PhUSE 2014

Paper DH02

Consistent Data Delivery

Thevaki Mahadevan, ICON Clinical Research, Marlow, United Kingdom

ABSTRACT

Often data is delivered to Biostatistics with inconsistencies in the repeated extract of the raw data from one delivery to another, while programming is ongoing for interim deliveries. Some are true data issues that need to be queried by Data Management (DM). However, occasionally these are issues that occur during a mapping (e.g. to SDTM) or during the extraction/transfer process, and are not true data issues. These need to be highlighted to DM immediately for action and redelivery. How does Biostatistics best approach these raw data sanity/acceptance checks between each data delivery? One approach is to manually check subject counts in key datasets, another is to write SAS® code to check data programmatically. The advantages of doing checks programmatically are that it is more reliable, minimizes human error and less time consuming in the long run. A program can be written to cross check all raw datasets, variables and observations that are delivered each time. This paper will cover various approaches we can take.

INTRODUCTION

Raw data changes, from one delivery to another, and need to be very closely monitored, as this has become one of the typical routes to major quality findings at reporting. Some of the typical findings are:

- External data e.g. Lab/AE data, missing some records by mistake, that were delivered previously.
- Reference ranges used in Lab flagging has changed.
- Conversion formulas used to derive the standard values have changed.
- Code/decode values have changed
- Datasets, e.g. SAE dataset that get collected via different source, were not updated as part of the new delivery.

In addition to the above, there may be other things that need to be checked as well. These include making sure:

- There are no extra variables (not required) with all blank values.
- There are no missing required variables/datasets.
- There are no empty datasets (not required) with 0 observations being delivered.

As a standard process, generally programmers don't write programs to check the differences in the datasets between each delivery. Programmers tend to concentrate on any data issues with the current data and hardly cross reference the previous versions, as this is often overwritten by new data. Even the stat/lead reviewers are likely to miss these issues, as these can be very difficult to spot manually.

PROGRAMATICALLY CHECK DATA DELIVERY

One of the quick and best solutions for this issue is to save the previous version of datasets into another folder and then to check the differences between the new and old datasets using simple programs. If previously delivered datasets and the current datasets are saved in different folders, i.e. libraries, they can be referenced by libnames.

This referencing allow for SAS help files, i.e. sashelp.vcolumn files, to be referenced, to obtain metadata details on the content of the libraries. The number of datasets found within each library and their variables and their attributes within each dataset can be obtained from here.

```
/* Location of Old DM data with the new delivered DM data*/
libname DMOLD "/Volumes/app/sites/project/icon_derm/protocol/data/data_old"; /*Old RAW data*/
libname DMNEW "/Volumes/app/sites/project/icon_derm/protocol/data/data_old"; /*New RAW data*/
```

```
/*Get list of datasets and variables names from both libnames by accessing Dictionaries*/
  data ardmv;
   set sashelp.vcolumn (keep=libname memname name length label type format informat);
   where libname in ('DMOLD','DMNEW');
  run;
```

EWTABLE: Sashelp.Vcolumn

dit View Tools Data Solutions Help

LIBNAME	MEMNAME	MEMTYPE	NAME	TYPE	LENGT	NPOS	VARNUM	LABEL	FORMAT	INFORMAT
DMNEW	ADVERSE	DATA	AESITE	num	8	0	1	AE Site Category	BODSITE.	
DMNEW	ADVERSE	DATA	AECLAS	num	8	8	2	AE Class	AECLS.	
DMNEW	ADVERSE	DATA	COLLDATF	char	80	248	3	Char Collection Date of CRF Page		
DMNEW	ADVERSE	DATA	FROMDATF	char	80	328	4	Char Adverse Event From Date - Partial		
DMNEW	ADVERSE	DATA	TODATEF	char	80	408	5	Char Adverse Event To Date - Partial		
DMNEW	ADVERSE	DATA	ACCESSTS	num	8	16	6	Accessible TS	DATETIME.	DATETIME20.
DMNEW	ADVERSE	DATA	LOGINTS	num	8	24	7	Login TS	DATETIME.	DATETIME20.
DMNEW	ADVERSE	DATA	LSTCHGTS	num	8	32	8	LstChg Ts	DATETIME.	DATETIME20.
DMNEW	ADVERSE	DATA	LOCKFLAG	char	1	488	9	Data Lock Flag	$1.	$1.
DMNEW	ADVERSE	DATA	RDCIID	num	8	40	10	Received DCI Id	10.	11.
DMNEW	ADVERSE	DATA	DOCNUM	char	20	489	11	Document Number	$20.	$20.
DMNEW	ADVERSE	DATA	DCMNAME	char	16	509	12	DCM Name	$16.	$16.

