Making a List, Checking it Twice (Part 1): Techniques for Specifying and Validating Analysis Datasets

Elizabeth Li, PharmaStat LLC, Newark, California

Linda Collins, PharmaStat LLC, Newark, California

ABSTRACT

In the CDISC era, biotechnology and pharmaceutical companies are paying increasing attention to how analysis dataset specifications are documented and accuracy of datasets that are generated. It has always been a desirable practice to record the details about analysis datasets, including the structure of the dataset, the source of data variables, the logic of derivations, and methods of special data handling. For FDA submissions that include analysis data model (ADaM) datasets, the analysis data specifications must be included in submission documentation. The use of independent programming is increasingly a gold standard validation method. In this paper, we describe techniques for leveraging analysis data specifications to automate processes in producing analysis datasets, quality control of the data by independent programming validation, and generating Data Definitions (define.xml) content. The result of this process is increased confidence in the quality of the data and the reliability of the documentation.

KEY WORDS

CDISC, ADaM, Macros, SAS® programming validation, analysis data specifications

INTRODUCTION

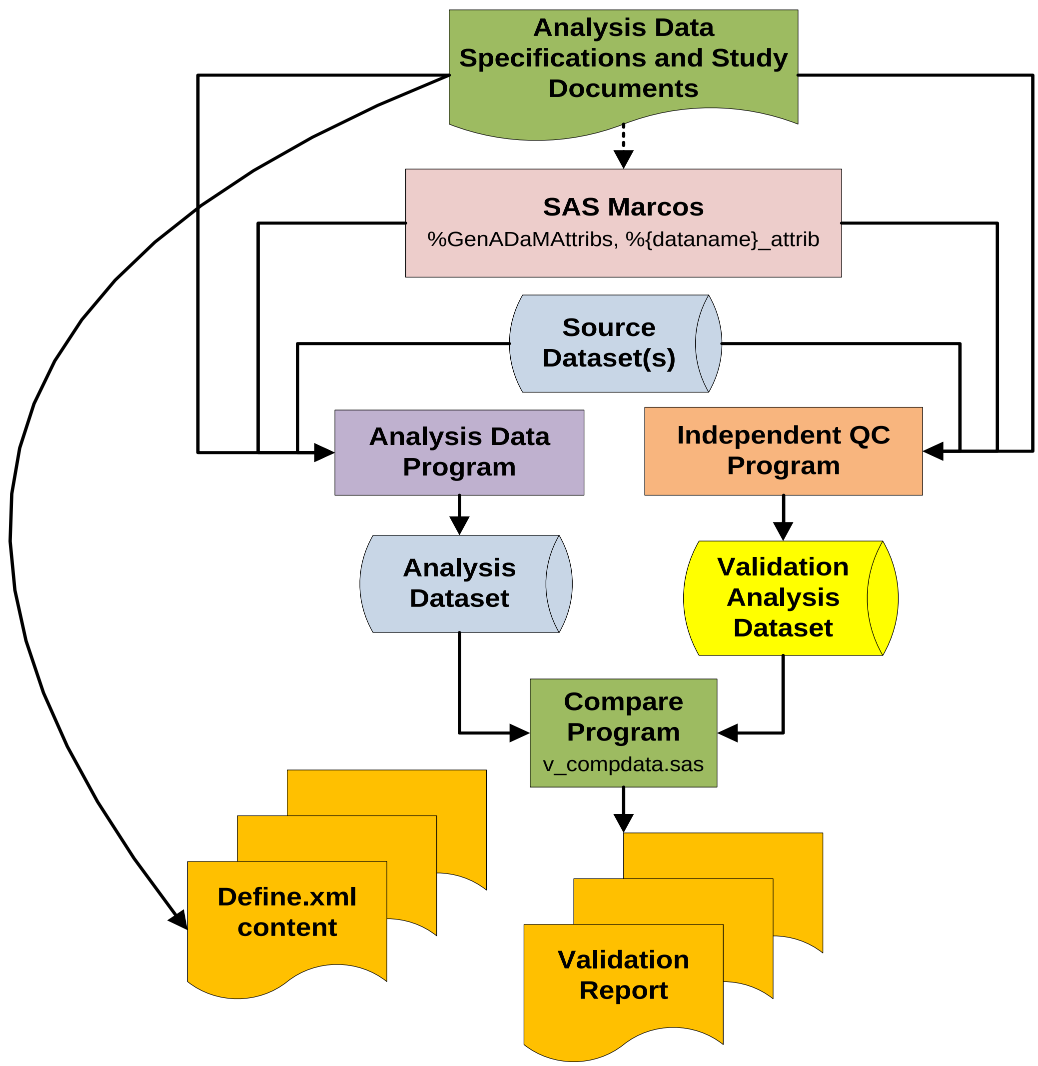
Biotechnology and pharmaceutical industry has adopted many industry standards. Recently, many more companies in the industry have started implementation of the standards from Clinical Data Interchange Standards Consortium (CDISC). One of the key sets of standards is Analysis Data Model (ADaM), which is used for submitting analysis datasets, along with define.xml, to the US Food and Drug Administration (FDA). When implementing ADaM standards, much effort is focused on how to document and maintain analysis dataset specifications, as well as how to generate accurate analysis datasets. In this paper, we present techniques for specifying analysis datasets, such that the specification document can be used to produce the analysis datasets, compare the data by independent programming validation, keep analysis data and the document in sync, and generate Data Definitions (define.xml) content. The techniques presented here are depicted by the flowchart in Figure 1.

