PharmaSUG 2019 - Paper DV-24

Free Duplicates Here! Get your Free Duplicates!

Kristen Harrington, Rho, Chapel Hill, NC USA

ABSTRACT

In clinical trials, it is common to produce an overall summary as well as several subset versions of that summary. The overall summary is typically referred to as “unique” and the subset versions as “duplicates”. For traceability, a separate program is required for each individual output; cramming the creation of the unique and duplicates into a single program won't pass muster. The most common approach to this problem is to create a macro. This macro is called by each of the separate unique and duplicate programs to produce the output files. The brute force method to create these separate programs is manually copying the unique program once for each duplicate to be produced, renaming files and changing values of macro subsetting parameters. This paper will explore an automated approach to creating the duplicate SAS® programs.

INTRODUCTION

Analyzing a single dataset with various subsets is extremely common in clinical trials. Normally, a unique mock template of an analysis display is duplicated numerous times, each time with a differing subset of the data. Commonly, a brute force method of manually creating an initial program, copying the program, renaming, and making updates to the appropriate parameters is used to develop these duplicate displays. While this process ensures a single SAS program, .log, .lst and final display are created, there are still drawbacks: The process is time consuming; Accidentally overwriting a SAS program is quite easy during the renaming and saving; It is easy to overlook a subset and, when creating hundreds of individual duplicate files, it is difficult to find where subsets were missed.

This discussion focuses on programmatically creating multiple SAS programs containing a header and an output specific macro call from a single source program. The process outlined below maintains all of the copying and pasting in one file, the source program. Keeping the macro calls contained adds the additional benefit of allowing the programmer to visually QC the subsets to ensure the appropriate items are considered, and that no subsets are duplicated unintentionally. Further, audit trails are fully traceable as each program produces a .log, .lst and final display with no additional .log, .lst or final output file redirection coding. Here are the process steps:

- Creation of a summary macro that supports the output summary
    - This is already a common practice for displays with numerous duplicate analyses
    - A reasonable example is adverse event frequencies with each display presenting a different demographic grouping and further subsets of the data (i.e. Treatment-Emergent, Serious Treatment-Emergent, etc)
- Creation of a shell file that contains a header and a macro call with unspecified parameters
- Creation of a source template program which produces the individual SAS programs containing appropriately populated headers and a macro call with specified parameters for each output

SUMMARY MACRO (AEGROUPS.SAS)

The summary macro is similar to a typical summary output macro. While the summary macro is not the primary focus of this process, it is important to note. The summary macro is developed robustly to handle various demographic groupings with differing number of categories, as well as differing subsets for the analysis. Typical parameters of a summary macro include the output file or display name, population, treatment group, additional analysis subsets, and specific row text for the first row of the display. Our example of an adverse event frequency table also requires a grouping identifier for the demographic information, maximum number of groups within the specified demographic variable, and display specific group label.

```
%macro aegroups(table=,  /*output file name*/
                pop=,    /*population*/
                trtgrp=, /*treatment group*/
                addsub=, /*additional subsets*/
                rowtxt=, /*row text for first row of the display*/
                grp=,    /*grouping identifier (ex. country)*/
                max=,    /*maximum number of categories within the group*/
                lbl=)    /*group label for the display header*/
                         ;

 <coding for adverse event frequencies with grouping and subsets>
%mend aegroups;
```

SHELL FILE (SHELLGRPTABLE.SAS)

The shell template file contains a header and a macro call to the summary macro with unspecified parameters. This file is a template only. Keywords are utilized such that the source program finds and replaces the appropriate parameters. In the example below, the keyword TABLENAME is used. Additionally, the program also uses text strings (i.e. the parameters) to determine the appropriate replacement text:

```
/*****************************************************************************
******

PROJECT:          <Sponsor – StudyID>

PROGRAM:          tablename.sas

PURPOSE:          Create Table tablename

INPUT:            <location of data>

OUTPUT:           <location of output files>\tablename.(log lst)

CALLS:            %aegroups


PROGRAM HISTORY:

DATE        PROGRAMMER        DESCRIPTION

---------   ---------------   -----------------------------------------------
------

YYYY-MM-DD  Programmer        Create

*****************************************************************************
*****/
```

<Additional inclusion statements if needed>

```
/*************

* CALL MACRO *

**************/

%aegroups(table=tablename,

          pop=,
          trtgrp=,
          addsub=,
          rowtxt=,
          grp=,
          max=,
          lbl=
            );
```

SOURCE PROGRAM (MAKETABLES.SAS)

The source program generates the multiple SAS programs that produce individual duplicate outputs. Each SAS program created by the source program mimics the shell file (SHELLGRPTABLE.sas) with the appropriate parameters specified. The source program is essentially a macro with multiple calls within a single program. Each call produces an individual SAS program. The source program macro utilizes the same parameters as the shell file with the addition of a parameter specifying the appropriate shell to use. Additional coding is added to include the date, programmer's name or any other appropriate information.

