Paper CS01

%Supp2Par_v1
Merging Supplemental Data onto Parent Domains

Dirk Van Krunckelsven, Merck NV/SA, Antwerp, Belgium

ABSTRACT

In May 2013 the author attended PhUSE/CSS in Silver Springs, Washington and took part in the Standard Scripts Working Group (WG5). In the spirit of collaboration our team decided to contribute to the standard scripts repository. We provided one of our standard SAS macros: %supp2par_v1, a macro that very simply merges supplemental data onto parent data. The core processing is very simple, yet the complete macro has over 1000 lines of code! This paper explains the macro's functioning as well as some of the design principles that have been employed. The experiences with the standard scripts repository will be briefly described.

TOWARDS SDTM ACCEPTANCE

Supplemental qualifiers have existed since early on in CDISC's SDTM. The principle basically provides a way to store non-standard variables in SDTM along with their association to parent records in general-observation-class datasets (Events, Findings and Interventions) and Demographics. At our company, when introducing SDTM, we had quite some difficulty selling this to people. Specifically a lot of push-back was received on the SUPPQUAL concept.

People were used to long and wide datasets, they did not understand why an additional dataset would be needed. Or worse: an additional dataset per data domain. They wondered why we wanted them to split things up only to put them back together for analysis. Even though they agreed that the remerge as such is relatively simple, they did claim that a lot of additional steps were now required. And yes, people also agreed that in principle standard data makes life easier further down the road with the possibility to employ standard programs. However, not having experienced this for real, they had a hard time truly believing it.

Therefore we started supporting standard data's promise of ease-through-standard-programs. One of the initiatives to support the delivery of SDTM-compliant data was the %supp2par_v1 macro (its first version was written in 2010). This macro easily merges supplemental and parent data back together. The author believes that the code was instrumental in obtaining acceptance for the concept of supplemental qualifiers specifically and CDISC's SDTM in general.

The actual code and its user documentation can be downloaded through the link [1] provided in the references section. The remainder of this paper will not just repeat the user documentation. The core processing and some of the principles applied in the program will be detailed. And the sharing of the code through the repository created by the Phuse/CSS Working Group on Standard Scripts will be addressed briefly.

%SUPP2PAR_V1: THE MACRO CALL

Calling the Supp2Par_v1 macro in its simplest form is straightforward since only 3 macro parameters are required. The signature of the macro:

```
%SUPP2PAR_v1(
   inlib=,   /* Name of input library for Parent and Supplemental Datasets */
   parent=,  /* Name of Parent dataset */
   supp=,    /* Name of Supplemental dataset */
   outlib=,  /* Name of destination library for Merged dataset */
   outname=, /* Name of Merged dataset */
   clean=,   /* Whether or not (Y/N) to clean work environment at end */
   dev=,     /* Whether or not (Y/N) to show all messages in the log */
   Info=,    /* Level of information to provide in the log:
                    1 (all), 2 (start and summary), 3 (only summary) */
   RC=       /* Name of Return Code Macro variable */);
```

REQUIRED MACRO PARAMETERS

As stated, only the three macro parameters that point the code to the input data are required. These parameters are INLIB, PARENT and SUPP. INLIB requires the library where the data is present, PARENT needs the name of the parent dataset and SUPP needs that of the supplemental dataset. The macro thus assumes that the supplemental and parent datasets are both located in the same library.

OPTIONAL MACRO PARAMETERS

The parameters that describe where and how to write the output are optional and will have defaults. These are OUTLIB, requiring the library where the resulting re-merged dataset is to be written, and OUTNAME which needs the name of that re-merged dataset. By default the data will be written in WORK with a name that adds the suffix FULL to the name provided in the required macro parameter PARENT.
Similarly the Control and Log parameters are also optional with default behavior. These provide the user with control over how the log should look like, how the return code is to be named and whether the environment is to be cleaned by the macro. Further details are to be found in the user documentation that is available for download.

%SUPP2PAR_V1: CORE PROCESSING

In essence the code for merging the supplemental qualifiers onto the parent domain is rather simple: one needs to transpose the supplemental data to then merge these back onto the parent domain using the specified keys.

TRANSPOSITION

In SAS the first step, transposition, is performed using the transpose procedure as shown below. QNAM and QLABEL contain information for the resulting variable name and label and are thus used in respectively the ID (❶) and IDLABEL (❷) statements. The actual variable to be transposed is QVAL which therefore goes onto the VAR statement (❸).

```
proc transpose data = S2P_SUPP[NAME] (rename = (RDOMAIN = DOMAIN))
               out  = S2P_SUPP[NAME]T (drop = _NAME_ _LABEL_);
  by STUDYID DOMAIN USUBJID [IDVAR(VAL)]; ❹
  var qval; ❸
  id qnam; ❶
  idlabel qlabel; ❷
run;
```

We also enter a BY statement in the transpose procedure (requiring the input data to be sorted). These variables will not be transposed. These are actually the keys by which we will later on merge the transposed dataset onto the parent domain. As shown above (❹), the by-statement leads us to a first complication: full specification of the keys: What is `[IDVAR(VAL)]` in the above? In the actual code it is a macro variable with counter obtained from the information contained in IDVAR on the supplemental dataset.

