PharmaSUG 2024 - Paper SD- 266

A Tool for Automated Comparison of Core Variables Across ADaM Specifications Files

Amy Zhang and Huei-Ling Chen, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

A specifications file serves as fundamental guidance for creating CDISC Analysis Dataset Model (ADaM) datasets. Dataset variables and their associated attributes in the specifications file have predefined standards. In particular, the core attribute is essential; per CDISC IG, the Core column defines whether a variable is required, conditionally required, or permissible. When preparing the ADaM specifications for a study, the programmer frequently encounters the task of ensuring the core variable categorization follows ADaMIG standards, company standards, or other protocols of the same compound or indication. To ease this process, having a tool to quickly compare the list of core variables across standards or studies can help facilitate ADaM dataset preparation. Storing these specifications in an Excel file is a common approach. Some existing software tools provide spreadsheet comparisons, but the findings are often overwhelming to digest. This simple SAS macro provides programmers with a quick glimpse into differences in core variable categorizations across multiple Excel files based on user-specified ADaM datasets and selection criteria. The results are summarized in an Excel file format, with each ADaM dataset as its own spreadsheet presenting the list of core variables across standards or studies.

INTRODUCTION

Creating an accurate ADaM specifications file for a study is a critical part of programmers' workflow. This file becomes the reference for the development of ADaM datasets and their variables. Specifications provide essential information on variables' attributes, for example, their label, type, controlled terms, core, derivation rules, etc, as seen in Figure 1. We focus on the core column: each ADaM variable is classified into one of three 'core' categories - Required, Conditionally Required, and Permissible. A Required variable must be included in the dataset; a Conditionally Required variable must be included if the condition holds true; a Permissible variable does not need to be included. These 'core' categorizations are globally defined for common variables in ADaMIG. There may be other categorizations to consider, like company standards for a specific therapeutic area or standards from prior protocols of the same study compound or indication. The programmer is responsible for verifying that each variable in the study is correctly classified according to (potentially) multiple standards, and in particular, that all Required variables are present in the study's specifications.

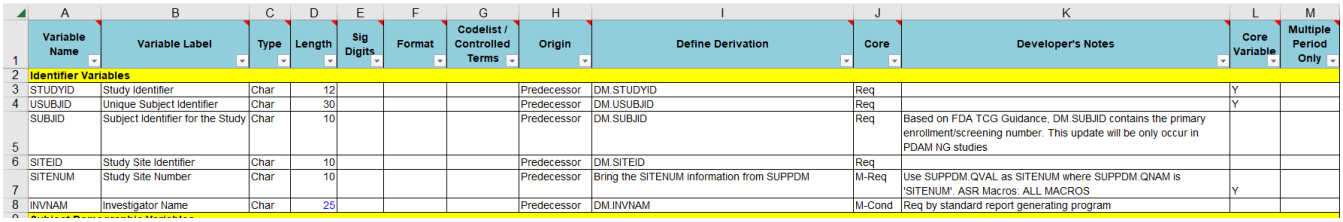
Table 3.2.2 ADSL Subject Demographics Variables

Variable Name	Variable Label	Type	Codelist/ Controlled Terms	Core	CDISC Notes
AGE	Age	Num		Req	DM.AGE. If analysis needs require a derived age that does not match DM.AGE, then AAGE must be added
AGEU	Age Units	Char	(AGEU)	Req	DM.AGEU
AGEGRy	Pooled Age Group y	Char		Perm	Character description of a grouping or pooling of the subject's age for analysis purposes. For example, AGEGR1 might have values of "<18", "18-65", and ">65"; AGEGR2 might have values of "Less than 35 y old" and "At least 35 y old".
AGEGRyN	Pooled Age Group y (N)	Num		Perm	Numeric representation of AGEGRy. Orders the grouping or pooling of subject age for analysis and reporting. There must be a one-to-one relationship between AGEGRyN and AGEGRy within a study. AGEGRyN cannot be present unless AGEGRy is also present. When AGEGRy and AGEGRyN are present, then on a given record, either both must be populated or both must be null.
AAGE	Analysis Age	Num		Cond	Age used for analysis that may be derived differently than DM.AGE. AAGE is required if age is calculated differently than DM.AGE.
SEX	Sex	Char	(SEX)	Req	DM.SEX.
RACE	Race	Char	(RACE)	Req	DM.RACE.
RACEGRy	Pooled Race Group y	Char		Perm	Character description of a grouping or pooling of the subject's race for analysis purposes.

