SESUG Paper 259-2019

One Click to Analysis Results Metadata

Srivathsa Ravikiran and Priscilla Gathoni, AstraZeneca Pharmaceuticals LP

ABSTRACT

Analysis Results Metadata (ARM) is a plugin to ADaM Define.xml which provides information on Analysis data used to generate efficacy, safety and clinical study results. It establishes one to one relationship between the analysis result, ADaM data and variables used to perform the analysis. ARM plug-in benefits the reviewer by providing clear outline of the critical analysis displays, details on the analysis performed and methods used. Due to these added benefits of the ARM, in recent days there have been increased interest from regulatory agencies to submit ARM along with the ADaM define.xml. Creation of ARM will soon become a necessity for the regulatory submissions.

Pinnacle21 Community version is a popular tool to generate the define.xml, however it does not have the ARM creation feature in the current version (v2.2.0). Due to this limitation and increasing interest to include the ARM in ADaM define.xml by the regulatory authorities, there is an unmet need for an alternative approach to generate the ARM metadata for submissions.

This paper will present an approach using SAS to generate the ARM which can be integrated to an existing ADaM define.xml. The Resulting Integrated define.xml (ADaM define.xml + ARM) passes all compliance checks using the Pinnacle21 validator tool and is compliant for all regulatory submissions.

Keywords: CDISC, ADaM, Analysis Results Metadata, ARM, Define-XML, Define.xml

INTRODUCTION

This paper was written to address the unmet need for having an alternative approach for generation of the ARM information when creating an ADaM define.xml file. Analysis results metadata provides the reviewer with details on the analysis performed by identifying the reason, documentation, data and programming statements used to generate the statistical result outlined in the Analysis results display.

Benefits of ARM incorporated into a define document includes,

- **Traceability**
  - enabling the reviewer to understand flow of data from data collection to SDTM to ADaM and to analysis results metadata.
  - Enabling the reviewer to trace back the statistical results display in the analysis results metadata in the Analysis data.
- Provides **documentation** and reproduction of the identified critical analysis displays and important results.
- Provides enough information for the reviewer to regenerate the outlined analysis results in the ARM display and helps shorten the review time.

As per the PMDA Study data technical Conformance Guide (PMDA, 2015) the agency desires that the define.xml includes the ARM and it would be in no time that other regulatory agencies would start recommending submitting ARM along with the define.xml due to benefits associated with it.

There are very limited resources available to generate ARM which could be implemented as a part of the define.xml generated using any tool. Method discussed below helps automate the process of creating the ARM which might be a necessity for the regulatory submissions.

REQUIREMENTS FOR CREATING ARM

Analysis results metadata is not required for all analysis in the clinical study report or integrated summary report. It may be included for the critical analysis like primary & secondary endpoint analysis and any other analysis which is identified as critical by the sponsor. Additionally, when analysis results metadata is

provided for a specific results there is no requirement to describe every single piece of analysis in the identified TLF, as an example if primary endpoint table is identified for an analysis results display it might include several results like Mean, Median, SD, 95% CI's and p-values. Sponsor can make choice to include a display for all these results or include display for only complicated statistics like p-values. Once the sponsor has selected the list of outputs for the analysis results display, its key to collect the metadata information required for each results display.

An excel file (Table 1) with the listed tabs below can be used to capture all the information required for the analysis results metadata. This Excel template includes several tabs that must be filled out correctly for the tool to use the information for analysis results display in the define.xml. Information required in each of the tabs is specified in detail below. It's important to be cautious when populating the template with information, because:

- **Auto-correction feature in Excel.** Please be sure that the information entered in the sheet is not altered by this feature.
- **Special characters entered in the sheet can render an unreadable XML file.** To prevent this problem, avoid copy-paste directly from an external file, instead use "Paste-special" option. Additionally, using the "<" or ">" symbols could interfere with the XML language. In such cases, "&gt" or "&lt" can be used (in some cases, without "&").

CREATING AN ARM TAB/WORKSHEET IN THE ADAM DEFINE METADATA REPOSITORY SPECIFICATIONS

This tab captures most of the relational information required for analysis result metadata and renders links between the results, source ADaM datasets, external documentation and reference documents for the metadata information within the ARM section of the ADaM define. The Column in the spreadsheet is used to fill in the metadata information required for the analysis results display. Table 1 below identifies the Columns within the ARM tab, Metadata captured and their Purpose. Also, see Display 1 for the sample values in the template.

