OUT= IS IN: VISUALIZING PROC COMPARE RESULTS IN A DATASET

Prasad Ilapogu, Ephicacy Consulting Group; Masaki Mihaila, Pfizer;

ABSTRACT

Proc compare is widely used in the pharmaceutical world to validate almost every statistical output. The procedure compares and shows the differences between the contents of two SAS data sets in the ‘results window’, however, these differences appear as discrete, variable-by-variable differences. Often, programmers find themselves spending time opening base and compare datasets to locate the observations where the values differ. For large datasets, this can be a time-consuming process. This paper discusses the various options in PROC COMPARE that can make the validation process more efficient. Using the out= option in PROC COMPARE along with other options addressed in the paper will allow statistical programmers to produce an output dataset that not only provides the differences between the values of the datasets, but also provides a visual reference using other matching variable values . This increases the efficiency of finding the source of the differences and hence resolving these differences.

INTRODUCTION

A straightforward way to compare two datasets in SAS is by using the PROC COMPARE procedure. Considering the cost of reporting erroneous data in the pharmaceutical world, it is imperative that two programmers attempt to independently produce the same dataset (or output) and arrive at the same end product. The frequently utilized way to ensure that the two datasets are identical is by comparing the datasets using PROC COMPARE.

The ‘results’ of the proc compare provides various details about the datasets being compared such as, the number and attributes of variables, the number of observations and an overall message about the similarity of the datasets. Ideally, we hope to see a note in the results window that says ‘No unequal values were found, all values compared are exactly equal’.

In reality, we often encounter datasets that differ, and so the next steps are to explore the source of the differences, make appropriate programming changes, and repeat the process till we get the ideal ‘note’. While the basic proc compare syntax informs us about the differences, it is does not always sufficiently provide the details necessary to explore the source of the differences due to the inherent limitations of the results window display. This paper aims to explore the limitations and discuss the advantages a one stop solution for the limitations.

For all the examples in this paper we use a dummy Adverse Event dataset that’s structured similar to what’s encountered in the clinical trials arena.

PROC COMPARE RESULTS WHEN DATASETS MATCH

To establish a baseline understanding of proc compare, we use two copies of a dataset, AE_BASE and AE_QC, and provide the proc compare results of the two identical copies below:

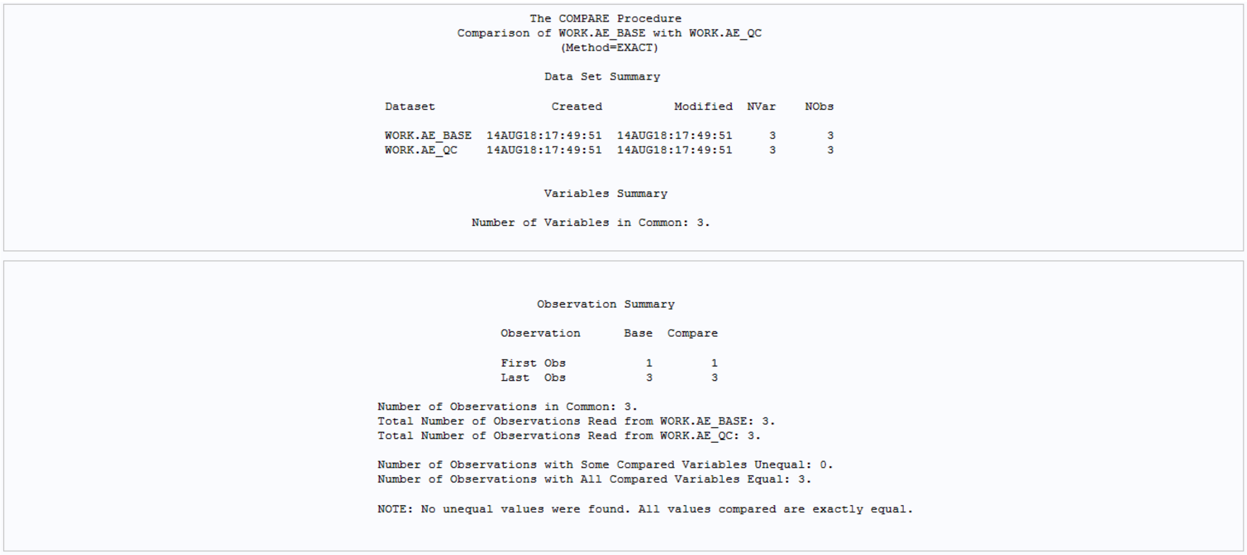
Table 1 AE_DUMMY

SUBJID	AGE	AETERM
PROC-COMPARE-OUT-101-601-1001	54	HEPATIC CYTOLYSIS
PROC-COMPARE-OUT-101-601-1002	61	NEUTROPENIA

PROC-COMPARE-OUT-101-601-1003	28	REDUCED APPETITE

```
data ae_base ae_qc;
 set ae_dummy;
run;

proc compare base=ae_base comp=ae_qc ;
run;
```

Figure 1 RESULT 1



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