Within a data _null_ step, the shell program is read in using an infile statement and by use of a put statement a file, named as specified within the macro call, is created. The new file is modified to replace either keywords or text strings as noted below. Note in the example below, the pop parameter is reassigned as pop=saffl and is not included in the call to %maketables, but will be replaced within the new SAS program:

```
%macro MakeTables(shell=,tablename=,trtgrp=,addsub=,rowtxt=,grp=,maxN=,lbl=);
   %*-----------------------------------------------;
   %* Copy template shell, modify parameters, and    ;
   %* output individual SAS files                    ;
   %*-----------------------------------------------;

     DATA _null_;
       infile "<location of shell macro>\&shell..sas";
       file   "<location of output files>\&tablename..sas";

       input;
            _infile_ = tranwrd(_infile_,"tablename","&tablename");
            _infile_ = tranwrd(_infile_,"pop=","pop=saffl");
            _infile_ = tranwrd(_infile_,"trtgrp=","trtgrp=&trtgrp");
            _infile_ = tranwrd(_infile_,"addsub=","addsub=&addsub");
            _infile_ = tranwrd(_infile_,"rowtxt=","rowtxt=&rowtxt");
            _infile_ = tranwrd(_infile_," grp="," grp=&grp");
            _infile_ = tranwrd(_infile_,"max=","max=&maxn");
            _infile_ = tranwrd(_infile_,"lbl=","lbl=&lbl");
```

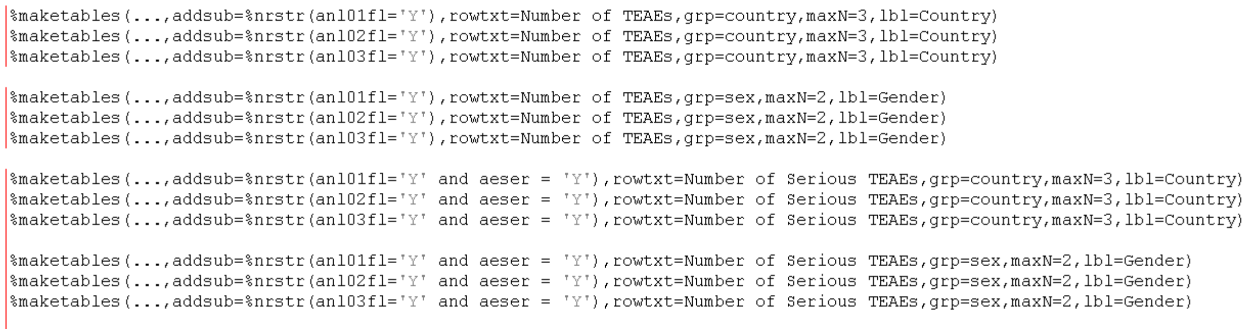
```
          put _infile_;

     RUN;
%mend MakeTables;
```

Example call:

```
%maketables(shell=shellgrptable,    /*name of the shell file to read in*/
            tablename=ae-taa,     /*This was specified as a keyword*/
            trtgrp=1,
            addsub=%nrstr(anl01fl='Y'),
            rowtxt=Number of TEAEs,
            grp=country,
            maxN=2,
            lbl=Pooled Country)
```

Within the source program, all calls to %maketables are coded on a single line. Doing so, allows easier viewing of each parameter's specifications. This allows the programmer to quickly QC the parameters and ensure the appropriate specifications are included. A few examples of this with limited parameters are shown below:



```
%maketables(...,addsub=%nrstr(anl01fl='Y'),rowtxt=Number of TEAEs,grp=country,maxN=3,lbl=Country)
%maketables(...,addsub=%nrstr(anl02fl='Y'),rowtxt=Number of TEAEs,grp=country,maxN=3,lbl=Country)
%maketables(...,addsub=%nrstr(anl03fl='Y'),rowtxt=Number of TEAEs,grp=country,maxN=3,lbl=Country)

%maketables(...,addsub=%nrstr(anl01fl='Y'),rowtxt=Number of TEAEs,grp=sex,maxN=2,lbl=Gender)
%maketables(...,addsub=%nrstr(anl02fl='Y'),rowtxt=Number of TEAEs,grp=sex,maxN=2,lbl=Gender)
%maketables(...,addsub=%nrstr(anl03fl='Y'),rowtxt=Number of TEAEs,grp=sex,maxN=2,lbl=Gender)

%maketables(...,addsub=%nrstr(anl01fl='Y' and aeser = 'Y'),rowtxt=Number of Serious TEAEs,grp=country,maxN=3,lbl=Country)
%maketables(...,addsub=%nrstr(anl02fl='Y' and aeser = 'Y'),rowtxt=Number of Serious TEAEs,grp=country,maxN=3,lbl=Country)
%maketables(...,addsub=%nrstr(anl03fl='Y' and aeser = 'Y'),rowtxt=Number of Serious TEAEs,grp=country,maxN=3,lbl=Country)

%maketables(...,addsub=%nrstr(anl01fl='Y' and aeser = 'Y'),rowtxt=Number of Serious TEAEs,grp=sex,maxN=2,lbl=Gender)
%maketables(...,addsub=%nrstr(anl02fl='Y' and aeser = 'Y'),rowtxt=Number of Serious TEAEs,grp=sex,maxN=2,lbl=Gender)
%maketables(...,addsub=%nrstr(anl03fl='Y' and aeser = 'Y'),rowtxt=Number of Serious TEAEs,grp=sex,maxN=2,lbl=Gender)
```

CONCLUSION

As most programmers utilize macros to produce numerous duplicate output summary files, the process of using a shell file in conjunction with a program that replaces text and produces or replaces SAS files saves time. This process eliminates the need to copy a SAS program with a macro call, save it as a new file, and update each header and individual parameters. Starting with a robust summary macro which handles various types of subsets and groupings is key to this process. With the method discussed, all duplicate macro calls are contained in a single program making it very easy for the programmer to make subsetting updates as well as verify that the appropriate subsets were used. The source program produces individual SAS files, that in turn issue individual log and output files, ensuring traceability.

To expand further on this, the process outlined above is one step away from being handled via an external file which contains all of the appropriate information (i.e. parameters for the source program). This ensures that the file is read in and the individual SAS programs are produced with no further editing of the source program.

CONTACT INFORMATION

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

Kristen Harrington
Rho, Inc.
919-408-8000
Kristen_Harrington@RhoWorld.com