Paper 156-2025

Highlighting the Differences: PROC COMPARE in Excel

Jane Eslinger, Eslinger Enterprises, LLC

ABSTRACT

PROC COMPARE is great for examining the differences across two data sets—but the default output doesn't always paint the full picture. Though the printed report is informational, its wordy presentation doesn't facilitate the identification of patterns in the differences within and across variables.

This paper shows how to turn raw comparison data from the OUT= options in PROC COMPARE into a polished spreadsheet that strategically uses color to highlight the details and make variable-level differences easy for the reviewer to spot.

INTRODUCTION

Printed reports generated by PROC COMPARE provide a multitude of information and detail about the differences between two data sets. However, these reports can be difficult to parse when there are large number of variables or differences. A programmer might find it beneficial to create a data set with the value differences to use for investigation. Though it is very important to have matching attributes, the focus of this paper is on matching values.

Programmers are accustomed to opening a data set and viewing the values in their preferred programming environment. However, it can be difficult to pinpoint the correct variable and observation when scrolling is involved. This paper demonstrates how using and modifying the OUT= data set generated by PROC COMPARE can alleviate this issue. The goal is to change the background color of the cells where the values do not match. By highlighting the differences, it is easier to see both which variables have a problem and if there is a pattern to the value differences.

The paper focuses on sending the results to the ODS destination for Excel for two reasons: Excel is an extremely popular program, and Excel is not a paging destination; with the ability to scroll it, the output is readable no matter how many columns or rows are in the Excel sheet.

DATA SETS FOR COMPARISON

This paper uses the data from the SASHELP.SPRINGS data set. The data set is available in SAS OnDemand for Academics but might not be available in other SAS installations. The code was tested with real-world data sets as well. The concepts and code covered in the paper are applicable to any data sets that might need to be compared through PROC COMPARE. Therefore, it is not necessary to have the SASHELP.SPRINGS data set to try out the code.

A work version of the data was created with changes made so that PROC COMPARE captures multiple types of differences.

```
data prod_springs;
    set sashelp.springs;
run;

data val_springs1;
    set sashelp.springs;
    if celsius > 120 then celsius = 120;
    name = tranwrd(name,'Icy', 'Iyc');
    format farenheit 8.1;
    label area = 'Geographical Area';
run;
```

Prod_springs is used as the base data set. Val_springs1 is used as the comparison data set.

EXCEL RESULTS

SASHELP.SPRINGS contains 1587 observations and seven columns. For this paper, only two variables and three observations were manipulated so that PROC COMPARE finds differences. This allows output to be short enough in both directions to model how the final report looks in Excel without being too cumbersome. Figure 1 shows the Excel report.

	A	B	C	D	E	F	G	H	I
1	Dataset	OBS	Latitud	Longitud	Name	Area	Type	Farenheit	Celsius
2	Production	14	53.892	-166.93	Makushin Volcano Fumaroles	Unalaska	Hotspring	310	154
3	Validation		53.892	-166.93	Makushin Volcano Fumaroles	Unalaska	Hotspring	310	120
4									
5	Production	88	58.37	-137.09	Hot Springs Near Icy Point	Mt. Fairweathe	Hotspring	153	67
6	Validation		58.37	-137.09	Hot Springs Near Iyc Point	Mt. Fairweathe	Hotspring	153	67
7									
8	Production	1400	48.77	-121.813	Sherman Crater Fumaroles	Concrete	Hotspring	266	130
9	Validation		48.77	-121.813	Sherman Crater Fumaroles	Concrete	Hotspring	266	120
10									
11									

Figure 1. Annotated Excel Report

1. The order of the columns matches the variable order in the OUT= data set.
2. The values of ‘Production’ and ‘Validation’ are supplied in a macro parameter. They correspond to the BASE= data set and the COMP= data set.
3. Blank rows are inserted between each observation (comparison pair).
4. The value from each data set is highlighted when there is a difference. Foreground and background colors match those in Excel’s default Neutral cell style.

PROC COMPARE CODE

The PROC COMPARE code in this paper is very simple and does not use BY or ID statements. Therefore, the comparison is done observation by observation. If the two data sets do not have the same number of observations, the OUT= data set will contain more results than it might if the BY or ID statement was used. However, the program provides an option for sorting the two input data sets prior to the PROC COMPARE step so that the observation-by-observation comparison is more likely to match up.

The PROC COMPARE statement has many options for generating a data set with the values of the two data sets. This paper uses the options that will create observations for both input data sets and an observation for the differences in the matching observations. The data set will be restricted to only those observations with unequal values. See the documentation for an explanation of each option.

