Paper 1654-2018

Identifying Duplicate Variables in a SAS ® Data Set

Bruce Gilsen, Federal Reserve Board, Washington, DC

ABSTRACT

In the big data era, removing duplicate data from a data set can reduce disk storage use and improve processing time. Many papers have discussed removal of duplicate observations, but it is also useful to identify duplicate variables for possible removal. One way to identify duplicate variables is with PROC COMPARE, which is commonly used to compare two data sets, but can also compare variables in the same data set. It can accept a list of variable pairs to compare and determine which variable pairs are identical. This paper shows how to obtain a summary report of identical variables for all numeric or character variables in a data set, using the following steps:

1. Dynamically build a list of all possible numeric or character variable pairs for PROC COMPARE to analyze.
2. Convert PROC COMPARE pairwise results (for example, N1 is identical to N3, N3 is identical to N5, N1 is identical to N5, and so on) into a summary report that groups all identical variables (for example, N1, N3, and N5 are identical).

For very large data sets, the paper shows how to substantially improve performance by first executing PROC COMPARE one or more times on a small number of observations to reduce the number of extraneous comparisons.

DATA SET USED IN THIS PAPER

Data set ONE is defined as follows.

- One million observations and five numeric variables, N1-N5.
- N1, N3, and N5 are identical, and N2 and N4 are identical.
- The five variables are identical in all but the last observation.

```
    obs       N1       N2       N3       N4      N5
      1        1        1        1        1       1
      2        2        2        2        2       2
      3        3        3        3        3       3
                            .....
 999999   999999   999999   999999   999999  999999
1000000        1        2        1        2       1
```

EXAMPLE 1: A BASIC DUPLICATE VARIABLE REPORT FOR SELECTED VARIABLES

In this example, we check three variable pairs in data set ONE (N1 and N3, N2 and N5, and N3 and N5) to see if they are identical.

```
ods output comparesummary=_comparesummary;
```

```
proc compare base=one listequalvar novalues;
  var  n1 n2 n3;
  with n3 n5 n5;
run;
```

The following output is written to data set _COMPARESUMMARY.

- Records where TYPE=H are header records or blank records.
- Records where TYPE=D are detail records or blank records.

```
Obs    type   batch

  1     d
  2     d
  3     h                          Values Comparison Summary
  4     h
  5     d    Number of Variables Compared with All Observations Equal: 2.
  6     d    Number of Variables Compared with Some Observations Unequal: 1.
  7     d    Total Number of Values which Compare Unequal: 1.
  8     d    Maximum Difference: 1.
  9     d
 10     d
 11     h                       Variables with All Equal Values
 12     h
 13     h                  Variable  Type  Len   Compare   Len
 14     d
 15     d                  n1        NUM     8   n3          8
 16     d                  n3        NUM     8   n5          8
 17     d
 18     h                        Variables with Unequal Values
 19     h
 20     h          Variable  Type  Len   Compare   Len  Ndif   MaxDif
 21     d
 22     d          n2        NUM     8   n5          8     1    1.000
 23     d
```

EXAMPLE 2: THE DUPLICATE VARIABLE REPORT WE WANT

We want a report that groups all identical numeric variables in the data set, as follows.

```
The following variables are identical:
n1 n3 n5
n2 n4
```

Our code must do the following.

1. Generate a complete list of variable pairs to supply in the VAR and WITH statements. For variables N1-N5 in data set ONE, the following variables are compared.
   - N1 with N2, N3, N4, N5 (4 comparisons)
   - N2 with N3, N4, N5 (3 comparisons)
   - N3 with N4, N5 (2 comparisons)
   - N4 with N5 (1 comparison)

The following VAR and WITH statements are needed for N1-N5.

```
var  n1 n1 n1 n1 n2 n2 n2 n3 n3 n4;
with n2 n3 n4 n5 n3 n4 n5 n4 n5 n5;
```

In general, for N variables, the number of comparisons is the sum from 1 to N-1, for example

- For 5 variables, 1+2+3+4 = 10 comparisons
- For 10 variables, 1+2+3+4+5+6+7+8+9 = 45 comparisons
- For 20 variables, 1+2+ ... +19 = 190 comparisons
- For 100 variables, 1+2+ ... +99 = 4950 comparisons

2. Convert PROC COMPARE pairwise results into a summary report that groups all identical variables.

As shown in the previous section, PROC COMPARE generates a data set with pairwise results (for example, N1 is identical to N3, N3 is identical to N5, N1 is identical to N5, and so on) that includes some extra detail. We will read the data set in a DATA step and generate a summary report that groups all identical variables together.

