SESUG Paper 042-2021

SAS® Macro to Identify Potentially Obsolete Variables in a File

Thomas E. Billings, MUFG Union Bank, N.A., San Francisco, California

ABSTRACT

The problem: you have SAS® datasets and/or database tables and want to identify variables that might be obsolete. A SAS macro is presented that examines all the variables in each row of a file and determines the earliest and latest as-of dates/datetimes when each variable is populated. The macro requires that the file has an as-of date/datetime variable in each row, and that variable cannot be missing/null. The macro creates a SAS dataset with the derived by-variable metadata, and those metadata are easily tested to identify variables that may be obsolete/deprecated.

Keywords: DATA step, RETAIN, CALL SYMPUTX.
SAS® products: Base SAS.
User level: intermediate+.

INTRODUCTION

The situation: a tool that is used to enter data for a specific purpose/application – let's call it the old or legacy system – has run for many years and experienced numerous changes over the years, and now is scheduled to be replaced by a new system. Both the old and new systems produce data, and those data are captured in SAS® datasets (and/or a relational database). During the lifespan of the old system, many new variables were added, and some variables and even some tables/files, have become deprecated over time in the system-related datasets and database. Also, the documentation for the system is poor; the deprecated tables are well known but it is unclear which variables in active tables are deprecated or obsolete.

To preserve/support the numerous applications and reports based on the old system, the new system data will be mapped into the existing datasets for the old system, with a few new variables added. However, deprecated variables will not be supplied in the new system, so those variables will be set to missing (or other default values) for records coming from the new system. We need to identify deprecated variables in the old system, to make sure that all active (non-deprecated) variables are supplied in the new system, and to avoid wasting resources on deprecated variables.

A SAS macro is presented here to identify potentially obsolete variables in a dataset/database table. For the macro to be successfully applied, the target datasets/tables must have an as-of date variable (which can be a SAS date or datetime) in each row. The user can specify the SAS format to be used with the as-of date. For the analysis here, a variable is considered obsolete if the most recent as-of date when the value is non-missing/non-null is before some reference date. (The reference date can be a single date for the entire set of tables being analyzed or can vary/be by-table/dataset. Reference dates should be determined in consultation with the relevant SMEs and/or system users.)

PROCESSING OVERVIEW

The macro analyzes input dataset/tables and produces an output (WORK) dataset that lists the tables, variable names, and earliest/latest population dates for each variable. Given a target SAS dataset (or database table) to be analyzed, the processing is as follows.

1. Use PROC CONTENTS with the OUT= option to create a dataset with the metadata for the target file.
2. Using a DATA _NULL_ step, create separate macro variables: count - # numeric variables in the file, count - # character variables in the file, a string that contains the character variable names (separated by blanks), and a similar string that contains the numeric variable names.
3. Check: if the file has no variables (# numeric = # character = 0) then print warning message and quit.
4. Use a DATA step to create an output file that contains the: table/dataset name, variable name, earliest date the variable is populated (i.e., not missing/not null), the latest date the variable is populated. The main logic steps are, with similar but separate blocks of code for character and numeric variable types:
   - Define ARRAYs that contain the variable values, plus RETAIN variable ARRAYs that will contain the earliest and latest non-missing as-of date for each variable
   - Use DO loops over the arrays to check the values for each variable (missing/non-missing) in each row, setting the value of earliest ("first") and latest as-of date associated with the variable.
   - At end of the input file, output the array values into a SAS dataset that contains 1 row for each variable, with the values: libref, dataset/table name, variable name, earliest non-missing as-of date, latest non-missing as-of date.
5. Finally, concatenate the dataset created above with other tables created by the macro. This is to allow checking a group of (related) datasets/tables (this is optional/can be deleted from the macro if you prefer).

