Paper CT06

A SAS solution to compare folders instead of single files

Guido Wendland, Cytel, United States

ABSTRACT

Comparing SAS datasets is a common task when double programming ADaM, SDTM, or TFL files. From a project/study lead perspective it is desirable to collect all the comparison results in a consolidated fashion e.g. when checking the results of double programming or comparing two different data snapshots.

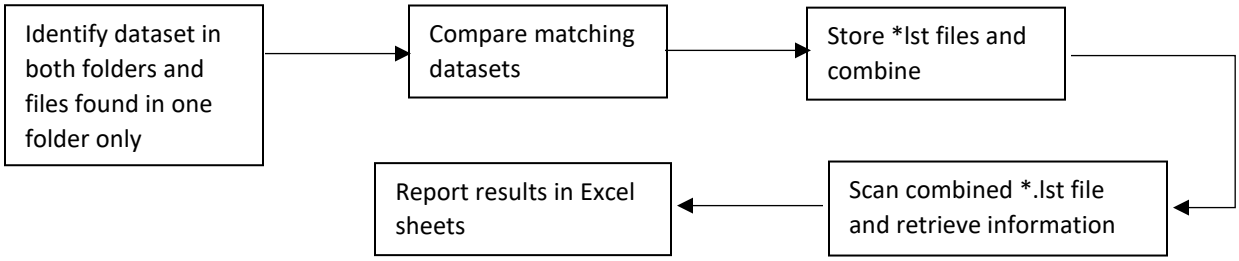
This paper presents

- programming solutions to various challenges (such as different id variables and non-matching file names across the two folders)
- how the complete scanned proc compare output is summarized into a consolidated Excel report with 2 sheets whilst additionally storing the combined proc compare output to elaborate on details.

INTRODUCTION

The paper presents a SAS macro to compare SAS dataset across two folders. The program runs PROC COMPARE on all matching data set pairs and combines the proc compare output in one *.lst file.

Results are then summarized in an Excel report that provides the overview of all data comparisons.



Different data types (Raw, SDTM, ADaM, TFL (QC files)) can be compared in a variety of potential situations:

1. To check if and what data has changed across two different data snapshots
2. To monitor the file progress over time
3. To validate multiple datasets

The purpose of this paper is to describe the key features of the macro and its main macro parameters and to show how particular challenges were addressed.

IDENTIFY DATA SETS FOR COMPARISON

There are a few macro parameters that control the location and the datasets to compare. Data sets to be compared must be located in librefs specified in macro parameters LIBNAME1 and LIBNAME2, which are associated with storage locations before the macro is called. A list of space-separated librefs can also be provided but files to be compared must be in different librefs.

The macro determines the data sets in the librefs via a proc datasets statement and by default compares the data sets with the same name unless one of the two macro parameters PREFIX or MATCHFILE is specified. With PREFIX a prefix can be specified for the name of the data sets in the second libref. The parameter MATCHFILE covers cases where the dataset pairing is less straightforward and specifies a metadata set containing the variables “from” and “to” to define the pairs to be compared.

Furthermore, the data sets in the first libref can be selected via macro parameters INCLUDE or EXCLUDE. In this case, users would use only one of these two parameters that lists the data sets to be included or excluded. Multiple data sets can be addressed by a colon (:) suffix.

MODIFY COMPARISON DETAILS

PROC COMPARE comes with a couple of options and statements that modify the output and the comparison. The parameter OPTIONS lists the preferred options which will be used in every single comparison. The parameter SELECT is a further specification of the input dataset and would usually be used to provide a where-condition or a keep/drop statement which is applied to all the base and all the comparison data sets. Thus, it is possible to only compare the data for one or more subjects or study centers. If a keep/drop statement is specified in the SELECT parameter then only the variables which are found in the data are dropped or kept and the other variables are ignored in that statement.

One of the key features of the compare procedures is to distinguish id variables and comparison variables. When dealing with standardized clinical data such as SDTM and ADaM the list of potential id variables depend on which data standard is used. The macro uses a macro parameter TYPE to be able to define a standard set of 'potential' ID variables.

