PharmaSUG 2015 - Paper QT27

The concept of "Dynamic" SAS programming

Sergey Sian, Quintiles, Cambridge, MA

ABSTRACT

The situation of writing SAS code which generates SAS programs, and these generated programs are then run to produce output is not new. However, this is sometimes a necessary and certainly useful tool to use.

The concept of "Dynamic" SAS programming will be discussed in this paper, along with why it is useful and how to do it. This will be augmented with an example of being a series of Adverse Event Tables. Along the way there will also be some discussion on the SAS options available to record what has been run and how it ran.

INTRODUCTION

"Dynamic" SAS Programming is the situation when your SAS program automatically generates SAS code which is included and executed by the same program later on. As usual, the best way to describe it is to provide an example. Say you need to report Preferred Term (AEPT) of all serious adverse events (SAE) patients experienced.

	Placebo		Treated		Total	
	1 Year n (%)[a]	Cumulative n (%)[b]	1 Year n (%)[a]	Cumulative n (%)[b]	1 Year n (%)[a]	Cumulative n (%)[b]
Patients with >= 1 SAE[c]	0 (0.0%)	0 (0.0%)	13 (3.4%)	21 (4.1%)	13 (2.2%)	21 (2.9%)
Angina pectoris	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (0.2%)	0 (0.0%)	1 (0.1%)
Arrhythmia	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (0.2%)	0 (0.0%)	1 (0.1%)
Arthralgia	0 (0.0%)	0 (0.0%)	1 (0.3%)	1 (0.2%)	1 (0.2%)	1 (0.1%)
Cardiac failure congestive	0 (0.0%)	0 (0.0%)	1 (0.3%)	1 (0.2%)	1 (0.2%)	1 (0.1%)

...

Let's also assume you have an SAE dataset which has all Preferred Terms reported. Since the above table is on "patient" level, and the same patient might report many Preferred Terms, you need to define "Yes-No" flag (for example, 'Angina_pectoris' flag) for each Preferred Term per patient like this:

If AEPT = "Angina pectoris" then Angina_pectoris = 1; ← 1 = "Yes"
Else Angina_pectoris = 2; ← 2 = "No"

Now imagine that there are dozens of Preferred Terms reported, and you have to type those 2 lines of code for every unique Preferred Term. It's a time consuming and error prone process. Can such code be generated automatically and included and executed by your program later on? Yes, with a few steps.

STEP 1: Identify unique Preferred Term (AEPT) and create the name of the flag for each of them

```
proc sql noprint;
          create table unique_sae_bool1 as
          select distinct AEPT
          from ads.sae ;
quit;

data unique_sae_bool(drop = scan_sae1 scan_sae2) ;
 length scan_sae1 scan_sae2 $15 scan_sae $25 counter $8 counter_num 8 ;
   retain counter_num ;
   set unique_sae_bool1 ;

 if _N_ = 1 then counter_num = 1 ;
 else counter_num+1 ;
 counter = put(counter_num,8.) ;
```

```
 counter = left(counter) ;
 scan_sae1 = scan(AEPT,1,' ') ;
 scan_sae2 = scan(AEPT,2,' ') ;
 scan_sae = left(trim(compress(scan_sae1,'-')))||'_'||left(trim(compress(scan_sae2,'-
')));
run ;

proc sort data = unique_sae_bool ;
   by AEPT ;
run ;
```

The “unique_sae_bool” SAS dataset has the full Preferred Term name in AEPT field (like ‘Angina pectoris’ or ‘Arrhythmia’) and respective flag in SCAN_SAE field (like ‘Angina_pectoris’ or ‘Arrhythmia_’).

