PharmaSUG2010 - Paper AD16

Automating the Link between Metadata and Analysis Datasets
Misha Rittmann, Octagon Research Solutions, Inc., Wayne, PA

ABSTRACT

This paper introduces the %MGENZOD macro, a metadata management tool that generates a zero observation dataset from an Excel® specification document. By using this automatically generated dataset, programming teams are one step closer to ensuring consistency between metadata and analysis data throughout the course of a project/delivery/submission. The tool offers benefits and process improvements to the team. Also, the inclusion of zero observation datasets in the development process is consistent with the process used in the CDISC SDTM/ADaM Pilot Project.

INTRODUCTION: A CASE FOR METADATA MANAGEMENT AND CONSISTENCY

Analysis dataset specification documents are currently the primary method of managing metadata in our programming team. The specification documents are the basis for proceeding to other tasks assigned to us, such as dataset programming, table, listing and figure (TLF) output programming, and so, they are usually the first to be completed. As is typical of working in a development environment, deadlines advance while changes and updates are made. Maintaining specification documents is not always at the top of the team’s priority list when faced with the urgency to finalize outputs. In fact, what usually happens is that these documents are revisited after the other tasks have been completed and teams are charged with the often tedious task of verifying how accurately specification documents reflect analysis datasets. In some cases, revisions will have to be made retrospectively or worse, the consistencies go unnoticed. At that point, however, it is not a simple question of revising and rerunning datasets, it is also a matter of confirming that these changes do not affect TLF output programming, a possibly time intensive task.

%MGENZOD (Macro to GENerate Zero Observation Datasets) is a macro that creates a link between specification documents and analysis datasets, by automatically generating a zero-observation (0-obs) dataset. The specification document contains variable names, types, lengths and labels, which is all the information necessary to set the structure of the final analysis dataset. In using this macro, programming teams will be able to enforce a certain level of consistency between the analysis data and metadata for the duration of the project. Additionally, incorporating this tool offers a number of process improvements.

Finally, using analysis dataset metadata to create 0-obs datasets is part of the process used in the CDISC SDTM/ADaM Pilot Project (“Pilot 1”) to address the following: “The regulatory review team identified lack of consistency between the Define file and the data as a problem in many submissions.” (Pilot Project, p. 24). The Define file is also based on the specification document and incorporating this method will increase efficiency and quality in the long run.

HOW DOES %MGENZOD WORK?

The inner workings of the macro are generally quite simple: filter the document for information that is pertinent to the structure of the dataset – variable attributes, such as name, label, length, and variable type, as well as the dataset name and label, read in the specification document from Excel, translate the information into SAS® code, and voila! We can set the final working dataset into the 0-obs dataset and the structure of the specification document will be retained.
The following (Fig. 1) is a sample specification document for an ADaM ADSL dataset in the format currently used by the team:

**Figure 1. Sample Specification Document**

<table>
<thead>
<tr>
<th>Sponsor</th>
<th>XYZ Pharma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>123-459-789</td>
</tr>
<tr>
<td>Dataset Name</td>
<td>ADSL</td>
</tr>
<tr>
<td>Dataset Label</td>
<td>Subject Level Analysis Dataset</td>
</tr>
<tr>
<td>Prepared By</td>
<td>Misha Rittmann</td>
</tr>
<tr>
<td>Date</td>
<td>March 1, 2010</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Length</th>
<th>Label</th>
<th>Dataset Variable</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDYID</td>
<td>Char</td>
<td>20</td>
<td>Study Identifier</td>
<td>DM</td>
<td>STUDYID = DM.STUDYID</td>
</tr>
<tr>
<td>USUBJID</td>
<td>Char</td>
<td>40</td>
<td>Unique Subject Identifier</td>
<td>DM</td>
<td>USUBJID = DM.USUBJID</td>
</tr>
<tr>
<td>SEX</td>
<td>Char</td>
<td>1</td>
<td>Sex</td>
<td>DM</td>
<td>SEX = DM.SEX</td>
</tr>
<tr>
<td>CONSDT</td>
<td>Num</td>
<td>date9.</td>
<td>Date of Informed Consent</td>
<td>DS</td>
<td>CONSDT = date(DSSTDTC) where DSDECOD = &quot;INFORMED CONSENT&quot;</td>
</tr>
<tr>
<td>BIRTHDT</td>
<td>Num</td>
<td>date9.</td>
<td>Date of Birth (Num)</td>
<td>DM</td>
<td>BIRTHDT = date(BIRTHDT)</td>
</tr>
<tr>
<td>AGE</td>
<td>Num</td>
<td></td>
<td>Age in Years</td>
<td>ADSL</td>
<td>AGE = INT(YRDIF(BIRTHDT, CONDT, 'ACTUAL'))</td>
</tr>
</tbody>
</table>

The first step is to filter the document for usable information. This document contains variable attributes, derivations, source variables, and administrative information such as project information, date of preparation, and author. To create the 0-obs dataset only the first four columns are needed. Variable name, type, length and label information are located in the bolded area. The dataset name and label in the upper left hand corner are also used, however there needs to be a way to distinguish them from variable attributes in the lower part of the document. The fact that a variable must be defined as either a numeric type or character type can be used to determine which rows specify variable attributes and which include the dataset name, label, and the header row labeling the columns.