                     Data Set Summary

Dataset                 Created          Modified  NVar    NObs

WORK.AE_BASE  14AUG18:17:49:51  14AUG18:17:49:51     3       3
WORK.AE_QC    14AUG18:17:49:51  14AUG18:17:49:51     3       3


                     Variables Summary

            Number of Variables in Common: 3.
```

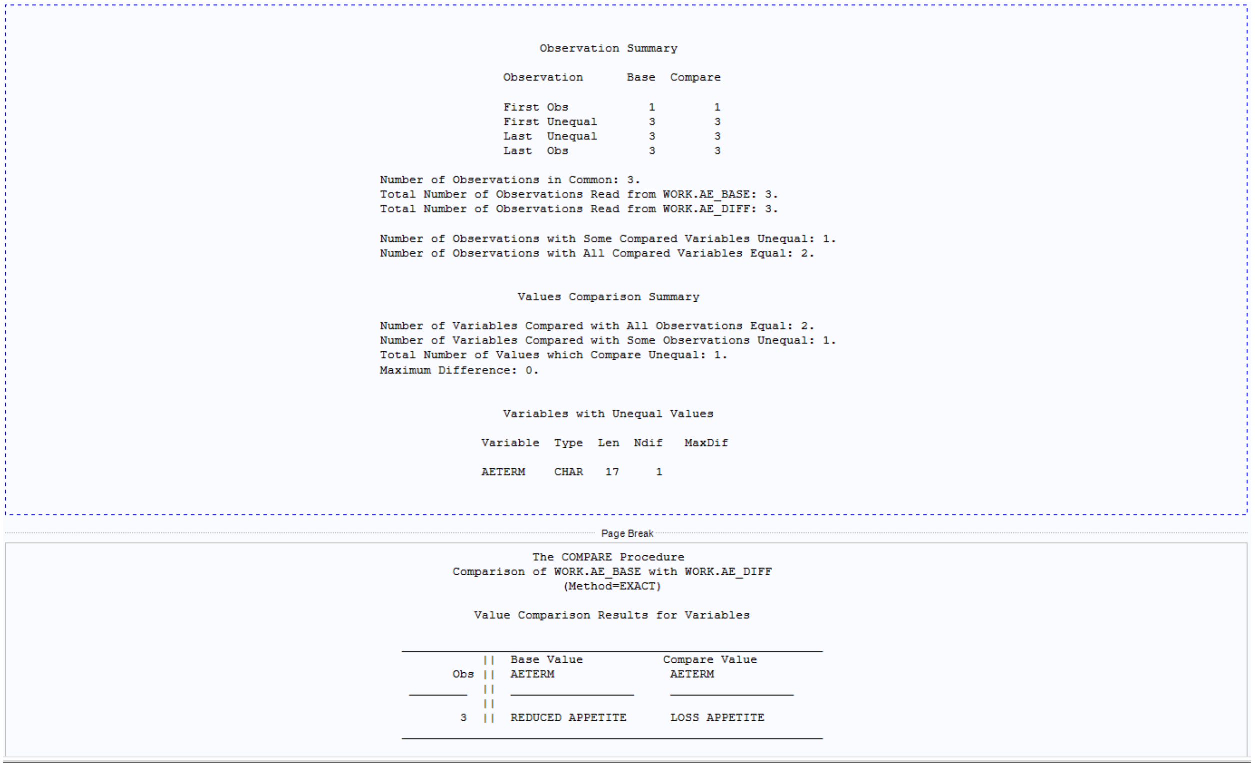
```
                    Observation Summary