The printed results are suppressed because the focus is the Excel output.

```
proc compare base=Base comp=Compare out=__rslt0 outbase outcomp outnoequal
    outdif criterion=&criterion noprint;
run;
```

The full code can be found in the Appendix. It has yet to be tested with a BY or ID statement, but it can be updated to use them if desired.

The __rslt0 data set is shown in Figure 2.

TYPE	_OBS_	Latitude	Longitude ▲	Name	Area	Type	Farenheit	Celsius
BASE	14	53.892	-166.93	Makushin Volcano Fumaroles	Unalaska	Hotspring	310	154
COMPARE	14	53.892	-166.93	Makushin Volcano Fumaroles	Unalaska	Hotspring	310	120
DIF	14	E	E	.....	.....	.....	E	-34
BASE	88	58.37	-137.09	Hot Springs Near Icy Point	Mt. Fairweathe	Hotspring	153	67
COMPARE	88	58.37	-137.09	Hot Springs Near Iyc Point	Mt. Fairweathe	Hotspring	153	67
DIF	88	E	E	.....XX.....	.....	.....	E	E
BASE	1400	48.77	-121.813	Sherman Crater Fumaroles	Concrete	Hotspring	266	130
COMPARE	1400	48.77	-121.813	Sherman Crater Fumaroles	Concrete	Hotspring	266	120
DIF	1400	E	E	.....	.....	.....	E	-10

Figure 2. PROC COMPARE Data

DIF OBSERVATIONS

The OUTDIF option generates an observation that is identified as 'DIF' in the _type_ variable. This observation is the key to highlighting the differences between the two data sets. However, it does not need to be shown in the final Excel report. DIF observations need to be separated from the BASE and COMPARE observations. Those observations will be used to create indicators, while the other observations will remain unchanged before being written to Excel by PROC REPORT.

```
data __rslt1 __rslt1_dif(drop=dataset);
    set __rslt0;
    length Dataset $10;

    if _type_ = 'BASE' then dataset = "&baselbl"; ①
    else if _type_ = 'COMPARE' then dataset = "&complbl"; ①

    if _type_ = 'DIF' then output __rslt1_dif;
    else output __rslt1;

    drop _type_;
run;
```

① The text is provided through a macro parameter, seen in column A of the Excel report.

VARIABLE LISTS

PROC COMPARE maintains the variable type. If the variable is character, the DIF observation value will contain dots and Xs. The dots indicate that the character at that location matches. An X indicates that the character at that location differs between the two data sets. If the variable is numeric, the DIF observation value will be .E when the values match and the numeric difference between the values when they don't match. This means that the character and numeric variables must be processed separately. The data-driven approach used in this paper is to create separate macro variables with the names of the variables of each type.

The BASE and COMPARE observation values must remain unchanged. Therefore, instead of adding inline formatting or some other distinction to the original values, a separate set of variables need to be created to indicate that there is a difference. For simplicity, the new variable names will start with an underscore (_).

Finally, the columns in Excel should be in the same order that PROC COMPARE processed them. A macro variable with the variable names in the original order is needed. The macro variable can then be used in the PROC REPORT COLUMN statement.

```
proc contents data=__rslt0(drop=_obs_ _type_)
          out=__rslt1_vars(keep=name type varnum) noprint;

proc sort data=__rslt1_vars; by varnum;
```

```
run;

proc sql noprint;
    select name into: varorder separated by ' ' from __rslt1_vars; ①
    select name into: charnames separated by ' ' from __rslt1_vars
          where type = 2; ②
    select '_'||name into: _charnames separated by ' ' from __rslt1_vars
          where type = 2; ③
    select name into: numnames separated by ' ' from __rslt1_vars
          where type = 1; ④
    select '_'||name into: _numnames separated by ' ' from __rslt1_vars
          where type = 1; ⑤
quit;
```

① &varorder holds the original variable names in the order they appeared in the base input data set.

② &charnames holds the names of the character variables.

③ &_charnames holds the names of the character variables prefixed with an underscore.

④ &numnames holds the names of the numeric variables.

⑤ &_numnames holds the names of the numeric variables prefixed with an underscore.

DIFFERENCE INDICATOR VARIABLES

The next step is to create and populate the indicator variables; again, those are the variables with the names that start with an underscore. They will be used within PROC REPORT to change the foreground and background colors. For programming simplicity, indicator variables are the same data type as the original variables. ARRAY statements and iterative DO loops are easier to program when the indicator variables for character variables also have the character type.

