PharmaSUG 2023 - Paper SI-106

Automation of Dataset Programming Based on Dataset Specification

Liqiang He, Atara Biotherapeutics, Inc.

ABSTRACT

In the clinical trial field, standard datasets, such as SDTM domains and ADaM datasets, are an integral part of electronic submission package, and are also a prerequisite for TFL generation. Dataset programming is a time consuming, tedious task for SAS® programmer. A highly efficient, automatic programming for dataset generation will prevent manual programming typos, save programming time and resources, and deliver high-quality work. Dataset specification is a detailed instruction for dataset programming and a major reference for dataset validation. This paper demonstrates a new practical approach to automate dataset programming based on dataset specification. It is pivotal for successful auto-programming to rewrite variable derivation to SAS-readable with the aid of keywords and punctuation marks in the dataset specification.

INTRODUCTION

As CDISC standardized SDTM/ADaM datasets are widely applied in the clinical trial field, more and more programming automation strategies and techniques are reported in ADaM programming, compliance checking of metadata against CDISC standards etc on [1-4]. However, automatic programming for SDTM dataset generation is seldom reported. Compared to ADaM programming, SDTM dataset programming faces more challenges since SDTM datasets are generated directly from raw datasets, which are not generated under the CDISC standards.

In general, a primary programmer creates a program based on a written dataset specification, and the program in turn generates a dataset. This specification is a detailed instruction for the dataset generation, including variable name, label, data type, length, origin, derivation etc on. On the other hand, this specification is also a major reference for validation programmer to validate the produced dataset. Thus, dataset specification may be used as a basis for programming automation.

This paper demonstrates a new approach to automate dataset programming based on dataset specification. This strategy can be applied to the development of SDTM domains as well as ADaM datasets.

MAIN STEPS FOR AUTOMATION

In general, a dataset specification is stored in a spreadsheet of an excel file. The following are the main steps to automate a program from a dataset specification.

STEP 1: REWRITE VARIABLE DERIVATION TO SAS-READABLE

In dataset specification, except for variable derivation, other information can be directly read by SAS. In general, variable derivation is written in plain language for human reading. To be conveniently read by SAS, this derivation needs to be rewritten in a standardized format.

The following is a simple model:

VALUE if CONDITION at DATASET where SUBSETTING

In all of them, "***if***", "***at***", "***where***" are keywords, which are convenient for SAS to take apart ***VALUE***, ***CONDITION***, ***DATASET*** and ***SUBSETTING***.

VALUE may be:

1. Constant character value. It should be included with " ". Single quote marker ' ' is not fit to be used as the first string of ***VALUE*** since it would not be successfully read into the string.
2. Constant numeric value. It should not be included with ' ' or " ".
3. Variable. It is from ***DATASET***.
4. Function. It accepts variable as arguments.

CONDITION is the condition under which, the variable is equal to ***VALUE***. It will be interpreted as "if statement". The "***if***" may be removed when ***CONDITION*** has no value.

DATASET is a dataset name. It may be two-level name if the libref is not work. When the ***DATASET*** is the produced dataset, "***at DATASET***" is not needed.

SUBSETTING is another condition, which is used to select some observations from the dataset ***DATASET***. It will be interpreted as "where statement". When ***SUBSETTING*** is missing, "***where SUBSETTING***" may be removed.

If the same ***VALUE*** comes from different datasets, each dataset may be separated by "**,**" and the model may be:

VALUE *if CONDITION1 at DATASET1 where SUBSETTING1***,** *if CONDITION2 at DATASET2 where SUBSETTING2***, ……**

If the ***VALUE*** has different values and each comes from a different dataset, then each dataset may be separated by "**;**" and the model may be:

VALUE1 *if CONDITION1 at DATASET1 where SUBSETTING1***;** ***VALUE2*** *if CONDITION2 at DATASET2 where SUBSETTING2***; ……**

If the variable derivation is more complicated and it will be better to use a macro, just written as %MACRO (MACRO is the macro name) instead.

