Paper DM-40

Using Base SAS® to Automate Quality Checks of Excel Workbooks That Have Multiple Worksheets

Lisa Mendez, PhD, IQVIA Government Solutions
Andrew T. Kuligowski

ABSTRACT

This case study provides a real-world example of how Base SAS was used to read in over 985 Excel workbooks to check the structure of over 90,000 worksheets – and to repeat the Process quarterly. It will illustrate how components such as the LIBNAME XLSX Engine, PROC SQL (to create macro variables), SAS Dictionary Tables, and SAS Macros were used together to create exception reports exported to MS Excel workbooks. The structure of the worksheets, such as worksheet names and variable names, were checked against pre-loaded templates. Values within the worksheets were also checked for missing and invalid data, percent differences of numeric data, and ensuring specific observations were included in specific worksheets. This case study describes the Process from its inception to the ongoing enhancements and modifications. Follow along and see how each challenge of the Process was undertaken and how other SAS User Group conference proceeding papers contributed to this quality check Process.

INTRODUCTION

This paper is broken out into sections starting with the background and problem, with sections of code to conduct the quality checks. We wanted to provide the entire Process from determining how to identify the problems, finding the different code for SAS, and then the ways to implement the code, that will complete the quality check Process.

BACKGROUND

The data are created and delivered by an internal team (the developers) and delivered to the GS team. The data are delivered every quarter. The specifics of the data for each quarter is as follows:

- There were five markets (ADHD, BNZD, CNNB, CDNE, and PAIN)
- Each market had seven Excel Workbooks that needed to be checked.
- Each Workbook had various multiple worksheets
    - ADHD – 7 worksheets
    - BNZD – 24 worksheets
    - CNNB – 7 worksheets
    - CDNE – 5 worksheets
    - PAIN – 55 worksheets

THE PROBLEM

Previously, each quarter the GS team reviewed the Excel workbooks manually. Checking workbooks manually is not only time consuming, but can be prone to errors. Lisa was asked to create an automated Process where the multiple MS Excel® workbooks can be checked before being sent to the client. This Process had to be completed in two weeks.

Doing the math there were 5 markets multiplied by 7 workbooks (35 workbooks) that had a total of 98 worksheets that needed to be checked. That was 3,430 worksheets.

That wasn't the biggest challenge…we not only had to run the current quarter, but also the historical data as well. Therefore, there were 27 quarters total, which totaled 92,610 worksheets worth of data that had

to be checked…in two weeks! This problem was solved doing lots of Google searches for SAS code, and "down and dirty" programming. Please, keep this in mind when reading this paper.

GETTING THE DATA INTO SAS

One of the first things that had to be solved was how to get the data into SAS - Reading in the Excel Workbooks and worksheets in order to check the structure (worksheet names and variable names). The issue wasn't so much getting the data into SAS, but checking the worksheet names and the variable names, which is part of the metadata and not the data itself.

After doing a few Google searches, the LIBNAME XLSX engine seemed to be the best choice. SAS 9.4 Maintenance 2 added the XLSX engine, which allows you to read and write Microsoft Excel files as if they were data sets in a library [Hemedinger] The advantage of using this engine is that it accesses the XLSX file directly, and does not use the Microsoft data APIs as a go-between. (LIBNAME EXCEL and LIBNAME PCFILES rely on those Microsoft components.) [Hemedinger]. What that means is that you can use this engine on Windows or Unix systems without having to worry about bitness (32-bit versus 64-bit) or setting up a separate PC Files Server Process [Hemedinger]. Make a note that you have to have a license for SAS/ACCESS to PC Files to utilize the XLSX engine. If you use SAS University Edition, the SAS/ACCESS product is part of the that package [Hemedinger].

Sample LIBNAME XLSX engine code:

```
libname cadhd 'C:\Users\lmendez\Documents\RMPDC\Deliverables\2017_Q3\ADHD';
```

Sample LIBNAME XLSX engine code with macro variables:

```
libname cadhd "C:\Users\lmendez\Documents\RMPDC\Deliverables\&year._&qtr.\ADHD";
```

LOADING THE DATA

Now that there is a way to potentially get the data into SAS in an efficient way, how do we load the data for every worksheet in the workbook? Remember the LIBNAME XLSX engine just created the datasets; it doesn't load the data into the datasets.

