SESUG Paper HP-120-2019

Pooled Database Loader (PDL)

From Document to Code to Deliverable: a streamlined process of combining documentation and SAS code to produce a pooled database

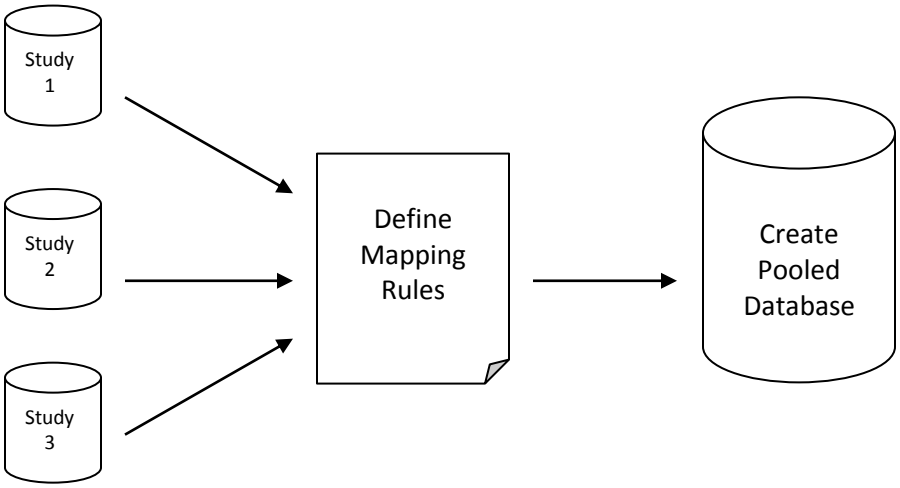
David Hartman, UCB Biosciences, Raleigh NC, USA,Pranjali Dafe, Covance Inc., Cary, NC, USA

ABSTRACT

The combining of 2 or more studies into a single database is very difficult and labor intensive. While it is virtually impossible to eliminate human intervention, a process has been created where documentation containing everything from what studies are to be pooled to the mapping rules that are to be used to convert data from multiple studies into a common structure. This information (in the form of spreadsheets) is then executed via a tool called PDL (Pooled Database Loader) to produce the pooled database. This process is completely data independent and can pool data from multiple studies together regardless of the structure of the "original" data. The purpose of this paper is to illustrate how a user of PDL goes from individual study data to a final pooled dataset. Included in this paper will be examples of the information contained in the spreadsheets as well as the various calls to PDL that will create the pooled dataset.

INTRODUCTION

The common procedure for building pooled datasets is to work from documents including the Statistical Analysis Plan (SAP) and table shells written for the purpose of analyzing integrated data for such things as safety signal detection or submission packages. Based upon those documents, a set of specifications is written, usually in the form of a Data Definition Table (DDT). Once the DDT is written then a programmer begins the task of writing code to create the pooled dataset. Depending on the complexity of the structure of the pooled dataset and the degree in which each study has to be mapped to the pooled structure, this could take dozens of pages of code. Adding new studies or just going back at a later time to understand what exactly was done in that process can require a very long learning curve, especially if the programmer doing the follow-up is not the original author. The solution to this problem is the development of a new tool known at UCB Biosciences as the Pooled Database Loader (PDL). This tool combines the documentation regarding all aspects of the data from the source to the final pooled database with SAS® code that takes the information provided by the documentation and builds the dataset.



THE INTERFACE

There is a series of 4 spreadsheets used in this process beginning with the spreadsheet containing the list of studies to be pooled and the location of their data and ending with the spreadsheet containing the mapping rules for each individual study. Once the final spreadsheet is completed then loading of the data is performed.

THE TOOL

PDL is a program written entirely in SAS®. The front end (macro call) appears as follows:

```
%pdbload(loadfunc=PDB function(task) ,
                  xlspath=path to spreadsheets,
                  pdbpath=path to project database,
                  pgmpath=path to customized functions,
                  loadmac=list of customized functions to be loaded,
                  domlst =list of domains to be mapped and loaded into
                          project database,
           sasopts=valid SAS option(s))
```

PDL is very modular in its design, making it adaptable to any structure of data for any project.

THE STEPS

There are 4 steps to building a pooled dataset using this tool beginning with identifying what studies are to be included in the pool to the eventual loading of the data.