No matter the variable type, PROC COMPARE assigns a missing value to the DIF observation when the values between the BASE and COMPARE observations are equal. Unequal observations will have some sort of value based on the variable type. The indicator variables need to be standardized. For the purposes of the report, it matters more that the values are different than what the difference is.

In this code, the character indicator variables will have a value of X when there is a difference. The numeric ones will have a value of 9999. Please note that these values are arbitrary, as the purpose is to have standardized values. The results are shown in Figure 3.

```
data __rslt2_dif;
    set __rslt1_dif;

    array _char1{*} $ &charnames;
    array _nums1{*} &numnames;
    array _char2{*} $ &_charnames; *indicator;
    array _nums2{*} &_numnames; *indicator;

    do i=1 to dim(_char1);
          if ^index(_char1{i},'X') ① then _char2{i} = ' ';
          else _char2{i} = 'X'; ②
    end;
    do j=1 to dim(_nums1);
          if _nums1{j} = .E ③ then _nums2{j} = .;
          else _nums2{j} = 9999; ④
    end;

    keep _:;
```

```
run;
```

① PROC COMPARE indicates a character difference with an X. If an X is not found within the DIF observation value, it means the value matches between the two input data sets.

② The indicator variable is set to X when a difference is found.

③ PROC COMPARE indicates equality with .E in numeric values.

④ The indicator variable is set to 9999 when a difference is found.

OBS	_Name	_Area	_Ty...	_Latitude	_Longitude	_Farenheit	_Celsius
14				.	.	.	9999
88	X			.	.	.	.
1400				.	.	.	9999

Figure 3. __rslt_dif2 Data

From this DATA Step, only the indicator variables are kept along with the _obs_ variables. They are merged onto the data set with the BASE and COMPARISON observations.

```
data tab_dif;
    merge __rslt1 __rslt2_dif;
    by _obs_;
run;
```

REPORT CREATION

The COLUMN statement in PROC REPORT dictates the variables used from the input data set and the order of the columns in the final report. For this report, the dataset variable is first. Remember, it contains the labels provided through macro parameters. The next variable is _obs_. This variable contains the original observation number. Some programmers might not find this useful, but to maintain the same structure as the original output data set, it is kept for this report. It is also used to insert blank rows between comparison pairs. If _obs_ is omitted, another variable will be needed as an order variable.

The next variables in the COLUMN statement come from the variable listed created previously. &varorder contains the original data set variables in the desired order. Next come the indicator variables, followed by a dummy variable, neither of which will be visible in the final report. The indicator variables are used to change the foreground and background color. The dummy variable is used as the last variable in the COLUMN statement so that all the work can be done in one compute block.

```
proc report data=tab_dif ;
    column dataset _obs_ &varorder &_charnames &_numnames dummy;
    define _obs_ / order;
```

DEFINE statements are needed for each one of the indicator variables to apply the NOPRINT option, which suppresses them from the final report. Unfortunately, PROC REPORT does not allow multiple variables in a single DEFINE statement unless it is a numeric suffix list. A macro loop is needed to generate a DEFINE statement for each indicator variable. This loop focuses on the character variables.

The names of the indicator character variables are stored in &_charnames, separated by a space. &_curr1 will hold one name at a time. Its initial value is the first name in &_charnames. The %DO %WHILE loops through the entire list of names, and inside the loop is a DEFINE statement in which the report-item resolves to the name of the current indicator variable.

```
%let ind1=1;
%let _curr1 = %scan(&_charnames,&ind1,%str( ));
%do %while (%length(&_curr1)>0);
    define &_curr1 / noprint;
    %let ind1 = %eval(&ind1 + 1);
```

```
    %let _curr1 = %scan(&_charnames,&ind1,%str( ));
%end;
```

Numeric variables default to sum analysis variables, but in this situation, they are not needed as analysis variables. To make style assignments easier, the numeric variables need to be set as DISPLAY. A second macro loop is used to accomplish this, changing both the visible numeric columns and the ones that are suppressed.

