SESUG Paper IACS60-2025

Obtaining Metadata Information on Data Transfers for Dataset Profiling

Amy Zhang, Chen Wang, and Jeff Xia, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

During a clinical trial, it is common to extract data from a live database for various purposes, such as medical monitoring, TLFs development, or evaluation of the safety profile for an investigational drug. These data extractions can happen multiple times throughout a trial lifecycle, and it is beneficial for statistical programmers to understand the overall context of the data at each data transfer. Some particular questions we want to answer are how many datasets were included in this transfer and were there any missing or new datasets? Have there been changes to the dataset structure? Were any variables or their controlled terms introduced or removed? Was new value-level metadata added to the finding domains? Due to the interdependent nature of SDTM and ADaM codes, without awareness of all the updates, such changes can result in potential mapping errors that impact ADaM programming or the contents of the final outputs. Analyzing all these aspects manually can be time-consuming. To address this problem, we developed a utility tool to check the contents of SDTM datasets and produce a summary report of differences between the newest delivery and previous deliveries. We assembled a customized list of common changes that we expect to occur or need to pay close attention to across the SDTM domains, and implemented SAS® macros to automatically summarize and output the information we are interested in.

INTRODUCTION

As statistical programmers, we frequently encounter multiple data extractions or data transfers when working on a study. Because we are using a live database, updates to data can happen at any time. When we extract and use the most recent versions of data, it can cause unexpected issues during SDTM or ADaM dataset development if we are not fully aware of the changes and how they affect our downstream programming codes. There can be major changes like the addition of new datasets or missing some critical datasets. Other changes might involve the addition or removal of variables, changes to variables' controlled terms, and new or removed data points that impact programming. Such changes will require mapping adjustments when generating SDTM or ADaM datasets.

It is difficult to predict where these updates will occur beforehand, so we would like to have some method to profile datasets and check for changes. Attempting to do this by hand for each study is too time-consuming. Our solution is to develop a utility tool incorporating user-developed SAS macros to automate these checking steps. We want to extract and summarize key information of interest from specific SDTM domains, and produce a summary report with the profiled information as well as documentation of any changes between the latest data transfer compared to previous transfers.

This paper follows steps to introduce and explain the SAS macros as utility tools to profile datasets and keep track of changes among multiple data transfers. First, it covers a range of predetermined criteria for profiling, and is not limited to the subject-level demographic information, variables of user interest, records of missing MedDRA or WHODRUG coding, etc. Next, the specific programming procedures that correspond to each criteria is outlined in detail. Lastly, the comparison results are generated using a previously developed macro.

1. DETERMINE KEY INFORMATION FOR PROFILING

Because it would be impossible and inefficient to summarize all the variables from every dataset, we first defined the key information that we want to profile from a data transfer. This is not an exhaustive list, but the advantage of creating a checking tool is its flexibility for study teams to add new comparison criteria or remove irrelevant ones to fit their study's needs.

For the purpose of our tool, the main issues we decided to pay attention to are:

- Metadata of all SDTM datasets received, including domain name, dataset label, and number of observations in the dataset
- Metadata of all variables from SDTM datasets received, including variable name, variable label, and variable length
- List of QNAM, QLABEL, and ORIGIN for all SUPP- domains
- List of EPOCH, -TPT, and -TPTNUM for all domains where applicable
- List of -GRPID for all domains where applicable (-GRPID in SDTM domains is used to record CRF form names in the operational database, any addition or removal would mean potential mapping changes)
- List of ARM and corresponding ARMCD, ACTARM and corresponding ACTARMCD from DM domain
- List of COUNTRY, SITE, and INVESTIGATOR NAME from DM domain
- List of EXTRT, EXDOSE, EXDOSFRM, and EXROUTE from EX domain
- List of VISIT and corresponding VISITNUM from SV domain
- List of -CAT, -TEST, -TESTCD, and -STRESU from Finding domains where applicable
- Any records with missing MedDRA coding from AE domain or missing WHODRUG coding from CM domain
- Any inconsistent death information across DD, DS, DM, and AE domains

2. EXTRACT METADATA BASED ON PREDEFINED CRITERIA

Within each of the criteria, we want to automate, or “macro-tize”, as much as possible to minimize the number of user-specific edits while still accounting for therapeutic area or study-level differences.

