Paper P004

All-purpose Programs

James Young, Exelixis Inc., South San Francisco, CA

ABSTRACT

A frequent practice in programming for clinical trial reporting is to reuse a program by copying the code and making a small change to meet new specifications. The new program file might be called by a different name and yet have only one line changed. Or the program might have the same name, but creates output that differs from before due to a change in SAS® code. The original program can be changed several times to meet the needs of new ad hoc analyses or new exploratory analyses. The resultant proliferation of new or changing programs is a source of confusion and can lead to results that cannot be replicated or that are difficult to validate. Associated files, such as the log and output, may also have different file names if there is a change in the new program name, and this adds to the confusion.

This paper presents strategies for creating a SAS program that does not change, but that produces different output depending on environmental variables, global SAS macro variables, or simply the location of the program in the file hierarchy. By building this versatility into a program once, the programmer can avoid file proliferation and the need for repeated validation, while at the same time meet the needs of people who want changes in analyses and displays of data.

The strategies and code described here have been developed on a Unix platform but should be adaptable to any SAS version and any platform, and should be of use to SAS users at all levels of experience.

INTRODUCTION

In most long term ongoing clinical trials, such as those in oncology, there is a need for repeated reporting of data. Safety is evaluated regularly, posters and papers are prepared for conferences, and data needs to be reviewed by data management. For example, routine quarterly reports can include the following listings:

Output file name	Title of Listing
`l_aeother_qtr.out`	Adverse Events with Missing Basic Information, or Onset Before First Dose
`l_aeall_qtr.out`	All Adverse Events
`l_aeallpp_qtr.out`	All Possibly or Probably Related Adverse Events
`l_aewd_qtr.out`	All Adverse Events for Subjects Who Have at Least one AE with Action Taken=Drug Withdrawal
`l_aewdds_qtr.out`	All Adverse Events for Subjects for Whom the Primary Reason for Study Drug Discontinuation Was an Adverse Event
`l_aetox_qtr.out`	All Adverse Events, sorted by Toxicity Grade, Subject, Preferred Term, and AE Onset Date

Since the PROC REPORT displays in the above files are almost identical in appearance, except for titles, sort order, and which subset of the data is used, it is convenient to have one program, `l_ae.sas`, produce all of the output files. The same program, without any modification, also produces the following files for our annual IND update reporting:

Output file name	Title of Listing
`l_aeother_ann.out`	Adverse Events with Missing Basic Information, or Onset Before First Dose
`l_aewd_ann.out`	All Adverse Events for Subjects Who Have at Least one AE with Action Taken=Drug Withdrawal

Compared to the similarly named quarterly reports in the previous table, these files have a different suffix (ann rather than qtr), a different subset of the data (only AEs with an onset date occurring in the previous year), and a footnote that specifies the 1-year reporting dates.

In a like manner, the same `l_ae.sas` program produces output for another task, which was to report data for a poster being prepared for an upcoming conference. The file names for this output are the same as for the quarterly reporting except that the suffix is "asco" rather than "qtr," as follows:

```
l_aeother_asco.out
l_aeall_asco.out
l_aeallpp_asco.out
l_aewd_asco.out
l_aewdds_asco.out
l_aetox_asco.out
```

"asco," which is the acronym of the conference where the poster appears, identifies these files as being produced for this poster task. The appearance of the listings is identical to that for the quarterly reporting except that data is from a different transfer date (noted in a footnote on all pages of the listings).

All the above output files, as well as other variations not described, are produced by the single program, `l_ae.sas`, and are possible without modifying the program or changing the time stamp of the program file. The same coding strategies can be applied to programs that create tables or do analyses. The following sections describe the computing environment and SAS code in this program that make this possible.

THE COMPUTING ENVIRONMENT

STANDARD HIERARCHICAL FILE STRUCTURE

Different studies have the same directory structure, so that relative directory locations are the same across all studies. The libname, rawdata, for example, refers to the directory "../rawdata", a directory parallel to the one where all SAS jobs for a study are run. A directory at a global level contains the macro library. All programs are run in batch. Each run references the local autoexec.sas file, which in turn %includes a global file that sets up the libnames and a standard set of global macro variables.