Figure 1: Snapshot of specifications for ADSL dataset from ADaMIG v1.3

Typically, specifications files are Excel files, in either xlsx or xlsm format like Figure 2. Specifications following updated global standards from the most recent ADaMIG versions or therapeutic area standards can be stored as unique template files for easy reference and use. Thus, a programmer can have multiple template files that need to be compared against a study deliverable's specifications. Additionally, in many instances, studies run across a long period of time spanning many months or years; when carrying specifications over from deliverable to deliverable, programmers will need to check that the study's specifications are still up to date with global standards. Current software tools, ie. Synkronizer, perform

comparisons across entire Excel worksheets, but they may require long runtime, and their results are often overwhelming, too detailed, and beyond what the programmer is looking for.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Variable Name	Variable Label	Type	Length	Sig Digits	Format	Codelist / Controlled Terms	Origin	Define Derivation	Core	Developer's Notes	Core Variable	Multiple Period Only
2	Identifier Variables												
3	STUDYID	Study Identifier	Char	12				Predecessor	DM.STUDYID	Req		Y	
4	USUBJID	Unique Subject Identifier	Char	30				Predecessor	DM.USUBJID	Req		Y	
5	SUBJID	Subject Identifier for the Study	Char	10				Predecessor	DM.SUBJID	Req	Based on FDA TCG Guidance, DM.SUBJID contains the primary enrollment/screening number. This update will be only occur in PDAM NG studies		
6	SITEID	Study Site Identifier	Char	10				Predecessor	DM.SITEID	Req			
7	SITENUM	Study Site Number	Char	10				Predecessor	Bring the SITENUM information from SUPPDM	M-Req	Use SUPPDM.QVAL as SITENUM where SUPPDM.QNAM is 'SITENUM'. ASR Macros: ALL MACROS	Y	
8	INVNAM	Investigator Name	Char	25				Predecessor	DM.INVNAM	M-Cond	Req by standard report generating program		

Figure 2: Global specifications from ADaMIG stored in an Excel file

This simple tool gives a targeted glimpse to help programmers confirm that their study's specifications capture all Required variables and correct core categorizations based on available template specifications. The user has flexibility to select which specification files to be used for comparison, the ADaM datasets of interest, and the subsetting criteria. The tool summarizes the findings in an outputted Excel file, allowing the user to identify and resolve any potential inaccuracies efficiently.

MACRO DESCRIPTION AND USER INPUTS

The macro, **%compare0adam0specs**, takes six input parameters and outputs one final Excel file. The macro is written as:

```
%macro compare0adam0specs(
        adam_specs_files=,
        adam_specs_tags=,
        sheetlist=,
        column=,
        condition=,
        output_excel_file =
);
```

where

- adam_specs_files: pathnames to the locations where Excel specifications files are stored, entered as a list separated by |
- adam_specs_tags: selected name for specifications' source (ie. global, ta, study-protocol, etc), entered as a list separated by |. The number of tags must match the number of specifications files being used
- sheetlist: ADaM datasets of interest to be compared (ie. ADSL, ADAE, etc), entered as a list separated by |. Individual Excel worksheets within the specifications file should be named according to the ADaM dataset names
- column: column of interest from specifications file (ie. Core)
- condition: subsetting condition of interest (ie. (where not missing(Core)))
- output_excel_file: pathname to the location where comparison summary will be exported

An example macro call will look like:

```
%compare0adam0specs(
        adam_specs_files=%str(/path/Global-template-specs.xlsm|
                              /path/TA-template-specs.xlsm|
                              /path/Study-specs.xlsm),
        adam_specs_tags=%str(global|ta|study),
        sheetlist=%str(adsl|adae|adrs),
        column=%str(core),
        condition=%str(where core in ("Req", "M-Req", "O-Req")),
```

```
    output_excel_file =%str(/path/Output-results.xlsx)
);
```

MACRO PROGRAM FLOW

This macro follows 3 main steps:

1. Read in Excel specifications files as SAS datasets

From the specifications files, the macro has a check to convert the file from xlsm to xlsx if needed:

```
%if %upcase(&extension)=XLSM %then %do;
        %if "&sysscp" = "WIN" %then %let cmd = copy &file_path.\&file_name.
&file_path.\xxxxxx_.xlsx  /b ;
        %else %let cmd = cp &file_path./&file_name.  &file_path./xxxxxx_.xlsx;
%end;
```

The macro parses through each worksheet (an ADaM dataset's specifications) of each file (global specs, TA-specific specs, study's specs, etc) via a nested do-loop:

```
%let filenum=%eval(%sysfunc(countc(&adam_specs_files, '|' ))+1);
%let tagnum=%eval(%sysfunc(countc(&adam_specs_tags, '|' ))+1);
%do i=1 %to &filenum;
        %let sheetnum = %eval(%sysfunc(countc(&sheetlist., '|' ))+1);

        %do j=1 %to &sheetnum;
```

