Paper 097-29

SAS Macros for Large Scale Data Analysis and Quality Management of Corporate Actuarial Data Mart

Dennis Tang, Actuarial Associate III, WellPoint Inc., Woodland Hills, CA
Don Cooper, Director, Actuarial Data Management, WellPoint Inc., Thousand Oaks, CA

ABSTRACT

The Actuarial Data Management department at WellPoint Health Networks often works with large amounts of new health care claim, membership and other types of data. Advanced SAS macros were developed to provide descriptive statistics as well as conduct quality control and process testing. This paper discusses these macros and the advanced SAS techniques used, i.e., saving output from PROC CONTENTS; SELECT INTO in PROC SQL; coding varying SAS code using CALL EXECUTE; and recursions of macros.

INTRODUCTION

The Actuarial Data Management department at Wellpoint Health Networks supports the capture and analysis of analytical data. It is charged with ensuring the availability and quality of this data for setting reserves and the analysis of other aspects of the business. Every month membership, claims and other data from various sources and departments are extracted and processed. These files are destined for Data Warehouses, Data Marts and WEB based applications and can contain a million rows and hundreds of variables.

WHAT THE MACROS DO

The main SAS macro is *DataView*. It analyzes large files efficiently generating descriptive statistics such as: grand totals; averages; and most frequent values. The macro also produces data quality related statistics and provides structural information about the data. The data quality related statistics are the number of lines with missing numeric variables and the number of lines with blank or null character values. The structural information identifies any duplicate lines and the logical key.

For purpose of discussions in this paper data is defined as numeric or character values in a tabular form consisting of rows (lines, records, or observations) and columns (variables). Descriptive statistics contains simple yet important information about each column or variable. DataView calculates grand totals for the two-fold purposes of measuring the magnitude and providing control totals of the data.

DataView also provides insight to data quality as revealed by the number of records with missing or blank fields. There may be times when missing values are valid. However missing values often represent problems in data, especially when they occur frequently. See Figure 1 below for an illustrative output of macro DataView.

Figure 1. Illustrative Output from DataView

```
             Sample Health Insurance Data
         Data Set is Reported_members

     (Total # of Records is 1,296,196)

                            Grand
Variable    Type   Missing   Total        Note

Members      Num.     0      993,847
Subscribers Num.     0      871,765
Coverage_ID Char     -          -
Funding      Char     -          -
Product      Char     -          -
Contract     Char     -          -
Transfers    Num. 1,296,196     -      AllMissing
Month        Num.     -          -      Date variable
```

Another descriptive statistic that applies to both numeric and character variables is frequency distribution. DataView identifies the top 10 most frequent values for each variable. This output helps test for reasonableness and compliance with the business requirements. See Figure 2 below for an illustrative output.

Figure 2. Illustrative Output of DataView

```
      Top 10 values of variable Members
              (Sample of 100000)

       Obs     Members    COUNT     PERCENT

        1       1.00      38989      38.989
        2       2.00      16144      16.144
        3       3.00       9539       9.539
        4       4.00       9502       9.502
        5       0.00       5784       5.784
        6       5.00       5596       5.596
        7       6.00       3206       3.206
        8       7.00       1963       1.963
        9       8.00       1448       1.448
       10       9.00       1022       1.022

    Top 10 values of variable Coverage_ID
              (Sample of 100000)

       Obs    Coverage_ID    COUNT     PERCENT

        1         01        100000    100.000

      Top 10 values of variable Funding
              (Sample of 100000)

       Obs     Funding       COUNT     PERCENT
        1         1          99808      99.808
        2         2            192       0.192
```

A logical data key is the one or many variables that uniquely identify each record. Data keys are said to be broken if multiple records have the same values. DataView displays completely duplicate lines. Duplicate records usually indicate a problem in the data and require investigation and correction. If no duplicates exist DataView will attempt to find the best machine searched logical data key. Figure 3 below shows an example of a composite key that DataView found.