The names of the indicator numeric variables are stored in &_numnames, separated by a space. &_curr2 will hold the name of the indicator variable. The names of the visible numeric variables are stored in &numnames. &_xcurr2 will hold the name of the visible numeric variable. The %DO %WHILE loops through the entire list of names. And inside the loop are two DEFINE statements. In the first DEFINE statement, the report-item resolves to the name of the current indicator variable. In the second DEFINE statement, the report-item resolves to the name of the current visible variable.

```
%let ind2=1;
%let _curr2 = %scan(&_numnames,&ind2,%str( ));
%let _xcurr2 = %scan(&numnames,&ind2,%str( ));
%do %while (%length(&_curr2)>0);
    define &_curr2 / display noprint;
    define &_xcurr2 / display;
    %let ind2 = %eval(&ind2 + 1);
    %let _curr2 = %scan(&_numnames,&ind2,%str( ));
    %let _xcurr2 = %scan(&numnames,&ind2,%str( ));
%end;
define dummy / noprint
```

Recall that one of the features of the report is a blank row between each comparison pair. This is produced by a LINE statement within a computer after block for _obs_.

```
compute after _obs_;
    line ' ';
endcomp;
```

Finally, a compute block is needed for the style changes. The dummy variable is used on the COMPUTE statement because it is the last report-item in the COLUMN statement. The compute block often acts like a DATA Step within PROC REPORT, which means ARRAY statements and iterative DO loops are allowed for working through the indicator variables.

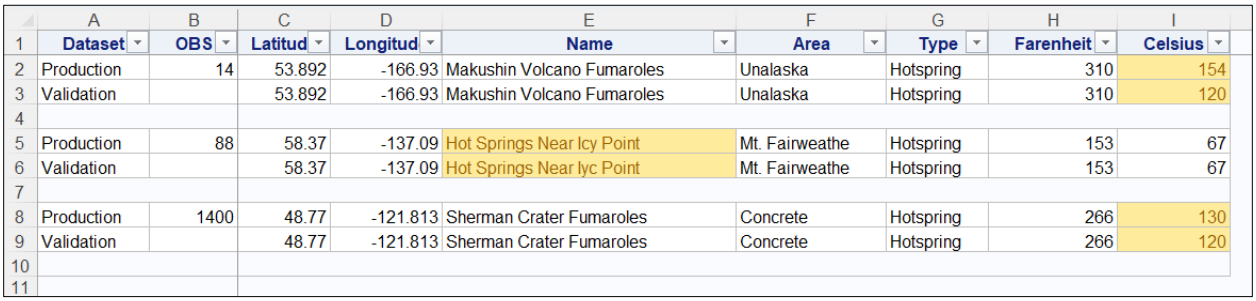
Because both character and numeric variables are involved with a different number of total variables, one iterative DO loop is needed for characters, and one loop is needed for numerics. Both will contain an IF condition looking for a non-missing value in the indicator variable. Remember: Values of X or 9999 indicate a difference. When either value is found, the foreground and background of the original/visible variable is changed.

```
compute dummy;
    array _char1{*} $ &charnames;
    array _nums1{*} &numnames;
    array _char2{*} $ &_charnames;
    array _nums2{*} &_numnames;

    do i=1 to dim(_char1);
          if ^missing(_char2{i}) then call define(vname(_char1{i}),
                'style','style={foreground=#9C6500 background=#FFEB9C}');
    end;
    do j=1 to dim(_nums1);
          if ^missing(_nums2{j}) then call define(vname(_nums1{j}),
```

```
                    'style','style={foreground=#9C6500 background=#FFEB9C}');
    end;
endcomp;
```

The final report, shown in Figure 4, looks very similar to the original OUT= data set, except it is much easier to scroll through and has color to help the reviewer easily see the differences.



	A	B	C	D	E	F	G	H	I
1	Dataset	OBS	Latitud	Longitud	Name	Area	Type	Farenheit	Celsius
2	Production	14	53.892	-166.93	Makushin Volcano Fumaroles	Unalaska	Hotspring	310	154
3	Validation		53.892	-166.93	Makushin Volcano Fumaroles	Unalaska	Hotspring	310	120
4									
5	Production	88	58.37	-137.09	Hot Springs Near Icy Point	Mt. Fairweathe	Hotspring	153	67
6	Validation		58.37	-137.09	Hot Springs Near Iyc Point	Mt. Fairweathe	Hotspring	153	67
7									
8	Production	1400	48.77	-121.813	Sherman Crater Fumaroles	Concrete	Hotspring	266	130
9	Validation		48.77	-121.813	Sherman Crater Fumaroles	Concrete	Hotspring	266	120
10									
11									

Figure 4. Final Excel Report

CONCLUSION

This paper focused on the PROC COMPARE OUT= data set. Sending the resulting data set to Excel helps make scrolling easier. But the real benefit is highlighting the differences found by PROC COMPARE. With added coloring, a programmer's job is much easier in locating the differences and discerning any patterns among them. The report can also be shared with the production programmer to help reconcile the differences faster.

