PharmaSUG 2014 - Paper CC22

A SAS® Macro Tool for Visualizing Data Comparison Results in an Intuitive Way

Hui Wang, Biogen Idec, Boston, MA
Weizhen Ying, Biogen Idec, Boston, MA

ABSTRACT

In clinical data analysis, PROC COMPARE is widely used in data quality control. However, sometimes the results of this procedure can be quite challenging to understand. For example, it only displays the first 20 characters in value comparison output. Consequently, character mismatches in lengthy text strings may not be seen directly. In addition, it shows mismatches separately and does not simultaneously show values of variables that might be relevant to mismatches, which leads to difficulty in figuring out why mismatches happen.

This paper intends to introduce a SAS macro tool which is based on PROC COMPARE but gives an intuitive comparison report. First of all, this macro juxtaposes mismatched values vertically and thus allows the display of whole variable content. Secondly, the macro can extract unique rows from datasets compared and then tells directly which rows are missing in which dataset. Finally, it enables the display of values of pre-specified variables along the same row of mismatches. In this way, the programmer can intuitively observe the possible involvement of other variables in causing mismatches.

In short, this macro, with its user-friendly output, functions as an extension tool of PROC COMPARE and is able to improve efficiency of clinical data analysis.

INTRODUCTION

PROC COMPARE is a powerful SAS® procedure capable of comparing two datasets at dataset level, variable level and value level. In the field of clinical programming, its output is usually used to evaluate the quality of validation. However, this procedure can only give a simple vertical listing of inconsistent items between two datasets. When discrepancy occurs to multiple variables and multiple observations, the long mismatch list with an endless-like scroll bar could be a programmer's nightmare to read and further figure out what happened. In addition, there are blind spots in the PROC COMPARE output where programmers are easy to fall in the trap, especially those who simply scan the output for a note at the bottom saying "No unequal values were found. All values compared are exactly equal." These cases are described in detail in the article "Don't Get Blinded by PROC COMPARE" from PharmaSUG 2013.

That is why we developed a SAS utility macro tool named as COMPARE. This tool still uses PROC COMPARE as the basis for dataset comparison. But it can transform the output of PROC COMPARE to generate a more understandable report to overcome the downside of this SAS procedure.

MACRO PARAMETERS AND OUTPUT

Basically, this macro tool looks like as below:

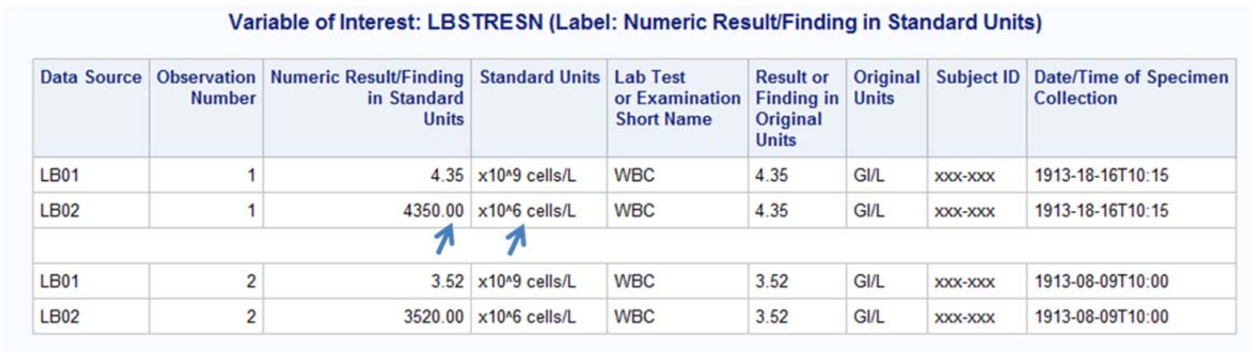
```
%COMPARE(BASE=, COMPARE=, WHERE=, VARINT=, OTHERVAR=, IDVAR=, CRITERIA=, CLEAN=);
```

First of all, it needs at least two parameters BASE and COMPARE to specify datasets to be compared. All variables will be compared if all other parameters are left null. Furthermore, the parameter WHERE refers to the condition for subsetting both BASE and COMPARE datasets. For example, if we only want to review data of a specific subject, it will be useful. Secondly, the parameter VARINT refers to the variable at which you want to take a closer look after initial comparison. If we also want to review other variables simultaneously to facilitate the review of variable of interest, we can assign names of the variables to the parameter OTHERVAR which will list the value next to the VARINT. Thirdly, the parameter IDVAR indicates the variable used as an ID variable in PROC COMPARE, which is necessary for the macro to grab unique observations from both datasets. The parameter CRITERIA sets the comparison criteria as that from PROC COMPARE. Lastly, if we want to remove all intermediate datasets, we can assign 'Y' to the parameter CLEAN.

