PROC COMPARE: What Did I Inadvertently Change?

Melissa R. Pfeiffer, Children's Hospital of Philadelphia, Philadelphia, Pennsylvania

ABSTRACT

SAS® PROC COMPARE is a valuable tool to use for comparing specific variables within a single data set, all variables across two different data sets, or specific variables across two different data sets. For example, when working with new functions for the first time, you may want to verify the results of the code against those generated by code you have used previously and are familiar with. Similarly, when revising inherited code for efficiency, it can be helpful to compare a data set resulting from the new code with a previously generated data set. Information generated by PROC COMPARE includes the number of variables and observations in each data set, the number of variables and observations with unequal values, and displays of specific value differences, which can be extremely helpful in verifying that values have not inadvertently changed.

INTRODUCTION

The COMPARE procedure can be used to compare variables within the same data set, the entire contents of two data sets, or just selected variables that have either the same or different names in distinct data sets.

This paper will introduce PROC COMPARE to SAS users at all levels who are unfamiliar with this under-used procedure. It will present the basic structure of PROC COMPARE, how the procedure can be augmented with a few simple statements and options, and how to use this versatile procedure to make comparisons within a single data set or across data sets. Examples used were developed in SAS 9.3.

COMPARISONS WITHIN A SINGLE DATA SET

First, I'll demonstrate how to compare variables within one data set. I find this especially useful for learning about new functions or evaluating new coding. I might create one variable using a familiar method, create another variable using the new method, and then use PROC COMPARE to see if the results are the same for both methods. For instance, perhaps I have one variable for age coded by flooring the results of the YRDIF function with the AGE basis, available in SAS 9.3, and another variable with the floored results of the YRDIF function with the ACT/ACT basis.

The following code can be used to compare those two variables within the same data set:

```
PROC COMPARE BASE = data_set_name ;
    VAR age_yrdifage ;
    WITH age_yrdifactual ;
RUN;
```

The option BASE = allows you to specify the data set, in the VAR statement you list one of the variables you want compared, and in the WITH statement you list the other. The variables that you are comparing must be the same type, either both numeric or both character; other attributes, such as formats and labels, do not need to be the same.

The COMPARE Procedure
Comparisons of variables in WORK.DATA_SET_NAME
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.DATA_SET_NAME	27APR12:11:03:36	27APR12:11:03:36	25	10395

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 0.
Number of Variables Compared with Some Observations Unequal: 1.
Total Number of Values which Compare Unequal: 9.
Maximum Difference: 1.

All Variables Compared have Unequal Values

Variable	Type	Len	Compare	Len	Ndif	MaxDif
age_yrdifage	NUM	8	age_yrdifactual	8	9	1.000

Value Comparison Results for Variables

Obs	Base age_yrdif age	Compare age_yrdif actual	Diff.	% Diff
416	19.0000	18.0000	-1.0000	-5.2632
718	19.0000	18.0000	-1.0000	-5.2632
1945	18.0000	17.0000	-1.0000	-5.5556
4288	19.0000	18.0000	-1.0000	-5.2632
4406	19.0000	18.0000	-1.0000	-5.2632
6946	18.0000	17.0000	-1.0000	-5.5556
7048	17.0000	16.0000	-1.0000	-5.8824
7352	19.0000	18.0000	-1.0000	-5.2632
8854	19.0000	18.0000	-1.0000	-5.2632

Display 1. Results Generated by Running PROC COMPARE with One Pair of Variables in a Single Data Set

The information generated by this code includes:

1. data set information, including
    a. creation and modification dates
    b. the number of variables and observations
2. a comparison summary, including
    a. the number of variables compared with all observations equal
    b. the number of variables compared with some observations unequal
    c. the number of values that are unequal
    d. the maximum difference in values (if the variables are numeric)
3. information for specific observations when they have different values, including
    a. values in each variable
    b. the difference between values (again, if the variables are numeric)