We start by maintaining a list of SDTM domains available from the data transfer. We use a DATA step with filename(), dopen(), dnum(), and dread() functions to find all files in “.sas7bdat” format under a given folder path. We then use PROC SQL to go through and add all the files’ names into a macro variable. A similar strategy can be used to subset for all SDTM domains that begin with “SUPP-” and save only the SUPP- domains into a separate macro variable.

```
data sdtmnames;
    length fref $8 fname $200;
    did = filename(fref, "&protpath/datasdtm");
    did = dopen(fref);
    do i=1 to dnum(did);
        fname = dread(did,i);
        if find(fname, ".sas7bdat") then output;
    end;
    did = dclose(did);
    keep fname;
run;

proc sql noprint;
    select substr(fname, 1, find(fname,".")-1) into :sdtmlist separated by ' ' from sdtmnames;
quit;
```

METADATA OF ALL SDTM DATASETS AND VARIABLES

First, we use a %DO loop to loop through all the SDTM domains. Within each domain, PROC CONTENTS can help to get the domain name and label; variable name, label, and length; and number of observations.

```
%do i=1 %to &sdtmnum.;
    %let sdtm = %sysfunc(upcase(%sysfunc(strip(%sysfunc(scan(&sdtmlist., &i, ' '))))));
    proc contents data=lptss.&sdtm. out=&sdtm._info
    (keep=MEMNAME MEMLABEL NOBS NAME LABEL LENGTH
    rename=(MEMNAME=DOMAIN MEMLABEL=DSLABEL NAME=VARNAME LABEL=VARLABEL)) noprint; run;
```

STUDY-LEVEL INFORMATION FROM SPECIFIC DOMAINS

Now, we want to extract specific information from a subset of domains based on our predefined profiling list. In order to obtain QNAMs and their corresponding QLABELs and ORIGINs, we use a similar %DO loop to loop through only the SUPP- domains and call PROC SQL to output a list of non-missing distinct values.

```
%do i=1 %to &suppnum.;
    %let supp = %sysfunc(upcase(%sysfunc(strip(%sysfunc(scan(&supplist., &i, ' '))))));
    data &supp.;
        length domain $8;
        set lptss.&supp.;
        DOMAIN = %sysfunc(upcase("&supp."));
    run;
    proc sql;
        create table &supp._sum as
        select distinct domain, qnam, qlabel, qorig from &supp.
        where not missing(qnam);
    quit;
```

A challenge we face with getting a list of EPOCH, -TPT and -TPTNUM, and -GRPID by SDTM domain is that these variables are found in some, but not all, domains. Another challenge is that -TPT, -TPTNUM, and -GRPID have the corresponding domain as a prefix in their variable name, like “LBTPT” or “DSGRPID” for example, but EPOCH does not. This means we need to take additional considerations into account when automating this search strategy. We still use a %DO loop to loop through all the SDTM domains, but we first check if these particular variables exist in each domain. Only if they exist and are valued do we use PROC SQL to output a list of non-missing distinct values.

```
                data &var.&i.;
                    length searchvar $20 sortvar $20;
                    dsid = open("&sdtm.");
                    if upcase("&var.")='EPOCH' then do;
                        searchvar = 'EPOCH';
                        check = varnum(dsid, searchvar);
                        sortvar = upcase("&sort.");
                    end;
                    else do;
                        searchvar = upcase("&sdtm.")||upcase("&var.");
                        check = varnum(dsid, searchvar);
                        %if %length(&sort)>0 %then %do;
                        sortvar = upcase("&sdtm.")||upcase("&sort."); %end;
                    end;
                    rc = close(dsid);
                run;

    proc sql;
        create table &var._&sdtm. as
        select distinct domain, &searchvar. as &var. %if %length(&sort)>0 %then %do;,&sortvar as &sort. %e
        from &sdtm.
        where not missing(&searchvar.)
        %if %length(&sort)>0 %then %do; order by &sortvar %end;;
    quit;
```

We can also use PROC FREQ to help us summarize data. One example is calling PROC FREQ on the DM domain and combine with the TABLES statement to show the cross-tabulation of SITEID, INVID, and INVNAM variables in list form.

```
/*DM: SITE/Investigator/Investigator Name*/
proc freq data=lptss.dm noprint;
    tables SITEID*INVID*INVNAM/missing list out=dm_site(drop=count percent where=(not missing(SITEID)));
run;
```

