PhUSE 2017

Paper SI06

ADaM mapping - key considerations for a metadata driven realization

Elena Glathe, Bayer AG, Berlin, Germany

ABSTRACT

One key task in statistical programming is the mapping of CDISC compliant analysis data sets following ADaM guidelines. Analysis data sets need to be clearly derived and traceable from SDTM data sets and need to consider the study specific analysis specification per SAP. This wish puts up some challenges for a (semi)-automated solution of this process. At Bayer we took on this challenge by implementing “AdaMap”: Key elements of AdaMap are the metadata management handled in SAS data sets, the mapping from SDTM+ to ADaM and extensive QC checks.

Some questions addressed in this paper:

- How to derive and document new records
- How to derive variables in several iteration steps where variables derived in early iterations are not allowed to be changed in later iterations
- How to allow the mapping of several variables in only one step
- Automatic decode handling

INTRODUCTION

At Bayer an internal developed mapping solution is in place that supports the mapping process to CDISC compliant analysis data sets following ADaM guidelines out of SDTM+. The tool is called AdaMap and is used globally for more than 5 years now for phase 1 to phase 3 trials. It is a SAS based macro solution with metadata maintained in SAS data sets.

The AdaMap SAS macro framework provides following data flow support to the user:

Mapping:

- ⇨ Semi-automated mapping from SDTM+ to ADaM data, e.g.:
  - ▪ Collect required variables from input data (SDTM+)
  - ▪ Automatic derivation of variable if algorithm is fixed and mapping method defined in the metadata
  - ▪ Automatically adding decode variables and merge ADSL core variables
  - ▪ Automatically applying formats, labels, types as defined in metadata

Codelists and Formats:

- ⇨ Set up and maintenance of ADaM codelists
- ⇨ Set up of study analysis format catalog needed for statistical analysis

ADaM Metadata:

- ⇨ Set up and maintenance of standard ADaM metadata. This includes mapping algorithms that are automatically processed by AdaMap
- ⇨ Set up of study ADaM metadata out of the standard ADaM metadata

ADS Specification:

- ⇨ Create ADaM mapping specification derived from the final study ADaM metadata

QC Checks:

- ⇨ Check macros to help ensure quality and adherence to Bayer standard rules as well as CDISC ADaM rules.

Prerequisite to automated mapping:

To enable automated data flow the adherence to data standards is needed. Starting point for ADaM mapping at Bayer is SDTM+. SDTM+ is following SDTM guidelines and controlled terminology but numerical variables and coded variables are still present in parallel to the character decode variables and supplemental variables are still merges to the source ADaM data set.

In the following sections concepts and design decisions are highlights during the development of AdaMap. Key requirement was to streamline the mapping process as much as possible, to support the statistical programmer in the ADaM derivation but still keep them in control and to allow for the necessary flexibility that is required to derive study statistical analysis data sets according to ADaM.

ADAM METADATA

ADaM developed by CDISC, provides guidelines for creating data sets and their accompanying documentation to support statistical analyses. The basic concepts are:

- Data is analysis ready, meaning that reviewers can replicate and explore these results with little or no data manipulation, allowing reviewers to concentrate on the results, not on programming. (**Fehler! Verweisquelle konnte nicht gefunden werden.**, CDISC Analysis Data Model: Version 2.1)
- All derived variables, derived observations, and imputations should be permanently stored within an analysis data set
- Minimize programming for subsequent users, i.e., Health Authority Reviewers.

ADaM Metadata is data about the structure and attributes of the ADaM datasets. These are produced in line with ADaM guidelines.

Within Bayer it was decided to develop standard (global) ADaM metadata that can be used as basis for all clinical studies. With the decision to develop AdaMap it was further decided to add standard mapping specification and mapping algorithm as part of those standard ADaM metadata.

Standardized ADaM metadata with embedded mapping algorithms built the foundation for automatization. Decisions about the structure and content of the metadata play a key role. Therefore the next two sections provide an insight into the chosen metadata concept for ADaM and AdaMap.

DATA SET LEVEL METADATA

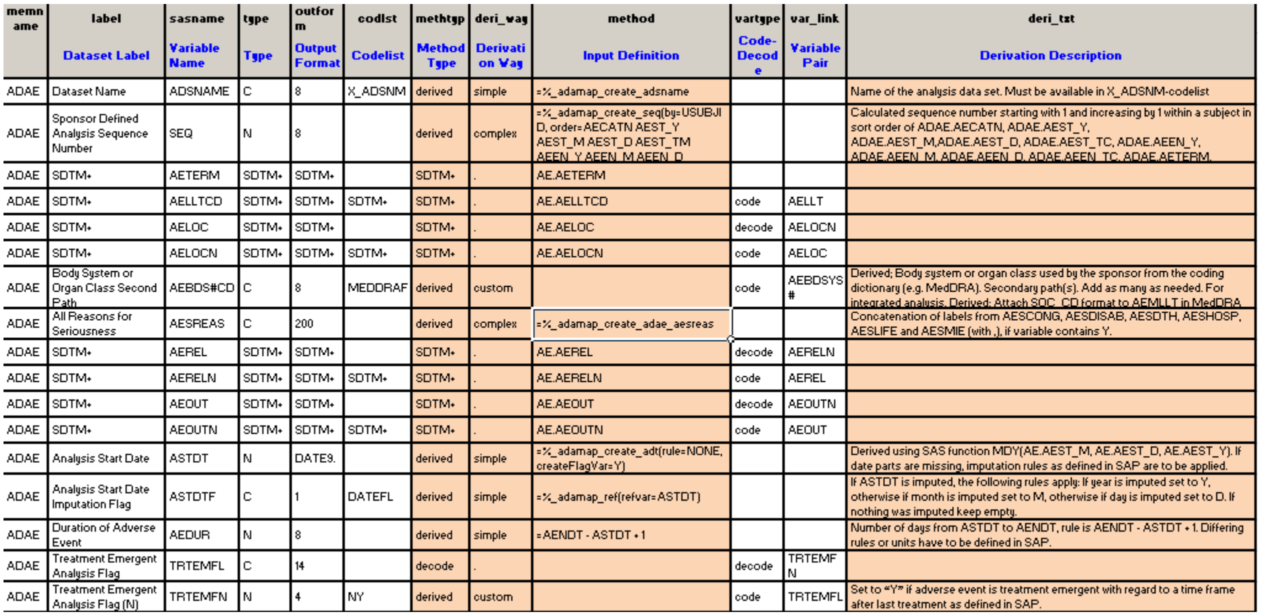
MEM NAME	LABEL	KEYS	METHOD	CLASS	STRUCTUR
ADAE	Adverse Events Analysis Dataset	STUDYID,USUBJID, AESEQ,SEQ	AE(where= (AENYN=1))	ADAE	one record per subject per each AE recorded in SDTM AE domain
ADCM	Concomitant Medications Analysis Dataset	STUDYID,USUBJID, CMSEQ,SEQ	CM(where= (CMNYN=1))	OTHER	One record per recorded medication occurrence or constant-dosing interval per subject recorded in SDTM CM domain
ADLB	Laboratory Analysis Dataset	STUDYID,USUBJID, SEQ	LB	BDS	One or more records per subject per analysis parameter
ADDA	Drug Accountability Analysis Dataset	STUDYID,USUBJID, DASEQ,SEQ	DA	BDS	one record per subject per parameter

Figure 1: Example for data set level metadata

Data set level metadata is specified in one data set where each row defines one analysis domain. An example for four ADaM data sets with the most important columns is given in Figure 1.
Typical required attributes are MEMNAME, LABEL, KEYS, STRUCTUR and CLASS for classification per CDISC definition.
Important for automatization is column METHOD where the input SDTM+ data set is specified for the ADaM generation. A SAS WHERE condition can be defined to subset the SDTM+ observations, that are taken over to ADaM, if necessary.