A typical macro output to identify mismatches is shown in Display 1. In this example, we will compare two laboratory datasets LB01 and LB02. The comparison report title points out the variable that is being examined: LBSTRESN (Numeric Result in Standard Units). Below the title, the comparison result is horizontally structured. In each comparison pair separated by a blank line, the top row is from BASE dataset while the bottom row is from COMPARE dataset. If we look at the report vertically, the first and second columns, Data Source and Observation Number, indicate location of mismatches. The third column is about the VARINT, which is followed by OTHERVAR variables. Clearly shown in the report, both Standard Units (LBSTRESU) and Numeric Result/Finding in Standard Units

(LBSTRESN) value do not match. Base on this information, we can easily puzzle out that conversion factor is not consistent between the two parallel programming practices.

```
%COMPARE(BASE=LB01, COMPARE=LB02, VARINT=LBSTRESN,
         OTHERVAR=LBSTRESU LBTESTCD LBORRES LBORRESU SUBJID LBDTC);
```



Variable of Interest: LBSTRESN (Label: Numeric Result/Finding in Standard Units)

Data Source	Observation Number	Numeric Result/Finding in Standard Units	Standard Units	Lab Test or Examination Short Name	Result or Finding in Original Units	Original Units	Subject ID	Date/Time of Specimen Collection
LB01	1	4.35	x10^9 cells/L	WBC	4.35	GI/L	xxx-xxx	1913-18-16T10:15
LB02	1	4350.00	x10^6 cells/L	WBC	4.35	GI/L	xxx-xxx	1913-18-16T10:15
LB01	2	3.52	x10^9 cells/L	WBC	3.52	GI/L	xxx-xxx	1913-08-09T10:00
LB02	2	3520.00	x10^6 cells/L	WBC	3.52	GI/L	xxx-xxx	1913-08-09T10:00

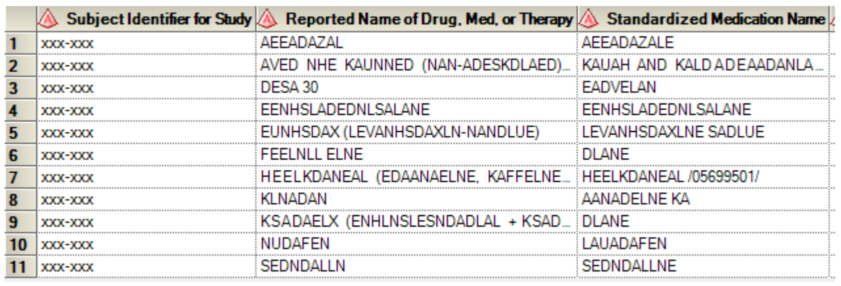
Display 1. A typical comparison report created by MACRO COMPARE

Next let us talk about the advantages of this macro tool over PROC COMPARE by showing examples.

CASE #1:

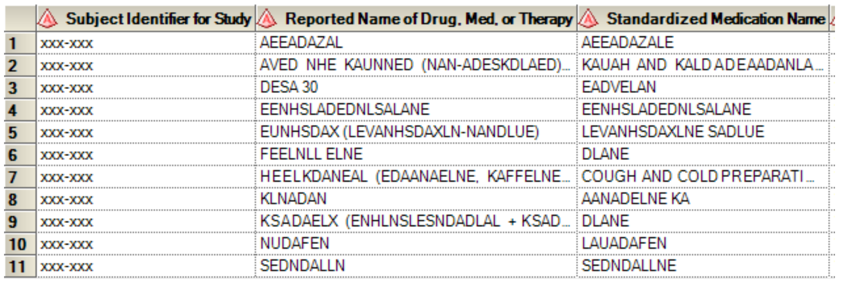
Our first example is about comparison of two concomitant medication datasets CM01 and CM02. They look identical if we eyeball the data (Display 2). However, the output of PROC COMPARE unambiguously shows that the variable CMTRT has one mismatch (Display 3). Due to the limitation of this procedure, although we know which row contains the difference, we are not able to review the mismatch directly. That CMTRT value is long and the mismatch is masked. Then, let us try our macro tool. Since the comparison result is listed horizontally and thus leaves enough space to display the whole variable content, the extra blank from CM2, which is indicated by arrow, stands out (Display 3).

CM01



	Subject Identifier for Study	Reported Name of Drug, Med, or Therapy	Standardized Medication Name
1	xxx-xxx	AEEADAZAL	AEEADAZALE
2	xxx-xxx	AVED NHE KAUNNED (NAN-ADESKDLAED)...	KAUAH AND KALD ADEAADANLA...
3	xxx-xxx	DESA 30	EADVELAN
4	xxx-xxx	EENHSLADEDNLSALANE	EENHSLADEDNLSALANE
5	xxx-xxx	EUNHSDAX (LEVANHSDAXLN-NANDLUE)	LEVANHSDAXLNE SADLUE
6	xxx-xxx	FEELNLL ELNE	DLANE
7	xxx-xxx	HEELKDANEAL (EDAANAELNE, KAFFELNE...	HEELKDANEAL /05699501/
8	xxx-xxx	KLNADAN	AANADELNE KA
9	xxx-xxx	KSADAELX (ENHLNSLESNDADLAL + KSAD...	DLANE
10	xxx-xxx	NUDAFEN	LAUADAFEN
11	xxx-xxx	SEDNDALLN	SEDNDALLNE