STEP 2: Define all unique AEPT and respective flags as macro variables (through %let statement).

```
data MAKE_code_for_let_sae(keep = string_let_sae)
     MAKE_code_for_let_sae_name(keep = string_let_sae_name)
;
 length string_let_sae $50 string_let_sae_name $100 ;
   set unique_sae_bool ; by AEPT ;

if 1=1 then do ;
   string_let_sae = "%"||"let"||' '||'sae'||trim(left(counter))||' =
'||trim(left(scan_sae))||' ;' ;
   output MAKE_code_for_let_sae ;
end ;

if 2=2 then do ;
   string_let_sae_name = "%"||"let"||' '||'sae_name'||trim(left(counter))||' =
'||trim(left(AEPT))||' ;' ;
   output MAKE_code_for_let_sae_name ;
end ; run ;
```

The “Make_code_for_let_sae_name” SAS dataset has just one field ‘string_let_sae_name’ with values like “%let sae_name1 = Angina pectoris ;” or “%let sae_name2 = Arrhythmia ;”. The “MAKE_code_for_let_sae” SAS dataset has just one field ‘string_let_sae’ with values like “%let sae1 = Angina_pectoris ;” or “%let sae2 = Arrhythmia_ ;”.

All that we have to do now is “convert” those SAS datasets into a SAS code and include generated SAS code into your program.

```
%macro generate_sas(sas_pgr_name,field) ;
filename FLAT "&project.\Programs\macros\&sas_pgr_name..sas"
         lrecl = 150 ;

data _null_ ;
   set &sas_pgr_name. ;
file FLAT ;
put
@1   &field. ;
run;
%mend generate_sas ;
%generate_sas(Make_code_for_let_sae,string_let_sae) ;
%generate_sas(MAKE_code_for_let_sae_name,string_let_sae_name) ;
```

Include and execute generated SAS code:

```
%let mac_lib = &project\Programs\macros ;
%include "&mac_lib.\Make_code_for_let_sae.sas" ;
%include "&mac_lib.\MAKE_code_for_let_sae_name.sas" ;
```

STEP 3: Generate "if-then" SAS code (highlighted in yellow above) and include and execute it by your program

```
data for_if_then_statement(keep = string_for_if_then) ;
 length string_for_if_then $150 ;
   set unique_sae_bool ;
   by AEPT ;

   string_for_if_then = 'if AEPT = "&sae_name'||trim(left(counter))||'."
                        then &sae'||trim(left(counter))||'. = 1 ;
                        else &sae'||trim(left(counter))||'. = 2 ; ' ;
run ;
```

The "for_if_then_statement" SAS dataset has just one field 'string_for_if_then' with values like
'if AEPT = "&sae_name1." then &sae1. = 1 ; else &sae1. = 2 ;' or
'if AEPT = "&sae_name2." then &sae2. = 1 ; else &sae2. = 2 ;'.

```
%generate_sas(for_if_then_statement, string_for_if_then) ;
```

Include and execute generated SAS code:

```
%include "&mac_lib.\for_if_then_statement.sas" ;
```

CONCLUSION

The above code shows how to "dynamically" generate and include into SAS program the "if_then" statement. A similar approach can be used to generate "attributes" code such as:

```
attrib
Angina_pectoris length = 8 format = dyesno. label = 'Angina pectoris '
;
attrib
Arrhythmia_ length = 8 format = dyesno. label = 'Arrhythmia '
;
…
```

Or "format" code, like:

```
value linef
1 = 'Patients with >= 1 SAE~{super c}'
2 = '     Angina pectoris'
3 = '     Arrhythmia'
…
;
```

The described approach will take care of 3 or 300 reported SAEs without error prone typing, and time consuming and annoying "copy-and-paste" procedure.

ACKNOWLEDGMENTS

A special thank you to David Franklin, Manager statistical programming, who has provided valuable suggestions and inputs to my paper. I'm also grateful to Greg Tebbenkamp, Director of statistical programming, whose support and encouragement has helped me to write this paper.

CONTACT INFORMATION

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

Name: Sergey Sian
Enterprise: Quintiles
Address: 201 Broadway
City, State ZIP: Cambridge, MA 02139, USA
Work Phone: 1-617-715-6854
Fax: 1-617-621-1620
E-mail: sergey.sian@quintiles.com
Web: www.quintiles.com