Step 1 – Creating the study library spreadsheet (studylib.xlsx)

This step involves the creation of a spreadsheet that contains information regarding the studies to be pooled and the location of the data for those studies.

studylib.xlsx

	A	B	C	D	E
1	studyid	study	index	studylib	load
2	Study1	First Study		X:\study1\ad\data	
3	Study2	Second Study		X:\study2\ad\data	
4	Study3	Third Study		X:\study3\ad\data	
5	Study4	Fourth Study		X:\study4\ad\data	

Column A (studyid): contains the list of studies to be pooled

Column B (study): contains the text which describes the study

Column C (index): is used if data to be pooled for a study is coming for 2 or more folders

Column D (studylib): contains the path to where the data for each individual study is stored

Column E (load): used to indicate which study is to be loaded into the pooled database. An ‘x’ tells the tool to load that study.

Example of data coming from 2 different folders:

studylib.xlsx

	A	B	C	D	E
1	studyid	study	index	studylib	load
2	Study1	First Study		X:\study1\ad\data	
3	Study2	Second Study	sd	X:\study2\sd\data	
4	Study2	Second Study	ad	X:\study2\ad\data	
5	Study3	Third Study		X:\study3\ad\data	
6	Study4	Fourth Study		X:\study4\ad\data	

In this example, Study2 will be pooled from 2 different folders for at least one of the datasets to be created. Index id's of "sd" and "ad" are defined but any values can be placed here.

There are basically 2 ways in which the STUDYLIB spreadsheet can be created:

Manual – From a blank spreadsheet, the columns described above are added and then one by one, each study ID and path to the data is entered.

Automated – The spreadsheet can be built by PDL using the function STUDYLIB. In the LOADFUNC parameter, the key word STUDYLIB is used. This instructs PDL to create the STUDYLIB spreadsheet. Along with this are 2 tasks. The first is SEARCH. With this task the path is defined telling PDL where to begin the search for folders containing SAS datasets. The other is FILTER which provides instruction on how to subset the list of possible folders containing SAS datasets.

Illustration of how the automation works to produce STUDYLIB spreadsheet

Example using keyword **STUDYLIB**:

```
%pdbload(loadfunc=studylib)
```

Example using keyword **SEARCH**:

```
%pdbload(loadfunc=studylib*search=C:\level1\level2\level3\level4\product_id
```

In this example you see the keyword SEARCH. This is instructing PDL to begin the search for folders at the product_id level. PDL will search every folder under product_id looking for folders containing SAS datasets.

Examples using keyword **FILTER**

With **DATES**

```
%pdbload(loadfunc=studylib*search=path*filter=dates=5)
```

	A	B	C	D	E
1	LOAD	STUDYID	STUDYLIB	DATE	BLINDED
2		Study1	M:\level1\level2\level3\level4\level5\level6\level7\dryrun\data\adam	03-Feb-19	
3		Study1	M:\level1\level2\level3\level4\level5\level6\level7\dryrun\data\adam\adam_define	05-Feb-19	
4		Study1	M:\level1\level2\level3\level4\level5\level6\spi\ADAM data\working	26-Jun-19	
5		Study1	M:\level1\level2\level3\level4\level5\level6\level7\final\data\adam	28-Feb-19	
6		Study1	M:\level1\level2\level3\level4\level5\level6\level7\a2\data\adam\2018-12-11-fa	30-Jan-19	

This is a list of the folders containing 5 most recent ADaM SAS datasets.

With **STUDIES**

```
%pdbload(loadfunc=studylib*search=path*filter=studies=Study1 Study2,
```

	A	B	C	D	E
1	LOAD	STUDYID	STUDYLIB	DATE	BLINDED
2		Study1	M:\level1\level2\level3\level4\level5level6\level7\analysis1\data\adam	05-Aug-19	
3		Study1	M:\level1\level2\level3\level4\level5level6\level7\analysis1\data\adam\2019-08-02-ssd-2019q3	05-Aug-19	
4		Study2	M:\level1\level2\level3\level4\level5level6\level7\analysis1\data\adam	05-Aug-19	
5		Study2	M:\level1\level2\level3\level4\level5level6\level7\analysis1\data\adam\2019-07-23-ssd	05-Aug-19	

This is a list of the folders containing the 2 most recent ADaM SAS datasets for 2 different studies.