VARIABLE LEVEL METADATA

Variable level metadata for ADaM are stored in data sets with the same name as the related analysis data set. It contains typical information like LABEL, LENGTH, SASNAME, OUTFORM, CODELIST and TYPE.



memn ame	label	sasname	type	outfor m	codlst	methtyp	deri_way	method	vartype	var_link	deri_txt
	Dataset Label	Variable Name	Type	Output Format	Codelist	Method Type	Derivati on Way	Input Definition	Code-Decod e	Variable Pair	Derivation Description
ADAE	Dataset Name	ADSNAME	C	8	X_ADSNM	derived	simple	=%_adamap_create_adsname			Name of the analysis data set. Must be available in X_ADSNM-codelist
ADAE	Sponsor Defined Analysis Sequence Number	SEQ	N	8		derived	complex	=%_adamap_create_seq(by=USUBJID, order=AECATN AEST_Y AEST_M AEST_D AEST_TM AEEN_Y AEEN_M AEEN_D			Calculated sequence number starting with 1 and increasing by 1 within a subject in sort order of ADAE.AECATN, ADAE.AEST_Y, ADAE.AEST_M,ADAE.AEST_D, ADAE.AEST_TC, ADAE.AEEN_Y, ADAE.AEEN_M, ADAE.AEEN_D, ADAE.AEEN_TC, ADAE.AETERM,
ADAE	SDTM+	AETERM	SDTM+	SDTM+		SDTM+	.	AE.AETERM			
ADAE	SDTM+	AELLTCD	SDTM+	SDTM+	SDTM+	SDTM+	.	AE.AELLTCD	code	AELLT	
ADAE	SDTM+	AELOC	SDTM+	SDTM+		SDTM+	.	AE.AELOC	decode	AELOCN	
ADAE	SDTM+	AELOCN	SDTM+	SDTM+	SDTM+	SDTM+	.	AE.AELOCN	code	AELOC	
ADAE	Body System or Organ Class Second Path	AEBDS#CD	C	8	MEDDRAF	derived	custom		code	AEBDSYS#	Derived; Body system or organ class used by the sponsor from the coding dictionary (e.g. MedDRA). Secondary path(s). Add as many as needed. For integrated analysis. Derived: Attach SOC_CD format to AEMLLT in MedDRA
ADAE	All Reasons for Seriousness	AESREAS	C	200		derived	complex	=%_adamap_create_adae_aesreas			Concatenation of labels from AESCONG, AESDISAB, AESDTH, AESHOSP, AESLIFE and AESMIE (with ,), if variable contains Y.
ADAE	SDTM+	AEREL	SDTM+	SDTM+		SDTM+	.	AE.AEREL	decode	AERELN	
ADAE	SDTM+	AERELN	SDTM+	SDTM+	SDTM+	SDTM+	.	AE.AERELN	code	AEREL	
ADAE	SDTM+	AEOUT	SDTM+	SDTM+		SDTM+	.	AE.AEOUT	decode	AEOUTN	
ADAE	SDTM+	AEOUTN	SDTM+	SDTM+	SDTM+	SDTM+	.	AE.AEOUTN	code	AEOUT	
ADAE	Analysis Start Date	ASTDT	N	DATE9.		derived	simple	=%_adamap_create_adt(rule=NONE, createFlagVar=Y)			Derived using SAS function MDY(AE.AEST_M, AE.AEST_D, AE.AEST_Y). If date parts are missing, imputation rules as defined in SAP are to be applied.
ADAE	Analysis Start Date Imputation Flag	ASTDTF	C	1	DATEFL	derived	simple	=%_adamap_ref(refvar=ASTDT)			If ASTDT is imputed, the following rules apply: If year is imputed set to Y, otherwise if month is imputed set to M, otherwise if day is imputed set to D. If nothing was imputed keep empty.
ADAE	Duration of Adverse Event	AEDUR	N	8		derived	simple	=AENDT - ASTDT + 1			Number of days from ASTDT to AENDT, rule is AENDT - ASTDT + 1. Differing rules or units have to be defined in SAP.
ADAE	Treatment Emergent Analysis Flag	TRTEMFL	C	14		decode	.		decode	TRTEMFN	
ADAE	Treatment Emergent Analysis Flag (N)	TRTEMFN	N	4	NY	derived	custom		code	TRTEMFL	Set to "Y" if adverse event is treatment emergent with regard to a time frame after last treatment as defined in SAP.

Figure 2: Example ADaM Metadata set for AdaMap

The four columns METHOD, METHTYP, DERI_WAY and DERI_TXT in Figure 2 are highlighted. They are used by the AdaMap system to allow automated processing based upon mapping algorithms, defined directly with the ADaM metadata:

METHTYP	Method Type	SDTM+, derived, Decode	3 different mapping method types are possible: SDTM+ means that the variable and its content is taken over from SDTM Decode means that the variable is the character representation of a coded or numerical variable. The creation of decode variable is fully automated in AdaMap including QC (see section QC AND VALIDATION)
DERI_WAY	Derivation way	simple complex empty (custom)	**Simple:** Method can be a simple assignment or mathematical operation, also using inline SAS functions (all used within a data step) **Complex**: for complex derivations that require sorting or merging ➔ here a macro call is specified with parameters **Custom:** no information on derivation for the variable
METHOD	Origin/ Computational Method		Source code fragments that are embedded in data steps and in the AdaMap mapping framework
DERI_TXT	Derivation description		For all derived Variables (METHTYP=derived), it has to contain the derivation rule. For all decode and SDTM+ variables it has to be empty.

HASH (#) VARIABLES

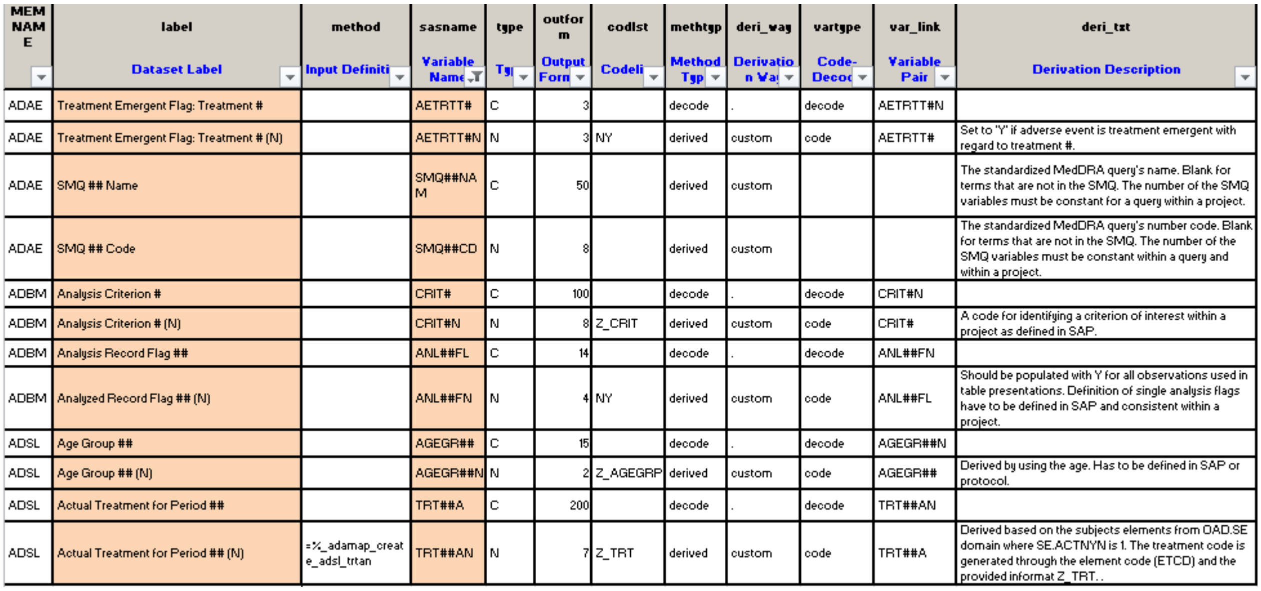
ADaM defines analysis variables where a number in the variable name indicates a collection of several variables of the same type.

Typical examples are shown in Figure 3. TRT##A fo example, is used to define treatment variables per period, where the ## is replaced with the period number. Similarly, this applies to Analysis Record Flags or Analysis Criterion Flags, where several can be defined. The exact number depends on the study needs and the SAP.

In the design of the standard ADaM metadata it was decided to use # as wildcard for easier maintenance of standard metadata.

