Need to Find a Date?
William Coar, Axio Research, Seattle, WA

ABSTRACT
In a Contract Research Organization (CRO), a large variation in data structure from client to client introduces complexity to seemingly ordinary SAS® programming. Something as simple as working with date variables often requires pre-processing to obtain a structure in a more usable format. For example, it is common to find date information as one or more character variables, or to find date information in three separate month, day, and year variables.

Although the format of variable(s) with date information usually changes from project to project, it is often the case that they are consistently defined across datasets within a project. If the attributes of these variables are also consistently defined they can be particularly helpful in writing a SAS® macro to automatically scan variable attributes within a dataset and subsequently create numeric date variables when appropriate.

A SAS® macro IDDATE is introduced that looks to define such numeric dates without much user input. Relying on variable attributes such as variable names, labels, and type, IDDATE scans the input dataset for potential date variables. If certain project specific conditions are met, IDDATE automatically creates numeric dates.

Although these conditions tend to be project specific, the concept of automatically scanning a dataset and creating numeric dates extends across all projects and clients. This quick tips presentation briefly introduces the concepts of IDDATE and provides two applications to clinical trial data to show its utility. This application uses SAS® 9.2 in a Windows environment.

INTRODUCTION
Data capture is obviously an integral part of conducting a clinical trial. With both electronic data capture (EDC) and the use of more traditional (paper-based) case report forms, there are a variety of systems to choose from. Regardless, it seems as though data entry forms are designed more for ease of use and data entry rather than for data processing. Additionally, fields with calendar information appear throughout.

Emphasis is on the verbiage fields with calendar information, which is not the same as numeric variables with date formats attached. In the CRO setting there are inherent complexities due to large variety of data sources across projects and clients. Obtaining numeric dates in necessary, yet tedious.

Although data dictionaries exist, they are often structured inconsistently and sometimes difficult to digest. An alternative approach is proposed where SAS® automatically scans in input dataset and creates numeric dates when appropriate. A macro based approach relying on consistently defined metadata (variable names, types, and labels) is presented. Macro programming language is outside of the scope of this paper, and discussed elsewhere[1].

Data for adverse reactions to study mediation are considered. Of particular interest is when an adverse reaction (eg, headache, nausea, etc.) starts and stops. These dates are often referred to as the onset date and resolution date of an adverse event. Numeric equivalents are necessary to distinguish whether events start while a patient is receiving study mediation.

The focus of this paper is on creating numeric dates based on metadata without respect to the individual data captured. Specific rules for imputation associated with partial date information are not considered here.

The following two examples will be used for demonstration.

EXAMPLE 1
The first example is a scenario where calendar date information comes in a (single) character variable structured as a string in the form of DD-MMM-YYYY.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Onset Date Derived</th>
<th>Resolution Date Derived</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16-APR-2013</td>
<td>23-APR-2013</td>
</tr>
<tr>
<td>2</td>
<td>12-APR-2013</td>
<td>23-APR-2013</td>
</tr>
<tr>
<td>3</td>
<td>15-MAR-2013</td>
<td>16-MAR-2013</td>
</tr>
</tbody>
</table>
It is known that text within the variable label can be used to identify character date fields. By design of the original database, all labels of character dates will have the string “Date” in them.

EXAMPLE 2

This example is a scenario such that for every calendar date of interest (mm-dd-yyyy) there exists three numeric fields for each component resulting from specific data entry conventions. It is known from correspondence with database developers (for this specific scenario) that any time there is a need for a calendar date, three fields will exist with well defined naming conventions. All three fields will be numeric and named based on root information pre-pending the letters MM, DD, and YY.

**Table 2: Example 2 Adverse Event Data**

<table>
<thead>
<tr>
<th>Obs</th>
<th>AESTDD</th>
<th>AESTMM</th>
<th>AESTYY</th>
<th>AEENDD</th>
<th>AEENMM</th>
<th>AEENYY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>12</td>
<td>2012</td>
<td>30</td>
<td>12</td>
<td>2012</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>1</td>
<td>2013</td>
<td>17</td>
<td>1</td>
<td>2013</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
<td>2013</td>
<td>8</td>
<td>3</td>
<td>2013</td>
</tr>
<tr>
<td>4</td>
<td>27</td>
<td>2</td>
<td>2013</td>
<td>28</td>
<td>2</td>
<td>2013</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>2013</td>
<td>8</td>
<td>3</td>
<td>2013</td>
</tr>
</tbody>
</table>

The month, day, and year for the onset date are: AESTMM, AESTDD, and AESTYY. The fields for resolution date are similarly defined: AEENMM, AEENDD, and AEENYY. The roots of the variable names are AEST and AEEN.