The following is a sample for DSTERM variable in DS domain:

"COMPLETED" at rawdata.ie where not missing(iedat) or not missing(iecrit), or at rawdata.eot where upcase(EOTTRC) in ("YES") ;

"SCREEN FAILURE" at rawdata.ie where iecrit ^= 'Yes';

"STUDY ENROLLED" at rawdata.ie where ieenroll_std='Y';

"MAIN INFORMED CONSENT OBTAINED" at rawdata.dm where upcase(ICSIGN) in ("Y" "YES");

"ELIGIBILITY CRITERIA MET" at rawdata.ie where upcase(IECRIT_STD)="Y" or iewaiv_std ne ".

DSTERM has different values such as "COMPLETED", "SCREEN FAILURE", "STUDY ENROLLED", "MAIN INFORMED CONSENT OBTAINED", "ELIGIBILITY CRITERIA MET" derived from different raw datasets, and they are separated by "**;**". The value "COMPLETED" is from two datasets IE and EOT, which are separated by "**,**". The values "SCREEN FAILURE", "STUDY ENROLLED", "MAIN INFORMED CONSENT OBTAINED", "ELLIGIBILITY CRITERIA MET" are from single raw dataset, and their derivations are written in a simple model, respectively.

STEP 2: DETERMINE THE ORDER AND METHOD OF VARIABLES COMBINING INTO PRODUCT DATASET

A dataset consists of a variety of variables, and these variables may have different sources, and some variables may be directly derived from other variables in the product dataset. Therefore, variables are added to product dataset in a certain order. They may be temporarily stored in some intermediate

datasets, and then these intermediate datasets are combined horizontally or vertically, to eventually assemble into a product dataset. Thus, in the dataset specification, an additional column is needed to assign variable's combining order and intermediate dataset's combining method. For example, in the right-most column "Combination" of **Display 1**, "1 set" means that the variable combining order is the first order and the combining method is by set statement. "2 merge subject" means that the variable combining order is the second order and the combining method is by merge statement and the variable "subject" as sorting variable.

CDISC Variable Name	CDISC Variable Label	Role	Core	Type	Length	Mapping Rule	Mapping Rule (for programming)	Combination
DOMAIN	Domain Abbreviation	Identifier	REQ	Char	2	Set to "DS"	"DS"	3
USUBJID	Unique Subject Identifier	Identifier	REQ	Char	40	sdtmdata.dm.USUBJID	USUBJID at sdtmdata.DM	2 merge subject
DSSEQ	Sequence Number	Identifier	REQ	Num	8	This sequence is derived using the sort order in 'Key Variables' column in the TOC tab.	%seq	6
DSTERM	Reported Term for the Disposition Event	Topic	REQ	Char	200	Set to "COMPLETED" , if collected from rawdata.ie where iedat is not misssing or iecrit is not missing, or collected from rawdata.eot where upcase(EOTTRC) in ("Y","YES") ; Set to "SCREEN FAILURE", if collected from rawdata.ie where iecrit ^= 'Yes'; Set to "STUDY ENROLLED", if collected from rawdata.ie where ieenroll_std='Y'; Set to "INFORMED ASSENT OBTAINED" if collected from rawdata.dm where upcase(ASSIGN) in ("Y","YES"); Set to "MAIN INFORMED CONSENT OBTAINED" if collected from rawdata.dm where upcase(ICSIGN) in ("Y","YES"); Set to "DEATH" if collected from rawdata.ds where dsdecod='DEATH' or dthdat_raw ne "; or collect from rawdata.eot where (upcase(EOTTRNCR)="DEATH" or collected from rawdata.sae_argus where upcase(outcome_event)="FATAL" or collected from rawdata.sur where upcase(svsvst)="DECEASED"; Set DSTERM=upcase(EOTTRNCR), if collected from rawdata.eot where upcase(EOTTRC)='No'; if not missing(EOTTRNCRO) then DSTERM=strip(upcase(EOTTRNCR))||": "||strip(upcase(EOTTRNCRO)); Set DSTERM=dsdecod, if collected from rawdata.ds; if not missing(DSDCDOS) then dsterm=upcase(strip(dsdecod))||': '||upcase(strip(DSDCDOS)); Set to "ELIGIBILITY CRITERIA MET", if collected from rawdata.ie where	"COMPLETED" at rawdata.ie where not missing(iedat) or not missing(iecrit), or at rawdata.eot where upcase(EOTTRC) in ("Y" "YES") ; "SCREEN FAILURE" at rawdata.ie where iecrit ^= 'Yes'; "STUDY ENROLLED" at rawdata.ie where ieenroll_std='Y'; "INFORMED ASSENT OBTAINED" at rawdata.dm where upcase(ASSIGN) in ("Y" "YES"); "MAIN INFORMED CONSENT OBTAINED" at rawdata.dm where upcase(ICSIGN) in ("Y" "YES"); "DEATH" at rawdata.ds where dsdecod='DEATH' or dthdat_raw ne '', or at rawdata.eot where (upcase(EOTTRNCR))="DEATH", or at rawdata.sae_argus where upcase(outcome_of_event)="FATAL"; ifc(not missing(EOTTRNCRO), upcase(EOTTRNCR)||': '||strip(upcase(EOTTRNCRO)) , upcase(EOTTRNCR)) at rawdata.eot where EOTTRC='No' ; ifc(not missing(DSDCDOS), strip(upcase(dsdecod))||':	1 set