Step 2 – Creating and maintaining the domain mapping spreadsheet (domainmapv1.xlsx)

The first of 2 spreadsheets are created in this step. This is created by running PDL as follows:

```
%pdbload(loadfunc=domainmap ,
              .
              .
)
```

The DOMAINMAP spreadsheet will contain a list of all studies from the STUDYLIB spreadsheet along with a list of all datasets that exist for each study. Appended to the name of the spreadsheet is the suffix “vn” indicating the version of the spreadsheet. Every time DOMAINMAP is recreated using PDL, the “n” is incremented by 1.

domain_mapv1.xlsx

	A	B	C	D	E
1	PRJDOMAIN	STDYStudy1	STDYStudy2	STDYStudy3	STDYStudy4
2		ADAE	AE_Ssd	ADAE	ADAE
3		ADAUEC	CE_Ssd	ADBILAG	ADBILAG
4		ADCM	CM_Ssd	ADCM	ADCM
5		ADCO	ADAE_Sad	ADDV	ADCSSRS
6		ADCP	ADAUEC_Sad	ADEG	ADDV

If data comes from 2 or more folders for a study, PDL attaches ‘_S’ to the end of the dataset name along with the index ID. For example, ADAE for Study2 is defined as ADAE_Sad. This ensures that the name of the dataset applies to the correct library as defined in the STUDYLIB spreadsheet.

From this list of studies and datasets, a decision is made regarding which of the datasets will be mapped into the pooled database. These decisions are done on a domain by domain basis.

domain_mapv1.xlsx

	A	B	C	D	E
1	PRJDOMAIN	STDYStudy1	STDYStudy2	STDYStudy3	STDYStudy4
2	ADSL	ADSL	ADSL_Sad DM_Ssd	ADSL	ADSL
3	ADLB	ADLB	ADLB_Sad	ADLB	ADLBCHEM ADLBHEMA ADLBOTH ADLBURIN
4		ADAE	AE_Ssd	ADAE	ADAE

This example demonstrates how anywhere from 1 to multiple study level datasets can be defined for a single pooled domain. The pooled ADSL domain has 2 datasets defined for Study2 while ADLB has 4 datasets defined for Study4.

Step 3 – Creating and maintaining the pooled database code list spreadsheet (pdb_codelist)

The pooled database code spreadsheet contains a list of format names, codes, and decodes that can be applied across all studies or to individual studies depending on how the mapping rules are defined.

pdb_codelist.xlsx

	A	B	C
1	format name	value	label
2	sexnf	1	Male
3		2	Female
4		3	Missing
5			
6	$sexs	Male	M
7		Female	F

pooled | Study1 | Study2 | Study3 | Study4

In this example, the tab labeled “pooled” contains a list of formats that can be applied across all studies. Prior to loading the study level data into the pooled database, PDL is run with LOADFUNC=fmtload.

```
%pdbload(loadfunc=fmtload ,
              .
              .
)
```

This will load the formats defined in this tab into a SAS® dataset. Upon loading of the data into the pool, the format SAS® dataset is converted into a FORMAT catalog where it becomes available for formatting as needed during the loading process.

pdb_codelist.xlsx

	A	B	C	D	E
1	recode fmt	old value	old label	new value	new label
2	trt01af	1	Dose 1	2	Low Dose
3		2	Placebo	1	Placebo
4		9	Screen Failure	0	Screen Failure
5					
6	$racef	WHITE		5	
7		OTHER/MIXED		6	

pooled | Study1 | Study2 | Study3 | Study4

In this example, the tab labeled “Study1” contains a list of formats that can be applied to a specific study (in this case, Study1). Columns A, B, and D contain the information used by PDBLoader to produce a FORMAT catalog. This gets appended to the “pool” formats as each study is loaded into the pool. Columns C and E are for documentation purposes only. These columns serve as a reminder as to what the original labels to the codes were (Column C) and what they have been converted too (Column E).

Step 4 – Creating and maintaining the variable mapping spreadsheet DOMAINdomain_var_mapv1)

This is the step where the work of mapping begins. First, PDL is run with LOADFUNC=varmap.

```
%pdbload(loadfunc=varmap ,
                  domlst=name of dataset to be created,
            .
)
```

Running this function produces the DOMAINdomain_var_mapv1.xlsx spreadsheet. The DOMAIN refers to the name of the pooled dataset that is to be created. In addition to LOADFUNC, DOMLST must be defined, letting PDL know what dataset is about to be created .