To check for inconsistent death information, we merge DD, DS, DM, and AE domains together, while only retaining the records that capture relevant death information. Then, we can compare the death date from each of the domains and output any that are inconsistent into a WARNING variable.

```
proc sort data=lptss.dm out=dm(keep=usubjid dthdtc); by usubjid; where not missing(dthdtc); run;
proc sort data=lptss.dd out=dd(keep=usubjid dddtc); by usubjid; where not missing(dddtc); run;
proc sort data=lptss.ds out=ds(keep=usubjid dsstdtc); by usubjid; where find(upcase(dsdecod), 'DEATH');
proc sort data=lptss.ae out=ae(keep=usubjid aeendtc); by usubjid; where aesdth='Y'; run;

data deaths_warning;
    length WARNING $800 DOMAIN $20;
    merge dm dd ds ae;
    by usubjid;
    if not missing(aeendtc) and dthdtc = dddtc = dsstdtc = aeendtc then dthfl='Y';
    else if missing(aeendtc) and dthdtc = dddtc = dsstdtc then dthfl='Y';

    if dthfl ne 'Y' and not missing(aeendtc) then do;
        output;
        WARNING = cat("Check inconsistent death dates for subject: ", usubjid, ", DTHDTC: ", dthdtc, ",
        DOMAIN = "DM, DD, DS, AE";
    end;
    else if dthfl ne 'Y' and missing(aeendtc) then do;
        output;
        WARNING = cat("Check inconsistent death dates for subject: ", usubjid, ", DTHDTC: ", dthdtc, ",
        DOMAIN = "DM, DD, DS";
    end;
    keep DOMAIN usubjid WARNING;
run;
```

As we extract the metadata from each profiling criteria, the results get outputted into its own new dataset. Doing so will help us when creating our summary report and performing comparison between two data transfers.

OUTPUT RESULTS TO SUMMARY REPORT

We use ODS steps to create an Excel summary report. In particular, ODS EXCEL FILE specifies the name and path of the output file, ODS EXCEL OPTIONS customizes the worksheet's name, and PROC PRINT adds the dataset created from each criteria to the summary report. ODS EXCEL OPTIONS gets repeatedly called for as many times as the number of result datasets we have, and within this step, the AUTOFILTER option will add filters on all columns.

```
    ods excel file=&outfile.;

    ods excel options(embedded_titles='yes' sheet_name="DOMAIN" autofilter='yes');
        proc print data=lptuser._1_allsdtmdom_info label; run;

    ods excel options(embedded_titles='yes' sheet_name="VARIABLE" autofilter='yes');
        proc print data=lptuser._2_allsdtmvar_info label; run;

    ods excel options(embedded_titles='yes' sheet_name="QNAM/QLABEL" autofilter='yes');
        proc print data=lptuser._3_suppvar_info label; run;
```

Below is an example from the summary report. Each tab corresponds to profiling criteria listed in this paper.

O	Country	Country Name
1	ARG	Argentina
2	AUS	Australia
3	BEL	Belgium
4	CAN	Canada
5	CHE	Switzerland
6	CHL	Chile
7	COL	Colombia
8	DEU	Germany
9	ESP	Spain
10	FRA	France
11	GBR	United Kingdom
12	ISR	Israel
13	JPN	Japan
14	KOR	Korea, Republic of
15	MEX	Mexico
16	NLD	Netherlands, Kingdom of the
17	TUR	Turkiye

DOMAIN | QNAM-QLABEL | EPOCH | TPT-TPTNUM | GRPID | TREATMENT | COUNTRY | SITE-INV | VISIT | LB | LC | IE-CO | DECOD

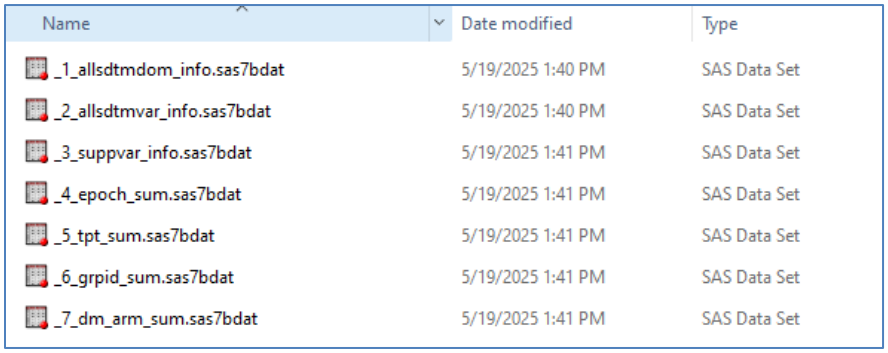
3. PERFORM COMPARISON AND PRODUCE COMPARISON REPORT

To compare the metadata profiling results between two data transfers, we use a macro, **%compare_dataset**, previously developed by Jeff Xia et al. Details can be found in the referenced paper,

and similar logic of automatically comparing contents of two datasets based on key sorting variables can be replicated by a study team.