Figure 1: Sample sashelp.vcolumn dataset

The common datasets found in both deliveries and the missing datasets found from each delivery can be produced programmatically just by doing a simple merge on these datasets metadata.

```
/*Check which datasets are in both library and if missing from one area*/
proc sort data=ardmv  out=dmo(drop=libname) nodupkey; where libname='DMOLD'; by memname; run;
proc sort data=ardmv out=dmn(drop=libname) nodupkey; where libname='DMNEW'; by memname; run;

  data both_data(keep=memname) inold_data innew_data ;
   merge dmo(in=a) dmn(in=b);
   by memname;
   if a and b then output both_data;
   if a  and not b then output inold_data;
   if b and not a then output innew_data;
  run;
```

MEMNAME	IN
ALLERGY	inold_dat
LAB_SDW	inold_dat
LB	inold_dat
NRS	inold_dat
OMIC	inold_dat

Figure 2: Example of output showing datasets only in old data

In addition, the common variables found in both deliveries and the missing variables found from each delivery can also be produced programmatically just by doing a simple merge on these datasets metadata.

```
/*Check which datasets and varibles are in both library and if missing from one area*/
proc sort data=ardmv  out=dmov(drop=libname); where libname='DMOLD'; by memname name; run;
proc sort data=ardmv out=dmnv(drop=libname); where libname='DMNEW'; by memname name; run;

  data both_var inold_var innew_var;
   merge dmov(in=a) dmnv(in=b);
   by memname name;
   if a and b then output both_var;
   if a  and not b then output inold_var;
   if b and not a then output innew_var;
  run;
```

MEMNAME	NAME	TYPE	LENGTH	LABEL	FORMAT	INFORMAT	IN
ADVERSE	DBSSITEF	char	200	Body Site Involvement-FUL	$200.	$200.	inold_var
ADVERSE	DBSSITEL	char	200	Body Site Involvement-DLV	$200.	$200.	inold_var
ADVERSE	DBSSITEN	num	8	Body Site Involvement-DVN	10.		inold_var

Figure 3: Example of output showing only variables in old dataset

PROC COMPARE OVERVIEW

Once the common datasets are identified the compare procedure can be used to closely look at the data. A macro variable referencing can be made, to call each dataset and some of their associated common variables. This can then be referenced by PROC COMPARE to obtain all the differences between each delivery datasets. We can then add a do loop to do the compare of all datasets, in one step.

```
/*_____________________________________________________________________________________
/          Proc compare DM old with DM new
/ Create macro variables for each common datasets found in both library and for total number
/ of datasets found.
/_____________________________________________________________________________________*
proc sort data=both_data out= both_dat1 nodupkey; by memname; run;

   data _null_;
    set both_dat1 end=last;
    if last then  call symput('tot1',_n_);
    call symput('DSET1'||left(_N_), trim(left(memname)));
   run;
  %put &dset12;
```

The PROC COMPARE first compares the data set attributes,variables, attributes (type, length, labels, formats, and informats of matching variables.), and observations. Then it checks each observation in one data set to determine whether it matches an observation in the other data set. PROC COMPARE either matches observations by their position in the data sets or by the values of the ID variable. After making these comparisons, PROC COMPARE compares the values in the parts of the data sets that match.

The PROC COMPARE produces first produses a Data Set Summary report. This lists the attributes of the data sets being compared. These attributes include the following: the data set names,the data set types, the data set labels, the dates created and last modified, the number of variables in each data set, the number of observations in each data set.

The Variables Summary report compares the variables in the two data sets. The first part of this report lists the the number of variables the data sets have in common, the number of variables in the base data set that are not in the comparison data set and vice versa,the number of variables in both data sets that have different types, the number of variables that differ on other attributes and the number of BY, ID, VAR, and WITH variables specified for the comparison.The second part of the report lists matching variables with different attributes and shows how the attributes differ. The observation summary report provides information about observations in the base and comparison data sets. PROC COMPARE also produces a Values Comparison Summary and a Value Comparison Results.