REFERENCES

SAS Institute Inc. 2025. COMPARE Procedure. SAS® Viya® Platform Programming Documentation. Cary, NC: SAS Institute Inc. Available at https://documentation.sas.com/doc/en/pgmsascdc/v_064/proc/n1nwxbchh5hpu1n1h28kmici2awd.htm.

SAS Institute Inc. 2025. SAS® Functions and CALL Routines: Reference. SAS® Viya® Platform Programming Documentation. Cary, NC: SAS Institute Inc. Available at https://documentation.sas.com/doc/en/pgmsascdc/v_064/lefunctionsref/titlepage.htm.

ACKNOWLEDGMENTS

The author is grateful to Josh Horstman for the challenge of making PROC COMPARE output pretty in Excel. I would also like to thank Inka Leprince and Richann Watson for reviewing the paper and Lauree Shepard for editing it.

RECOMMENDED READING

- Eslinger, Jane. 2015. "The REPORT Procedure: A Primer for the Compute Block." In Proceedings of the SAS Global 2015 Conference. Cary, NC: SAS Institute Inc. Available at support.sas.com/resources/papers/proceedings15/SAS1642-2015.pdf.
- Eslinger, Jane. 2024. "Incorporating Macro with PROC REPORT Code." In Proceedings of the WUSS Conference. CA. Available at https://www.wuss.org/proceedings/2024/WUSS-2024-Paper-103.pdf.