Table 1. Information Captured and purpose of the Columns in the ARM TAB

Column	Captured ARM Metadata	Purpose
Table Num	DISPLAY IDENTIFIER	Output table number as in the Clinical study report/Integrated summary report.
Table page No	DISPLAY IDENTIFIER	Output page number in the pdf document. If it's a single file, then it can be blank.
Table Description	DISPLAY NAME	Output display title, if needed additional information can be included to identify the analysis population.
Description of Analysis	RESULT IDENTIFIER	Specific analysis result.
Dataset Name	ANALYSIS DATASET	Dataset used to generate the analysis result. If there are multiple datasets it can be entered by ','.
CommentsID	ANALYSIS DATASET	Comments are provided when there is more than one dataset used to generate the analysis result to describes how the multiple datasets are used to generate the analysis result.
Analysis Paramter	ANALYSIS PARAMETER	Analysis parameter within BDS dataset used to generate the analysis result, can be blank if results is from a non-BDS dataset (like ADAE).
Analysis Variable	ANALYSIS VARIABLE	Analysis variable used to generate the analysis result specified in the display. If there is more than one variable used in analysis it could be included separated by ','. It's expected that at least one

		variable is specified.
Analysis Reason	ANALYSIS REASON	Reason analysis performed.
Analysis Purpose	ANALYSIS PURPOSE	Purpose of the analysis performed.
WhereClauseID	SELECTION CRITERIA	Describes the subset that will be used to generate the analysis result. Column only requires a unique ID for the dataset subset criteria, if there are multiple where clauses that needs to be specified then the where clause ID's can be separated by ",". Further filling instruction are provided in **WHERECLAUSES TAB** section below.
Documentation	DOCUMENTATION	Documentation from the SAP or CSP for the analysis result can be included. Optional hyperlink can also be added to the referenced document by filling the "DocumentID" and "Doc Page Number" columns in the template.
DocumentID	DOCUMENTATION	Name of the referenced document
Doc Page Number	DOCUMENTATION	Page number in the referenced document
Programming Code	PROGRAMMING STATEMENTS	SAS code used to perform the analysis specified. Includes the model statement (using the specific variable names) and all technical specifications needed for reproducing the analysis (e.g. covariance structure). Code can be provided in this part of the section or as a direct link to the SAS program.
SASCode	PROGRAMMING STATEMENTS	Optional Link to the program can be added if the code cannot be provided with in the programming statement section of the ARM display, if there is a need to link the program file within the module-5 programs folder, please specify the name of the SAS code file (in txt format)
SAS Version	PROGRAMMING STATEMENTS	SAS Version used

Display 1. Sample Screenshot of the ARM Tab

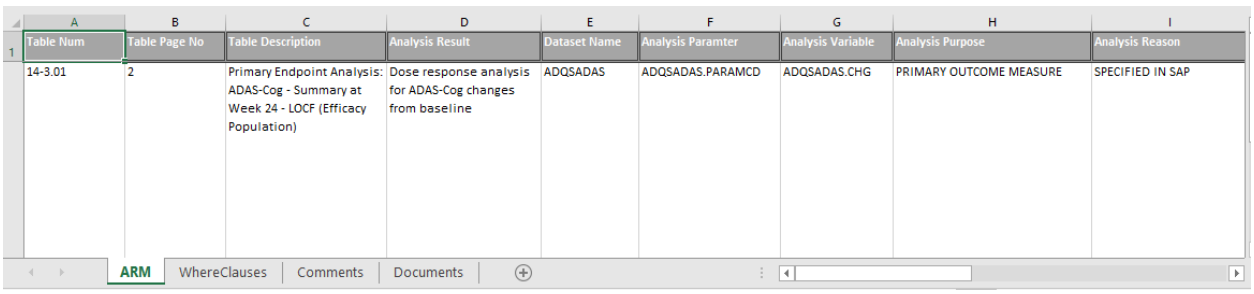
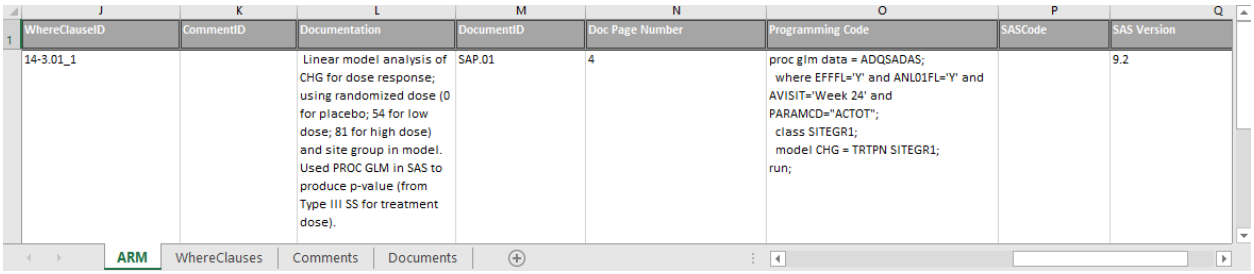


