PharmaSUG China 2019 – Paper CC-034

Repeated manual QC by Eyes? No, try to work easier

Cecilia (Xiaoyuan) Yang, Sanofi, Chengdu, Sichuan

ABSTRACT

During the long clinical trial journey, either as the production programmer or validation programmer, we face many repetitive tasks. Along the way we meet many milestones, DMC, DSR, CSR etc., and before each milestone we have to create outputs and QC them over and over again, completing each validation step in a limited time. With so many datasets or outputs to validate, how can we do this efficiently and accurately? By opening the QC results one by one? Of course not! Certain statements and procedures we use in our daily work can help us achieve this goal and make the QC process much easier to complete successfully. Using SASHELP.VTABLE, DO…loops etc. we show you how.

INTRODUCTION

In this article, we explain each statement/procedure how to meet their goal and provide the overall code in appendix section at the end. Libname statement, data statement, macro variable, Do…loop statement and compare procedure, which statement or procedure we use every day help us to implement it.

When we compare those datasets, there are some places we should pay attention.

1) The validation outputs should be created after the production outputs.
2) All variables and all observation should be compared.

OVERVIEW OF THE PROGRAM

STEP 1 LOCATE LIBRARY

In the example, we need to compare data set under "XXX/XXX/XXX/PRD/DATA" and "XXX/XXX/XXX/QC/DATA". By using libname statements to locate these two libraries.

```
libname library  "XXX/XXX/XXX/XXX" access=readonly;
```

Note: Add access=readonly to avoid modifying the data set under these folder.

Give the real path we need to take place of "XXX/XXX/XXX/XXX" and we could get library prd and library qc.

STEP 2 GET DATA SET'S NAME

How to get the all data set's name automatic? This is the first problem we need to solve.

There are 2 methods we could easily understand and use, we list below:

1) By using sashelp.vtable in data statement

```
data prd_data;
     length dtname $200;
     set sashelp.vtable end=last;
     where libname =upcase("prd");
     dtname=lowcase(memname);
     keep dtname crdate nobs nvar;
proc sort; by dtname;
run;
```

2) By using dictionary.tables in sql procedure

```
proc sql;
     create table prd_data
     as select lowcase(memname) as dtname
               length=200,crdate,nobs,nvar
     from dictionary.tables
     where libname=upcase("prd")
     order by dtname;
quit;
```

The same result we could get by using the different method. The table below is a dummy result and all columns we keep will be used at the following step.

Dummy result after running the code above:

dtname	crdate	nobs	nvar
AAA	2019-05-12	44	33
BBB	2019-06-24	24	44

STEP 3 AGGREGATE AND SPLIT DATA SET

Get an overall data set by using merge statement and split this overall data set into: 2 parts.

Need to create 3 data sets in this step:

1) dt_wocp.sas7bdat is the data set include name of all data sets which could not be compared, caused by the data set just exist in prd_data library or qc_data library. ➔ this data set is used at step 3
2) dt_wtcp.sas7bdat include name of all data sets need to compare. ➔ this data set is used at step 4 and step 5
3) chk_status.sas7bdat include name of all data set ➔this data set is used at step 5

```
data dt_wtcp dt_wocp chk_status;
     length compare_result $200;
     merge prd_data(in=a) qc_data(in=b);
     by dtname;
     if a and not b then do;
        compare_result="Data set is not in qc";
        output dt_wocp;
     end;
     else if not a and b then do;
        compare_result="Data set is not in prd";
        output dt_wocp;
     end;
     else output dt_wtcp;
     output chk_status;
run;
```

Note: compare result about the dataset which just exist in library prd or libray qc , are recorded in variable compare_result at this step. Compare result about the dataset exist in library prd and libray qc will recorded at step 5.