If TYPE= SDTM then potential ID variables by default are identified as:
`usubjid xxcat xxscat xxtestcd xxtest visitnum visit xxtptnum xxseq`

If TYPE= ADaM then potential ID variables by default are identified as:
`usubjid visitnum paramn paramcd avisitn avisit atptn atpt xxseq.`

In the lists above, "xx" will be replaced by the corresponding domain. Again, as described for the keep/drop option only variables contained in both the compared data sets are used in the id statement, hence the term 'potential' was used.

The macro parameter ID allows the user to override the default set of id variables and define a separate set of potential id variables. See the following chapter to see how this is technically implemented.

SOME PROGRAMMING CHALLENGES AND SOLUTIONS

This chapter addresses some of the main challenges and show code to tackle them.

COMPARE SELECTED DATA SETS/VARIABLES/RECORDS

The technique used to pick elements from a list is shown below:

```
%let _index=1;
%do %until (%qscan(%bquote(&LIBNAMES.),&_index.,%str( )) = %str() ) ;
    %let LIBNAMES&_index.=%qscan(%bquote(&LIBNAMES.),&_index.,%str( ));
    %let _index=%eval(&_index.+1);
%end;
```

This code is used to process the data sets found in multiple librefs. A single libref from the list can then be specified via `&&LIBNAMES&_index.` The same technique is also used to pick single variables from the macro parameter INCLUDE and EXCLUDE.

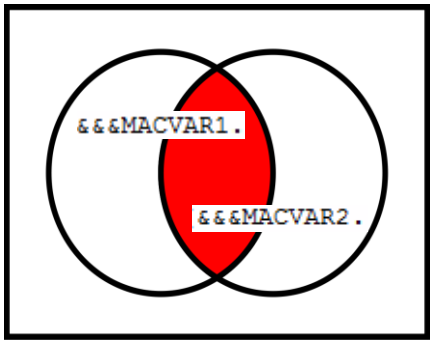
PAIRS OF FILES USE DIFFERENT SET OF VARIABLES (ID VARIABLES, VARIABLES TO BE KEPT ETC.)

When defining a set of 'potential' ID variables for multiple paired data comparisons the same list will not usually apply to all data sets. Let's assume we compare SDTM data sets and we use the standard set of ID variables as laid out above. These are variables usually occurring in the findings domain class but not in all other domain classes, e.g. the variable VISITNUM is not in the events domain class. Furthermore, it is possible that a variable is only contained in one of the two compared data sets.

That means that the list of ID variables needs to be adjusted for each data set comparison. In the first of two steps we create two macro variables, one with the list of variables contained in the base data set (&LIST1) and one with the variables contained in the compare data set (&LIST2). This is implemented by scanning through the list of variables and adding the variables successively to a new list only when the selected variable is contained in the corresponding data set.

In the second step we determine the intersection of the two macro variables &LIST1 and &LIST2 to identify identical variables in both variable lists. The list of variables in the intersection is then used in the ID statement for a comparison. Following a modular approach an inner macro is used to determine the intersection of the variable lists. The macro variables &LIST1 and &LIST2 need to be provided as value of two macro parameters – MACVAR1 and MACVAR2 – of the inner macro.

In this situation we have a nested macro resolution that requires the use of 3 ampersands (&&&) to retrieve the variable lists as illustrated in the example below. It shows the call of %put statements and the corresponding resolution from the log of the program.




```
%let macvar=test;
%let test=var1 var2 var3;

%put 1) &macvar.  ;  1) test
%put 2) &&macvar. ;  2) test
%put 3) &&&macvar.;  3) var1 var2 var3
```

To find the intersection we scan simultaneously through the two lists of variables (&&&MACVAR1. and &&&MACVAR2.) and add the variable to a newly generated list when a match is found. Programmatically this can be implemented via two nested %do loops as shown below:

```
 %let _counti=%sysfunc(countw(&&&MACVAR2.));
 %let _count1=1;
%do %until (%qscan(&&&MACVAR1.,&_count1.) = %str( ) );
   %let _count2=1;
   %let _flag=0;                        /** Reset _flag for next macro variable of &&&MACVAR1. **/
    %do %until (%qscan(&&&MACVAR2.,&_count2.) = %str( ));
        %if %upcase(%qscan(&&&MACVAR1.,&_count1.))=%upcase(%qscan(&&&MACVAR2.,&_count2.))
                                                                %then %let _flag=1;
          %if &_count2=&_counti. %then %do;              /** Last element of list reached **/
            %if &_flag ne 1 %then %let _notfd=&_notfd %upcase(%qscan(&&&MACVAR1.,&_count1.));
            %if &_flag eq 1 %then %let _found=&_found %upcase(%qscan(&&&MACVAR1.,&_count1.));
          %end;
        %let _count2=%eval(&_count2.+1);
    %end;
   %let _count1=%eval(&_count1.+1);
%end;
```

