PharmaSUG 2022 - Paper DV-173

Copy comments: A Dynamic solution to solve the age-old need

Joseph Cooney, OPKO Health, Inc.

ABSTRACT

Over the years, I have come to know that just about every Data Management team needs Copy Comments functionality. Whether it be achieved through fancy applications, Excel macros, or SAS programs, one thing is always certain, DM needs their comments carried from one output to the next. With nearly 15 years of Clinical Data Programming experience spread across multiple employers and sponsors, I have seen what works, and simply, what does not work. I have come up with a solution that trumps each of the methods that I have seen used in the past and requires only Microsoft Excel and SAS access with limited programming experience. I have created a SAS Copy Comments macro that when executed prior to exporting data, will dynamically import a listings previous output, compare to current final data set, and copy DM comments. Since this macro is run at the end of each program and therefore, the final data set is available, the proc contents can be used to dynamically generate the import statement for the previous output to ensure an exact match in variables and formatting. This ensures that same records will always compare, and most importantly, DM comments will be copied. Lastly, since the macro is dynamic, it can be implemented into your automated listings refresh process with zero need for manual manipulation.

INTRODUCTION

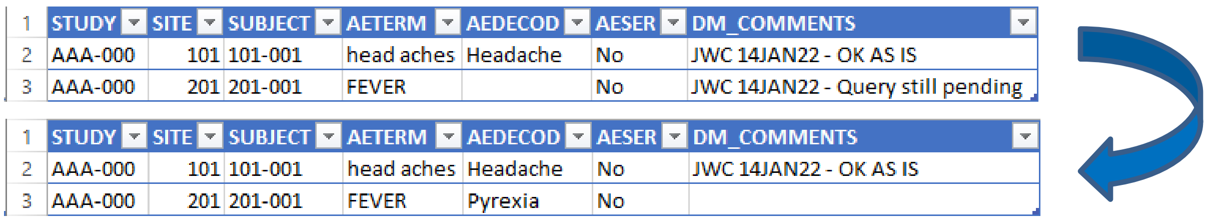
Data cleaning can be a cumbersome and time consuming task. SAS data review listings are a very helpful tool, but even then, the review can be inefficient when Data Management must process many listings and reports, sometimes daily, and some containing hundreds of records of output. Each individual record must be addressed and often, DM is required to record a note on the output record of query status, resolution, or just a general comment to assist in overall review. Unfortunately, once each listing is refreshed, DM must once again re-enter each comment. Often, records within the individual listings remain exactly the same from one scheduled refresh to the next. Studies with higher volumes of data often produce findings that have already been addressed, however, will remain to be data inconsistencies, such as when protocol deviations are reported for the discrepancy. Going back through those findings is a very time consuming task that has no benefit to the overall data review process. So, how do we make this process more efficient?

COPY COMMENTS: WHAT IS IT, AND WHY DO WE NEED IT?

It is very common for reconciliation output records to remain the same over multiple refreshes. For exception output, this can be due to the delays in queries being answered by site and reflected in source data, or recorded protocol deviations that will remain the same and therefore the specific record can be documented as a known issue and ignored. Sometimes reporting output contains clean records that only need to be reviewed once and can be filtered out of proceeding refreshes. Either way, it is clear that a solution is needed to copy Data Management comments from one listing output to the next so that the DM can concentrate on only new and updated records.

Now, here is the good news, there are multiple way to handle this issue programmatically using SAS or Excel macros. Code can read in the old listing output, compare to the new, and carry the DM comments for all records that match exactly. This functionality saves the cleaning team a great deal of time because now they can focus on only those records that are either new or updated and do not have a copied comment associated.

Figure 1 below demonstrates the expected copy comments functionality. The top table is an older dump output of select variables in the Adverse Events data set. The bottom table is the new output for the same Adverse Events dump. As you can see, record 2 remained exactly the same when comparing the old output to the new, and therefore, the DM_COMMENTS text was copied. Since record 3 was updated to add the DECOD value for FEVER in the new output, the DM_COMMENTS text was dropped, which would indicate this is a new or updated record (updated in this case).



