

"How Does Your Data Compare?"

SAS's COMPARE PROCEDURE

Jenna Heyen, William M. Mercer, Inc., Deerfield, IL

ABSTRACT

The Compare Procedure is a powerful tool for identifying differences between two data sets because it eliminates often complicated Data Step processing. Comparisons can be made at the observation or variable levels and are controlled by using a wide range of options. The ability to choose among six report types and create output data sets increases the value of the Procedure. At William M. Mercer, PROC COMPARE is used to identify differences between a master data set and new data prior to updating the master, thereby identifying possible bad data in the new data set.

INTRODUCTION

William M. Mercer is one of the leading employee benefits outsourcing companies. Other companies hire Mercer to manage the benefit packages that they provide to their employees, such as health and dental insurance plans, 401(K) plans, and company-sponsored retirement plans. Each month, clients send raw data files that must be cleaned before loading into an integrated database. Consistent, valid, and accurate data is critical to Mercer's mission of providing excellent service to our clients.

As part of the data validation and verification process, we must compare the current month's data to an already existing master data file on a record-by-record basis and report inconsistencies between employee records. To accomplish this sort of task without excessive coding and data manipulation, BASE SAS software provides a procedure that allows the user to compare a master data file with another data file and report differences or similarities between the two: PROC Compare.

This paper gives a brief overview of the Compare procedure (PROC Compare) and covers specifications of PROC Compare, available options, and the different report summaries. It also provides a sample program of how we incorporate the procedure in the validation and integrity processes of our data.

OVERVIEW OF THE COMPARE PROCEDURE

The Compare procedure compares a master or "base" SAS data set to another SAS data set. Provided that the two data sets are somehow different, PROC Compare will do the following:

- 1) *compares the data set attributes;*
- 2) *checks the variables in the data sets, looking for variables in one but not the other; checks to make sure that each variable in one data set matches a variable in the other data set*
- 3) *compares the attributes of matching variables (type, length, labels, formats, and informats);*
- 4) *checks that each observation in one data set matches an observation in the other data set*

The results of the comparisons that the Compare procedure produces can be reported in any combination of the four different types of output. Printed reports, an output data set, messages in the SAS log, and a numeric return code stored in the automatic macro variable &SYSINFO are the output that PROC Compare can produce.

SPECIFICATIONS OF THE COMPARE PROCEDURE

The basic structure of the compare procedure is:

```
PROC COMPARE <OPTIONS>;  
  VAR <VARIABLES>;  
  WITH <VARIABLES>;  
  ID <VARIABLES>;  
  BY <VARIABLES>;  
RUN;
```

The VAR statement allows for specification of the particular variables that are to be compared. If the VAR statement is not used, PROC Compare compares the values of all matching variables. If the variable names in one data set do not match those in the second data set, the WITH statement allows for comparison of these variables. This statement must be used in conjunction with the VAR statement. The first variable listed in the WITH statement must be the first variable listed in the VAR statement, and so on.

The ID statement can be used to match observations in the base data set with corresponding observations in the comparison data set. Also, the ID variables allow for identification of observations on the printed reports and in the output data set.

When using the BY statement, both the master data set and the comparison data set must be sorted by the BY variables or have an appropriate index. The BY variable comparison relies on whether all BY variables are in the comparison data set and if so whether their attributes match those of the BY variables in the base data set.

Along with the basic syntax of PROC Compare, there are numerous options that can be used to further customize this procedure. Depending on the particular output needed the options are broken down into five categories.

Categories	Examples of Available Options
<i>Specific data set names</i>	<i>BASE=, COMPARE=, OUTSTATS=</i>
<i>Specifying the output data set</i>	<i>OUT=, OUTNOEQUAL, OUTDIF, OUTPERCENT</i>
<i>Control comparisons of data values</i>	<i>METHOD=, CRITERION=, NOMISSBASE</i>
<i>Detail in printed reports</i>	<i>PRINTALL, NOPRINT, NOSUMMARY, BRIEF</i>
<i>Listing of variables and observations</i>	<i>LISTALL, LISTOBS, LISTVAR, LISTEQUALVAR</i>

THE PROC COMPARE REPORTS

The results of PROC Compare can be reported in 6 different layouts:

- 1) *Data Set Summary*
- 2) *Variables Summary*
- 3) *Observation Summary*
- 4) *Values Comparison Summary*
- 5) *Value Comparison Results*
- 6) *Summary Statistics*

The default reports for PROC Compare are all but the Summary Statistics report. Options are available that allow for the suppressing or printing of one or all of these reports.

