PharmaSUG 2014 - Paper CC18

Macro to check Audit compliance and standards of SAS® programs

Seeja Shetty, inVentiv Health Clinical, Mumbai, India

ABSTRACT

As a lead statistical programmer in clinical trial study you are responsible to submit an accurate representation of the collected data in the form of analysis datasets, summary tables, listing and figures. In addition, you need to ensure the SAS programs which created the above mentioned reports are compliant to Good Programming Practices. Even considering a small study it is tedious and time consuming to go through all the programs individually.

This paper demonstrates a SAS macro to ensure basic checks for the GPP compliance which might divulge inconsistencies during study Audits.

INTRODUCTION

This paper would be covering some of the basic irregularities observed in a study from the programming perspective. Specifically,

1) Was any information deleted using hardcoding?
2) Is log missing for any program?
3) Is log clear of errors, warnings, uninitialized and repeats?
4) Is any output listed in Status Sheet missing from output folder?
5) Is any output missing from those mentioned in Programming Sheet?
6) Is the program re-run date updated as per the latest run in the Status Sheet?
7) Last saved date of the program is later than “Validation complete” date?

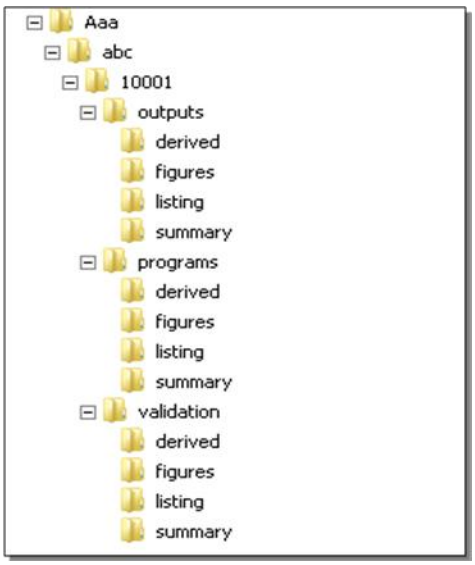
BASIC LOGIC BEHIND THE MACRO

To flag the irregularities, we would need the dates on which the files were created, the dates on which they were modified. And to check for deletion, errors, warnings etc. we would also need to read the actual content of the files. Once we have all this in one place it becomes very easy to check for issues.

Now creation of this very dataset is elaborated in Steps #1 to #4 on page 2.

PRE-REQUISITES AND ASSUMPTIONS

To begin with, we would expect a CDISC based study. For example, let us assume the study is for AAA Pharmaceuticals Inc. with the compound name as ABC and protocol number 10001 and has the following folders.



Also, we expect a basic Status Sheet in the below format.

A	B	C	D	E	F	G	H	I	J	K
ID Number	Program Name	First Title	Output File Name	Unique/ Copy	Original Program Release Date	Original Program Release Validation Date	Programmer	Program Re-Run Date	Validation Programmer	Validation
1.01	rglab	Haematology - ANC	F_01_01_anc	Unique	4/14/2014					
1.02	rglab	Haematology - Basophils	F_01_02_basop	Copy						
1.03	rglab	Haematology - Eosinophils	F_01_03_eosp	Copy						
1.04	rglab	Haematology - Haematocrit:	F_01_04_hct	Copy						

If the standards are followed then all the programmer would have to do is pass the below details in the macro parameters and get a resultant collated Issue list on the macro call. Where the macro call has the parameters,

```
%macro codename(stdyfldr = , /*Study path for the programs*/
                Plogfldr = , /*Study path for the program logs*/
                Outfldr  = , /*Study path for the outputs*/
                Valfldr  = , /*Study path for the validation programs*/
                Prshtpt  = , /*Study path for the programming Staus Sheet*/
                Dbms     = , /*DBMS for the Status Sheet*/
                Sheet    =   /*Sheet name incase of multiple sheets*/);
```

STEPWISE EXPLANATION OF THE MACRO

We assume the study is ready from delivery perspective i.e. all the Outputs, Logs are created and verified .The Status Sheet is updated for the programs with the respective outputs, dates etc. The macro verifies this readiness.

Now in this section, we will try and understand to create a master dataset which will be used as base for all the checks listed on page #1.

