PharmaSUG 2015 - Paper PO13

Leveraging ADaM Principles to Make Analysis Database and Table Programming More Efficient

Andrew L Hulme, PPD, Kansas City, MO

ABSTRACT

The purpose of the CDISC Analysis Data Model (ADaM) is to create an analysis database that is both traceable to source data and analysis-ready. In most studies, the source data will consist mostly of SDTM domains and thus it may seem practical to create an analysis database that is only an extended version of SDTM. A method like this will tend to neglect the analysis-ready principle of ADaM, specifically that ADaM domains should be designed using the planned table outputs such that analyses can be recreated from the analysis database with minimal additional programming. Using ADaM principles, this paper discusses several techniques that can increase the efficiency of analysis database programming and make table programming simpler. Examples include storing of treatment information in ADSL, date imputation, creating new records in Basic Data Structure (BDS) domains for new analysis parameters, and deriving specific ADaM variables to prevent inefficient rework in downstream programs.

INTRODUCTION

The primary goal of the Analysis Data Model (ADaM) is the production of an analysis database that adheres to the following two principles:

1. Traceability: meaning traceable to the source data. Variables in the analysis database should have a clear link to SDTM or other source data and results in the statistical outputs should have a clear link to the analysis database via either descriptions of how something is derived in metadata (e.g. define.xml or Results Level Metadata) or by storing links to source data in the data itself (e.g. sequence variable). In the programming process, traceability manifests itself through analysis database specification documents and table shells. A project with high quality traceability will have clear specification documents allowing programming to be done quickly and efficiently.
2. Analysis-ready: meaning the analysis database should be setup in such a way that statistical results can be generated from the data with little additional programming. Stated another way, the results displayed in table should be generated using only a single SAS procedure on the input analysis domain, a principle known as "one PROC away". Striving for this on a project leads to simpler table programs that are easier to validate and allows reviewers to recreate the outputs in only a single step and without complex data manipulation.

Since the CDISC Analysis Data Model (ADaM) version 2.1 and ADaM Implementation Guide (hereafter ADaMIG) were released in 2009, a plethora of information has been published regarding the ADaM standard. There are several excellent papers detailing the rules of the model (Brucken and Slagel 2010, Freimark et al. 2012, Barrick and Troxell 2014) and how the data should be submitted to the FDA (Jain and Minjoe 2014). These papers offer a multitude of suggestions for creating ADaM-compliant data sets using the traceability and analysis-ready principles.

An underrepresented topic concerning ADaM, however, is how it relates to the production of tables and figures that summarize the statistical results of a study. These statistical outputs are one of the most important end products of a clinical trial and figure prominently in the Case Study Report (CSR) which is submitted to regulatory agencies. By considering the ADaM principles of traceability and analysis-ready, this paper will outline several techniques that make the table production process simpler and more straightforward, thereby reducing the amount of effort required to produce statistical outputs.

WHY USE ADAM PRINCIPLES AND STANDARDS?

During the programming process, CDISC-standardized databases (SDTM and ADaM) are primarily reviewed by other database programmers and statisticians. Tables on the other hand are consumed by a much wider audience with varying areas of expertise including medical writers, physicians, and therapeutic area experts. Sometimes these additional reviewers do not begin reviewing the tables until much later in a project life cycle. Hence, the table production process from source data to statistical outputs can introduce significant risk to a study's quality and timeliness if it is not carefully and proactively designed. To combat these risks, the process needs to be robust enough such that initial mistakes are minimized and later adjustments are able to be made quickly and efficiently.

For table production, three steps are required (Figure 1). First, several complex programming actions are performed to manipulate the source data into a data set that can be used for analysis. This step includes combining multiple source data sets, mapping source variables into the required structure, and deriving new variables or parameters. Next, results are calculated as required for the table output usually by a SAS procedure. Finally, the results are formatted to enable display and the tables are output.

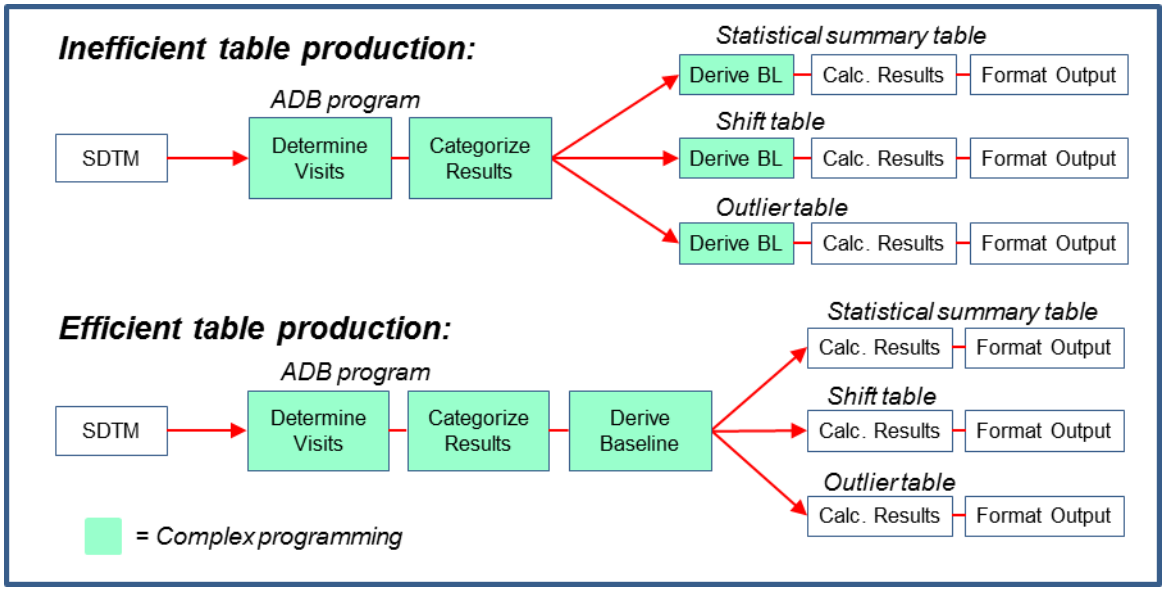

Figure 1. Two possible schemes of table production. In the first example, baseline is derived in the table programs. This is inefficient because the derivation is done more than once and any change to the derivation must be done in all three table programs. In the second example, baseline is derived in the analysis database (ADB) program. This will be faster and the baseline derivation is ensured to be consistent across table outputs.