ANNOTATED CODE

The macro below has a rather odd option, zerofilt=. This is because one of the database tables to be analyzed has an odd feature: for numeric variables, zero usually means the value is missing (i.e., zero is used instead of null). The keyword option defaults to N (for No), so it can be ignored when analyzing datasets that use missing or null in the standard manner.

```
/* Parameters: inlibref, inlibn=: libref and table/dataset name.
    datevar=, target date/datetime variable, datefmt=format (no .) to use
    zerofilt=N the zero=missing option */

%macro date_util(inlibref=, inlibn=, datevar=, datefmt=, zerofilt=N);
      %local for_set;

/* get metadata, save in temp file */

      proc contents data=&inlibref..&inlibn. out=work.temp  (keep= memname libname
name type length) noprint;
      run;

/* input temp metadata file */

      data _null_;
            set work.temp end=temp_end;
            length num_numeric num_char 8   nvarlist cvarlist  $32000;
            retain num_numeric num_char nvarlist cvarlist;

            if _n_ = 1 then   /* initialize variables */
```

```
                do;
                        num_numeric = 0;
                        num_char = 0;
                        call missing(nvarlist, cvarlist);
                end;

        if (type = 1) then   /* count numeric variables, append names to list */
                do;
                        num_numeric = num_numeric+1;
                        nvarlist=cat(strip(nvarlist)," ",strip(name));
                end;
        else
                do;      /* count character variables, append name to list */
                        num_char = num_char+1;
                        cvarlist=cat(strip(cvarlist)," ",strip(name));
                end;

        if (temp_end) then   /* copy counts, varnames to macro variables */
                do;
                        call symputx("num_numeric",strip(put(num_numeric,5.)),G);
                        call symputx("num_char",strip(put(num_char,5.)),G);
                        call symputx("nvarlist",strip(nvarlist),G);
                        call symputx("cvarlist",strip(cvarlist),G);
                end;
    run;

/* debug prints */

    %put File: &inlibref..&inlibn.: num_numeric=&num_numeric.;
    %put File: &inlibref..&inlibn.: num_char=&num_char.;

    %put nvarlist=&nvarlist.;
    %put cvarlist=&cvarlist.;

/* if file has no variables, exit with warning message */

    %if (&num_numeric. = 0) and (&num_char. = 0) %then
        %do;
            %put WARNING:  FILE &inlibref..&inlibn. has no variables;;

            %return;
        %end;

/* process target table, identify earliest/latest as-of dates for each variable */

    data work.s_&inlibn.;
        set &inlibref..&inlibn. end=endinp;
        length inlibref $40 inlib_file_name $8 varname $40 vartype $20;

        %if (&num_numeric ne 0) %then
            %do;
                length nv_first1-nv_first&num_numeric.  nv_latest1-
nv_latest&num_numeric.  8.;
                retain nv_first1-nv_first&num_numeric.  nv_latest1-
nv_latest&num_numeric.  8.;
/* NVF = first/earliest
                array NVF{&num_numeric.} nv_first1-nv_first&num_numeric.;
                array NVL{&num_numeric.} nv_latest1-nv_latest&num_numeric.;
                array NN{&num_numeric.} &nvarlist.;

                if _n_ = 1 then
                    do;  /* initialize arrays */
                        do i = 1 to &num_numeric.;
```

```
                                        call missing(NVF{i},NVL{i});
                                end;
                        end;

                    do i = 1 to &num_numeric.;
                        %if (&zerofilt. ne N) %then
                                %do;
                                        if NN{i} = 0 then
                                                call missing(NN{I});
                                %end;
                    /*  check, set values for earliest, latest as-of dates */
                                if not missing(NN{I}) then
                                        do;
                                                if missing(NVF{i}) then
                                                        NVF{i} = &datevar.;
                                                else NVF{i} = min(&datevar.,NVF{I});

                                                if missing(NVL{i}) then
                                                        NVL{i} = &datevar.;
                                                else NVL{i} = max(&datevar.,NVL{I});
                                        end;
                    end;

                    if (endinp) then  /* at end of file, output results */
                                do;
                                        do i = 1 to &num_numeric.;
                                                inlibref = "&inlibref.";
                                                inlib_file_name = "&inlibn.";
                                                varname = vname(NN{I});
                                                earliest = NVF{i};
                                                latest = NVL{i};
                                                vartype = "NUMERIC";
                                                output;
                                        end;
                                end;
                        %end;

/*  similar logic for character variables in target table  */
                        %if (&num_char ne 0) %then
                                %do;
                                        length cv_first1-cv_first&num_char.  cv_latest1-
cv_latest&num_char.  8.;
                                        retain cv_first1-cv_first&num_char.  cv_latest1-
cv_latest&num_char.  8.;
                                        array CVF{&num_char.} cv_first1-cv_first&num_char.;
                                        array CVL{&num_char.} cv_latest1-cv_latest&num_char.;
                                        array CN{&num_char.} &cvarlist.;

                                        if _n_ = 1 then
                                                do;
                                                        do i = 1 to &num_char.;
                                                                call missing(CVF{i},CVL{i});
                                                        end;
                                                end;

                                        do i = 1 to &num_char.;

                                                if not missing(CN{I}) then
                                                        do;
                                                                if missing(CVF{i}) then
                                                                        CVF{i} = &datevar.;
                                                                else CVF{i} = min(&datevar.,CVF{I});
```

```
                                        if missing(CVL{i}) then
                                              CVL{i} = &datevar.;
                                        else CVL{i} = max(&datevar.,CVL{I});
                                  end;
                            end;

                            if (endinp) then
                                  do;
                                        do i = 1 to &num_char.;
                                              inlibref = "&inlibref.";
                                              inlib_file_name = "&inlibn.";
                                              varname = vname(CN{I});
                                              earliest = CVF{i};
                                              latest = CVL{i};
                                              vartype = "CHARACTER";
                                              output;
                                        end;
                                  end;
                      %end;

                  keep inlibref inlib_file_name varname earliest latest vartype;
                  format earliest latest &datefmt..;
            run;

/*  output is to a WORK file, same name as input file  */
            proc sort data=work.s_&inlibn.;
                  by latest;
            run;

            data work.concat_files;  /*  can delete this part if not useful to your app  */
                  %if (&setflag. = 0) %then
                        %do;
                              %let for_set=;
                              %let setflag=1;
                        %end;
                  %else
                        %do;
                              %let for_set=%str( )work.concat_files%str( );
                        %end;

                  set &for_set. work.s_&inlibn.;
            run;

            proc datasets lib=Work nolist nowarn;
                  delete temp;
            quit;

%mend;
```

