PharmaSUG 2012 - Paper DS16

Common Misunderstandings about ADaM Implementation

Nate Freimark, Theorem Clinical Research, King of Prussia, PA
Susan Kenny, Amgen Inc, Thousand Oaks, CA
Jack Shostak, Duke Clinical Research Institute, Durham, NC
John Troxell, John Troxell Consulting LLC, Bridgewater, NJ

ABSTRACT

The December 2009 release of Version 1.0 of the Analysis Data Model Implementation Guide (ADaMIG), and its endorsement by the United States Food and Drug Administration (FDA), has lead to widespread industry implementation. As sponsors attempt to implement the ADaM standard, there has been some variety in interpretation of aspects of the ADaMIG. The authors describe some common difficulties with implementation of ADaM, and explain best practices. Variables AVALC and DTYPE, the relationship between PARAM and AVAL, and other aspects of the ADaMIG are discussed. In each case, the authors attempt to clarify the intent of the ADaMIG, in order to assist current implementers and to set the stage for improved clarity in the next version of the standard.

Keywords: ADaM, ADaMIG, CDISC, FDA

INTRODUCTION

This paper concerns implementation of the Analysis Data Model (ADaM) developed by the Clinical Data Interchange Standards Consortium (CDISC). As such, it is assumed that readers are familiar with the ADaM standard and the CDISC Study Data Tabulation Model (SDTM). In particular, it is assumed that the reader has studied the ADaM Implementation Guide and the Analysis Data Model. CDISC standards are available at www.cdisc.org.

The authors of this paper provide their opinions based on experience and involvement in the development of the ADaM standard.

"HERE ARE MY ADAM DATASETS…"

Prior to the CDISC ADaM model there were "analysis datasets" provided as part of a submission. With the advent of the CDISC ADaM model we now have a specific type of analysis datasets called "ADaM" datasets. ADaM datasets are a type of analysis dataset, but not all analysis datasets are necessarily ADaM datasets. Some people are now referring to any and all analysis dataset submissions as "ADaM datasets" whether or not the ADaM model was followed and this is a problem. If you provide analysis datasets to a reviewer and call them "ADaM datasets" but you did not follow the ADaM model this will lead to confusion and frustration on the part of the reviewing party. In addition, using 'ADaM" as an adjective to refer to any analysis dataset, regardless of design, reduces the meaning of the common goal of data standards.

The Analysis Data Model v.2.1 defines two standard data structures, ADSL and the ADaM Basic Data Structure (BDS). ADSL is the only member of the ADSL class of datasets. All BDS datasets are members of the BDS class. In the near future the Analysis Data Model v.2.1 document and the ADaM Implementation Guide v1.0 will be augmented by final published versions of ADTTE and ADAE documents. ADTTE is a special example of the BDS class of datasets. ADAE will necessitate the creation of a new ADaM dataset class.

ADaM recognizes that the currently published structures of BDS and ADSL do not yet encompass the universe of all statistical modeling needs and there may be scenarios where one would have to create other analysis datasets that do not fit any of the defined structures. Are these datasets where the ADaM dataset metadata for "CLASS OF DATASET" is set to "OTHER" considered ADaM datasets? These datasets do not fit the ADaM specified structures, but if these additional "OTHER" class datasets are submitted as part of an analysis dataset package including ADSL and BDS structured datasets, and the "OTHER" class datasets follow the fundamental principles described in the Analysis Data Model document, then referring to the whole package as an ADaM dataset submission seems reasonable. As a reminder, the fundamental principles are:

Analysis datasets and their associated metadata must:

- facilitate clear and unambiguous communication
- provide traceability between the analysis data and its source data (ultimately SDTM)
- be readily useable by commonly available software tools

Analysis datasets must:

- be accompanied by metadata
- be analysis-ready

The fact that a submission might contain some analysis datasets that do not comply with ADSL or BDS could make things somewhat confusing for the recipient.