ADSLdomain_var_mapv1.xlsx

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	PRJDOMAIN	PRJVAR	PRJTYP	PRJLEN	PRJLBL	PRJFMT	PRJKEY	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
2	ADSL							ADSL	ACTARM	char	200	Description of Actual Arm	
3	ADSL							ADSL	ACTARMCD	char	20	Actual Arm Code	
4	ADSL							ADSL	AGE	num	8	Age	BEST12.
5	ADSL							ADSL	AGECAT	char	200	Age Category	
6	ADSL							ADSL	AGEU	char	6	Age Units	
7	ADSL							ADSL	AGE_CALC	num	8	Age calculated	BEST12.
8	ADSL							ADSL	AGGDRC	char	200	Gender,Age,Race,Weight(kg) combined	
9	ADSL							ADSL	AGGRC	char	200	Gender,Age,Race combined	
10	ADSL							ADSL	ARM	char	200	Description of Planned Arm	

This spreadsheet can be broken down into 2 sections:

Section 1: The template

ADSLdomain_var_mapv1.xlsx

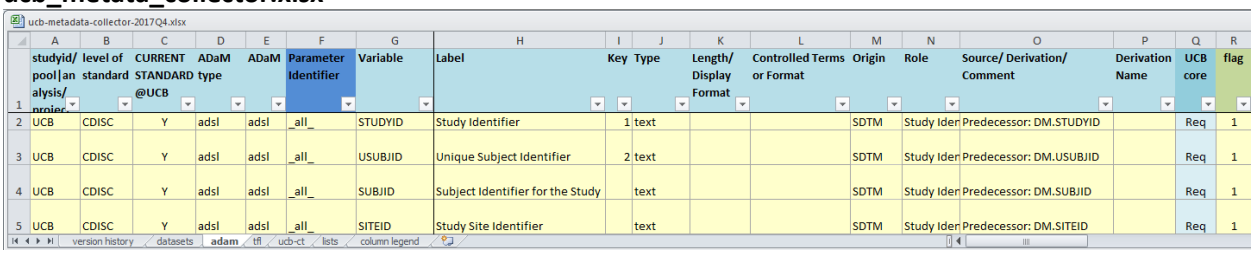
	A	B	C	D	E	F	G
1	PRJDOMAIN	PRJVAR	PRJTYP	PRJLEN	PRJLBL	PRJFMT	PRJKEY
2	ADSL						
3	ADSL						
4	ADSL						
5	ADSL						
6	ADSL						
7	ADSL						
8	ADSL						
9	ADSL						
10	ADSL						

This is where the “target” dataset is to be defined including the variables and their attributes (Columns B through F). Column G (PRJKEY) defines how the “target” dataset is to be sorted. There are 3 ways in which the template can be populated.

Option 1: Manually enter the variable names and their attributes.

Option 2: At UCB there is a workbook called the UCB metadata collector. This contains the list of the possible variables for many domains as defined by the CDISC ADaM Implementation guide. PDL reads this information in using the DDT parameter.

ucb_metata_collector.xlsx



ucb-metadata-collector-2017Q4.xlsx

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	studyid/ pool|analysis/ projec	level of standard	CURRENT STANDARD @UCB	ADaM type	ADaM	Parameter Identifier	Variable	Label	Key	Type	Length/ Display Format	Controlled Terms or Format	Origin	Role	Source/ Derivation/ Comment	Derivation Name	UCB core	flag
2	UCB	CDISC	Y	adsl	adsl	_all_	STUDYID	Study Identifier	1	text			SDTM	Study Iden	Predecessor: DM.STUDYID		Req	1
3	UCB	CDISC	Y	adsl	adsl	_all_	USUBJID	Unique Subject Identifier	2	text			SDTM	Study Iden	Predecessor: DM.USUBJID		Req	1
4	UCB	CDISC	Y	adsl	adsl	_all_	SUBJID	Subject Identifier for the Study		text			SDTM	Study Iden	Predecessor: DM.SUBJID		Req	1
5	UCB	CDISC	Y	adsl	adsl	_all_	SITEID	Study Site Identifier		text			SDTM	Study Iden	Predecessor: DM.SITEID		Req	1

version history | datasets | adam | tfl | ucb-ct | lists | column legend

The “flag” column enables the project programmers to select the variables from the list that applies to a specific study or pool of studies. PDL using the flag to obtain the list of variables required for the pooling.