SAS COMMAND WRAPPER

A short Unix shell script, `esas`, acts as a wrapper for the batch sas command and controls each SAS job. The script allows options to control the SAS batch job. One of these options, -r, puts the text that follows the "-r" into a global macro variable, `_runf1`. This global run flag macro variable is available to any program and can help control the processing and output of the program. Depending on the syntax of the text that follows the -r, esas can also create `_runf2` and `_runf3` macro variables.

STANDARD GLOBAL SAS MACROS

In this environment, several standard macros located in the SAS program code create or reference macro variables and files external to the SAS program and help the programmer manage the output without modifying the program file. Some of those commonly used in all report generation programs are described below.

%subset

Subset evaluates the run flag 2 automatic macro variable, `_runf2`. If its value is "_ss," then it executes code that will subset the data being processed.

%mktitles

All of our titles and footnotes come by default from a text file called `tnf.txt`. The mktitles macro within the program searches this external text file for the lines that match the output file name of the program. An example of an entry in this file, for the output name `l_aewd_asco.out` might look like:

```
l_aeall_asco  | t | 3 |c|  | Listing 7
l_aeall_asco  | t | 4 |c|  | All Adverse Events
l_aeall_asco  | t | 5 |c|  |
l_aeall_asco  | fn| 1 |l|  | Note: data transfer date, &datadate..
```

The above lines define three titles and one footnote that will appear on all pages of the output. The mktitles macro keys off the name of the program output, which is "l_aeall_asco." In the first line, for example, "t" defines a title, "3" indicates title 3 on the page, "c" indicates that the title should be centered, and "Listing 7" is the text for title 3.

mktitles also assigns ".out" to output files, rather than the default ".lst".

%tnfother

This macro looks for a macro variable in the global environment that will be added to the output file name as a suffix, such as "_asco." If a file exists by the name of `tnf_asco`, then the program will use that table and footnote file. This helps to manage table and footnote files by preventing the default file, `tnf.txt`, from getting too large, and by keeping titles and footnotes associated with a particular project or task in a file identified with that task, in this case, the "tnf" file with a suffix of "asco."

%mkpages

Mkpages processes the output file and adds page numbers by modifying the string that appears in all our text output in title 2, "`Page XX of YYY.`"

STANDARDIZED DATA

The program, `l_ae.sas`, will run as is in any study which contains a standard analysis dataset, adae. This dataset uses CDISC SDTM standard variable names and structure.

THE PROGRAM CODE

OUTPUT FILES WITH DIFFERENT NAMES AND CONTENTS

```
esas –r_abc l_ae.sas
```

will create the global macro variable, _runf1, with a value of _abc. This is by default assigned to an internal macro variable, _suff, which becomes the suffix of the output file.

```
 %doit(_suff=&_runf1 );
[SAS code that executes differently depending on the value of &_suff.]
 %mend doit;
```

The program code will tack on _abc to the output filename so that the output name is `l_ae_abc.out.` This is handled by the mktitles macro:

```
%mktitles ( report=&_prognm.&_suff  );
```

If there is code in the doit macro that does different subsetting or other processing depending on the value of `_suff,` the output contents, not only the output name, will also be different. For example, the data will be subset by the in-reporting-period flag variable, `in_rp`, if the value of `_suff` is `wd_ann`:

```
%if &_suff = wd_ann  %then %do ;
    data &_suff ;
         set wdae ;
         if in_rp = 'Y' ;
      run ;
%end ;
```

MULTIPLE OUTPUT FILES WITH DIFFERENT NAMES

Multiple calls to the doit macro, each specifying a different value of `_suff`, will output a different file name and contents with each call. `l_ae.sas` contains a second macro, called `doit2`, that is used to generate multiple routine reports depending on the value of `_runf1`. The macro below will generate quarterly or annual reports, depending on `_runf1`.

```
%macro doit2 ;
%If &_runf1 ^= %then %do;
       %if &_runf1 = _qtr  %then %do ;
              %doit(_suff=other_qtr) ;
              %doit(_suff=all_qtr) ;
              %doit(_suff=allpp_qtr) ;
              %doit(_suff=wd_qtr) ;
              %doit(_suff=wdds_qtr) ;
              %doit(_suff=tox_qtr) ;
       %end ;
       %else %if &_runf1 = _ann  %then %do ;
              %doit(_suff=other_ann) ;
              %doit(_suff=wd_ann) ;
       %end ;
%end;
```

If `_suff` is `tox_qtr`, for example, the output will be sorted by toxicity grade first and then by subject. The output will be named `l_aetox_qtr.out`. If `_suff` is `wd_qtr` then the output will subset for any subjects who had an AE resulting in withdrawal of treatment. The output will be named `l_aewd_qtr.out`. The `tnf.txt` file or `tnf_qtr` file, if it exists, will define the titles and footnotes for each of the different variations in output name.