THE CODE FOR THE DUPLICATE VARIABLE REPORT

CREATE MACRO VARIABLES CONTAINING THE VARIABLE NAMES

PROC SQL creates the following macro variables.

- VAR1, VAR2, ..., VAR5 contain the names of the numeric variables in data set ONE.

- NUM_VALUES contains the number of numeric variable names copied to macro variables. It is copied from the automatic macro variable SQLOBS, which contains the number of rows selected by the last PROC SQL statement.

```
proc sql noprint;
  select name
  into :var1-:var&sysmaxlong
  from dictionary.columns
  where libname="WORK"
  and memname="ONE"
  and type = "num"
  ;
  %let num_values = &sqlobs; /* Number of macro variables created */
quit;
```

In this example, the macro variables have the following values:

```
VAR1:       N1
VAR2:       N2
VAR3:       N3
VAR4:       N4
VAR5:       N5
NUM_VALUES: 5
```

Notes on this code

- We are reading the COLUMNS DICTIONARY table, which contains information (name, type, and so on) about all variables in all currently referenced tables in the current SAS session.
- The WHERE clause limits the select to numeric variables in data set WORK.ONE.
- SYSMAXLONG is an operating system-specific automatic macro variable containing the largest possible integer value. PROC SQL will create as many macro variables as necessary, up to the value of SYSMAXLONG. Specifying the macro variable range this way allows the number of macro variables created by PROC SQL to be data driven.
- SQLOBS is overwritten by the next PROC SQL statement, so I always copy it to another macro variable for safekeeping.

CREATE MACRO VARIABLES CONTAINING VARIABLE NAME PAIRS TO COMPARE

Macro NAMELISTS creates macro variables VAR and WITH that contain all possible variable name pairs for PROC COMPARE to analyze. As noted above, for five variables there are nine variable name pairs.

```
%global var with;
%macro namelists;
  %local i j;
```

```
    %do i = 1 %to &num_values;
      %do j = &i+1 %to &num_values;
        %let VAR=&VAR &&var&i;
        %let WITH=&WITH &&var&j;
      %end;
    %end;
  %mend namelists;
  %namelists
```

In this example, macro variables VAR and WITH have the following values:

```
   VAR: n1 n1 n1 n1 n2 n2 n2 n3 n3 n4
  WITH: n2 n3 n4 n5 n3 n4 n5 n4 n5 n5
```

RUN PROC COMPARE, WRITE RESULTS TO DATA SET

PROC COMPARE compares all variable pairs, and the ODS OUTPUT statement writes the COMPARESUMMARY output object to the data set _COMPARESUMMARY. There is no way to obtain just the variable comparison summary we need, but this output object along with the options specified in the PROC COMPARE statement minimizes extraneous output.

```
  ods output comparesummary=_comparesummary;
  proc compare base=one listequalvar novalues;
    var  &var;
    with &with;
  run;
```