               Observation      Base  Compare

               First Obs           1        1
               Last  Obs           3        3

 Number of Observations in Common: 3.
 Total Number of Observations Read from WORK.AE_BASE: 3.
 Total Number of Observations Read from WORK.AE_QC: 3.

 Number of Observations with Some Compared Variables Unequal: 0.
 Number of Observations with All Compared Variables Equal: 3.

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

Figure 1 is an example of what we, as programmers, desire when comparing two datasets. It validates that the dataset created above is in line with the intentions of the specification document.

To also provide a scenario where proc compare provides a summary of the differences between non identical datasets, we modified the AE_QC dataset as shown below and present the result of the procedure:

```
data ae_diff ;
      set ae_qc ;
      if _n_ = 3 then aeterm = 'LOSS APPETITE' ;
run ;

proc compare base=ae_base comp=ae_diff ;
run ;
```

Figure 2 RESULT 2



```
                      Observation Summary

                 Observation      Base  Compare

                 First Obs           1        1
                 First Unequal       3        3
                 Last  Unequal       3        3
                 Last  Obs           3        3

Number of Observations in Common: 3.
Total Number of Observations Read from WORK.AE_BASE: 3.
Total Number of Observations Read from WORK.AE_DIFF: 3.

Number of Observations with Some Compared Variables Unequal: 1.
Number of Observations with All Compared Variables Equal: 2.


                   Values Comparison Summary

Number of Variables Compared with All Observations Equal: 2.
Number of Variables Compared with Some Observations Unequal: 1.
Total Number of Values which Compare Unequal: 1.
Maximum Difference: 0.


                 Variables with Unequal Values

              Variable  Type  Len  Ndif   MaxDif

              AETERM    CHAR   17     1
```