MACRO APPROACH

The proposed approach is macro based and assumes date related fields are consistently defined across datasets within a project. In both examples, it is critical that variables with date information be defined consistently within a given project. The general macro flow is as follows:

1. Step 1: Identify how many variables are in the input dataset.
2. Step 2: Obtain metadata for each variable (name, type, label, etc.)
3. Step 3: Using project specific criteria associated with metadata, obtain a list(s) of variables to be used to create numeric dates.
4. Step 4: If appropriate, perform necessary programming steps to create numeric dates within a data step.

Details associated with each step are provided below.

STEP 1

One method to obtain metadata about each variable is to open the data and read attributes using the SYSFUNC routine. Once open, the number of variables and number of observations can easily be determined using the ATTRN function with parameters NVARS and NOBS as shown below.

Step 1: Determining the number of variables.

```bash
%let ds=%sysfunc(open(&IDDTIN,i));
%let columns=%sysfunc(attrn(&DS,NVARS));
%let rows=%sysfunc(attrn(&DS,NOBS));
```

The input data should subsequently be closed after the metadata are obtained. This can be seen in the sample code provided at the conclusion of this paper.
STEP 2
The second step is to obtain a list of date fields used to create numeric date variables. Looping through each variable, metadata about each is stored in temporary macro variables. This information will be used to generate a list of variables that meet the (project specific) criteria for identifying dates.

Step 2: Determining the attributes of each variable.

* For each variable (i), obtain metadata.;

\[\texttt{\%let col}&i=\%sysfunc(varname(&ds,&i));\]
\[\texttt{\%let ctype}&i=\%qsysfunc(vartype(&ds,&i));\]
\[\texttt{\%let clbl}&i=\%qsysfunc(varlabel(&ds,&i));\]
\[\texttt{\%let clen}&i=\%qsysfunc(varlength(&ds,&i));\]

STEP 3
The metadata that resides in the above macro variables is used to identify date related fields. Specific logic based on the metadata are programed to identify date fields where a running list of variables is created when the criteria are met. Note that this is generally project specific as it relies on consistencies in data structure in order to have any utility. Sample code from both EXAMPLE 1 and EXAMPLE 2 is provided below.

Step 3: Project specific search criteria.

Example 1:
\[\texttt{\%if (&&ctype}&i=C)&(%index(&&clbl}&i,\text{Date}>0) \%then %do;\]
\[\texttt{\%let nc=%eval(&nc+1);}\]
\[\texttt{\%let clist=&clist &&col}&i;\]
\[\%end;\]

Example 2:
\[\texttt{\%if (&&ctype}&i=N)&(%index(&&col}&i,YY)>0 \%then %do;\]
\[\texttt{\%let nc=%eval(&nc+1);}\]
\[\texttt{\%let clist=&clist &&col}&i;\]
\[\%end;\]

STEP 4
Provided there is actually a list of date related fields for which to create numeric dates, much of the remaining processing will be done within a data step making use of the ARRAY statement. This again is a component of the macro that makes this project specific and relies on consistency in data structure within a project.

There are some additional macro processing steps to create lists (of variables) in preparation for the final data step. Each example will be discussed individually.

Example 1
Recall that this example relies on the original character string following a DD-MMM-YYYY date convention. In addition to the list of character date variables, a second list is needed to denote the numeric variables to be created. The convention used to establish variable names for the numeric date simply prepends the string “ZD_” to the original variable name. These two lists are used with ARRAY statements in data set processing.