To state the problem in more detail, how do we get the worksheet names into a dataset so that a macro can be used to load the data for all worksheets, and to have a list of worksheet names (dataset names) to compare against later? Using the XLSX the worksheet names are now data set names. Researched revealed a few resources that discussed SAS Dictionary tables and utilized PROC CONTENTS for the solution. A few papers were found that utilized PROC SQL and the dictionary tables to create a dataset of the worksheet names that are read in from the XLSX engine [see list of resources].

The following code creates a dataset with all the worksheet/dataset names, creates a macro variable with all the worksheet/dataset names, and loads the data from each worksheet of a workbook:

```
/*----------------------------------------------------------------------------*/
/* create a data set with all of the Excel Workbook sheet names               */
/*----------------------------------------------------------------------------*/
proc sql noprint;
    create table cadhd.cadhd1 as
    select *
    from sashelp.vmember
    where libname="CADHD1"
    ;
quit;
```

Figure 1 shows the output dataset created from the PROC SQL code.

	Library Name	Member Name	Member Type	DBMS Member Type	Engine Name	Indexes	Pathname
1	CADHD1	LOOKUP	DATA		XLSX	no	C:\Users\lmendez\Documents\RMPDC\De Tracking_adhd_NDW_2018Q2.xlsx
2	CADHD1	STATE_SUBGRP	DATA		XLSX	no	C:\Users\lmendez\Documents\RMPDC\De Tracking_adhd_NDW_2018Q2.xlsx
3	CADHD1	STATE_SUPERGRP	DATA		XLSX	no	C:\Users\lmendez\Documents\RMPDC\De Tracking_adhd_NDW_2018Q2.xlsx
4	CADHD1	ZIP_SUBGRP_AMPH	DATA		XLSX	no	C:\Users\lmendez\Documents\RMPDC\De Tracking_adhd_NDW_2018Q2.xlsx
5	CADHD1	ZIP_SUBGRP_METH	DATA		XLSX	no	C:\Users\lmendez\Documents\RMPDC\De Tracking_adhd_NDW_2018Q2.xlsx
6	CADHD1	ZIP_SUBGRP_OTH_ANAL	DATA		XLSX	no	C:\Users\lmendez\Documents\RMPDC\De Tracking_adhd_NDW_2018Q2.xlsx
7	CADHD1	ZIP_SUBGRP_OTH_ANTI	DATA		XLSX	no	C:\Users\lmendez\Documents\RMPDC\De Tracking_adhd_NDW_2018Q2.xlsx
8	CADHD1	ZIP_SUPER	DATA		XLSX	no	C:\Users\lmendez\Documents\RMPDC\De Tracking_adhd_NDW_2018Q2.xlsx

Figure 1. Output from the PROC SQL Code.

The following is the SAS code to utilize PROC SQL and macro variables:

```
/*----------------------------------------------------------------------------*/
/* create a list of all sheet names into a macro variable that will be used to */
/* import all of the data within the worksheet into datasets in the work      */
/* library                                                                    */
/*----------------------------------------------------------------------------*/
proc sql noprint ;
     select memname
     into :snamlist_1 separated by '*'
     from cadhd.cadhd1
     ;
quit;

/*----------------------------------------------------------------------------*/
/* determine the number of worksheets in the workbook and store the value in a */
/* macro variable                                                             */
/*----------------------------------------------------------------------------*/
proc sql noprint;
     select count(memname)
     into :n_1
     from cadhd.cadhd1
     ;
quit;

%put &snamlist_1;  /* show the macro variable snamlist in the log */
%put &n_1;         /* show the macro variable n_1 I the log */
```

Figure 2 is the log output where the macro variables are written.

```
61
62   %put &snamlist_1;
LOOKUP*STATE_SUBGRP*STATE_SUPERGRP*ZIP_SUBGRP_AMPH*ZIP_SUBGRP_METH*ZIP_SUBGRP_OTH_ANAL*ZIP_SUBG
RP_OTH_ANTI*ZIP_SUPER
63   %put &n_1;
8
```

Figure 2. Log output.

