A Few Quick and Efficient Ways to Compare Data
Abraham Pulavarti, ICON Clinical Research, North Wales, PA

ABSTRACT
In the fast-paced environment of a clinical research organization, a SAS programmer has to frequently compare clinical trial data present in two data sets to validate results. This paper is intended to explore the various possibilities of comparing data using the COMPARE procedure, and get acquainted with the various techniques available. Once these techniques are learned, the programmer will have a big impact on the data quality and be better equipped to meet the timelines of the clinical study.

Generally double programming is done to validate clinical research data presented in tables or derived data sets. The production programmer and the validation programmer independently write programs and compare the results electronically. Some of the common problems are that the variables do not have the same name or the variable attributes are different or the values of the variables are different.

This paper discusses a macro which can be used on any study on any operating system to get an overall view of the validation results from the comparison of any two data sets. This macro makes use of the return codes generated by the automatic system macro variable SYSINFO.

INTRODUCTION
Typically on any clinical research study, there are derived data sets to be prepared for analysis of the study data. Generally these are double programmed and documented to show that the production dataset and validation dataset are exactly equal. There are approximately X to Y derived data sets on any given study and the lead programmer has to make sure that all these data sets have been validated and each one has a PROC COMPARE output to show that both the data sets are exactly equal.

The output from PROC COMPARE can be lengthy depending on the results. Many times, the output can show that the values are exactly equal in both the data sets, but there can be differences in formats or labels of the variables.

It would be wonderful to have a macro which will compare all the data sets, and give a report of the results and if they are different then show where they are different. This will not only be efficient (since there will be no need to check each of the PROC COMPARE output) but also ensure more compliance to the analysis dataset specifications.

USING THE SYSINFO MACRO VARIABLE
Before we go further, let us understand how the macro variable SYSINFO works. PROC COMPARE stores a return code in the automatic macro variable SYSINFO. The value of the return code provides information about the result of the comparison. By checking the value of SYSINFO after PROC COMPARE has run and before any other step begins, we can use the SYSINFO value from this PROC COMPARE step to determine exactly what is different in these two data sets.

The following table is a key for interpreting the SYSINFO return code from PROC COMPARE. For each of the conditions listed, the associated value is added to the return code if the condition is true. Thus, the SYSINFO return code is the sum of the codes for the applicable conditions. These codes are ordered and scaled to enable a simple check of the degree to which the data sets differ.

All of the columns on the table are self-explanatory except for the last column where the binary values are listed to use in bit-testing. To use the macro presented in this paper, we need to understand a little bit of numeric precision and representation in SAS®. SAS® stores all numeric values using floating point or real binary.

For example, if the SYSINFO macro variable has a value of 14369, then it is represented as ‘0011100000100001’ in binary form. This number is derived using the ‘base 2’ number system.
Base 2 – uses digits 0 and 1. Each of the code values below can be represented as an exponential of 2. For example if the ‘Variable has different informat’ then the code value is 4, which can be represented as $2^2$. Thus the value of 14369 in binary form can be derived as

$$32768 \times 2^{15} + 16384 \times 2^{14} + 4096 \times 2^{13} + 2048 \times 2^{12} + 1024 \times 2^{11} + 512 \times 2^{10} + 256 \times 2^9 + 128 \times 2^8 + 64 \times 2^7 + 32 \times 2^6 + 16 \times 2^5 + 8 \times 2^4 + 4 \times 2^3 + 2 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Taking the number before each parenthesis, we get the binary representation as ‘0011100000100001’.

From the values in the last column of the below table, we can infer that ‘1’b, ‘1…..’b, ‘1………..’b and ‘1………….’b conditions are satisfied. (See the position of 1 in the binary representation).

Using this logic, the macro is built to produce a report of all the 16 different types of differences which can exist in two data sets. If the two data sets are exactly equal then the SYSINFO will have a value of 0.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Condition</th>
<th>Code</th>
<th>Description</th>
<th>Value to use for Bit-Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DSLABEL</td>
<td>1</td>
<td>Data set labels differ</td>
<td>‘1’b</td>
</tr>
<tr>
<td>2</td>
<td>DSTYPE</td>
<td>2</td>
<td>Data set types differ</td>
<td>‘1.’b</td>
</tr>
<tr>
<td>3</td>
<td>INFORMAT</td>
<td>4</td>
<td>Variable has different informat</td>
<td>‘1.’b</td>
</tr>
<tr>
<td>4</td>
<td>FORMAT</td>
<td>8</td>
<td>Variable has different format</td>
<td>‘1…’b</td>
</tr>
<tr>
<td>5</td>
<td>LENGTH</td>
<td>16</td>
<td>Variable has different length</td>
<td>‘1…..’b</td>
</tr>
<tr>
<td>6</td>
<td>LABEL</td>
<td>32</td>
<td>Variable has different label</td>
<td>‘1…….’b</td>
</tr>
<tr>
<td>7</td>
<td>BASEOBS</td>
<td>64</td>
<td>Base data set has observation not in comparison</td>
<td>‘1…….’b</td>
</tr>
<tr>
<td>8</td>
<td>COMPOBS</td>
<td>128</td>
<td>Comparison data set has observation not in base</td>
<td>‘1…….’b</td>
</tr>
<tr>
<td>9</td>
<td>BASEBY</td>
<td>256</td>
<td>Base data set has BY group not in comparison</td>
<td>‘1…….’b</td>
</tr>
<tr>
<td>10</td>
<td>COMPBY</td>
<td>512</td>
<td>Comparison data set has BY group not in base</td>
<td>‘1…….’b</td>
</tr>
<tr>
<td>11</td>
<td>BASEVAR</td>
<td>1024</td>
<td>Base data set has variable not in comparison</td>
<td>‘1………..’b</td>
</tr>
<tr>
<td>12</td>
<td>COMPVAR</td>
<td>2048</td>
<td>Comparison data set has variable not in base</td>
<td>‘1………..’b</td>
</tr>
<tr>
<td>13</td>
<td>VALUE</td>
<td>4096</td>
<td>A value comparison was unequal</td>
<td>‘1………..’b</td>
</tr>
<tr>
<td>14</td>
<td>TYPE</td>
<td>8192</td>
<td>Conflicting variable types</td>
<td>‘1………..’b</td>
</tr>
<tr>
<td>15</td>
<td>BYVAR</td>
<td>16384</td>
<td>BY variables do not match</td>
<td>‘1………..’b</td>
</tr>
<tr>
<td>16</td>
<td>ERROR</td>
<td>32768</td>
<td>Fatal error: comparison not done</td>
<td>‘1………..’b</td>
</tr>
</tbody>
</table>
APPLICATION OF THE SYSINFO MACRO VARIABLE
For quick checking the following code can be used to see the differences in any two data sets, for example if we want to check whether two data sets contain the same variables, observations and values but do not care about labels, formats and so on then we can use the following SAS® code,

```
PROC COMPARE base=one compare=two;
run;
%macro test;
  %if &sysinfo >= 64 %then
    %do;
      %put 'WARNING: There are differences in values or variables or observations ';
    %end;
%mend test;
%test;
```