Here are some suggestions when referring to ADaM datasets verbally and within submissions:

- If analysis datasets do not follow the ADaM fundamental principles, then do not call them "ADaM datasets."
- If a submission contains ADSL, and possibly BDS structured data, but also includes datasets of other structures then consider the following for those other datasets:
    1. Make sure the ADaM dataset metadata has "CLASS OF DATASET" set to "OTHER."
    2. Consider explaining in the reviewer's guide which analysis datasets are not ADaM-compliant and why.
    3. Consider using an analysis dataset naming convention that would make the "OTHER" non ADaM compliant datasets stand out. For example, since compliant ADaM dataset names must start with "AD," consider prefacing the "OTHER" analysis dataset names with some other prefix, such as perhaps 'AX' instead.

SAME NAME, SAME VALUE, SAME METADATA

A basic principle of data warehouses is that all aspects of a given variable should be the same regardless of where it appears in a database. Changing either the metadata or the values of a variable, yet retaining the same variable name, violates this basic principle and adds to confusion and lack of traceability. Since ADaM datasets utilize variables from multiple sources, this principle is important and must not be violated. For example, if an ADaM dataset brings in the variable DM.AGE and leaves the variable name as 'AGE', then the values of this variable must remain the same. It is not allowed to convert the values of DM.AGE to a different unit or to change the label from 'Age' to 'Age at Baseline', for example. If it is desired to make any changes in the values and/or the metadata, then the user must change the name of the variable and reference the original variable name in the metadata definition of the new variable.

THE PURPOSE OF SCRDOM, SRCSEQ, SRCVAR

Traceability is of high importance in the ADaM model and often one BDS dataset will use multiple SDTM domains as the source for analysis variables. The ADaM variables SCRDOM, SRCSEQ, and SRCVAR were defined for situations where multiple SDTM domains and/or ADaM datasets were used to create one ADaM dataset. For example, if a BDS dataset utilized data from both the LB and the AE domain to define AVAL, then all 3 variables are needed for traceability, as demonstrated in the following table:

USUBJID	**PARAM**	**AVALC**	**SRCDOM**	**SRCSEQ**	**SRCVAR**
ZZZ101-01	My Event of Interest	Hypercholesteremia	AE	34	AEDECOD
ZZZ101-01	My Event of Interest	Grade 4 Liver Toxicity	LB	21	LBTOXGR

However, if an ADaM dataset uses just one SDTM domain and the source of information is just one SDTM variable, then it is not parsimonious to include SRCDOM and SRCVAR when all records would have the same value. For example, the following table illustrates an overuse of these traceability variables:

USUBJID	**PARAM**	**AVAL**	**SRCDOM**	**SRCSEQ**	**SRCVAR**
ZZZ101-01	Total Cholesterol (mg/dL)	210	LB	19	LBSTRESN
ZZZ101-02	Total Cholesterol (mg/dL)	178	LB	21	LBSTRESN

Keeping SRCDOM and SRCVAR in ADaM when all values are the same throughout the dataset provides no measureable benefit and should be avoided. In this case, LBSEQ would be sufficient to provide the needed traceability, such as:

USUBJID	**PARAM**	**AVAL**	**LBSEQ**
ZZZ101-01	Total Cholesterol (mg/dL)	210	19
ZZZ101-02	Total Cholesterol (mg/dL)	178	21

SHAPE-SHIFTING SDTM VARIABLES

ADaM analysis datasets often require the concatenation of records from various SDTM domains. In order to save space when maintaining SDTM variables for traceability purposes, some people rename a set of variables from one SDTM domain to match the variable names from another SDTM in order to “save” a set of columns. This way all rows for the columns will be populated instead of one set of variables populated for one section of observations and another set of variables populated for the second section of observations. While this is a creative method to save space and may not be in technical violation of “SAME NAME, SAME VALUE” since these values never existed in SDTM in this form, it still breaks the traceability back to the SDTM domains that the data was sourced from.