ADD CALL PROGRAM PARAMETERS INTO "METADATA" DATASET

In order to reuse the **%compare_dataset** macro, the key sorting variables are required as input. Thus, we generate a metadata dataset to save the domain name and its sorting variables. This dataset becomes part of the input for the comparison macro. We also need to retrieve and read in the names of all resulting profiling datasets from the output folder of a data transfer. We execute this step twice, once for the current data transfer and again for the previous transfer. Afterwards, the two datasets and the metadata dataset are merged.



Name	Date modified	Type
_1_allsdtmdom_info.sas7bdat	5/19/2025 1:40 PM	SAS Data Set
_2_allsdtmvar_info.sas7bdat	5/19/2025 1:40 PM	SAS Data Set
_3_suppvar_info.sas7bdat	5/19/2025 1:41 PM	SAS Data Set
_4_epoch_sum.sas7bdat	5/19/2025 1:41 PM	SAS Data Set
_5_tpt_sum.sas7bdat	5/19/2025 1:41 PM	SAS Data Set
_6_grpid_sum.sas7bdat	5/19/2025 1:41 PM	SAS Data Set
_7_dm_arm_sum.sas7bdat	5/19/2025 1:41 PM	SAS Data Set

```
data pre;
    length fref $8 _fname $200;
    did = filename(fref, "&lptpre");
    did = dopen(fref);
    do i=1 to dnum(did);
        _fname = dread(did,i);
        if find(_fname, ".sas7bdat") and _fname ne 'metadata.sas7bdat' then output;
    end;
    did = dclose(did);
    keep _fname;
run;
```

CALL COMPARISON MACRO TO GENERATE COMPARISON REPORT

The following information is required to pass into the **%compare_dataset** macro:

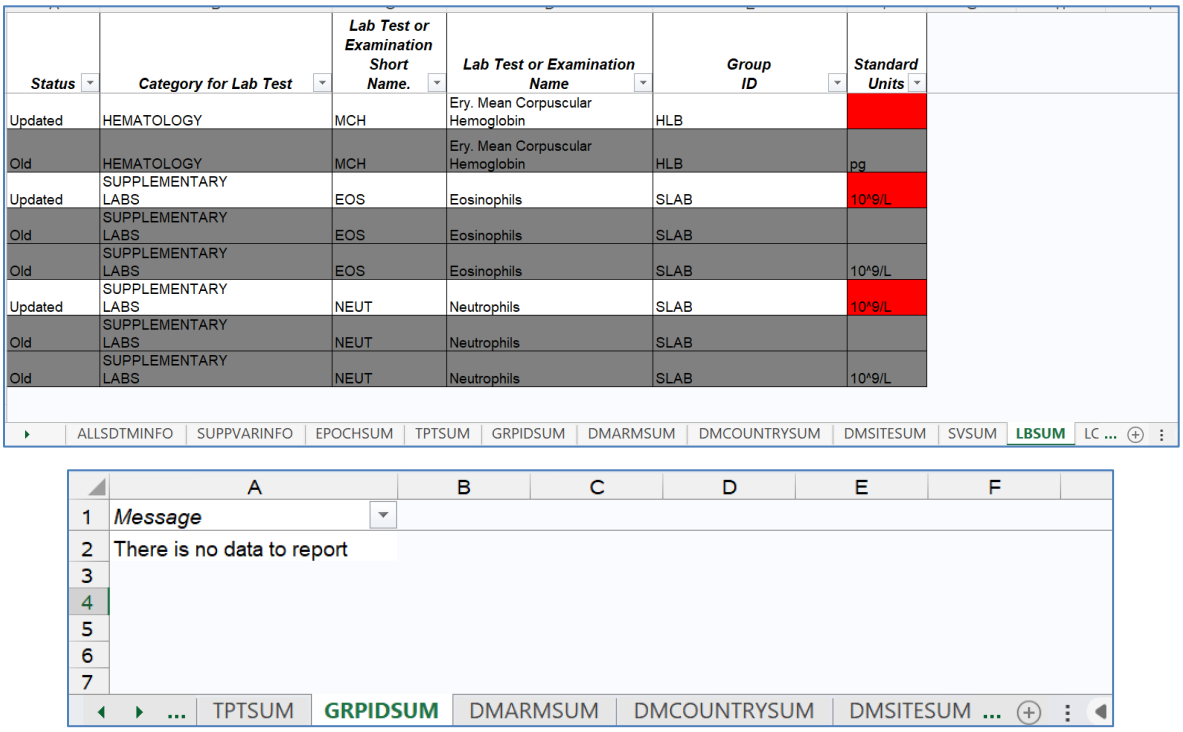
- Paths to current (new) and previous (old) datasets' folders
- The dataset to be compared, which we saved inside a macro variable
- The key sorting variables for each dataset
- The tab name to put in the worksheet of the output comparison report

```
%let j=1;
%do %while (&j<=&num.);
    %let summary = %sysfunc(upcase(%sysfunc(strip(%sysfunc(scan(&comparelist., &j, ' '))))));
    %let keyvar = %sysfunc(strip(%sysfunc(tranwrd(%sysfunc(upcase(%sysfunc(strip(%sysfunc(scan(&key
    %let tabname = %sysfunc(compress(&summary,%str( ),%str(ak)));

    %compare_dataset(newlib=lptnew,
                     oldlib=lptold,
                     ds = &summary,
                     compds = &summary,
                     keyvars= &keyvar,
                     rptname= &tabname,
                     except =,
                     sample_size= 1,
                     sampling_id=,
                     sampling_lib=,
                     sampling_ds=,
                     debug = N);

    %data2xml(stage = 2, indsn=lptuser.final, sheetname=&tabname);

    %let j = %eval(&j + 1);
```

