PharmaSUG 2012 - Paper CC11

Short-circuiting PROC COMPARE: Techniques for Focusing Dataset Comparisons

Tracy Sherman, ICON Clinical Research, Redwood City, CA
Brian Fairfield-Carter, ICON Clinical Research, Redwood City, CA

ABSTRACT

Electronic file comparisons are a staple of analysis dataset programming and validation, with PROC COMPARE output as the end product of most independent double-programming. This output can be frustrating to work with, particularly in early stages of program validation when discrepancies can be extensive, since PROC COMPARE is not particularly adept at matching non-discrepant records when the number of records on the 'base' and 'compare' datasets don't match (this in contrast to many 'diff' utilities used in text-file comparisons, which can often accommodate record-count differences). When confronted with record-count differences, most programmers resort to temporary sub-setting and ad hoc/throw-away blocks of code to help trim down PROC COMPARE output and focus review efforts. However, these methods follow a reasonably consistent pattern, and as such it seems plausible that more generalized techniques could be developed. This paper proposes two such methods, the first designed to highlight among key variables where record-count differences exist (with the option of trimming off record-count differences to expose underlying value-level differences), and the second intended to allow for a more manageable 'step-wise' evaluation of discrepancies, starting with the first combination of key variables where a discrepancy exists.

INTRODUCTION

Independent double-programming typically involves little or no communication between the program developer and validation programmer on the interpretation of the analysis dataset specifications. This type of programming generally contains independently written code to produce validation output and the use of an electronic means (for example, SAS Proc Compare or UNIX diff command) to compare these datasets.

PROC COMPARE can be very useful for determining value differences if the PROC COMPARE ID variables are selected accurately and the number of records being compared are equal. When there are record-count differences, even a very small number, PROC COMPARE can output a vast number of discrepancies.

For example, in the output shown below, there was a record-count difference of only 3 records and an update to one variable (LBTESTCD) that collectively resulted in ***987 discrepancies***.