STEP 1:- We create a temporary library in the WORK library to avoid overwriting any file or SAS dataset in the study area.

```
filename mkdir pipe "mkdir %sysfunc(pathname(work))\temp";

data _null_;
   infile mkdir;
run;
libname temp "%sysfunc(pathname(work))\temp";
```

STEP 2:- Next we read all the external file names from the study folder. At the end of the execution of below datastep, CODELIST dataset will consist of all the file names in the specified folder.

```
data temp.codelist;
      length name pgmname cat $20 filepath $200;
      drop rc did i;
      %if &stdyfldr ne %then %do;
      did=dopen("stdyfldr");
      if did > 0 then do;
            do i=1 to dnum(did);
                  pgmname=dread(did,i);
                  filepath=strip("&stdyfldr")||"\"||strip(pgmname);
                  if index(compress(lowcase(pgmname)),".sas")>0 then
                        cat="Program";
                  else if index(compress(lowcase(pgmname)),".log")>0 then
                        cat="Log";
                  name=compress(scan(pgmname,1,"."));
```

```
            output;
      end;
            rc=dclose(did);
      end;
      else put 'Could not open directory';
      output;
run;
```

STEP 3:- We get the creation and modified dates of the files in the specified folder.

```
data temp.info;
   length infoname infoval $250;
   set temp.codelist;
   drop rc fid infonum i close;
   rc=filename("abc",filepath);
   fid=fopen("abc");
   infonum=foptnum(fid);
   do i=1 to infonum;
        infoname=foptname(fid,i);
      infoval=finfo(fid,infoname);
      output;
   end;
   close=fclose(fid);
run;
```

STEP 4:- Now using INFILE we will convert the filenames read from from *STEP 2* to a SAS dataset. We will append all the programs in one dataset CODE.By the end of the execution of STEP 4 we will have a master dataset CODEFIN with the program names, file path, their created/modified dates. The variable LINES will have the .SAS codes.

This master dataset can be used to check all the anomalies listed on page #1 and more.

```
data temp.code;
      length lines $32767 filename $200;
      lines="";
      filename="";
run;

*****************************************
converting the programs to datasets
*****************************************;
%do i=1 %to &n;

filename codename "&&filename&i";

data temp.dummy;
      length filename $200;
      infile codename missover length=reclen;
      input lines $varying32767. reclen;
      filename="&&filename&i";
run;

* appending all the codes in a folder;
proc append base=temp.code data=temp.dummy force;
run;
%end;
```

```
data temp.codefin(where=(pgmname ne ""));
      merge temp.t_info(where=(index(lowcase(pgmname),".sas")>0
      or index(lowcase(pgmname),".log")>0))
              temp.code;
      by filename;
run;
```

OUTPUT:- Once we have our all our information under one dataset we would proceed with the checks (from page 8). Please follow the full macro in the Appendix section for the same.
Now following the macro call, the FINAL dataset will show the Issue List as below.
Depending on the variables SUBCAT and DETAILS, appropriate action will have to be taken with respect to the corresponding file listed under PGMNAME.

```
%codename(
       stdyfldr=C:\Users\SShetty\Desktop\rd\Aaa\abc\10001\programs\figures,
      plogfldr  =C:\Users\SShetty\Desktop\rd\Aaa\abc\10001\programs\figures ,
      outfldr   =C:\Users\SShetty\Desktop\rd\Aaa\abc\10001\outputs\figures ,
      valfldr   =C:\Users\SShetty\Desktop\rd\Aaa\abc\10001\validation\figures,
      prshtpt=C:\Users\SShetty\Desktop\rd\Aaa\abc\10001\Programming_Status_Final.xls,
      dbms=xls,
      sheet=GRAPH);
```

pgmname	cat	subcat	details
rgcoag.sas	Program	Deleted records	where subject ne '101'
rgcoag.sas	Log	Missing Log file	
v_rglab.sas	Log	Missing Log file	
v_rgvs.sas	Log	Missing Log file	
F_03_01_itt	Output	Dates mismatch	Output missing from Output folder however it is released in Status sheet.

Output 1. Final output

CONCLUSION

The macro will work well with pre-requisites listed on pages #1 & #2. Minor tweaks will be required to the macro depending on the folder structure and the status sheet. The macro only flags some basic irregularities. However, the master dataset which is created by the end of Step #4 can be used for further processing.

REFERENCES

SAS online documentation: Dictionary of Language Elements, Features of the SAS Language for Windows

https://support.sas.com/documentation

ACKNOWLEDGMENTS

I would like to thank all my colleagues who have helped me with their valuable inputs and discussion on the check which inturn added value to this MACRO. I am grateful to Sandeep Sawant, Manager, Statistical Programming, inVentiv Health Clinical, India, whose guidance and encouragement has helped me while writing this paper.