There are few options we can use in PROC COMPARE statement that is worth stating here. To specify the data sets to compare the BASE= option specify the base data set and the COMPARE= option specify the comparison data set. There are some options that control the output dataset. The OUTDIF option Write an observation that contains the differences for each pair of matching observations. The OUTNOEQUAL option Suppress the writing of observations when all values are equal. There are some options that Control the details in the default report including NOVALUES option which suppress the value comparison results. LISTALL option control the listing of variables and observations and list all variables and observations found in only one data set. LISTBASE option list all variables and observations found only in the base data set, while LISTCOMP list all variables and observations found only in the comparison data set.

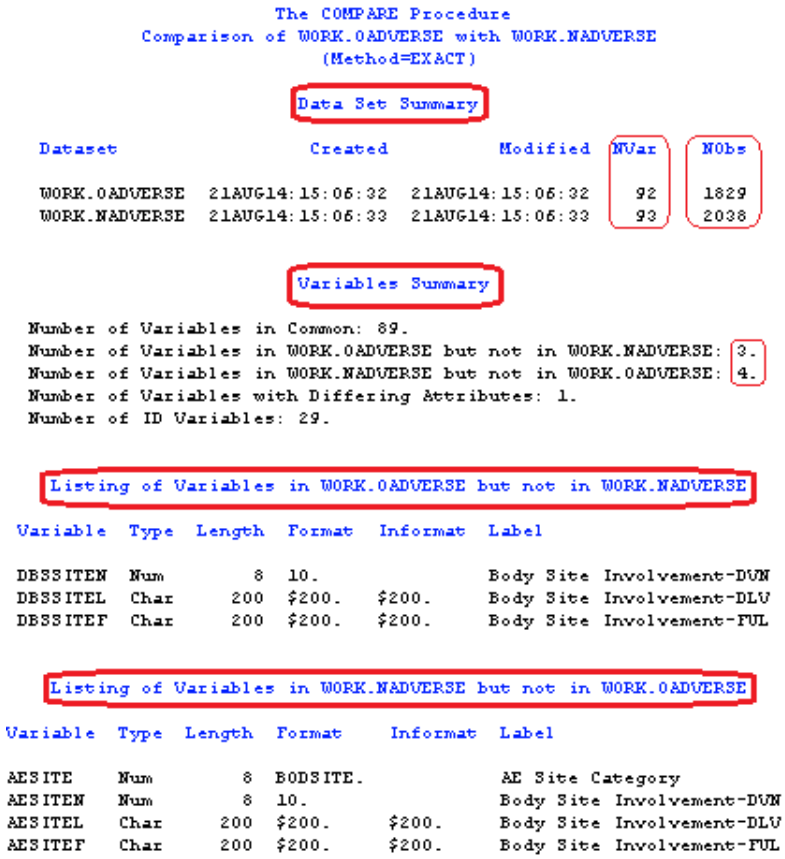
Adding a By statement to the PROC COMPARE produces a separate comparison for each BY group. Selecting all variables in the dataset as a by variable for compare is meaningless at this point. As more observations and modified data are expected for each subsequent data extract and hence differences in observations and values are also expected. If few by group variables are selected than required, then the compare is not done properly as a duplicate observation message comes up. Hence only the numeric by variables along with subjid, study, and inv, if present, are selected to produce a separate comparison. Selecting only some of the key by group variables this way is just adequate for this comparison and to get an overall view of the dataset.