But more than one pair of variables can be compared within the same procedure. This is accomplished by listing more than one variable in the VAR statement and more than one variable in the WITH statement. Values of the first variable listed in the VAR statement are compared against values of the first variable listed in the WITH statement; similarly, values of the second variables in each statement are compared against each other, values of the third variables in each statement are compared, et cetera.

```
PROC COMPARE BASE = data_set_name ;
    VAR variable1 variable3 variable5 ;
    WITH variable2 variable4 variable6 ;
RUN;
```

The COMPARE Procedure
Comparisons of variables in WORK.DATA_SET_NAME
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.DATA_SET_NAME	27APR12:10:23:46	27APR12:10:23:46	25	10395

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 2.
Number of Variables with Missing Value Differences: 1.
Total Number of Values which Compare Unequal: 5285.
Maximum Difference: 1.

Variables with Unequal Values

Variable	Type	Len	Compare	Len	Ndif	MaxDif	MissDif
variable1	NUM	8	variable2	8	5248	1.000	0
variable3	NUM	8	variable4	8	37	0	37

Value Comparison Results for Variables

Obs	Appropriate label for variable1 Base variable1	Compare variable2	Diff.	% Diff
1	17.0000	18.0000	1.0000	5.8824
2	18.0000	19.0000	1.0000	5.5556
6	18.0000	19.0000	1.0000	5.5556
7	19.0000	20.0000	1.0000	5.2632
9	20.0000	21.0000	1.0000	5.0000
87	19.0000	20.0000	1.0000	5.2632
88	20.0000	21.0000	1.0000	5.0000
91	19.0000	20.0000	1.0000	5.2632
92	17.0000	18.0000	1.0000	5.8824
96	20.0000	21.0000	1.0000	5.0000

NOTE: The MAXPRINT=50 printing limit has been reached for the comparison of variable variable1 with variable2. No more values will be printed for this comparison.

Display 2. Partial Results Generated by Running PROC COMPARE with Three Pairs of Variables in a Single Data Set – the Section Between Observation 9 and Observation 87 Has Been Omitted

By default, the maximum number of observations with unequal values that will be displayed is 50 per comparison and 500 total. This prevents producing pages and pages of output when there are many differences. At times, you may need to see examples of more differences. You can change these defaults with the MAXPRINT option on the PROC COMPARE statement. If you just want to change the overall maximum, list the desired maximum after the equals sign. To change both the per comparison and the overall maximum, list the per comparison number and maximum number in parentheses, separated by a comma.

```
PROC COMPARE BASE = data_set_name MAXPRINT = (200, 1000) ;
    VAR variable1 variable3 variable5 ;
    WITH variable2 variable4 variable6 ;
RUN;
```

401	||	17.0000	18.0000	1.0000	5.8824
402	||	20.0000	21.0000	1.0000	5.0000
403	||	18.0000	19.0000	1.0000	5.5556
405	||	18.0000	19.0000	1.0000	5.5556
410	||	19.0000	20.0000	1.0000	5.2632
411	||	17.0000	18.0000	1.0000	5.8824
412	||	18.0000	19.0000	1.0000	5.5556

NOTE: The MAXPRINT=200 printing limit has been reached for the comparison of variable variable1 with variable2. No more values will be printed for this comparison.

Display 3. Partial Results Generated by Running PROC COMPARE with MAXPRINT Option Set to 200 (Per Comparison) and 1000 (Overall)

You can also compare the same variable with more than one variable by repeating the variable name in one of the statements. Perhaps you want to compare an original variable against variables generated by several new methods. You can do this by listing the original variable multiple times in your VAR statement (or WITH statement), once for each variable you wish to compare the original against. All of those comparison variables should be listed in your WITH statement (or VAR statement, if the original is in the WITH statement).

```
PROC COMPARE BASE = data_set_name ;
    VAR variable1 variable1 variable1 ;
    WITH variable2 variable4 variable6 ;
RUN;
```