Further discussions of these options, reports and the PROC Compare procedure can be found in the SAS Procedures Guide. SAS/Windows provides extended on-line help for PROC Compare, and there is a sample program in the SAS Sample Library.

DATA VALIDATION AND UPDATING

In the employee benefits outsourcing industry, the need to consistently and accurately check and report client-provided data before adding it to a master data set is critical. Because of this, Mercer needed to develop a single SAS program that would validate new data, print a hardcopy of questionable data, and then add valid records to the master data set. Other specifications that the program needed to meet included reporting of employee status changes and duplicate data records, the deleting of the duplicate records from the new data sets, and backup of the master file prior to updating.

The Compare procedure used in conjunction with the Update statement and PROC Datasets allows us to accomplish these tasks. Being able to pinpoint data problems from month to month enables us to work with the client to debug these problems and increase data integrity.

To make the data more manageable, the data are split into and maintained as smaller data sets based on record-type: human resources, payroll, defined contributions, etc. Since this is a monthly process, all data set names reflect the month that the data was provided. Several macro variables are set up at the beginning of the program to specify the month of the backup file, the current month of the new data feeds, and the month of the duplicate feeds that are expected to be found. In our example, the existing "master" data set (which will be backed up) was created in January and the current month is February.

```
%LET MONTH=JAN;
%LET DUP=FEB;
%LET FEED=FEB;
```

Since we are comparing observations, the new monthly data must be read in from the flatfile provided by the client and sorted by the variable that will be used to make the comparisons between data sets, in this case the employee's social security number (ssn). Since multiple files are being read (each client division provides a separate data set), it is important to keep the filename for tracking purposes should questions arise concerning the data (in this example only one file will be listed). The INFILE option FILENAME= is used to do this.

```
FILENAME DDIR 'BS.EBS.MONTH.DATA';          /*External Files to be read in by SAS*/

DATA &FEED;                                  /*Data Step to input the variables for the HR */
  LENGTH FN DSN $44.;                        /*      record                                */
  RETAIN FILENAME;                          /* Keep the filename for future reference */
  INFILE DDIR FILENAME=FN;
  FILENAME=FN;
  INPUT  @5    SSN          $9.
         @12   DIVLOC      $6.
         @24   STATDATE    MMDDYY8.
         ...
         ...

RUN;
```

Because there should only be one record for each employee, the specification requires extracting the duplicates and keeping the most recent record according to the status date. For this reason, the data set is sorted by descending status date and uses the NODUP option to delete exact duplicates.

```
PROC SORT DATA=&FEED NODUP;                /* Sort data by ID variables of SSN & */
  BY SSN DESCENDING STATDATE;              /* Status Date. Use NODUP to get rid of any */
                                           /* duplicates. Use DESCENDING so that */
                                           /* records are in descending date order */
                                           /* where every variable is the same */

RUN;
```

Although the NODUP option has eliminated exact duplicates, some duplicates remain and we want to output those to a data set to use later. In the code below, we process the data by social security number and use a RETAIN statement to count the number of times each social security number is in the data set. Where the count is one, the record is deleted so that only duplicates are output.

```

DATA PERM.DUP&DUP;
  SET &FEED;
  BY SSN;
  RETAIN SSNNO 0;
  IF FIRST.SSN THEN
    NOSSN=0;

  NOSSN + 1;

  IF LAST.SSN AND SSNNO=1 THEN DELETE;
;

RUN;

```

Because it's possible that a duplicate record resulted from a status date change and is therefore a valid record, PROC Print is used to produce a hardcopy report for visual checking.

```

PROC PRINT DATA=PERM.DUP&DUP;
  Title "Duplicate records from the &Feed feed";
RUN;