Each worksheet gets read in as a SAS dataset using the PROC IMPORT procedure:

```
proc import out=&sheet._&tag.
datafile="&file."
dbms=xlsx replace;
sheet="&sheet.";
getnames=YES;
run;
```

The SAS dataset is subsetted based on the user-entered condition. The macro also includes a check that the ADaM datasets have existing specifications within each file; it continues to run via an empty dataset placeholder if the ADaM dataset specifications are missing from the specifications file(s):

```
%if %sysfunc(exist(&sheet._&tag.)) %then %do;
        data &sheet._&tag;
                set &sheet._&tag (rename=(Variable_Name=varnam));
                &condition.;

        run;
%end;
%else %do;
        data &sheet._&tag.;
                dataset='';
                Variable_Name='';
                &tag.='';
        run;
%end;
```

2. Perform comparison between specifications within each ADaM dataset

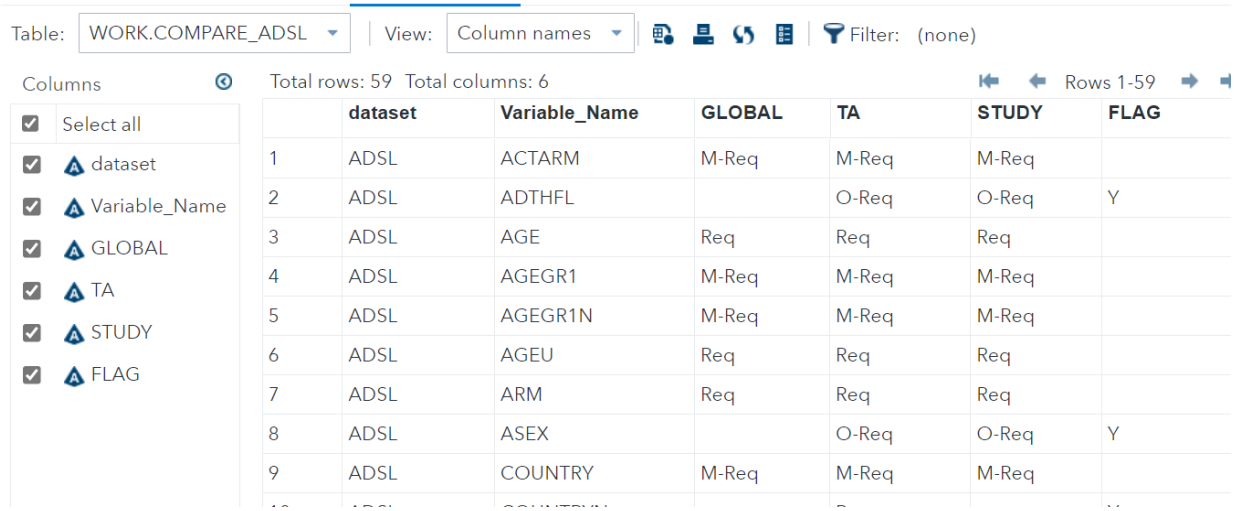
The macro combines data from the specifications datasets to create a grouped comparison dataset for each ADaM dataset. Using another nested do-loop, the SET statement sets the first specifications dataset and the subsequent MERGE statement automatically joins those with the same ADaM dataset.

A flag variable is also created during the loop to make any differences more readily identifiable. Once a difference is spotted between any two specifications, the flag gets assigned the value 'Y'. To carry out this task, there is an initial comparison for the first specifications with the second specifications when merging the datasets. If the value between the two datasets are different, it assigns 'Y' to a flag variable; otherwise, the flag has a null value, meaning that the values are the same. The loop continues to compare the second specifications with the third specifications. If the values between the first and second specifications were the same, but the values between the second and third specifications are different, it now assigns 'Y' to the flag variable. This process continues until all specifications are reviewed:

```
%do j=1 %to &sheetnum;
        %let sheet=%sysfunc(scan(&sheetlist., &j, '|'));

        %do i=1 %to &tagnum;
                %let tag=%sysfunc(scan(&adam_specs_tags, &i, '|'));
                %if &i=1 and %sysfunc(exist(&sheet._&tag.)) %then %do;
                        data compare_&sheet.;
                                set &sheet._&tag.;
                                original=&tag.;
                                origflag='';
                        run;
                %end;
                %else %if &i~=1 and %sysfunc(exist(&sheet._&tag.)) %then %do;
                        data compare_&sheet.;
                                merge compare_&sheet. &sheet._&tag.;
                                by dataset Variable_Name;
                                if missing(Variable_Name) then delete;
                                if original ne &tag. then flag_='Y';
                                if origflag='Y' or (origflag=' ' and original
ne &tag. ) then flag_='Y';
                                origflag = flag_;
                        run;
                %end;
        %end;
%end;
```

