

## Don't Get Blindsided by PROC COMPARE

Joshua Horstman, Nested Loop Consulting, Indianapolis, IN  
Roger Muller, Data-to-Events.com, Carmel, IN

### ABSTRACT

"NOTE: No unequal values were found. All values compared are exactly equal." In the clinical trial world, that message is the holy grail for the programmer tasked with independently replicating a production dataset to ensure its correctness. Such a validation effort typically culminates in a call to PROC COMPARE to ascertain whether the production dataset matches the replicated one. It is often assumed that this message means the job is done. Unfortunately, it is not so simple. The unwary programmer may later discover that significant discrepancies slipped through. This paper surveys some common pitfalls in the use of PROC COMPARE and explains how to avoid them.

### INTRODUCTION

PROC COMPARE is a SAS<sup>®</sup> procedure which provides the programmer with a simple facility for comparing two datasets. While the procedure is capable of providing exhaustive listings of the differences between two datasets, it is frequently the case that the programmer is simply interested in confirming that two datasets are the same.

For example, a programmer may compare an output dataset to a previous version of the same dataset to verify that changes to the code which produced the dataset have had no unintended consequences. In statistical programming related to clinical trials, analysis datasets are often "double programmed". The specified output is independently constructed by two separate programmers and the results are then compared.

In any case, the programmer often scans the PROC COMPARE output for a note at the bottom saying "No unequal values were found. All values compared are exactly equal." This note is taken as a definitive statement that the two datasets are absolutely identical. This is a serious mistake which can result in the programmer being blindsided.

Fortunately, there are ways to avoid being caught in this trap. Just as a rearview mirror provides a driver with additional information to help avoid being struck by an unseen vehicle, so the COMPARE procedure output provides the programmer with additional information to help avoid being struck by unseen data discrepancies. In both cases, the operator needs to be looking in the right places to eliminate any blind spots.

In this paper, we will review some of the common blind spots that occur in PROC COMPARE and describe where the savvy programmer should be looking to avoid them.

### BLIND SPOT #1: MISSING VARIABLES

Our first example will use the two demography (DM) datasets shown below. We have a production dataset, DM\_PROD, and a validation or QC dataset, DM\_QC. The validation programmer wishes to use PROC COMPARE to determine if the two datasets are the same.

DM\_PROD Dataset

SUBJID	AGE	SEX
101	45	M
102	37	F
103	61	F

DM\_QC Dataset

SUBJID	AGE
101	45
102	37
103	61

Observe that the QC dataset does not contain all of the variables that are in the production dataset. Specifically, the SEX variable is not present. Below are the PROC COMPARE code and output.

```
proc compare base=dm_prod compare=dm_qc; run;
```

**Output 1. Output from PROC COMPARE of DM\_PROD and DM\_QC.**

```

                                The COMPARE Procedure
                                Comparison of WORK.DM_PROD with WORK.DM_QC
                                (Method=EXACT)

                                Data Set Summary

Dataset              Created          Modified  NVar    NObs
WORK.DM_PROD        30MAR13:17:23:18  30MAR13:17:23:18    3      3
WORK.DM_QC          30MAR13:17:23:18  30MAR13:17:23:18    2      3

                                Variables Summary

Number of Variables in Common: 2.
Number of Variables in WORK.DM_PROD but not in WORK.DM_QC: 1.

                                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.DM_PROD: 3.
Total Number of Observations Read from WORK.DM_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.

```

Notice the last line of the procedure output: “NOTE: No unequal values were found. All values compared are exactly equal.” The careless programmer will interpret this to mean that the job is finished. However, a more thorough inspection of the procedure results reveals an important blind spot.

PROC COMPARE alerts us to the blind spot in two different ways. In the first section of the report, the “Data Set Summary”, we see that one dataset has 3 variables while the other has only 2. Further down in the “Variables Summary”, the report indicates that there is 1 variable in WORK.DM\_PROD but not in WORK.DM\_QC.

These items are red flags to the programmer. Even though “all values compared are exactly equal,” not all values were compared. The COMPARE procedure simply compares the datasets based on the variables in common and reports its findings based on that comparison. If the goal of the validation programmer was to ensure that the SEX variable was derived correctly in the production dataset, that goal has not been accomplished.

**BLIND SPOT #2: MISSING OBSERVATIONS**

Our second example will use the two vital signs (VS) datasets shown below. We have a production dataset, VS\_PROD, and a validation or QC dataset, VS\_QC. As before, the validation programmer wishes to use PROC COMPARE to determine if the two datasets are the same.

VS_PROD Dataset				VS_QC Dataset			
SUBJID	VISITNUM	SYSBP	DIABP	SUBJID	VISITNUM	SYSBP	DIABP
101	1	120	80	101	1	120	80
101	2	126	84	101	2	126	84
102	1	132	90	102	1	132	90
102	2	131	85				

Observe that the QC dataset does not contain all of the observations that are in the production dataset. Specifically, the record for subject 102 at visit 2 is not present. Below are the PROC COMPARE code and output.

```
proc compare base=vs_prod compare=vs_qc; run;
```

## Output 2. Output from PROC COMPARE of VS\_PROD and VS\_QC.

```

                                The COMPARE Procedure
          Comparison of WORK.VS_PROD with WORK.VS_QC
                    (Method=EXACT)

                    Data Set Summary

Dataset              Created              Modified  NVar    NObs
-----              -
WORK.VS_PROD         30MAR13:17:23:18  30MAR13:17:23:18    4      4
WORK.VS_QC           30MAR13:17:23:18  30MAR13:17:23:18    4      3

                    Variables Summary

          Number of Variables in Common: 4.

                    Observation Summary

          Observation      Base  Compare
          -----
          First Obs       1      1
          Last Match      3      3
          Last Obs        4      .

          Number of Observations in Common: 3.
          Number of Observations in WORK.VS_PROD but not in WORK.VS_QC: 1.
          Total Number of Observations Read from WORK.VS_PROD: 4.
          Total Number of Observations Read from WORK.VS_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.
```

We find that familiar note at the bottom of the procedure output: “NOTE: No unequal values were found. All values compared are exactly equal.” Once again, however, a more careful reading of the output is necessary.

