Paper DH11

Flow dependencies in ADaM generation

Jeppe Rømer Juul, Novo Nordisk A/S, Copenhagen, Denmark
Niels-Kristian Kjøller, Independent, Copenhagen, Denmark
Ari Siggaard Knoph, Novo Nordisk A/S, Copenhagen, Denmark

ABSTRACT

Generating the ADaM ADSL dataset often involves deriving a number of variables that advantageously can be derived in other ADaM datasets. The ADSL program should extract the relevant information from other ADaM datasets when possible, and thereby avoid repeated calculations and/or derivations. Likewise, it is often necessary to distribute variables from ADSL to other ADaM datasets in which case special attention is required to avoid circular dependencies.

In this paper, we present a metadata driven approach to this problem. We create an intermediate "pre-ADSL" dataset before the final ADSL. For all other ADaM datasets, variables to be included from ADSL are specified in metadata and using a macro the ADSL variables are distributed to all other ADaM datasets after ADSL is finalized. This approach ensures single point of definition for all variables and allows us to add/remove ADSL variables without updating the code in any ADaM programs.

INTRODUCTION

When ADaM datasets are created it is important to take into account dependencies of other datasets in the derivation of each variable. For instance, in order to determine if an adverse event is treatment emergent in the dataset ADAE, the treatment start date is needed from the subject level dataset ADSL. Therefore, ADSL will often be created prior to ADAE and most other ADaM datasets.

The interdependencies of different ADaM datasets get complicated when we consider derivation of core variables, which should be present in all or nearly all datasets. Core variables typically include population flags such as the full analysis set flag FASFL and covariates of the primary analysis, which for instance could be the body mass index at baseline BMIBL. Both the FDA technical conformance guide [1] and the ADaM implementation guide [2] require that such variables are included in each ADaM dataset. The derivation of core variables and the process of distributing them to all other datasets, known as denormalization of the variables, introduce a risk of creating circular dependencies between ADaM datasets.

HOW CORE VARIABLES COMPLICATE THE ADAM FLOW

The most convenient place to derive core variables vary. Population flags such as FASFL are derived in ADSL. The same is the case for some typical covariates such as the verified stratification factor STRATAV.

On the other hand, baseline values such as BMIBL are most easily derived from the ADaM dataset containing the parameter in question. For BMI at baseline, this would be the vital signs analysis dataset ADVS. The reason why it can be complicated to derive BMIBL without the ADVS dataset is that the baseline value may depend on several data derivation rules such as imputation, visit reallocation, evaluation of analysis eligibility, etc. Furthermore, the parameter itself may be derived from other observed parameters, which is often the case for BMI. All these derivations are part of the analysis program generating the ADVS dataset, and it is therefore easy to derive BMIBL after ADVS has been created, but very cumbersome to do it before.

Thus, variables such as FASFL and STRATAV are most conveniently derived in ADSL and subsequently copied to ADVS, while variables such as BMIBL are most conveniently derived in ADVS and subsequently copied to ADSL. This creates a circular dependecy between the ADSL and ADVS datasets (see Figure 1a).

Another complication arises when the selection of core variables is changed late in trial conduct. This could for instance be the case if the statistical analysis plan is updated to include more or other covariates in the statistical model for the primary analysis. As an example, consider the case where the verified strata STRATAV is replaced by the randomised strata STRATAR in the statistical model (see Figure 1b). In this case, all programs generating ADaM datasets needs to be modified, if the core variables have been included by explicitly merging them on from ADSL.

In the following two sections, we will explain how we have addressed these two complications related to core variables.

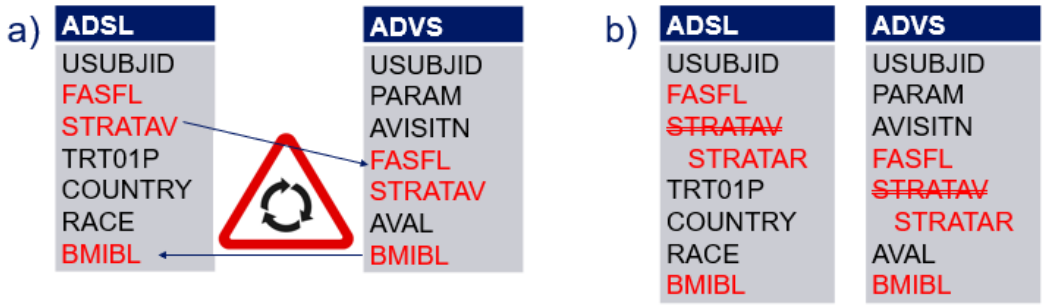

Figure 1 ***a)*** *First complication related to core variables: FASFL, STRATAV, and BMIBL should be included both in ADSL and ADVS. FASFL and STRATAV are most conveniently derived in ADSL and subsequently copied to ADVS while BMIBL is most conveniently derived in ADVS and subsequently copied to ADSL. This creates a circular dependecy.* ***b)*** *Second complication related to core variables: Late in trial conduct the set of core variables is changed from including STRATAV to instead include STRATAR. This may mean that all programs creating ADaM datasets must be modified.*