For example:

SDTM EX

USUBJID	**EXCAT**	**EXTRT**	**EXDOSE**	**EXDOSU**	**EXSTDTC**
ZZZ101-01	Rescue Medication	Acetaminophen	500	mg	2012-03-01
ZZZ101-02	Rescue Medication	Acetaminophen	1000	mg	2012-03-15

SDTM CM

USUBJID	**CMCAT**	**CMTRT**	**CMDOSE**	**CMDOSU**	**CMSTDTC**
ZZZ101-01	Prior Medication	Acetaminophen	500	Mg	2012-02-01
ZZZ101-02	Prior Medication	Acetaminophen	1000	Mg	2012-02-15

ADaM ADCM Combining SDTM EX and CM

USUBJID	**CMCAT**	**CMTRT**	**CMDOSE**	**CMDOSU**	**CMSTDTC**
ZZZ101-01	Prior Medication	Acetaminophen	500	Mg	2012-02-01
ZZZ101-01	Rescue Medication	Acetaminophen	500	Mg	2012-03-01
ZZZ101-02	Prior Medication	Acetaminophen	1000	Mg	2012-02-15
ZZZ101-02	Rescue Medication	Acetaminophen	1000	Mg	2012-03-15

SDTM VS

USUBJID	**VSTESTCD**	**VSTEST**	**VSORRES**	**VSORRESU**	**VSDTC**
ZZZ101-01	SYSBP	Systolic Blood Pressure	140	mmHg	2012-03-01
ZZZ101-02	SYSBP	Systolic Blood Pressure	130	mmHg	2012-03-15

SDTM QS

USUBJID	**QSCAT**	**QSTESTCD**	**QSTEST**	**QSSTRESN**	**QSDTC**
ZZZ101-01	ECOG	ECOGPS	ECOG Performance Status	1	2012-02-01
ZZZ101-02	ECOG	ECOGPS	ECOG Performance Status	0	2012-02-15

ADaM ADEFF Combining SDTM VS and QS (only a few of the traceability variables are shown)

USUBJID	**VSTESTCD**	**VSTEST**	**VSORRES**	**VSORRESU**	**VSDTC**
ZZZ101-01	SYSBP	Systolic Blood Pressure	140	mmHg	2012-03-01
ZZZ101-02	SYSBP	Systolic Blood Pressure	130	mmHg	2012-03-15
ZZZ101-01	ECOGPS	ECOG Performance Status	1		2012-02-01
ZZZ101-02	ECOGPS	ECOG Performance Status	0		2012-02-15

As a default all columns should be kept instead of forcing data into inappropriate columns.

DON'T AUTOMATICALLY COPY --STRESN INTO AVAL AND --STRESC INTO AVALC

A very common misunderstanding by beginning implementers is to think of the ADaM Basic Data Structure as an SDTM findings domain structure with ADaM variable names. They will note that ADaM BDS has numeric and character analysis value variables AVAL and AVALC, respectively, and they will think that all they have to do to populate these is to copy the seemingly equivalent pair of SDTM standardized result variables ---STRESN and --STRESC. But this is a recipe for problems.

The BDS does bear some superficial similarities to the SDTM Findings class structure. Both are vertical. Both also contain numeric and character data columns. Indeed, often, the BDS is in fact the best structure for ADaM datasets supporting statistical analysis of data derived from SDTM findings domains, e.g. Vital Signs (VS) or Laboratory Test Results (LB). Despite these surface similarities, however, there are many crucial differences. Key among them is that the ADaM numeric and character analysis value variables AVAL and AVALC, respectively, are not equivalent to the SDTM numeric and character standardized result variables --STRESN and --STRESC.

As background, SDTM findings domains function primarily as tabulations of collected study data, whereas ADaM datasets function primarily as containers for observed and derived data organized and ready for statistical analysis and review. ADaM BDS datasets are very general in scope. They may contain observed and derived data for a parameter equivalent to a particular test in a findings domain, but they can also contain data for an entirely derived parameter whose input is from several SDTM domains and classes and/or ADaM datasets, and which has no logical equivalent in any SDTM domain.