Example 1: Data step processing.

array \texttt{yd[&NC]} &CLIST;
array \texttt{nd[&NC]} &DLIST;
format &DLIST yymmd10.;
* Loop through the character list creating a numeric for each.;
Example 2

Recall that in EXAMPLE 2, there are three strategically (or conveniently) named variables to identify the month, day, and year. In this case, additional lists and ARRAY statements are utilized. Details for creation of these additional lists can be found in the sample code at the conclusion of this document.

Example 2: Data step processing.

```plaintext
array yd[&NC] &CLIST;
array md[&NC] &MMLIST;
array dd[&NC] &DDLIST;
array nd[&NC] &DLIST;
format &DLIST yymmdd10.;;
* Loop through the character list creating a numeric for each.;
do j=1 to &NC;
   if (md[j] ge 0) and (dd[j] ge 0) and (yd[j] ge 0) then
      nd[j]=mdy(md[j],dd[j],yd[j]);
   end;
* Loop through the list of dates and add a label to each one.;
%do j=1 to &NC;
   %let thisone=%scan(&CLIST,&j);
   label %scan(&DLIST,&j) = "Macro generated DATE field from &THISONE";
%end;
```

RESULTS

A simple macro call allows for creation of numeric dates without the need to specify anything other than input and output data set names.

Sample Macro Call

```
%iddate(iddtin=aes, iddtout=aesd);
```

The results for both examples are displayed below.

Table 3: Results for Example 1

<table>
<thead>
<tr>
<th>Obs</th>
<th>Onset Date Derived</th>
<th>Macro generated DATE field from AEODDV</th>
<th>Resolution Date Derived</th>
<th>Macro generated DATE field from AERDDV</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>15-MAR-2013</td>
<td>2013-03-15</td>
<td>16-MAR-2013</td>
<td>2013-03-16</td>
</tr>
<tr>
<td>4</td>
<td>17-MAY-2013</td>
<td>2013-05-17</td>
<td>07-JUN-2013</td>
<td>2013-06-07</td>
</tr>
<tr>
<td>5</td>
<td>17-MAY-2013</td>
<td>2013-05-17</td>
<td>07-JUN-2013</td>
<td>2013-06-07</td>
</tr>
</tbody>
</table>
Table 4: Results for Example 2

<table>
<thead>
<tr>
<th>Obs</th>
<th>Onset Date, day</th>
<th>Onset Date, month</th>
<th>Onset Date, year</th>
<th>Macro generated DATE field from AESTYY</th>
<th>Resolution Date, day</th>
<th>Resolution Date, month</th>
<th>Resolution Date, year</th>
<th>Macro generated DATE field from AEENYY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>12</td>
<td>2012</td>
<td>2012-12-24</td>
<td>30</td>
<td>12</td>
<td>2012</td>
<td>2012-12-30</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>1</td>
<td>2013</td>
<td>2013-01-12</td>
<td>17</td>
<td>1</td>
<td>2013</td>
<td>2013-01-17</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
<td>2013</td>
<td>2013-03-07</td>
<td>8</td>
<td>3</td>
<td>2013</td>
<td>2013-03-08</td>
</tr>
<tr>
<td>4</td>
<td>27</td>
<td>2</td>
<td>2013</td>
<td>2013-02-27</td>
<td>28</td>
<td>2</td>
<td>2013</td>
<td>2013-02-28</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>2013</td>
<td>2013-03-07</td>
<td>8</td>
<td>3</td>
<td>2013</td>
<td>2013-03-08</td>
</tr>
</tbody>
</table>

As seen in both Tables 3 and 4, numeric dates are automatically created based on the appropriate criteria using variable level metadata. These numeric dates are now readily available for other programming tasks such as sorting or making comparisons with other numeric dates.

The complete code for IDDATE associated with EXAMPLE 2 is provided at the end of this paper. This code is easily adapted to accommodate the simpler EXAMPLE 1.

CONCLUSION

The two examples provided describe two straightforward yet common settings in working with clinical trial data. Although each example came from different projects, IDDATE only has minor modifications to accommodate project specific criteria for determining date variables. The macro has tremendous utility since the date variables (structure and attributes) are consistently defined within each project. Once the project specific form of the macro is in place, the user is able to obtain numeric dates with little effort and without the need to reference a data dictionary. As emphasized repeatedly throughout this paper, consistency of variables with date information is directly related to the usefulness of this approach.

