION

Any programming language will have more than one way to write a code. One way, this is the routine, which most of the SAS® users follow and the other, which is less used by the programmers. Now, we shall focus on the road that is not taken. To begin with, arrays and functions are used to reduce the time that we take for coding. This paper will focus on simple techniques that can save a lot of time in your busy schedule. This should be a good start for the beginners and intermediate SAS users. This paper is based upon my experience as a student, where I started using SAS for academic purposes, and later grew into a lazy programmer. This has valuable short cuts from several SAS books and several publications put together. This will compare the usual methods with the smart methods.

PROC FREQ PROCEDURE

This is one of the simple examples that can be used in your daily programming. Let us start with the PROC FREQ procedure. Suppose your dataset contains lot of character variables and you are not sure of the exact names of the character variables, you can still write the programs to create the frequency tables. This is the routine method that beginners use, when they start to write the SAS programs.

METHOD 1:

```
proc freq data = test;
 table gender_pre marital_pre education_pre race_pre role_pre       organization_pre
insurance_pre health_rating_pre/ nocum  nopercent;
run;
```

Method 2 gives the better way for producing the frequency tables for all the character variables in the dataset "test".

METHOD 2:

```
proc freq data = test;
 table _character_/ nocum nopercent;
run;
```

Method 3 is another form of programming for selected character variables, in this case the frequency tables are produced for the selected variables.

METHOD 3:

```
proc freq data = test;
 table gender_pre -- race_pre;
run;
```

ARRAYS AN IMPORTANT TOOL IN SAS

Arrays are a valuable tool in SAS. They can be one dimensional to multi dimensional arrays. They are used in a various programs including. In this example, we wanted to rename the variables that were initially assigned as default MS Excel. Say for example, I wanted to rename the variable f25 to tech2. To code each and every variable from f25 – f46, there are several ways to code it. Here are the two common methods that you can use to recode the variables, Method2 is more effective program compared to Method1.

Method 1 is used to rename all the variables in the dataset named “texas” starting from f25 to Tech25, because all were open- ended responses for a similar question. Each and every variable is renamed separately.

METHOD 1:

```
data texas;
   set texas_year2;

  tech2= f25;
  tech3= f26;
  tech4= f27;
  tech5= f28;
  tech6=f29;
  tech7=f30;
  tech8= f31;
  tech9= f32;
  tech10= f33;
  tech11= f34;
  tech12= f35;
  tech13= f36;
  tech14= f37;
  tech15= f38;
  tech16= f39;
  tech17= f40;
  tech18= f41;
  tech19= f42;
  tech20= f43;
  tech21= f44;
  tech22= f45;
  tech23= f46;
run;
```

Method 2 is used to convert all the twently variables into new variables. However, this time I used the arrays named “old” and “new” to rename all the twenty variables.

METHOD 2:

```
data texas1;
   set texas_year2;

   array old (21)$ 70 f26 - f46;

   array new (21)$ 70 tech3 - tech23;

   do i = 1 to 21;
   new(i) = old(i);
  end;run;
```

Thus within a few data lines statement we can rename all the variables in the dataset.

ARRAYS USED IN COMBINATION WITH FUNCTIONS

There are a variety of SAS functions that are used for several purposes. The most important SAS functions include, the ANYDIGIT, ANYALPHA, ANYFIRST, CATX, SUBSTR and several others. When used in combination with the arrays they are one of the most efficient tools. Here are the two examples Method 1 without using function and the other after using function.

If you want to recode the values, in the following example, “1 – Not Started” to “Not Started” below are the two different methods that are used they are given below:

In the Method1, array named recode is used to recode all the 6 variables, but however, each and every value has to be typed every time. Also note that if then else is used instead of the else if statement. Method1 is one of the less efficient ways of writing the SAS program.