KEY DETERMINATION AND RE-SPECIFICATION

In a supplemental dataset one is not restricted to a single identifying variable (IDVAR). Multiple IDVARs can be present in the supplemental dataset and each of these dictates a transposition (and a merge). Therefore in the %supp2par_v1 macro every IDVAR is first identified and the transposition (and merge) is placed in a loop over the IDVARs encountered.

The actual merge is performed on IDVARVAL which needs renaming towards the variable name on the parent domain as provided by IDVAR. A complication here is that IDVARVAL is always character while the actual variable on the parent domain may not be. In addition to renaming IDVARVAL may require transformation to achieve a clean remerge. In addition, an empty IDVAR-IDVARVAL combination indicates that the link between the supplemental record and the parent can be achieved by just STUDYID, (R)DOMAIN, USUBJID.

ROBUSTNESS: IDVAR NOT PRESENT

Possibly the IDVAR indicated in the supplemental dataset is not present in the parent domain which will result in an error. %supp2par_v1 is used extensively in our company, therefore we want to explicitly capture this event, terminate the macro and tell the user that this occurs. This saves a lot of time in possible investigations by the user(s).

MERGING KNOWINGLY

Having transposed the supplemental dataset and knowing the keys by which to merge (❶), the actual merge is easily applied using the same by-statement as above on a data step. However, we want our users to know what happens in this merge and therefore we create a dataset *_DROP that contains all the supplemental records that did not find a parent record. This is easily done using if-then-output (❷ and ❸) processing based on the option in= (❹ and ❺) .

```
data S2P_[NAME] S2P_[NAME]_DROP;
  merge S2P_PAR[NAME] (in = PARENT) ❹
        S2P_SUPP[NAME]T     (in = SUPP); ❺
  by STUDYID DOMAIN USUBJID [IDVAR(VAL)]; ❶
  if PARENT then output S2P_[NAME]; ❷
  else if SUPP and not(PARENT) then output S2P_[NAME]_DROP; ❸
run;
```

%SUPP2PAR_V1: ROBUST AND INFORMATIVE

In our group we attempt to write robust macros [2] that provide the users and the programs within which they operate with information regarding the macro's processing using incremental return codes [3]. In addition to the check on the existence of IDVAR and the dropped records indicated above, many more conditions are being checked during the macro's processing. This is the reason why the code is over 1000 lines long rather than a more realistic number as can be expected from the above description.

INPUT AND PARAMETER CHECKS AND CORRECTIONS

The input provided to the macro is checked. Whenever possible the code strives for continuation rather than termination. I.e. we consider it more useful to create output that is valid rather than to stop the processing. The user will be notified whenever the following non-termination conditions are met:

- If the destination output library does not exist, the code defaults to the work library.
- If the supplemental does not exist or if it is empty, the output will be a copy of the parent.
- If an illegal output dataset name was provided a default value will be used.
- When IDVAR and IDVARVAL are empty the USUBJID is assumed sufficient for the merge.
- When the value in QNAM already exists as a variable on the parent or if it results in an invalid variable name, then an alternative variable name is used.
- When records in the supplemental find no parent record, these are written to a dataset for verification.
- When records on the supplemental had to be ignored because of the value for STUDYID, USUBJID, RDOMAIN and/or QNAM being empty, these are written to a dataset for verification.

Some conditions would lead to nonsensical results and/or simply make the processing impossible. These lead to termination and the user is notified:

- The input library or the parent dataset does not exist or is empty.
- Required variables on the parent (including those dictated by IDVAR) or the supplemental are not present.
- The same QNAM shows up with different labels.
- IDVAR is empty while IDVARVAL is not.
- Non-uniqueness of the supplemental dataset on USUBJID RDOMAIN IDVAR IDVARVAL QNAM. Although this leads to termination, the duplicate records are written to a dataset for verification to ease remedying this condition.
- Data step and Procedure errors and unexpected errors.

%SUPP2PAR_V1: SOME SHORTCOMINGS

Although %supp2par_v1 has proven very valuable within our company and even though it offers a complete solution to automate the remerging of supplemental data onto its parent domain, we do recognize that there are some shortcomings.

MERGES 1-BY-1

The %supp2par_v1 macro merges supplemental datasets one-by-one, i.e. one needs to call the macro repeatedly for all datasets that need remerging. This can easily be overcome by including the macro within a calling program that loops over all datasets in a directory. This is exactly what is being done by our data managers. A good calling program would employ the return-codes created by the macro to determine whether each call succeeded before continuing.