On study set-up based on standard metadata, these #-es can either be resolved to one or more records with real numbers, or can stay as is. Examples where resolving makes sence, are analysis record flags. Here the derivation method chanes from flag to flag. An example where resolving does not make sence, is the assignment of ATC variables (CMATC##), where the final number of required variables are unknown until the final coding is done.

However it is important that in the end DERI_TXT gets updated for a study to reflect correct derivation description.



MEMNAME	label	method	sasname	type	outform	codlst	methtyp	deri_way	vartype	var_link	deri_txt
	Dataset Label	Input Definiti	Variable Name	Ty	Output Forn	Codeli	Method Typ	Derivation Va	Code-Decod	Variable Pair	Derivation Description
ADAE	Treatment Emergent Flag: Treatment #		AETRTT#	C	3		decode	.	decode	AETRTT#N	
ADAE	Treatment Emergent Flag: Treatment # (N)		AETRTT#N	N	3	NY	derived	custom	code	AETRTT#	Set to "Y" if adverse event is treatment emergent with regard to treatment #.
ADAE	SMQ ## Name		SMQ##NAM	C	50		derived	custom			The standardized MedDRA query's name. Blank for terms that are not in the SMQ. The number of the SMQ variables must be constant for a query within a project.
ADAE	SMQ ## Code		SMQ##CD	N	8		derived	custom			The standardized MedDRA query's number code. Blank for terms that are not in the SMQ. The number of the SMQ variables must be constant within a query and within a project.
ADBM	Analysis Criterion #		CRIT#	C	100		decode	.	decode	CRIT#N	
ADBM	Analysis Criterion # (N)		CRIT#N	N	8	Z_CRIT	derived	custom	code	CRIT#	A code for identifying a criterion of interest within a project as defined in SAP.
ADBM	Analysis Record Flag ##		ANL##FL	C	14		decode	.	decode	ANL##FN	
ADBM	Analyzed Record Flag ## (N)		ANL##FN	N	4	NY	derived	custom	code	ANL##FL	Should be populated with Y for all observations used in table presentations. Definition of single analysis flags have to be defined in SAP and consistent within a project.
ADSL	Age Group ##		AGEGR##	C	15		decode	.	decode	AGEGR##N	
ADSL	Age Group ## (N)		AGEGR##N	N	2	Z_AGEGRP	derived	custom	code	AGEGR##	Derived by using the age. Has to be defined in SAP or protocol.
ADSL	Actual Treatment for Period ##		TRT##A	C	200		decode	.	decode	TRT##AN	
ADSL	Actual Treatment for Period ## (N)	=%_adamap_create_adsl_trtan	TRT##AN	N	7	Z_TRT	derived	custom	code	TRT##A	Derived based on the subjects elements from OAD.SE domain where SE.ACTNYN is 1. The treatment code is generated through the element code (ETCD) and the provided informat Z_TRT. .

Figure 3: Hash variables in Standards ADaM metadata as indicator for a collection of numbered variables

REUSE ATTRIBUTES

There are quite many variables in ADaM that are taken over from SDTM+.

In those cases it is very helpful to reuse the attributes of SDTM+ variables instead of double maintenance in both SDTM+ and ADaM, which is easily error prone. Using the wildcard SDTM+ within the ADaM metadata indicates that label, codelist, length and type are inherit from SDTM+ metadata.

In the example below you can also see how the link to the source SDTM+ variable in METHOD works: simply reference the SDTM+ dataset and variable, e.g. AE.STUDYID.

Clear traceability is implemented between SDTM+ and ADaM this way, as required.

memname	sasname	Label	Method	type	outform	codlst	methtyp	deri_way	vartype	var_link
Data set Name	Variable Name	Dataset Label	Computational method	Type	Output Format	Codelist	Method Type	Derivation Way	Code-Decode	Variable Pair
ADAE	STUDYID	SDTM+	AE.STUDYID	SDTM+	SDTM+	SDTM+	SDTM+	.		
ADAE	USUBJID	SDTM+	AE.USUBJID	SDTM+	SDTM+	SDTM+	SDTM+	.		
ADAE	AEDECOD	SDTM+	AE.AEDECOD	SDTM+	SDTM+	SDTM+	SDTM+	.	decode	AEPTCD
ADAE	AELLT	SDTM+	AE.AELLT	SDTM+	SDTM+	SDTM+	SDTM+	.	decode	AELLTCD
ADAE	AELLTCD	SDTM+	AE.AELLTCD	SDTM+	SDTM+	SDTM+	SDTM+	.	code	AELLT
ADAE	AEPTCD	SDTM+	AE.AEPTCD	SDTM+	SDTM+	SDTM+	SDTM+	.	code	AEDECOD
ADAE	AEHLT	SDTM+	AE.AEHLT	SDTM+	SDTM+	SDTM+	SDTM+	.	decode	AEHLTCD
ADAE	AEHLTCD	SDTM+	AE.AEHLTCD	SDTM+	SDTM+	SDTM+	SDTM+	.	code	AEHLT
ADAE	AEHLGT	SDTM+	AE.AEHLGT	SDTM+	SDTM+	SDTM+	SDTM+	.	decode	AEHLGTCD

Figure 4: Example: SDTM+ is referenced for variables directly received from SDTM+ datasets. During study ADaM mapping the actual SDTM+ variable is applied

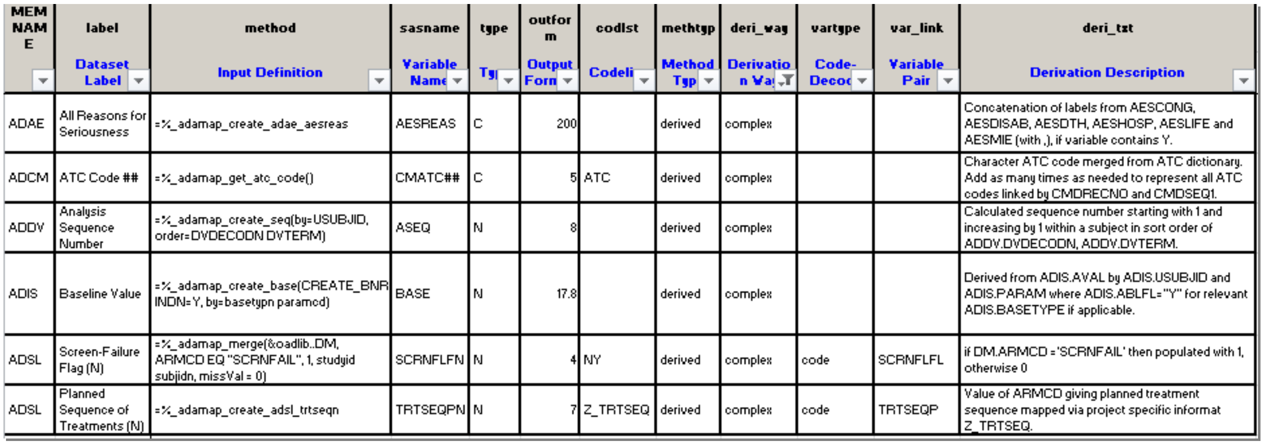
DERIVATIONS

Derivations can be classified into simple, complex and custom.

Simple derivations: Those are for example assignments of values, concatenations, a simple merge or a calculation using various variables within the same record (e.g. BMI calculation). All such derivations can be handled directly within the metadata and the derivation is automatically handled by AdaMap.

All simple derivations, that are equal between studies, should already be specified in the standard ADaM metadata. Then all studies and all statistical programmers can benefit from this standardization.

Complex derivations: For more complex, but still standard derivations, Standard AdaMap macros are implemented. The macro call can then be added as well to the METHOD column in the standard ADaM metadata. Some examples are shown in Figure 5.