IGNORE CERTAIN FORMATTING DIFFERENCES

Especially when it comes to comparing data sets reflecting the TFL contents, users might want to concentrate on the value comparison and not so much on the formatting of values. Therefore, the macro optionally allows the conversion of certain parameters and the compression of spaces and non-printable and special characters (NBSP) such as line breaks or special hex character codes required for PDF output such as (“A0”x). This can be implemented via PERL regular expressions (prxchange) or by applying the compress function and extending it to all kinds of special characters and control characters (blank, horizontal tab, vertical tab, carriage return, line feed, form feed, and NBSP ('A0'x, or 160 decimal ASCII)) via the compress function modifiers S and C. In the below expression cols [i] stands for the i-th array element, i.e. the i-th character variable:

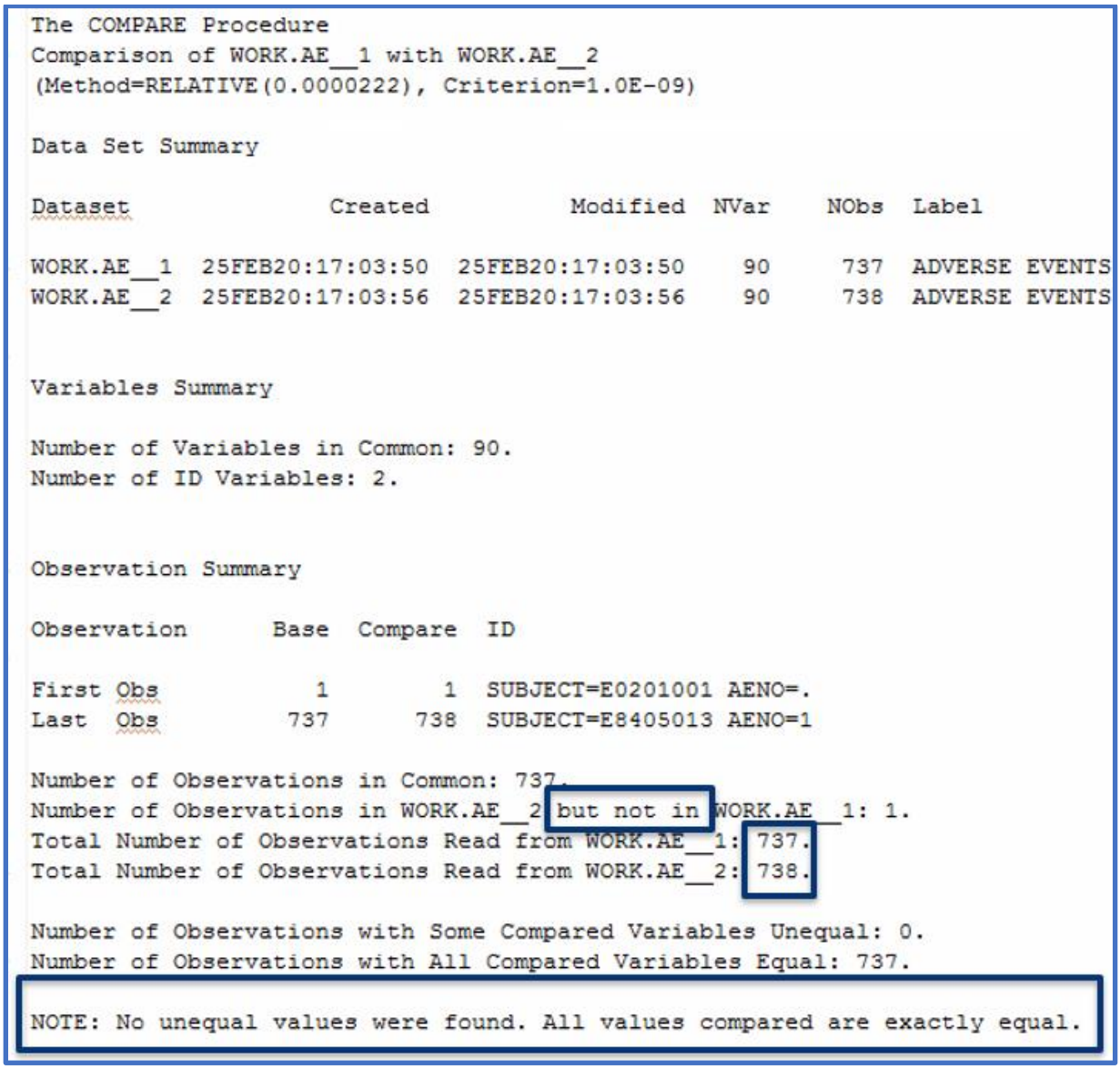
```
prxchange('s/[\x03]n|[\xA0]//', -1, cols[i]);
 cols[i] = strip(compress (cols[i]," ","SC"));
 cols[i] = strip(translate (cols[i],"-","-"));
```