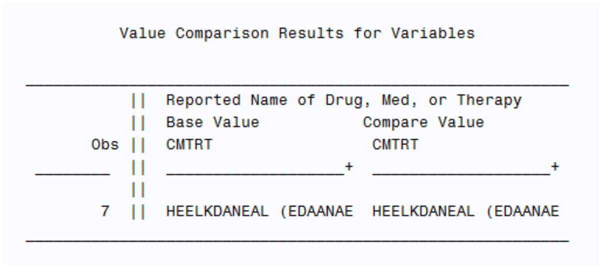
CM02



	Subject Identifier for Study	Reported Name of Drug, Med, or Therapy	Standardized Medication Name
1	xxx-xxx	AEEADAZAL	AEEADAZALE
2	xxx-xxx	AVED NHE KAUNNED (NAN-ADESKDLAED)...	KAUAH AND KALD ADEAADANLA...
3	xxx-xxx	DESA 30	EADVELAN
4	xxx-xxx	EENHSLADEDNLSALANE	EENHSLADEDNLSALANE
5	xxx-xxx	EUNHSDAX (LEVANHSDAXLN-NANDLUE)	LEVANHSDAXLNE SADLUE
6	xxx-xxx	FEELNLL ELNE	DLANE
7	xxx-xxx	HEELKDANEAL (EDAANAELNE, KAFFELNE...	COUGH AND COLD PREPARATI...
8	xxx-xxx	KLNADAN	AANADELNE KA
9	xxx-xxx	KSADAELX (ENHLNSLESNDADLAL + KSAD...	DLANE
10	xxx-xxx	NUDAFEN	LAUADAFEN
11	xxx-xxx	SEDNDALLN	SEDNDALLNE

Display 2. Concomitant medication datasets to be examined

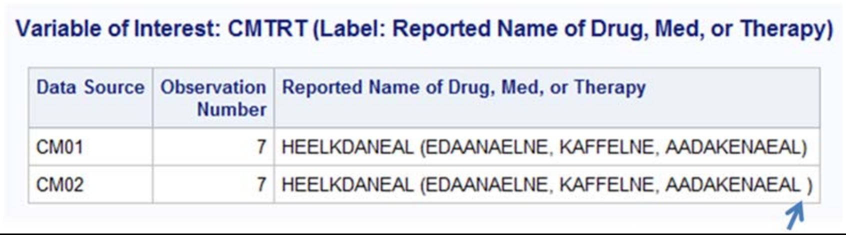
```
PROC COMPAREBASE=CM01 COMPARE=CM02;
  VAR CMTRT;
RUN;
```



```
                Value Comparison Results for Variables

 __________________________________________________________
            ||  Reported Name of Drug, Med, or Therapy
            ||  Base Value              Compare Value
       Obs  ||  CMTRT                    CMTRT
 ________   ||  ___________________+   ___________________+
            ||
         7  ||  HEELKDANEAL (EDAANAE   HEELKDANEAL (EDAANAE
 __________________________________________________________
```

```
%COMPARE(BASE=CM01, COMPARE=CM02, VARINT=CMTRT);
```



Variable of Interest: CMTRT (Label: Reported Name of Drug, Med, or Therapy)

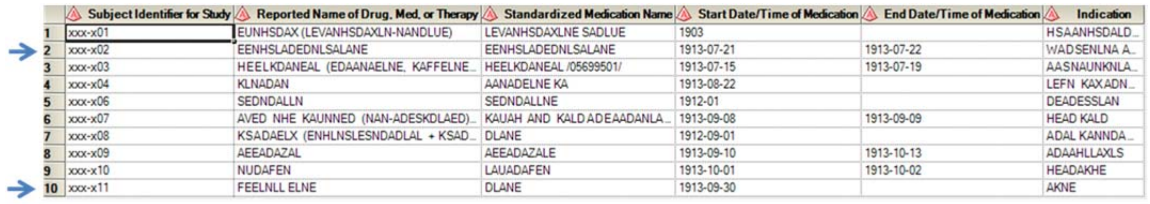
Data Source	Observation Number	Reported Name of Drug, Med, or Therapy
CM01	7	HEELKDANEAL (EDAANAELNE, KAFFELNE, AADAKENAEAL)
CM02	7	HEELKDANEAL (EDAANAELNE, KAFFELNE, AADAKENAEAL)

Display 3. Outputs from PROC COMPARE and MACRO COMPARE

CASE #2:

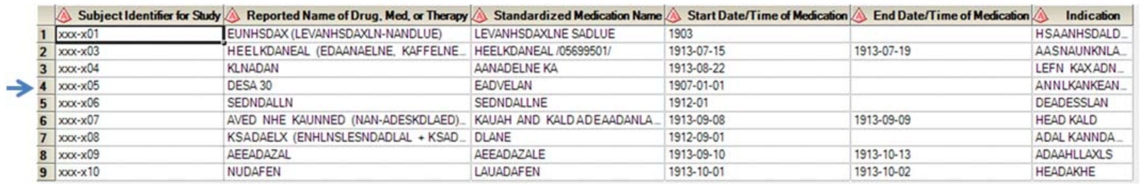
Let us explore the second example by using another two concomitant medication datasets CM03 and CM04, where observation numbers of the BASE dataset and the COMPARE dataset differ but there are no mismatches besides that (Display 4, arrow indicates unique rows in each dataset). We first use PROC COMPARE and specify SUBJID as the ID variable. As every programmer can guess, PROC COMPARE only says “No unequal values were found. All values compared are exactly equal.”. This statement is right for sure. But it does not provide the information which we are looking for. We expect to find out which unique observations come from BASE dataset and which unique observations come from COMPARE dataset. Again, we seek help from our macro tool. As shown in Display 5, three rows are extracted. The first two columns demonstrate where each row comes from. Explicitly, the BASE dataset contributes two unique records and the COMPARE dataset contributes one.

CM03



	Subject Identifier for Study	Reported Name of Drug, Med, or Therapy	Standardized Medication Name	Start Date/Time of Medication	End Date/Time of Medication	Indication
1	xxx-x01	EUNHSDAX (LEVANHSDAXLN-NANDLUE)	LEVANHSDAXLNE SADLUE	1903		HSAANHSDALD...
2	xxx-x02	EENHSLADEDNLSALANE	EENHSLADEDNLSALANE	1913-07-21	1913-07-22	WADSENLNA A...
3	xxx-x03	HEELKDANEAL (EDAANAELNE, KAFFELNE...	HEELKDANEAL /05699501/	1913-07-15	1913-07-19	AASNAUNKNLA...
4	xxx-x04	KLNADAN	AANADELNE KA	1913-08-22		LEFN KAXADN...
5	xxx-x06	SEDNDALLN	SEDNDALLNE	1912-01		DEADESSLAN
6	xxx-x07	AVED NHE KAUNNED (NAN-ADESKDLAED)...	KAUAH AND KALD ADEAADANLA...	1913-09-08	1913-09-09	HEAD KALD
7	xxx-x08	KSADAELX (ENHLNSLESNDADLAL + KSAD...	DLANE	1912-09-01		ADAL KANNDA...
8	xxx-x09	AEEADAZAL	AEEADAZALE	1913-09-10	1913-10-13	ADAAHLLAXLS
9	xxx-x10	NUDAFEN	LAUADAFEN	1913-10-01	1913-10-02	HEADAKHE
10	xxx-x11	FEELNLL ELNE	DLANE	1913-09-30		AKNE

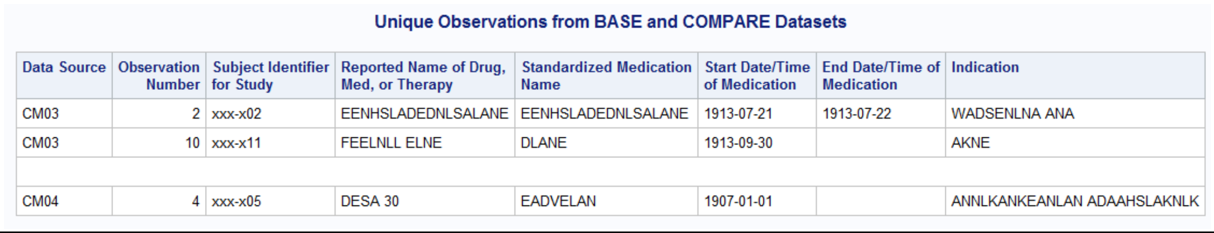
CM04



	Subject Identifier for Study	Reported Name of Drug, Med, or Therapy	Standardized Medication Name	Start Date/Time of Medication	End Date/Time of Medication	Indication
1	xxx-x01	EUNHSDAX (LEVANHSDAXLN-NANDLUE)	LEVANHSDAXLNE SADLUE	1903		HSAANHSDALD...
2	xxx-x03	HEELKDANEAL (EDAANAELNE, KAFFELNE...	HEELKDANEAL /05699501/	1913-07-15	1913-07-19	AASNAUNKNLA...
3	xxx-x04	KLNADAN	AANADELNE KA	1913-08-22		LEFN KAXADN...
4	xxx-x05	DESA 30	EADVELAN	1907-01-01		ANNLKANKEAN...
5	xxx-x06	SEDNDALLN	SEDNDALLNE	1912-01		DEADESSLAN
6	xxx-x07	AVED NHE KAUNNED (NAN-ADESKDLAED)...	KAUAH AND KALD ADEAADANLA...	1913-09-08	1913-09-09	HEAD KALD
7	xxx-x08	KSADAELX (ENHLNSLESNDADLAL + KSAD...	DLANE	1912-09-01		ADAL KANNDA...
8	xxx-x09	AEEADAZAL	AEEADAZALE	1913-09-10	1913-10-13	ADAAHLLAXLS
9	xxx-x10	NUDAFEN	LAUADAFEN	1913-10-01	1913-10-02	HEADAKHE