MEMNAME	label	method	sasname	type	outform	codlst	methtyp	deri_way	vartype	var_link	deri_txt
	Dataset Label	Input Definition	Variable Name	Ty	Output Forn	Codeli	Method Typ	Derivation Wa	Code-Decoc	Variable Pair	Derivation Description
ADAE	All Reasons for Seriousness	=%_adamap_create_adae_aesreas	AESREAS	C	200		derived	complex			Concatenation of labels from AESCONG, AESDISAB, AESDTH, AESHOSP, AESLIFE and AESMIE (with ,), if variable contains Y.
ADCM	ATC Code ##	=%_adamap_get_atc_code()	CMATC##	C	5	ATC	derived	complex			Character ATC code merged from ATC dictionary. Add as many times as needed to represent all ATC codes linked by CMDRECNO and CMDSEQ1.
ADDV	Analysis Sequence Number	=%_adamap_create_seq(by=USUBJID, order=DVDECODN DVTERM)	ASEQ	N	8		derived	complex			Calculated sequence number starting with 1 and increasing by 1 within a subject in sort order of ADDV.DVDECODN, ADDV.DVTERM.
ADIS	Baseline Value	=%_adamap_create_base(CREATE_BNR INDN=Y, by=basetypn paramcd)	BASE	N	17.8		derived	complex			Derived from ADIS.AVAL by ADIS.USUBJID and ADIS.PARAM where ADIS.ABLFL="Y" for relevant ADIS.BASETYPE if applicable.
ADSL	Screen-Failure Flag (N)	=%_adamap_merge(&oadlib..DM, ARMCD EQ "SCRNFAIL", 1, studyid subjidn, missVal = 0)	SCRNFLFN	N	4	NY	derived	complex	code	SCRNFLFL	if DM.ARMCD = 'SCRNFAIL' then populated with 1, otherwise 0
ADSL	Planned Sequence of Treatments (N)	=%_adamap_create_adsl_trtseqn	TRTSEQPN	N	7	Z_TRTSEQ	derived	complex	code	TRTSEQP	Value of ARMCD giving planned treatment sequence mapped via project specific informat Z_TRTSEQ.

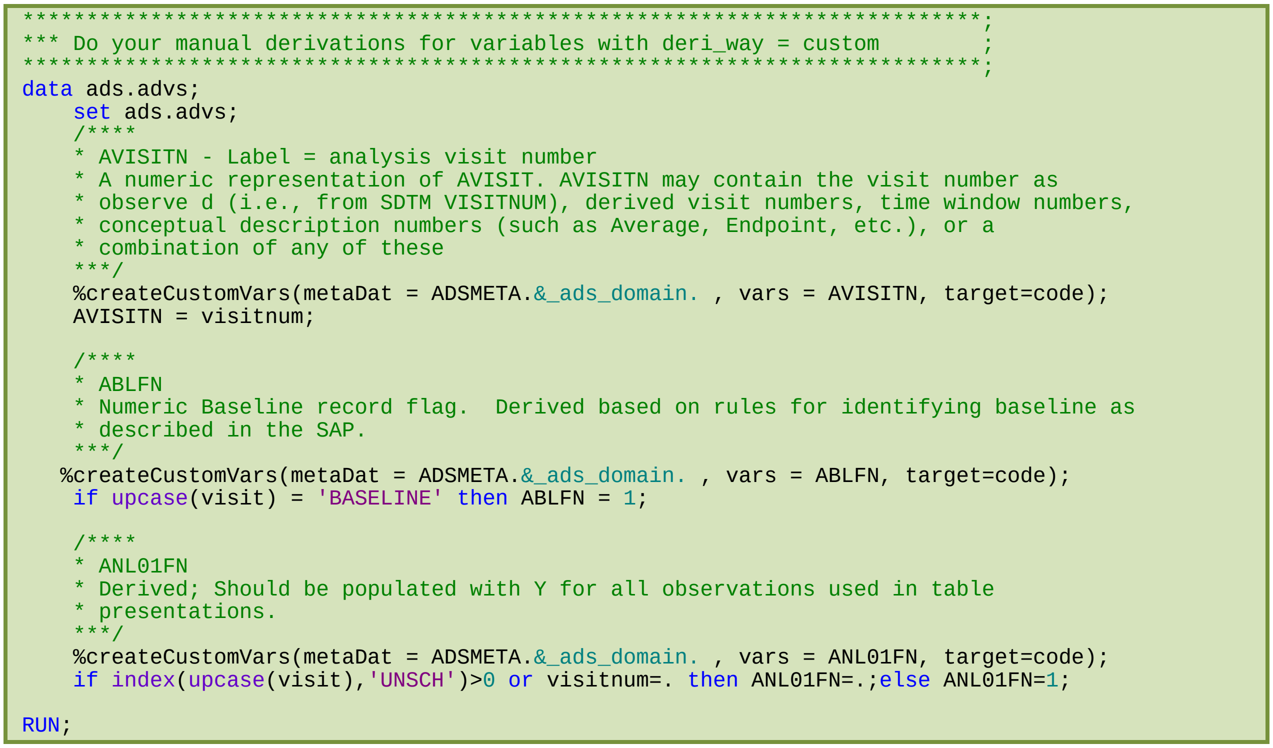
Figure 5: Some complex mapping examples

Custom derivations: The focus of the study statistical programmer is on custom derivations. Those are study specific derivations which can become rather complex. For example the creation of efficacy ADaM data sets.

CUSTOM MAPPINGS

For some derived variables it is not possible to define a standard derivation. Mostly the reason is that the derivation differs from study to study. In this case no automated derivation can be performed and no rule is added to the metadata.

In this case the derivation is to be done in the specific derivation program. This is the case when the column METHOD is empty and DERI_WAY is set to “custom”. Figure 6 shows how custom variables can be derived for the analysis data set ADVS in three simple examples.



```
*************************************************************************;
*** Do your manual derivations for variables with deri_way = custom      ;
*************************************************************************;
data ads.advs;
    set ads.advs;
    /****
    * AVISITN - Label = analysis visit number
    * A numeric representation of AVISIT. AVISITN may contain the visit number as
    * observe d (i.e., from SDTM VISITNUM), derived visit numbers, time window numbers,
    * conceptual description numbers (such as Average, Endpoint, etc.), or a
    * combination of any of these
    ***/
    %createCustomVars(metaDat = ADSMETA.&_ads_domain. , vars = AVISITN, target=code);
    AVISITN = visitnum;

    /****
    * ABLFN
    * Numeric Baseline record flag.  Derived based on rules for identifying baseline as
    * described in the SAP.
    ***/
   %createCustomVars(metaDat = ADSMETA.&_ads_domain. , vars = ABLFN, target=code);
    if upcase(visit) = 'BASELINE' then ABLFN = 1;

    /****
    * ANL01FN
    * Derived; Should be populated with Y for all observations used in table
    * presentations.
    ***/
    %createCustomVars(metaDat = ADSMETA.&_ads_domain. , vars = ANL01FN, target=code);
    if index(upcase(visit),'UNSCH')>0 or visitnum=. then ANL01FN=.;else ANL01FN=1;

RUN;
```

Figure 6: Example program for custom mapping of ADVS

For custom derivations a few helpful things are available in AdaMap.

Particularly macro %createCustomVars is tiny but quite handy. It offers 2 options to provide the attributes of derived variables:

1. Called within a data step %createCustomVars provides an easy initialization of the custom derived variable with all attributes (length, type, format and label) based on information from study ADaM metadata. It is recommended to be used when creating custom variables. Length, type, format and label don’t need to be hardcoded in the source code and potential late changes in the metadata don’t cause inconsistency or a need to update the program. See source code above.

2. Same macro call run with parameter target=LOG prints the attributes in the log as nice SAS comment. The statistical programmer can copy and paste the SAS comment from the log and place it above into the SAS program for documentation purpose.

```
%createCustomVars(metadat = adsmeta.ADLB, target  = LOG)

Log output:
 *< AVISITN;
/** LABEL = Analysis Visit (N) - Format:_AVISIT - Outform: 8.3
 ** A numeric representation of AVISIT. AVISITN may contain the visit number as
 ** observed (i.e., from SDTM VISITNUM), derived visit numbers, time window numbers,
 ** conceptual description numbers (such as Average, Endpoint, etc.) or a combination
 ** of any of these **/
```

SEQUENCE OF DERIVATIONS

An interesting question in a complex mapping system like AdaMap is the sequence of derivation. The derivation of one variable might depend on other derivations that have to be performed first. So the question was how to make the system as robust as possible but of course ensure correct derivation sequence takes place.