CONTACT INFORMATION

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

Jane Eslinger
Eslinger Enterprises, LLC

eslingerent@gmail.com

APPENDIX

```
/************************************************************************
* Program: procompare_awesome
* Author: Jane Eslinger
* Date: 23Jun2025
*
* Purpose: To output Proc Compare results to Excel in a readable format
*
* Inputs: Base data set
*         Compare data set
*
* Output: &filepath.\&filename..xlsx with two tabs
*
* Revisions:
************************************************************************/

/*
REQUIRED Macro Parameters:
     basedr - libref for the base/production/main data set
     compdr - libref for the compare/validation/qc data set
     baseds - name of the base/production/main data set
     compds - name of the compare/validation/qc data set
OPTIONAL Macro Parameters:
     keys - variables to use in the ID statement in Proc Compare
     sortby - variables to sort the data sets by before Proc Compare
     filepath - the folder path for saving Excel
     filename - the name of the Excel file to create
     criterion - criterion for numeric variable comparison
     baselbl - label for the base/production/main data set in Differences tab
     complbl - label for the compare/validation/qc data set in Differences tab
*/

%macro pcompout(
     basedr = ,
     compdr = ,
     baseds = ,
     compds = ,
     keys = ,
     sortby = ,
     filepath = ,
     filename = ProcCompareOutput,
     criterion = 0.000000001,
     baselbl = Production,
     complbl = Validation,
);

************************************************************************
*             CHECK FOR REQUIRED MACRO PARAMETERS                      *
************************************************************************;

     %if %length(&basedr) = 0 %then %do;
          %put %sysfunc(compress(ERR OR:)) Parameter BASEDR Base libref not provided;
          %abort;
     %end;
     %if %length(&compdr) = 0 %then %do;
          %put %sysfunc(compress(ERR OR:)) Parameter COMPDR Comparison libref not provided;
          %abort;
     %end;
     %if %length(&baseds) = 0 %then %do;
          %put %sysfunc(compress(ERR OR:)) Parameter BASEDS Base data set not provided;
          %abort;
     %end;
     %if %length(&compds) = 0 %then %do;
          %put %sysfunc(compress(ERR OR:)) Parameter COMPDS Comparison data set not provided;
          %abort;
     %end;

************************************************************************
*                              SET UP                                  *
************************************************************************;
```

```
filename temp "&filepath.\&filename..xlsx";
data _null_;
       if fexist('temp') then rc = fdelete('temp');
run;
filename temp clear;

%if &sortby ne %then %do;
       proc sort data=&basedr..&baseds out=base;
              by &sortby;
       run;

       proc sort data=&compdr..&compds out=compare;
              by &sortby;
       run;
%end;
%else %do;
       data base;
              set &basedr..&baseds;
       run;

       data compare;
              set &compdr..&compds;
       run;
%end;


*************************************************************************
*                          DIFFERENCES TAB                              *
*************************************************************************;

     /***************** HORIZONGAL SUMMARY *****************/
     *Only observations with differences will be included;
     proc compare base=Base comp=Compare out=__rslt0 outbase outcomp outnoequal outdif
            criterion=&criterion noprint;
     run;

     *separate dif rows;
     data __rslt1 __rslt1_dif(drop=dataset);
            set __rslt0;
            length Dataset $10;

            if _type_ = 'BASE' then dataset = "&baselbl";
            else if _type_ = 'COMPARE' then dataset = "&complbl";

            if _type_ = 'DIF' then output __rslt1_dif;
            else output __rslt1;

            drop _type_;
     run;

     *create macro lists of variable names;
     proc contents data=__rslt0(drop=_obs_ _type_)
            out=__rslt1_vars(keep=name type varnum) noprint;
     proc sort data=__rslt1_vars; by varnum;
     run;

     proc sql noprint;
            select name into: varorder separated by ' ' from __rslt1_vars;
            select name into: charnames separated by ' ' from __rslt1_vars
                   where type = 2;
            select '_'||name into: _charnames separated by ' ' from __rslt1_vars
                   where type = 2;
            select name into: numnames separated by ' ' from __rslt1_vars
                   where type = 1;
            select '_'||name into: _numnames separated by ' ' from __rslt1_vars
                   where type = 1;
     quit;

     *change variable names in dif data set
      these variables will be used to change cell colors in proc report;
```

```
    data __rslt2_dif;
           set __rslt1_dif;

           array _char1{*} $ &charnames;
           array _nums1{*} &numnames;
           array _char2{*} $ &_charnames;
           array _nums2{*} &_numnames;

           do i=1 to dim(_char1);
                  if ^index(_char1{i},'X') then _char2{i} = ' ';
                  else _char2{i} = 'X';
           end;
           do j=1 to dim(_nums1);
                  if _nums1{j} = .E then _nums2{j} = .;
                  else _nums2{j} = 9999;
           end;

           keep _:;
    run;

    *merge on variables that indicate a difference;
    data tab_dif;
           merge __rslt1 __rslt2_dif;
           by _obs_;
    run;

*************************************************************************
*                    SEND REPORT TO EXCEL                               *
*************************************************************************;
    ods excel file= "&filepath.\&filename..xlsx";

    options nolabel;
    ods excel options(sheet_name='Differences' autofilter='ALL'
           frozen_headers='1' frozen_rowheaders='2');
    proc report data=tab_dif ;
           column dataset _obs_ &varorder &_charnames &_numnames dummy;
           define _obs_ / order;

           *noprint the difference indicator variables;
           %let ind1=1;
           %let _curr1 = %scan(&_charnames,&ind1,%str( ));
           %do %while (%length(&_curr1)>0);
                  define &_curr1 / noprint;
                  %let ind1 = %eval(&ind1 + 1);
                  %let _curr1 = %scan(&_charnames,&ind1,%str( ));
           %end;

           *set numeric variables to display
            noprint the difference indicator variables;
           %let ind2=1;
           %let _curr2 = %scan(&_numnames,&ind2,%str( ));
           %let _xcurr2 = %scan(&numnames,&ind2,%str( ));
           %do %while (%length(&_curr2)>0);
                  define &_curr2 / display noprint;
                  define &_xcurr2 / display;
                  %let ind2 = %eval(&ind2 + 1);
                  %let _curr2 = %scan(&_numnames,&ind2,%str( ));
                  %let _xcurr2 = %scan(&numnames,&ind2,%str( ));
           %end;
           define dummy / noprint;

           compute after _obs_;
                  line ' ';
           endcomp;

           *change the background color for the cells where the Base and Compare
            do not match;
           compute dummy;
                  array _char1{*} $ &charnames;
                  array _nums1{*} &numnames;
                  array _char2{*} $ &_charnames;
```

```
                array _nums2{*} &_numnames;

                do i=1 to dim(_char1);
                        if ^missing(_char2{i}) then call define(vname(_char1{i}),
                        'style','style={foreground=#9C6500 background=#FFEB9C}');
                end;
                do j=1 to dim(_nums1);
                        if ^missing(_nums2{j}) then call define(vname(_nums1{j}),
                        'style','style={foreground=#9C6500 background=#FFEB9C}');
                end;
        endcomp;
    run;
    options label;

    ods excel close;

************************************************************************
*                          CLEAN UP                                    *
************************************************************************;
    /*proc datasets lib=work noprint;
        delete __:;
    quit;*/

%mend pcompout;
```