METHOD 1:

```
data texas11;
  set texas1;
  array recode (2) $ 70 Prog_tob prog_act;
  do j= 1 to 2;
   if recode(j)= "1 - Not Started" then recode(j)= "Not started";
   if recode(j) = "2 - Delayed" then recode(j)= "Delayed";
   if recode (j) = "3 - Just Started" then recode(j) = "Just started";
   if recode (j) = "4 - On Target" then recode (j) = "On Target";
   if recode (j) = "5 - Completed" then recode(j) = "Completed";
   if recode (j) = "1 - Poor" then recode(j) = "Poor";
   if recode (j) = "2 - Below Average" then recode(j) = "Below Average";
   if recode (j) = "3 - Average" then recode(j) = "Average";
   if recode (j) = "4 - Above Average" then recode(J) = "Above Average";
   if recode (j) = "5 - Excellent" then recode(j) = "Excellent";
end;
run;
```

In this method 2, I used the array “recode” that includes all the 6 variables. In this method, I used the SUBSTR function to recode the variables.

METHOD 2:

```
data texas_new;
 set texas1;
 array recode (6) $ 70 Prog_tob -- rate_com;
 do j= 1 to 6;
 recode (j)= substr (recode (j),1,20) ;
end;run;
```

MACROS IN SAS

In SAS, if there are steps that have to be repeated for several datasets or several variables again and again, the efficient way of performing this is through MACROS. They can be used either in the DATA step or the PROC step. However, the invocation of the macro may take longer time as every time you have to type each and every value of all the variables that are used in the Macros. Method 1 and Method 2 are used to compare with each other.

In the method 1 we used the routine methods to invoke the macro called “test”. Here the macro variables named “dataset” and “var” are used. During the invocation of the macro, we use the routine method where we have to type each and every value for the variable.

METHOD 1:

```
%macro test (dataset, var);
 proc freq data = &dataset;
 tables &var;
 run;
%mend test;
%test (texas_new, Prog_tob);
%test (texas_new, Prog_act);
%test (texas_new, Prog_com);
%test (texas_new, rate_tob);
%test (texas_new, rate_act);
%test (texas_new, rate_com);
%test (texas1, Prog_tob);
%test (texas1, Prog_act);
%test (texas1, Prog_com);
%test (texas1, rate_tob);
%test (texas1, rate_act);
%test (texas1, rate_com);
```

In the method 2 all the steps in Macros are the similar to Method 2, except the invocation, where we had used just the first and the last variables separated by “—“ sign to invoke the macros for all the 6 variables, This could save you a lot of time in typing the variable names.

METHOD 2:

```
%macro test (dataset,var);
 proc freq data = &dataset;
 tables &var;
 run;
%mend test;

%test (texas_new, Prog_tob -- rate_com);
%test (texas1, Prog_tob -- rate_com);
```

CONCLUSION

Thus, you can always find smart ways for programming. There are several tools where we can search for this programming they include the SAS forum papers and support sas.com. However for beginners it is difficult to start with faster and efficient programming. With these resources and the tools like do loops, arrays and macros, even beginners can make their programs small and efficient.

REFERENCES

- Cody, Ronald. Feb 4 2013. SAS Functions by Example, Second Edition .47, 149. Cary, North Carolina : SAS Institute
- Cody, Ronald. Feb 4 2013. Cody’s Data Cleaning techniques using SAS, Second Edition .47, 149. Cary, North Carolina : SAS Institute
- SAS Support. SAS sample 95729: “Using Arrays in SAS programming.” Feb 4 2013. Available at http://support.sas.com/resources/papers/97529_Using_Arrays_in_SAS_Programming.pdf.

RECOMMENDED READING

- *SAS® Functions by Example*

CONTACT INFORMATION

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

Name: Prashanthi Selvakumar
Enterprise: University of North Texas Health Science Center
Address: 3500 Camp Bowie Blvd,
EAD 706
City, State ZIP: Fort Worth, Texas, 76107
Work Phone: (817) 617 1023
Fax:
E-mail: Prashanthi.Selvakumar@live.unthsc.edu