MAKING A LIST

Traditionally, the most important document that a SAS programmer creates is the SAS code. When a question is raised on how a particular variable was derived, the first place a programmer will look is in the program that created the dataset that contains the variable. We all know this type of documentation is not sufficient. Using a SAS program as documentation has the following drawbacks:

1) Not a central location for analysis data specifications
2) Hard to perform maintenance for changes or updates
3) Cannot be used as a basis for independent validation.

Figure 1. Analysis Dataset Production Process Flow



Performing data analysis without specifications will cost time and money. Documentation saves time in overall programming effort. With implementation of ADaM standards in mind, we have used the following analysis data specifications. The specifications for analysis datasets are stored in a spreadsheet in a 'metadata' folder with the structure described in Table 1. There is a separate specification file for each analysis dataset, with the analysis file name as the specification file name. The advantages of this approach for analysis data documentation are

1) Having a central location for all analysis dataset specifications
2) Can be used as basis for independent validation for analysis datasets
3) Easy to maintain changes or updates to the specifications
4) The metadata information will be programmatically turned into ATTRIB statements and KEEP statements, which are invoked by the analysis dataset program. This keeps the analysis attributes of dataset variables in sync with the specifications
5) If necessary, the specifications can also be converted to a define.xml.

Table 1: Structure of analysis dataset specifications	
Column	**Content**
Variable Order Number	Integer for order in the data vector
Variable Name	Name of variable
Variable Label	Label of variable
Type	Char or Num
Data Type	text, float, or integer
Length	Length of variable
Codelist Name	(Optional) Name of code list
Origin of Existing Variable	Name of source variable(s)
ADaM Define Specification	Logic for derivation
DisplayFormat	SAS format for the variable
SignificantDigit	Significant digit, for numeric data

DETERMINATION OF THE TYPE OF ANALYSIS DATASET AND VARIABLES

Upon the sign-off of the statistical analysis plan (SAP), analysis specifications are drafted. Most common types of analysis datasets are:

1) Subject level analysis dataset (ADSL) – demographics, baseline characteristics, population flags
2) Basic data structure (BDS) – findings type of data
3) Time to event (ADTTE)
4) Adverse events (ADAE)

One way to ensure all the analysis variables are included in appropriate analysis datasets is to annotate mock-shells with the names of analysis dataset and variables that will be used for summary tables, figures, and listings. By annotation, a statistician or senior programmer analyst will determine the variable attributes. In referencing annotated case report forms (CRFs), protocols, and SAPs, the source data name and source variable names that will be used to derive the analysis variables can be identified. Once the type of data structure, type of variables, source data, and derivations are determined, specifications can be developed. Table 2 shows an example of an analysis data specification.