The following code loads the data into the created datasets using a macro and the created macro variables:

```
/*-----------------------------------------------------------------------------*/
/* Macro: M1 - this macro reads in the names of the worksheets and creates and */
/* populates all of the data in the worksheet into datasets in the work library*/
/*-----------------------------------------------------------------------------*/
options mprint;
%macro cp1_1;
    %do i = 1 %to &n_1;
    %let var = %scan(&snamlist_1,&i,*);

    PROC IMPORT OUT= cadhd.&var
        DATAFILE = "C:\Users\lmendez\Documents\RMPDC\Deliverables\&year._&qtr.\ADHD\&workbook..xlsx"
            DBMS = XLSX REPLACE;
        SHEET = "&var";
        GETNAMES = YES;
    RUN;

    %end;

%mend cp1_1;

/* invoke macro */
%cp1_1;
```

VALIDATE WORKSHEET/DATASET NAMES

Now that the data are loaded, how do we verify that the worksheet/dataset names and the variable names are correct? It was determined the optimal solution under the circumstances would be to create some templates comparison. Having a dataset that lists the valid worksheet names, and another data set of valid variables names for each market, will allow us to ensure everything is accounted for, spelled correctly -- and that no additional worksheets or variables are included. This is possible for this particular situation since the worksheet names and variables are the same for all workbooks within a market. Therefore, only five worksheet name lists and five variable name lists were needed.

At this point it is assumed that the worksheet names data set and the variable names dataset have been loaded to a permanent library that will be referenced later in the paper.

To compare the worksheet names from the template to the current data, it is recommended that the variable for the worksheet/dataset name be different than the template variable name. It is not necessary, but helps when comparing the datasets.

The following SAS code compares the worksheet names.

```
 /* compare the worksheet names from adhd1 to the adhd template and create error report */
proc sql;
     create table adhdw_compare as
     select t.memname, c.memname2
     from tadhd.tadhd1 as t FULL JOIN cadhdw as c
     on t.memname = c.memname2
     ;
 quit;
```

It was determined that error reports would be created that will allow individuals without SAS to review and determine if any data are incorrect. Utilizing flags, an error report is created for all worksheets and is output to a MS Excel Workbook. Each worksheet generated will house an error report for the respective worksheet being validated.

The following is a data step where the table (dataset) that was created from the Prod SQL join is checked to see if there are additional or missing worksheet names. If there are, then an error message is created:

```
/* Create Error Report Dataset */
/* If this data set has zero observations, then no errors were found */
data error.worksheet_error_report_adhd&x (drop = memname memname2);
    length error_msg $175. workbook_name $50.;
    set adhdw_compare;

    workbook_name = "&workbook";

    if memname = '' then error_msg = 'Error - An additional worksheet was found or a worksheet name does not match the template:
                                      Current worksheetname = ' || memname2;

    if memname2 = '' then error_msg = 'Error - Sheet name missing from the current workbook.  Sheetname missing: ' || memname;

    if memname ne '' AND memname2 ne '' then delete;
run;
```

VALIDATE VARIABLE NAMES

Variable names are checked a little differently than checking the worksheet/dataset names, but both use similar code. The difference is that we utilized the macro variable for the list of dataset names and a macro with PROC Contents to get the list of the variable names for each worksheet/dataset.