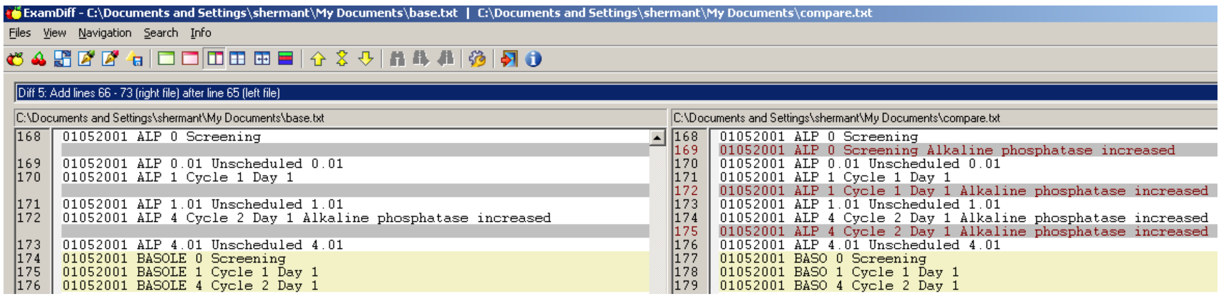
```
%let war=WAR;
%let ning=NING;

  proc compare data=base compare=compare listall maxprint=4 &war.&ning;
   id usubjid lbtestcd visitnum visit lbseq lbtox;
  run;
```

```
The COMPARE Procedure
Comparison of WORK.BASE with WORK.COMPARE
(Method=EXACT)

Data Set Summary

Dataset                 Created          Modified  NVar    NObs

WORK.BASE     19DEC11:23:08:13  19DEC11:23:08:13    34    8272
WORK.COMPARE  19DEC11:23:08:15  19DEC11:23:08:15    34    8275
```

```
Variables Summary

Number of Variables in Common: 34.
Number of ID Variables: 6.

Comparison Results for Observations

Observation 70 in WORK.COMPARE not found in WORK.BASE: SUBJID=01001001
 LBTESTCD=BASO VISITNUM=0 VISIT=Screening LBSEQ=282 LBTOX=.

Observation 71 in WORK.COMPARE not found in WORK.BASE: SUBJID=01001001
 LBTESTCD=BASO VISITNUM=1 VISIT=Cycle 1 Day 1 LBSEQ=283 LBTOX=.

Observation 72 in WORK.COMPARE not found in WORK.BASE: SUBJID=01001001
 LBTESTCD=BASO VISITNUM=2 VISIT=Cycle 1 Day 8 LBSEQ=284 LBTOX=.

Observation 73 in WORK.COMPARE not found in WORK.BASE: SUBJID=01001001
 LBTESTCD=BASO VISITNUM=3 VISIT=Cycle 1 Day 15 LBSEQ=285 LBTOX=.
NOTE: The MAXPRINT=(4,4) printing limit has been reached. No more values will be printed.


Observation Summary

Observation      Base  Compare  ID

First Obs           1        1  SUBJID=01001001 LBTESTCD=ALB VISITNUM=0 VISIT=Screening
LBSEQ=1 LBTOX=
Last  Obs        8272     8275  SUBJID=27032002 LBTESTCD=WBC VISITNUM=4 VISIT=Cycle 2 Day 1
LBSEQ=89 LBTOX=

Number of Observations in Common: 7288.
Number of Observations in WORK.BASE but not in WORK.COMPARE: 984.
Number of Observations in WORK.COMPARE but not in WORK.BASE: 987.
Total Number of Observations Read from WORK.BASE: 8272.
Total Number of Observations Read from WORK.COMPARE: 8275.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 7288.

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

Output 1. PROC COMPARE output, presenting both record-count and value-level differences

In contrast, consider the simple but complete picture of the record-count differences and value differences given by ExamDiff (Display 1). It is now apparent that there are extra visits in the 'compare' dataset for Screening, Cycle 1 Day 1, and Cycle 2 Day 1 for LBTESTCD=ALP. It is also evident that LBTESTCD=BASOLE was shortened to 'BASO' in the compare dataset.


ExamDiff - C:\Documents and Settings\shermant\My Documents\base.txt | C:\Documents and Settings\shermant\My Documents\compare.txt
Files View Navigation Search Info
Diff 5: Add lines 66 - 73 (right file) after line 65 (left file)
C:\Documents and Settings\shermant\My Documents\base.txt
C:\Documents and Settings\shermant\My Documents\compare.txt

```
168  01052001 ALP 0 Screening
169  01052001 ALP 0.01 Unscheduled 0.01
170  01052001 ALP 1 Cycle 1 Day 1
171  01052001 ALP 1.01 Unscheduled 1.01
172  01052001 ALP 4 Cycle 2 Day 1 Alkaline phosphatase increased
173  01052001 ALP 4.01 Unscheduled 4.01
174  01052001 BASOLE 0 Screening
175  01052001 BASOLE 1 Cycle 1 Day 1
176  01052001 BASOLE 4 Cycle 2 Day 1
```