STORE *LST FILES AND COMBINE

The proc compare is run for each pair of datasets applying the appropriate id statements and the specified options as outlined before. This is embedded into two proc printto statements, one to define the printto location for the output and one to reset the destination for the SAS procedure output to the default. Thus, all proc compare output is combined in one output *.lst file.

SCAN COMBINDED *.LST FILE AND RETRIEVE INFORMATION

Subsequent to the combined *.lst file being generated, the second part of the program scans this file to retrieve information which is to be displayed in the report. The picture below shows an example proc compare output (restricted to one data set pair) with some of the information used for the report being highlighted:



```
The COMPARE Procedure
Comparison of WORK.AE__1 with WORK.AE__2
(Method=RELATIVE(0.0000222), Criterion=1.0E-09)

Data Set Summary

Dataset              Created          Modified  NVar    NObs  Label

WORK.AE__1  25FEB20:17:03:50  25FEB20:17:03:50    90     737  ADVERSE EVENTS
WORK.AE__2  25FEB20:17:03:56  25FEB20:17:03:56    90     738  ADVERSE EVENTS


Variables Summary

Number of Variables in Common: 90.
Number of ID Variables: 2.


Observation Summary

Observation      Base  Compare  ID

First Obs           1        1  SUBJECT=E0201001 AENO=.
Last  Obs         737      738  SUBJECT=E8405013 AENO=1

Number of Observations in Common: 737.
Number of Observations in WORK.AE__2 but not in WORK.AE__1: 1.
Total Number of Observations Read from WORK.AE__1: 737.
Total Number of Observations Read from WORK.AE__2: 738.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 737.

NOTE: No unequal values were found. All values compared are exactly equal.
```

More specifically the following terms are searched for:

```
index(text,"ERROR:")>0
index(text,"Conflicting Types")>0
index(text,"but not in")>0 and index(text,"Observations")>0
index(text,"but not in")>0 and index(text,"Variables")>0
index(text,"First Un")>0
index(text,"not found in")>0
index(text,"Number of Observations in Common: 0.")>0
index(text,"Differing Attributes")>0
index(text,"Duplicate Observations")>0
index(text,"All values compared are exactly equal")>0
```

The relevant information is retrieved from the text that follows the key words. An index is provided for each type of information which will later be used in a proc transpose to constitute the columns of the report.

In addition, a tag in the header is applied to distinguish the single outputs from the combined output file to be able to assign the found information to each pair of the compared data sets.