However, in this section, we consider the much simpler case where an ADaM parameter represents data from a SDTM findings domain for a given combination of SDTM test and its qualifiers such as specimen type, location, position etc. We will demonstrate that even in this seemingly simple situation, it is quite wrong simply to copy LBSTRESN into AVAL and LBSTRESC into AVALC.

Numeric tests

First, let's consider the case where results are normally numeric. We will use an example of white blood cell count tabulated in the SDTM LB domain.

The focus in SDTM is to report the collected result as a character string in LBORRES, convert if possible the result into standard units and report the standardized character string in LBSTRESC, and then, if LBSTRESC is a number, also place the numeric result into LBSTRESN. To make the example simple, we will assume the result was collected in standard units, so that there is no need to convert units between LBORRES and LBORRESC.

In ADaM, for a numeric lab test, PARAM describes what is in AVAL, and AVAL contains a numeric analysis value. There is no need to populate the character results column AVALC. So one might naturally conclude that to populate numeric analysis value AVAL, one copies from SDTM the numeric result LBSTRESN, which seems to work fine for this result:

AVISIT	PARAMCD	LBORRES	LBSTRESC	LBSTRESN	AVAL	AVALC	DTYPE
Week 8	WBC	5300	5300	**5300**	**5300**		

Now let's consider what happens when a second result is reported as less than the lower detectable limit of the test. For this test, the lower detectable limit is 4000. In this case, the result might be reported by the lab as <4000. Note that only the LBORRES and LBSTRESC columns are populated. The LBSTRESN column is null because to populate it would be to guess (technically speaking, "impute") what the numeric result should be. SDTMIG version 3.1.2 states that in this case, the rules for imputation should be specified in the Statistical Analysis Plan, and the value should only be imputed in the analysis dataset. One common rule, shown here, is to impute the value as one half of the lower detectable limit; however this is by no means the only possible imputation method. For this second record then, one is copying from neither LBSTRESN, nor LBSTRESC. One is deriving the numeric analysis value from LBSTRESC.

AVISIT	PARAMCD	LBORRES	LBSTRESC	LBSTRESN	AVAL	AVALC	DTYPE
Week 8	WBC	5300	5300	5300	5300		
Week 8	WBC	<4000	**<4000**		**2000**		

Finally, the average result over Week 8 might be needed for input to a particular actual analysis. To meet this need, another record is added as follows:

AVISIT	PARAMCD	LBORRES	LBSTRESC	LBSTRESN	AVAL	AVALC	DTYPE
Week 8	WBC	5300	5300	5300	**5300**		
Week 8	WBC	<4000	<4000		**2000**		
Week 8	WBC				**3650**		AVERAGE

For this third record then, one is deriving the analysis value from two other records, which were in turn themselves derived from two different columns in SDTM.

We hope the reader can see that even in such a simple example as a single numeric lab test, AVAL has no single direct equivalent in SDTM. AVAL might be copied from LBSTRESN; or derived from LBSTRESC; or derived from AVAL on other records.

Of course, as always, the WBC parameter value-level metadata for AVAL in this example should describe completely the derivation algorithm for AVAL, which has at least the three branches described.

Character tests

Next let's turn our attention to character tests. We use a simple example of answers to one question on a standard questionnaire. As described in SDTMIG 3.1.2, data from a standard questionnaire may be handled by populating the collected answer text in QSORRES, the standardized score in QSSTRESC, and the score copied from QSSTRESC into QSSTRESN. In ADaM, depending on the need, it will be useful to have the score in AVAL, and the question answer text in AVALC. In this example, AVAL can be used to sort AVALC, and AVALC is a decode of AVAL. Both are helpful for different purposes. They are a one-to-one map within the parameter, as is required by ADaMIG 1.0 when both are populated for a parameter.

Note that AVAL came from QSSTRESN. However, AVALC did not come from QSSTRESC, but rather from QSORRES. So this is another example why it is not correct merely to copy --STRESN into AVAL and –STRESC into AVALC.