	STUDY	SITE	SUBJECT	AETERM	AEDECOD	AESER	DM_COMMENTS
2	AAA-000	101	101-001	head aches	Headache	No	JWC 14JAN22 - OK AS IS
3	AAA-000	201	201-001	FEVER		No	JWC 14JAN22 - Query still pending

	STUDY	SITE	SUBJECT	AETERM	AEDECOD	AESER	DM_COMMENTS
2	AAA-000	101	101-001	head aches	Headache	No	JWC 14JAN22 - OK AS IS
3	AAA-000	201	201-001	FEVER	Pyrexia	No	

Display 1. Example expectation of copy comments functionality

WHAT SOLUTIONS HAVEN'T WORKED IN THE PAST AND WHY?

I have spent my entire professional career in the clinical data programming field specializing mostly in SAS data review listings. In those fifteen plus years, I have worked on many projects, for many sponsors, and for many study teams. Each different situation seemed to have its own solution to the copy comments requirement. Each option had its pros, and cons, but ultimately, I knew that I could create something better. Let's review some of the solutions that I have come across over the years.

MANUAL COMPARE AND COPY COMMENTS USING SAS OR EXCEL

This solution requires a programmer or DM user to manually compare each new output against the previous refresh and copy comments if applicable. This can be achieved using SAS by importing each output, mapping, and comparing to the previous output file or data set. DM can also produce the desired result by using VLOOKUP in Excel. Although this solution will yield desired results most of the time, it is extremely time consuming, problematic, and nearly impossible to document for audit purposes. And for those reasons, I'm out!

EXCEL MACROS SUPPORTED BY NON-PROGRAMMERS

An Excel macro can be written to compare two spreadsheets, whether single or multi-tabbed, and copy all applicable comments for records that are a 100% match. End user simply executes the excel macro, then once prompted, enters the old output, new output, and final output destination path. Although this solution may sound ideal, it has proven overtime to be very unstable. For starters, the macro constantly fails due to hidden rows and columns, filters, and other standard Excel functionality required by the data cleaning team. These failures are deemed user error and therefore the macro development team refuses to support. Unfortunately this issue has become very common which ultimately creates strain on the programming and cleaning team's relationship. And for those reasons, I'm out!

MY COPY COMMENTS SOLUTION

Having ample experience working with multiple copy comments solutions and exposure to the end user experience, I came to know exactly what works, and unfortunately, what does not. I was able to then use this knowledge to design and implement my own copy comments macro that not only capitalizes on all of strengths from previous solutions, but also eliminates each of the weaknesses. Each key point below drove the requirements for my copy comments solution:

1. Reusable dynamic code programmed in SAS
2. Fully supported by programmers
3. SAS Macro's to enable automation
4. Executed prior to printing listing output
5. Zero manual manipulation required

HIGH LEVEL CONCEPT

The SAS code is executed just prior to writing your report or listing output to Excel and just after the final output data set is created. The final data set contents can then be used to derive the import statement for the previous output dynamically. This not only saves extra steps and time needed to read in both the old and new outputs to compare and copy comments after the listing in refreshed, but also enables the solution to be dynamic, and ensure the old and new data set structure will match exactly. Once the previous output is read into SAS, it can then be sorted and compared to the current final data set output, and the comments from the previous file are merged onto the current file for all records that have remained exactly the same. Now you have the current output, with comments copied, ready to output for the end user. The solution is fully supported by the programming team, and the final output already includes the previous outputs comments prior to delivering to the end user.

CODE REVIEW

Macro Parameters

Let's start with the macro parameters. First, we need the output location where the previous output with DM_Comments is stored. Next, we will need the listing number as provided by the SAS listings specification. Lastly, the final data set name prior to outputting to Excel is required.

```
/*Params*/

/*outputLoc - Location that prior output with DM_Comments is located*/
/*listingNumber - Standard listing number from spec*/
/*finalName - Final SAS listing data set name*/

%macro copyComments(outputLoc, listingNumber, finalName);
```