COMPARISONS ACROSS TWO DATA SETS

PROC COMPARE can be used to compare two different data sets; this is particularly useful when revising inherited code. The process for comparing variables across two data sets is very similar to that used to compare variables within one data set. On your PROC COMPARE statement, simply add the COMPARE = option and list the data set you would like the BASE data set to be compared against. If there are no other statements than your PROC COMPARE and RUN statements, all variables that have the same name and type in the two data sets will be compared and the values from the first observation in the BASE data set will be compared against values from the first observation in the COMPARE data set.

```
PROC COMPARE BASE = data_set_a COMPARE = data_set_b ;
RUN;
```

The COMPARE Procedure
Comparison of WORK.DATA_SET_A with WORK.DATA_SET_B
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.DATA_SET_A	27APR12:10:25:39	27APR12:10:25:39	22	10395
WORK.DATA_SET_B	27APR12:10:25:39	27APR12:10:25:39	22	10395

Variables Summary

Number of Variables in Common: 22.
Number of Variables with Conflicting Types: 1.

Listing of Common Variables with Conflicting Types

Variable	Dataset	Type	Length	Format	Label
gender	WORK.DATA_SET_A	Num	8	GENDERF.	Gender: 0 = Male, 1 = Female
	WORK.DATA_SET_B	Char	1		

Observation Summary

Observation	Base	Compare
First Obs	1	1
First Unequal	3	3
Last Unequal	10393	10393
Last Obs	10395	10395

Number of Observations in Common: 10395.
Total Number of Observations Read from WORK.DATA_SET_A: 10395.
Total Number of Observations Read from WORK.DATA_SET_B: 10395.

Number of Observations with Some Compared Variables Unequal: 1056.
Number of Observations with All Compared Variables Equal: 9339.

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 17.
Number of Variables Compared with Some Observations Unequal: 4.
Number of Variables with Missing Value Differences: 2.
Total Number of Values which Compare Unequal: 2126.
Maximum Difference: 5.

The COMPARE Procedure
Comparison of WORK.DATA_SET_A with WORK.DATA_SET_B
(Method=EXACT)

Variables with Unequal Values

Variable	Type	Len	Ndif	MaxDif	MissDif
season	NUM	8	1026	4.000	0
schoolyear	NUM	8	1026	5.000	0
variable3	NUM	8	37	0	37
variable4	NUM	8	37	0	37

Value Comparison Results for Variables

Season: 1=Winter(Jan-Mar), 2=Spring(Apr-
.. Jun), 3=Summer(Jul-Sep), 4=Fall(Oct-N
.. ov)

Obs	Base season	Compare season	Diff.	% Diff
3	3.0000	7.0000	4.0000	133.3333
5	3.0000	7.0000	4.0000	133.3333
13	3.0000	7.0000	4.0000	133.3333
22	3.0000	7.0000	4.0000	133.3333
26	3.0000	7.0000	4.0000	133.3333
40	3.0000	7.0000	4.0000	133.3333
47	3.0000	7.0000	4.0000	133.3333
52	3.0000	7.0000	4.0000	133.3333
54	3.0000	7.0000	4.0000	133.3333

Display 4. Partial Results Generated by Running PROC COMPARE with Two Data Sets and No VAR Statement

In addition to the output you get when variables from a single data set are compared, this code generates:

1. the number of variables in common,
2. the number of and names of common variables with conflicting types,
3. the observation numbers of the first and last unequal observations,
4. the number of observations in common, and
5. for variables in common,
    a. the number of observations with some variables unequal and
    b. the number of observations with all variables equal.