BULK RUN OF PROC COMPARE

```
%macro comp;
  %do i= 1 %to &tot;
 /*
 /Obtain byvar variables for proc sort before proc compare
 /Take only the numeric variables found in both datasets to create a macro by variable
 /(Selecting all variables as a by var is pointless for data delivery consistency check)
 /Where subjid is collected create another character macro by variable
 /Create where clause macro variable if needed
*/
   data dbyvar(keep=memname byvar byvar2 nonel);
    set both_var;
    retain byvar byvar2 nonel;
    format byvar2 $18. byvar $900. nonel $5.;
    by memname;
    if strip(type) eq 'num' then do;
    if first.memname then byvar=name;
    else byvar=catx('',byvar,name);
    end;
    if first.memname then byvar2='';
    if name eq 'SUBJID' then byvar2='study inv subjid';
    if first.memname then nonel='';
    if name eq 'NONEL' then nonel='nonel';
     if last.memname then output;
   run;

   data _null_;
    set dbyvar;
    if _n_;
    if strip(memname)  eq "&&dset&i" then call symput ('bvar' || left(_n_),
trim(left(byvar)));
    if strip(memname)  eq "&&dset&i" then call symput ('bvar2' || left(_n_),
trim(left(byvar2)));
    if strip(memname)  eq "&&dset&i" then call symput ('non' || left(_n_),
trim(left(nonel)));
    run;

  %put &&bvar&i;
  %put &i;

  proc sort data=dmold.&&dset&i out=o&&dset&i;
   by %if %index(&&bvar2&i,subjid) %then %do;  &&bvar2&i &&bvar&i; %end;
      %else %do;&&bvar&i; %end;
      %if %index(&&non&i,nonel) %then %do;
       where nonel ne 'NONE'; %end;
  run;

  proc sort data=dmnew.&&dset&i out=n&&dset&i;
  %if %index(&&non&i,nonel) %then %do;
    where nonel ne 'NONE';
  %end;
   by %if %index(&&bvar2&i,subjid) %then %do;  &&bvar2&i &&bvar&i; %end;
      %else %do;&&bvar&i; %end;
  run;

  proc compare data=o&&dset&i compare=n&&dset&i listall novalues out=&&dset&i outnoequal ;
/*outnoequal*/
   id %if %index(&&bvar2&i,subjid) %then %do;  &&bvar2&i &&bvar&i; %end;
      %else %do;&&bvar&i; %end;
  run;

   proc means data = n&&dset&i n nmiss;
   run;

  %end;
  %mend;
  %comp;
```

Below is an example of compare procedure top level dataset summary report with details of the total number of observations and the variables present in the two datasets.[1] In this example the observation count has gone up in the new dataset, as expected with more data but also the total number of variables has gone up as well. The reasons on why this has happened need to be closely looked into.

The variable summary provides information on the number of variables differences found between the two datasets. Here we can see why there are differences on the total variable count and what they are.



The COMPARE Procedure
Comparison of WORK.OADVERSE with WORK.NADVERSE
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.OADVERSE	21AUG14:15:06:32	21AUG14:15:06:32	92	1829
WORK.NADVERSE	21AUG14:15:06:33	21AUG14:15:06:33	93	2038

Variables Summary

Number of Variables in Common: 89.
Number of Variables in WORK.OADVERSE but not in WORK.NADVERSE: 3.
Number of Variables in WORK.NADVERSE but not in WORK.OADVERSE: 4.
Number of Variables with Differing Attributes: 1.
Number of ID Variables: 29.

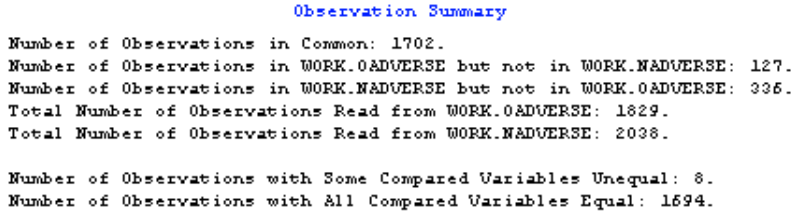
Listing of Variables in WORK.OADVERSE but not in WORK.NADVERSE

Variable	Type	Length	Format	Informat	Label
DBSSITEN	Num	8	10.		Body Site Involvement-DVN
DBSSITEL	Char	200	$200.	$200.	Body Site Involvement-DLU
DBSSITEF	Char	200	$200.	$200.	Body Site Involvement-FUL

Listing of Variables in WORK.NADVERSE but not in WORK.OADVERSE

Variable	Type	Length	Format	Informat	Label
AESITE	Num	8	BODSITE.		AE Site Category
AESITEN	Num	8	10.		Body Site Involvement-DVN
AESITEL	Char	200	$200.	$200.	Body Site Involvement-DLU
AESITEF	Char	200	$200.	$200.	Body Site Involvement-FUL

The observation summary provides information on the number of observation differences found between the two datasets. Any significant drop in observation count from the old data is of potential concern. If needed, we can have a closer look at the observation differences between the two datasets with a merge statement, as shown further below.



Observation Summary

Number of Observations in Common: 1702.
Number of Observations in WORK.OADVERSE but not in WORK.NADVERSE: 127.
Number of Observations in WORK.NADVERSE but not in WORK.OADVERSE: 336.
Total Number of Observations Read from WORK.OADVERSE: 1829.
Total Number of Observations Read from WORK.NADVERSE: 2038.

Number of Observations with Some Compared Variables Unequal: 8.
Number of Observations with All Compared Variables Equal: 1694.