The following code creates a macro variable with the worksheet names:

```
/*=============================================================================*/
/*                              Load Varnames                                  */
/*=============================================================================*/

/* cadhd.cadhd1 is a list of all worksheets in the template */

proc sql noprint;
    select memname
    into :snamlist_2 separated by '*'
    from cadhd.cadhd1
    ;
quit;


proc sql noprint;
    select count(memname)
    into :n_2
    from cadhd.cadhd1
    ;
quit;

%put &snamlist_2;
%put &n_2;
```

Writing debugging information to the log to ensures that the worksheet names are read in correctly, along with validating the number of worksheets processed (there are eight worksheets). See Figure 3.

```
132
133   %put &snamlist_2;
LOOKUP*STATE_SUBGRP*STATE_SUPERGRP*ZIP_SUBGRP_AMPH*ZIP_SUBGRP_METH*ZIP_SUBGRP_OTH_ANAL*ZIP_SUBG
RP_OTH_ANTI*ZIP_SUPER
134   %put &n_2;
8
```

Figure 3. Log output showing worksheet names and number of worksheets.

The following code uses a macro to create the list of variable names for each worksheet:

```
/*-------------------------------------------------------------------------*/
/* create a list of all variable names into a macro variable that will be used  */
/* to create a lists of variable per dataset                                */
/*-------------------------------------------------------------------------*/

options mprint;
%macro cp1_2;
    %do i = 1 %to &n_2;
    %let var = %scan(&snamlist_2,&i,*);
    proc contents data = cadhd.&var
        out = cadhd.vars_&var (keep = varnum name)
        noprint;
    run;

    %end;

%mend cp1_2;

/* invoke macro */
%cp1_2;
```

The methodology used to compare the variables names to a pre-loaded template is the same as the methodology that was employed prior to comparing the worksheet/dataset names.

After the error reports are created, they are exported to another Excel spreadsheet so that other team members can verify the errors. It is best practice not to have the same person/people who run the SAS QC code, perform the checking of the error reports if possible. This provides another level of QC check.

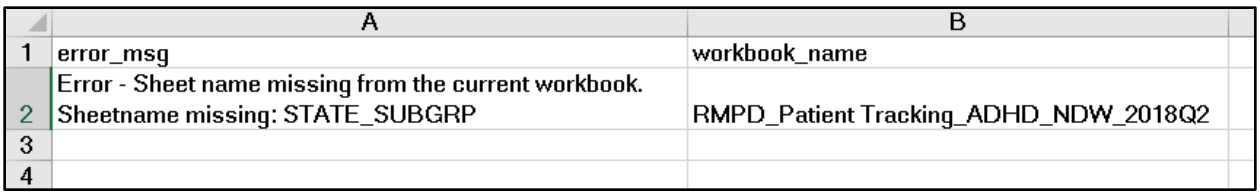
The following code exports the error reports to an Excel workbook. Each Workbook contains all of the error reports for each worksheet:

```
/* Export the error reports */
PROC EXPORT DATA= error.Variables_error_report_adhd&x
            OUTFILE= "C:\Users\lmendez\Documents\RMPDC\Deliverables\&year._&qtr.\
                        ADHD\error\Variables_error_report_&year._&qtr._adhd.xls"
            DBMS=EXCEL LABEL REPLACE;
     SHEET="Variables_error_report_adhd&x";
RUN;

/* Export the error reports */
PROC EXPORT DATA= error.Worksheet_error_report_adhd&x
            OUTFILE= "C:\Users\lmendez\Documents\RMPDC\Deliverables\&year._&qtr.\
                        ADHD\error\Worksheet_error_report_&year._&qtr._adhd.xls"
            DBMS=EXCEL LABEL REPLACE;
     SHEET="Worksheet_error_report_adhd&x";
RUN;
```

The macro variable ‘x’ is used to number the reports that correspond with each workbook. In the worksheets error report, there is a worksheet for each workbook. The error report will provide the error message about the workbook.