REPORTING

One macro parameter, OXLSXPATH, is reserved for the location and name of the output file. The report is generated via an ODS Excel statement. ODS Excel was chosen because it generates a user-friendly report in Excel with all the known features such as autofilter, rowheader, sheet_names. Furthermore, value-depending color coding was implemented in the proc report statement:

```
compute allequal;
  if allequal="Yes" then call define(_col_, "style", "style=[background=green]");
 endcomp;
```

The report consists of two sheets/tabs: The tab “Proc compare files summary” lists files that were compared and files that could only be found in one folder

	A	B	C
1	Data sets compared	Data sets in base but not in compare	Data sets in compare but not in base
2	AE,AEADDOM,AESI,AESIA,ASMPERF,AZAWSAE,BLEVNT ,BMRESON,CAPRX,CAPRXR,CAPRXROM,CM,CNTLIN ,COMMENTS,CONPRO,CONSENT,CONSWD,CTSCAN ,D9106C00001_DUMMY_KITS,D9106C00001_RAN_DUMMY _DATA,DEATH,DISEXOM,DM,DOSDISC,DS,EG,ENROL,EX ,FDGSCAN,FDGSUV,GROUP,HISS,HOSPAD,IE,ILDIS,LB ,LIVERDI,LIVERRF,LIVERSS,METADATA,MH,OVERDOSE ,PATHGEN,PATHGOM,PATHROM,PE,PFU,PREG ,PREGREP,PSTAT,PULMOA,RECIST1,RECIST2,REVPRDI ,SERAE,SPCBEDB,SPCBEDT,SPCGB,SPCPKB,SURG ,SURVIVE,SU_NIC,VISIT,VS	AESI_D,CDISC_ TERMINOLOGY, LBREF,LIST,LIV RF ,SDTM_METAD ATA,TADEF,TE DEF,TIDEF,TSD EF ,TVDEF,UNITCO NV	
3			

Proc compare files summary | Proc compare results

The second tab “Proc compare results” contains the comparison details:

	A	B	C	D	E	F	G
1	ID Variables	Dataset	Obs.in Commo	not matching Observations	Duplicate obs	All values ex. equal	Observation numbers
2	subject aeno	Ae		in __cleaned but not in __final:1		Yes	737 vs 738
3	subject aeno	Aeaddom				Yes	738 vs 738
4	subject aeno	Aesi			Number of Duplicate Observations found in WORK.AESI__1: 6.	Yes	50 vs 50
5	subject aeno	Aesia				Yes	5 vs 5
6	subject	Asmperf			found in WORK.ASMPERF__1: 35.	Yes	186 vs 186
7	subject aeno	Azawsae				Yes	39 vs 39
8	subject aeno	Blevnt				Yes	196 vs 196
9	subject	Bmreson			found in WORK.BMRESON__1:	Yes	285 vs 285
10	subject	Caprx			found in WORK.CAPRX__1: 6.	Yes	164 vs 164
11	subject	Caprxr			found in WORK.CAPRXR__1: 5.	Yes	167 vs 167
12	subject	Caprxrom	0				0 vs 0
13	subject	Cm			found in WORK.CM__1: 3887.	Yes	4035 vs 4035

Proc compare files summary | Proc compare results

This sheet contains the base dataset (i.e. the dataset names form the first libref) and the selected ID variables. Furthermore, it depicts one column for each of the selected information from the output and a column that compares the observation numbers. Columns will be skipped if they contain no information.

Green color coding is used to show the statement “All values are exactly equal”, red color coding highlights differences in the number of records.

CONCLUSION

The macro can be a useful tool both for the programming lead and for each single programmer for any type of SAS dataset, as an overview of the proc compare findings across multiple comparisons can be generated within a minute.

Lead programmers overseeing a project or study with multiple programmers involved might want to use such a tool to get an overview of the comparison progress. They could also use the macro call to compare modifications across received data snapshots. Other programmers might find the tool helpful to compare all assigned deliverables in one go.

Whereas the report provides high level information the combined proc compare output shows the comparison details. In most situation both files should therefore be used in conjunction with each other.

REFERENCES

1. Macro utility to compare multiple SAS data sets
http://www.scsug.org/wp-content/uploads/2013/11/Macro-utility-to-compare-multiple-SAS-data-sets-Krish-Krishnan.pdf

2. Non-Printable and Special Characters? … BYTE me!
https://www.lexjansen.com/phuse/2016/pp/PP10.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Guido Wendland
Cytel Inc.
Work Phone: +49 152 08724198
Email: guido.wendland@cytel.com