CONTACT INFORMATION

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

Seeja Shetty
inVentiv Health Clinical
Marwah Centre, Ground Floor, B Wing, Krishnalal Marwah Marg,
Andheri(East), Mumbai – 400072
Phone: +91 22 4095 7362
Fax: +91 22 4095 7399
E-mail: seejasagar@gmail.com

APPENDIX

```
options nosymbolgen nomlogic;

/*****************************************
 Creating libname for temporary data storage
*****************************************************/
filename mkdir pipe "mkdir %sysfunc(pathname(work))\temp";

data _null_;
   infile mkdir;
run;
libname temp "%sysfunc(pathname(work))\temp";

*********************************************
getting program/log /output names from the folder
************************************;
%macro codename(stdyfldr= ,plogfldr= ,outfldr= ,valfldr= ,prshtpt= ,
                  dbms= ,sheet=);

%if &stdyfldr ne %then %do; filename stdyfldr "&stdyfldr.";%end;
%if &plogfldr ne %then %do;filename plogfldr "&plogfldr.";%end;
%if &outfldr  ne %then %do;filename outfldr "&outfldr.";%end;
%if &valfldr  ne %then %do;filename valfldr "&valfldr.";%end;

data temp.codelist(where=(pgmname ne "" and
            (index(lowcase(compress(pgmname)),".sas") or
            index(lowcase(compress(pgmname)),".log") or
            index(lowcase(compress(pgmname)),".rtf"))));
      length name pgmname cat $20 filepath $200;
      drop rc did i;
      %if &stdyfldr ne %then %do;
      did=dopen("stdyfldr");
      if did > 0 then do;
            do i=1 to dnum(did);
                  pgmname=dread(did,i);
                  filepath=strip("&stdyfldr")||"\"||strip(pgmname);
                  if index(compress(lowcase(pgmname)),".sas")>0
                  then cat="Program";
                  else if index(compress(lowcase(pgmname)),".log")>0
                  then cat="Log";
                  name=compress(scan(pgmname,1,"."));
            output;
      end;
            rc=dclose(did);
      end;
      else put 'Could not open directory';
      output;
      %end;
      %if &plogfldr ne and (&stdyfldr ne &plogfldr) %then %do;
      did=dopen("plogfldr");
      if did > 0 then do;
            do i=1 to dnum(did);
                  cat="Log";
                  pgmname=dread(did,i);
                  filepath=strip("&plogfldr")||"\"||strip(pgmname);
                  name=compress(scan(pgmname,1,"."));
```

```
                output;
      end;
            rc=dclose(did);
      end;
      else put 'Could not open directory';
      output;
      %end;
      %if &outfldr ne %then %do;
      did=dopen("outfldr");
      if did > 0 then do;
            do i=1 to dnum(did);
                  cat="Output";
                  pgmname=dread(did,i);
                  filepath=strip("&outfldr")||"\"||strip(pgmname);
                  name=compress(scan(pgmname,1,"."));
            output;
      end;
            rc=dclose(did);
      end;
      else put 'Could not open directory';
      output;
      %end;
      %if &valfldr ne %then %do;
      did=dopen("valfldr");
      if did > 0 then do;
            do i=1 to dnum(did);
                  cat="Validation";
                  pgmname=dread(did,i);
                  filepath=strip("&valfldr")||"\"||strip(pgmname);
                  name=compress(scan(pgmname,1,"."));
            output;
      end;
            rc=dclose(did);
      end;
      else put 'Could not open directory';
      output;
      %end;
run;

proc sort noduprecs; by cat name pgmname filepath;run;

*****************************************
getting modified/created dates of program
*****************************************;

data temp.info;
   length infoname infoval $250;
   set temp.codelist;
   drop rc fid infonum i close;
   rc=filename("abc",filepath);
   fid=fopen("abc");
   infonum=foptnum(fid);
   do i=1 to infonum;
        infoname=foptname(fid,i);
      infoval=finfo(fid,infoname);
      output;
   end;
```

```
   close=fclose(fid);
run;

proc sort; by cat name pgmname filepath; run;

proc transpose data=temp.info out=temp.t_info;
      by cat name pgmname filepath ;
      var infoval;
      id infoname;
run;

proc sort; by filename;run;

* count and read number of files names into macro variables;
proc sql noprint;
      select count(distinct pgmname) into :n from temp.t_info
            where index(lowcase(compress(pgmname)),".sas")>0;
      select distinct filename into :filename1 - :filename%trim(&n)
            from temp.t_info
            where index(lowcase(compress(pgmname)),".sas")>0;
quit;

data temp.code;
      length lines $32767 filename $200;
      lines="";
      filename="";
run;

****************************************
converting the programs to datasets
****************************************;
%do i=1 %to &n;

filename codename "&&filename&i";

data temp.dummy;
      length filename $200;
      infile codename missover length=reclen;
      input lines $varying32767. reclen;
      filename="&&filename&i";
run;
* appending all the codes in a folder;
proc append base=temp.code data=temp.dummy force;
run;
%end;

data temp.codefin(where=(pgmname ne ""));
      merge temp.t_info(where=(index(lowcase(pgmname),".sas")>0
      or index(lowcase(pgmname),".log")>0))
              temp.code;
      by filename;
run;

*to import the programming status sheet;