DOC | Version | TOC | DM | CO | SE | SV | SS | ZO | CM | EC | EX | SU | AE | DS | HO | MH | DD | CE | ...

Display 1. Partial dataset specification for DS domain

STEP 3: TRANSLATE VARIABLE DERIVATION INTO SAS CODES

Once variable derivation is rewritten and variable combining order and method are assigned in the dataset specification, next step is to read the spreadsheet into a SAS dataset by IMPORT procedure. The variable derivation in the column "Mapping Rule (for programming)" (shown in **Display 1**) is read into a string for each variable. And then the string can be translated into SAS code (s).

The simple model of variable will be translated to:

```
set DATASET;
where SUBSETTING;
if CONDITION then VARIABLE=VALUE;
```

To conveniently organize and export the translated SAS codes, the values for ***DATASET***, ***SUBSETTING***, ***VALUE*** and ***CONDITION*** are stored into different variables, respectively, as shown in **Table 1**.

No.	Variable	Dataset	Subsetting	Value
1	VARIABLE	DATASET	SUBSETTING	if CONDITION then VARIABLE= VALUE

Table 1. Disassembly of SAS codes for variable derivation in a simple model

If the same variable derives from different ***DATASET*** or ***SUBSETTING***, its codes need to be separated into different data steps. Thus, the translated SAS codes in complex model should be separated into multiple records. There will be per ***DATASET*** per ***SUBSETTING*** per record per variable, as shown in **Table 2**.

No.	Variable	Dataset	Subsetting	Value
1	VARIABLE	DATASET1	SUBSETTING1	if CONDITION1 then VARIABLE= VALUE1
2	VARIABLE	DATASET2	SUBSETTING2	if CONDITION2 then VARIABLE= VALUE2

Table 2. Disassembly of SAS codes for variable derivation in a complex model

SETP 4: REARRANGE SAS CODES FOR VARIABLE DERIVATION

In **STEP 2**, variable combining order and method are assigned in the “Combination” column of data specification. Once variable derivation is translated into SAS codes, the produced codes need to rearrange based on variable combining order and method to export orderly them. In addition, if different variables share the same ***DATASET*** and ***SUBSETTING***, their codes can be combined into one data step. Thus, these codes need to be put together to export them correctly. As shown in **Display 2**, DSSTDTC, DSTERM, DSSCAT co-exist in the same dataset rawdata.ds, rawdata.eot where (upcase(EOTTRNCR))="DEATH", rawdata.eot where upcase(EOTTRC) in ("Y" "YES"), etc on. Their codes are merged into corresponding data steps, as shown in **Display 3**.