The following output is written to data set _COMPARESUMMARY,

```
Obs    type    batch


  1     d
  2     d
  3     h                          Values Comparison Summary
  4     h
  5     d    Number of Variables Compared with All Observations Equal: 4.
  6     d    Number of Variables Compared with Some Observations Unequal: 6.
  7     d    Total Number of Values which Compare Unequal: 6.
  8     d    Maximum Difference: 1.
  9     d
 10     d
 11     h                       Variables with All Equal Values
```

```
12     h
13     h                    Variable  Type  Len   Compare   Len
14     d
15     d                    n1        NUM     8   n3          8
16     d                    n1        NUM     8   n5          8
17     d                    n2        NUM     8   n4          8
18     d                    n3        NUM     8   n5          8
19     d
20     h                        Variables with Unequal Values
21     h
22     h            Variable  Type  Len   Compare   Len  Ndif   MaxDif
23     d
24     d            n1        NUM     8   n2          8     1    1.000
25     d            n1        NUM     8   n4          8     1    1.000
26     d            n2        NUM     8   n3          8     1    1.000
27     d            n2        NUM     8   n5          8     1    1.000
28     d            n3        NUM     8   n4          8     1    1.000
29     d            n4        NUM     8   n5          8     1    1.000
30     d
```

READ DATA SET, BUILD LIST OF VARIABLES WITH IDENTICAL VALUES

The following DATA step reads the data set created by PROC COMPARE and generates a report that groups identical variables. In this example, the following report is generated.

```
The following variables are identical:
n1 n3 n4
n2 n5
```

Following the code are descriptions of sections of the code that correspond to numbered items on the right.

```
data _null_;
  array identical_vars (*) $500 identical_1 - identical_20 ;         /*1*/
  length ident1 ident2 $32;
  retain num_equalvars . num_equalvars_count 0
         nrepeat 0 past_summary 0 identical_vars;                    /*2*/

      /* The DO WHILE loop reads records until all records
         containing identical pairs, if any, have been read. */
  do while (num_equalvars ne num_equalvars_count);                   /*3*/
  set _comparesummary;
    /* Determine number of identical variables */
  if index(batch,
    'Number of Variables Compared with All Observations Equal:')>0
    then num_equalvars=input(scan(batch,-1,' '),best7.);
  else if index(batch,'Variables with All Equal Values') ne 0
    then past_summary=1; /* start of useful stuff */

  if past_summary = 1 and type="d" and length(batch) > 1 then do;  /*4*/
    ident1 = scan(batch,1," ");  /* extract the 2 identical variables */
    ident2 = scan(batch,4," ");
    num_equalvars_count+1;
                                                                    /*5*/
    foundmatch=0;  /* Is at least one of current variables a match? */
    do i = 1 to nrepeat while (foundmatch=0);  /* 1x/existing var list*/
      if indexw(identical_vars(i),ident1) then foundmatch+1;
      if indexw(identical_vars(i),ident2) then foundmatch+2;
      if foundmatch=1 then identical_vars(i) =
          catx(' ', identical_vars(i), ident2);                 /*5.1*/
      else if foundmatch=2 then identical_vars(i) =
          catx(' ', identical_vars(i), ident1);
    end;   /* of do i = 1 to nrepeat while (foundmatch=0); */

      /* Did not find current variables, so start a new
         array element with the current variables */
    if foundmatch = 0 then do;                                  /*5.2*/
      identical_vars(i) = catx(' ', ident1, ident2);
      nrepeat+1;
    end;
  end; /* of if past_summary = 1 and type = "d".... */
end; /* of do while (num_equalvars ne num_equalvars_count); */

  /* Finished processing any identical variables.  If there
     are any identical variables, print summary report. */         /*6*/
if num_equalvars_count = 0
  then put "No variables are identical in the data set";
else do;
  put "The following variables are identical:";
```

```
      do i = 1 to nrepeat;
        put identical_vars (i);
      end;
    end;
    stop; /* Do not read rest of data set */
  run;
```

1. Each array element will contain one list of identical variables, so the array size must be large enough to hold as many lists as necessary.

The size of the array elements must be large enough to hold one set of variables with identical values, space-separated. Since the maximum variable name length is 32, if (for example) there are at most 10 variables that are identical, the array element size should be (32*10) + 9 spaces = 329.

2. Some variables used in this DATA step are as follows.

- NUM_EQUALVARS is the number of identical variable pairs.
- NUM_EQUALVARS_COUNT counts the number of identical variable records read so far.
- NREPEAT counts the number of array elements populated with variables so far.
- PAST_SUMMARY is 0 before we reach the section of the output that has detail records for identical variables, then 1.