Notice the “Data Set Summary” indicates one dataset contains 4 observations while the other has only 3. We also see the “Observation Summary” that there is 1 observation in WORK.VS\_PROD but not in WORK.VS\_QC. Again, these are red flags. The comparison was performed based only on the observations in common. If there are any problems with the last record in the production dataset (the record that is missing from the QC dataset), we won’t discover it because we have a blind spot.

Those familiar with the COMPARE procedure may be wondering if this blind spot can be avoided though the use of the ID statement. The ID statement is used to specify key variables that the COMPARE procedure should use to match observations. The logical ID variables in this example would be SUBJID and VISITNUM because the combination of these two variables uniquely identifies an observation. While the ID statement is extremely useful in certain situations, it does not help us here. The resulting report is very similar and still concludes with “NOTE: No unequal values were found. All values compared are exactly equal.”

### BLIND SPOT #3: CONFLICTING TYPES

For our third example, consider the two laboratory results (LB) datasets shown below. We have a production dataset, LB\_PROD, and a validation or QC dataset, LB\_QC. Once again, the validation programmer wishes to use PROC COMPARE to determine if the two datasets are the same.

LB_PROD Dataset				LB_QC Dataset			
SUBJID	VISITNUM	LBTESTCD	LBORRES	SUBJID	VISITNUM	LBTESTCD	LBORRES
101	1	ALB	3.6	101	1	ALB	13.6
101	1	ALP	47.2	101	1	ALP	57.2
101	1	AST	13.5	101	1	AST	23.5
101	1	BILI	0.8	101	1	BILI	10.8

This time, the difference is more subtle. In the production dataset, LBORRES (the original lab result) is a numeric variable, but in the QC dataset, it is a character variable. In addition to this difference, the results themselves have been altered in the QC dataset to highlight the danger here. Below are the PROC COMPARE code and output.

```
proc compare base=lb_prod compare=lb_qc; run;
```

#### Output 3. Output from PROC COMPARE of LB\_PROD and LB\_QC.

```

                                The COMPARE Procedure
          Comparison of WORK.LB_PROD with WORK.LB_QC
                    (Method=EXACT)

                                Data Set Summary

Dataset              Created              Modified  NVar    NObs
WORK.LB_PROD        30MAR13:18:57:21  30MAR13:18:57:21    4        4
WORK.LB_QC          30MAR13:18:57:21  30MAR13:18:57:21    4        4

                                Variables Summary

Number of Variables in Common: 4.
Number of Variables with Conflicting Types: 1.

Listing of Common Variables with Conflicting Types

Variable  Dataset      Type  Length
lborres   WORK.LB_PROD  Num    8
          WORK.LB_QC  Char   12

                                Observation Summary

Observation      Base  Compare
First Obs        1      1
Last Obs         4      4

Number of Observations in Common: 4.
Total Number of Observations Read from WORK.LB_PROD: 4.
Total Number of Observations Read from WORK.LB_QC: 4.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 4.

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

```

Despite the fact that the two datasets contain completely different results for LBORRES, we once again see the message the "No unequal values were found. All values compared are exactly equal." Even checking the number of variables and the number of observations in the "Data Set Summary" does not help us here.

We must look further down to the "Variables Summary" to see that there is 1 variable with conflicting types. The report also contains a new section called "Listing of Common Variables with Conflicting Types."

What is important to note here is that variables with conflicting types are excluded from the comparison. This is a huge blind spot! The inattentive validation programmer is at serious risk here of declaring valid a production dataset that may in fact be erroneous.

## RECOMMENDATIONS

In order to avoid leaving yourself exposed to one of the above blind spots, there are several good habits the SAS programmer should develop.

First, know thy data. Know what variables you have in each dataset and their types. Know how many observations are in each dataset and where they came from. Examine your datasets carefully to make sure they contain what you are expecting. While automated tools like the COMPARE procedure are tremendously powerful, they are no substitute for a good, working knowledge of the data.

Second, always review the entire PROC COMPARE output, particularly if your goal is to confirm that two datasets are identical. Don't be lulled into a false sense of security by seeing "NOTE: No unequal values were found. All values compared are exactly equal." Check that the number of variables and the number of observations. If these don't match, all other conclusions about the data are suspect.

Finally, if your validation dataset contains extraneous variables that aren't part of the production dataset being validated, drop them before you call PROC COMPARE. In so doing, you will be able to more easily recognize when one dataset is missing variables because the variable counts won't match.

## CONCLUSION

Just as a skilled craftsman must be thoroughly familiar with the proper use of his tools, so the SAS programmer must understand how the various SAS procedures work. It's not enough to throw input at a procedure and blindly trust that the results mean what you think they should mean. In the case of PROC COMPARE, this means understanding what is actually being compared and what is not being compared.

## CONTACT INFORMATION

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

Name: Roger D. Muller  
Company: Data-to-Events.com  
Phone: 317-846-5782  
E-mail: rdmuller@hotmail.com

Name: Joshua M. Horstman  
Company: Nested Loop Consulting  
Phone: 317-815-5899  
E-mail: jmhorstman@gmail.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.