Display 4. Concomitant medication datasets to be examined

```
%COMPARE(BASE=CM03, COMPARE=CM04, IDVAR=SUBJID);
```



Unique Observations from BASE and COMPARE Datasets

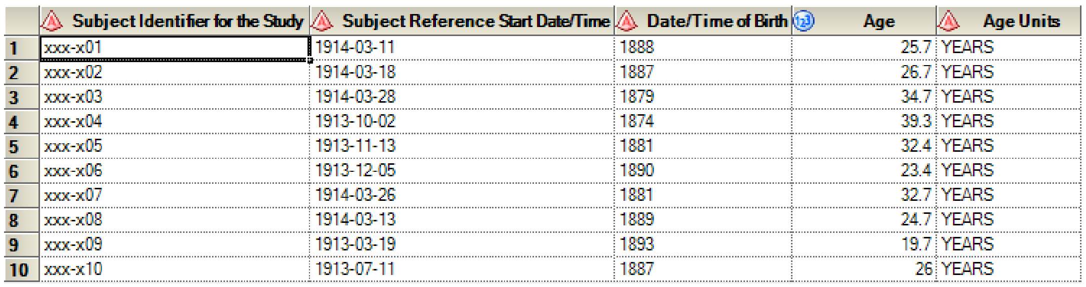
Data Source	Observation Number	Subject Identifier for Study	Reported Name of Drug, Med, or Therapy	Standardized Medication Name	Start Date/Time of Medication	End Date/Time of Medication	Indication
CM03	2	xxx-x02	EENHSLADEDNLSALANE	EENHSLADEDNLSALANE	1913-07-21	1913-07-22	WADSENLNA ANA
CM03	10	xxx-x11	FEELNLL ELNE	DLANE	1913-09-30		AKNE
CM04	4	xxx-x05	DESA 30	EADVELAN	1907-01-01		ANNLKANKEANLAN ADAAHSLAKNLK

Display 5. Output of MACRO COMPARE

CASE #3:

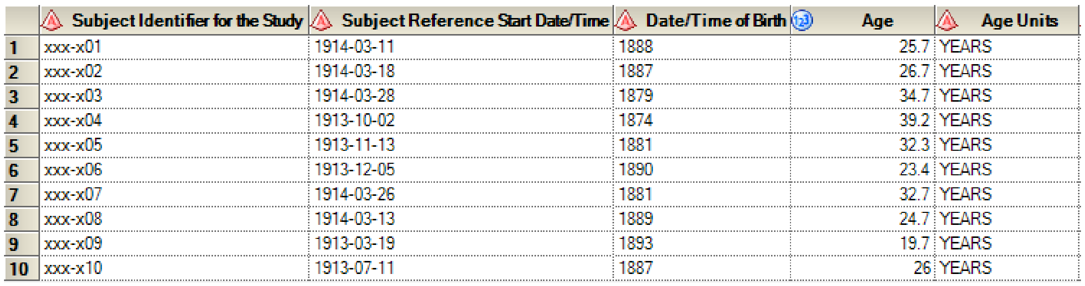
Let us go a little further to see an example with derived variable (Display 6) in two demographic datasets DM01 and DM02. In this case, AGE is derived based on values of birth year and reference start date per certain pre-defined algorithm. As always, we try PROC COMPARE first to see if age is calculated correctly and then see two mismatches. What causes these, original data or algorithm? With the help of macro COMPARE, we are able to decompose the issue. First, we assign AGE to the macro parameter VARINT to list all the discrepancy for variable of interest. We also want to check if there is any discrepancy for relevant BRTHDTC and RFSTDTC that are used to derive AGE so that we assign BRTHDTC and RFSTDTC in addition to SUBJID (subject ID) to the macro parameter OTHERVAR. In Display 7, the comparison pairs suggest that age is different when BRTHDTC and RFSTDTC values match. Clearly, at least one of the programming practices messes up the algorithm. With this feature, a programmer would feel more comfortable to navigate complicated situations where lots of mismatches occur to multiple variables and multiple observations in big datasets.

DM01



	Subject Identifier for the Study	Subject Reference Start Date/Time	Date/Time of Birth	Age	Age Units
1	xxx-x01	1914-03-11	1888	25.7	YEARS
2	xxx-x02	1914-03-18	1887	26.7	YEARS
3	xxx-x03	1914-03-28	1879	34.7	YEARS
4	xxx-x04	1913-10-02	1874	39.3	YEARS
5	xxx-x05	1913-11-13	1881	32.4	YEARS
6	xxx-x06	1913-12-05	1890	23.4	YEARS
7	xxx-x07	1914-03-26	1881	32.7	YEARS
8	xxx-x08	1914-03-13	1889	24.7	YEARS
9	xxx-x09	1913-03-19	1893	19.7	YEARS
10	xxx-x10	1913-07-11	1887	26	YEARS