To maximize efficiency in this process, each unique programming action should be performed exactly once. This ensures that all calculations in a study will be internally consistent, and that if any changes are made they only need to be made in one place. Since multiple tables will typically be generated from a single analysis domain, the amount of programming in a table program should be limited to only the calculation and formatting steps to avoid having derivations in multiple places. In other words, all complex manipulations of the source data should be done in the analysis database (ADB) programs.

Minimizing the amount of programming in the table programs also has the following benefits:

1.) Traceability is increased because all important derivations are handled in ADB programs with a clear description of the method in the metadata. If variables are derived in the table program it will be unknown to a reviewer how the variables were programmed. Also, since analysis domains will be programmed before tables, discussions on what derivations are needed for the tables occur earlier in a project's life cycle.

2.) Table specifications (annotated table shells or results-level metadata) are more succinct because only a simple list of the variables required for the analysis and the selection criteria are needed.

3.) Tables are consistent because it is much easier to ensure that the same selection criteria are being used for tables that display the same data differently, e.g. statistical summary and shift tables.

4.) Unnecessary programming is not repeated in multiple table programs improving consistency and run times. Deriving a parameter once in the analysis database ensures that it is consistently presented in the tables and edits to the derivation are only made in one place.

5.) Tables are easier to validate since table programs are simpler and all derivations have already been done in ADB programs. SAS has procedures to compare data sets during double programming, but comparing tables is far more laborious and differences are harder to detect.

6.) Reviewers of a submission to a regulatory agency can quickly recreate an analysis from the analysis database by only knowing the variables used for the analysis and applying a simple SAS procedure.

The above benefits can be accomplished without using CDISC, but additional efficiencies are realized by using ADaM standards. The chief strength of adopting ADaM standards is that alike variables will always have the same variable name regardless of the source data used to create the analysis domain. Because of this, analysis domains will have similar structures and, since all derivations will be done in the ADB programs, a few simple table programs can be used to generate a large percentage of the tables for a study. If ADaM standards are adopted companywide, the same table programs can be used across multiple studies with very little modification. An additional benefit of using ADaM standards is that it enables the use of readily-available validation applications like OpenCDISC Validator.

By now, it is hopefully evident that implementing ADaM principles and standards can greatly reduce the amount of effort required to produce tables. While it is relatively straightforward to produce data sets that are ADaM compliant, the benefits discussed here are not fully realized unless the data sets are proactively designed with the end result, the tables, in mind. The remainder of this paper suggests some ideas to make the analysis database and table programming process more efficient by examining the topics of deriving treatment variables in ADSL that can be used by all other analysis domains, imputing dates/times to the same precision across all analysis domains, and designing analysis domains by working backward from the table shells.

EFFICIENT USE OF TREATMENT VARIABLES IN ADSL

ASSIGN TREATMENTS USING ANALYSIS PERIOD VARIABLES

The purpose of ADSL is to contain all subject-level variables for a subject that may be in multiple other analysis domains including population flags, treatment information, and analysis covariates. ADSL must contain exactly one record per subject to allow for easy merging with all other analysis domains and it must be created for CDISC submissions (ADaMIG Sections 2.3.1 and 3.1).

Treatment is the most important variable used to summarize the effect of a drug. Hence, the most important set of variables for use in tables are those related to the treatment assigned/given to a subject and over what duration the treatment is administered. To maintain consistency and efficiency, treatment variables should be derived once in the ADSL program and stored in the ADSL standard variables (Appendix 1) for type of treatment (TRT01P/TRT01A) and treatment duration (TR01SDTM and TR01EDTM). For studies where multiple treatments are given to an individual subject, multiple sets of treatment variables should be stored in ADSL with the index in the variable name incremented by one (e.g. TRT02P, TRT03P, and so on).

When assigning data points to a particular treatment, however, the duration a subject is considered on treatment for analysis reasons will typically not correspond to the actual treatment duration. For instance, adverse events that occur a few days after the end of treatment may still be considered treatment-emergent. In this situation, the analysis period set of variables APxxSDTM and APxxEDTM should be included in ADSL to store the dates/times a subject is considered under the treatment TRTxxP.

(a) EX

USUBJID	EXTRT	EXSTDTC	EXENDTC
ABC-001	A	2013-05-01T12:30	2013-05-04T12:30
ABC-001	B	2013-05-08T12:30	2013-05-11T12:30

(b) AE

USUBJID	AETERM	AESTDTC	AESTDTM	AESTTMF	APERIOD	TRTP
ABC-001	Nausea	2013-05-01	2013-05-01T23:59	H	1	A
ABC-001	Headache	2013-05-06T06:00	2013-05-06T06:00		1	A
ABC-001	Hypokalemia	2013-05-12T14:30	2013-05-12T14:30		2	B

(c) LB

USUBJID	LBTEST	LBSTRESN	VISIT	LBDTC	LBNRIND	ADTM	ATMF	TRTP
ABC-001	Potassium	3.9	Visit 1	2013-05-01		2013-05-01T00:00	H	
ABC-001	Potassium	3.8	Visit 2	2013-05-05		2013-05-05T00:00	H	A
ABC-001	Potassium	3.6	Visit 3	2013-05-08		2013-05-08T00:00	H	A
ABC-001	Potassium	3.1	Visit 4	2013-05-12	LOW	2013-05-12T00:00	H	B

(d) ADSL

USUBJID	TRT01P	TRT02P	TR01SDTM	TR01EDTM	TR02SDTM	TR02EDTM
ABC-001	A	B	2013-05-01T12:30	2013-05-04T12:30	2013-05-08T12:30	2013-05-11T12:30
USUBJID			**AP01SDTM**	**AP01EDTM**	**AP02SDTM**	**AP02EDTM**
ABC-001			2013-05-01T12:30	2013-05-08T12:30	2013-05-08T12:30	2013-05-18T12:30

Table 1. Sample SDTM data for (a) EX, (b) AE, and (c) LB. Variables created in analysis domain shaded in gray. APERIOD excluded from LB display. (d) One record in sample ADSL split onto two lines for display.