combination	ds	subset	CDISC Variable Name	code
1 set	rawdata.dm	where upcase(ASSIGN) in ("Y" "YES")	DSSTDTC	DSSTDTC=put(input(compress(asdat_raw,' -'), date9.), is8601da.)
1 set	rawdata.dm	where upcase(ASSIGN) in ("Y" "YES")	DSTERM	DSTERM = "INFORMED ASSENT OBTAINED"
1 set	rawdata.dm	where upcase(ICSIGN) in ("Y" "YES")	DSSTDTC	DSSTDTC=put(input(compress(icdat_raw,' -'), date9.), is8601da.)
1 set	rawdata.dm	where upcase(ICSIGN) in ("Y" "YES")	PROTDTC	PROTDTC=put(input(compress(icprdat_raw,' -'), date9.), is8601da.)
1 set	rawdata.dm	where upcase(ICSIGN) in ("Y" "YES")	DSTERM	DSTERM = "MAIN INFORMED CONSENT OBTAINED"
1 set	rawdata.ds		DSSTDTC	DSSTDTC=put(input(compress(dsstdat_raw,' -'), date9.), is8601da.)
1 set	rawdata.ds		LCONTDTC	LCONTDTC = strip(dsrectx)
1 set	rawdata.ds		DSSCAT	DSSCAT = "STUDY PARTICIPATION"
1 set	rawdata.ds		DSTERM	DSTERM = strip(ifc(not missing(DSDCDOS), strip(upcase(dsdecod))||': '||upcase(strip(DSDCDOS)), upcase(dsdecod)))
1 set	rawdata.ds	where dsdecod='DEATH' or dthdat_raw ne ''	DSSTDTC	DSSTDTC=put(input(compress(dthdat_raw,' -'), date9.), is8601da.)
1 set	rawdata.ds	where dsdecod='DEATH' or dthdat_raw ne ''	DSTERM	DSTERM = "DEATH"
1 set	rawdata.eot	where (upcase(EOTTRNCR))="DEATH"	DSSTDTC	DSSTDTC=put(input(compress(eotdt_raw,' -'), date9.), is8601da.)
1 set	rawdata.eot	where (upcase(EOTTRNCR))="DEATH"	DSSCAT	DSSCAT = "STUDY TREATMENT"
1 set	rawdata.eot	where (upcase(EOTTRNCR))="DEATH"	DSTERM	DSTERM = "DEATH"
1 set	rawdata.eot	where EOTTRC='No'	DSSTDTC	DSSTDTC=put(input(compress(eotdt_raw,' -'), date9.), is8601da.)
1 set	rawdata.eot	where EOTTRC='No'	DSSCAT	DSSCAT = "STUDY TREATMENT"
1 set	rawdata.eot	where EOTTRC='No'	DSTERM	DSTERM = strip(ifc(not missing(EOTTRNCRO), upcase(EOTTRNCR)||': '||strip(upcase(EOTTRNCRO)) , upcase(EOTTRNCR)))
1 set	rawdata.eot	where upcase(EOTTRC) in ("Y" "YES")	DSSTDTC	DSSTDTC=put(input(compress(eotdt_raw,' -'), date9.), is8601da.)
1 set	rawdata.eot	where upcase(EOTTRC) in ("Y" "YES")	DSSCAT	DSSCAT = "STUDY TREATMENT"
1 set	rawdata.eot	where upcase(EOTTRC) in ("Y" "YES")	DSTERM	DSTERM = "COMPLETED"
1 set	rawdata.ie	where iecrit ^= 'Yes'	DSSTDTC	DSSTDTC=put(input(compress(iedat_raw,' -'), date9.), is8601da.)
1 set	rawdata.ie	where iecrit ^= 'Yes'	DSSCAT	DSSCAT = "SCREENING"
1 set	rawdata.ie	where iecrit ^= 'Yes'	DSTERM	DSTERM = "SCREEN FAILURE"
1 set	rawdata.ie	where ieenroll_std='Y'	DSSTDTC	DSSTDTC=put(input(compress(iedat_raw,' -'), date9.), is8601da.)
1 set	rawdata.ie	where ieenroll_std='Y'	DSSCAT	DSSCAT = "SCREENING"
1 set	rawdata.ie	where ieenroll_std='Y'	DSTERM	DSTERM = "STUDY ENROLLED"
1 set	rawdata.ie	where not missing(iedat) or not missing(iecrit)	DSSTDTC	DSSTDTC=put(input(compress(iedat_raw,' -'), date9.), is8601da.)
1 set	rawdata.ie	where not missing(iedat) or not missing(iecrit)	DSSCAT	DSSCAT = "SCREENING"
1 set	rawdata.ie	where not missing(iedat) or not missing(iecrit)	DSTERM	DSTERM = "COMPLETED"
1 set	rawdata.ie	where upcase(IECRIT_STD)="Y" or iewaiv_std ne ''	DSSTDTC	DSSTDTC=put(input(compress(iedat_raw,' -'), date9.), is8601da.)
1 set	rawdata.ie	where upcase(IECRIT_STD)="Y" or iewaiv_std ne ''	DSSCAT	DSSCAT = "SCREENING"
1 set	rawdata.ie	where upcase(IECRIT_STD)="Y" or iewaiv_std ne ''	DSTERM	DSTERM = "ELIGIBILITY CRITERIA MET"
1 set	rawdata.invchk		DSSTDTC	DSSTDTC=put(input(compress(insign_raw,' -'), date9.), is8601da.)
1 set	rawdata.invchk		DSTERM	DSTERM = "INFORMED CONSENT OBTAINED"
1 set	rawdata.sae_argus	where upcase(outcome_of_event)="FATAL"	DSSTDTC	DSSTDTC=put(input(compress(event_stop_date,' -'), date9.), is8601da.)
1 set	rawdata.sae_argus	where upcase(outcome_of_event)="FATAL"	DSTERM	DSTERM = "DEATH"
2 merge subject	sdtmdata.DM		USUBJID	USUBJID = strip(USUBJID)