```

Only the record with the most recent status date is used in the master file. So next we need to create a data set that has only one instance of each ssn. The following code is an example of deleting duplicate records. Since the data set is sorted by status date in descending order, the most recent record will be the first record. To simplify the lines of code even further, the following data step could be combined with the previous one.

```

DATA &FEED;
  SET &FEED;
  BY SSN;
  IF FIRST.SSN THEN OUTPUT;
  ELSE DELETE;
RUN;

```

Once the data set is reduced to one record per employee social security number, the Compare procedure is run to find differences between the records. The importance of PROC Compare is it not only verifies changes to the records but it can also identify bad data. The next block of code shows how PROC Compare is used for our monthly processing.

```

PROC COMPARE BASE=PERM.HR96 COMPARE=&FEED;
  LISTALL
  ID SSN;
  VAR DIVLOC FNAME LNAME STATREAS STATDATE
  MAXPRINT=(3000,32000);
  OUT=BADDATA OUTNOEQUAL
;

RUN;

```

Looking at the compare statements, the base or master file is the permanent SAS data set PERM.HR96 to which new data is later added. The comparison data set is the monthly data &FEED, which in our example is FEB. The LISTALL option lists all the variables and observations found unequal. This option is equivalent to using the LISTBASEOBS, LISTCOMPOBS, LISTBASEVAR, and the LISTCOMPVAR options. The ID option tells PROC Compare to use the SSN variable as the identifier between the data sets and the VAR option specifies the variables to look at for each SSN.

Depending on the type of output, there are many other options that suppress or print the observations and variables. The MAXPRINT= option is used to increase the number of differences to print between the data sets. The first value is the number of differences to print per variable and the second value is the total number of differences to print. Because our monthly data set is around 60,000 records and the data are known to be dirty, the default of 1000,500 is too small.

Besides the reports produced by PROC Compare, an output data set containing the unequal/different observations is created. This data set is created using the OUT= option. The OUTNOEQUAL option limits this output data set to those observations and variables that are judged unequal. A listing of the output data set is then printed using PROC Print.

```
PROC PRINT DATA=BADDATA;  
    Title "Data from the &Feed feed that are unequal when compared to the Master &Month data set";  
RUN;
```

The reports produced from PROC Compare are the Observation Summary, the Values Comparison Summary, the Data set Summary, the Variables Summary, and the Value Comparison Results. The output data listing is also printed. In the output listing, the character variables are represented by a dot if they are equal or an X if they are not equal. Numeric values are represented by an E if the value is equal. For unequal numeric values, the difference is reported. (See example of the default reports and the output listing at the end of the paper.)

The method for judging equality between numeric values can be specified by the METHOD= option. The default for the METHOD= option is EXACT. The other available options are absolute, relative or percent. If the METHOD is not EXACT then the CRITERION= option must be used along with the METHOD= option. The CRITERION= option further specifies the criterion for judging the equality of the compared numeric values.

The various reports printed by PROC Compare enable us to verify that changes being reported by the client are correct and valid. Also, identifying bad data is another way these reports are valuable. By resolving the inconsistencies of data in the validation process using PROC Compare, the client can report the data correctly for next month's processing.

No editing of the questionable data takes place at this time. The master is intended to contain just what the client provided, excluding duplicates. But before updating the master data set with new information, the existing master is copied to a backup library using PROC Datasets.

```
PROC DATASETS LIB=PERM;  
    CHANGE HR96=HR96&MONTH;  
    COPY OUT=BACKUP MOVE;  
    SELECT HR96&MONTH;  
RUN;
```

Following the backup of the file, the master file is then updated using the Update statement.

```
DATA PERM.HR96;  
    UPDATE BACKUP.HR96&MONTH &FEED;  
    BY SSN;  
RUN;
```

CONCLUSION

The Compare procedure is a powerful tool for the data validation process. For Mercer, it has reduced the need of using complicated and long lines of code and allows us to accurately report inconsistencies between a master data file and new monthly data files. The use of PROC Compare has in turn substantially increased the efficiency of our SAS programming efforts and has contributed to the consistency and accuracy of our data.