Table Num	Table Page No	Table Description	Analysis Result	Dataset Name	Analysis Paramter	Analysis Variable	Analysis Purpose	Analysis Reason
14-3.01	2	Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population)	Dose response analysis for ADAS-Cog changes from baseline	ADQSADAS	ADQSADAS.PARAMCD	ADQSADAS.CHG	PRIMARY OUTCOME MEASURE	SPECIFIED IN SAP



WhereClauseID	CommentID	Documentation	DocumentID	Doc Page Number	Programming Code	SASCode	SAS Version
14-3.01_1		Linear model analysis of CHG for dose response; using randomized dose (0 for placebo; 54 for low dose; 81 for high dose) and site group in model. Used PROC GLM in SAS to produce p-value (from Type III SS for treatment dose).	SAP.01	4	proc glm data = ADQSADAS; where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT"; class SITEGR1; model CHG = TRTPN SITEGR1; run;		9.2

ARM | WhereClauses | Comments | Documents

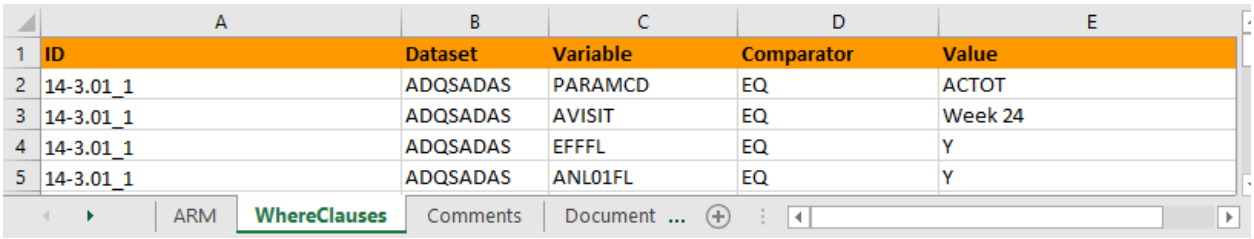
WHERECLAUSES TAB

Primarily used to list the subsets for the datasets or to outline the selection criteria for the dataset used for the generation of the analysis result. Table 2 outlines the description and purpose of the columns used in the whereClause tab. Please see Display 2 for-sample values.

Table 2. WhereClause Tab: Columns Description and Purpose

Column	Purpose
ID	Columns contains the selection criteria as specified in the WhereclauseID in the ARM tab
Dataset	Reflects the dataset for which the selection criteria or subset will be applied, will be same as specified in the Dataset Name column in the ARM tab.
Variable	Filled with the variable in the dataset on which the subset criteria will be applied.
Comparator	Aids to specify the where clause condition obtained using the comparator. List of allowable values include EQ, NE, GT, LT, GE, LE, IN and NOT IN.
Value	Specify the variable value of the condition in the where clause.

Display 2. Filled Sample screenshot of the Wherclause Tab



ID	Dataset	Variable	Comparator	Value
14-3.01_1	ADQSADAS	PARAMCD	EQ	ACTOT
14-3.01_1	ADQSADAS	AVISIT	EQ	Week 24
14-3.01_1	ADQSADAS	EFFFL	EQ	Y
14-3.01_1	ADQSADAS	ANL01FL	EQ	Y

ARM | WhereClauses | Comments | Document ...

COMMENTS TAB

Comments tab is typically populated when there are multiple datasets used to generate analysis results. Further comments can be added to clarify the reviewer on how the datasets will be linked together to get the expected results.

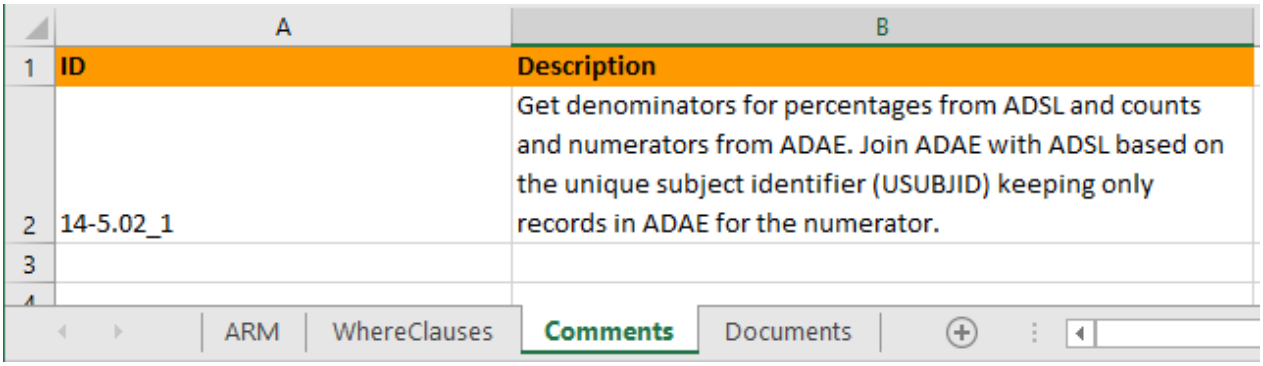
Table 3 outlines the description and purpose of the columns used in the Comments tab. Please see Display 3 for-sample values.