The log can be checked after running this code, to see if the warning message exists. If yes then there are differences in the datasets in relation to the variable values or variables or observations.

CREATING A MACRO USING THE SYSINFO MACRO VARIABLE
Using the DATA step bit-testing feature (as discussed in the previous page), we can examine individual bits in the SYSINFO value to check for specific conditions. A macro has been created to use the SYSINFO value and get a consolidated report of all the PROC COMPARE results for all the data sets in a study. The macro uses the binary representation of the SYSINFO value; these values are shown in the table above (last column). This macro can be adapted to any study and to any number of comparisons.

```
/* Assigning libname for 2 folders, production and validation. */
libname prod 'X:\YYY\YYYY\YYYYYY';
libname valid 'X:\QQQ\QQQQ\QQQQQ';

/* Macro to compare data sets and get the value of system macro variable &sysinfo for each PROC COMPARE. Then using PROC APPEND, &sysinfo values are stored in a data set for each PROC COMPARE. */
%macro acomp(base=, compare=, name=);
PROC COMPARE base= &base. compare= &compare. noprint;
run;
%let CompareSysinfo= &sysinfo;
DATA valid;
  length item $20; 
  sysinfo= &CompareSysinfo. ;
  item= "&name.";
run;
PROC APPEND base= alldata data= valid;
run;
%mend acomp;

%acomp(base= prod.adae, compare= valid.adae, name= ADEA);
%acomp(base= prod.adcm, compare= valid.adcm, name= ADECM);
%acomp(base= prod.adds, compare= valid.adds, name= ADDS);
```
/* Just add as many datasets as you need by calling the macro acomp for each comparison */

/* Using bit-testing, the SYSINFO value is interpreted, and the differences in each of the PROC COMPARE are captured in a data set */

DATA final;
settadata;
attribmsglength=$60;
ifsysinfo='1'b then do;
msg='Data set labels differ';
output;
end;
ifsysinfo='1.'b then do;
msg='Data set types differ';
output;
end;
ifsysinfo='1..'b then do;
msg='Variable has different informat';
output;
end;
ifsysinfo='1....'b then do;
msg='Variable has different format';
output;
end;
ifsysinfo='1......'b then do;
msg='Variable has different length';
output;
end;
ifsysinfo='1.......'b then do;
msg='Base data set has observation not in comparison';
output;
end;
ifsysinfo='1........'b then do;
msg='Comparison data set has observation not in base';
output;
end;
ifsysinfo='1.........'b then do;
msg='Base data set has BY group not in comparison';
output;
end;
ifsysinfo='1..........b then do;
msg='Comparison data set has BY group not in base';
output;
end;
ifsysinfo='1...........'b then do;
msg='Base data set has variable not in comparison';
output;
end;
ifsysinfo='1............b then do;
msg='Comparison data set has variable not in base';
output;
end;
ifsysinfo='1.............b then do;
msg='A value comparison was unequal';
CONCLUSIONS

In summary, the techniques learned through this paper will enable a clinical SAS programmer to be more productive and efficient on the job which leads to better results and easier documentation. This macro can be customized to suit your needs by adding a 'by' parameter to sort the two datasets in a particular order before the COMPARE procedure or adding time stamp information to the report to see whether the validation data set was created after the production data set.

REFERENCES


Reiss, Maria Y. 2006 “Dare to Compare Tailoring PROC COMPARE Output” Proceeding of the South East SAS Users Group Conference 2006, Atlanta, GA.

SAS Institute Inc., Technical Paper “Numeric Precision 101”

Posting from Mofeel Groups comp.soft-sys.sas.
http://www.mofeel.net/1169-comp-softsys-sas/3928.aspx

ACKNOWLEDGMENTS
I thank my Lord Jesus Christ to help me write this paper.

I would like to thank Lex Fennel, whose paper I extensively referred, to get the concept of the process. I express my gratitude to Tony Pisegna for the valuable insight and feedback. I also thank Vandana Rajanbabu for reviewing the paper.

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 registered trademarks or trademarks of their respective companies.

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

Abraham Pulavarti
ICON Clinical Research
212 Church Road
North Wales PA 19454
Tel: 215.616.3020 / 646.309.0582
Fax: 215.616.8685
Email: abraham.pulavarti@iconplc.com / abrahamtp@yahoo.com
Web: www.iconplc.com