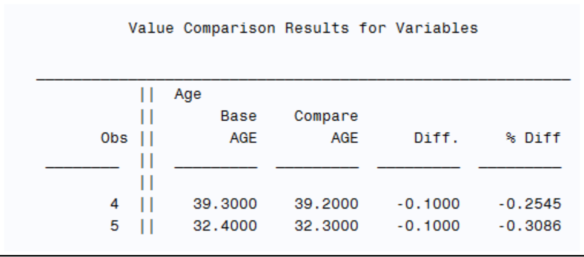
DM02



	Subject Identifier for the Study	Subject Reference Start Date/Time	Date/Time of Birth	Age	Age Units
1	xxx-x01	1914-03-11	1888	25.7	YEARS
2	xxx-x02	1914-03-18	1887	26.7	YEARS
3	xxx-x03	1914-03-28	1879	34.7	YEARS
4	xxx-x04	1913-10-02	1874	39.2	YEARS
5	xxx-x05	1913-11-13	1881	32.3	YEARS
6	xxx-x06	1913-12-05	1890	23.4	YEARS
7	xxx-x07	1914-03-26	1881	32.7	YEARS
8	xxx-x08	1914-03-13	1889	24.7	YEARS
9	xxx-x09	1913-03-19	1893	19.7	YEARS
10	xxx-x10	1913-07-11	1887	26	YEARS

Display 6. Demographic datasets to be examined

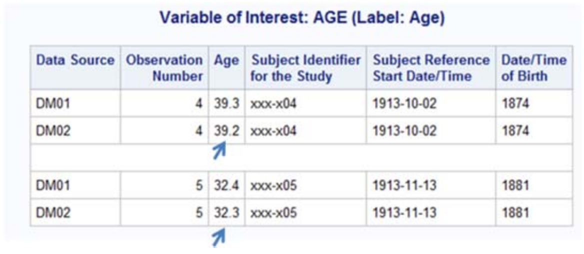
```
PROC COMPARE BASE=DM01 COMPARE=DM02;
  VAR AGE;
RUN;
```



```
                 Value Comparison Results for Variables

 __________________________________________________________
               ||  Age
               ||       Base     Compare
          Obs  ||        AGE         AGE      Diff.    % Diff
   ________    ||  _________   _________  _________  _________
               ||
            4  ||    39.3000     39.2000    -0.1000    -0.2545
            5  ||    32.4000     32.3000    -0.1000    -0.3086
```

```
%COMPARE(BASE=DM01, COMPARE=DM02,
         VARINT=AGE, OTHERVAR=SUBJID
         RFSTDTC BRTHDTC);
```



Variable of Interest: AGE (Label: Age)

Data Source	Observation Number	Age	Subject Identifier for the Study	Subject Reference Start Date/Time	Date/Time of Birth
DM01	4	39.3	xxx-x04	1913-10-02	1874
DM02	4	39.2	xxx-x04	1913-10-02	1874
DM01	5	32.4	xxx-x05	1913-11-13	1881
DM02	5	32.3	xxx-x05	1913-11-13	1881

Display 7. Outputs from PROC COMPARE and MACRO COMPARE

CONCLUSION

So far, we have seen that the macro COMPARE can 1) show whole content of variable examined; 2) directly demonstrate unique rows from different datasets compared; 3) evaluate impact of other variables on variable of interest. Certainly, it can deal with intricate situations as well. For example, if BASE and COMPARE datasets have both mismatches and unequal observation numbers, it can write two separate reports to help the programmer figure out what is going on.

In conclusion, this macro tool makes it easy to visualize and understand the comparison results created by PROC COMPARE and thus facilitates programming efficiency.

REFERENCE

Horstman, Joshua and Muller, Roger
"Don't Get Blinded by PROC COMPARE"
Proceeding of the Pharmaceutical SAS Users Group, PharmaSUG 2013, Paper CC36

ACKNOWLEDGMENT

We thank Arthur Collins for reviewing the manuscript and providing comments.

CONTACT INFORMATION

Your comments or suggestions will be highly appreciated. Please contact the authors at:

Hui Wang
Biogen Idec
hui.wang@biogenidec.com

Weizhen Ying
Biogen Idec
maggie.ying@biogenidec.com

Appendix SAS code of the macro COMPARE

```
%macro compare(base=, compare=, where=, varint=, othervar=, idvar=, criteria=,
               clean=y);

data _null_;  ***Manipulate macro parameters for next steps***;
  if strip("&criteria") ^= ' ' then do; call symputx('cri', "criteria=" ||
    "&criteria.", 'L'); end;
  else do; call symputx('cri', ' ', 'L'); end;
  if strip("&varint") = ' ' then do; call symputx('varint', '_all_', 'L'); end;
run;

data _null_;   ***Obtain dataset labels***;

  aa=open("&base.");
  call symputx("base_label", attrc(aa, 'LABEL'), 'L');
  bb=close(aa);

  aa=open("&compare.");
  call symputx("compare_label", attrc(aa, 'LABEL'), 'L');
  bb=close(aa);

run;

data _null_;  ***Separate library name and dataset name***;
  if index("&base", '.') then do;
    call symputx("lib_base", scan("&base", 1, '.'), 'L');
    call symputx("base", scan("&base", 2, '.'), 'L');
  end;
  else call symputx("lib_base", 'WORK', 'L');
run;

data _null_;  ***Separate library name and dataset name***;
  if index("&compare", '.') then do;
    call symputx("lib_compare", scan("&compare", 1, '.'), 'L');
    call symputx("compare", scan("&compare", 2, '.'), 'L');
  end;
  else call symputx("lib_compare", 'WORK', 'L');
run;

proc sql noprint; ***Variable list from BASE dataset***;
  create table base_var as
  select distinct strip(name) as name, type
  from dictionary.columns
  where strip(upcase(memname)) = upcase("&base") and strip(upcase(libname)) =
        upcase("&lib_base")
  order by name;
quit;

proc sql noprint; ***Variable list from COMPARE dataset***;
  create table comp_var as
  select distinct strip(name) as name, type
  from dictionary.columns
  where strip(upcase(memname)) = upcase("&compare") and strip(upcase(libname)) =
        upcase("&lib_compare")
  order by name;
quit;

***find common variables between two datasets***;

data common_var (drop=type source) diff_var (drop=type);
```

```
  merge base_var (in=aaa) comp_var (in=bbb);
  by name type;
  attrib source label='Data Source' length=$20;
  if aaa and bbb then output common_var;
  if (aaa and not bbb) or (not aaa and bbb) then do;
    if aaa and not bbb then source="&base.";
    if not aaa and bbb then source="&compare.";
    output diff_var;
  end;
run;

data _null_; ***Output variables missing in BASE of COMPARE dataset******;
  aa=open("diff_var");
  bb=attrn(aa, 'ANY');
  cc=close(aa);
  if bb = 1 then call execute('

    title "Variables Missing in BASE or COMPARE Dataset";
    proc print data=diff_var label;
    run;
    title;

  ');
run;

proc sql noprint;  ***Common variable list macro variable***;
  select distinct strip(name) into :varlist separated by ' '
  from dictionary.columns
  where strip(memname) = upcase("&base") and
        strip(libname) = upcase("&lib_base") and
        strip(name) in (select distinct strip(name) from common_var);
quit;

data base_tmp;  ***Temporary BASE dataset used for comparison***;
  retain &varlist.;
  set &lib_base..&base;
  n=_N_;
  ori_n=_N_;
  keep &varlist. n ori_n;
run;

data comp_tmp;  ***Temporary COMPARE dataset used for comparison***;
  retain &varlist.;
  set &lib_compare..&compare;
  n=_N_;
  ori_n=_N_;
  keep &varlist. n ori_n;
run;

proc sql noprint;
  create table comp01 as
  select libname, memname, memlabel, nobs, obslen, nvar, num_character, num_numeric
  from dictionary.tables
  where (strip(upcase(memname)) = upcase("&base") or
        strip(upcase(memname)) = upcase("&compare")) and
        strip(libname) = 'WORK';
quit;

***Find unique rows form BASE and COMPARE datasets***;

data _null_;
  if strip("&idvar.") ^= ' ' then call execute('
    proc sort data=base_tmp;
      by &idvar.;
    run;

    proc sort data=comp_tmp;
      by &idvar.;
    run;

    proc compare base=base_tmp (drop=ori_n) compare=comp_tmp (drop=ori_n) out=diffu01
outbase outcomp outnoequal novalues;
      id &idvar.;
    run;

    data diffu02;
      set diffu01;
      length idvar tmp1 $200;
      idvar=strip(&idvar.);
      tmp1=lag(idvar);
      n=_N_;
    run;

    proc sort data=diffu02 out=diffu03;
      by descending n;
    run;

    data diffu04;
      set diffu03;
      length tmp2 $200;
      tmp2=lag(idvar);
    run;

    proc sort data=diffu04 out=diffu05;
      by n;
    run;

    data diffu06;
      set diffu05;
      if strip(_type_) = "BASE" then do;
        _type_=upcase("&base.");
        if strip(idvar) = strip(tmp2) then delete;
      end;
      if strip(_type_) = "COMPARE" then do;
        _type_=upcase("&compare.");
        if strip(idvar) = strip(tmp1) then delete;
      end;
    run;

    data diffu06;
      set diffu06;
      label _type_="Data Source";
      attrib source_ length=$20;
      if strip(_type_) = upcase("&base.") then source_="BASE";
      if strip(_type_) = upcase("&compare.") then source_="COMPARE";
    run;

    proc sort data=diffu06;
      by source_ _type_ &idvar.;
    run;

    proc sql noprint;
      select count(strip(_type_)) into :tot trimmed
      from diffu06
      where strip(_type_) = upcase("&base.");
    quit;

  ');
run;