Due to the one record per subject structure of ADSL, the analysis period dates can be merged onto the remaining analysis domains and each on-treatment record can be assigned consistently to an analysis period (APERIOD) and treatment (TRTP). This calculation is done by comparing the analysis date of each observation with the range of each analysis period defined by APxxSDTM and APxxEDTM. For dates that fall into an analysis period "xx", APERIOD is set to the number "xx" and TRTP is set to TRTxxP. Simple code using arrays can do this as shown below for a study with three treatment periods:

```
data ADLB2;
  merge ADLB1 ADSL;
  by studyid usubjid;

  array apst{3} ap01sdtm ap02sdtm ap03sdtm; **start date variables;
  array apen{3} ap01edtm ap02edtm ap03edtm; **end date variables;
  array trt {3} trt01p trt02p trt03p;       **treatment for period;

  do i=1 to 3;
    if apst{i} <= adtm < apen{i} then do;
      trtp=trt{i};
      aperiod=i;
    end;
  end;
run;
```

Note that for clear mapping to analysis periods, the analysis period dates in ADSL must not overlap and should contain as much precision as possible.

Deriving an APERIOD and TRTP using this method for all on-treatment observations in a study is helpful for several reasons. First, it makes analysis domains analysis-ready for tables that summarize on-treatment results (add APERIOD to subset) or are summarized by treatment at onset (add TRTP to covariates). Second, the rules for deriving the analysis period dates are clearly noted in ADSL metadata and these dates are traceable to source treatment data. Lastly, if adjustments to the analysis period derivation are required, code only has to be modified in ADSL.

As an example, consider the data presented in Table 1 containing a few SDTM domains for a subject participating in a crossover trial named study ABC. Deriving TRT01P, TRT02P, TR01SDTM and TR02SDTM for ADSL is relatively easy as these can be directly copied from the EX records. However, the SAP for study ABC says to consider a subject on a given treatment from treatment start until the either the next treatment starts or seven days after the treatment ends. For subject ABC-001, analysis period 01 begins at TR01SDTM and ends at TR02SDTM (when treatment 02 begins). Analysis period 02, being the last analysis period, begins at TR02SDTM and ends at TR02EDTM plus 7 days (Table 1d).

The variables APERIOD and TRTP are then derived for each observation in AE and LB by comparing the observation date/time (see next section for date imputation discussion) with the start and end of each period. For example, APERIOD is set to 2 and TRTP is set to B for the Hypokalemia AE record since AESTDTM is between AP02SDTM and AP02EDTM. The subject also has a related LB record for a low potassium result. By using the analysis period dates in ADSL, it is guaranteed that the result is assigned to the same analysis period and treatment as the related adverse event.

IMPUTE DATES TO SAME PRECISION AS TREATMENT DATES

In SDTM, dates/times are stored in ISO8601 format to allow depiction of partial dates. This introduces complications when partial dates are part of an analysis computation that requires full dates/times. To help with this, the preferred method of storing dates/times in ADaM is to convert the dates/times to integers and apply SAS date and time formats (ADaMIG Section 3 General Timing Variable Conventions). This principle suggests that it is beneficial for all dates/times used for analysis to be imputed and stored in the relevant numeric variable (*DT, *TM, *DTM).

While this may seem like extra work, imputing all dates makes table programming more efficient in two ways. First, the rules for date imputation will be listed in the ADB metadata and imputed dates will be flagged using the *DTF/*TMF flags for greater traceability (Timing Variable Convention #6). Second, programming can use more straightforward logic like simple equality relations instead of multiple conditions based on the various lengths of the dates being compared. In other words, partial dates are imputed first and then all dates are compared using the same relation, rather than doing separate comparisons based on the length of the dates. Additionally, imputed dates will be stored in the analysis domain for easy reference or reuse in other analysis domains.

For the study with adverse event data listed in Table 1b, the SAP for study ABC says to consider AEs treatment-emergent if they occur between the date/time of first dose and the date/time of last dose plus 7 days. If a time is not collected for an AE, the event should be considered treatment emergent if it occurs on the same day as first dose. Subject ABC-001 has an event that meets this criteria (date part of AESTDTC = date part of TR01SDTM). Now, logic could be added to the program to account for this specific case by checking the length of AESTDTC but this would make it unclear that any imputation had been done and AESTDTM would be missing unlike other records with full dates/times. Instead, AESTDTM should be imputed with a time that will always be after the date/time of first dose (e.g. AESTDTC concatenated with 23:59) and the imputation flag AESTTMF should be set to "H". After doing this for all records with partial dates, all treatment-emergent AEs will have a non-missing AESTDTM that is after TR01SDTM.

Note that imputation rules should be drafted carefully as they may differ based on when the partial date occurs in an interval (i.e. in the example above a partial date at the end of treatment would need a different rule).

For maximum efficiency, it makes sense to impute to a full date-time (*DTM) even when the dates/times are not technically partial. This is particularly helpful when dosing time points are recorded as dates and times, but a site is not required to collect time for other study measurements.