This is not meant to be a catch all macro, but continues to be extremely useful for many projects, including those with date information in character strings following the ISO8601 date format[2]. There are cases where the assumptions for indentifying dates within a given dataset fail. These cases are easily dealt with on an individual bases rather than updating the macro code with added complexities.

The intention of IDDATE is to simplify creation of numeric dates when possible. The utility is directly related to the number of datasets where assumptions hold. Although the examples presented only deal with adverse events data, the application easily extends to any dataset within a project as long as the consistency assumption holds true.

REFERENCES


ACKNOWLEDGMENTS

The author would like to thank Darrin Despain for his review and insights contributing to the paper. The author would also like to acknowledge the many programmers publishing information on the internet, and apologizes to those not properly referenced above citing the inherent nature of internet searching makes it difficult to track so many individual ideas.
Need to Find a Date?, continued

CONTACT INFORMATION

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

William Coar, PhD
Biostatistician/Director of Central Region Operations
Axio Research, LLC
11001 West 120th Avenue, Suite 400
Broomfield, CO 80021
Email: williamc@axioresearch.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.
%macro iddate(iddtin=, iddtout=);
%local i;

* Open the input dataset to find out how many variables and observations there are.;
%let ds=%sysfunc(open(&IDDTIN,i));
%let columns=%sysfunc(attrn(&DS,NVARS));
%let rows=%sysfunc(attrn(&DS,NOBS));

* Initialize counts of numeric and character variables, and initialize a list of each.;
%let nc=0;
%let nn=0;
%let nlist=;
%let clist=;

* Loop though each column. Identify the variable name, type, label, and length (all can be used for finding date variables). In this example, variables associated with dates are identified by an existing YEAR variable, where YY is in the variable name. This assumes there exists month and day variables similarly named.;
%do i=1 %to &columns;
  %let col&i=%sysfunc(varname(&ds,&i));
  %let ctype&i=%qsysfunc(vartype(&ds,&i));
  %let clbl&i=%qsysfunc(varlabel(&ds,&i));
  %let clen&i=%qsysfunc(varlength(&ds,&i));

  * If the variable is numeric and has a YY in it, then consider this in the list of dates. If so, add it to the list of date variables for this dataset.;
  %if (&&ctype&i=N)&(%index(&&col&i,YY)>0) %then %do;
    %let nc=%eval(&nc+1);
    %let clist=&clist &&col&i;
  %end;
%end;

* Close the dataset since it is no longer needed.;
%let rc=%sysfunc(close(&DS));

* Only do the processing if there are date variables to create.;
%if &nc>0 %then %do;

* In this example, need 3 lists. One is a list of the root variable name (ie, name without the YY), and two that are lists of the month and day variables.;
  %let TCLIST=%sysfunc(tranwrd(&CLIST,YY, ));
  %let MMLIST=%sysfunc(tranwrd(&CLIST,YY,MM));
  %let DDLIST=%sysfunc(tranwrd(&CLIST,YY,DD));

  * Create strings for numeric dates and times to be created. Note the variable name consists of ZD_ appended to the root variable name.;
  %let dlist=;
%end;
%do i=1 %to &nc;
   %let dlist=&dlist ZD_%scan(&Tclist,&i," ");
%end;

* Creation of the output dataset.;
data &iddtout;
   set &iddtin;
   format &DLIST yymmdd10.;
   array yd[&NC] &CLIST;
   array md[&NC] &MMLIST;
   array dd[&NC] &DDLIST;
   array nd[&NC] &DLIST;

   * Only try to create dates if all components are non-missing non-negative numbers. These criteria are in place do to data entry conventions in this example.;
   do j=1 to &NC;
      if (md[j] ge 0) and (dd[j] ge 0) and (yd[j] ge 0) then
         nd[j]=mdy(md[j],dd[j],yd[j]);
   end;
   * Add labels to the macro generated numeric date.;
   %do j=1 %to &NC;
      %let thisone=%scan(&CLIST,&j);
      %let label=%scan(&DLIST,&j) = "Macro generated DATE field from &THISONE";
   %end;
   drop j;
   run;
%end; %else %do;
   data &IDDTOUT;
      set &IDDTIN;
      run;
%end;
%mend;