For AdaMap it was decided to go through the pre-specified mappings from the ADaM metadata (METHTYP=derived and METHOD not empty) in several iteration loops. In each loop the newly added variables are counted. If no new variable is added in one loop, the system can stop.

A single derivation can be performed when all input variables are available. This has to be checked by AdaMap. If not all input variables are available the derivation will be postponed (which AdaMap documents in the log for the statistical programmer)

```
NOTE: CREATE_INITIAL_ADS_DAT - Will create ADS.ADSL from SDTMP.DM!
INFO: CREATE_INITIAL_ADS_DAT - Add SDTMP variables from DM to ADS.ADSL
INFO: CREATE_INITIAL_ADS_DAT - Will rename DM --> PPROT to PPROTFL
INFO: CREATE_INITIAL_ADS_DAT - Will rename DM --> PPROTN to PPROTFN
NOTE: CREATE_INITIAL_ADS_DAT - Will create ADS.ADAE from SDTMP.AE (where = (AENYN=1))!
NOTE: CREATE_INITIAL_ADS_DAT - Will create ADS.ADEGF from SDTMP.EG (where = (EGCAT="FINDING"))!
INFO: CREATE_INITIAL_ADS_DAT - Add SDTMP variables from EG to ADS.ADEGF
INFO: PERFORM_DERIVATIONS_DAT - Will Derive UASR method =
             COMPRESS(CATX('/', USUBJID,AGE, SEX, PUT(RACEALLN, X_RACES.)))
INFO: PERFORM_DERIVATIONS_DAT - Will Derive BRTHDT method = %_ADAMAP_CREATE_DT
INFO: PERFORM_DERIVATIONS_DAT - Will Derive BRTHDTL method = %_ADAMAP_CREATE_DTL
INFO: PERFORM_DERIVATIONS_DAT - Will Derive SMOKHXN method = %_ADAMAP_CREATE_ADSL_VAR
INFO: ADDADSMETADATA2DAT - ATTRIB STUDYID LENGTH = $10 FORMAT = $10. LABEL = "Study Identifier"
INFO: ADDADSMETADATA2DAT - ATTRIB USUBJID LENGTH = $20 FORMAT = $20. LABEL = "Unique Subject
Identifier"
INFO: ADDADSMETADATA2DAT - ATTRIB SUBJIDN FORMAT = 9. LABEL = "Subject Identifier for Study(N)"
NOTE: PERFORM_DERIVATIONS_DAT - Postponed derivation of 1 variable(s) in ADS.ADDA: TRTPN
INFO: PERFORM_DERIVATIONS_DAT - Will Derive PARAMTYP method = ""
INFO: PERFORM_DERIVATIONS_DAT - Will Derive DTYPE method = ""
INFO: ADDDECODES2DAT - Code variable APERIOD is not available. Will not add decode APERIODC!
INFO: ADDDECODES2DAT - Add variable ANRLO with decode of ANRLON
INFO: ADDDECODES2DAT - Add variable ANRHI with decode of ANRHIN
INFO: MERGE_ADSL - Will merge ADSL core variables from ADS.ADSL to ADS.ADAE!
```

Figure 7: Example how AdaMap documents the automatic derivation in the log

DERIVATION OF SEVERAL ANALYSIS VARIABLES IN ONE STEP

In a variable based mapping table, as it is used in our ADaM metadata, each row specifies the mapping of one individual variable only. However sometimes a single algorithm results in multiple derived variables. As it would not make sence to execute such algorithms multiple times, a way had to be implemented to derive multiple variables at the same time and document this appropriately. An example is an imputed analysis date (ADT) where the imputation information is stored in the corresponding analysis date imputation flag variable (ADTF). Both variables should be derived in one undividable step.

To handle these cases, an empty macro called %adamap_ref was developed that does nothing else than to document the reference to the ADaM variable where the variable is derived together. In the example below, the derivation method =%_adamap_ref(refvar=ASTDT) of variable ASTDTF means that the variable is derived together with variable ASTDT.

memname Dataset Name	sasname Variable Name	mabel Dataset Label	Method Input Definition
ADAE	ASTDT	Analysis Start Date	=%_adamap_create_adt(rule=NONE, createFlagVar=Y)
ADAE	ASTDTF	Analysis Start Date Imputation Flag	=%_adamap_ref(refvar=ASTDT)

Figure 8: Example for metadata where 2 variables are derived together

This concept ensures clear documentation in the metadata and avoids rerun of the same code.
It is also implemented to work if more than 2 variables are produced at once.

CHARACTER VARIABLES (DECODES)

The programmer does not need to take care about the creation of decode variable that belong to numerical or coded variables (variables with format/controlled terminology attached).

The link between code and decode variables is handled over the metadata. Metadata variable VARTYPE defines whether it is a code or decode variable (see green column in Figure 10) and metadata variable VAR_LINK contains the name of the linked variable. As shown in Figure 10 even # wildcard variables are supported with the VAR_LINK concept (see row 5). The general naming convention rule per CDISC ADaM between code and decode variables is applied.

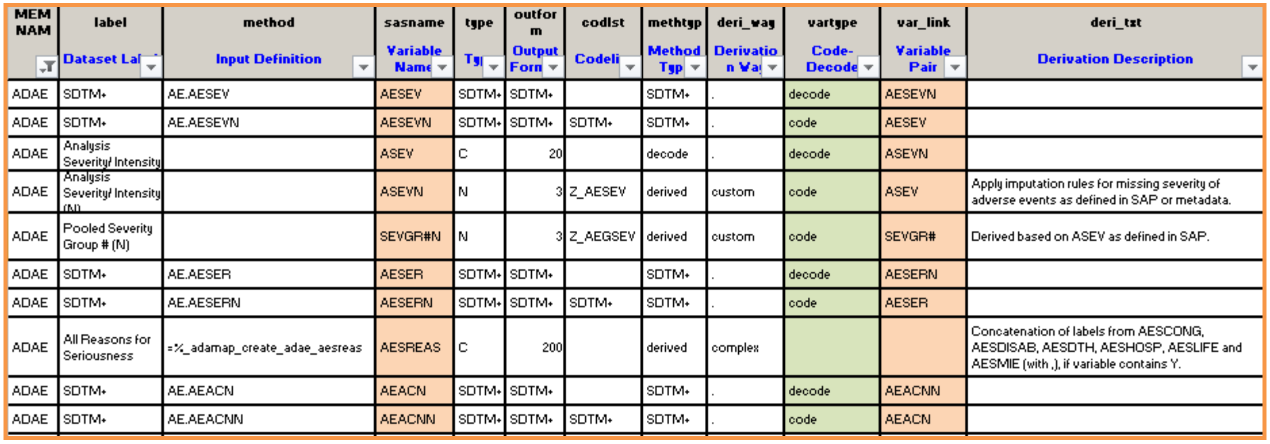
Code	Decode	Explanation
<var>N	<var>	Numeric
<var>	<var>C	Character
<var>CD	<var>	Codes
<var>FN	<var>FL	Flags

Figure 9: Basic naming convention between code and decode variables (exceptions exist due to 8 character limitation)

AdaMap Macro %addDecodes adds decode variables for all existing variables where METHTYP="decode". This is true for all variables, regardless whether custom derivation or automatic derivation.

Having those two columns VARTYPE and VAR_LINK available makes it also easy to check that code and decode content matches. Also it is easy to remove all decode variables and to recreate them when necessary.

Removing decode variables can be a resource saver.