Page Break

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

              Value Comparison Results for Variables

          __________________________________________________________
                     ||  Base Value           Compare Value
                 Obs ||  AETERM                AETERM
           ________  ||  _________________     _________________
                     ||
                   3 ||  REDUCED APPETITE      LOSS APPETITE
          __________________________________________________________
```

LIMITATIONS OF PROC COMPARE RESULT WINDOW

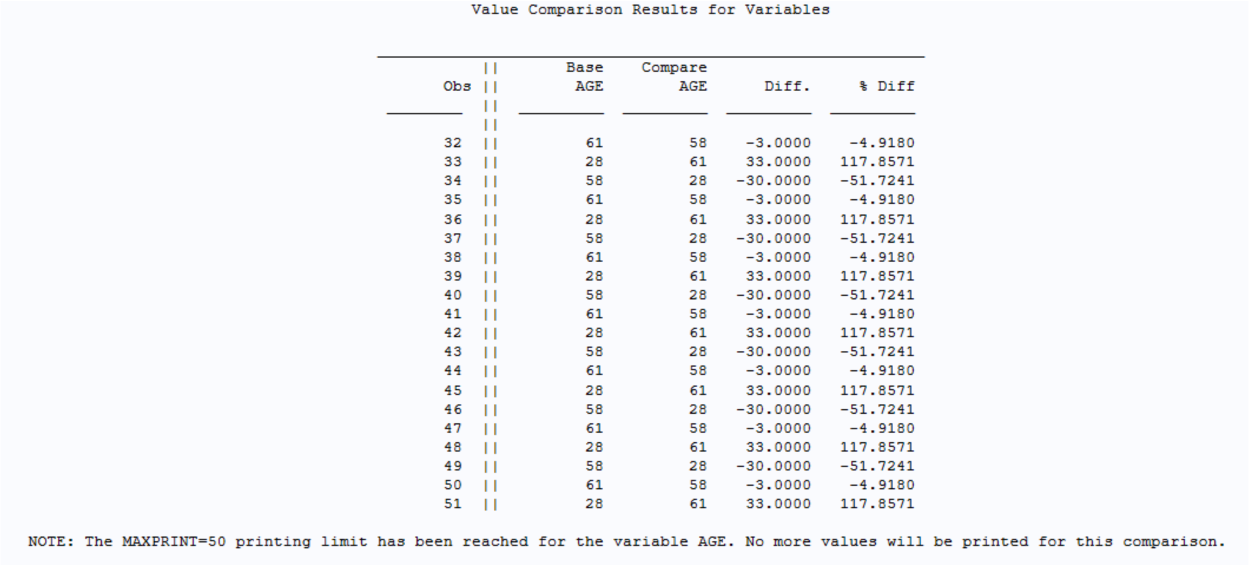
On the surface, the scenario presented above in Figure 2, seems sufficient to identify the differences in the datasets and to proceed with the exploration of the source of the differences. However, there are commonly encountered situations where the display of the differences is not enough to proceed to the next logical step or at best it requires additional digging to locate the observation that is different in the two datasets. Below we describe some of the commonly encountered situations where additional time and effort are needed beyond mere usage of proc compare in order to pinpoint the source of differences.

To showcase the limitations, we created a dataset by repeating the 3 observations in the AE_DUMMY dataset 20 times and thereby produced a base dataset with 60 observations. Further, we added a random observation to the first position (_N_=1) of the compare dataset and hence had 61 observations. This produced one mismatch in the base and compare dataset.

LACK OF ORIENTATION

While we artificially create two differing datasets and know how they were created, for the purposes of the next few sections the readers are encouraged to imagine not having the knowledge about the reason for the differences. The result of the default proc compare statement comparing the two datasets with 60 and 61 observations (partial) are displayed below:

Figure 3 RESULT 3



```
                         Value Comparison Results for Variables

                   __________________________________________________________
                             ||       Base    Compare
                         Obs ||        AGE        AGE      Diff.     % Diff
                    ________ ||  _________  _________  _________  _________
                             ||
                          32 ||         61         58    -3.0000    -4.9180
                          33 ||         28         61    33.0000   117.8571
                          34 ||         58         28   -30.0000   -51.7241
                          35 ||         61         58    -3.0000    -4.9180
                          36 ||         28         61    33.0000   117.8571
                          37 ||         58         28   -30.0000   -51.7241
                          38 ||         61         58    -3.0000    -4.9180
                          39 ||         28         61    33.0000   117.8571
                          40 ||         58         28   -30.0000   -51.7241
                          41 ||         61         58    -3.0000    -4.9180
                          42 ||         28         61    33.0000   117.8571
                          43 ||         58         28   -30.0000   -51.7241
                          44 ||         61         58    -3.0000    -4.9180
                          45 ||         28         61    33.0000   117.8571
                          46 ||         58         28   -30.0000   -51.7241
                          47 ||         61         58    -3.0000    -4.9180
                          48 ||         28         61    33.0000   117.8571
                          49 ||         58         28   -30.0000   -51.7241
                          50 ||         61         58    -3.0000    -4.9180
                          51 ||         28         61    33.0000   117.8571