Macro Call

Now let's take a look at an example of the macro call.

```
/*call the copy comments macro*/

%include "G:\SAS\MACROS\copyComments.sas";

%copyComments(G:\SAS\TA\COMPOUND\STUDY\LISTINGS\OUTPUT\COMMENTS, /*comments
files location*/
              AE1, /*listing number from output and spec*/
              AE1); /*final data set name*/
```

Load Macro Variable Lists With Final Data set Contents

Once the macro is called, we will then query the sashelp.vcolumn table to load multiple macro variable lists with the final data set contents like format, informat, and length.

```
/* Code to load multiple macro variables with content lists */


proc sql noprint;

 create table vcolumn_ as
 select *, case when TYPE = 'char' then '$'
                       else ''
               end as type1,
               case when TYPE = 'char' then '$'
                       else 'X'
               end as type2,
               case when missing(format) then 'X'
               else format
               end as format2,
               case when missing(informat) then 'X'
                       when informat like '_._' then 'best.' /* Added to
resolve import issue with decimal informats*/
               else informat
               end as informat2
 from sashelp.vcolumn
 where libname = "WORK" and memname = upcase("&finalName.");

 select NAME into :vars separated by '*'
 from vcolumn_;

 select NAME into :sort separated by ' '
 from vcolumn_;

 select compress(TYPE1||put(LENGTH, best12.)) into :varlentype separated by
'*'
 from vcolumn_;

 select informat2 into :infrmt separated by '*'
 from vcolumn_;

 select format2 into :frmt separated by '*'
 from vcolumn_;

 select TYPE2 into :vartype separated by '*'
 from vcolumn_;

 select count(NAME) into:varcount
 from vcolumn_;

 quit;
```

Load Original Sort Order into Macro Variable

By default, the old and final data sets will be sorted by _ALL_ in order to merge and compare records and therefore, we will need to re-sort the final output with comments by the original sort order found in the sashelp.vcolumn data set.

```
/* Code to load sortorder macro variable with variable sort list*/

proc sort data=vcolumn_ out=vcolumn_sort;
 by sortedby;
 where sortedby ne 0;
run;

data _null_;
```

```
 length retain_var $1000;
 retain retain_var ' ';
set vcolumn_sort end=eof;
 if _n_ = 1 then retain_var = strip(name);
  else retain_var = strip(retain_var)||' '||strip(name);
 if eof then call symput('sortorder', retain_var);
run;

%if %symexist(sortorder) %then %put &sortorder;
```

Import Previous Listing Dynamically

Now it is time to import the previous listing output containing comments, in this case, a CSV file. Since we have a list of variables, informats, formats, and lengths, we can handle this dynamically. The &csvname macro variable in the infile statement can be assigned individually for each listing or wild carded based on a pre-defined list or comment folder contents.

```
/* Import csv file with comments based on previous steps derived macro
variables*/

data WORK.COMMENTS_CSV    ;
%let _EFIERR_ = 0; /* set the ERROR detection macro variable */
infile "&outputloc.\&csvName." delimiter = ',' MISSOVER DSD lrecl=32767
firstobs=8;

%do i=1 %to &varcount;
%let var=%scan(&vars,&i,*);
%let lentype=%scan(&varlentype,&i,*);
%let infmt=%scan(&infrmt,&i,*);

      %if  %substr(&infmt,1,%length(&infmt)) ne X %then %do;

            informat %substr(&var,1,%length(&var))
%substr(&infmt,1,%length(&infmt)) ;

      %end;

      %else %do;

        informat %substr(&var,1,%length(&var))
%substr(&lentype,1,%length(&lentype)). ;

      %end;

%end;
      informat DM_Comments $2000.;

%do i=1 %to &varcount;
%let var=%scan(&vars,&i,*);
%let lentype=%scan(&varlentype,&i,*);
%let fmt=%scan(&frmt,&i,*);

      %if  %substr(&fmt,1,%length(&fmt)) ne X %then %do;
```

```
            format %substr(&var,1,%length(&var))
%substr(&fmt,1,%length(&fmt))  ;

      %end;

      %else %do;

            format %substr(&var,1,%length(&var))
%substr(&lentype,1,%length(&lentype)). ;

      %end;

%end;
    format DM_Comments $2000.;

      input

%do i=1 %to &varcount;
%let var=%scan(&vars,&i,*);
%let type=%scan(&vartype,&i,*);

      %if %substr(&type,1,%length(&type)) ne $ %then %do;

            %substr(&var,1,%length(&var))

%end;
%else %do;

     %substr(&var,1,%length(&var)) %substr(&type,1,%length(&type))

%end;
%end;

       DM_Comments $;

;
if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro
variable */
run;
```