MEM NAM	label	method	sasname	type	outfor m	codlst	methtyp	deri_way	vartype	var_link	deri_txt
	Dataset Lal	Input Definition	Variable Name	Ty	Output Form	Codeli	Method Typ	Derivation Wa	Code-Decode	Variable Pair	Derivation Description
ADAE	SDTM+	AE.AESEV	AESEV	SDTM+	SDTM+		SDTM+	.	decode	AESEVN	
ADAE	SDTM+	AE.AESEVN	AESEVN	SDTM+	SDTM+	SDTM+	SDTM+	.	code	AESEV	
ADAE	Analysis Severity/ Intensity		ASEV	C	20		decode	.	decode	ASEVN	
ADAE	Analysis Severity/ Intensity (N)		ASEVN	N	3	Z_AESEV	derived	custom	code	ASEV	Apply imputation rules for missing severity of adverse events as defined in SAP or metadata.
ADAE	Pooled Severity Group # (N)		SEVGR#N	N	3	Z_AEGSEV	derived	custom	code	SEVGR#	Derived based on ASEV as defined in SAP.
ADAE	SDTM+	AE.AESER	AESER	SDTM+	SDTM+		SDTM+	.	decode	AESERN	
ADAE	SDTM+	AE.AESERN	AESERN	SDTM+	SDTM+	SDTM+	SDTM+	.	code	AESER	
ADAE	All Reasons for Seriousness	=%_adamap_create_adae_aesreas	AESREAS	C	200		derived	complex			Concatenation of labels from AESCONG, AESDISAB, AESDTH, AESHOSP, AESLIFE and AESMIE (with ,), if variable contains Y.
ADAE	SDTM+	AE.AEACN	AEACN	SDTM+	SDTM+		SDTM+	.	decode	AEACNN	
ADAE	SDTM+	AE.AEACNN	AEACNN	SDTM+	SDTM+	SDTM+	SDTM+	.	code	AEACN	

Figure 10: Decode handling in Metadata over columns VARTYPE and VAR_LINK

ITERATIVE APPROACH TO DERIVE NEW ADSL VARIABLES

To create ADSL, it is frequently required to have variables already available that are derived in other domains. For instance baseline vital sign parameters are almost always required in ADSL. Often, also laboratory parameters or information from questionnaires are required.

It can be an iterative process to derive more derivations that are required as ADSL variables to be merged to other domains that serve as basis for further derivations for potential further ADSL core variables (variables that will be merged to all ADaM data sets, e.g. for subgroup analysis).

In the end the question is what comes first: ADSL that depends on some analysis data sets or the analysis data set that requires variables from ADSL for derivation.

A CHICKEN-EGG Problem.

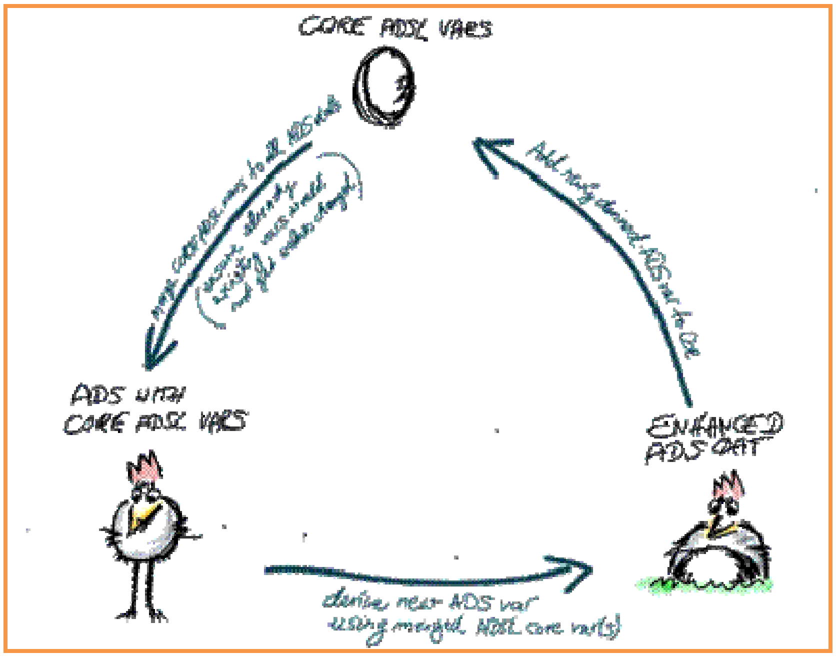

CORE ADSL VARS
merge core ADSL vars to all ADS data
(ensure already existing vars still valid after change)
Add newly derived ADS var to ADSL
ADS WITH CORE ADSL VARS
ENHANCED ADS DAT
derive new ADS var using merged ADSL core var(s)

Figure 11: Chicken and Egg problem – circular dependency during the derivation of ADSL and some ADS (ADaM) data sets

There are two possible ways to handle this problem.

These are explained on the derivation of vital signs (ADVS) where baseline definition within ADVS is needed first to select the right weight, height and BMI records for the ADSL variables baseline weight (BASEWEIG), baseline height (BASEHEIG) and baseline BMI (BASEBMI).

Way 1: Create a ADVS version 1 data set for baseline parameters only (ABLFN and BASETYPE)

Afterwards derive ADSL where, based upon the baseline flag from ADVS, the variables BASEWEIG, BASEHEIG and BASEBMI are derived.

In a post step (after ADSL is completely available) derive the complete ADVS dataset where other ADSL core variables are available and might be needed for further derivations.

Way 2: An alternative approach would be to create items that are needed for ADSL twice: first temporarily derive ABLFN and BASETYPE for BASEWEIG BASEHEIG and BASEBMI, as needed for the ADSL derivation. And later, during the derivation of the analysis data set ADVS, derive these variables again. Here it must be ensured, that in case of circular dependencies the same algorithm is applied, to ensure consistency. This requires even closer attention during the implementation and validation. The ADVS example given here is a rather easy example where the consistency check during validation is easy to do (see example below).

```
ADSL.BASEWEIG = ADVS.AVAL where ABLFN = 1(Y) and PARAMCD = "WEIGHT"
ADSL.BASEHEIG = ADVS.AVAL where ABLFN = 1(Y) and PARAMCD = "HEIGHT"
ADSL.BASEBMI  = ADVS.AVAL where ABLFN = 1(Y) and PARAMCD = "BMI"
```

But in more complex derivations or iteration steps, it may not be as easy to identify when derived variables are used within a complex derivation.

Year-long experience at Bayer has shown that Way 1 is the chosen way.

COLLECT NEW VARIABLES AUTOMATICALLY TO THE METADATA

During time critical analysis data set derivation phase, it quickly happens that additional variables are needed. Certainly first the ADaM metadata would have to be updated to reflect this additional need. In hectical times this can quickly be forgotten or result in a burden. For such situations we have macro %update_ads_meta. It collects variables that are present in ADaM data sets but are missing in the study ADaM metadata. It adds those automatically to the metadata with name, label, length, type and provides an overview to the programmer – for review and for completing the derivation documentation (fill column DERI_TXT).

APPLY METADATA

Another quite obvious advantage of having metadata controlled ADaM is that the statistical programmer doesn't need to take care about applying the right label to data sets, or the right label and format to variables.

This can be done automatically by a tiny macro that reads in the attributes from the ADaM metadata and applies or re-applies them whenever needed. No need to worry about correct labels or late typo corrections. Labels or formats will for sure fit to the defined metadata.

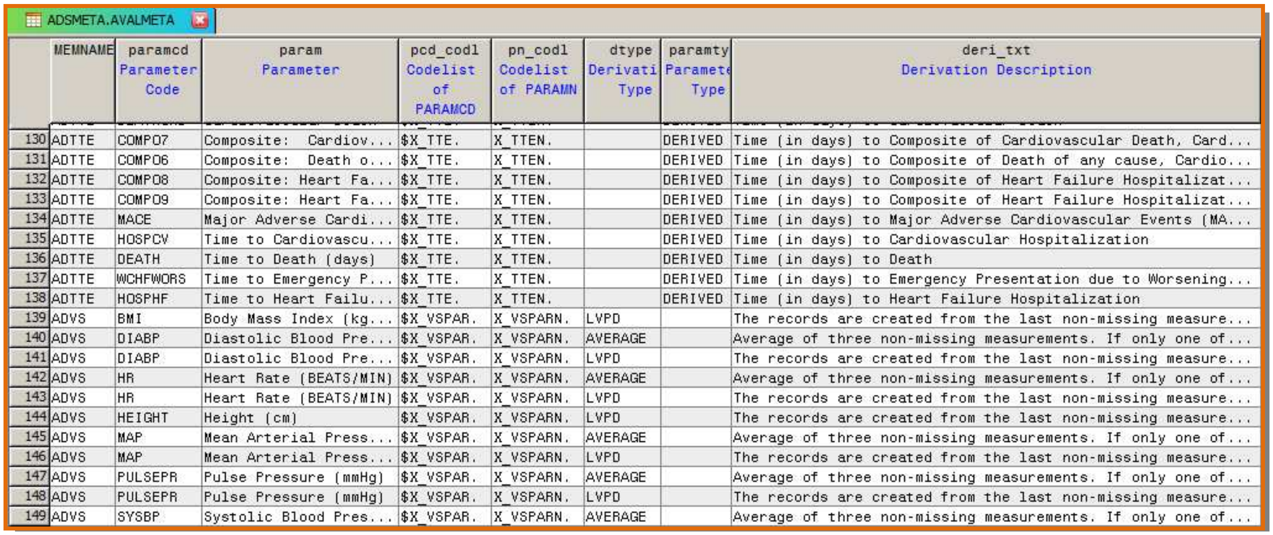
DOCUMENT DERIVED RECORDS

Study analysis data documentation does not only require specification about the analysis data sets and the analysis variables (variables within the analysis data sets). Beyond that, it is required to provide analysis value level metadata. Visually this is documentation about records within analysis data sets.