  NOTE: The MAXPRINT=50 printing limit has been reached for the variable AGE. No more values will be printed for this comparison.
```

Due to the addition of an extra observation in the compare dataset, the values of age are off by one record. Result 3 above clearly shows that the variable ‘AGE’ does not match between the two datasets, however, just by looking at the displayed result, we cannot ascertain what observation or subjects do not have matching age. In order to do that we would have to, as an extra step, open the base and compare datasets and, for example, look at the 32nd observation in both and figure out the cause for the difference. As a disclaimer, anchoring or orientation of the differing observations can be achieved by using the ID statement, but this paper proposes a better approach in sections below.

MAXIMUM PRINTED DIFFERENCES:

As is evident by the ‘NOTE’ in RESULT 3, while comparing large datasets, the results window only displays 50 differences and the entirety of the differences are not visualized. While a display of 50 differences provides sufficient idea about the nature of the issue, it is sometimes better to see all the differences to recognize the pattern of the discrepancy corresponding to different groups of observations.

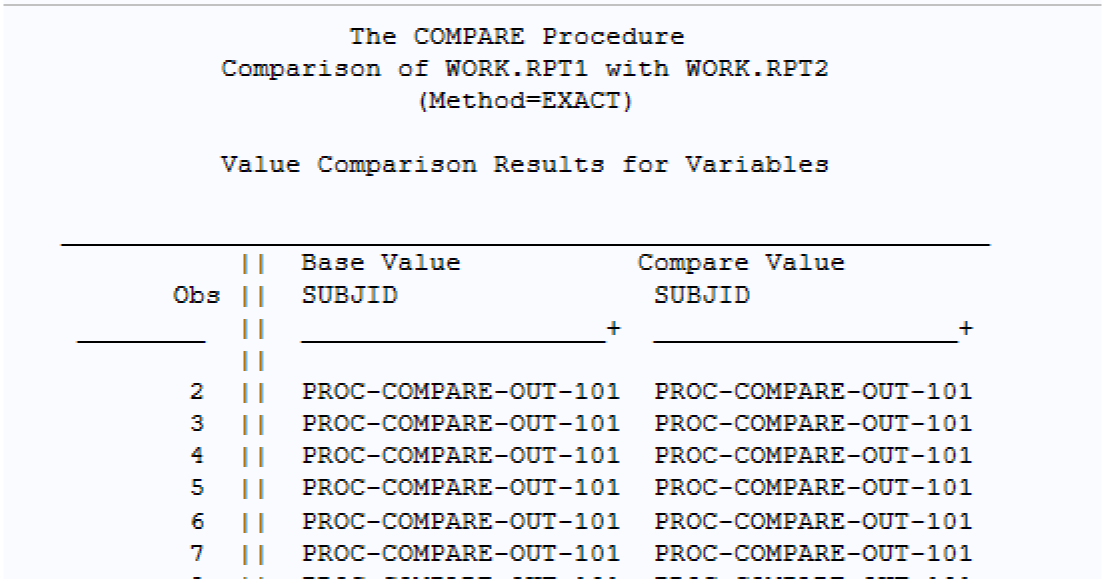
As an example, if a dataset was created by setting multiple other feeder datasets and age was calculated differently in each of the feeder datasets, then visualizing just 50 differences might not inform us about every scenario that deviated from the intentions of the dataset specifications. We might fix one set of problems, send it for validation and then get it back for rework with a new set of issues in the same variable. This is not an uncommon occurrence in the clinical trials world.

TRUNCATION OF LONG VALUES:

The base and compare datasets created above also showcase a limitation of the proc compare display. Observation values that exceed a 20 character limit are truncated. The last seven digits of value of ‘SUBJID’ variable in observation 2 in Base dataset is ‘601-1002’ and ‘601-1001’, and since they are getting truncated, it is difficult to know the exact difference unlike in RESULT 2.

As an extension, this scenario also happens when there are trailing blanks in an observation and not in the corresponding ‘compare’ observation. Unlike the truncation issue, trailing blanks are not visually evident even after opening the two datasets being compared.

Figure 4 RESULT 4



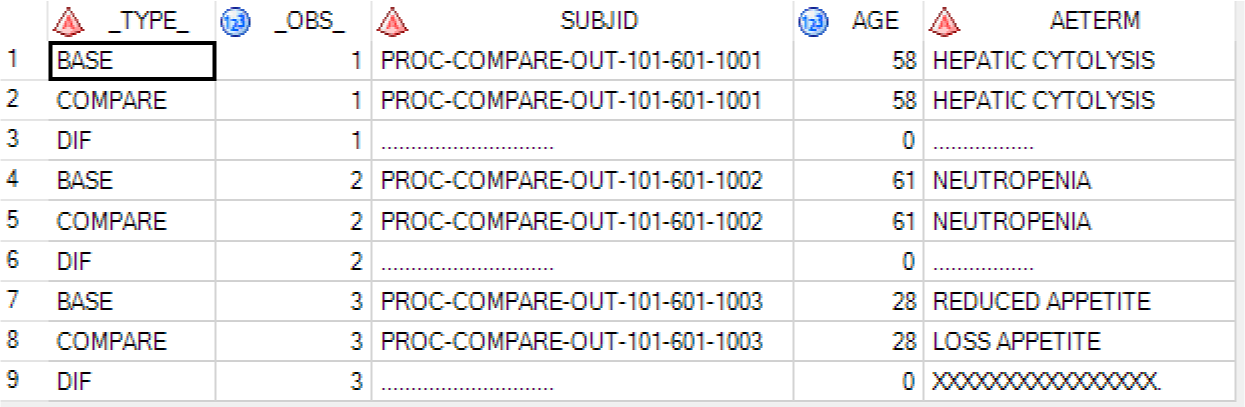
```
                    The COMPARE Procedure
            Comparison of WORK.RPT1 with WORK.RPT2
                       (Method=EXACT)