AVISIT	PARAMCD	QSORRES	QSSTRESC	QSSTRESN	AVAL	AVALC	DTYPE
Week 12	QS1	VERY BAD	1	1	1	VERY BAD	
Week 12	QS1	BAD	2	2	2	BAD	
Week 12	QS1	GOOD	4	4	4	GOOD	

Now what happens when for input to one analysis, the average over Week 12 is needed? Of course, for this fourth record, which is derived as denoted by DTYPE, AVAL is an average of AVAL from other ADaM records, and is not a copy of QSSTRESN.

AVISIT	PARAMCD	QSORRES	QSSTRESC	QSSTRESN	AVAL	AVALC	DTYPE
Week 12	QS1	VERY BAD	1	1	1	VERY BAD	
Week 12	QS1	BAD	2	2	2	BAD	
Week 12	QS1	GOOD	4	4	4	GOOD	
Week 12	QS1				2.333		AVERAGE

Also note that the one-to-one map between AVAL and AVALC now only holds on the records for the parameter on which both AVAL and AVALC are populated. So technically speaking, this example now violates ADaMIG 1.0, because AVAL and AVALC are no longer a one-to-one map within the parameter (considering that for other subjects and timepoints, null AVALC will map to values of AVALC other than 2.333).

Whether the restriction in the ADaMIG ought to be changed to accept this kind of situation is a subject for debate within the ADaM team, and potentially a matter for clarification in the next version of the ADaM IG. The authors point out that the ADaM Validation Checks version 1.1 does correctly state that the one-to-one mapping requirement applies within each PARAMCD separately, rather than across the entire dataset. However, the first version of the checks document omitted the within PARAMCD requirement. There are several reasons why one might be receiving validation errors related to this check from a validation tool such as OpenCDISC. It could be a tool problem, it could be a validation checks version problem, and it could be that the knowledgeable implementer chooses to deviate as in the above QS example. However, it also could be that the implementer is in the habit of using AVALC as a dumping ground for --STRESC even when not appropriate. As we have seen here, this habit is potentially problematic and ought to be re-examined. In summary, we hope that readers agree that it is quite wrong blindly just to copy --STRESN into AVAL, and --STRESC into AVALC. How and when to populate AVAL and AVALC on a given record should be the subject of careful consideration. The proper derivation algorithms should be based on statistical and clinical knowledge about each parameter and its analyses. The full detail about these algorithms should be reflected in programming specifications and ADaM metadata.

CONSISTENCY CONSIDERATIONS

Throughout the ADaM Implementation Guide there are requirements for consistency between variables. For example, if you create a numeric version of RACE in ADSL called RACEN you are supposed to ensure that the relationship of RACE to RACEN is a one-to-one mapping. This one-to-one mapping requirement also exists for TRTP/TRTPN, SHIFTy/SHIFTyN, AVISIT/AVISITN, ATPT/ATPTN, PARCATy/PARCATyN, as well as for PARAM/PARAMCD/PARAMN and other variables. For some of these variables such as AVISITN, ATPTN, and AVALC there is a stated requirement that the one-to-one relationship is within parameter. However, for others the one-to-one relationship is defined with less precision.

When implementing ADaM you need to take scope into consideration when defining your ADaM variables. For example, when you define PARAM and PARAMCD those are supposed to be one-to-one. How far removed do you take that relationship? Is it one-to-one for a particular dataset only, or for an entire study, or for a development program, or for a company, or for the industry? The further up the ladder that you can make that one-to-one relationship hold the better you will likely be.

ALIGNMENT OF APERIOD AND TRTXXP VARIABLES

ADSL allows for the definition of multiple treatment variables of the variable name format of ‘TRTxxP’ and date variables associated with this treatment, such as TRxxSDT or TRxxEDT. In subsequent BDS datasets, ADaM provides the numeric variable APERIOD that is used to describe a period of time during which the values within a row or a group of rows pertain. A constraint of the ADaM model is that for every xx value of APERIOD that is defined, there must be a corresponding TRTxxP variable and this constraint is also reflected in the ADaM compliance checks. For example, if a BDS dataset has APERIOD=5, then there must be a variable in ADSL defined as TRT05P.