IN WHICH ORDER SHOULD ADAM DATASETS BE CREATED TO AVOID CIRCULAR DEPENDECIES?

Different strategies can be applied to handle the circular dependencies that arise when core variables are denormalized to all ADaM datasets [3-6].

One approach is to first create the ADSL dataset including all core variables, and subsequently create all other ADaM datasets (see *Figure 2*a). This gives a very simple and intuitive programming flow. The downside is that baseline variables such as BMIBL are derived without having the associated ADaM dataset ADVS available. Therefore, all data derivations related to the parameter such as visit reallocation, analysis flags, imputation rules, etc. must be done in the ADSL program as well as in the ADVS program, or they may be included in macros that are called from both programs. This complicates the programs making each dataset.

A closely related approach is to first create a dedicated dataset ADCORE containing all core variables (see *Figure 2*b). Afterwards, all analysis datasets including ADSL can be generated in parallel. The advantage of this approach is that the ADSL dataset can be updated with respect to other variables than the core variables without affecting any other datasets. However, baseline variables are still derived without the associated ADaM dataset.

A third strategy is to first create the ADSL dataset with all variables except baseline variables. Next, other analysis datasets such as ADVS are created. As a last step, a dedicated dataset ADBASE containing baseline values is created (see *Figure 2*c). This approach has the advantage that the baseline variables are created from the associated ADaM dataset, and their derivations therefore becomes trivial. Alas, this approach does not fulfil the requirement of all core variables being present in each ADaM dataset. Also, it is inconvenient not to have the baseline variables included in the datasets where they are needed when creating tables, figures and listings.

A fourth strategy is to create a intermediate dataset without core variables, and subsequently update the datasets with these variables. Specifically, a “pre-ADSL” dataset can be created, which includes all other variables than baseline variables. Then all other datasets can be created, but still without core variables. Next, ADSL is updated with baseline variables, and finally all other datasets are updated with core variables (see *Figure 2*d). This approach to denormalizing core variables is mentioned in the ADaM implementation guide v1.2 section 3.2 [2] and is also our preferred and recommended approach. The advantage of this strategy is that all variables are derived in the most convenient order while still having core variables present in all datasets.