USAGE NOTES

The macro above tests only for missing (or in the zerofilt option, for 0 which is reset to missing). An obsolete variable might be set to – instead of null/missing - a specific default (e.g., 'N" for No for a Yes/No variable). If there are only a few such variables, the code above could be patched to set those variables to missing – similar to the zerofilt option. If there are many such variables with different default values, then another array can be created with the default values. That implies the default values are input into SAS and loaded into the array. We do not present such code here as it is "left as an exercise for the reader".

If the earliest and latest as-of dates are both missing, then the variable is not only deprecated, but it is uninitialized, i.e., was/is always null/missing.

PROC CONTENTS is used instead of the DICTIONARY tables/views, as it is more efficient. DICTIONARY calls can take a long time.

Some database tables are huge, and caution/discretion should be exercised when using this macro on a database. You should create a sample extract to test, instead of trying to read a huge file into SAS. Also, you might want to modify the macro to make the output libref and dataset names into keyword macro variables; that would allow you to set them in your macro calls.

SAMPLE OUTPUT (PARTIAL)

inlibref	inlib_file_name	varname	vartype	earliest	latest
redacted	testfil	var1	NUMERIC	.	.
redacted	testfil	var2	NUMERIC	.	.
redacted	testfil	var3	NUMERIC	.	.
redacted	testfil	var4	CHARACTER	.	.
redacted	testfil	var5	CHARACTER	.	.
redacted	testfil	var6	NUMERIC	29-Mar-17	29-Mar-17
redacted	testfil	var7	CHARACTER	29-Mar-17	25-Jul-20
redacted	testfil	var9	CHARACTER	29-Mar-17	01-Jun-21
redacted	testfil	var10	NUMERIC	24-May-18	20-Apr-19
redacted	testfil	var11	NUMERIC	24-May-18	30-Apr-19
redacted	testfil	var12	NUMERIC	21-Feb-17	24-Jul-20
redacted	testfil	var13	NUMERIC	24-Jul-18	9-Oct-20
redacted	testfil	var14	NUMERIC	17-Jan-17	23-Dec-20

TO IDENTIFY DEPRECATED VARIABLES

The macro creates an output dataset that contains the target table names, variable names, and earliest/latest populated as-of dates/datetimes. There is 1 row for each variable analyzed by the macro. Variables where both dates are missing are uninitialized, and variables where the latest population date is less than a reference date (that you determine), are potentially deprecated or obsolete.

EPILOGUE

This macro can create a list of probable obsolete and uninitialized variables in (a set of) SAS datasets and/or database tables. However variables that are rarely populated might be present in the target files, and before you formally classify a variable as obsolete, you should check with SMEs (subject matter experts) to confirm that the identified variables are indeed obsolete/deprecated or uninitialized. (In other words, apply due diligence to avoid false positives.)

Note: As of the date of publication, the author is an employee of MUFG Union Bank, N.A. The content of this paper does not reflect the views or opinions or work product of MUFG Union Bank.

APPENDIX 1:
BSD 2-CLAUSE COPYRIGHT LICENSE (OPEN SOURCE)

```
* All program code in this paper is released under a Berkeley Systems
  Distribution BSD-2-Clause license, an open-source license that permits
  free reuse and republication under conditions;
```


CONTACT INFORMATION

A list of the author's SAS-related papers, including URLs for free access, is available at the URL (hosted by Google Drive): https://goo.gl/uCUHoa

Note: Your enterprise web filter might prevent access to this URL from work, in which case you will need to access via a personal device.

Thomas E. Billings
Email: tebillings@gmail.com