3. The DO WHILE loop reads records until all records containing identical pairs, if any, have been read.

The record with the text ‘Number of Variables Compared with All Observations Equal:’ contains the number of identical records, which is copied to the variable NUM_EQUALVARS. If it is 0, then the DO WHILE loop does not execute again. Otherwise, the DO WHILE loop iterates until detail records for all identical variables have been read.

The record with the text 'Variables with All Equal Values' indicates that the detail records with identical variables are to follow.

4. We are reading a record that contains two identical variables. BATCH has a value like

```
                n1         NUM      8    n3            8
```

Copy the two variables to IDENT1 and IDENT2 and increment NUM_EQUALVARS_COUNT, which counts the number of identical variable records read so far.

5. Scan the identical variables we have already found until we either find a match or have checked all existing variables. For the current element of IDENTICAL_VARS

FOUNDMATCH	Result
0	Neither variable found. Continue the search if more non-blank array elements to check (section 5.1), otherwise copy both variables to a new array element (section 5.2).

1	Just IDENT1 found, add IDENT2 to current element, end search (section 5.1).
2	Just IDENT2 found, add IDENT1 to current element, end search (section 5.1).
3	Both IDENT1 and IDENT2 found, end search.

To illustrate, here are the four records with identical variables from our example and (for clarity) the few prior header records.

```
          Variables with All Equal Values


   Variable  Type  Len   Compare   Len


   n1        NUM     8   n3          8
   n1        NUM     8   n5          8
   n2        NUM     8   n4          8
   n3        NUM     8   n5          8
```

The four records are processed as follows.

- Record 1: N1 and N3 are not found. In section 5.2
    - IDENTICAL_VARS(1) is set to the following: n1 n3.
    - NREPEAT is incremented from 0 to 1.
- Record 2: N1 is found in IDENTICAL_VARS(1), N5 is not found. In section 5.1, N5 is appended to IDENTICAL_VARS(1) which now has the following value: n1 n3 n5.
- Record 3: N2 and N4 are not found. In section 5.2
    - IDENTICAL_VARS(2) is set to the following: n2 n4.
    - NREPEAT is incremented from 1 to 2.
- Record 4: N3 and N5 are both found in IDENTICAL_VARS(1). No action is taken.

6. Processing the data set is finished. Check if there are any identical variables and if there are, print a summary report. The STOP statement prevents the DATA step from executing any further.

ENHANCEMENT: A TWO STAGE VARIABLE COMPARISON

PROC COMPARE compares values in all observations; it does not stop checking a variable pair when unequal values are found. For large data sets in which many variables differ early in the data set, many extraneous comparisons could take place. In this case, a two stage approach, which could result in a major performance improvement, could be constructed as follows.

- Run PROC COMPARE with all variable pairs for the first 10 observations.
- Convert PROC COMPARE pairwise results into a second, reduced set of variable pairs to compare.

- Assuming the first PROC COMPARE finds any identical variable pairs, run PROC COMPARE with the reduced set of variable pairs for all but the first 10 observations, and use the results to build the final summary report.

Appendix 2 contains the code for the two stage variable comparison. It is very similar to the code in Example 2 above, with the following differences.

- The data set option OBS= limits the first PROC COMPARE to the first 10 observations and the data set option FIRSTOBS= causes the second PROC COMPARE to ignore the first 10 observations.
- The values used in the VAR and WITH statements for the second PROC COMPARE are generated from the first PROC COMPARE rather than the COLUMNS DICTIONARY table.
- The second PROC COMPARE only takes place if the first PROC COMPARE returns any identical variables.

I created data set TWO with values as follows.

- It has one million observations and 100 numeric variables, N1-N100.
- N1/N3/N5 and N2/N4 are identical in all observations.
- All 100 variables are identical in all but observations 5 and 1,000,000.
  - In observation 5, N1-N6 are the same, N11-N12 are the same, and N96-N100 are the same. All other variables have unique values.
  - In observation 1,000,000, N1, N3, and N5 are the same, and N2 and N4 are the same. All other variables have unique values.

PROC COMPARE processing differs as follows for these data values.