A detail look of the value comparison summary PROC COMPARE provides is of much less value at this stage and can be skimmed through, unless anything odd is noted. If needed, we can add the novalue option to suppress this summary.

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 58.
Number of Variables Compared with Some Observations Unequal: 2.
Number of Variables with Missing Value Differences: 1.
Total Number of Values which Compare Unequal: 8.

Variables with Unequal Values

Variable	Type	Len1	Len2	Label	Ndif	MaxDif	MissDif
AEACTT	CHAR	80	200	Action Treatment Other Text	6		0
AEDECD1	CHAR	100	100	Lowest Level Term (LLT)	2		2

MISSING DATA CHECKS

The number of missing and non-missing values for numeric variables can also be checked by using proc means. Any high number of missing values of concern level can be identified this way.

The SAS System

The MEANS Procedure

Variable	Label	N	N Miss
AESITE	AE Site Category	48	1990
AECLAS	AE Class	2038	0
ACCESSTS	Accessible TS	2038	0
LOGINTS	Login TS	2038	0
LSTCHGTS	LstChg Ts	2038	0
RDCIID	Received DCI Id	2038	0
SUBSETSN	DCM Subset Number	2038	0
QUALIFYQ	Qualifying Question	1990	48

CLOSER LOOK AT CONCERN DATASETS

We can also do individual dataset comparison of only the once that are needed, e.g. when only a few datasets were delivered in the last delivery.

```
/*_____________________________________________________________________________________
/   When only need to run few dataset use below Proc compare
/____________________________________________________________________________________*/
/*
  %let dataset=adverse;
  %let by=subjid aeterm colldate rdciid ;
*/
  proc sort data=dmold.&dataset out=o&dataset;
  by &by;
    where nonel ne 'NONE';
  run;

  proc sort data=dmnew.&dataset out=n&dataset;
    where nonel ne 'NONE';
  by &by;
  run;

  proc compare data=o&dataset compare=n&dataset listbase out=a&dataset outnoequal;
  id &by;
  run;
```

Where needed, a further merge allows for a closer look at the concern datasets with tailored key by variables identified. This will identify any concern observations that were in one dataset but are missing from the other.

```
/*List observations that were in the old data but missing from the new data*/

  data both_obs inold_obs innew_obs;
   merge o&dataset(in=a) n&dataset(in=b);
   by &by;
   if a and b then output both_obs;
   if a  and not b then output inold_obs;
   if b and not a then output innew_obs;
  run;

  data inobs;
   set inold_obs(in=a) innew_obs(in=b);
   by &by;
   if a then in='inold_obs';
   if b then in='innew_obs';
  run;
```

Subject ID	Lowest Level Term (LLT)	AE Investigator Text - Raw - MedDRA	Adverse Event From Date	Adverse Event To Date	IN
0011014		ARM ABRASIONS	09JUL2014	.	inold_obs
0011014	Abrasions	LEFT ARM ABRASIONS	09JUL2014	05AUG2014	innew_obs

Figure 4: Example of output showing the observation differences between the old and new data

In this example above, we can see that the old AE term has been updated with additional AE details and has acceptable level differences in the new data, and it is not of any data concern.

CONCLUSION

When expecting a newly extracted data delivery, the old delivery data need to be saved to a different folder, before copying the new data, and then both of these folders can be referenced via libnames. We can then use SAS dictionary dataset to reference these two libraries dataset metadata details. A program can be written to cross check all raw datasets, variables and observations that are delivered each time. The advantages of doing checks programmatically are that it is more reliable, minimizes human error and less time consuming in the long run.

This programmatic approach will help us to identify most of the differences, including: any differences in the variables or datasets in the current delivery to what was delivered previously, if significant drop in number of observations from the old data, any common pattern differences seen across the whole dataset, if all observations for a variable have missing records. Once these are checked and all of the differences are accounted for, one can be of more confident in the consistency of the data being delivered.

REFERENCES

1. http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm#n0c1y14wyd3u7yn1dmfcpaejllsn.htm

ACKNOWLEDGMENTS

I would like to thank Lynn Clipstone, Simon Jennings, Jennie McGuirk and Mark Crangle for their help and support in producing this document.

CONTACT INFORMATION

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

Thevaki Mahadevan
ICON Clinical Research UK Ltd
2 Globeside Business Park
Marlow
Buckinghamshire
United Kingdom
SL7 1HZ
Email: Thevaki.Mahadevan@iconplc.com