Here is an example of running PDL with the DDT parameter defined to pull in the information from the metadata collector.

```
%pdbload(loadfunc=varmap ,
         domlst=adsl,
          ddt=x:\pool\spreadsheet\ucb_metadata_collector.xlsx,
                    .
        )
```

The result of running PDBLoader with DDT pointing to the metadata collector is seen below.

ADSLdomain_var_mapv1.xlsx

	A	B	C	D	E	F	G
1	PRJDOM	PRJVAR	PRJTYP	PRJLEN	PRJLBL	PRJFMT	PRJKEY
2	ADSL	STUDYID	char	200	Study Identifier		1
3	ADSL	USUBJID	char	200	Unique Subject Identifier		2
4	ADSL	SUBJID	char	200	Subject Identifier for the Study		
5	ADSL	SITEID	char	200	Study Site Identifier		
6	ADSL	REGIONy	char	200	Geographic Region y		
7	ADSL	REGIONyN	num	8	Geographic Region y (N)		

With the metadata collector, there is no variable length provided, so PDL populates PRJLEN with 200 if the variable is character and 8 if the variable is numeric. These can be changed to best fit the data to be pooled.

Option 3: If a dataset does not exist in the metatdata collector then it is possible to get the list of variables and their attributes from an existing dataset that would be considered close in structure to what is needed for the pool. This is done using the DDT parameter in the following way:

```
%pdbload(loadfunc=varmap ,
                 domlst=adsl,
                    ddt= library=x:\study1\data\adam * file= adsl
        )
```

The result of populating the template using and existing dataset is as follows:

ADSLdomain_var_mapv1.xlsx

	A	B	C	D	E	F	G
1	PRJDOMAIN	PRJVAR	PRJTYP	PRJLEN	PRJLBL	PRJFMT	PRJKEY
2	ADSL	STUDYID	char	10	Study Identifier		
3	ADSL	USUBJID	char	20	Unique Subject Identifier		
4	ADSL	SUBJID	char	5	Subject Identifier for the Study		
5	ADSL	SITEID	char	5	Study Site Identifier		
6	ADSL	SITESUBJ	char	11	Site-Subj Number		
7	ADSL	RCSSUBJ	char	40	Region/Country/Site-Subject Number		
8	ADSL	BLGARW	char	100	Gender/Age(years)/Race/Weight(kg)		
9	ADSL	INVNAM	char	50	Investigator Name		
10	ADSL	RFICDT	num	8	Informed Consent Date	E8601DA.	

In this example, all of the variables and their attributes (including labels, lengths and formats) from the existing dataset appear in the template.

Section 2: The study level mapping

The study level mapping section contains 6 columns with each column header containing a 5 letter prefix followed by the study ID.

ADSLdomain_var_mapv1.xlsx

	B	G	H	I	J	K	L	M
1	PRJVAR	PRJKEY	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1

The 5 letter prefixes are defined as follows:

STDTA – Name of the study level dataset.

STVAR – List of variables in the study level dataset

STTYP – Variable type (i.e. char,num)

STLEN – Length of each variable

STLBL – Label for each variable

STFMT – Formats for each variable if applied

Normally, the template is built at the same time as at least one study is loaded into the VARMAP spreadsheet. When this is done, PDL does 2 things:

1. If a study level variable is of the same name as the template variable, PDL will automatically populate the template level rows under that study.

ADSLdomain_var_mapv1.xlsx

	B	G	H	I	J	K	L	M
1	PRJVAR	PRJKEY	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
2	STUDYID	1	ADSL	STUDYID	char	10	Study Identifier	
3	USUBJID	2	ADSL	USUBJID	char	20	Unique Subject Identifier	
4	SUBJID		ADSL	SUBJID	char	10	Subject Identifier for the Study	
5	SITEID		ADSL	SITEID	char	5	Study Site Identifier	
6	REGIONy							

2. The study level variable list now appears after the list of variables has been defined for the template. The list of study level variables will appear in alphabetical order.

ADSLdomain_var_mapv1.xlsx

	A	B	G	H	I	J	K	L	M
1	PRJDOMA	PRJVAR	PRJKEY	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
59	ADSL	PROTDATE							
60	ADSL	PROTVERS							
61	ADSL			ADSL	ACTARM	char	200	Description of Actual Arm	
62	ADSL			ADSL	ACTARMCD	char	20	Actual Arm Code	
63	ADSL			ADSL	AGE	num	8	Age	BEST12.
64	ADSL			ADSL	AGECAT	char	200	Age Category	

For those variables that do not match up with the template variables in terms of the variable names, they can be copied.