Table 3. Comments Tab: Columns Description and Purpose

Column	Purpose

ID	Contains the information from the CommentID column in the ARM tab
Description	Place the comment in this column.

Display 3. Sample screenshot of Comments tab



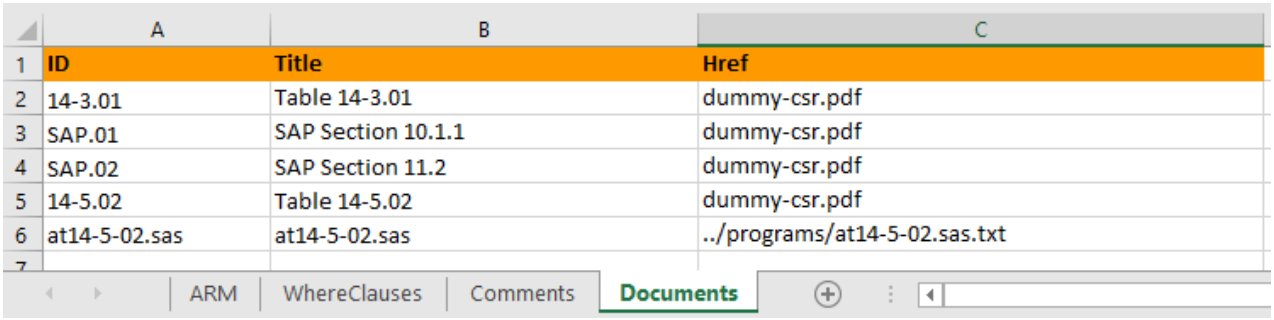
DOCUMENTS TAB

List any documents that are submitted to support the analysis results. This tab will be used to create hyperlinks to reference documents, outputs and programs within the analysis results display on the ADaM define. It is expected that all the referenced documents reside within the folder where the define is placed in the Module 5, only exceptions are the programs. If a link needs to be established for the program folder within the Module 5 then, ".../programs/filename.txt" can be used in the Href Column. For hyperlinking the table outputs, use table number specified in the Table Num column of the ARM tab as the document ID and specify the name of the output file document in the Href column. Table 4 outlines the description and purpose of the columns used in the Comments tab. Please see
Display 4 for sample values.

Table 4. Documents Tab: Columns Description and Purpose

Column	Purpose
ID	Contains the information from the Table Num, Documentation, & Programming Code columns in the ARM tab
Title	Title that should be displayed in the ARM define display.
Href	Link to the document

Display 4. Sample screenshot of Documents tab



GENERATE XML CODE FOR ADAM DEFINE

Analysis results metadata is an extension to the CDSIC Define-XML 2.0 and follows the same structure as of the Define-XML 2.0. As our main goal is to generate the ARM for an existing ADaM define some of the basic elements required for the ARM are already included in the ADaM define.

Basic elements which are already in the ADaM define without ARM include (Jansen, 2017),

- Study and Metadata version, ODM root element and XML header
- Datasets definitions
- Variables definitions
- Parameter Value definitions
- WhereClause definitions (excluding the wherclause used in the ARM display)
- Controlled terminology
- Comments definitions (excluding the comments included in the ARM display)
- Information about linked reference documents such as ADRG, source code files, etc

Analysis results metadata utilizes information from these basic elements section which are already in the define for the display. In addition to the above basic elements ARM requires the additional elements to complete the ARM display in define. Elements exclusive for the Analysis results metadata display are,

- WhereClause Definition Section to include the subsets used to filter the datasets to generate the analysis result
- Comments Definition Section, to additional clarity for usage of the analysis datasets for generating the analysis result
- Leaf (links) Definition section for the linked reference documents in the ARM such as Clinical study report, Statistical Analysis Plan and (or) linked SAS program files in the format as requested by the regulatory agencies
- Analysis results metadata Definition section, that provides the purpose and reasons for performing the analysis and selection criteria for the records subject to the particular analysis.

A SAS program will be used to generate the xml code for the define elements specific to the Analysis results metadata specified above. The SAS program uses a sequential approach by reading the individual tabs in the ARM template spreadsheet and generates xml code in the required syntax for the Define 2.0. Generated output from the program serves as plug-in to an existing ADaM define(without ARM) to include Analysis results metadata section. Output xml code follows the format as specified in Output 1.