- In the one stage solution shown in Example 2, PROC COMPARE compares 4,950 variable pairs for 1,000,00 observations.
- In the two stage solution shown in Appendix 2, the first PROC COMPARE compares 4,950 variable pairs for 10 observations and the second PROC COMPARE compares 26 variable pairs for 999,990 observations.

In Linux SAS 9.4TS1M4, the one stage solution took 65 seconds and the two stage solution took 4.5 seconds.

ADDITIONAL ENHANCEMENTS

Some possible enhancements of this code include the following. Other enhancements are possible as well.

1. To test for identical character variables, *and type* = *"num"* can be changed to *and type* = *"char"* in the PROC SQL step.
2. The code could be put in a macro that allows you to specify the data set, the variable type (character or numeric), and an optional list of variables to be checked instead of all numeric or character variables.
3. The list of identical variables could be used to generate a KEEP statement or KEEP= clause in a

subsequent step that only preserves one of the identical variables.

4. In Linux SAS 9.4TS1M4 with one million observations, PROC COMPARE ran in two seconds for 20 numeric variables, 14 seconds for 50 numeric variables, and 65 seconds for 100 numeric variables, so the code in Example 2 can be used on reasonably big data sets. If necessary, the two stage solution can be used instead. In extreme cases, the two stage solution could be generalized to run PROC COMPARE multiple times, with all but the last instance used to compare a small number of observations and eliminate variable pairs for subsequent iterations.

CONCLUSION

This paper showed how to easily identify duplicate variables in a SAS data set. Removing duplicate variables can reduce disk storage use and improve processing time.

REFERENCES

SAS Institute Inc. (2017), "Base SAS 9.4 Procedures Guide, Seventh Edition," Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2016), "SAS 9.4 Statements: Reference, Fifth Edition," Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2016), "SAS 9.4 Macro Language: Reference, Fifth Edition," Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The following people contributed extensively to the development of this paper and their support is greatly appreciated: Peter Sorock, Heidi Markovitz, and Donna Hill at the Federal Reserve Board, Mark Keintz at Wharton Research Data Services, and Rick Langston at SAS Institute. Special thanks to Mark and Heidi for prodding me to consider a two stage solution and helping me simplify the code.

CONTACT INFORMATION

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

Bruce Gilsen
Federal Reserve Board, Mail Stop N-122, Washington, DC 20551
202-452-2494
bruce.gilsen@frb.gov

APPENDIX 1: GENERATING THE DATA

The following code generates data set ONE used in this paper. To create a data set with more variables, just change the ARRAY definition to create the appropriate number of variables.

```
data one;
  drop i ii;
  array nall (*) n1-n5;
  do i=1 to 999999;  /* Write all but last observation, all vars equal */
```

```
      do ii=1 to dim(nall);
         nall(ii) = i;
      end;
      output;
    end;
    /* Write last observation: n1/n3/n5 and n2/n4 same, all others differ */
    n1=1;
    n3=1;
    n5=1;
    n2=2;
    n4=2;
    output;
  run;
```