ADSLdomain_var_mapv1.xlsx

	A	B	H	I	J	K	L	M
1	PRJDOMAIN	PRJVAR	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
71	ADSL		ADSL	BMI_CALC	num	8	BMI (kg/m²) calculated at Screening	BEST12.

The copied variable is then pasted in the row that applies to the variable of the template:

ADSLdomain_var_mapv1.xlsx

	A	B	H	I	J	K	L	M
1	PRJDOMAIN	PRJVAR	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
27	ADSL	BMI	ADSL	BMI_CALC	num	8	BMI (kg/m²) calculated at Screening	

This process is done for every variable where there is a one to one match between the study and template variable. For those variables that require additional processing, there are 2 levels from which modification to variables or derivations can be done.

Level 1 – Variable level:

Creating values using formats from the PDB_codelist spreadsheet and the RECODE function.

ADSLdomain_var_mapv1.xlsx

	A	B	H	I	J	K	L	M
1	PRJDOMAIN	PRJVAR	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
8	ADSL	AGE	ADSL	AGE	num	8	Age	BEST12.
9	ADSL	AGEU	ADSL	AGEU	char	6	Age Units	
10	ADSL	AGEGR1	ADSL					
11	ADSL	AGEGR1N	ADSL	*recode(agegr1n,age,agegr1nf.)	num			

Creating values from another variable or from scratch using the DERIVE function.

ADSLdomain_var_mapv1.xlsx

	A	B	H	I	J	K	L	M
1	PRJDOMAIN	PRJVAR	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
2	ADSL	STUDYID	ADSL	STUDYID	char	10	Study Identifier	
3	ADSL	USUBJID	ADSL	USUBJID	char	20	Unique Subject Identifier	
4	ADSL	SUBJID	ADSL	SUBJID	char	10	Subject Identifier for the Study	
5	ADSL	SITEID	ADSL	SITEID	char	5	Study Site Identifier	
6	ADSL	TEST	ADSL	*derive(test,'This is a cool tool.')	char			
7	ADSL	ACOUNTRY						
8	ADSL	AGE	ADSL	AGE	num	8	Age	BEST12.
9	ADSL	AGEU	ADSL	AGEU	char	6	Age Units	

Level 2 – Data step level:

For the data step level of processing there are 2 options available.

Option 1: Defining a function in the spreadsheet

ADSLdomain_var_mapv1.xlsx

	A	B	H	I	J	K	L	M
1	PRJDOMAIN	PRJVAR	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
6	ADSL	AGE	ADSL	AGE	num	8	Age	BEST12.
7	ADSL	AGEU	ADSL	AGEU	char	6	Age Units	
8	FUNCTION		data fun; set adsl; test='This is a cool tool'; run;					

Option 2: Defining a function as a macro

ADSLdomain_var_mapv1.xlsx

	A	B	H	I	J	K	L	M
1	PRJDOMAIN	PRJVAR	STDTAStudy1	STVARStudy1	STTYPStudy1	STLENStudy1	STLBLStudy1	STFMTStudy1
5	ADSL	SITEID	ADSL	SITEID	char	5	Study Site Identifier	
6	ADSL	TEST	ADSL	*derive(test,'This is a cool tool.')	char			
7	ADSL	ACOUNTRY						
8	ADSL	AGE	ADSL	AGE	num	8	Age	BEST12.
9	ADSL	AGEU	ADSL	AGEU	char	6	Age Units	
10	FUNCTION		%fun_tool					

Step 5 – Loading the mapped data into the pooled dataset

The final step in the loading of study data into the pooled database is performed by defining LOADFUNC as DTALOAD.

```
%pdbload(loadfunc=dtaload ,
     domlst=adsl,
     pgmpath=path to where the custom pooling macros are stored,
     loadmac=list of programs to be included in the pooling process
                              )
```

The PGMPATH tells PDL where to look for the modules that are to be included in the pooling process and LOADMAC tells PDL what program are to be included.