proc import out=temp.progsheet(where=(d ne ""))
                  datafile="&prshtpt."
                  dbms=&dbms replace;
```

```
                  %if &sheet ne %then %do;
                  sheet="&sheet";
                  %end;
                  getnames=no;
                  startrow=11;
run;

/*****************************************************
start programming to check for deviation from standards
*****************************************************/
*To check run dates issues;
proc sql;
      create table temp.prgsht1 as
            select a.*, pgmname, cat, last_modified, create_time
                  from temp.progsheet as a
                        left join
                  temp.t_info(where=(compress(lowcase(cat))="output")) as b
            on compress(lowcase(a.d)) = compress(lowcase(b.name))
      order by c, e;
quit;

data temp.dates(keep=pgmname cat subcat details);
      length subcat details $200;
      set temp.prgsht1;
      subcat="Dates mismatch";
      *Output missing from Output folder however it is released in
      Status sheet;
      if create_time ="" and g ^= . then do;
            cat="Output";
            pgmname=d;
            details="Output missing from Output folder however it is
                    released in Status sheet.";
            output;
      end;
      *Created date is after than Program Re-run date.;
      if input(compress(upcase(scan(last_modified,1,":"))),date9.)> i >.
      then do;
            details="Created date is after than Program Re-run date.";
            output;
      end;
      *Created date is after Validation Release date.;
      if input(compress(upcase(scan(last_modified,1,":"))),date9.) > l>.
      then do;
            details="Created date is after Validation Release date.";
            output;
      end;
run;

proc sql;
      create table temp.prgsht2 as
            select a.*, b.cat as pcat, b.last_modified as pmod,
            b.create_time as pcreate
                  from temp.prgsht1 as a
                        left join
                  temp.t_info(where=(compress(lowcase(cat))="program")) as b
            on compress(lowcase(a.b)) = compress(lowcase(b.name))
      order by c, e;
```

```
quit;

data temp.dates1(keep=pgmname cat subcat details);
      length subcat details $200;
      set temp.prgsht2;
      subcat="Dates mismatch";
      *Output missing from Output folder however it is released in
      Status sheet;
      if input(pmod,anydtdte.) > input(create_time,anydtdte.) >. then do;
            cat="Output";
            pgmname=d;
            details="Program is updated but output is not refreshed.";
            output;
      end;
      if input(pmod,anydtdte.) ne "" and input(create_time,anydtdte.)=""
      then do;
            cat="Output";
            pgmname=d;
            details="Per Status sheet the output "||compbl(d)||
                  "is expected for the program"||strip(b)||
                  ".sas .However currently it is missing. Please check
                  the output folder or the Status Sheet might need updates.";
            output;
      end;
run;

*To check absence of logs;
proc sql;
      create table temp.logcheck as
            select distinct pgmname, "Log" as cat length=100,
            "Missing Log file" as subcat length=200,
            "Log file is missing from the log folder." length=200
                        from temp.codelist
                  where index(lowcase(compress(pgmname)),".sas")>0 and
      name not in (select distinct name from temp.codelist where
index(lowcase(compress(pgmname)),".log")>0);
quit;

*check for errors, warnings and unin;
data temp.logerror;
      length pgmname $20 cat $100 subcat details $200;
      set temp.codefin(where=(index(lowcase(compress(pgmname)),".log")>0));
      subcat="Issues in log";
      if index(lowcase(lines),"error")>0 or index(lowcase(lines),"warn")>0
      or index(lowcase(lines),"unin")>0 or
      index(lowcase(lines),"w.d")>0 or index(lowcase(lines),"repeat")>0;
      details=lines;
run;

* To check all the deleted information;
data temp.deleted(keep=cat pgmname subcat details);
      length pgmname $20 cat $100 subcat details $200;
      set temp.codefin(where=(index(lowcase(compress(pgmname)),".sas")>0));
      if index(lowcase(lines),"delete")>0 or
         index(compress(lowcase(lines)),"subjidne")>0
         or index(compress(lowcase(lines)),"subjid^=")>0 or
         index(compress(lowcase(lines)),"subjid=")>0
```

```
         or index(compress(lowcase(lines)),"subjideq")>0 or
         index(compress(lowcase(lines)),"subjidnotin")>0
         or index(compress(lowcase(lines)),"subjidin")>0 or
         index(compress(lowcase(lines)),"ptne")>0 or
         index(compress(lowcase(lines)),"pt^=")>0 or
         index(compress(lowcase(lines)),"pt=")>0      or
         index(compress(lowcase(lines)),"pteq")>0 or
         index(compress(lowcase(lines)),"ptnotin")>0 or
         index(compress(lowcase(lines)),"ptin")>0;
      subcat="Deleted records";
      details=lines;
run;

data temp.final(keep=pgmname cat subcat details);
      length pgmname $20 cat $100 subcat details $200;
      set temp.deleted temp.logerror temp.logcheck temp.dates temp.dates1;
run;

proc sort; by pgmname cat subcat; run;

%mend;
```