As an example, times were not collected for the lab data in Table 1c. The protocol for study ABC says that labs should be collected pre-dosing on Visits 1 and 3. While it may seem that an imputation is unnecessary, leaving the dates/times as partial would cause them to be stored differently than other dates/times in the analysis database. Instead, a time should be chosen that would be before first dose at those visits (i.e. 00:00) and the same derivation of APERIODC/TRTA can be used for all domains.

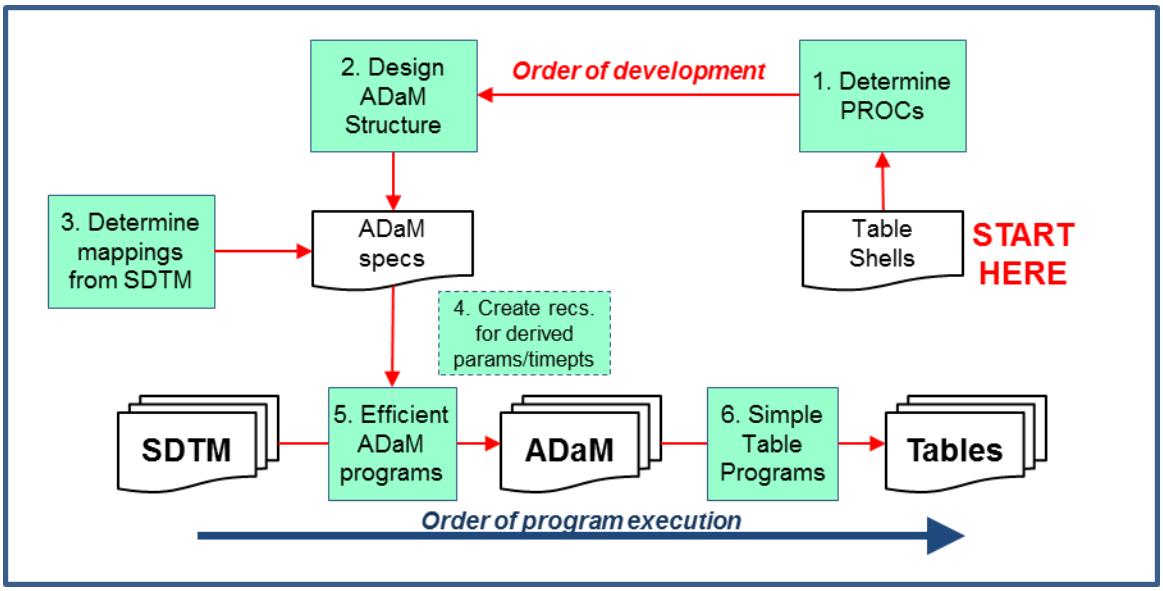

Figure 2. Schematic detailing the table production process described in this paper.

DESIGNING ANALYSIS-READY ADAM DOMAINS USING THE BDS STANDARD

In a typical study, the order of program execution is SDTM, then ADaM, and finally table programs. While it may seem like this should also be the order in which specifications and programs are developed, the order needs to be reversed to design analysis-ready ADaM domains that enable generation of table results with a single SAS procedure (Figure 2). This section describes step-by-step the optimal way to develop ADaM domains for table production. The process begins with inspecting table shells to determine what is required for the tables, then the analysis-ready ADaM structure is designed, next mappings from the source SDTM data to ADaM are determined and finally efficient ADB and simple table programs are written.

STEP 1. DETERMINE SAS PROCEDURE AND VARIABLES REQUIRED TO PRODUCE TABLES

The first step for designing analysis domains should be to examine the table shells and determine what SAS procedures will be used to create them. A list of variables should be made detailing which variables will be analyzed, which variables the analysis will be grouped by, and which variables will be needed to subset the input data for the analysis. Doing this first, ensures that the analysis domains will be analysis-ready and contain all of the information required to produce the table results with a single SAS procedure.

Study BCD Table 2. Summary of Laboratory Parameters (Population: Safety)				
Parameter	**Visit**	**Statistic**	**Treatment A (N=XX)**	**Treatment B (N=XX)**
Potassium (mmol/L)	Baseline	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
	Week 1	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
	Week 2	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
	Week 4	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
Baseline is last value prior to start of treatment.				

Table 2. Sample table shell for a typical statistical summary table. Only one of many parameters is shown.

To illustrate this, consider the table shell displayed in Table 2. The MEANS procedure can be used to calculate the summary statistics with the variable containing the lab measurements on the VAR statement. Variables for treatment, parameter, and visit need to be placed on the BY statement to group the analysis and variables used to select the records for the analysis need to be included on a WHERE statement. Basically the code will look like this:

```
proc means data=ADEG;
  var numeric-value;
  by treatment-variable parameter-variable visit-variable;
  where population-flag="Y" and visit-variable in (list-of-visits);
run;
```

The data set that is input into this procedure at a minimum needs to contain all of the variables in the VAR, BY, and WHERE statements and have a structure of one record per subject per parameter per visit.

STEP 2. FIT REQUIRED VARIABLES INTO ADAM BDS STRUCTURE

The next step is to fit variables required for each table into an analysis-ready database structure that can produce all table results in one step. The ADaM Basic Data Structure (BDS) can be used to accommodate all basic statistical summary tables of numeric variables and most frequency tables of categorical variables (ADaMIG Sections 2.3.2 and 3.2). The variables AVAL and/or AVALC should be used to contain the values of the parameter being summarized. A description of this parameter should be stored in PARAM and visit information should be stored in AVISIT or ATPT. Treatment information can be merged in from ADSL (along with other analysis covariates like gender) or derived in the ADB program as TRTP. See Appendix 2 for further information on BDS standard variables.