```
168  01052001 ALP 0 Screening
169  01052001 ALP 0 Screening Alkaline phosphatase increased
170  01052001 ALP 0.01 Unscheduled 0.01
171  01052001 ALP 1 Cycle 1 Day 1
172  01052001 ALP 1 Cycle 1 Day 1 Alkaline phosphatase increased
173  01052001 ALP 1.01 Unscheduled 1.01
174  01052001 ALP 4 Cycle 2 Day 1 Alkaline phosphatase increased
175  01052001 ALP 4 Cycle 2 Day 1 Alkaline phosphatase increased
176  01052001 ALP 4.01 Unscheduled 4.01
177  01052001 BASO 0 Screening
178  01052001 BASO 1 Cycle 1 Day 1
179  01052001 BASO 4 Cycle 2 Day 1
```

Display 2. ExamDiff output, presenting both record-count and value-level differences

When working with PROC COMPARE output, these 'legitimate' discrepancies are often masked by 'artifact' discrepancies; when confronted with extensive discrepancies, what we tend to ask is:

1. If there are record-count differences, where exactly do they occur (are they systemic, or are they isolated to a relatively small sub-set of records)?

2. How extensive would the value-level differences be if they weren't obscured by the record-count differences?
3. If value-level differences are extensive, are there small and manageable sub-sets of the data that can help isolate specific problems in program logic, and allow for incremental revision of program code?

This paper presents two simple methods intended to address these questions, the first designed to deal with record-count differences (specifically, to identify using key variables where the differences exist, give some indication of the overall extent of the differences, and expose value-level differences obscured by the record-count differences), and the second using the iterative removal of discrepant records in order to provide insight on where to focus code-revision efforts.

AN ALTERNATIVE TO DETERMINE RECORD-COUNT DIFFERENCES

As an alternative to temporary sub-setting and ad hoc/throw-away blocks of code to determine differences, we first retrieve a few key variables from sashelp.vtable, do a simple PROC FREQ on the 'base' and 'compare' datasets, merge the FREQ output and print out records where the counts differ. Then by removing the record-count differences, the value-level differences can be better seen with the 'dummied down' PROC COMPARE output.

One of the most important things in reviewing record-count differences is in the 'extent' of the differences. For example, if we attempt to 'classify' records where these differences occur (for instance, according to something as simple as lab parameter (LBTESTCD)), will we discover that the differences occur throughout the data, or that they're isolated to one or two specific parameters? The former would probably indicate a problem in program logic, while the latter would more likely indicate an inconsistency in the raw data. In light of this, the first thing we capture is the record count of the 'base' and 'compare' datasets:

```
proc sql noprint;
  select count(*) into :baserecs from &lib..&base;
  select count(*) into :comprecs from &lib..&compare;
```

Since we want to 'classify' records where record-count differences exist, probably the most logical place to start is with the variables providing the inherent sort order for the datasets (since prior to running PROC COMPARE, the 'base' and 'compare' datasets will invariably be sorted by a common set of key variables). Note however that since the record-counts at these classification levels will be provided by PROC FREQ, we may be constrained by memory limitations; for this reason, the number of key variables is here being arbitrarily limited by the '&maxvars' parameter:

```
  create table vars as select name, sortedby from sashelp.vcolumn
    where upcase(libname)=upcase("&lib") &
          upcase(memname)=upcase("&base") & 0<sortedby<=&maxvars
    order by sortedby
  ;
  select name into :varlst separated by "*" from vars;
  select name into :varlst_ separated by " " from vars;
```

The macro variables '&varlst' and '&varlst_' now contain the ordered list of sorting variables (up to the arbitrary maximum set by '&maxvars'), delimited for use in a 'table' statement and a 'keep' statement, respectively. Record counts at the classification levels given by these key variables can now be provided via PROC FREQ, and by merging the output datasets, record counts from the 'base' and 'compare' datasets can be compared:

```
proc freq data=&lib..&base noprint;
  table &varlst / out=b_frq(keep=&varlst_ count rename=(count=b_count));
run;
proc freq data=&lib..&compare noprint;
  table &varlst / out=c_frq(keep=&varlst_ count rename=(count=c_count));
run;

data frq;
  merge b_frq c_frq;
  by &varlst_;
run;
```

Record-count differences can now be viewed by selecting records where frequency counts derived from the 'base' and 'compare' datasets are not equal:

```
title "-----RECORD-COUNT DIFFERENCES-----";
proc print data=frq;
  where b_count^=c_count;
run;
title;
```

This listing will display the key-variable combinations where record-count differences occur, and show the extent of these differences at each level:

```
        -----RECORD-COUNT DIFFERENCES-----
Obs     SUBJID     PARAMCD    VISITNUM    b_count    c_count
 22    01001001    BASO           1          .          1
 23    01001001    BASO           2          .          2
 24    01001001    BASO           3          .          1
 25    01001001    BASO          80          .          1
 26    01001001    BASOLE         1          1          .
 27    01001001    BASOLE         2          2          .
 28    01001001    BASOLE         3          1          .
 29    01001001    BASOLE        80          1          .
```

Output 2. Key-variable combinations showing record count differences

The last step is to remove records from the ‘base’ and ‘compare’ datasets having these key-variable combinations, and see what the remaining value-level differences are now that they are not obscured by record-count differences:

```
data base_(drop=b_count c_count);
  merge &lib..&base(in=_1) frq(in=_2 where=(b_count^=c_count));
  by &varlst_;
  if _2 then delete;
run;
data compare_(drop=b_count c_count);
  merge &lib..&compare(in=_1) frq(in=_2 where=(b_count^=c_count));
  by &varlst_;
  if _2 then delete;
run;

title1 "RECORD COUNT DIFFERENCES HAVE BEEN REMOVED!";
title2 "ORIGINAL RECORD COUNTS: BASE: %cmpres(&baserecs), COMPARE: %cmpres(&comprecs)";
proc compare base=base_ compare=compare_;
run;
```

Note that the second title line provides the original record counts retrieved in the first step (above), so a quick review of the listing output will show the overall extent of these differences, taking into consideration the key-variable combinations that have proven to be problematic. The complete macro, along with sample call, is provided in Appendix 1.

‘STEP-WISE’ REVIEW

To identify and work through discrepancies in sequence, we can select the first combination of key variables where a discrepancy is shown. This combination of variables is taken from the proc compare output dataset and looped through a step-wise set of programming statements to show the value-level discrepancies.

Regardless of whether or not record-count differences exist, in some cases PROC COMPARE just simply generates large numbers of discrepancies; the sheer volume can be more than a little bewildering. There is little else to do but ‘begin at the beginning’, but it’s often a clumsy process, again involving a lot of ad hoc, throw-away code. Are there specific combinations of variables, and/or sets of records that are particularly instructive? Or at a more basic level, can we select out a small set of ‘relevant’ records pertaining to the first identified discrepancy?

DETAILS OF THE FIRST DISCREPANT RECORD

In this excerpt, the objective is to identify the observation number of the first discrepant record, identify all the variables on this record that show discrepancies, and list out the key variables along with the discrepant variables on this observation for both the ‘base’ and ‘compare’ datasets. PROC COMPARE provides an output dataset where discrepancies are denoted in the form “..X..XXX.X”, where the ‘X’s indicate the specific points of departure within a given variable. The first step is to create an output dataset, which will then contain all the variables common to the ‘base’ and compare datasets, and create a list of all possible variables in which discrepancies might occur (in other words, the full list of variables on the ‘base’ dataset):

```
proc compare base=base_ compare=compare_ out=test noprint;
  run;

  proc sql noprint;
    select name into :varlst separated by " "
      from sashelp.vcolumn where upcase(libname)="WORK" & upcase(memname)="BASE_";
    select count(*) into :varn
      from sashelp.vcolumn where upcase(libname)="WORK" & upcase(memname)="BASE_";
  quit;
```

From the output dataset, we want to retain only the records showing discrepancies, and on each of these records provide a list of all variables in which a discrepancy occurs (the 'diffvars' variable). The list of discrepant variables is assembled via concatenation, by iterating through the list of all possible variables and selecting those where an 'X' appears. Similarly, records are only output where at least one variable shows a discrepancy, as indicated by an 'X':

```
data test;
  attrib diffvars length=$200.;
  set test;
  diffvars="";
  %let i=1;
    %do %until(%scan(&varlst,&i)=);

      if index(upcase(compress(%scan(&varlst,&i))),"X") then
         diffvars=trim(left(diffvars))||" "||"%scan(&varlst,&i)";