Note that the ADaM model does NOT require that TRT05P be used for the analysis of rows where APERIOD=5 since the particular ADaM treatment variable used to summarize records where APERIOD=5 is trial-specific and sponsor-defined. In most analysis situations, TRT05P would be the treatment variable to use for the analysis of APERIOD=5 record but it is not an ADaM requirement. Other treatment variables, such as TRTP or TRT04A may be used for specified analyses. An example is a situation where TRT04A captures the last double blind treatment received and is used for the analysis of open label records with APERIOD=5.

INDICATOR VARIABLES

Indicator variables can be added to ADSL or BDS structure datasets so long as the ADaM naming conventions are followed. [See “General Flag Variable Conventions” in section 3 of the ADaM Implementation Guide.] Unfortunately sometimes the ADaM naming conventions in section 3 of the ADaM Implementation Guide are forgotten.

For example, let us say that you wanted to add a “Modified Intent-to-Treat” indicator to your ADSL dataset. You might add a variable called MITT for that, but then you are not following ADaM naming conventions. A more proper addition might result in the following ADSL dataset metadata for MITTFL:

Variable Name	Variable Label	Type	Codelist / Controlled Terms
MITTFL	Modified ITT Population Flag	Char	Y, N

Remember also that null values are not allowed for any subject level population flag variable.

Conversely, sometimes people get a bit overzealous in assigning new indicator flag variables to ADaM datasets and they put things in indicator variables that are not indicators. For example, let us say that you want to flag a clinical response as significant or not. So, you create a BDS flag variable called SIGFL to capture that clinical significance like this:

USUBJID	PARAM	AVAL	SIGFL
ZZZ101-01	Clinical Measure	13.3	Y

In this example, SIGFL='Y' indicates clinical significance. This assessment of clinical significance is not so much an indicator variable as it is a categorization of AVAL. Therefore you might want an AVALCAT1 variable here instead of an indicator variable. Using an AVALCAT1 variable might look something like this in the BDS dataset:

USUBJID	PARAM	AVAL	AVALCAT1
ZZZ101-01	Clinical Measure	13.3	Significant

Here the ADaM variable metadata for AVALCAT1 would describe the categorization of AVAL.

ADaM gives you the flexibility to add indicator flag variables to your ADSL and BDS data structures to define new analysis populations. The key is to make sure that you adhere to the indicator naming conventions and try not to put things into indicator variables that belong somewhere else in ADaM such as AVALCAT, SHIFTy, or PARCATy for instance.

CRITY* VARIABLES ARE NOT GENERAL PURPOSE FLAGS

Version 1.0 of the ADaMIG does not clearly state the scope of criteria variables CRITy, CRITyFL and CRITyFN. These variables are used to indicate when criteria are satisfied about analysis values, e.g. AVAL, CHG, PCHG and other functions of AVAL and BASE (or AVALC and BASEC). They should not be used for flagging other unrelated kinds of things, like analysis period. In those other cases, other kinds of flags should be used.

The table below shows an **incorrect** use of the CRITy* variables to indicate whether data is pretreatment or not. This criterion is solely a function of ADY and has nothing to do with AVAL or AVALC. Therefore it is an inappropriate use of the criterion variables.

ADY	CRIT1	CRIT1FL
-28	Pre-Treatment	Y
-14	Pre-Treatment	Y
1	Pre-Treatment	
14	Pre-Treatment	
28	Pre-Treatment	

This table shows how this situation can be **correctly** handled using a flag variable and not the CRITy* variables. One is not restricted to the flags specified in the ADaMIG. Where necessary, other flags can be created.

ADY	PREFL
-28	Y
-14	Y
1	
14	
28	

CAN I JUST ADD ANOTHER COLUMN TO MY BDS DATASET?