To support this step in AdaMap, macro %find_derived_recs is provided that collects all unique records from all derived BDS analysis data sets per PARAMCD and PARAM where ADaM variables DTYPE or PARAMTYP are not empty.

The created analysis value metadata set is called AVALMETA (see Figure 12).

For each of those collected records in AVALMETA a clear definition and derivation specification has to be documented in column DERI_TXT. Certainly this derivation description is nothing that comes out of the blue, but should come from the SAP/programming spec. As part of the validation, a cross check between SAP, implemented algorithm and DERI_TXT in AVALMETA should be performed.



	MEMNAME	paramcd Parameter Code	param Parameter	pcd_codl Codelist of PARAMCD	pn_codl Codelist of PARAMN	dtype Derivati Type	paramty Paramete Type	deri_txt Derivation Description
130	ADTTE	COMPO7	Composite: Cardiov...	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Composite of Cardiovascular Death, Card...
131	ADTTE	COMPO6	Composite: Death o...	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Composite of Death of any cause, Cardio...
132	ADTTE	COMPO8	Composite: Heart Fa...	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Composite of Heart Failure Hospitalizat...
133	ADTTE	COMPO9	Composite: Heart Fa...	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Composite of Heart Failure Hospitalizat...
134	ADTTE	MACE	Major Adverse Cardi...	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Major Adverse Cardiovascular Events (MA...
135	ADTTE	HOSPCV	Time to Cardiovascu...	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Cardiovascular Hospitalization
136	ADTTE	DEATH	Time to Death (days)	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Death
137	ADTTE	WCHFWORS	Time to Emergency P...	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Emergency Presentation due to Worsening...
138	ADTTE	HOSPHF	Time to Heart Failu...	$X_TTE.	X_TTEN.		DERIVED	Time (in days) to Heart Failure Hospitalization
139	ADVS	BMI	Body Mass Index (kg...	$X_VSPAR.	X_VSPARN.	LVPD		The records are created from the last non-missing measure...
140	ADVS	DIABP	Diastolic Blood Pre...	$X_VSPAR.	X_VSPARN.	AVERAGE		Average of three non-missing measurements. If only one of...
141	ADVS	DIABP	Diastolic Blood Pre...	$X_VSPAR.	X_VSPARN.	LVPD		The records are created from the last non-missing measure...
142	ADVS	HR	Heart Rate (BEATS/MIN)	$X_VSPAR.	X_VSPARN.	AVERAGE		Average of three non-missing measurements. If only one of...
143	ADVS	HR	Heart Rate (BEATS/MIN)	$X_VSPAR.	X_VSPARN.	LVPD		The records are created from the last non-missing measure...
144	ADVS	HEIGHT	Height (cm)	$X_VSPAR.	X_VSPARN.	LVPD		The records are created from the last non-missing measure...
145	ADVS	MAP	Mean Arterial Press...	$X_VSPAR.	X_VSPARN.	AVERAGE		Average of three non-missing measurements. If only one of...
146	ADVS	MAP	Mean Arterial Press...	$X_VSPAR.	X_VSPARN.	LVPD		The records are created from the last non-missing measure...
147	ADVS	PULSEPR	Pulse Pressure (mmHg)	$X_VSPAR.	X_VSPARN.	AVERAGE		Average of three non-missing measurements. If only one of...
148	ADVS	PULSEPR	Pulse Pressure (mmHg)	$X_VSPAR.	X_VSPARN.	LVPD		The records are created from the last non-missing measure...
149	ADVS	SYSBP	Systolic Blood Pres...	$X_VSPAR.	X_VSPARN.	AVERAGE		Average of three non-missing measurements. If only one of...

Figure 12: Example for analysis value level metadata documentation (DERI_TXT)

DOCUMENTATION

Experience shows that it is quite helpful to prepare documentation directly at the source where and at the time when something is created. It results in increased quality of the documentation and it is less likely forgotten to update documentation compared to maintaining a separate document. There are a few cases where this principle is used in the process surrounding AdaMap.

METADATA AS DOCUMENTATION

The creation and maintenance of metadata serves several purposes:

- Metadata are part of the **pre-specification** of intended analysis data set derivation required by authorities and for compliant work.
- In case of automatic mapping solutions like AdaMap, it also serves as **mapping definition**.
- High quality of **documentation**.
  Matching metadata with good derivation descriptions makes it easier to understand the analysis data sets for all parties (programmer, statisticians, health authority reviewer, CROs).
  The derivation description is used also as input for the define.xml.

Having this in mind there was lots of attention put to clear and unambiguous derivation description (DERI_TXT) during the set-up of the standard ADaM mapping metadata. Well and nicely described derivations reduce the work for every study as these can be reused again and again.

ENFORCE DOCUMENTATION UPDATE

If a derivation METHOD is changes in the metadata then it is required to change the derivation description. Both changes go hand in hand and if one changes, the other needs to change as well. This is automatically enforced by the metadata management macro (%SetAdsMeta) and also ensured as a quality check of the check metadata macro (%check_ads_meta).

In the following call the imputation method is changed to use the minimal possible date but the respective derivation description DERI_TXT column is not updated.

```
%setAdsMeta(
    members = adsmeta.adcm
  , vars  = ASTDT
  , method   = %nrstr(=%_adamap_create_adt(rule=min, createFlagVar=Y))
);
```

This will result in a warning in following log warning:

```
WARNING: AdaMap - setAdsMeta - change in METHOD requires change in DERI_TXT
```

Figure 13: Example for controlled metadata changes

DERIVATION COMMENT IN SAS CODE

As explained earlier, informative SAS comments for custom derivations. Variable name, format name and of particular important the derivation description DERI_TXT of the intended derivation is printed based upon the information in the metadata. See also Section “Custom ”.

```
*< AVISITN;
/** LABEL = Analysis Visit (N) - Format:_AVISIT - Outform: 8.3
 ** A numeric representation of AVISIT. AVISITN may contain the visit number as
 ** observed (i.e., from SDTM VISITNUM), derived visit numbers, time window numbers,
 ** conceptual description numbers (such as Average, Endpoint, etc.) or a combination
 ** of any of these **/

Data ADS.ADLB;
   SET ...
```

Figure 14: Quality SAS comment for custom derivation programs

QC AND VALIDATION

The QC and validation activities are performed on various steps to ensure adherence to standards and correct implementation of analysis data sets. All validation activities have to follow the respective standard operation manuals (SOPs). General validation process is not a main topic of this paper. In this section the focus is on some general remarks regarding analysis data set validation.

Key message is that validation of mapping goes together with validation of the metadata definition where core elements of the mapping are embedded using AdaMap.

A. ADaM Metadata check

The macro %check_ads_meta has to be used at the end of the study ADaM metadata creation process, to ensure completeness and structural correctness of all ADaM Metadata data sets. Bayond that also adherence towards Standards ADaM metadata are implemented. Findings are listed in the SAS log.