THE CODE

Loading study level data into the WORK folder

When the DTALOAD function is run, PDL will process the information in VARMAP one study at a time. With each study a unique library reference is defined and the study level data is loaded into the WORK library. From here there is no need to worry about attaching a library references to the datasets if additional processing is needed.

```
MPRINT(DTALOAD):   libname stdy1_1 "M:\Sites\Braine\Biocdm\Products\CDP7657\SL0013A\gsp\analysis\prod\ad\data";
NOTE: Libref STDY1_1 was successfully assigned as follows:
      Engine:        V9
      Physical Name: M:\Sites\Braine\Biocdm\Products\CDP7657\SL0013A\gsp\analysis\prod\ad\data
MPRINT(DTALOAD):   options fmtsearch=(stdy1_1);
MPRINT(DTALOAD):   ****************************************************************;
MPRINT(DTALOAD):   * Step 5.3: load study datasets into WORK;
MPRINT(DTALOAD):   ****************************************************************;
MPRINT(DTALOAD):   data ADSL;
MPRINT(DTALOAD):   set stdy1_1.ADSL (keep=ACTARM ACTARMCD AGE AGECAT AGEU AGE_CALC AGGDRC AGGRC ARM ARMCD BMI_CALC
BRTHDTC BSA_CALC COUNTRY DSTRTCD DSTRTVB DTHDTC DTHFL ENRLFL ETHNIC FCDTC GEARWT ITTFL LSMEDYC PDPPSFL PKPPSFL RACE
RANDDT RANDDTC RANDNO RFENDTC RFICDTC RFPENDTC RFSTDTC RFXENDTC RFXSTDTC RSNCDBRK SAFFL SCRBMI SCRBSA SCRFL SCRHGT
SCRNDTC SCRWGT SEX SITEID STDTC STUDYID SUBJECT SUBJID TERMSPEC TRT01A TRT01AN TRT01P TRT01PN TRTEDT TRTEDTM TRTETM
TRTSDT TRTSDTM TRTSTM USUBJID WEIGHT);
MPRINT(DTALOAD):   run;

NOTE: There were 106 observations read from the data set STDY1_1.ADSL.
NOTE: The data set WORK.ADSL has 106 observations and 63 variables.
```

Here you see a unique study library reference is defined and the data copied to the WORK library.

Running a DATASTEP level function

As mentioned earlier, PDL will execute custom designed programs written to perform various tasks. This can be done in 2 ways.

1. Executing the code that is written directly into the spreadsheet

When running DTALOAD, the code in the VARMAP spreadsheet will be run before the “final” dataset is created.

```
MPRINT(DTALOAD):   ****************************************************************;
MPRINT(DTALOAD):   * Step 5.4: run datastep functions;
MPRINT(DTALOAD):   ****************************************************************;
MPRINT(DTALOAD):   data fun;
MPRINT(DTALOAD):   set adsl;
MPRINT(DTALOAD):   test='This is a cool tool';
MPRINT(DTALOAD):   run;

NOTE: There were 106 observations read from the data set WORK.ADSL.
NOTE: The data set WORK.FUN has 106 observations and 64 variables.
```

2. Executing the code that is written as a macro.

In this example the name of the macro appears in the .log along with the code.

```
MPRINT(DTALOAD):   ******************************************************************;
MPRINT(DTALOAD):   * Step 5.4: run datastep functions;
MPRINT(DTALOAD):   ******************************************************************;
MPRINT(FUN_TOOL):   data fun;
MPRINT(FUN_TOOL):   set adsl;
MPRINT(FUN_TOOL):   test='This is a cool tool';
MPRINT(FUN_TOOL):   run;

NOTE: There were 106 observations read from the data set WORK.ADSL.
NOTE: The data set WORK.FUN has 106 observations and 64 variables.
```