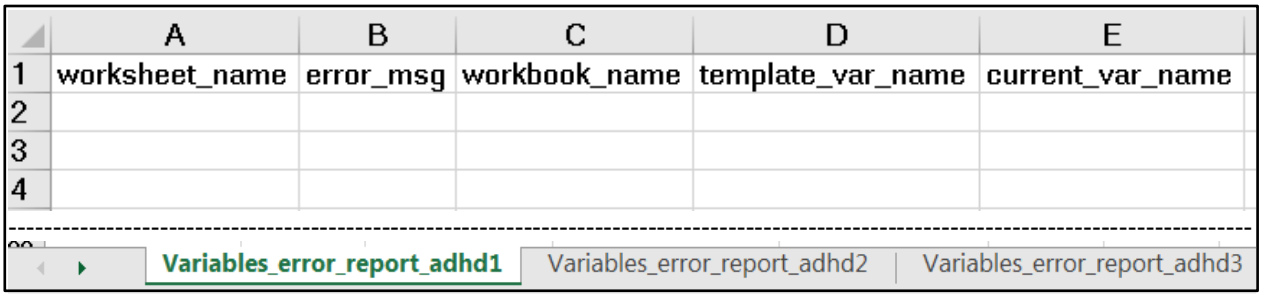
In Figure 4, you can see that one error was found in the workbook. One sheet was missing in the current deliverable.



	A	B
1	error_msg	workbook_name
2	Error - Sheet name missing from the current workbook. Sheetname missing: STATE_SUBGRP	RMPD_Patient Tracking_ADHD_NDW_2018Q2
3		
4		

Figure 4. Sample of Excel Error Report.

In the variables error report, there is a worksheet for each workbook. The error report will provide the worksheet name, error message, workbook name, template variable name, and the current variable name. Figure 5 shows that there were no errors reported for the corresponding workbook.



	A	B	C	D	E
1	worksheet_name	error_msg	workbook_name	template_var_name	current_var_name
2					
3					
4					

Variables_error_report_adhd1 | Variables_error_report_adhd2 | Variables_error_report_adhd3

Figure 5. Sample of Excel Error Report for Variables.

CHECKING THE VALUES (THE DATA) OF THE WORKSHEETS (DATASETS)

Checking the data is more straight forward. A macro variable was created, using the same methods as described above for all the worksheet/dataset names. The macro variable was used in conjunction with a macro to execute a data step multiple times to check all the data within a worksheet/dataset.

The following is an excerpt of the SAS code:

```
 /*-------------------------------------------------------------------------*/
 /* Value checks for Worksheets                                             */
 /*-------------------------------------------------------------------------*/
data qc.qc_STATE_SUPERGRP (keep = st UNPROJ_PATIENT_CNT PROJ_PATIENT_CNT UNPROJ_TRX_CNT
                                  PROJ_TRX_CNT error_msg_ST error_msg_prod);
        length error_msg_ST $125. error_msg_prod $125. ;
        set cadhd.STATE_SUPERGRP;

        row = _n_;

        /* set error flags */
        if PRODGROUP = ' ' then PRODGROUP = 'blank';

        /* Error flags will be set if there are blank rows in the file */
        if st = 'XX' OR st = ' ' then st_error_flag = 1;
        if prodgroup = 'XX' OR st = ' ' then prod_error_flag = 1;

        /* write out error messages */
        if st_error_flag = 1 then error_msg_ST = 'Variable STATE has a value of XX or is blank in worksheet ' ||
                                                 'STATE_SUPERGRP' || ', in row# ' || row;
        if prod_error_flag = 1 then error_msg_prod = 'Variable PRODGROUP is blank in worksheet '
                                                     || 'STATE_SUPERGRP' || ', in row# ' || row;

        /* delete observations that do not have errors */
        if st_error_flag = . then delete;
    run;
```

Similar code was written to check the products within a workbook; however, a pre-loaded template was used to ensure the correct products were in the correct worksheet/dataset. A macro was used, along with a data step, and a PROC SQL step to compare product names in the pre-loaded template with the product names of the current data.

An exception report was created for the values check. This method was discovered after the code was written for the previous worksheets and variables check. Due to time constraints the code was not changed, but this is an area that needs to be consistent and enhanced. For these exception reports, only MS Excel workbooks were created for each worksheet only if any errors were found.