Sort and Compare Old and New Output

We now have both the final data set from the SAS listing, and the previous output that contains the target comments to be copied. We can now go ahead and sort both by _ALL_ and then merge the data sets while keeping all records in the new final data set and copying the comments from the old output when the entire record matches exactly. This final data set with comments can now be exported and delivered to the end user.

```
/*code to compare base data (current final work SAS data set) and compare
data set (imported CSV file with comments)*/

proc sort data=&finalName.; by &sort.; run;
proc sort data=comments_csv; by &sort.; run;
```

```
data &finalName._comments;
merge &finalName.(in=a) comments_csv(in=b);
by &sort;
if a;

%if %symexist(sortorder) %then %do; /* Check for sort order, if found, then
sort, if not, do not */

 proc sort; by &sortorder.; run; /* Dynamically sort based on the final data
set sort order */

%end;

%else %do;

 run;

%end;
```

LESSONS LEARNED

Since this macro was trial and error and went through multiple productions versions, there was quite a bit of lessons learned. Some of which I would like to go through below.

To CSV Or Not To CSV, That Is The Question!

Depending on your SAS licenses, you may have the choice in output formatting. I chose CSV based on licensing limitations. You may choose to use XLS or XLSX. In my experience I have always found CSV to be more stable when importing files into SAS data sets. What I did not anticipate was how Excel manipulates CSV files when opened and saved in Excel and then read into SAS. Be careful, this manipulation can lead to records not comparing exactly and loss of DM_Comments. Each of the below will need to be anticipated and the final data set will need to be updated by either a macro or at the listing level prior to calling the copy comments macro. Let's take a look at each example:

- Dates will be reformatted and overwritten.
  - Dates like "yyyy-mm-dd" will be reformatted as "mm/dd/yyyy"
- Leading zero's will be removed from dates, time, and other numerical values
  - Dates like 08/12/2022" will be updated to "8/12/2022"
  - Time like "08:30" will be updated to "8:30"
  - Site numbers like "001" will be updated to "1"
- Incomplete dates are at risk of auto imputation.

Other Miscellaneous Rules

The below list of rules were also important to ensure the CSV file was imported correctly and will match the current final data set exactly:

- Decimals are still trial and error and will require some testing
- Ensure date9. formatted dates also have informat of date9.

- Date9 format works for dates, but incomplete dates will need to be imputed.
- When deriving variables, ensure informat is also set.
- Use standard output naming conventions and formatting.
- Sort the final data set prior to running the macro, the same sort order will be retained after merging with comments data set.

Automation

This macro in conjunction with each of the suggestions below can enable full automation of the SAS listings process:

- Standard folder structure
- Standard output naming conventions and formatting.
- Standard output macro.
- Batch files and driver programs to call each individual SAS Listing
- Dynamic code to loop through a folder containing many CSV's or an Excel spreadsheet with many tabs to select the appropriate output with comments to read in and compare to final data set
- Code to zip all outputs if applicable.
- Automated log checker

CONCLUSION

My favorite part of this profession is solving problems and helping others be more efficient as a result of my own programming. I knew everything that this solution could, and could not be (because I've seen it and lived it for years). I truly believe my take on this problem that just about every DM teams suffers from outperforms all solutions I have tried in the past. It's the combination of the ability to automate the CSV import process and anticipate the new data set will match the old data set in structure exactly that does it for me. I feel strongly about this process being supported by the programmer as opposed to the end user dependent on an unstable Excel macro or manual VLOOKUP. Ultimately, my goal is to help others and maintain a positive relationship between programmers and study team and I know my copy comments solutions does just that.

CONTACT INFORMATION

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

Joseph Cooney
OPKO Health, Inc.
JCooney@opko.com