Returning to the table shell displayed in Table 2, it is easy to determine which variables need to be created in the ADaM domain to allow the table results to be generated using only the procedure displayed in step 1. The structure required is presented in Table 3. The variable AVAL contains the numeric laboratory result, AVISIT contains the text to be displayed for the visits, and PARAM contains the name of the lab test. Variables TRT01P and SAFFL are included as well and will need to be merged in from ADSL.

USUBJID	PARAM	AVISIT	AVAL	TRT01P	SAFFL
BCD-XXX	Potassium (mmol/L)	Baseline	X.X	A or B	Y
BCD-XXX	Potassium (mmol/L)	Week 1	X.X	A or B	Y
BCD-XXX	Potassium (mmol/L)	Week 2	X.X	A or B	Y
BCD-XXX	Potassium (mmol/L)	Week 4	X.X	A or B	Y

Table 3. Planned structure from the ADaM BDS domain required to produce Table 2. Variables PARAMCD and AVISITN have been withheld for brevity but should be included.

Once the structure of one planned BDS domain has been designed, the remaining BDS domains for the entire study should be assembled similarly to achieve maximum efficiency. This will allow simple table programs to be used repeatedly for table outputs generated from many different source analysis domains (i.e. table macros can be used) and domain-specific derivations will be limited to the ADB programs.

Note that while the ADaM BDS standard is most similar to the SDTM Findings standard, BDS structures are not limited for use with only analysis domains derived from Findings domains. Any table output, regardless of source, that is a summary of numeric or categorical variables can be generated from domains using a BDS structure. For instance, severity data taken at each visit for pre-specified adverse/clinical events can be mapped into a BDS domain (AVALC = AESEV). Even parameters derived from several records over an entire study can be fit into BDS, like exposure summary variables (AVAL = number of doses administered, PARAM = Doses administered).

Lastly, determining the structure of the ADaM domains from the table shells is particularly helpful when records need to be created for methods like last observation carried-forward (LOCF) or if imputation of missing visits is required. This forces a programmer to determine what the data should contain for each subject before considering what the source data contains. Doing this first, will reduce mistakes and increase the likelihood that the first draft of the analysis domain will contain all of the data that is required for the tables.

STEP 3. DETERMINE MAPPINGS FROM SDTM TO PLANNED ADAM DOMAINS

At first glance, there seems to be many variables in the ADaM BDS standard that correspond directly to similarly-named variables in the SDTM Findings standard. Thus, it may seem desirable to simply map SDTM variables into their similarly-named ADaM counterparts and then figure out how to program tables from there.

Doing this, however, ignores the fact that the required structure of the analysis data needed to produce the tables may vary substantially from how the data is collected on the CRF and stored in SDTM. Additionally, the values needed for the tables for something like visits may not match those that are used in SDTM, nor may there even be a one-to-one mapping of these values.

Hence, the best way to proceed after determining the ADaM structure is to work backwards deciding which SDTM data is needed for each record in the planned ADaM structure. This may result in some SDTM records not having fully populated data (e.g. missing AVISIT) when mapped into ADaM, but this is acceptable because it is typical that some SDTM records will not be summarized in all of the tables.

LB

USUBJID	LBTEST	LBSTRESN	VISIT	LBDY	LBSTAT	LBBLFL	AVISIT
BCD-011	Potassium	3.7	Visit 1	-3			
BCD-011	Potassium	3.8	Visit 2	1		Y	Baseline
BCD-011	Potassium	4.2	Visit 3	8			Week 1
BCD-011	Potassium	4.1	Visit 4	15			Week 2
BCD-011	Potassium	4.1	Visit 5	29			Week 4
BCD-012	Potassium	3.6	Visit 1	-3		Y	Baseline
BCD-012	Potassium	.	Visit 2	.	NOT DONE		
BCD-012	Potassium	3.5	Visit 3	8			Week 1
BCD-012	Potassium	3.0	Visit 4	15			Week 2
BCD-012	Potassium	3.1	Unscheduled	18			
BCD-012	Potassium	3.8	Visit 5	29			Week 4

Table 4. Sample SDTM data for two subjects in study BCD that will be included in Table 2 and derived AVISIT (shaded gray) to be included in the analysis domain.

Table 4 contains sample SDTM data for two subjects participating in study BCD that are to be included in Table 2. The table requires a summary of the baseline results for each subject where baseline is the last measurement prior to first treatment. In most cases, lab measurements are taken a few minutes before the subject starts study treatment during Visit 2, but this measurement was not done for subject BCD-012.

The simplest way to map the SDTM data would be to do a one-to-one mapping between VISIT and AVISIT (Visit 2 to Baseline, Visit 3 to Week 1, etc.), but this would cause subject BCD-012 to have no baseline record for Table 2. Perhaps, the baseline flag LBBLFL could be used by the table program to identify the baseline records for all subjects, but the analysis domain would no longer be analysis-ready and extra inefficient programming would be required in the table program.

Thus, the best solution is to include programming in the ADB program that sets the LBBLFL=Y records to have an AVISIT of Baseline, maps the scheduled on-treatment visits to AVISIT accordingly, and leaves additional pre-treatment or unscheduled visits with a missing AVISIT. This derivation will be listed in the ADB metadata for traceability to the source data and the table results can be generated using only the SAS procedure in step 1.

STEP 4. CREATE NEW RECORDS FOR DERIVED TIMEPOINTS OR PARAMETERS

An intricate feature of ADaM BDS is the suggestion to generate new records for values that are derived from a function of multiple other records instead of adding new columns/variables (ADaMIG section 4.2.1). Adding new non-BDS standard variables is inefficient because it makes the analysis database less uniform across a study and makes it less clear what records and variables should be used for the table outputs.

Study BCD Table 5. Summary of ECG Parameters (Population: Safety)				
Parameter	**Visit**	**Statistic**	**Treatment A (N=XX)**	**Treatment B (N=XX)**
QT Duration (msec)	Baseline	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
	Week 4	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
RR Duration (msec)	Baseline	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
	Week 4	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
QTcB [1]	Baseline	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
	Week 4	n Mean (SD)	XX X.XX (X.XXX)	XX X.XX (X.XXX)
Displayed results are the averages of the three results recorded at each visit. Baseline is last value prior to start of treatment. [1] QTcB = QT in seconds divided by square root of RR in seconds.				