Output 1. Sample Output syntax from the SAS Program

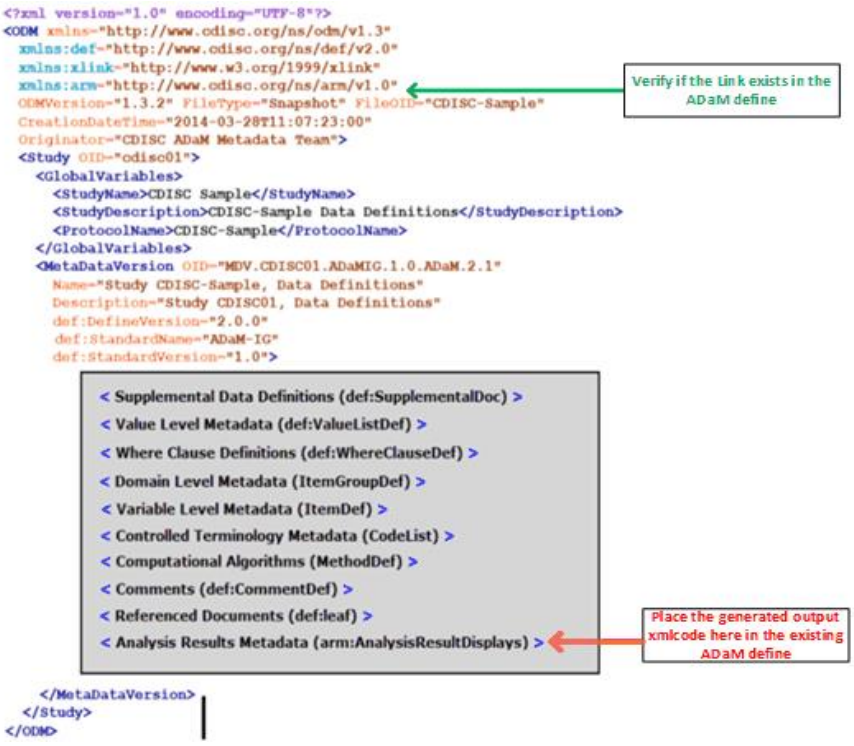
```
def:WhereClauseDef>
<RangeCheck>
<CheckValue>

<def:CommentDef>
    <Description>
        <TranslatedText>
    <def:DocumentRef>
        <def:PDFPageRef>

<def:leaf>
<def:title>

<arm:AnalysisResultDisplays>
    <arm:ResultDisplay>
       <Description>
          <TranslatedText>
       <def:DocumentRef>
          <def:PDFPageRef>
       <arm:AnalysisResult>
          <Description>
            <TranslatedText>
          <arm:AnalysisDatasets>
            <arm:AnalysisDataset>
               <def:WhereClauseRef>
               <arm:AnalysisVariable>
           <arm:Documentation>
            <Description>
               <TranslatedText>
            <def:DocumentRef>
               <def:PDFPageRef>
         <arm:ProgrammingCode>
         <arm:Code>
         <def:DocumentRef>
           <def:PDFPageRef>
```

PASTING XML CODE IN ADAM DEFINE

Outputs generated from the program will be manually copied to the ADaM define to plug-in the Analysis results metadata. Below are steps involved to copy the generated xml code to an existing ADaM define without the Analysis results metadata.

- Open the ADaM define.xml in an edit mode by using the text editor or notepad++
- Verify 'xmlns:arm=http://www.cdisc.org/ns/arm/v1.0' is specified in the ADaM define as shown in Figure 1
- Navigate to the end of define
- Paste the code generated by the SAS program in the location as shown in Figure 1.