Instead of comparing observations in the two data sets according to their observation numbers (e.g., first with the first, second with the second), you can use an ID statement to compare observations that have the same value for a specific variable or variables; the values of the ID variable(s) determine which observations will be compared against each other. The ID variables must exist in both data sets and have the same type; your data sets should be sorted by these variables.

```
PROC COMPARE BASE = data_set_a COMPARE = data_set_b ;
    ID uniqueid ;
RUN;
```

```
                    Observation Summary

          Observation      Base  Compare  ID

          First Obs           1        1  uniqueid=3
          First Unequal       3        3  uniqueid=5
          Last  Unequal   10393    10393  uniqueid=16995
          Last  Obs       10395    10395  uniqueid=16997

Number of Observations in Common: 10395.
Total Number of Observations Read from WORK.DATA_SET_A: 10395.
Total Number of Observations Read from WORK.DATA_SET_B: 10395.

Number of Observations with Some Compared Variables Unequal: 1056.
Number of Observations with All Compared Variables Equal: 9339.


                Value Comparison Results for Variables

                                 ||  Season: 1=Winter(Jan-Mar), 2=Spring(Apr-
                                 ||  .. Jun), 3=Summer(Jul-Sep), 4=Fall(Oct-N
                                 ||  .. ov)
                                 ||     Base     Compare
uniqueid                         ||   season      season       Diff.     % Diff
--------                         ||  ---------  ---------  ---------  ---------
                                 ||
       5                         ||     3.0000     7.0000     4.0000   133.3333
       7                         ||     3.0000     7.0000     4.0000   133.3333
      17                         ||     3.0000     7.0000     4.0000   133.3333
      28                         ||     3.0000     7.0000     4.0000   133.3333
      32                         ||     3.0000     7.0000     4.0000   133.3333
      58                         ||     3.0000     7.0000     4.0000   133.3333
```

Display 5. Partial Results Generated by Running PROC COMPARE with Two Data Sets and an ID Statement – Section Between Observation Summary and Value Comparison Results for Variables Omitted

Note that when the ID statement is used, the values for the variable(s) on the ID statement are added for first and last observations in common and unequal observations and the values replace the observation number in the value comparison results. This may make it easier to conduct further investigation or data management when values are different.

When using the ID statement, the values for those variables should be relatively unique. If there is more than one observation with the same value, then the first observations with the value will be compared against each other, the second observations with the same value will be compared against each other, etc. and you will get a message to the log about duplicate observations. Depending on the order of the observations in the data sets, this could make it appear as though values are different, when the observations are just not paired properly. You may need to use several variables, such as uniqueid and visitdate, when you sort the data sets and in your ID statement in order to ensure that each observation has a unique combination of values for those variables.

```
63   proc compare base=data_set_aa compare=data_set_bb ;
64        id uniqueid;
65   run;

WARNING: The data set WORK.DATA_SET_AA contains a duplicate observation at observation number 3.
NOTE: At observation 3 the current and previous ID values are:
      uniqueid=4.
NOTE: Further warnings for duplicate observations in this data set will not be printed.
WARNING: The data set WORK.DATA_SET_BB contains a duplicate observation at observation number 3.
NOTE: At observation 3 the current and previous ID values are:
      uniqueid=4.
NOTE: Further warnings for duplicate observations in this data set will not be printed.
NOTE: There were 16190 observations read from the data set WORK.DATA_SET_AA.
NOTE: There were 16190 observations read from the data set WORK.DATA_SET_BB.
NOTE: PROCEDURE COMPARE used (Total process time):
      real time           0.12 seconds
      cpu time            0.06 seconds
```

Display 6. Log Display When Comparison Data Sets Contain Duplicate Observations

There is a NOTSORTED option for the ID statement, which indicates to SAS that the data are not sorted alphabetically or numerically. However, PROC COMPARE still expects corresponding observations to have the same values for the ID variables. In other words, if you use the ID statement, sort both data sets by the ID variables.

The BY statement is similar to the ID statement and can be used to create separate comparisons for each level of the BY group; as with the ID statement, more than one variable can be listed. There is a NOTSORTED option that will cause SAS to look for continuous groups of values; without that option the data sets must be sorted by the variables listed in the BY statement. I find this statement to be less useful than the ID statement. Instead of simply indicating which variables should be used to match observations, the BY statement creates separate comparisons for each level of the BY group. That means that it can be more resource intensive and produces much more output. Variables listed in an ID statement or BY statement are not included among the variables being compared, since the observations must have matching values.