STEP 4 CREATE MACRO VARIABLE AND CALCULATE CODE BY USING DO…LOOP STATMENT

We need a macro variable to identify how many data sets we need to compare and pin down each data set's location in dt_wtcp.sas7bdat

```
proc sql;
     select count(distinct dtname) into: nobs_data
     from dt_wtcp;
quit;
```

Do…loop statement could help us to run SAS code multiple times. At this step, we create a macro named nobs_data and this macro let the program execute more automatically. How many times the code run depends on this macro variable.

```
%do i =1 %to &nobs_data;
                                …
                                …
%end;
```

STEP 5 COMPARE PROCEDURE

The COMPARE procedure compares the contents of two SAS data sets: base data set and compare data set. Following the code below, all compare result output to the diff data set.

```
proc compare base=base compare=compare listall
out=diff outbase outall outcompare outdiff outnoeq;run;
```

If the diff data set is empty, observations and variables in base data set and compare data set are equal, we could say these data set are equal. We could create 3 macro variable from the data set we get at the step 2: prd_nobs, prd_nvar, prd_create_date to know how many observations, variables and the creation date for each data set.

```
data _null_;
    set prd_data;
    if dtname="&pgm.";
    call symput("prd_nobs",put(nobs,best.));
    call symput("prd_nvar",put(nvar,best.));
    call symput("prd_create_date",put(crdate,e8601dt.));
run;
```

If QC creation date is earlier than production creation date, we give the value compare_result='QC created before prd'.

```
data  chk_status;
    set  chk_status;
    if … then do;
    …
    …
    ;
    end;
    else do;
    compare_result='QC created before prd';
    end;
run;
```

CONCLUSION

In this article, we use the statements and procedures in our daily work and try to simplify our QC process. By using this code, it save our times and our energies. The automatic QC is more efficient and accurate than our manual check by eyes with opening compare result output one by one. Sometimes, we may miss some places and let the QC quality fall off.

This code is modified easily to update the other studies or other folders. Except for being used to do the QC, this code can be used to compare same data set under different path. With changing only the libname statement, the code will execute well.

CONTACT INFORMATION

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

Name: Cecilia (Xiaoyuan) Yang

Enterprise: Sanofi

Address: 39F, in99 Chengdu Yintai Center, 1199 North Section of Tianfu Road, High-Tech Zone, Chengdu, P.R.C.

E-mail: Cecilia.yang@sanofi.com

APPENDIX

```
libname prd "XXX/XXX/XXX/XXX" access=readonly;
libname qc  "XXX/XXX/XXX/XXX" access=readonly;

data prd_data;
     length dtname $200;
     set sashelp.vtable end=last;
     where libname =upcase("prd");
     dtname=lowcase(memname);
proc sort; by dtname;
run;

data qc_data;
     length dtname $200;
     set sashelp.vtable end=last;
     where libname =upcase("qc");
     dtname=lowcase(memname);
proc sort; by dtname;
run;

data dt_wtcp dt_wocp chk_status;
    length compare_result $200;
    merge prd_data(in=a) qc_data(in=b);
    by dtname;
    if a and not b then do;
       compare_result="Dataset is not in qc";
        output dt_wocp;
    end;
    else if not a and b then do;
       compare_result="Dataset is not in prd";
      output dt_wocp;
    end;
    else output dt_wtcp;
    output chk_status;
run;

proc sql;
    select count(distinct dtname) into: nobs_data
    from dt_wtcp;
quit;

%macro repeat;
```

```
%do i =1 %to &nobs_data;

data ind_dt;
    set dt_wtcp;
    if _n_=&i.;
    call symput("pgm",dtname);
run;

data _null_;
    set prd_data;
    if dtname="&pgm.";
    call symput("prd_nobs",put(nobs,best.));
    call symput("prd_nvar",put(nvar,best.));
    call symput("prd_create_date",put(crdate,e8601dt.));
run;

data _null_;
    set qc_data;
    if dtname="&pgm.";
    call symput("qc_nvar",put(nvar,best.));
    call symput("qc_nobs",put(nobs,best.));
    call symput("qc_create_date",put(crdate,e8601dt.));
run;

proc compare base=prd.&pgm. compare=qc.&pgm. listall
out=diff outbase outall outcompare outdiff outnoeq;
run;

proc sql noprint; select count(*) into: diff from diff;quit;

data  chk_status;
    set  chk_status;
    if dtname eq "&pgm" then do;
      if input("&prd_create_date",e8601dt.) lt
      input("&qc_create_date",e8601dt.) then do;
      if compare_result="" then do;
           if &diff eq 0 and &prd_nobs eq &qc_nobs and
           &prd_nvar eq &qc_nvar then do;
           compare_result="Equal";
          end;
          else do;
           compare_result="No Equal";
          end;
```

```
          end;
        end;
     end;
     else do;
     compare_result='QC creat before prd';
     end;
run;
%end;
%mend repeat;

%repeat;
```