Figure 1. Location in the Define to Paste the output xml code



SAS CODE TO GENERATE THE XML CODE FOR THE ANALYSIS RESULTS DISPLAY:

```
*======Assign Location of the ARM Spreadsheet==================;
    %let loc= ****Specify location of the ARM template
                Ex: C:\Users\Desktop\ARM******;
    %let input=****Specify the name of the excel file
                Ex: arm_template.xlsx*****;
*================================================================;

/*read in the collected information in Excel spreadsheet format,
output code will be generated in the same area*/;
options validvarname=V7 mlogic mprint symbolgen;

/*****************************************************************
Step 1: Generate WhereClause section
*****************************************************************/
proc import datafile ="&loc./&input..xlsx"
      dbms = xlsx
      out = whereclause replace;
      sheet = 'WhereClauses';
      getnames = yes;
```

```
run;

Proc sort data=whereclause;
      by id;
run;

data wc;
      length armcode word $2000 ;
      set whereclause end=last;
      by id ;
      temp_seq=_n_;

      if _n_=1 then do;
            armcode = '<!--
            ********************************************************
            ******************** -->';
            output;
            armcode = '<!-- WhereClause Definitions Section for ARM
            (Used/referenced in ARM Definitions)******-->';
            output;
            armcode = '<!--
            ********************************************************
            ******************** -->';
            output;
      end;

      if first.id then do;
            armcode='<def:WhereClauseDef OID="WC.'||strip(id)||'">';
            output;
      end;

      armcode = '<RangeCheck SoftHard="Soft" def:ItemOID="IT.'||
      strip(dataset)||"."||strip(variable)||'"Comparator="'
                        ||strip(comparator)||'">';
      output ;

      if comparator in ("EQ", "NE", "GT", "LT", "GE", "LE") then do;
            armcode = '<CheckValue>'||strip(value)||'</CheckValue>';
            output ;
      end;

      else if comparator in ("IN", "NOT IN") then do;
            do until(word=' ' or count-1);
                  count+1;
                  word = scan(value, count, ',');
                  armcode = '<CheckValue>'||strip(word)
                              ||'</CheckValue>';
                  output ;
            end;
      end;

      armcode = ' </RangeCheck>';
```

```
      output ;

      if last.id then do;
            armcode='</def:WhereClauseDef>';
            output;
      end;

      if last then do;
            armcode = '<!-- ******************************** -->';
            output;
            armcode = '<!-- End of WhereClause Sec for ARM***-->';
            output;
            armcode = '<!-- ******************************** -->';
            output;
            armcode = ' ';
            output;
      end;

      keep armcode;
run;

/*********************************************************************
Step 2: Generate Comments section
*********************************************************************/
proc import datafile ="&loc./&input..xlsx"
      dbms = xlsx
      out = comments replace;
      sheet = 'Comments';
      getnames = yes;
run;

Proc sort data=comments;
      by id;
      where id ne " ";
run;

data com;
      length armcode $2000;
      set comments end=last;

      if _n_=1 then do;
            armcode = '<!-- ************************************** -
            ->';
            output;
            armcode = '<!-- Comments Defintion Section for ARM********-
            ->';
            output;
            armcode = '<!-- ************************************** -
            ->';
            output;
      end;
```

```
      armcode= '<def:CommentDef OID="COM.'||strip(id)||'">';
      output;
      armcode='<Description>';
      output;
      armcode='<TranslatedText xml:lang="en">'||
                  strip(description)||'</TranslatedText>';
      output;
      armcode='</Description>';
      output;
      armcode='</def:CommentDef>';
      output;

      if last then do;
            armcode = '<!-- ****************************** -->';
            output;
            armcode = '<!-- End of Comments Sec for ARM******-->';
            output;
            armcode = '<!-- ****************************** -->';
            output;
            armcode = ' ';
            output;
      end;

      keep armcode;
run;

/*********************************************************************
Step 3: Generate Leafs (links) Definition for ARM
*********************************************************************/
proc import datafile ="&loc./&input..xlsx"
      dbms = xlsx
      out = doc replace;
      sheet = 'Documents';
      getnames = yes;
run;

data doctext;
      length armcode $2000;
      set doc end=last;

      if _n_=1 then do;
            armcode = '<!-- ********************************** -->';
            output;
            armcode = '<!-- Leafs Definitions Section for ARM***-->';
            output;
            armcode = '<!-- ********************************** -->';
            output;
      end;

      armcode='<def:leaf ID="LF.'||strip(id)||'" xlink:href="'||
                  strip(href)||'">';
      output;
```

```
      armcode='<def:title>'||strip(title)||'</def:title>';
      output;
      armcode='</def:leaf>';
      output;

      if last then do;
            armcode = '<!-- ****************************** -->';
            output;
            armcode = '<!-- End of Leafs Definitions Section -->';