Beyond automatic checks of course manual checks between Metadata and SAP specification have to be done!

```
ERROR: CHECK_ADS_META - Sequence number 12 is specified multiple times in panel ADXA.
ERROR: CHECK_ADS_META - Sequence number 40 is specified multiple times in panel ADXA.
NOTE: CHECK_ADS_META - Will check structure of ADS meta data
ERROR: CHECK_ADS_META - DERI_TXT is not allowed to be missing for derived variables!(ADCE.STATUS
METHTYP=derived DERI_TXT= )
WARNING: CHECK_ADS_META - METHOD is not allowed to be missing for DERI_WAY in 1-4!(ADCE.ANL01FN deri_way=4 )
ERROR: CHECK_ADS_META - OUTFORM contains an invalid value! (ADDS.EXTENNY OUTFORM= )
ERROR: CHECK_ADS_META - DERI_TXT is not allowed to be filled for not derived variables !(ADDS.EXTENNY
METHTYP=decode DERI_TXT=Decode of EXTENNYN )
ERROR: CHECK_ADS_META - OUTFORM contains an invalid value! (ADDS.EXTENNYN OUTFORM= )
WARNING: CHECK_ADS_META - OUTFORM should be SDTM+US when METHTYP=SDTM+US! (ADDS.EXTENNYN OUTFORM= )
WARNING: CHECK_ADS_META - TYPE should be SDTM+US when METHTYP=SDTM+US! (ADDS.EXTENNYN TYPE=N )
ERROR: CHECK_ADS_META - VAR_LINK cannot be missing when vartype is not missing!(ADEGM.EGLERTFN VAR_LINK=
VARTYPE=code )
ERROR: CHECK_ADS_META - DERI_TXT is not allowed to be filled for not derived variables !(ADEX.EXADJN
METHTYP=SDTM+US DERI_TXT=EX.EXADJN)
ERROR: CHECK_ADS_META - METHTYP contains an invalid value! (ADMH.BMQ1389 METHTYP= )
ERROR: CHECK_ADS_META - DERI_TXT is not allowed to be filled for not derived variables !(ADMH.BMQ1389
METHTYP=
WARNING: CHECK_ADS_META - No decode available for variable ADXL-->ANL01FN
```

Figure 15: Example log findings from the ADaM metadata check

B. ADaM Data checks

Study ADaM data has to be checked for compliance to ADaM metadata.

Ensure that the completed ADaM data set reflects the ADaM metadata accurately.

- Check that all decode variables contain decodes of the related code variables
- Check for completeness of formats / codelists
- Check that no required observations from SDTM+ are removed or SDTM+ values changed (traceability check)
- Check that all derived observations are marked as such (DTYPE or PARAMTYP not missing)
- Check that all subject related SDTM+ data sets are used in ADaM data
- Check that all Variables with DERI_WAY = “custom” are derived.

All previous checks can be processed with the macros %check_ads. What is being checked is printed in detail to the log to give the user sufficient feedback and control what is happening within the check macros. See next two figures below.

The check, that study ADaM data complies with CDISC ADaM requirements, is done with the Pinnacle validator.

```
150: ERROR: CHECK_ADS_DAT - ADS.ADAE->TRTEMFL does not match TRTEMFN!
151:        TRTEMFL=NO         TRTEMFN=N
       Further decode checks of this variable suppressed! (x2)
248: WARNING: CHECK_ADS_DAT - SDTMP values in 2542 not derived observations have changed!
249:          Have compared the following variables: AMENDNO DSCAT DSCATN DSCNSNYN DSCNTYPN
              DSCONSNY DSCONTYP DSDECOD DSDECODN DSHOSP
303: WARNING: CHECK_ADS_DAT - No variable for metadata entry ANL01FL found!
305: WARNING: CHECK_ADS_DAT - Variable ABLFL not in ADS.ADEGM!
306: WARNING: CHECK_ADS_DAT - Variable ABLFN not in ADS.ADEGM!
422: ERROR:  CHECK_ADS_DAT - Key-Variables (STUDYID USUBJID SEQ) are not exist on ADS.ADEGM!. -
Will not perform KEYS checks!
528: ERROR:  CHECK_ADS_DAT - ADS.ADLB is not unique by STUDYID,USUBJID,SEQ!
```

Figure 16: Example for log findings during automatic ADaM data check

```
INFO: CHECK_ADS - Will check ADAE!
INFO: CHECK_ADS_DAT - Will merge ADSL core variable metadata to ADSMETA.ADAE
INFO: CHECK_ADS_DAT - Will check metadata compliance
INFO: CHECK_ADS_DAT - No metadata compliance error found. Great Job!
INFO: CHECK_ADS_DAT - Will perform decode / codelist checks!
INFO: CHECK_ADS_DAT - No wrong decode values found. Great Job!
INFO: CHECK_ADS_DAT - Will need to perform extra checks on following variables:  ADURU
INFO: CHECK_ADS_DAT - Will check variable ADURU against codelist UNIT!
INFO: CHECK_ADS_DAT - Will check number of non-derived observations between ADS.ADAE and
SDTMP.AE(WHERE=(AETERM NE ''))
INFO: CHECK_ADS_DAT - Found 1358 non-derived observations in both data sets. Great Job!
INFO: CHECK_ADS_DAT - Will check if SDTMP variables are unchanged
INFO: CHECK_ADS_DAT - Found no changed SDTMP values. Great Job!
INFO: CHECK_ADS_DAT - Will check keys variables as defined in ADS.AD
INFO: CHECK_ADS_DAT - ADAE is defined in ADS.AD. Great Job!
INFO: CHECK_ADS_DAT - Will check if keys variables is defined in ADS.AD
INFO: CHECK_ADS_DAT - Keys variables are defined in ADS.AD as STUDYID,USUBJID,SEQ. Great Job!
INFO: CHECK_ADS_DAT - Will check if keys variables exist in ADS.ADAE
INFO: CHECK_ADS_DAT - All keys variables are exist in ADS.ADAE. Great Job!
INFO: CHECK_ADS_DAT - Will check if any duplicate (by STUDYID,USUBJID,SEQ) exist in ADS.ADAE
INFO: CHECK_ADS_DAT - No duplicated found in ADS.ADAE. Great Job!
```

Figure 17: Overview of all checks that are performed by macro check_ads. Shown for example ADAE

C. Analysis dataset validation

Beside the automatic validation of analysis data sets regarding structural and CDISC compliance, of course also study specific validation has to be performed according to the validation SOP. This requires a combination of necessary steps like double programming, source code review, log review and cross check with ADaM Metadata specs (and SAP).

CONCLUSION

Study Analysis data set derivation is science driven and must ensure the analysis can be performed as specified in the SAP. The degree of standardized mapping of ADaM compared to SDTM is smaller. In consequence the used mapping solution needs to be flexible enough to allow the required freedom to derive ADaM datasets but on the other hand helps the statistical programmer as best as possible with robust automatic mapping of standard derivations. A key element is to make use of metadata control of mapping in the optimal way. Several concepts how this was achieved in AdaMap have been discussed in this paper:

- ❑ AdaMap is an embedded environment. Mapping macros depend on the underlying AdaMap macro environment are interlinked with the ADaM metadata
- ❑ It fully utilizes the provided SDTM+ data and metadata and ensures traceability from SDTM+ to ADaM. Including QC
- ❑ Takeover of SDTM+ variables and inheritance of their original attributes (label, type, length, core) → less meta data maintenance ☺
- ❑ Support of variable wildcards #
- ❑ Support of adamap macros that create multiple derived variables at once clear documentation in metadata with =%_adamap_ref(refvar=AENDT)
- ❑ Automatic handling of decode creation
- ❑ Automatically adds labels and codelist to ADS data sets → apply how often you wish
- ❑ Merge of Core ADSL in multiple stages possible, with consistency check

ACKNOWLEDGMENTS

AdaMap would not have been possible without Michael Weiss. He and I developed the key concepts of AdaMap together and he implemented all the macros. Without Michael AdaMap wouldn’t be what it is today.
In addition I want to thank my complete statistical analyst team (CD2). With such a great team even the most challenging projects can be successfully tackled.

CONTACT INFORMATION

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

Elena Glathe
Bayer AG
Work Phone: +49 (0)30-468-17731
Email: elena.glathe@bayer.com

Brand and product names are trademarks of their respective companies.