ALTERNATE OUTPUT FOR ZERO OBSERVATIONS

The final dataset for proc report, especially if it is reporting on a small subset of subjects, may contain no observations. Rather than produce no output file, which is the default behavior when PROC REPORT processes an empty dataset, `l_ae.sas` tests for no observations

```
data _null ;
   if &nobs = 0 then call execute ('%emptylst') ;
run ;
```

and executes the emptylst macro which by default produces a PROC REPORT output page as follows:

```
**********************************************************************************************************
                                        ***** No Observations *****
**********************************************************************************************************
```

AUTOMATICALLY REMOVING OR ADDING COLUMNS IN THE LISTING

a) By checking program location

When creating a listing for a single study, the leftmost column is the subject id. When creating a listing of data from combined studies, the leftmost column indicates the study number. This is handled by code that determines if the program is being run from the combined study area or from the regular study area. Combined study directories have the characters “all,” such as “547all,” whereas regular study directories have numbers, such as “547001”, for study “001.” The following code in `l_ae.sas` determines which directory location the program is running in from an automatic global macro variable holding the name of study directory. The code then creates several macro variables, and these control code in PROC REPORT that does or does not add the column.

```
%let _study=;
%let _idwidth=8;
%let _allloc=;
data _null_;
   dloc="x&_studynm";
   if index(dloc, 'all') > 0 then do;
      call symput('_allloc','Y');
      call symput('_study','_study');
      call symput('_idwidth','8');
   end;
run;
```

The PROC REPORT code contains the following code fragments that reference these macro variables:

- `column ( &_study _id aeterm aedecod …`

- `define _id   / group width=&_idwidth 'Subject*ID*(Cohort)' flow;`

- ```
  %if &_allloc = Y %then %do;
     define _study   / group   width=4  'Stu-*dy' flow;
  %end;
  ```

b) By checking for existence of a dataset

The following code fragment also controls code in the SAS program for showing or not showing an additional column, depending on the existence of a dataset.

```
%if %sysfunc(exist(sdtmdata.vs)) %then %do;  ….
```

AUTOMATICALLY FORMATTING REPORTED DATA BASED ON DATASET CONTENTS

If studies use two different schemes to code adverse event relationship, the following program code detects this:

```
data _null_;
   set ae;
   if aereln=2 and index(upcase(aerel),'POSSIBLY') then do;
      call symput('_relnpf',"Y");
   end;
run;
```

The values (the `aerel` format) and titles (the `_relttl` macro variable that becomes part of the title) for these data appear in the output differently, depending on the value of the macro variable, `_relnpf`, as shown by the following code:

```
proc format ;
%if &_relnpf = Y %then %do;
    value aerel
        1 = 'Not related'
        2 = 'Possibly'
        ;
    %let _relttl=Possibly;
%end;
%else %do;
   value aerel
        1 = 'Unrelated'
        2 = 'Unlikely '
        3 = 'Possibly '
        4 = 'Probably '
        ;
   %let _relttl=Possibly or Probably;
%end;
```

CONCLUSION

A large variety of coding techniques allows creation of multiple different outputs from a single program. These are especially useful when the same basic listing, table, or figure needs to be created with slight variations. Reusing the program in this manner helps to reduce the number of programs needing maintenance and allows for flexibility in meeting new ad hoc reporting requirements. A standardized computer environment also allow and extends the usefulness of these techniques.

ACKNOWLEDGEMENTS

I would like to thank the statistical programmers in the Biostatistics and Data Management Department at Exelixis for their numerous ideas and code contributions.
```

CONTACT INFORMATION

Your comments and questions are welcome. Contact the author at:
James Young
Exelixis, Inc.
South San Franciso, CA
Email: jayoung@exelixis.com