The next step is to generate SAS code to create the dataset. A data step was the first candidate:

```sas
data adsl_zod;
stop;
attrib STUDYID length = $20 label= "Study Identifier"
USUBJID length = $40 label= "Unique Subject Identifier"
SEX length = $1 label= "Sex"
CONSDT format = date9. label= "Date of Informed Consent"
BIRTHDT format = date9. label= "Date of Birth (Num)"
AGE label= "Age in Years" ;
run;
```

Yielding the following log message:

**Figure 3. SAS Log Using a DATA Step**

NOTE: Variable STUDYID is uninitialized.
NOTE: Variable USUBJID is uninitialized.
NOTE: Variable SEX is uninitialized.
NOTE: Variable CONSDT is uninitialized.
NOTE: Variable BIRTHDT is uninitialized.
NOTE: Variable AGE is uninitialized.
NOTE: The data set WORK.ZOD has 0 observations and 5 variables.
NOTE: DATA statement used (Total process time):
  real time 0.01 seconds
  cpu time 0.01 seconds
The log is proof that a data step can create a 0-obs dataset. Still, this method is not ideal. The data step would ordinarily create a one-observation dataset with all missing values, but including the ‘STOP’ statement fixes. Usually, the length of a numeric variable would default to 8 bytes in the case that it was undefined. Without any actual data though, the data step excluded the AGE variable altogether. Lastly, cluttering up the log with an ‘uninitialized’ message for every variable is any programmer’s nightmare.

Using PROC SQL, instead of a data step, solves our AGE issue:}

```sas
proc sql;
create table work.ADSL_zod (label = "Subject-Level Analysis Dataset")
(
    STUDYID char (20) label = "Study Identifier",
    USUBJID char (40) label = "Unique Subject Identifier",
    SEX char (1) label = "Sex",
    CONSDT num format = date9. label = "Date of Informed Consent",
    BIRTHDT num format = date9. label = "Date of Birth (Num)"
    AGE num label = "Age in Years"
); quit;
```

Figure 4. Using Proc SQL to Create a 0-Obs Dataset

It gets rid of the ‘uninitialized’ messages as well:

```sas
NOTE: Table WORK.ADSL_ZOD created, with 0 rows and 6 columns.
NOTE: PROCEDURE SQL used (Total process time):
   real time           0.01 seconds
   cpu time            0.01 seconds
```

Figure 5. SAS Log Using Proc SQL

The following code takes variable attributes, converted into macro variables, to generate the necessary PROC SQL code:

```sas
proc sql;
create table & outlib. &dsname._zod (label = "Subject-Level Analysis Dataset")
(
    %do i=1 %to &totvar;
    %if %eval (&&varf&i) %then
       /*variable name*/
       &&var&i
    /*variable type*/
    %if %eval (&&typef&i) %then
       &&type&i;
    /*numeric variable format*/
    %if %eval (&&typef&i = 1) %then
       format = &&lngth&i.;
    /*character variable length*/
    %if %eval (&&typef&i = 2) %then
       (&&lngth&i);
    /*label*/
    %if %eval (&&labelf&i) %then
       label = "&label&i";
    /*comma, for all but last variable*/
    %if %eval (&i<&totvar) %then
       ,
    %end;
%end;
quit;
```

Figure 6. Macro-tized SQL Step
Each variable attribute can be uniquely identified with its assigned macro variable ($i$). The attribute macro variables ($&\text{var}&i$, $&\text{type}&i$, $&\text{lngth}&i$, and $&\text{label}&i$) are assigned flags ($&\text{var}&f$i, $&\text{type}&f$i, $&\text{lngth}&f$i, and $&\text{label}&f$i) indicating that values are populated. The macro assumes that lengths will be used for character type variables while formats will be used for numeric type variables. The type flag is set to ‘1’ for a numeric variable and ‘2’ for a character variable controlling for this. The dataset name and label are also assigned macro variables ($&\text{dsname}$, $&\text{dslabel}$) based on the information in the top left hand corner of the specification document (Fig. 1). The ‘$&\text{dsname}$’ macro variable names the 0-obs dataset. Unfortunately, the label of the 0-obs dataset will not be retained in a set statement, but as a global macro variable it can be used (see Fig. 7 below) to label the final output dataset.