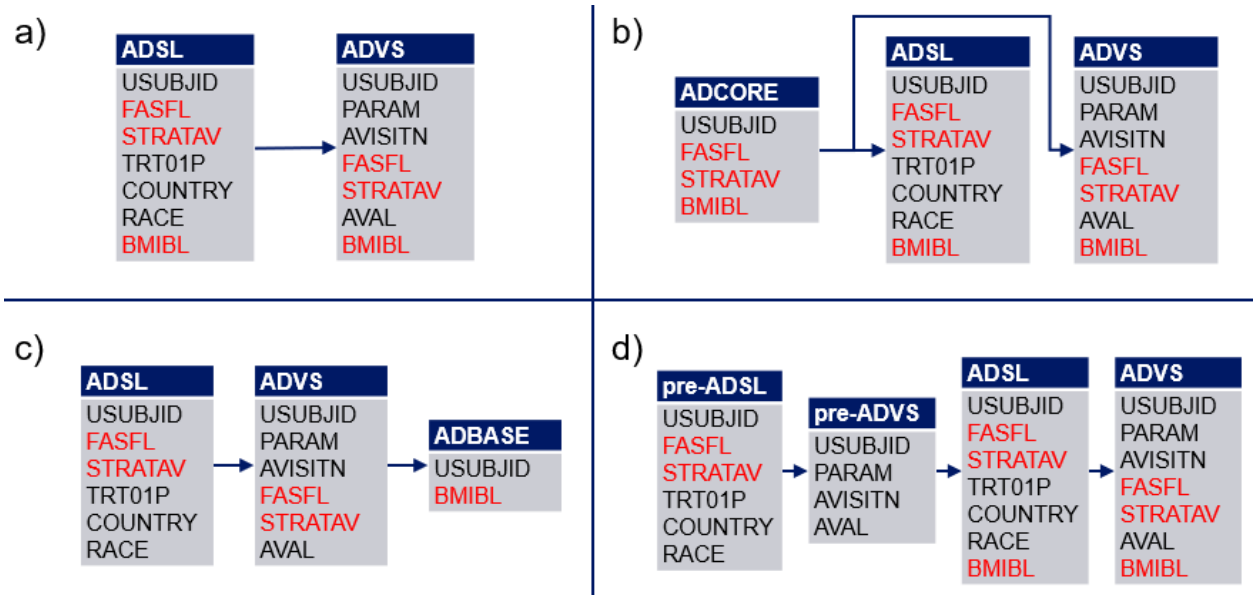

Figure 2: Four strategies for distributing core variables between datasets in the ADaM flow. ***a)*** *All core variables are derived in ADSL.* ***b)*** *Core variables are created in a dedicated dataset and subsequently distributed to all other datasets, which can be created in parallel.* ***c)*** *Baseline variables are created last in a dedicated dataset.* ***d)*** *Our preferred strategy: A "pre-ADSL" dataset is created, which includes all ADSL records except baseline variables. When all other analysis datasets have been created, ADSL is finalized by including baseline variables. Subsequently, core variables are denormalized to all other datasets.*

HOW CAN THE SELECTION OF CORE VARIABLES BE CHANGED WITHOUT MODIFYING ALL ADAM PROGRAMS?

The typical way of denormalizing core variables from ADSL to all other datasets is by using a simple data merge. For instance, the following code could be included in all programs creating ADaM datasets except ADSL:

```
* Add core variables from ADSL;
data advs_with_core;
   merge advs_without_core
         adam.adsl(keep=usubjid fasfl stratav bmibl);
   by usubjid;
run;
```

However, if the selection of core variables is changed late in trial conduct, then this approach will require that all programs creating ADaM datasets are modified and consequently re-reviewed. To avoid that scenario, we have followed a metadata driven approach to denormalizing core variables.

We have defined the metadata dataset MDCOL, which is based on the input to define-xml. This dataset includes one row per analysis dataset per column, and contains information regarding the label, length, type, format, and order of each variable (see Figure 3). The main purpose of MDCOL is applying these attributes to each variable as the last programming step when an ADaM dataset is generated. We have furthermore included the column COREFL in the dataset to identify the core variables.

TABLE	COLUMN	LABEL	COREFL
ADSL	USUBJID	Unique Subject Identifier	
ADSL	FASFL	Full Analysis Set Population Flag	Y
ADSL	STRATAV	Strata from Verification Source	Y
ADSL	TRT01P	Planned Treatment for Period 01	
ADSL	COUNTRY	Country	
ADSL	RACE	Race	
ADSL	BMIBL	Body Mass Index at Baseline	Y

Figure 3: Selected columns of the metadata dataset MDCOL including the variable COREFL that identifies the core variables. MDCOL is created from the input to define-xml and can also be used to finalize each ADaM dataset by applying each variable with the correct label, length, format, etc.

When the “pre-ADSL” dataset has been updated with baseline variables, and all core variables are ready to be denormalized, we call the custom macro *adamaddcorevars* once for every other ADaM dataset. This macro, which is included in appendix 1, goes through the following four steps:

1. Extract list of core variables from the metadata dataset MDCOL
2. Drop any core variables already existing in the input ADaM dataset
3. Read the core variables from ADSL
4. Join the input ADaM dataset and ADSL by USUBJID to add the core variables

Using this macro, the selection of core variables can be changed simply by changing the metadata dataset MDCOL. Since this dataset is based on the input for define-xml, it is ensured that define-xml and the physical ADaM datasets are always in correspondence.

CONCLUSION

The order in which ADaM datasets are created impacts how core variables can be derived and denormalized. In this process it is necessary to pay special attention to circular dependencies between ADaM datasets. Several different strategies can be applied to avoid such circular dependencies. We recommend to first create the intermediate dataset “pre-ADSL” without baseline variables, then create intermediate versions of all other ADaM datasets without core variables. Subsequently, ADSL can be updated with baseline variables and core variables can be denormalized to all other ADaM datasets. We recommend to carry out the denormalization using a macro that identifies the core variables from a metadata dataset. Using this approach, the selection of core variables can be changed without modifying any ADaM program.