As with PROC COMPARE using a single data set, the VAR and WITH statements can be used to compare specific variables in the two data sets. This can be useful when coding changed for only some of your 3,000 variables, for instance, or when you are trying to determine why specific results are not quite the same. Perhaps the mean age from two data sets is different. PROC COMPARE can show you differences demonstrating that age is rounded in one data set but floored in the other. These statements allow you to compare variables in your two data sets that do not have the same name (age_floor and age_round, for instance).

```
PROC COMPARE BASE = data_set_a COMPARE = data_set_b ;
    ID uniqueid ;
    VAR age_floor ;
    WITH age_round ;
RUN;
```

With variables named “age_floor” and “age_round,” the variable names themselves might have revealed the reason for the discrepant results. However, if the variables were simply named “age,” you could use PROC COMPARE with just the VAR statement.

```
PROC COMPARE BASE = data_set_a COMPARE = data_set_b ;
    ID uniqueid ;
    VAR age ;
RUN;
```

Unlike the single data set process, in a comparison of two data sets, you can have a VAR statement without a WITH statement. This restricts the data set comparison to just the variable(s) specified in the VAR statement; these variables must have the same name and type in both data sets. Attributes such as length and informat can differ. As when comparisons are made with a single data set, multiple comparisons can be made in the same procedure, and a single variable in one data set can be compared against more than one variable in the other data set.

Orienting the results by observation rather than by variable, achievable with the TRANSPOSE option on the PROC COMPARE statement, can be particularly useful when variables with different values are related to each other. For instance, if a mistake was made in coding “season” for observations with a date in July, then a similar mistake might have been made coding the variable “schoolyear.”

```
PROC COMPARE BASE = data_set_a COMPARE = data_set_b TRANSPOSE ;
    ID uniqueid ;
    VAR season schoolyear ;
RUN;
```

```
                Comparison Results for Observations

uniqueid=5:
  Variable     Base Value       Compare          Diff.        % Diff
     season       3.000000      7.000000      4.000000    133.333333
 schoolyear              2             7      5.000000    250.000000

uniqueid=7:
  Variable     Base Value       Compare          Diff.        % Diff
     season       3.000000      7.000000      4.000000    133.333333
 schoolyear              2             7      5.000000    250.000000

uniqueid=17:
  Variable     Base Value       Compare          Diff.        % Diff
     season       3.000000      7.000000      4.000000    133.333333
 schoolyear              2             7      5.000000    250.000000
```

Display 7. Partial Results by Running PROC COMPARE with the TRANSPOSE Option

Keep in mind that the note “No unequal values were found. All values compared are exactly equal” is not necessarily cause for celebration. All *compared* values may be the same, but be sure to also look at the number of variables in common: if it’s not the same as the number of variables in your data sets (excluding any ID or BY variables), then there could still be important differences. Similarly, observations found in one data set but not the other may signal problems.

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

                              Data Set Summary

    Dataset                      Created           Modified  NVar    NObs

    WORK.DATA_SET_A    27APR12:10:25:39  27APR12:10:25:39    22   10395
    WORK.DATA_SET_BBB  27APR12:10:48:26  27APR12:10:48:26    14   10367

                             Variables Summary

    Number of Variables in Common: 14.
    Number of Variables in WORK.DATA_SET_A but not in WORK.DATA_SET_BBB: 8.
    Number of Variables with Conflicting Types: 1.
    Number of ID Variables: 1.

            Listing of Common Variables with Conflicting Types

Variable  Dataset            Type  Length  Format    Label

gender    WORK.DATA_SET_A    Num        8  GENDERF.  Gender: 0 = Male, 1 = Female
          WORK.DATA_SET_BBB  Char       1

                            Observation Summary

                 Observation      Base  Compare  ID

                 First Obs           1        1  uniqueid=3
                 Last  Obs       10395    10367  uniqueid=16997

  Number of Observations in Common: 10367.
  Number of Observations in WORK.DATA_SET_A but not in WORK.DATA_SET_BBB: 28.
  Total Number of Observations Read from WORK.DATA_SET_A: 10395.
  Total Number of Observations Read from WORK.DATA_SET_BBB: 10367.