The following SAS Code creates the Excel workbooks. A workbook is only created if a dataset exists. If no errors are found a dataset wasn't created; therefore, an Excel workbook would not be created.

```
'
/*-------------------------------------------------------------------------------*/
/* export only those datasets that have error messages                           */
/*-------------------------------------------------------------------------------*/
options mprint;
%macro export1;
    %do i = 1 %to &n6;
    %let var = %scan(&snamlist6,&i,*);

    /* Init check flag */
    %let dsempty=0;

    /* Check for empty data */
    data _null_;
    if eof then
        do;
            call symput('dsempty',1);
            put 'NOTE: EOF - no records in data!';
        end;
    stop;

    set qc.&var end=eof;
    run;

    %macro export2;
    %if &dsempty. %then
        %do;
         %put WARNING: Export step skipped - no output records.;
        %end;
    %else
        %do;
        /*if not empty then execute the export proc*/
        PROC EXPORT DATA= qc.&var
            OUTFILE= "C:\Users\lmendez\Documents\RMPDC\Deliverables\&year._&qtr.\
                      ADHD\error\Values_error_report_&year._&qtr._adhd&x..xls"
            DBMS=EXCEL LABEL REPLACE;
        SHEET="&var";
        RUN;
        %end;
    %mend export2;

   /* invoke macro */
   %export2;

   %end;

%mend export1;

/* invoke macro */
%export1;
```

DELETE WORKING DATASETS

Many macros are used to create many datasets in the process of checking one workbook. To ensure there is enough space in the SAS session, PROC Datasets is used to clean up the libraries used in the program.

The following is the SAS code to clean up the libraries using the "kill" option.

```
/*----------------------------------------------------------------------------*/
/* Clean up work and perm library                                             */
/*----------------------------------------------------------------------------*/
proc datasets lib = cadhd nolist kill;
quit;
run;

proc datasets lib = work nolist kill;
quit;
run;

proc datasets lib = error nolist kill;
quit;
run;

proc datasets lib = qc nolist kill;
quit;
run;
```

CONCLUSION

When faced with a challenging task, that can initially be overwhelming, break down the process in steps and solve one problem at a time. Doing research online may help provide different solutions. Don't be afraid to code your program and do some steps that are not as efficient, or the "down and dirty" way. You can always enhance your program for efficiency when you have more time.

REFERENCES

Brill, I. & Eberhardt, P. March 2006. "An Introduction to SAS Dictionary Tables". Proceedings of the SAS Users Group International 31 Conference (SUGI 31). San Francisco, CA. Paper 259-31. Tutorials.

DelGobbo, V. March 2006. "Creating AND Importing Multi-Sheet Excel Workbooks the Easy Way with SAS". Proceedings of the SAS Users Group International 31 Conference (SUGI 31). San Francisco, CA. Paper 115-31. Hands on Workshops.

Heaton, Edward. "Reading Excel Workbooks". Proceedings of SAS Global Forum 2007. Orlando, FL. Paper 119-2007. Hands-on-Workshops.

Heaton, Edward. "So, Your Data Are In SAS?". Proceedings of SAS Users Group International 31 Conference (SUGI31). San Francisco, CA. Paper 020-31. Applications Development.

Hemedinger, Chris. "Using LIBNAME XLSX to read and write Excel files". The SAS Dummy Blog. May 20, 2015. Available at https://blogs.sas.com/content/sasdummy/2015/05/20/using-libname-xlsx-to-read-and-write-excel-files. Last accessed July 2018.

King, Kevin. "Comparing 2 SAS Data Sets: An Alternative to Using PROC COMPARE".

Thompson, Stephanie R. "Putting SAS Dataset Variable Names into a Macro Variable". https://analytics.ncsu.edu/sesug/2006/CC01_06.PDF.

Wang, Mindy. “Bring Excel file multiple sheets to SAS”. Proceedings of the Northeast SAS Users Group Conference. Coder’s Corner. 2012.

ACKNOWLEDGMENTS

The authors want to thank Lex Jansen for continuing to publish SAS Proceedings from 1976 to present. The website searches 33,193 papers (as of the date this paper was uploaded to the Proceedings – an increase of approximately 1500 papers in the year since this paper was first prepared) and is a wealth of information for SAS Users. https://www.lexjansen.com

CONTACT INFORMATION

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

Lisa Mendez
sasebmendez@gmail.com

Andrew T. Kuligowski
kuligowskiconference@gmail.com