    %let i=%eval(&i+1);
  %end;
  if
     %let i=1;
       %do %until(%scan(&varlst,&i)=);
         index(upcase(compress(%scan(&varlst,&i))),"X")
           %if &i<&varn %then %do;
             or
           %end;
       %let i=%eval(&i+1);
     %end;
  then output;
run;
```

Listing this dataset will produce something like the following:

```
OBSERVATION NUMBERS FOR DISCREPANT RECORDS, AND DISCREPANT VARIABLES

 Obs diffvars        _TYPE_    _OBS_     SUBJID     PARAMCD              PARAM
   1 PARAMCD PARAM    DIF        29     ........    ....XX..  ....XXXXXXXXXXX.XXXXXXXXXX....
  ..Etc..
```

Output 3. Listing of variables showing discrepant records

In other words, 'diffvars' lists the names of all variables showing discrepancies, on any record showing at least one discrepancy, and '_OBS_' gives the observation number of the discrepant record. Taking the first record from this dataset will give an appropriate starting point for more detailed investigation, and since we have the observation number of the first discrepancy, we can use this to print the given record from the 'base' and 'compare' datasets, with the option of listing only the discrepant variables (retrieved from our 'diffvars' list) and/or key variables (sorting variables, retrieved as shown earlier in the paper). Identifying the values of key variables on this record will be useful for subsetting the 'base' and 'compare' datasets in order to dig into the root of the discrepancy.

```
%*** THIS WILL GIVE THE TOTAL COUNT OF DISCREPANCIES AT EACH ITERATION;
  proc sql noprint;
    select count(*) into :ndiffrecs from test
    ;
  quit;
```

```
    %*** KEEP ONLY THE FIRST DISCREPANCY IN THE GIVEN ITERATION;

     data test;
       set test;
       if _n_=1 then output;
     run;

     %*** CAPTURE THE OBSERVATION NUMBER OF THIS FIRST DISCREPANCY;

     proc sql noprint;
       select _obs_ into :_obs_ from test
       ;
     quit;

     title1 "=======================================================";
     title2 " -----ITERATION &_i, DISCREPANCY AT RECORD %cmpres(&_obs_)-----  ";
     title3 "=======================================================";

     title4 "-----BASE DATASET, OBSERVATION &_obs_-----";
     data base__;
       set base_;
       if _n_=&_obs_;
     run;
     proc print data=base__;
     run;

     title4 "-----COMPARE, OBSERVATION &_obs_-----";
     data compare__;
       set compare_;
       if _n_=&_obs_;
     run;
     proc print data=compare__;
     run;
```

These steps produce listing output from the ‘base’ and ‘compare’ datasets with the iteration number and dataset observation.

```
=======================================================
 -----ITERATION 1, DISCREPANCY AT RECORD 29-----
=======================================================
-----BASE DATASET, OBSERVATION 29-----

Obs     SUBJID     PARAMCD                PARAM                 LBSTRESC    LBSTRESU
 1     01001001    BASOLE     Basophils/Leukocytes (x10^3/uL)       0        x10^3/uL

-----COMPARE, OBSERVATION 29-----

Obs     SUBJID     PARAMCD           PARAM            LBSTRESC    LBSTRESU
 1     01001001     BASO      Basophils (x10^3/uL)       0        x10^3/uL
```

Output 4. Listing output with iteration number and dataset observation

RECURSION

To capture the key-variable values from the first discrepancy and trim off those records from the ‘base’ and ‘compare’ datasets, we apply a recursive macro call that repeats the process until either there are no more discrepancies, or some arbitrary maximum number of iterations has been reached.