Table 5. Sample table shell for a typical statistical summary table. Only one of many parameters is shown.

(a) Inefficient analysis domain for ECG data

USUBJID	EGTEST	EGSTRESN	VISIT	EGTPTNUM	EGAVG	QTCB
BCD-011	QT Duration	356	Visit 5	1	355	420
BCD-011	QT Duration	358	Visit 5	2	355	416
BCD-011	QT Duration	351	Visit 5	3	355	411
BCD-011	RR Duration	717	Visit 5	1	730	
BCD-011	RR Duration	739	Visit 5	2	730	
BCD-011	RR Duration	734	Visit 5	3	730	

(b) Efficient analysis domain for ECG data

USUBJID	PARAM	AVAL	AVISIT	ATPTNUM	DTYPE	PARAMTYP
BCD-011	QT Duration (msec)	356	Week 4	1		
BCD-011	QT Duration (msec)	358	Week 4	2		
BCD-011	QT Duration (msec)	351	Week 4	3		
BCD-011	QT Duration (msec)	355	Week 4	99	AVERAGE	
BCD-011	RR Duration (msec)	717	Week 4	1		
BCD-011	RR Duration (msec)	739	Week 4	2		
BCD-011	RR Duration (msec)	734	Week 4	3		
BCD-011	RR Duration (msec)	730	Week 4	99	AVERAGE	
BCD-011	QTcB	420	Week 4	1		DERIVED
BCD-011	QTcB	416	Week 4	2		DERIVED
BCD-011	QTcB	411	Week 4	3		DERIVED
BCD-011	QTcB	415.6667	Week 4	99	AVERAGE	DERIVED

Table 6. (a) One possible way to derive additional parameters in an analysis domain, added variables (shaded gray) are repeated on multiple records and are not in the BDS standard. (b) ADaM compliant data set with new rows created for new time points and parameters.

Instead, new rows should be created whenever the following two cases occur. First, a value is needed that is a function of other time points under the same parameter, for example an average of two visits or maximum value during treatment. In this case, AVISIT can contain the description of the time point (AVISIT is not limited to protocol-defined visit names) and a simple description of the derivation can be included in DTYPE. Second, a value is needed that is a function of other parameters typically at the same time point, for example body mass index. For these records, a new parameter (PARAM) will be defined and PARAMTYP will be used to label it.

Consider Table 5, another table that is to be created for study BCD that summarizes ECG parameters. The table requires that averages be calculated for ECG results that are collected three times in quick succession at each visit (i.e. triplicates). It would be rather easy in a single SQL procedure call to create a new column containing the average value and add it to each of the three component records (EGAVG in Table 6a). However, this method would

make it unclear which value should be analyzed in the table (EGSTRESN or EGAVG) and additional table programming would be required to select only one of the triplicate records.

Instead, a new record should be created for the visit with AVAL containing the averaged value of the triplicate measurements (Table 6b). The created record must have the same value of PARAM as the record it was derived from, but it will be flagged with DTYPE=AVERAGE. Now when generating the table results, the standard variable AVAL is the value to be summarized and, since the BDS database structure is maintained, a simple table program can still be used. In fact, the program should be very similar to the one used to generate Table 2 but with the addition of DTYPE=AVERAGE to the selection criteria.

Table 5 also requires that a new parameter QTCB be derived for each visit. This parameter is a function of the QT and RR intervals (namely QT divided by the square root of RR). Records will be created for each time point and then the triplicate average will be calculated (Table 6b). Since the new parameter is derived from other records with different values of PARAM, a new value of PARAM will be created and PARAMTYP should be set to DERIVED. The derivation for the new parameter should be noted in the metadata to provide traceability to the source records.

STEP 5. WRITE EFFICIENT ANALYSIS DATABASE PROGRAMS

To use the techniques discussed in this paper and to ensure that all of the data required for tables is stored in the analysis database, ADB programs should be planned out carefully for maximum efficiency. An individual derivation should only occur once so that changes are only required in one place and programs will run as quickly as possible. For instance, all dates/times should be imputed before any time-dependent calculations are made like finding the last record in a treatment period. Another example is that mappings of a value to AVAL dependent variables (e.g. AVALCAT1) should be made before deriving baseline, so that these variables can be stored in similar baseline variables (e.g. BASECAT1) without a recalculation.

An additional consideration for efficiency is that ADaM domains only need to contain the records and variables required to produce and support the statistical results of a study. Hence, if a variable or parameter (e.g. non-protocol lab tests) is not used in any table programs, it does not need to be included in the analysis database. This is particularly important for domains where records are created for derived time points or parameters. These created records will not have source data so it is easier to just drop most source variables from the domain entirely. For traceability, it is sufficient to use variables like SRCSEQ to link analysis data to its source (ADaMIG Section 3.2.8).

By limiting the information in the analysis database to only what is needed for the table outputs, data sets will be smaller allowing for faster run times. It will also be clearer to programmers and external reviewers what records are used for an analysis since the analysis database will not contain extraneous information.

STEP 6. WRITE SIMPLE TABLE PROGRAMS

If applied correctly, the steps above should enable most tables to be generated from relatively simple programs. Since the analysis database has been designed with the table outputs in mind, the table programs should consist of only the SAS procedure required to do the calculation, and then any formatting steps to write out the output. As discussed here, if several analysis domains are designed similarly, a large number of tables can generated from a SAS macro or very similar programs. Lastly, by making the ADaM domains analysis-ready, the results in the tables should be easy to both validate by double programming and recreate by an external reviewer with simple SAS code.