There are many kinds of columns that can be added freely to BDS datasets, for example, timing variables, population and other flags, model covariates and treatment variables, and variables useful for traceability or supportive of review. However, when it comes to analysis values, which are generally speaking the kinds of "dependent variable" data that appear on the left hand side of a MODEL statement, like AVAL, CHG, etc., there are rules. Section 4.2 of the ADaM Implementation Guide describes these rules. For analysis values, the sole rule that permits adding a new column is, “Rule 1. A parameter-invariant function of AVAL and BASE on the same row that does not involve a transform of BASE should be added as a new column.” So in general, you will more likely than not be adding new rows or PARAMs to your BDS datasets instead of new variables. The problem is that inherently people like to add new columns to their BDS datasets.

For example, let us say that you wanted to add a cumulative pain score to your BDS analysis dataset and call that variable CSCORE like this:

USUBJID	AVISIT	PARAM	AVAL	CSCORE
ZZZ101-01	Week 1	Pain Score	7	7
ZZZ101-01	Week 2	Pain Score	3	10
ZZZ101-01	Week 3	Pain Score	2	12

Adding that CSCORE variable seems intuitive enough and people are doing it, but it is in violation of Rule 4 of the Implementation Guide that states, “A function of multiple rows within a parameter should be added as a new parameter.” So you would need to add the cumulative score like this in the BDS structure to be ADaM compliant:

USUBJID	AVISIT	PARAM	AVAL
ZZZ101-01	Week 1	Pain Score	7
ZZZ101-01	Week 2	Pain Score	3
ZZZ101-01	Week 3	Pain Score	2
ZZZ101-01	Week 1	Cumulative Pain Score	7
ZZZ101-01	Week 2	Cumulative Pain Score	10
ZZZ101-01	Week 3	Cumulative Pain Score	12

When you are going to add information either with new rows or columns in your BDS analysis datasets it is important to review the six rules and the many examples provided in section 4.2 of the ADaM Implementation Guide. You will find that more often than not that you need to add new rows instead of columns to your BDS structure datasets, so fight the urge to automatically add new columns. Adherence to these rules maintains compliance; without the rules, we would have no standard.

THE PURPOSE OF DTYPE

It is not uncommon in statistical analysis to impute missing observations for a subject for a given parameter. For example, suppose the primary efficacy measure is Forced Expiratory Volume (FEV) observed at the last clinical visit yet a subject discontinued the study prior to the last visit. In this case, there are various ways to estimate what the subject’s FEV measure might have been at this visit and by estimating this value, this subject can then be included in analyses of data from this last visit. Often the statistical analysis plan will specify the summary of the data both ‘as observed’, which uses only those values that are truly collected data and ‘observed and imputed’, which uses both observed and imputed values. Therefore, a way to distinguish which observations were observed and which ones were imputed is important. Indeed, in the first CDISC/FDA pilot project, the FDA indicated that it is very helpful for them to identify observations that were collected on the CRF versus those that were imputed.

In the ADaM BDS model, the variable DTYPE was defined to be used in this situation. It is to be used to identify records **within a given parameter** that have been derived and the value of DTYPE indicates the method used for derivation. The metadata for DTYPE will give further information about the details of any algorithm or statistical method used to derive these imputed values. Using the presence or absence of DTYPE will allow a user to summarize the data in different ways. For example, to summarize ‘data as observed’ records, then the select statement “WHERE PARAMCD=’FEV’ and DTYPE=’ ‘“ will return just the records with observed values of FEV. Alternatively, using the select statement “WHERE PARAMCD=’FEV’” will return all records, both collected and imputed.

It is important to use DTYPE appropriately for identifying derived records within a parameter that represents data that could have been collected. If a parameter is wholly derived, such as a Time to Event parameter, then it is a misapplication to populate DTYPE because, by definition, all records are derived, and not derived from other records within the same parameter. Using PARAMTYP=’DERIVED’ in this case is an option if the user wants to identify parameters that are derived versus observed.