Table 2. Sample Analysis Specifications (not all columns are shown)

Variable Order Number	Variable Name	Variable Label	Type	Length	Core Variable	Codelist	Origin of Existing Variable	ADaM Define Specification
1	STUDYID	Study Identifier	Char	20	Yes		DM.STUDYID	
2	USUBJID	Unique Subject Identifier	Char	20	Yes		DM.USUBJID	
3	SUBJID	Subject Identifier for the Study	Char	4	Yes		DM.SUBJID	
4	SITEID	Study Site Identifier	Char	1	Yes		DM.SITEID	
5	AGE	Age	Num	8	Yes		DM.AGE	
6	AGEU	Age Units	Char	5			DM.AGEU	
7	SEX	Sex	Char	1	Yes		DM.SEX	

Table 2. Sample Analysis Specifications (not all columns are shown)								
Variable Order Number	**Variable Name**	**Variable Label**	**Type**	**Length**	**Core Variable**	**Codelist**	**Origin of Existing Variable**	**ADaM Define Specification**
8	SEXN	Sex (N)	Num	8	Yes	SEXN	Derived	Derived from DM.SEX when SEX=M then SEXN=1, SEX=F then SEXN=2.
9	SAFFL	Safety Population Flag	Char	1	Yes	YN	Derived	If a patient has at least one record in EX domain and EX.EXOCCUR=Y then SAFFL=Y. Otherwise, SAFFL=N.
10	ITTFL	Intent-To-Treat Population Flag	Char	1	Yes	YN	Derived	Derived from DM.ARM: if DM.ARM is not blank then ITTFL=Y.

AUTOMATED PROCESS FOR PRODUCING ANALYSIS DATASETS

Once the specifications are drafted, the SAS program for the analysis data can be created. The analysis dataset creation program will have the same name as the dataset it produces. At the beginning of the program, there is a call to macro %GenADaMAttribs. This macro reads the analysis specification Excel file and writes a macro {dataset name}_attrib.sas in the local macros directory.

Sample SAS code of %GenADaMAttribs :

```
*****  Fetch data from ADaM spec spreadsheet Variables tab *****;
   proc import
         out = SpecData
         DATAFILE = "&SpecLib.\&ADaMName..xls"
         DBMS = EXCEL
         REPLACE
         ;
        SHEET    = "Variables$";
        GETNAMES = YES;
         TEXTSIZE = 2000;
   run ;

   data _null_ ;
      file "&MacroLib.\&MacroName"  noprint notitles ;
         loopcnt = 1
      set SpecData (where = (Source_Tab = 'Variables')) end = _eof_ ;

***** writing AdaMKeepList ************;

      if loopcnt = 1 then put / "            " @ ;
      put Variable_Name @ ;
      loopcnt + 1 ;
      if loopcnt >= 8 then loopcnt = 1 ;

      if _eof_ then put / "            ;" ;
   run ;

**** Writing the contents of the ATTRIB statement for the main ADaM
**** dataset, in data vector order.;
```

```
data _null_ ;
   file "&MacroLib.\&MacroName"  noprint notitles mod ;

   if _N_ = 1 then
      do ;
         put / '   %let ADaMVarAttribs = ' ;
      end ;

   set SpecData (where = (Source_Tab = 'Variables')) end = _eof_ ;

   put '            ' Variable_Name @24 'length = ' @ ;
   if upcase(Type) = 'CHAR' then put '$' @ ;
   _clen = trim(left(put(Length,4.))) ;
   put _clen @ ;

   _lablen = length(Variable_Label) ;
   if index(Variable_Label, "'") = 0 then
      put @39 "label = '" Variable_Label $varying. _lablen "'" @ ;
   else put @39 'label = "' Variable_Label $varying. _lablen '"' @ ;

   if DisplayFormat ^= ' ' then do ;
      _fmtlen = length(DisplayFormat) ;
      put @92 'format = ' DisplayFormat $varying. _fmtlen @ ;
   end ;
   put ;
   if _eof_ then
      put "            ;" ;
run ;
```