Figure 3 shows an example output of the comparison loops for the ADSL dataset using three specifications files. If the values are the same across all specifications, the flag variable remains empty. However, if any of the values are different or missing, the flag gets populated with 'Y'.



	dataset	Variable_Name	GLOBAL	TA	STUDY	FLAG
1	ADSL	ACTARM	M-Req	M-Req	M-Req	
2	ADSL	ADTHFL		O-Req	O-Req	Y
3	ADSL	AGE	Req	Req	Req	
4	ADSL	AGEGR1	M-Req	M-Req	M-Req	
5	ADSL	AGEGR1N	M-Req	M-Req	M-Req	
6	ADSL	AGEU	Req	Req	Req	
7	ADSL	ARM	Req	Req	Req	
8	ADSL	ASEX		O-Req	O-Req	Y
9	ADSL	COUNTRY	M-Req	M-Req	M-Req	

Figure 3: SAS dataset generated by macro for one ADaM dataset's comparison

3. Export results to one Excel file

The macro creates one xlsx file at the user-specified path. Inside the xlsx file, each ADaM dataset has its own worksheet:

```
%if "&output_excel_file." ne "" %then %do;
        libname final xlsx "&output_excel_file.";
        proc datasets;
                copy in=work out=final;
                select compare:;
        run;
        quit;
        libname final clear;
%end;
```

As shown in Figure 4, the individual worksheet tabs are labeled with the ADaM dataset and its associated comparison. The worksheet lists the ADaM dataset where the variable came from (Column A), the variable name (Column B), the 'core' categorization values from each respective specification file source (Columns C-E), and a flag to easily display which variables have different values between the specification files (Column F).

	A	B	C	D	E	F
1	dataset	Variable_Name	GLOBAL	TA	STUDY	FLAG
2	ADSL	ACTARM	M-Req	M-Req	M-Req	
3	ADSL	ADTHFL		O-Req	O-Req	Y
4	ADSL	AGE	Req	Req	Req	
5	ADSL	AGEGR1	M-Req	M-Req	M-Req	
6	ADSL	AGEGR1N	M-Req	M-Req	M-Req	
7	ADSL	AGEU	Req	Req	Req	
8	ADSL	ARM	Req	Req	Req	
9	ADSL	ASEX		O-Req	O-Req	Y
10	ADSL	COUNTRY	M-Req	M-Req	M-Req	
11	ADSL	COUNTRYN		Req		Y
12	ADSL	DCEFURS		O-Req		Y
13	ADSL	DCTADY		O-Req		Y
14	ADSL	DCTDT		O-Req		Y
15	ADSL	DCTFL		O-Req		Y

COMPARE_ADAE | COMPARE_ADRS | **COMPARE_ADSL**

Figure 4: Excel file generated by macro

CONCLUSION

Comparing information across multiple ADaM specification files can be tedious and time-consuming. This paper presents a solution that automates the comparison process based on what information the user wants to see and the entered selection criteria. The detected differences are efficiently summarized in one file, allowing for quick comprehension of results in a glance.

Because this macro is simple to digest and understand, it will be easy to adapt according to each company's setup, for example, modifying the file read-in portion of the code to accommodate SAS datasets if specifications are stored in SAS datasets instead of Excel files. While this macro performs a straightforward function, its' user flexibility allows for expansions to other use cases. The user can change the subsetting condition or select a different column to compare, like comparing the variables' labels across the available specifications files. Specifications are also needed and created for SDTM datasets, and the user can use SDTM specification files as the inputs into this tool, given that the SDTM specifications follow the same column structure and format as ADaM specifications.

One potential area of difficulty arises when the user is interested in comparing variables' derivation rules across the specifications files. Derivations extend to long lines of text, and rules that instruct the

programmer to perform the same derivation can be written in different ways. Because this tool can only handle exact matches, results generated under this circumstance may be misleading. Thus, future expansion of this macro requires the integration of AI or machine learning algorithms to correctly identify differences between derivation rules.

REFERENCES

CDISC ADaM Guidelines

- ADaM | CDISC

SAS Macro References

- SAS Macro Language 1 Essentials

ACKNOWLEDGMENTS

The authors 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 authors at:

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

Huei-Ling Chen
Merck & Co., Inc., Rahway, NJ, USA
E-mail: huei-ling_chen@merck.com