            output;
            armcode = '<!-- ****************************** -->';
            output;
            armcode = ' ';
            output;
      end;

run;

/*********************************************************************
Step 4: Generate ARM definition Section
*********************************************************************/
proc import datafile ="&loc./&input..xlsx"
      dbms = xlsx
      out = readarm replace;
      sheet = 'ARM';
      getnames = yes;
run;

/*Basic house-keeping for further information manipulation*/
data readarm;
      set readarm;
      where table_num ne ' ';
      temp_seq = _n_;
      analysis_purpose=compress(analysis_purpose,,'kw');

      temp_tb1 = tranwrd(strip(table_num), ' ', '_');
      temp_tb2 = tranwrd(strip(table_num), '.', '_');

      proc sort;
      by table_num;
run;

/*Prepare the dataset for output in text format*/
data armtext(keep = armcode);
      set readarm end = last;
      by table_num;
      length armcode word word2 word3 $2000;
      suborder = _n_;

      if _n_ = 1 then do;
            armcode = '<!-- ****************************************
                        ************* -->';
```

```
      output;
      armcode = '<!-- *********Analysis results
                 metadata Definitions Section -->';
      output;
      armcode = '<!-- ****************************
                 ********************** -->';
      output;
      armcode = '<arm:AnalysisResultDisplays>';
      output;
end;

if first.table_num then do;
      armcode = ' <arm:ResultDisplay OID="RD.' ||
                 strip(temp_tb1) ||'" Name="' ||
                 'Table '||strip(table_num) || '">';
      output;
      armcode = ' <Description>';
      output;
      armcode = ' <TranslatedText xml:lang="en">'||
                 strip(Table_Description) || '</TranslatedText>';
      output;
      armcode = ' </Description>';
      output;
      armcode = ' <def:DocumentRef leafID="LF.'||
                 strip(table_num)|| '">';
      output;
      armcode = ' <def:PDFPageRef PageRefs="'||
                 strip(vvalue(table_page_no))|| '" ' ||
                 'Type="PhysicalRef"/>';
      output;
      armcode = ' </def:DocumentRef>';
      output;
end;

armcode = ' <arm:AnalysisResult';
output;
armcode = ' OID="AR.'|| strip(temp_tb1) ||
           '.R.' || strip(vvalue(temp_seq)) || '"';
output;

if Analysis_Paramter ne " " then do;
      armcode = ' ParameterOID="IT.'||
                 strip(Analysis_Paramter) || '"';
      output;
end;

armcode = ' AnalysisReason="'|| strip(Analysis_Reason) || '"' ;
output;
armcode = ' AnalysisPurpose="'|| strip(Analysis_Purpose)
           || '">' ;
output;
armcode = ' <Description>';
```

```
output;
armcode = ' <TranslatedText xml:lang="en">'||
          strip(Analysis_Result) || '</TranslatedText>';
output;
armcode = ' </Description>';
output;

if commentid ne " " then do;
     armcode='<arm:AnalysisDatasets def:CommentOID="COM.'||
               strip(commentid)||'">';
     output;
end;

else do;
     armcode = ' <arm:AnalysisDatasets>';
     output;
end;

ds=COUNTW(dataset_name,',') ;

do j=1 to ds;
     count+1;
     word = scan(dataset_name, j, ',');
     word2= scan(whereclauseid, j, ',');
     armcode = ' <arm:AnalysisDataset ItemGroupOID="IG.'
               || strip(word) || '">';
     output;
     if word2 ne " " then do;
          armcode = ' <def:WhereClauseRef WhereClauseOID="WC.'||
                    strip(word2)|| '"/>';
          output;
     end;
     c=COUNTW(analysis_variable,',') ;

     do i=1 to c ;
          word3= scan(analysis_variable,i,',');
               if scan(strip(word3), 1, '.')=word then do;
               armcode = ' <arm:AnalysisVariable ItemOID="IT.'
                         ||compress(word3)|| '"/>';
               output;
               end;
     end;

     armcode = ' </arm:AnalysisDataset>';
     output;
end;

armcode = ' </arm:AnalysisDatasets>';
output;
armcode = ' <arm:Documentation>';
output;
armcode = ' <Description>';
```

```
output;
armcode = ' <TranslatedText xml:lang="en">'||
           strip(Documentation) || '</TranslatedText>';
output;
armcode = ' </Description>';
output;

if documentid ne '' then do;
     armcode = ' <def:DocumentRef leafID="LF.'||
               strip(documentid) || '">';
     output;
     armcode = ' <def:PDFPageRef PageRefs="' ||
               strip(vvalue(Doc_Page_Number)) || '" ' ||
               'Type="PhysicalRef"/>';
     output;
     armcode = ' </def:DocumentRef>';
     output;
end;

armcode = ' </arm:Documentation>';
output;

armcode = ' <arm:ProgrammingCode Context="SAS version '||
          strip(vvalue(SAS_Version)) || '">';
output;

if programming_code ne ' ' then do;
     armcode = ' <arm:Code>';
     output;
     armcode = strip(programming_code);
     output;
     armcode = ' </arm:Code>';
     output;
end;

if sascode ne ' ' then do;
     armcode = '<def:DocumentRef leafID="LF.'||strip(sascode)||
               '" />';
     output;
end;

armcode = ' </arm:ProgrammingCode>';
output;
armcode = ' </arm:AnalysisResult>';
output;

if last.table_num then do;
     armcode = ' </arm:ResultDisplay>';
     output;
end;

if last then do;
```

```
            armcode = '</arm:AnalysisResultDisplays>';
            output;
      end;
run;

data final;
      set wc(keep = armcode)
            com(keep = armcode)
            doctext(keep = armcode)
            armtext(keep = armcode);

      armcode=strip(armcode);
run;

/*generate the xml syntax in text format*/
data _null_;
      set final;
      file "&loc./armxml.txt";
      put armcode;
run;
```