Figure 3. Illustrative Output of DataView

```
Best Machine Searched Composite Key
(Total # of Records for Data Set Reported_members is
1,296,196 )

Product Lob Delivery_system Funding  Provider_network
Policy_holder_ID Zip Contract is the best machine searched
composite key.
```

In summary the macro DataView will:

- Show the # of records with missing values for each numeric variable;
- Show the # of records with blank or null values for each character variable;
- Calculate averages and grand totals of each numeric field;
- Detect duplicate records;
- Detect any constant variables;
- Display the top 10 values by frequency for each variable;
- Determine data keys; and

- Print the first 20 records

EFFICIENCY CONSIDERATIONS

Efficiency was a main consideration as DataView was designed to process large files. A few of the techniques used to maximize efficiency were:

- Use 'KEEP=' to reduce I/O;
- Minimize use of PROC MEANS by calculating grand totals and averages with operations in a DATA step;
- Minimal executions of PROC SORT; and
- Use of optional parameters for maximum controls

The PROC SORT procedure is one of the most resource-intensive in terms of CPU time, memory and I/O [1]. Minimizing the use of PROC SORT was an important consideration. DataView executes PROC SORT only once unless required by a user specified parameter.

The DATA step is also resource-intensive when dealing with large files. DataView uses only one DATA step. Additional efficiencies are gained by only outputting the last observation containing calculated measurements.

SAVING OUTPUT FROM PROC CONTENTS

Several advanced techniques are utilized in DataView. One of them is saving output from PROC CONTENTS. Often output from procedures are saved to SAS data sets using the "out=" option. This is most common in procedures such as PROC MEANS, PROC FREQ, or PROC REG. One often-overlooked use is in PROC CONTENTS. The output of PROC CONTENTS contains rich and useful information on variable names, types, formats, and the total number of observations in the SAS data set. This information can be saved in another SAS data set. See below for an example including output from SAS.

Figure 4. Saving Output from PROC CONTENTS

```
data claims;
   length dcn $11 clm_item_cde $4;
   input dcn line_nbr clm_item_cde paid;
   format paid dollar12.1;
   cards;
01045800615 1 84 1520.0
01045800615 2 84  833.8
01045800615 3 84   64.0
01045800615 4 84 1330.8
01045800615 5 84   54.0
;

Proc contents data=claims out=cont;

Proc print data=cont(keep=name type nobs);
   title "Sample Output";
Run;
```

```
                Sample Output
               17:09 Wednesday, July 23, 2003

Obs    NAME           TYPE   NOBS
 1     clm_item_cde     2       5
 2     dcn              2       5
 3     line_nbr         1       5
 4     paid             1       5
```

USING "INTO:" IN PROC SQL

Once variable names are available in a SAS data set the next step is to store them in macro variables. This is done with PROC SQL[2] and enables grand total calculations and other calculations for all variables without having to code or even know each variable.

In Figure 5 below the macro variables num_vars and char_vars are created, containing all numeric variable and character variable names respectively.

Figure 5. Using "INTO:" in PROC SQL

```
Proc sql noprint;
   select name
   into :num_vars separated by  " "
   from cont
   where type=1;

   select name
   into :char_vars separated by " "
   from cont
   where type=2;

quit;

%put &num_vars;

%put &char_vars;
```

```
                Sample SAS Log

3803 NOTE: PROCEDURE SQL used:
          real time           0.01 seconds
          cpu time            0.01 seconds
3813 %put &num_vars;

3814 line_nbr paid

3815 %put &char_vars;

3816 clm_item_cde dcn
```

CODING SAS USING CALL EXECUTE

Another technique used by DataView is the CALL EXECUTE routine. CALL EXECUTE allows varying SAS code to be executed in the middle of a DATA step. The syntax is illustrated below.

CALL EXECUTE('proc print; run');

Different values of each variable become part of the SAS code. The code below shows how CALL EXECUTE is used in a DATA step. In this example, PROC FREQ is executed four times, one for each of variables in original data set CLAIMS.

Figure 6. Using CALL EXECUTE

```
Data _null_;
    set cont;
    call execute('proc freq data=claims;'
      ||'table '
      || name
      ||   '/list missing; run;' );
run;
```

```
                  Sample Output
                    17:09 Wednesday, July 23, 2003

                The FREQ Procedure

clm_
item_       Cumulative           Cumulative
cde         Frequency  Percent   Frequency  Percent
===================================================
84                5     100.00         5     100.00
```

```
                   Sample Output
                     17:09 Wednesday, July 23, 2003

                 The FREQ Procedure

            Cumulative          Cumulative
Dcn         Frequency  Percent  Frequency  Percent
==================================================
01045800615       5     100.00      5      100.00

            Cumulative          Cumulative
line_nbr    Frequency  Percent  Frequency  Percent
==================================================
1                 1      20.00      1        20.00
2                 1      20.00      2        40.00
3                 1      20.00      3        60.00
4                 1      20.00      4        80.00
5                 1      20.00      5       100.00
```

RECURSIVE EXECUTIONS OF MACROS

In addition to the techniques discussed above, DataView also uses recursions of macros to detect composite data keys. A sub-macro carries out the testing and detection of each possible data key and is repeatedly executed until the best machine searched key is obtained.