```
%macro comp1_(lib=WORK,base=,compare=,maxvars=3);
...
%if &ndiffrecs>0 AND &_i<10 %then %do; %*** (STOP WHEN EITHER THERE ARE
         NO FURTHER DISCREPANCIES, OR WHERE A MAXIMUM NUMBER OF ITERATIONS HAS BEEN REACHED);

%*** REMOVE ALL RECORDS SHARING THE SAME COMBINATION OF KEY VARIABLES
     AS THE DISCREPENT RECORD;

     data base_;
       merge base_(in=_1) base__(in=_2);
       by &varlst_;
       if _2 then delete;
     run;
```

```
      data compare_;
        merge compare_(in=_1) compare__(in=_2);
        by &varlst_;
        if _2 then delete;
      run;


  %*** RECURSIVE CALL, TO CAPTURE THE NEXT DISCREPANCY...;

      %comp1_(lib=WORK,base=base_,compare=compare_,maxvars=3);

    %end;
  %mend comp1_;
```

(Space limitations prevent the inclusion of this macro in its entirety, but it is available as an attachment.)

CONCLUSION

Dataset comparisons can be frustrating and time-consuming, given the large volumes of output that can be generated by PROC COMPARE, and in particular given the large numbers of spurious discrepancies that PROC COMPARE will identify when confronted with record-count differences. This paper presents a few simple techniques to pin down relevant differences where record-count differences exist, and to focus review on manageable sub-sets of discrepancies. The first technique eliminates record-count differences in order to expose underlying value-level differences, and the second technique allows for 'step-wise'/recursive evaluation of discrepancies, starting with the first combination of key variables where a discrepancy exists, in order to isolate specific types of differences among the otherwise intimidating mass of PROC COMPARE output.

ACKNOWLEDGMENTS

We would like to thank ICON Clinical Research, Syamala Schoemperlen and Randi McFarland for encouraging and supporting conference attendance.

CONTACT INFORMATION

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

Name: Tracy Sherman
Enterprise: ICON Clinical Research
Address: 303 Twin Dolphin Drive, Suite 600
City, State ZIP: Redwood City, CA
E-mail: shermantracy@gmail.com
Web: www.iconplc.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.

APPENDIX 1. MACRO FOR HANDLING RECORD-COUNT DIFFERENCES

```
%macro comp_(lib=WORK,base=,compare=,maxvars=3);

  proc sql noprint;
    select count(*) into :baserecs from &lib..&base;
    select count(*) into :comprecs from &lib..&compare;
    create table vars as select name, sortedby from sashelp.vcolumn
      where upcase(libname)=upcase("&lib") &
            upcase(memname)=upcase("&base") & 0<sortedby<=&maxvars
      order by sortedby
    ;
    select name into :varlst separated by "*" from vars;
    select name into :varlst_ separated by " " from vars;
  quit;

  proc freq data=&lib..&base noprint;
    table &varlst / out=b_frq(keep=&varlst_ count rename=(count=b_count));
  run;
  proc freq data=&lib..&compare noprint;
    table &varlst / out=c_frq(keep=&varlst_ count rename=(count=c_count));
  run;

  data frq;
    merge b_frq c_frq;
    by &varlst_;
  run;

  title "-----RECORD-COUNT DIFFERENCES-----";
  proc print data=frq;
    where b_count^=c_count;
  run;
  title;

  data base_(drop=b_count c_count);
    merge &lib..&base(in=_1) frq(in=_2 where=(b_count^=c_count));
    by &varlst_;
    if _2 then delete;
  run;
  data compare_(drop=b_count c_count);
    merge &lib..&compare(in=_1) frq(in=_2 where=(b_count^=c_count));
    by &varlst_;
    if _2 then delete;
  run;

  title1 "RECORD COUNT DIFFERENCES HAVE BEEN REMOVED!";
  title2 "ORIGINAL RECORD COUNTS: BASE: %cmpres(&baserecs), COMPARE: %cmpres(&comprecs)";
  proc compare base=base_ compare=compare_;
  run;

%mend comp_;

%comp_(lib=WORK,base=base,compare=compare,maxvars=3);
```