***Print out the dataset with unique rows from either or both datasets***;

data _null_;
  if exist('diffu06', 'DATA') then do;
    aa=open('diffu06');
    bb=attrn(aa, 'ANY');
    cc=close(aa);
    if bb = 1 then call execute('

      title "Unique Observations from BASE and COMPARE Datasets";
      proc print data=diffu06 (drop=idvar tmp1 tmp2 n source_) label noobs
           blankline=(count=&tot.);
      run;

    ');
  end;
run;

****End of unique observation extraction*****;

***Find mismatches***;

data _null_; ***At first, remove unique rows based on ID variable values***;
  if exist('diffu06', 'DATA') then do;
    call execute('

      data diffu06;
        set diffu06;
        n=_obs_;
      run;

      proc sort data=diffu06;
        by n;
      run;

      proc sort data=base_tmp;
        by n;
      run;

      proc sort data=comp_tmp;
        by n;
      run;

      data base_tmp;
        merge base_tmp diffu06 (where=(strip(_type_)=upcase("&base."))
                                keep=_type_ n in=bbb);
        by n;
        if not bbb;
        drop _type_;
      run;

      data base_tmp;
        set base_tmp;
        n=_N_;
      run;

      data comp_tmp;
        merge comp_tmp diffu06 (where=(strip(_type_)=upcase("&compare."))
                                keep=_type_ n in=bbb);
        by n;
        if not bbb;
        drop _type_;
      run;

      data comp_tmp;
        set comp_tmp;
        n=_N_;
      run;

    ');
    call symputx('output', 'noprint', 'L');
  end;
  else call symputx('output', 'novalues', 'L');
run;

data _null_;
  if upcase(strip("&varint.")) = '_ALL_' then
     call symputx('varint', tranwrd("&varlist.", "&idvar.", ' '), 'L');
run;

proc compare base=base_tmp (label=" &base_label.")
             compare=comp_tmp (label=" &compare_label.")
             out=diff01 outnoequal outbase outcomp outdif &output. &cri.;
  id &idvar.;
  var &varint.;
run;

data diff01;
  set diff01;
  _obs_=lag(_obs_);
  if strip(_type_) = 'DIF';
run;

proc contents data=diff01 out=diff01_con (keep=name label type) noprint;
run;

data _null_;
  set diff01_con;
  if strip(lowcase(name)) not in ('_type_' '_obs_' 'n') then call execute('

    proc sql noprint;
      select distinct '|| strip(name) ||' into
        :'|| strip(name) ||'_value separated by ","
      from diff01;
    quit;

    ');
run;

data diff02;
  set diff01;
run;

data _null_;
  set diff01_con;
  if symexist(strip(name) || '_value') then do;
    if type = 1 then do;
      if symget(strip(name) || '_value') = 'E' then call execute('

        data diff02;
          set diff02;
          drop '|| strip(name) ||';
        run;

      ');
    end;
    if type = 2 then do;
      if compress(symget(strip(name) || '_value'), '.') = ' ' then call execute('

        data diff02;
          set diff02;
          drop '|| strip(name) ||';
        run;

      ');
    end;
  end;
run;

proc contents data=diff02
              out=diff02_con (keep=name type label
              where=(strip(name) not in ('_OBS_' 'N' '_TYPE_' "&idvar."))) noprint;
run;

data _null_;
  set diff02_con;

  if type = 1 then call execute('

    proc sql noprint;
      create table diff_' || strip(name) || ' as
      select a.*, upcase("&base.") length=20 as source, "BASE   " as source_
      from  base_tmp as a
      where a.n in (select _obs_ from diff02 where '|| strip(name) ||' ^= .E)
      union all
      select b.*, upcase("&compare.") length=20 as source, "COMPARE" as source_
      from comp_tmp as b
      where b.n in (select _obs_ from diff02 where '|| strip(name) ||' ^= .E) ;
    quit;

  ');

    if type = 2 then call execute('

    proc sql noprint;
      create table diff_' || strip(name) || ' as
      select a.*, upcase("&base.") length=20 as source, "BASE   " as source_
      from  base_tmp as a
      where a.n in (select _obs_ from diff02
                    where compress('|| strip(name) ||', " .") ^= " ")
      union all
      select b.*, upcase("&compare.") length=20 as source, "COMPARE" as source_
      from comp_tmp as b
      where b.n in (select _obs_ from diff02
                    where compress('|| strip(name) ||', " .") ^= " ") ;
    quit;

  ');

  call execute('

    data diff_'|| strip(name) ||';
      retain source_ source n ori_n '|| strip(name) ||' &othervar;
      set diff_'|| strip(name) ||';
      label source="Data Source" n="Observation Number";
      keep source_ source n ori_n '|| strip(name) ||' &othervar;
    run;

    ');

  call execute('

    proc sort data=diff_' || strip(name) ||';
      by ori_n source_ source;
    run;

    data diff_' || strip(name) ||';
      set diff_' || strip(name) ||';
      n=ori_n;
      drop ori_n source_;
    run;

  ');

  call execute('

    title
    "Variable of Interest: '|| upcase(strip(name)) ||' (Label: '|| strip(label) ||')";
    proc print data=diff_'|| strip(name) ||' label noobs blankline=(count=2);
      where &where.;
    run;
    title;

  ');

run;

title ' ';

%if %sysfunc(upcase(&clean)) = Y %then %do;  ***Remove intermediate datasets***;

  proc datasets lib=work nolist;
    delete base_tmp comp_tmp comp01 diff: base_var comp_var diff_var common_var;
  run;
  quit;

%end;

%mend compare;
```