SAS handles recursion of macros using several macro statements. One starts with %DO %WHILE(), and ends with %END. Statements between the two are executed repeatedly until the condition in the parentheses becomes false.

This is illustrated in Figures 7 and 8 below. In Figure 7, test data keys are prepared in a SAS data set and the macro variable STOPIT is set to "no."

Figure 7. Recursion of Macros – Preparation

```
%let stopit=no;

data cont;
    set cont end=eofile;
    length KEY $200;
    retain key;
    name=trim(name);
    if _n_=1 then key=name;
    else key=trim(key)|| ' ' || name;

    if eofile then call symput('allvars',
       key);

Proc print data=cont;
    var key;
    title "Keys to be Tested";
Run;
```

```
                Keys for Testing

   Obs    KEY
    1     clm_item_cde
    2     clm_item_cde dcn
    3     clm_item_cde dcn line_nbr
    4     clm_item_cde dcn line_nbr paid
```

In Figure 8, the sub-macro "repeat" is repeatedly executed in the macro "doit." It tests each possible data key starting with the longest and stops after the true data key is found.

Figure 8. Recursion of Macros – Example

```
%macro repeat;
%global key lastpart;

data cont;
    set cont end=eofile;
    if eofile then do;
        call symput('key',trim(key));
        call symput('lastpart',name);
        delete;
    end;
run;

data duplicate;
    set claims;
    by &key;
    if first.&lastpart*last.&lastpart=0
       then do;
       call symput('stopit','yes');
       output;
    end;
run;
%mend repeat;

%macro doit;

   proc sort data=claims; by &allvars;

  %do %while(&stopit=no);
     %repeat;
  %end;
run;
%mend doit;

%doit;

Proc print data=duplicate;
  title "Duplicates found "
    "when key " "&key is used";
run;
```

```
   Duplicates found when key clm_item_cde dcn is
                     used
                  09:27 Tuesday, August 19, 2003


                        clm_
   Obs      dcn         item_cde  line_nbr    paid

    1   01045800615   84         1      $1,520.0
    2   01045800615   84         2        $833.8
    3   01045800615   84         3         $64.0
    4   01045800615   84         4      $1,330.8
    5   01045800615   84         5         $54.0
```

CONCLUSIONS
Utilizing some often overlooked techniques such as saving output from PROC CONTENTS; some advanced techniques such as PROC SQL's "INTO:"; CALL EXECUTE; and iterative use macros, DataView gives analysts the capabilities to analyze large amounts of data without having to custom code SAS. The insight from DataView provides a clear picture of, and clues about the structure and integrity of data. This macro was used to expedite the testing of various new processes and data sources. Readers can apply the same techniques to their own data analysis and process testing.

REFERENCES

[1] SUGI 26, ***Writing the "Best" Program: The How and When of Efficient Programming*** by Frank C. DiIorio at Advanced Integrated Manufacturing Solutions, Co. Durham NC,

[2] SUGI 27, ***Using the Magical Keyword "INTO:" in Proc SQL,*** by Thiru Satchi at Blue Cross and Blue Shield of Massachusetts

ACKNOWLEDGMENTS

The authors would like to thank the following persons for their support: Ed Mailander, VP of Data Management and Analysis, Rob Edwards, Staff VP Actuary, Luba Neymit, Manager of Actuarial Data Analysis. The authors would also like to thank the following persons for their testing and comments: Mike Schwartz, Manager of Actuarial Systems, Brian Yasui, Manager of Membership Reporting, Richard Patterson, Senior Actuarial Systems Analyst, and Manny Ibara, Actuarial Systems Analyst IV.

CONTACT INFORMATION

Don Cooper, Director
Actuarial Data Management, Wellpoint,
One WellPoint Way
Thousand Oaks, CA 91362
don.cooper@wellpoint.com
Tel: 805-557-6226

Dennis Tang, Actuarial Associate III
Actuarial Data Management, Wellpoint,
One WellPoint Way
Thousand Oaks, CA 91362
dennis.tang@wellpoint.com
Tel: 818-234-3119