REFERENCES

[1] Study Data Technical Conformance Guide, March 2018,
Available at https://www.fda.gov/media/88173/download

[2] Analysis Data Model (ADaM) Implementation Guide, Version 1.2, Published by CDISC 03-Oct-2019, ,
Available at https://www.cdisc.org/standards/foundational/adam/adamig-v12

[3] "The 5 Biggest Challenges of ADaM", Peterson T and Izard D, NESUG 2010,,
Available at https://www.lexjansen.com/nesug/nesug10/ph/ph05.pdf

[4] "It Depends On Your Analysis Need", Minjoe S, PharmaSUG 2015, ,
Available at https://www.pharmasug.org/proceedings/2015/DS/PharmaSUG-2015-DS11.pdf

[5] "ADaM mapping - key considerations for a metadata driven realization", Glathe E, PhUSE 2017, ,
Available at https://www.phusewiki.org/docs/Conference%202017%20SI%20Papers%20NEW/SI06.pdf

[6] "Challenges in Developing ADSL with Baseline Data", Liu H and Pang H, PharmaSUG 2015, ,
Available at https://www.pharmasug.org/proceedings/2015/PO/PharmaSUG-2015-PO22.pdf

CONTACT INFORMATION

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

Jeppe Rømer Juul
Novo Nordisk A/S
Novo Allé 1
2880 Bagsværd
Denmark
Work Phone: +4530750940
Email: jxju@novonordisk.com

APPENDIX 1: ADAMADDCOREVARS MACRO

```
/************************************************************************
  Description: Macro will add subject level variables. No subsetting
               of data is done in this macro - this should be handled by
               the user either as a where-clause on the output dataset or
               in subsequent programming steps.
  Parameters : inds           : Dataset which should have subject level
                                variables added.
                                Dataset name can be one or two level.
                                No dataset options can be applied.
               outds          : Name of output dataset.
                                Dataset name can be one or two level.
                                Dataset options can be applied.
               adslds         : Name of subject level dataset. Dataset name
                                can be one or two level. Dataset options
                                can be applied.
                                Default value is ADAM.ADSL.
               additional_vars: Additional variables to add from subject
                                level data that are not already marked
                                with COREFL. Use a space separated list
                                of variable names.
  Input      : Dataset specified in the parameter inds, subject level
               data specified in adslds and METADATA.MDCOL.
  Usage notes: The metadata dataset MDCOL should contain variable COREFL.

  Example    : %adamaddcorevars(inds            = advs_without_core
                              , outds           = advs_with_core
                              , additional_vars = %str(intrdurd ontldurd))
  Programmer :
************************************************************************/


%macro adamaddcorevars( inds            =
                      ,outds            =
                      ,adslds           = adam.adsl
                      ,additional_vars =
                      );

   %put NOTE: ************************************************************;
   %put NOTE- Macro &sysmacroname. is executing.;
   %put NOTE- ************************************************************;

   %let libname = %upcase(work);
   %let memname = %upcase(&inds.);
   %if %sysfunc(find(&inds., .)) > 0 %then
   %do;
      %let libname = %upcase(%scan(&inds., 1, .));
      %let memname = %upcase(%scan(&inds., 2, .));
   %end;

   %* Define variable list to add from subject level data    ;
   %* Do not take STUDYID and USUBJID from subject level data;
   %* as they are the variables used to join on              ;
   %let core = ;
   proc sql noprint;
      select catt('b.', column)
      into :core separated by ', '
      from metadata.mdcol
      where upcase(corefl) = "Y" and table = "ADSL" and upcase(column) not in ("STUDYID"
"USUBJID")
      ;
      %* Create a list of variables to drop from input data;
      %* as they will be added from subject level data     ;
      %let droplist = ;
      select name
      into :droplist separated by ' '
      from dictionary.columns
      where upcase(libname) = "&libname." and upcase(memname) = "&memname." and find("&core.",
strip(name), 'i') > 0
      ;
   quit;

   %* Add subject level data to input data           ;
   %* Add ', b.' to all additional variables (if any);
```

```
   proc sql noprint;
      create table &outds. as
      select a.*
           %if %length(&core.)       > 0 %then , &core.;
           %if %length(&additional_vars.) > 0 %then %sysfunc(prxchange(s/(\w+)/%str(,) b.$1/, -1,
&additional_vars.));
      from &inds.%if %length(&droplist.)  > 0 %then (drop = &droplist.); as a
         inner join
           &adslds. as b
      on a.studyid = b.studyid and a.usubjid = b.usubjid
      ;
   quit;

   %put NOTE: ************************************************************;
   %put NOTE- Macro &sysmacroname. ended.;
   %put NOTE- ************************************************************;
%mend;
```