The complete explanation of the **%compare_dataset** macro was presented in the "A Vivid and Efficient Way to Highlight Changes in SAS Dataset Comparison". In this paper, we reused this macro without any

updates. We called this macro inside a %DO loop to automatically repeat the comparison step for as many datasets as we have; this flexibility can be handled by saving the number of inputs into a macro variable &num.

The screenshot below is an example of a comparison result. This specific example shows updates to the LB domain between the previous and current data transfer. The status “Updated” reflects the updated information in “Standard Units”, which is also highlighted in red. Alternatively, if information in a particular dataset remains unchanged between data transfers, the message “There is no data to report” is displayed in that tab.



Status	Category for Lab Test	Lab Test or Examination Short Name.	Lab Test or Examination Name	Group ID	Standard Units
Updated	HEMATOLOGY	MCH	Ery. Mean Corpuscular Hemoglobin	HLB	
Old	HEMATOLOGY	MCH	Ery. Mean Corpuscular Hemoglobin	HLB	pg
Updated	SUPPLEMENTARY LABS	EOS	Eosinophils	SLAB	10^9/L
Old	SUPPLEMENTARY LABS	EOS	Eosinophils	SLAB	
Old	SUPPLEMENTARY LABS	EOS	Eosinophils	SLAB	10^9/L
Updated	SUPPLEMENTARY LABS	NEUT	Neutrophils	SLAB	10^9/L
Old	SUPPLEMENTARY LABS	NEUT	Neutrophils	SLAB	
Old	SUPPLEMENTARY LABS	NEUT	Neutrophils	SLAB	10^9/L

ALLSDTMINFO | SUPPVARINFO | EPOCHSUM | TPTSUM | GRPIDSUM | DMARMSUM | DMCOUNTRYSUM | DMSITESUM | SVSUM | LBSUM | LC ...

	A	B	C	D	E	F
1	Message					
2	There is no data to report					
3						
4						
5						
6						
7						

... | TPTSUM | GRPIDSUM | DMARMSUM | DMCOUNTRYSUM | DMSITESUM ...

CONCLUSION

The utility tools demonstrated in this paper, along with integration with a previously developed comparison macro, allow for a comprehensive overview of study data extracted from a live database during an ongoing clinical trial. It is beneficial to not only be aware of changes to a study’s metadata, but also to plan ahead for the downstream programming impacts. Our solution can be broken down to two main parts. The first part is to generate a summary report to show the profiled metadata based on our predefined data checking criteria; this helps study programmers quickly gain high level understanding of their data without needing to go through all datasets. The second part is to generate a comparison report to track the differences in metadata between two data transfers done at different points in time; this streamlines the comparison process through automation of a manual task and creation of a results file for easy visualization. Together, these two reports can improve the efficiency of data review.

REFERENCES

- Xia, J., L. Xie, and S. Zhao. 2017. “A Vivid and Efficient Way to Highlight Changes in SAS Dataset Comparison” Proceedings of PharmaSUG 2017. https://www.lexjansen.com/pharmasug/2017/AD/PharmaSUG-2017-AD04.pdf

ACKNOWLEDGMENTS

The author would like to thank the management teams at Merck & Co., Inc., Rahway, NJ, USA for their advice on this paper/presentation.

CONTACT INFORMATION

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

Amy Zhang
Merck & Co., Inc., Rahway, NJ, USA
Email: amy.zhang2@merck.com

Chen Wang
Merck & Co., Inc., Rahway, NJ, USA
Email: chen.wang16@merck.com

Jeff Xia
Merck & Co., Inc., Rahway, NJ, USA
Email: jeff.xia@merck.com