Sample SAS macro ADSL_attrib.sas that is created by %GenADaMAttribs:

```
%macro ADSL_Attrib ;

   %global ADaMKeepList ADaMVarAttribs TempVarAttribs ;


   %let ADaMKeepList =
            STUDYID USUBJID SUBJID SITEID AGE AGEU SEX
            SEXN SAFFL ITTFL
            ;

   %let ADaMVarAttribs =
            STUDYID    length = $20   label = 'Study Identifier'
            USUBJID    length = $20   label = 'Unique Subject Identifier'
            SUBJID     length = $4    label = ‘Subject Identifier for the Study’
            SITEID     length = $1    label = 'Study Site Identifier'
            AGE        length = 8     label = 'Age'                          Format=8.0
            AGEU       length = $5    label = 'Age Units'
            SEX        length = $1    label = 'Sex'
            SEXN       length = 8     label = 'Sex (N)'                      Format=8.0
            SAFFL      length = $1    label = 'Safety Population Flag'
            ITTFL      length = $1    label = 'Intnet-To-Treat Population Flag'
;
%mend;
```

This macro, when invoked, will generate two macro variables:

- &ADaMVarAttribs   A set of attribute statements for the variables in the metadata.
- &ADaMVarKeep   A KEEP statement for the variables in the metadata.

The analysis dataset programs will invoke corresponding analysis dataset attrib macros and use the ATTRIB and KEEP macro variables to define the variable attributes and list of variables to be included in the final output of the analysis datasets. This process ensures the variables and their attributes match those specified in the analysis data specifications.

CHECK IT TWICE

The analysis datasets are the source for the analysis results, in the form of summary tables, listings, and figures. The accuracy and integrity of the datasets are essential to analysis conclusions in study reports. In order to ensure the accuracy of the analysis datasets, programming validation is necessary. Validating analysis datasets before using them to generate summary tables or figures will save review time. Finding errors in the data and in the programs that generate analysis datasets will save rework in generating analysis results. We present a technique of independent programming validation. In Figure 1, a programmer generates an analysis dataset based on source data, analysis data specifications, and other study documents. A different programmer will use the same information to create a validation analysis dataset. In theory, both datasets should match, which is the ultimate goal of the validation. There are many sources of discrepancies, when the two datasets are compared for the first time:

1) Different interpretations of the specifications - different programming logic.
2) Specifications did not cover methods of certain data handling and programmers used their own methods to handle data derivations.
3) Data issues, i.e. inconsistent data, incomplete data, or missing data
4) Programming logic error

When discrepancies are found, the analysis data specifications may be updated for clarity, additional specifications may be needed, or rules for special data handling may be defined. When programmers identified errors in the program code, they will fix them, generate a new set of analysis datasets, compare the datasets, and resolve discrepancies. This process continues until both datasets match. After first sorting the datasets by key variables (&sortby), the following SAS macro V_COMPDATA was created to speed up the comparison of the datasets.

```
**** Checking mismatch observations *****;
data nomatch compobs(keep=&sortby) baseobs(keep=&sortby);
  length mismatch $20.;
  merge valid(in=incomp keep=&sortby) develop(in=inbase keep=&sortby);
  by &sortby;

  if incomp and inbase then do;
      if incomp then output compobs;
      if inbase then output baseobs;
  end;
  else do;
      if incomp then mismatch="Not in develop";
      if inbase then mismatch="Not in valid";
      put _all_;
      output nomatch;
  end;
run;

***** comparing two datasets *****;
ods output compare.CompareSummary=compsum;
ods output compare.CompareVariables=varsum;
proc compare base=develop compare=valid
   criterion=.0001
   out=result outnoequal outbase outcomp outdif;
   id &keys;
run;

data summary;
  set compsum %if %sysfunc(exist(varsum)) %then varsum;;
run;

**** output to an Excel file ****;
proc export
   data= summary
   outfile = "&outfile"
   dbms = excel
```

```
   replace
   ;
   sheet = "&tabname Summary";
run;
```

GENERATING DEFINE.XML CONTENT

Using the specification structure from Table 1, most of the define.xml content will already have been documented. The content of define.xml should also contain the following information: dataset metadata, value level metadata, and code list.