Running a variable level function

The last step in the load process is the "copying" of the study level data into the pooled dataset. It is here that the dataset is constructed using the template information and any variable level functions are executed. This is done one study at a time. The dataset name is STUDYn where n is the number in a sequence from 1 to the last number of the studies to be pooled. So, if there are 4 studies to be loaded into the pooled dataset, n would be 4.

```
MPRINT(DTALOAD):  ;
MPRINT(DTALOAD):   ******************************************************************;
MPRINT(DTALOAD):   * Step 5.6: produce domain dataset;
MPRINT(DTALOAD):   ******************************************************************;
MPRINT(DTALOAD):   data stdy1(keep= STUDYID USUBJID SUBJID SITEID TEST ACOUNTRY AGE AGEU );
MPRINT(DTALOAD):   attrib STUDYID length=$200 label="Study Identifier" USUBJID length=$200 label="Unique Subject
Identifier" SUBJID length=$200 label="Subject Identifier for the Study" SITEID length=$200 label="Study Site
Identifier" TEST length=$200 label="This is just a test" ACOUNTRY length=$200 label="Analysis Country" AGE length=8
label="Age" AGEU length=$200 label="Age Units" ;
MPRINT(DTALOAD):   set ADSL(rename=( STUDYID=tSTUDYID USUBJID=tUSUBJID SUBJID=tSUBJID SITEID=tSITEID AGE=tAGE
AGEU=tAGEU ));
MPRINT(CONVAR):   STUDYID=trim(left(tSTUDYID));
MPRINT(DTALOAD):  ;
MPRINT(CONVAR):   USUBJID=trim(left(tUSUBJID));
MPRINT(DTALOAD):  ;
MPRINT(CONVAR):   SUBJID=trim(left(tSUBJID));
MPRINT(DTALOAD):  ;
MPRINT(CONVAR):   SITEID=trim(left(tSITEID));
MPRINT(DTALOAD):  ;
MPRINT(DERIVE):   test='This is a cool tool.';
MPRINT(DTALOAD):  ;
MPRINT(DTALOAD):   ACOUNTRY='';
MPRINT(CONVAR):   AGE=tAGE;
MPRINT(DTALOAD):  ;
MPRINT(CONVAR):   AGEU=trim(left(tAGEU));
MPRINT(DTALOAD):  ;
MPRINT(DTALOAD):   run;

NOTE: There were 106 observations read from the data set WORK.ADSL.
NOTE: The data set WORK.STDY1 has 106 observations and 8 variables.
```

Once the mapping of all of the studies to be loaded is completed (remapped in terms of structure and content) the STUDYn datasets are set together and the job is finished.

```
-------------------------------------------------------------------------------
|                  LOAD ADSL DATA INTO PROJECT DATABASE                       |
-------------------------------------------------------------------------------

MPRINT(DTALOAD):   data ADSL;
MPRINT(DTALOAD):   set stdy1 stdy2 stdy3 stdy4 ;
MPRINT(DTALOAD):   run;

NOTE: There were 106 observations read from the data set WORK.STDY1.
NOTE: There were 49 observations read from the data set WORK.STDY2.
NOTE: There were 68 observations read from the data set WORK.STDY3.
NOTE: There were 399 observations read from the data set WORK.STDY4.
NOTE: The data set WORK.ADSL has 622 observations and 59 variables.
```

CONCLUSION

The combining of data from 2 or more studies into a single database can never be totally automated. However, it is possible to combine the documentation and writing of the code into a more seamless process. The combining of the documentation in MS Excel and SAS® code is what makes this possible.

PDL begins the documentation process in the STUDYLIB.xlsx file by identifying what studies are to be included in the pool and where the data for each study is located. This tool then drills down to the next level of information in the DOMAINMAP.xlsx file. It is here that it is documented what study level datasets (DOMAINs) will be used in the pool for each pooled level DOMAIN. Finally, the third level of documentation is the DOMAIN specific spread known as the VARMAP spreadsheet (DOMAINdomain_var_map) where the mapping of the study level variables into the pooled level variables is stored. One additional spreadsheet is called PDB_CODELIST.xlsx which contains all of the codes and decodes used in the resetting of many of the coded variables. In these 4 spreadsheets all documentation needed to follow the path from original study data to final pooled data is contained.

In addition to documentation, the spreadsheets provide the information needed to build the pooled datasets. From the STUDLIB spreadsheet the library references are defined and information is provided (from the LOAD column) regarding which of the studies are to be loaded at any given time. From the DOMAINMAP spreadsheet the PDL can determine what datasets will be used from each study to build the VARMAP spreadsheet. Once all of the mapping rules are defined in the VARMAP spreadsheet and the recoding list is defined in the PDB_CODELIST spreadsheet, the PDL converts that information into SAS® code that can be executed to produce the pooled datasets.

The combination of the MS Excel interface and the SAS® code, otherwise known as PDL provides the complete package to document and produce a pooled dataset of any type and structure from any study data type and structure. If at any time in the future, there is a need to go back to what was done to build a pooled database; one only has to go back to the spreadsheets.

CONTACT INFORMATION

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

David Hartman
UCB Biosciences
P.O. Box 110167
Research Triangle Park, NC 27709 USA
Work Phone: +1 919-767-2597
Fax: +1 919-767-2573
Email: david.hartman@ucb.com

Pranjali Dafe
Covance, Inc.
400 Centre Green Way, Cary, NC 27513
Email: pranjali.dafe@covance.com