TIME TO EVENT IN THE BASIC DATA STRUCTURE

It is expected that by the time this paper is presented, “The ADaM Basic Data Structure for Time-to-Event Analyses” appendix to the ADaM Implementation Guide will have been published. That document is a comprehensive guide that will help you to implement time to event analyses using the ADaM Basic Data Structure. In the meantime we would like to show a case where the implementation of time to event analysis in the Basic Data Structure somewhat stretches the intent or design of the Basic Data Structure.

Let us say that you want to do a time to event analysis where your event of interest is indication of liver damage which is indicated either by an adverse event of jaundice or an elevated ALT or AST lab result. If the event did not occur, then you want to censor the subject at end of study or last follow-up. Here is an example of how we saw this modeled in the basic data structure for one patient. CNSR=0 when the event occurs and CNSR=1 when it does not.

PARAMCD	ADT	AVAL	CNSR	EVNTDESC	DTYPE	SRCDOM	SRCVAR	SRCSEQ
AEDT	06JUN2009	65	0	Jaundice		AE	AESEQ	235
LBDT	19JUL2010	128	0	Elevated AST		LB	LBSEQ	1001
LIVERDAM	06JUN2009	65	0	Jaundice or Elevated AST	TTE			

There are a few problems with this implementation of the Basic Data Structure.

- PARAM is supposed to describe the contents of AVAL. Admittedly, PARAMCD has only 8 characters; however, AEDT or LBDT above makes it sound like AVAL is a date, whereas actually it appears to be a relative day. If these rows are included for traceability or analysis, perhaps consideration could be given to choosing apt PARAM and PARAMCD names that describe AVAL well. Likely candidates for PARAM/PARAMCD could be ‘Time to Jaundice (days)’/TTJAUND and “Time to Elevated AST (days)’/TTEAST.
- The DTYPE variable is intended to indicate rows where an AVAL is derived within a given PARAM. Since LIVERDAM is an entirely derived parameter setting DTYPE=TTE is not informative and is a misuse of DTYPE. The purpose of DTYPE is to both distinguish which records were observed versus those that were derived from other records within the parameter and to indicate the method of derivation.
- EVNTDESC should be “Jaundice” and not “Jaundice or Elevated AST” for the LIVERDAM parameter since that is the event that caused the record to be generated.
- It appears that for PARAMCD values of “AEDT” and “LBDT” that these records were included as traceability records to help the reviewer see where the event dates come from for PARAMCD of “LIVERDAM”. However, unless those records are themselves being used as survival analysis endpoints, they shouldn’t have CNSR and EVNTDESC values populated for them. Also, if you were going to try and include the event dates in the Basic Data Structure you also would want the censor date as well for completeness and not just the records where a triggering event happens. In the above case that would mean including the end of study date or date of last follow-up.

The following table shows how we would represent the above data in the Basic Data Structure:

PARAMCD	ADT	AVAL	CNSR	EVNTDESC	DTYPE	SRCDOM	SRCVAR	SRCSEQ
LIVERDAM	06JUN2009	65	0	Jaundice		AE	AESEQ	235

Here you can see that the traceability is maintained by referencing the jaundice adverse event back to the AE domain via the use of SRCDOM, SRCVAR, and SRCSEQ variables. Also note that DTYPE is null here as LIVERDAM is an entirely derived parameter. In general, time to event datasets do not need the DTYPE column because it rarely occurs that time to event records are derived from other records within the same parameter.

CONCLUSIONS

The authors hope that this paper proves useful to other fellow implementers of the ADaM Implementation Guide, and look forward to further experience sharing, knowledge exchange, and discussions about practical issues among the community of users, implementers, and developers. Such discussion can only help in efforts to improve implementation practice, the clarity of the Implementation Guide, and the standard itself.

CONTACT INFORMATION

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

Nate Freimark
nate.freimark@TheoremClinical.com

Susan Kenny
susan.kenny@amgen.com

Jack Shostak
jack.shostak@duke.edu

John Troxell
jktroxell@gmail.com