APPENDIX 2: CODE FOR THE TWO STAGE SOLUTION

Significant changes to the code are bolded.

```
  proc sql noprint;
    select name
    into :var1-:var&sysmaxlong
    from dictionary.columns
    where libname="WORK"
    and memname="TWO"
    and type = "num"
    ;
    %let num_values = &sqlobs; /* Number of macro variables created */
  quit;

  %global var with;
  %macro namelists;
    %local i j;
    %do i = 1 %to &num_values;
      %do j = &i+1 %to &num_values;
        %let VAR=&VAR &&var&i;
        %let WITH=&WITH &&var&j;
      %end;
    %end;
  %mend namelists;
```

```
   %namelists

     /* Run 1st occurrence of PROC COMPARE for 1st 10 observations */
   ods output comparesummary=_comparesummary;
   proc compare base=two (obs=10) listequalvar novalues;
     var  &var;
     with &with;
   run;

   data _null_;
     length var_list with_list $32767; /* variable lists for PROC COMPARE 2*/
     retain num_equalvars . num_equalvars_count 0
            past_summary 0 with_list var_list;

       /* DO WHILE loop reads all identical pairs if any */
     do while (num_equalvars ne num_equalvars_count);
       set _comparesummary;
         /* Determine number of identical variables */
       if index(batch,
         'Number of Variables Compared with All Observations Equal:')>0
         then num_equalvars=input(scan(batch,-1,' '),best7.);
       else if index(batch,'Variables with All Equal Values') ne 0
         then past_summary=1; /* start of useful stuff */

           /* Extract the 2 identical variables and append them to the VAR
              and WITH lists being built for the 2nd PROC COMPARE */
       if past_summary = 1 and type="d" and length(batch) > 1 then do;
         var_list=catx(' ', var_list, scan(batch,1," "));
         with_list=catx(' ', with_list, scan(batch,4," "));
         num_equalvars_count+1;
       end;
     end; /* of do while (num_equalvars ne num_equalvars_count); */
     if num_equalvars_count = 0
       then put "No variables are identical in the first 10 observations, no
further testing done.";
     else do;
         /*Copy identical variables to macro variables for 2nd PROC COMPARE*/
       call symput("var",trim(var_list));
```

```
    call symput("with",trim(with_list));
    put "Number of identical variables in first 10 observations: "
        num_equalvars_count;
  end;
      /* Save number of identical variables.  If 0, 2nd stage not run. */
  call symputx("num_equalvars_count",num_equalvars_count);
  stop; /* Do not read rest of data set */
run;

%macro twostage;
 %if &num_equalvars_count ne 0 %then %do;

     /* Run PROC COMPARE for the reduced set of identical variables */
   ods output comparesummary=_comparesummary;
   proc compare base=two (firstobs=11) listequalvar novalues;
     var  &var;
     with &with;
   run;

   data _null_;
     array identical_vars (*) $500 identical_1 - identical_20 ;
     length currentvar ident1 ident2 $32;
     retain num_equalvars . num_equalvars_count 0
            nrepeat 0 past_summary 0 identical_vars;

         /* DO WHILE loop reads all identical pairs if any */
     do while (num_equalvars ne num_equalvars_count);
       set _comparesummary;
       if index(batch,
         'Number of Variables Compared with All Observations Equal:')>0
        then num_equalvars=input(scan(batch,-1,' '),best7.);
       else if index(batch,'Variables with All Equal Values') ne 0
         then past_summary=1; /* start of useful stuff */

       if past_summary = 1 and type = "d" and length(batch) > 1 then do;
           /* extract the 2 identical variables */
         ident1 = scan(batch,1," ");
```

```
          ident2 = scan(batch,4," ");
          num_equalvars_count+1;
          foundmatch=0; /* is at least one of current variables a match? */
          do i = 1 to nrepeat while (foundmatch=0);/*1x/existing var list*/
            if indexw(identical_vars(i),ident1) then foundmatch+1;
            if indexw(identical_vars(i),ident2) then foundmatch+2;
            if foundmatch=1 then identical_vars(i) =
                catx(' ', identical_vars(i), ident2);
            else if foundmatch=2 then identical_vars(i) =
                catx(' ', identical_vars(i), ident1);
          end;   /* of do i = 1 to nrepeat while (foundmatch=0); */

            /* Did not find current variables, so start a new
              array element with the current variables */
          if foundmatch = 0 then do;
            identical_vars(i) = catx(' ', ident1, ident2);
            nrepeat+1;
          end;
        end; /* of if past_summary = 1 and type = "d".... */
      end; /* of do while (num_equalvars ne num_equalvars_count); */

        /* Finished processing any identical variables.
           Print summary report if there are any identical variables. */
      if num_equalvars_count = 0 then put "No variables are identical in
the data set";
      else do;
        put "The following variables are identical:";
        do i = 1 to nrepeat;
          put identical_vars (i);
        end;
      end;
      stop; /* Do not read rest of data set */
    run;
  %end; /* of %if &num_equalvars_count ne 0 */
 %mend twostage;
 %twostage
```