FURTHER EXAMPLES

SHIFT TABLES

Table 7 is a table shell for a typical shift table that is generated for many clinical trials. Shift tables usually consist of a comparison between on-treatment categorical values and a baseline value. Since this requires an analysis using two data points, shift tables are usually considered one of the more difficult types of tables to produce, but using the techniques presented in this paper can make them simpler.

First, it must be determined what SAS procedure will be used to produce the outputs. In this example, the FREQ procedure will be used with the variables containing the on-treatment value and the baseline value both included in the TABLES statement. The code below would produce the results needed for the shift table in one step (note that shift tables may require additional formatting steps after the procedure to transpose the baseline values and populate combinations that do not exist in the data).

```
proc freq data=ADLB;
  tables on-treatment-value * baseline-value;
  by treatment-variable parameter-variable visit-variable;
  where population-flag="Y" and visit-variable in (list-of-visits);
run;
```

Study BCD Table 7. Shift Table of Laboratory Parameters (Population: Safety)										
			Treatment A (N=XX)				Treatment B (N=XX)			
Parameter	**Visit**	**Indicator**	**Low n (%)**	**Normal n (%)**	**High n (%)**	**Missing n (%)**	**Low n (%)**	**Normal n (%)**	**High n (%)**	**Missing n (%)**
Potassium (mmol/L)	Week 4	Low	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)
		Normal	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)
		High	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)
		Missing	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)	X (X.X)
Baseline is last value prior to start of treatment.										

Table 7. Sample table shell for a typical shift table. Only one of many parameters is shown.

For the input data set to be analysis-ready, the on-treatment and baseline variables must be included in the same record. The ADaM BDS standard contains several different pairs of variables that can be used, e.g. AVALC/BASEC, AVALCAT1/BASECAT1, ATOXGR/BTOXGR, and ANRIND/BNRIND (which would be used for Table 8) where the baseline values must match the analysis values on the record where ABLFL=Y.

A shift table will also usually have a corresponding statistical summary that analyzes the exact same subset of records. So that the two tables are consistent, the ADaM domain should be designed so that the same selection criteria can be used for both tables.

A complicating issue with shift tables occurs when the output contains rows for missing values. While it is relatively easy to categorize an on-treatment record with missing baseline (e.g. a record will exist with AVALC populated but BASEC missing), it typically is difficult to analyze cases where a subject had a baseline result but the on-treatment value is missing since a record will not usually exist. In this case, it is worthwhile to create records in the analysis domain with AVALC set to the text "Missing" to ensure that all subjects have records for all of the visits they are to be analyzed at in the table (ADaMIG section 4.8.1).

Adding these records not only maintains the analysis-ready principle of ADaM, but should also initiate a discussion on how these missing visits should be handled earlier in a study's life cycle. Additionally, the rules for creating these records, which can be complex, will then be described in the ADB metadata instead of hidden in a table program.

Study BCD Table 8. On-treatment outliers for Laboratory Parameters (Population: Safety)				
Parameter	**Visit**	**Criteria**	**Treatment A (N=XX) n (%)**	**Treatment B (N=XX) n (%)**
Potassium (mmol/L)	Week 1	<2.7 mmol/L and >20% decrease from baseline	X (X.X)	X (X.X)
		>6.5 mmol/L and >20% increase from baseline	X (X.X)	X (X.X)
	Week 2	<2.7 mmol/L and >20% decrease from baseline	X (X.X)	X (X.X)
		>6.5 mmol/L and >20% increase from baseline	X (X.X)	X (X.X)
	Week 4	<2.7 mmol/L and >20% decrease from baseline	X (X.X)	X (X.X)
		>6.5 mmol/L and >20% increase from baseline	X (X.X)	X (X.X)
Baseline is last value prior to start of treatment. Outliers displayed occurred from after start of treatment until seven days after end of treatment.				

Table 8. Sample table shell for a typical outlier table. Only one of many parameters is shown.

OUTLIER TABLES

Table 8 is a table shell for a typical outlier table. Outlier tables display the frequency at which certain abnormal values (usually lab, ECG, or vitals) of a parameter meet a pre-defined threshold to be considered significant. The table can be generated using the FREQ procedure with a variable indicating the outlier (and possibly another variable describing the outlier) in the TABLES statement as in the code below.

```
proc freq data=ADLB;
  tables outlier-indicator-variable * outlier-description;
  by treatment-variable parameter-variable visit-variable;
  where population-flag="Y" and visit-variable in (list-of-visits);
run;
```

To make the ADaM domain analysis-ready, these outliers should be clearly indicated in the data. The outlier should not however be flagged by a variable not in the BDS standard since it will be unclear what the flag is indicating.

Instead, variables like AVALCATy and CRITy should be used since they can contain descriptions of the criteria used to flag the result as an outlier.

The variable AVALCATy (and similarly CHGCATy, PCHGCATy) can be used if the criteria is completed related to AVAL. The variable CRITy and accompanying flag CRITyFL should be used if the criteria cannot be derived directly from AVAL (e.g. ratio to reference ranges) or is related to multiple BDS variables (e.g. the recorded value AVAL and percent change from baseline PCHG). The outlier result must be flagged by CRITyFL and meaningful text should be given to CRITy so that it is clear what outlier is being flagged.

Additionally, some outlier tables are similar to adverse event tables in that occurrences are counted over the entire treatment period regardless of visit. In this case, it is important to include APERIOD and/or TRTP in the selection criteria for the table and then count unique occurrences per subject.

CONCLUSION

While CDISC standards may seem arbitrary, smartly adopting them increases the quality of a project and reduces the effort needed to complete it. In the case of ADaM, a successful implementation will create an analysis-ready ADaM database that is clearly traceable to source SDTM data. This paper has presented a few methods by which this can be accomplished.