EXAMPLE OF ARM (SUMMARY) FOR A CDISC STUDY

Analysis Results Metadata (Summary) for Study CDISC-Sample

Table 14-3.01 Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population) Dose response analysis for ADAS-Cog changes from baseline Pairwise comparisons to placebo for ADAS-Cog changes from baseline
Table 14-5.02 Incidence of Treatment Emergent Serious Adverse Events by Treatment Group Incidence of Treatment Emergent Serious Adverse Events by Treatment Group

Analysis Results Metadata (Detail) for Study CDISC-Sample

Table 14-3.01

Display	**Table 14-3.01 Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population)**
Analysis Result	**Dose response analysis for ADAS-Cog changes from baseline**
Analysis Parameter(s)	PARAMCD = "ACTOT" (Adas-Cog(11) Subscore)
Analysis Variable(s)	CHG (Change from Baseline)
Analysis Reason	SPECIFIED IN SAP
Analysis Purpose	PRIMARY OUTCOME MEASURE
Data References (incl. Selection Criteria)	ADQSADAS [PARAMCD = "ACTOT" and AVISIT = "Week 24" and EFFFL = "Y" and ANL01FL = "Y"]
Documentation	Linear model analysis of CHG for dose response; using randomized dose (0 for placebo; 54 for low dose; 81 for high dose) and site group in model. Used PROC GLM in SAS to produce p-value (from Type III SS for treatment dose). SAP Section 10.1.1

Programming Statements	[SAS version 9.2]
	`proc glm data = ADQSADAS;`
	`  where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT";`
	`  class SITEGR1;`
	`  model CHG = TRTPN SITEGR1;`
	`run;`

CONCLUSION

Define.xml plays a critical role in submission helping the reviewer to understand the database. Addition of the ARM metadata helps the reviewer to validate critical results in the study using the Analysis results display presented in the define. This approach also provides a structure for the metadata repository

contributes to an end-to-end standards strategy that influences a future-proof TFL implementation. Additionally, the is approach, increases robustness, quality, efficiency, and an avenue to a re-imagined possibility of the creation of interactive CSR reports/outputs.

Due to the limited availability of the tools to generate the ARM for an existing ADaM define, the suggested approach in the paper is straight forward and can be implemented by any Clinical SAS programmer without any prior knowledge of XML language from a pre-existing ADaM define. This is a cost saving approach that can benefit any organization that chooses to implement the method aforementioned in this paper.

REFERENCES

CDISC ADaM Metadata Sub-Team. (2015, 01 27). *Analysis Results Metadata (ARM) v1.0 for Define-XML v2.0.* Retrieved from CDSIC: https://www.cdisc.org/standards/foundational/define-xml/analysis-results-metadata-arm-v10-define-xml-v20

Lex Jansen. (2016). *Creating Define-XML version 2 including Anlaysis results Metadata with the SAS Clinical Standards Toolkit.* Retrieved from Lexjansen: https://www.lexjansen.com/pharmasug/2016/SS/PharmaSUG-2016-SS01-PPT.pdf

Lex Jansen. (PharmaSUG 2017 Proceedings.). *Creating Define-XML version 2 including Analysis Results Metadata with the SAS® Clinical Standards Toolkit.* Retrieved from LexJansen: https://pharmasug.org/proceedings/2016/SS/PharmaSUG-2016-SS01.pdf

Pinky Anandani Dutta, I. I. (2018). *Generating Define.xml from Pinnacle 21 Community.* Retrieved from PharmaSUG: https://www.pharmasug.org/proceedings/2018/AD/PharmaSUG-2018-AD29.pdf

PMDA. (2015, April 27). *Technical Conformance Guide on Electronic Study Data Submissions*. Retrieved from Pharmaceuticals and Medical Devices Agency: https://www.pmda.go.jp/files/000206449.pdf

ACKNOWLEDGMENTS

This paper is dedicated to all Statistical Programmers who have a desire to progress daily in their advancement in SAS and Clinical Submissions knowledge.

This paper is also a testament that any Statistical Programmer can write a paper that increases the knowledge base of what the future might look like, and Srivathsa Ravikiran is one great example, having been the SAS Imagineer of this paper you are reading.

RECOMMENDED READING

- *CDSIC Define-XML Specification, Version 2.0*
- *Study Data Technical Conformance Guide, March 2018*
- *Analysis results Metadata Specification, Version 1.0*
- *Study data technical Conformance Guide, April 2015*
- *Define-XML stylesheet version 2018-11-21 found at https://wiki.cdisc.org/display/PUB/Stylesheet+Library.*
- *Analysis Data Model (ADaM) version 2.1*
- *Analysis Data Model Implementation Guide Version 1.1.*

CONTACT INFORMATION

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

Name: Srivathsa Ravikiran, MS
Enterprise: AstraZeneca Pharmaceutical LP
Work Phone: (857) 207 5363
Email: srivathsarkiran@gmail.com

Name: Priscilla Gathoni, MS, MBA
Enterprise: AstraZeneca Pharmaceutical LP
Work Phone: (203) 907 8238
Email: gathonigachie@gmail.com