            Value Comparison Results for Variables

   __________________________________________________________
              ||  Base Value           Compare Value
          Obs ||  SUBJID                SUBJID
    ________  ||  ___________________+  ___________________+
              ||
            2 ||  PROC-COMPARE-OUT-101  PROC-COMPARE-OUT-101
            3 ||  PROC-COMPARE-OUT-101  PROC-COMPARE-OUT-101
            4 ||  PROC-COMPARE-OUT-101  PROC-COMPARE-OUT-101
            5 ||  PROC-COMPARE-OUT-101  PROC-COMPARE-OUT-101
            6 ||  PROC-COMPARE-OUT-101  PROC-COMPARE-OUT-101
            7 ||  PROC-COMPARE-OUT-101  PROC-COMPARE-OUT-101
```

USE OF THE OUT= OPTION:

The proc compare statement that produced RESULT 1 can be enhanced by various options and the “OUT=” option can be a one stop shop for solving all the above noted limitation of the result window display. The syntax we use and the resulting dataset are presented below:

```
proc compare base=ae_base comp=ae_diff
      out=sumry outbase outcomp outdiff ;
run;
```

Table 2 SUMRY DATA



	TYPE	_OBS_	SUBJID	AGE	AETERM
1	BASE	1	PROC-COMPARE-OUT-101-601-1001	58	HEPATIC CYTOLYSIS
2	COMPARE	1	PROC-COMPARE-OUT-101-601-1001	58	HEPATIC CYTOLYSIS
3	DIF	1	....	0	....
4	BASE	2	PROC-COMPARE-OUT-101-601-1002	61	NEUTROPENIA
5	COMPARE	2	PROC-COMPARE-OUT-101-601-1002	61	NEUTROPENIA
6	DIF	2	....	0	....
7	BASE	3	PROC-COMPARE-OUT-101-601-1003	28	REDUCED APPETITE
8	COMPARE	3	PROC-COMPARE-OUT-101-601-1003	28	LOSS APPETITE
9	DIF	3	....	0	XXXXXXXXXXXXXXXX.

The OUT= option produces a dataset that creates three records for each observation that is being compared and two new variables, _TYPE_ and _OBS_. For example, the first 3 records in the SUMRY dataset present various information about a single observation (_OBS_ = 1). The first observation is from the BASE dataset, the second observation is from the COMPARE dataset and the third observation tells us about the similarity or difference between the first two observations. This is evident from the newly generated variable ‘_TYPE_’.

The _TYPE_ = DIF rows are usually the focus of a validator programmer. For numeric variables (AGE) , the 'DIF' rows provides the numeric difference between the values and for identical values the difference is '0' as seen in all the 3 rows in our example. For character variables, a dotted line or a blank row indicates identical values, however any number of 'X' indicates a difference in the values as seen in the last row of the AETERM variable.

Unlike as seen in RESULT 1, here it is visually easy to recognize that the difference in AETERM is related to subject PROC-COMPARE-OUT-101-601-1003 and we thus have our orientation.

We are also not encumbered by the limitation of 50 maximum displayed differences as all the differences are displayed in the dataset generated by OUT= option.

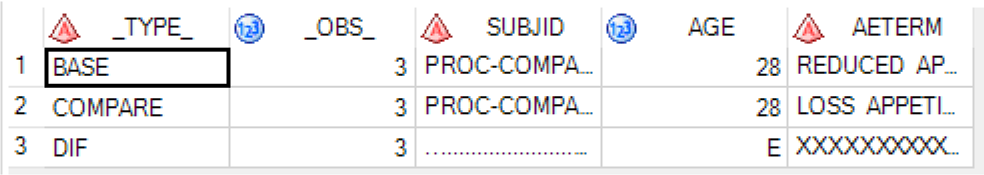
For character variables, a trailing blank also triggers 'X' in the DIF rows.

The copy paste feature is more intuitive in a dataset than in the 'RESULT' window and thus makes the communication of differences between the programmers easier.

If OUTNOEQUAL option is added, it suppresses all the identical observations between the two datasets as shown below:

```
proc compare base=ae_base comp=ae_diff
      out=sumry outbase outcomp outdiff outnoequal;
run;
```

Table 3 SMRY2 DATA



	TYPE	_OBS_	SUBJID	AGE	AETERM
1	BASE	3	PROC-COMPA...	28	REDUCED AP...
2	COMPARE	3	PROC-COMPA...	28	LOSS APPETI...
3	DIF	3	....	E	XXXXXXXXXX...

The SMRY2 data contains only the unequal observations and hence is different than the SMRY data where all the observations, regardless of whether they are identical between the two datasets being compared, are presented. When comparing large datasets, the programmer can choose to display all the observations or just the differences based on what would be most helpful in resolving the differences.

Uses of OUT= beyond routine validation

Numerous datasets are created to support the submission process of each study in the Clinical Trials environment. The normal practice is to validate each dataset individually and output the 'RESULT' into a permanent file and store it for perusal. At the completion of the validation of all individual datasets, the lead of the study either opens up each individual validation report or runs a macro to extract and check that a note describing that "All values compared are exactly equal" is found in all the outputs.

We propose that an alternative approach could be to check the size of the dataset that is produced by the OUT= option. If two datasets are identical then the size of the dataset produced by the OUT= option is 0 kb as it would have no observations in it when the 'OUTNOEQUAL' option is used. Only a few lines of code could check the size or the number of observations of all the datasets produced in a given folder as shown below:

```
proc sql ;
  create table VALRPT as
  select libname, memname, nlobs, filesize
  from dictionary.tables
  where libname = 'WORK'
```

```
    and memname = 'SMRY2'
  ;
quit ;
```

CONCLUSION

The OUT= option is one small syntax addition and offers extensive benefits, yet it is not widely practiced in our experience. It can significantly reduce the time and effort it takes to validate a dataset and It helps in the communication between the primary and the validator programmer as the entirety of the differences can be visualized rather than truncated or fragmented picture that the 'RESULT' window provides.

CONTACT INFORMATION

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

<PRASAD ILAPOGU>
<Ephicacy Consulting Group>
<Prasad.ilapogu@pfizer.com>

<MASAKI MIHAILA>
<Pfizer Inc>
<Masaki.mihaila@pfizer.com>