First, pertinent treatment information should be stored in ADSL and dates/times should be imputed whenever possible to allow complex derivations to be consistent and linkable to source data. Second, most other analysis domains should be designed based on the table outputs and should use the BDS standard so that statistical results can be generated with simple table programs. Both techniques aim to increase efficiency by making sure a distinct operation is only performed once in a study's entire programming and increase clarity by using common variable names with derivations that can be clearly explained in metadata.

REFERENCES

CDISC Analysis Data Model (ADaM) Implementation Guide Version 1.0 (2009). Available at cdisc.org.

Barrick, M. and Troxell, J (2014). A guide to ADaM Basic Data Structure for database designers. *Proceedings of PharmaSUG 2014.*

Brucken, N. and P. Slagle (2010). From SAP to ADAM: the nuts and bolts. *Proceedings of PharmaSUG 2010.*

Freimark, N., S. Kenny, J. Shostak, and J. Troxell (2012). Common misunderstandings about ADaM implementation. *Proceedings of PharmaSUG 2012.*

Jain, V. and S. Minjoe (2014). A road map to successful CDISC ADaM submission to FDA: guidelines, best practices and case studies. *Proceedings of PharmaSUG 2014.*

Lee, K. (2011). How to build ADaM BDS from mock up tables. *Proceedings of SAS Global Forum 2011.*

ACKNOWLEDGMENTS

The author would like to thank Rob Diseker for his encouragement and insightful comments during the writing of this paper.

DISCLAIMER

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of PPD.

CONTACT INFORMATION

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

Andrew L Hulme
Andrew.Hulme@ppdi.com

APPENDIX 1 – ADSL AND GENERAL TIMING VARIABLES

Variable	Label	Req?	Comment
ARM	Description of Planned Arm	Yes	From DM.ARM
TRTxxP	Planned Treatment for Period xx	Yes	Only TRT01P is needed for single treatment element studies.
TRTxxA	Actual Treatment for Period xx	**	Required if actual treatment is different than planned treatment.
TRTSDTM	Datetime of First Exposure to Treatment	Yes	Required if treatment is given. TRTSDT or TRTSTM can also be used.
TRTEDTM	Datetime of Last Exposure to Treatment	Yes	
TRxxSDTM	Datetime of First Exposure in Period xx	**	Required if multiple treatments, but good to include even if only one treatment for consistency with TRTP and APERIOD in BDS domains.
TRxxEDTM	Datetime of Last Exposure in Period xx	**	
APxxSDTM	Period xx Start Datetime		Start date/time of the period in which a subject is considered under the treatment TRTxxP.
APxxEDTM	Period xx End Datetime		End date/time of the period in which a subject is considered under the treatment TRTxxP.
SAFFL	Safety Population Flag		At least one population flag is required in ADSL.
*DTM	Datetime of xxxx		Datetime of an observation, see also *DT and *TM
*DTF	Date imputation flag of xxxx		Required if any dates are imputed
*TMF	Time imputation flag of xxxx		Required if any times are imputed

APPENDIX 2 – BDS VARIABLES

Variable	Label	Req?	Comment
ADTM	Analysis Datetime		The date/time associated with AVAL/AVALC in integer format, SAS formats should be applied for readability. See also ADT and ATM.
ADY	Analysis Relative Day		Days to a reference date of choice, does not need to match SDTM xxDY variables.
AVISIT	Analysis Visit		Should display the text used in table outputs and does not need to match CSR visit. Numeric code variable is AVISITN.
ATPT	Analysis Timepoint		
APERIOD	Period		The integer describing the analysis treatment period to which a record is assigned based on APxxSDTM/APxxEDTM from ADSL. APERIODC is used to describe the period.
TRTP	Planned Treatment	Yes	The planned treatment attributed to a record. Must match value of TRTxxP for the period number defined by APERIOD.
PARAM	Parameter	Yes	Describes the parameter in AVAL. Must include all qualifying information like units or location. Short name stored in PARAMCD (also required). Parameters can be ordered using PARAMN.
PARCATy	Parameter Category y		Used to group PARAMs. Must be same for each unique PARAM.
PARAMTYP	Parameter Type		Used for parameters not in source data and created for the analysis from other parameters (PARAMs). Should be set to DERIVED.
DTYPE	Derivation Type		Used for records not in source data and created for the analysis from other records in the same PARAM. Contains description of method (e.g. AVERAGE).
AVAL	Analysis Value	Yes, one of these	Numeric value analyzed in tables, described by PARAM.
AVALC	Analysis Value (C)		Character value analyzed in tables, described by PARAM. If both AVAL and AVALC are used there must be a 1-to-1 map between them.
AVALCATy	Analysis Category y		Used to group AVAL/AVALC. Must be same for each unique value of AVAL/AVALC inside a parameter.
ATOXGR	Analysis Toxicity Grade		Grade used for analysis, can be different than xxTOXGR from SDTM.
ANRIND	Analysis Reference Range Indicator		Indicator used for analysis, can be different than xxNRIND from SDTM.
ABLFL	Baseline Record Flag		Used to flag baseline result, must be included if BASE is used.

Variable	Label	Req?	Comment
BASE	Baseline Value		Baseline value, must equal AVAL from record where ABLFL=Y for that subject and parameter.
BASECATy	Baseline Category y		Must be AVALCATy from record where ABLFL=Y. Similar baseline variables exist for ATOXGR and ANRIND
CHG	Change from Baseline		Must be AVAL-BASE, can be categorized by CHGCATy.
PCHG	Percent Change from Baseline		Must be CHG/BASE*100, can be categorized by PCHGCATy
CRITy	Analysis Criterion y		Contains text describing a pre-specified condition that may be met by the record.
CRITyFL	Criterion y Evaluation Result Flag		Used to indicate if the conditions described in CRITy are met, required if CRITy is used.
SRCSEQ	Source Sequence Number		Can be used to link analysis domain record back to source data. See also SRCDOM and SRCVAR.