VALUE LEVEL METADATA

The transposed data created by the %supp2par_v1 macro is merged onto the parent domain exactly as it was found on the supplemental dataset in the QVAL variable. It will be character whereas truly the variable may not truly be character. The information for this can be found in the metadata, more precisely in the value level metadata. It should be possible for a routine such as %supp2par_v1 to refer to this metadata and apply it in one go. We have however not succeeded in doing this because we found no agreement on both the form of and the actual delivery of the metadata alongside the SDTM data.

DISAPPEARANCE OF QORIG AND QEVAL

When the variables QORIG and QEVAL are of importance, these need to be handled separately. This was done purposefully because including these would have to lead to the creation of 2 additional variables for every QNAM (i.e. variable to be transposed) in order for them to make sense. We chose not to do this because these variables are rarely needed for analysis purposes. An additional problem that would arise here is how to name these variables which are always linked to a specific QNAM should be named though given that the re-merged data is likely not a submission dataset, the limitation of variable names being no longer than 8 characters long (an XPT v5 restriction) does not apply, hence variable names like [QNAM]_ORIG and [QNAM]_EVAL will likely not cause any concern.

ASSOCIATED PERSONS (APID) AND POOL DEFINITIONS (POOLID)

The Associated Persons concept and the concept of Pool Definitions (SDTM V3.2, IG 1.4) will require the %supp2par_v1 macro to be rewritten or extended since the draft SDTM IG 1.5 includes both APID and POOLID as additional variables on the supplemental qualifier dataset.

SHARING THE CODE: PHUSE'S STANDARD SCRIPTS REPOSITORY

When reporting back on the 2013 Phuse/Computational Science Symposium our group decided to contribute to Working Group 5: Standard Scripts by uploading %supp2par_v1. We easily obtained the green light to do so by convincing management that a pharmaceutical company needs not shield and protect its data processing like it needs to do with the actual compounds. The sharing may actually prove to be beneficial.

NOTE: GOOGLE CODE IS NOW GITHUB

It is important to note at this stage that the Phuse/CSS Standard Scripts Working Group had to switch the platform from Google Code to Github because Google decided to stop the Google Code project. Even though the project is still accessible on Google Code as read-only, the reference provided below links to GitHub.

PHUSE: THE SHARING

The sharing of the code as such was relatively easy. We removed some company-specific pieces of code from the macro to ensure that the code would work outside of our environment. The result was code reviewed and then signed off as “ready for release”. Next the user documentation was verified and signed-off as well and the bundle was uploaded onto Google code.

PHUSE: THE COLLABORATION

Google code had features to write comments about the code in the online storage. The author wrote a few of these comments in the repository for others to pick up on. However, nothing happened. This may be due to a multitude of reasons. Maybe the repository is not sufficiently known to people. Maybe these features for collaboration are not. Maybe people are just too polite to comment or because they feel they would not be allowed to actively contribute in the open domain. This may possibly be one of the areas where the Phuse/CSS Working Group needs to do some more work: have more people know what is there and what is possible.

CONCLUSION

The %supp2par_v1 macro is a valuable tool that remerges supplemental data onto the parent domain. In doing so it actually provides a solid validation of the referential integrity of the received data. Whereas the sharing of the code through the platform provided by the Phuse/CSS WG5 was easy, actually collaborating in further bettering the code seemed more difficult. The latter is possibly due to the concept of collaboration over companies being new to most.

REFERENCES

[1] Van Krunckelsven, D. (2013). Supp2par_v1 as distributed in the PhUSE Standard scripts repository on GitHub: https://github.com/phuse-org/phuse-scripts/tree/504932bda0f297fd06fea85aacbc5839db0c66d9/lang/SAS/datahandle/supp2par or http://tinyurl.com/supp2par

[2] Gregory, M. (2009). "Techniques for writing robust SAS macros. PhUSE 2009.

[3] Van Krunckelsven, D. (2014). Incremental Return Codes. Be warned! About Everything. PhUSE Single Day Event Frankfurt, 2014.

ACKNOWLEDGMENTS

I thank my company for allowing me to contribute with this paper. I thank my manager for making that happen as well as for being very supportive towards me professionally and personally. I thank my colleagues for their input to my activities and for being who they are. Further, I wish to extend my thanks to everyone involved at PhUSE in general and the Standard Scripts Working Group in particular for providing the platforms for collaboration and for their continued dedication to the sharing of information.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Don not hesitate to contact the author at:

Dirk Van Krunckelsven
Merck NV/SA
Brusselsesteenweg 288
B-3090 Overijse
Email: dirk.van.krunckelsven@merckgroup.com