The last step is to set the final working dataset into the 0-obs dataset. After the following code runs, a permanent dataset will be output to the ‘derlib’ library reference, where our derived datasets are stored. ‘Adsl_zod’ is the 0-obs dataset created by the %MGENZOD macro and ‘adsl_f’ is the name of the final working dataset. Setting the 0-obs (adsl_zod) dataset first, ensures that the created dataset (derlib.adsl) will conform to the structure of the 0-obs dataset. Note also that the output dataset is labeled using “$&\text{dslabel}$.”

```
data derlib.adsl( label = "&dslabel");
set adsl_zod adsl_f;
run;
```

From that point forward, any changes or inconsistencies related to the specification document will be reflected in the analysis dataset when the code is run, or rerun. Common issues that might occur include additions or removals of variables, label or length changes, etc. or a specified variable length truncating data. In those cases, the programmer would need to ascertain the nature of the change immediately and address the issue. The method above allows plenty of flexibility, but also responsibility, for the programmer.

PROCESS IMPROVEMENTS

Any number of people will access analysis dataset metadata during the course of a project. As such, using %MGENZOD offers a number of advantages to those involved. In order to use the tool, the specification document needs to meet a number of requirements and conform to a certain structure (See Fig. 1 for an example of the document). This puts in place a necessary standard for the document, in addition to any that already exist. For example, defining a variable type as either numeric or character allows the tool to identify information needed to create the dataset. Meaning, that if a variable is not properly defined in the specification document the variable will not be created. Also, the tool identifies variable attributes based on the column order. Thus, the name of the variable will always have to be represented in the first column, the type in the second, and so on. Finally, the tool records the name and label of the dataset to be used in the analysis dataset program. While these requirements do not reflect any substantial change from practices already in use, having them in place moving forward will inhibit specification writers from creating new standards on a whim.

%MGENZOD also offers a number of advantages to the programmer. The 0-obs dataset acts as a preset structure for the final analysis dataset and is readily available during the development stage of the analysis dataset program. Programmers may rest assured that during the course of developing the dataset program the use of variable ordering tools such as LENGTH, ATTRIB, PROC SQL SELECT statements and/or other ordering tools are no longer necessary. Also, the programmer will no longer have to copy and paste labels from the specification document or worry about misspellings/typos. The labels and order of the variables represented in the specification document are captured and retained, respectively, when the 0-obs dataset is set first in a data step as in Figure 7.

The macro is equipped with a number of checks that produce log messages to quickly and efficiently communicate changes that need to be made. The macro checks for issues that will prevent %MGENZOD from running correctly. This includes checking that the location and path of the specification document is correct, that the column indicating the variable name is populated and the variable names are valid SAS names. The macro also checks the format of the specification document confirming that there is at least one variable that is defined as numeric or character, which might otherwise indicate that the type column isn’t where it should be. Additionally, %MGENZOD checks that basic SAS V5 .xpt format /99 compliance rules are followed. If any variable violates any of the rules listed below, a
message will be sent to the log:

- variable and label should not be the same
- variable length should not be greater than 8
- label should not be missing
- label length should be greater than 40
- variable value length should not be greater than 200

The immediate feedback to the programmer about the specification document and dataset that %MGENZOD creates, in turn facilitates communication with the specification-writer regarding updates that need to be made.

While the log messages provide useful feedback, the team uses a number of other tools that are thorough in checking final datasets, such as checking for consistency between datasets. For example, in the case that an ADaM dataset contains an SDTM variable, the attributes of the variable in the analysis dataset should match those in the SDTM dataset. With the metadata in SAS format, we are able to execute this check as soon as the specification documents are completed, rather than having to wait for finalized datasets.

CONCLUSION

Incorporating %MGENZOD into the team’s processes has proven to be a positive step in ensuring metadata and analysis data consistency. Adhering to this process, as suggested by the use of 0-obs datasets in the CDISC Pilot Project, coincides with the goals of Octagon’s programming team. Overall, automating part of this process alleviates the amount effort required to ensure metadata consistency. On the other hand, %MGENZOD does not ensure that specification documents accurately reflect how analysis variables are derived and which source variables are used to derive them. Perhaps, encouraging programmers to do that will be a beneficial side effect of using %MGENZOD.

Thank you for your time!

REFERENCES

“CDISC SDTM/ADaM Pilot Project, Project Report” -
http://www.cdisc.org/stuff/contentmgr/files/0/df91a087c6df43275288267c9fe92180/misc/sdtmadampilotprojectreport.pdf

ACKNOWLEDGMENTS

Thank you to Allan Glaser for his guidance and support during the development of the macro, as well as for the opportunity. Thanks also to the Octagon Research Solutions programming team, especially Terek Peterson, Brian Shilling, Ellen Brookstein and others who provided feedback and support.

CONTACT INFORMATION

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

Misha Rittmann
Octagon Research Solutions, Inc.
585 E. Swedesford Rd.
Wayne, PA 19087
E-mail: mrittmann@octagonresearch.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.