  Number of Observations with Some Compared Variables Unequal: 0.
  Number of Observations with All Compared Variables Equal: 10367.

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

Display 8. Results Generated by Running PROC COMPARE with Two Data Sets That Have All Values Equal

for Variables and Observations Compared

These issues can be explored further with options available on the PROC COMPARE statement that control the listing of variables and observations. This can be helpful in situations in which two data sets should contain the same records and/or the same variables but don't. Instead of a painstaking comparison of the data sets by hand, you could use PROC COMPARE with these options.

Three options focus on the BASE data set: LISTBASEOBS lists all the observations found only in the BASE data set, LISTBASEVAR does the same for variables – lists only those variables in the BASE data set, and LISTBASE lists all the variables and observations found in only the BASE data set. There's a similar set focusing on the COMPARE data set: LISTCOMPOBS, LISTCOMPVAR, and LISTCOMP. You may be interested in differences in either direction, in which case you can use the options without BASE or COMP: LISTOBS, LISTVAR and LISTALL. Finally, LISTEQUALVAR lists variables whose values are judged equal.

There are also options to control the details displayed in the default report. MAXPRINT and TRANSPOSE, described above, are two of these options. Others include NOVALUES, which suppresses the report of the value comparison results and BRIEFSUMMARY or just BRIEF, which produces a short comparison summary and suppresses the four default summary reports: the data set summary, variable summary, observation summary, and values comparison summary sections. The NOSUMMARY option also suppresses the four summary sections. However, be aware that if you use that option and the data sets being compared are exactly the same, no output will be produced and there will be no warnings or errors in your log.

There are half a dozen other options that control the details displayed, including NOPRINT, which suppresses all printed output. You may want to include that option if you want to create an output data set, which you can do with the OUT= option. You can also specify which observations are included in your output data set, such as all of the observations in the BASE data set, all of the observations in the COMPARE data set, or all of the observations in either. You can also suppress observations that have all values equal.

Speaking of equal, you may have noticed the message "Method = EXACT" at the top of the PROC COMPARE output. There are options to specify how the values are compared. You can change from EXACT to ABSOLUTE or PERCENT. You can also alter how missing values are treated. NOMISSING judges missing values in both the base and comparison data sets equal to any value. NOMISSBASE and NOMISSCOMP operate similarly, but with just the base or comparison data set variables.

CONCLUSION

PROC COMPARE is a useful tool in the programmers' repertoire to efficiently compare variables of interest within and across data sets, reducing the chances of unintended changes in data sets going undetected and increasing data quality. This procedure is adaptable to one data set or two and flexible with the variables included in the comparisons and the determination of which observations are compared. Options on the PROC COMPARE statement include those that expand displays of variables and observations in one data set but not the other, those that suppress sections of the default report, and those that control the destination of output. This versatile tool has still more options not presented here, including those that control the method and precision of value comparisons (i.e., METHOD and CRITERION).

REFERENCES

- SAS Institute, Inc. 2012. "COMPARE Procedure." *Base SAS® 9.3 Procedures Guide, Second Edition*, Cary, NC: SAS Institute Inc. July 7, 2014. Available at http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/n0c1y14wyd3u7yn1dmfcpaejllsn.htm
- Williams, Christianna S. 2010. "PROC COMPARE – Worth Another Look!" *Proceedings of the SAS Global Forum 2010 Conference.* Cary, North Carolina: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings10/149-2010.pdf

ACKNOWLEDGMENTS

The author wishes to thank Kristi Metzger, PhD, MPH for her thoughtful review of this paper as well as Megin Nichols, DVM, MPH and Marisa Bargsten, MPH for their support, inspiration, and review. Any mistakes herein are the responsibility of the author.

This paper was expanded for presentation at the 2018 South Central SAS Users Group (SCSUG) Forum. A previous, and shorter, version was presented at the 2014 Western Users of SAS Software (WUSS) regional conference.

CONTACT INFORMATION

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

Melissa R. Pfeiffer
PfeifferM@email.chop.edu