Display 2. SAS codes for variable derivation are rearranged for DS program

```
data ds_3 (keep=subject DSSTDTC LCONTDTC DSSCAT DSTERM );
     length DSSTDTC LCONTDTC $20 DSSCAT DSTERM $200 ;
      set rawdata.ds ;
     DSSTDTC=put(input(compress(dsstdat_raw,' -'), date9.), is8601da.) ;
     LCONTDTC = strip(dsrectx) ;
     DSSCAT = "STUDY PARTICIPATION" ;
     DSTERM = strip(ifc(not missing(DSDCDOS), strip(upcase(dsdecod))||': '||upcase(strip(DSDCDOS)), upcase(dsdecod) )) ;
run;
```

```
data ds_4 (keep=subject DSSTDTC DSTERM );
     length DSSTDTC $20 DSTERM $200 ;
      set rawdata.ds ;
      where dsdecod='DEATH' or dthdat_raw ne '' ;
     DSSTDTC=put(input(compress(dthdat_raw,' -'), date9.), is8601da.) ;
     DSTERM = "DEATH" ;
run;
```

```
data ds_5 (keep=subject DSSTDTC DSSCAT DSTERM );
     length DSSTDTC $20 DSSCAT DSTERM $200 ;
      set rawdata.eot ;
      where (upcase(EOTTRNCR))="DEATH" ;
     DSSTDTC=put(input(compress(eotdt_raw,' -'), date9.), is8601da.) ;
     DSSCAT = "STUDY TREATMENT" ;
     DSTERM = "DEATH" ;
run;
```

```
data ds_6 (keep=subject DSSTDTC DSSCAT DSTERM );
     length DSSTDTC $20 DSSCAT DSTERM $200 ;
      set rawdata.eot ;
      where EOTTRC='No' ;
     DSSTDTC=put(input(compress(eotdt_raw,' -'), date9.), is8601da.) ;
     DSSCAT = "STUDY TREATMENT" ;
     DSTERM = strip(ifc(not missing(EOTTRNCRO), upcase(EOTTRNCR)||': '||strip(upcase(EOTTRNCRO)) , upcase(EOTTRNCR) )) ;
run;
```

Display 3. Partial codes for DS program

Furthermore, if variables have the same ***CONDITION, DATASET*** and ***SUBSETTING***, their codes can be combined and simplified as:

```
if CONDITION then do;
        variable1=XXX;
        variable2=XXX;
        ……
    end;
```

In short, SAS codes for variable derivation are rearranged based on variable combining order, method, and variable sharing in data step to export them orderly to an external file.

SETP 5: EXPORT SAS CODES INTO AN EXECUTABLE FILE

Finally, all codes are exported to an external file to form a SAS program. FILENAME statement is used to set up an external file path and dataset program name. FILE statement and PUT statement are used to export all codes to an external file to form a complete dataset program. An expected dataset will be generated if the produced program works well.

When each dataset specification is put into one excel sheet and all datasets (SDTM or ADaM) specifications are wrapped up in an excel file, all dataset programs can be sequentially generated by reading these excel sheets.

DISCUSSION

CONFLICT ISSUE IN VARIABLE

There may be some inconsistency when the same variable exists in both original dataset and produced dataset. The inconsistency includes data type, length, and meaning. The following WARNING and NOTE frequently occur in manual programming. They are caused by variable inconsistency in variable length or data type, respectively.