Table 3 is an example of how to document the dataset metadata for a project. The analysis dataset label can be set as the dataset description, by using the description of the datasets in this table.

Table 3. Sample Dataset Metadata

Dataset	Description	Class	Structure	Purpose	Keys
ADSL	Subject level Analysis Dataset	Trial Design	One record per subject	Analysis	STUDYID, USUBJID
ADAE	Adverse Event Analysis Dataset	Events	One record per adverse event per sequence per subject	Analysis	STUDYID, USUBJID, AETERM, AESPID
ADEFF	Efficacy Analysis Dataset	Findings	One record per parameter per analysis category per subject	Analysis	STUDYID, USUBJID, PARAMCD, ANAL1CAT

The value level metadata can be documented to specify details in parameters of findings data and how the analysis variables are passed or derived from source data. The structure of the specification document for value level metadata is very similar to the one for variables in Table 2. Table 4 shows is an example of value level metadata.

Table 4. Sample Value Level Metadata

Value Order Number	Value	Label	Type	Length	Codelist	Origin of Existing Variable	ADaM Define Specification
1	DIST	Distance (m)	Num	8		Derived	XDSTRESN when XDTESTCD=DIST
2	SPEED	Speed (m/s)	Num	8		Derived	XDSTRESN when XDTESTCD=SPEED
3	MAX	Maximum Speed Reached	Char	1	YN	Derived	Y, when PARAM=SPEED if AVAL is > baseline maximum value; N otherwise.
4	MIN	Minimum Speed Reached	Char	1	YN	Derived	Y, when PARAM=SPEED is if AVAL < baseline minimum value; N otherwise.

Finally, a code list can be used as a source for data formatting. Table 5 shows an example of code list.

Table 5. Sample Code List

Codelist Name	Data Type	Code	Decode	Rank
LB_TOX	Integer	0	GRADE 0	1
LB_TOX	Integer	1	GRADE 1	2
LB_TOX	Integer	2	GRADE 2	3
LB_TOX	Integer	3	GRADE 3	4

Table 5. Sample Code List				
Codelist Name	**Data Type**	**Code**	**Decode**	**Rank**
LB_TOX	Integer	4	GRADE 4	5
YN	Text	Y	YES	1
YN	Text	N	NO	2

The information presented in Tables 2 to 5 is not only a source for define.xml content, but also a part of the documentation of analysis data specifications.

CONCLUSION

Documentation for analysis data specifications not only helps efficient programming, but also provides a basis for validation. We use the techniques, which have been presented in this paper, as part of our programming guidelines. We create analysis datasets with independent programming validation, leveraging the specifications and using SAS® macros to generate variable attributes. We maintain the specifications in sync with the analysis datasets. In addition, we use SAS® macros in the validation process to compare analysis datasets from both production and independent programs to ensure the data accuracy. Furthermore, we extract information from the specifications for define.xml content. In conclusion, the techniques help us deliver a quality product efficiently to our clients.

REFERENCES

Validating Clinical Trial Data Reporting with SAS®, by Carol I. Matthews and Brian C. Shilling

CONTACT INFORMATION

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

Elizabeth Li

PharmaStat, LLC
39899 Balentine Drive, Suite 109
Newark, CA 94560
Work Phone: 510 656-2080
Fax: 510 656-2081
elizabethli@pharmastat.com
Web: www.pharmastat.com

Linda Collins

PharmaStat, LLC
39899 Balentine Drive, Suite 109
Newark, CA 94560
Work Phone: 510 656-2080
Fax: 510 656-2081
lcollins@pharmastat.com
Web: www.pharmastat.com