TRADEMARK NOTICE

SAS is a registered trademark of the SAS Institute Inc., Cary, NC, USA and other countries.

REFERENCES

SAS Institute Inc., *SAS Language Reference, Version 6*, Cary, NC: SAS Institute Inc.

SAS Institute Inc., *SAS Procedures Guide, Version 6*, Cary, NC: SAS Institute Inc.

Any questions or comments regarding the paper may be directed to the author:

Jenna Heyen
Data Management Programmer Analyst
William M. Mercer, Inc.
Mercer Administration Center
1417 Lake Cook Road
Deerfield, Illinois 60015
Phone: (847) 267-3527
E-mail: Jenna_Heyen@Mercer.com

The SAS System

COMPARE Procedure
Comparison of PERM.HR with WORK.NEW
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
PERM.HR	27JUN96:13:46:33	27JUN96:13:46:34	13	678
WORK.NEW	27JUN96:13:47:49	27JUN96:13:47:49	13	290

Variables Summary

Number of Variables in Common: 13
 Number of ID Variables: 1.
 Number of VAR Statement Variables: 9.

Observation Summary

Observation	Base	Compare	ID
First Obs	1	1	SSN=001489350
First Unequal	1	1	SSN=001489350
Last Unequal	661	279	SSN=370584457
Last Match	677	290	SSN=584226051
Last Obs	678	.	SSN=947284429

Number of Observations in Common: 290.

Number of Observations in PERM.HR but not in WORK.NEW: 388.

Total Number of Observations Read from PERM.HR: 678.

Total Number of Observations Read from WORK.NEW: 290.

Number of Observations with Some Compared Variables Unequal: 24.

Number of Observations with All Compared Variables Equal: 266.

The SAS System

COMPARE Procedure
Comparison of PERM.HR with WORK.NEW
(Method=EXACT)

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 5.
 Number of Variables Compared with Some Observations Unequal: 4.
 Total Number of Values which Compare Unequal: 26.

Variables with Unequal Values

Variable	Type	Len	Ndif	MaxDif
EMPTYTYPE	CHAR	1	1	
HIREDATE	CHAR	8	6	
EMPSTAT	CHAR	2	15	
ELIG	CHAR	1	5	

Value Comparison Results for Variables

SSN	Base Value	Compare Value
	EMPTYTYPE	EMPTYTYPE
	-	-
048442900	E	N

SSN	Base Value	Compare Value
	ELIG	ELIG
	-	-
040449520	N	Y
044786112	Y	N
045628826	Y	N
046688291	Y	N
047847464	Y	N

OBS	_TYPE_	_OBS_	SSN	DIVLOC	FNAME	LNAME	EMPTYTYPE	HIREDATE	EMPSTAT	BARGAIN	ELIG
1	DIF	1	001489350				.		.X	.	.
2	DIF	2	007509377				.		.X	.	.
3	DIF	3	008365990				.	X..	.X	.	.
4	DIF	13	025425370				.		.X	.	.
5	DIF	15	027120956				.		.X	.	.
6	DIF	20	040300458				.		.X	.	.
7	DIF	22	040363617				.		.X	.	.
8	DIF	28	040449520				.		.X	.	X
9	DIF	40	040767365				.		.X	.	.
10	DIF	44	041362955				.		.X	.	.
11	DIF	58	041566503				.		.X	.	.
12	DIF	98	043702451				.		.X	.	.
13	DIF	117	044709438				.	XXXX..XX	..	.	.
14	DIF	119	044786112				.		..	.	X
15	DIF	138	045628826				.		..	.	X
16	DIF	159	046409097				.	...X...	..	.	.
17	DIF	163	046428363				.		.X	.	.
18	DIF	177	046688291				.		..	.	X
19	DIF	184	047227057				.		.X	.	.
20	DIF	204	047847464				.		..	.	X
21	DIF	211	048442900				X		..	.	.
22	DIF	223	048688572				.	XXXX..XX	..	.	.
23	DIF	238	049503514				.	XXXX..XX	..	.	.
24	DIF	279	370584457				.		.X	.	.