WARNING: Multiple lengths were specified for the variable SEX by input data set(s). This can cause truncation of data.

NOTE: Invalid numeric data, 'Unknown' , at line 252 column 37.

Similar scenario also occurs in the auto-programming. If it is not handled in programming, some unexpected values for the variable will be produced in the produced dataset. It is impractical to modify SAS code in the auto-programming by the trial-and-error method as used in the manual programming. Our strategy to this challenge is to check all product dataset variables in each original dataset and find out all conflicted variables. One solution is to change these variables to temporary variables first, such as SEX to _SEX, and then drop the original variables of the original dataset on the produced dataset by keep/drop statement, and then restore the product variable name by renaming such as rename=(_SEX=SEX) in the produced dataset. In **Display 4**, RACEOTH is a numeric variable in raw dataset dm, but it is a character variable in SUPPDM and this character variable is to store the value of another variable raceos in raw dataset dm. By saving the original raceos value into a temporary variable _RACEOTH, and finally restore its name to RACEOTH after original RACEOTH in raw dataset dm was removed.

```
data ds_3 (keep= subject _SEX RFMICDTC _SITEID SUBJID USUBJID _ETHNIC RACE _RACEOTH AGE rename = (_SEX= SEX _SITEID= SITEID _ETHNIC= ETHNIC _RACEOTH= RACEOTH ));
     length _SEX $1 RFMICDTC _SITEID SUBJID $20 USUBJID $40 _ETHNIC $66 RACE $150 _RACEOTH $200 ;
     set rawdata.dm (drop=STUDYID );
     _SEX = strip(substr(sex,1,1)) ;
     RFMICDTC=put(input(compress(icdat_raw,' -'), date9.), is8601da.) ;
     _SITEID = strip(site) ;
     SUBJID = strip(subject) ;
     USUBJID = strip(strip(project)||'-'||strip(site)||'-'||strip(subject)) ;
     _ETHNIC = strip(ethnic) ;
     if RACEAME=1 then race='American Indian or Alaska Native' ;
     else if RACEASI=1 then race='Asian' ;
     else if RACEBLA=1 then race='Black or African American' ;
     else if RACENAT=1 then race='Native Hawaiian or other Pacific Islander' ;
     else if RACEWHI=1 then race='White' ;
     else if RACEOTH=1 then race='Other' ;
     if raceos ne '' then _RACEOTH = strip(raceos) ;
run;
```

Display 4. Partial code for DM program

INTERMEDIATE DATASET NAMING AND CALLING IN

A dataset program may contain plenty of intermediate datasets. How to name them and call them at a right time to eventually generate a product dataset is also a challenge. Our strategy is to sequentially name them based on the dataset combining order in the “Combination” column. For example, in DS dataset program in **Display 3**, ds_1, ds_2, ds_3, ds_4, etc on. This will easily name them and finally combine them into product dataset.

CONCLUSION

This paper provides a practical approach for dataset programming automation. It is based on dataset specification. A SAS-readable variable derivation in the dataset specification is vital for successful automation. It can be applied to SDTM domain and ADaM dataset programming and save programming time and resource and deliver high quality datasets.

REFERENCES

1. Xiangchen (Bob) Cui, Min Chen, Tathabbai Pakalapati. An Innovative ADaM Programming Tool for FDA Submission PharmaSUG 2012 - Paper DS20
2. William Wei, Rinki Jajoo, Susan Kramlik. Automation of STDM dataset integration and ADaM dataset formation PharmaSUG 2018 - Paper AD-32
3. Tracy Sherman, Aakar Shah. Automating ADSL Programming Using Pinnacle 21® Specifications PharmaSUG 2019 - Paper 158
4. CDISC 360 Project White Paper https: //www.cdisc.org/sites/default/files/2021-06/CDISC_360_Project_White_Paper.pdf

ACKNOWLEDGMENTS

The author would like to thank Sam Wang, Xiaoming Li from Atara Biotherapeutics, Inc., Thousand Oaks, CA, USA for their programming support and this paper's review and comments.

CONTACT INFORMATION

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

Liqiang He
lhe@atarabio.com
Atara Biotherapeutics, Inc.