Streamlining Medicaid Enrollment Reporting and Calculation of Enrollment Trends Using Macro and PROC Statements in SAS

Deniz Soyer, MBA, District of Columbia Department of Health Care Finance

ABSTRACT

The Division of Analytics and Policy Research (DAPR) within the District of Columbia Department of Health Care Finance (DHCF) produces a monthly Medicaid enrollment report for the District’s Medical Care Advisory Committee (MCAC) to document enrollment trends in the District’s various Fee-For-Service and Managed Care programs. This retrospective report requires retrieving Medicaid eligibility data for multiple monthly enrollment spans, organization of the data by program type and month, and calculation of enrollment growth for programs of interest. Previously, DAPR used Microsoft Excel to organize multiple outputs of monthly enrollment data and perform calculations on trends. To minimize time spent to manually produce recurring reports, DAPR sought to develop a SAS program to streamline and automate its monthly report.

The use of %LET statements created macro variables to represent each monthly enrollment span, which allowed for the formatting and transposing steps, occurring later in the program, to automatically reference each month. Functions PUT and %SCAN were used to reference and redefine variable formats. PROC FREQ was used to obtain total enrollment counts, by month, for each Medicaid program. PROC TRANSPOSE was used to convert each month from an observation into a variable type, which allowed each row to display enrollment for each Medicaid program type, while the columns served to classify enrollment by month. DATA steps were used to perform calculations for enrollment trends.

INTRODUCTION

DHCF is responsible for administering the District of Columbia’s Medicaid program, which is the joint federal-state health insurance program providing health care coverage for low-income and disabled adults, children, and families. Eligibility for D.C. Medicaid is determined through a series of financial and non-financial requirements. Individuals eligible for D.C. Medicaid may be enrolled in a Fee-for-Service (FFS) program, Managed Care (MCO) program and may be dually eligible for Medicare. DHCF also administers the locally-funded D.C. Healthcare Alliance program, which serves low-income District residents who have no other health insurance and are not eligible for either Medicaid or Medicare.

DAPR is housed within DHCF’s Health Care Policy and Research Administration. DAPR’s roles and responsibilities include developing and conducting original data analysis, including research around health services utilization, expenditures, and program enrollment. Each month, DAPR produces an enrollment report for the D.C. Medical Care Advisory Committee (MCAC) monthly meeting. Below is an example of the standard template for the monthly MCAC Report:

Figure 1. MCAC Enrollment Report Template
DC MCAC is a federally mandated committee consisting of stakeholders, including consumers, advocates, community members and providers, who review Medicaid program operations and provide recommendations. The purpose of the monthly report is to provide a high-level portrayal of enrollment figures for the previous thirteen months and to report trends such as enrollment growth for programs of interest.

Up until early 2015, DAPR analysts used Microsoft Excel to organize multiple outputs of monthly enrollment data and perform trend calculations. The lengthy and cumbersome process took several hours to retrieve raw enrollment data, manually organize and manipulate the format, group the program codes into four high-level program categories, and finally calculate enrollment growth.

To minimize time spent to manually produce recurring reports, DAPR sought to develop a SAS program to streamline and automate its monthly report.

SUPPLEMENTING DATA STEPS WITH MACRO STEPS

DEFINING MACRO VARIABLE USING %LET

After retrieving raw enrollment data for the Medicaid and Alliance programs, our goal is to output total number of individuals enrolled in each program code group for the past thirteen months. In enrollment data, FFS program enrollments are grouped separately from MCO and the Alliance programs.

We use a series of %LET statements that will allow us to use generic names for each of the thirteen months of interest for the MCO and Alliance dataset. Each %LET statement defines a macro variable (MonthA through MonthM), each of which is supposed to represent a month of enrollment:

%LET MonthA = MCO_BEGIN_DATE LE '31MAR2014’d and MCO_END_DATE GE '01MAR2014’d;
%LET MonthB = MCO_BEGIN_DATE LE '30APR2014’d and MCO_END_DATE GE '01APR2014’d;
%LET MonthM = MCO_BEGIN_DATE LE '31MAR2015’d and MCO_END_DATE GE '01MAR2015’d;

We repeat this for each month variable until the thirteenth enrollment period of interest:

%LET Month1 = FFS_BEGIN_DATE LE '31MAR2014’d and FFS_END_DATE GE '01MAR2014’d;
%LET Month2 = FFS_BEGIN_DATE LE '30APR2014’d and FFS_END_DATE GE '01APR2014’d;
%LET Month13 = FFS_BEGIN_DATE LE '31MAR2015’d and FFS_END_DATE GE '01MAR2015’d;

The %LET statement allows us to define the dates of interest, which are the only variable values that change when we run this report each month. Any part of our code that required any manual manipulation by a DAPR analyst, we wanted to have up front in our code. Thus, once we defined our period of interest, we could run the full SAS program.

USING DATA STEPS TO GROUP ENROLLMENT DATA

Following our definition of macro variables using the %LET statement, we organized FFS and Medicaid Dual Eligible program codes into the program groups that are in the final output of the MCAC report. Table 1 displays the list of all FFS and Dual program groups:

<table>
<thead>
<tr>
<th>Medicaid Fee-For-Service</th>
<th>Total Medicaid Duals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Fee-for-Service</td>
<td>Duals Non-Institutional</td>
</tr>
<tr>
<td>Total SSI or SSI Related Beneficiaries</td>
<td>Duals Institutional LTC</td>
</tr>
<tr>
<td>SSI or SSI Related Beneficiaries</td>
<td>EPD Waiver Duals</td>
</tr>
<tr>
<td>EPD Waiver</td>
<td>DD Waiver Duals</td>
</tr>
<tr>
<td>DD Waiver</td>
<td>MFP DD Waiver Duals</td>
</tr>
<tr>
<td>MFP, EPD Waiver Disab Over 65</td>
<td>MFP, EPD Physical Disab Over 65 QMB</td>
</tr>
<tr>
<td>MFP, EPD Physical Disab Under 65</td>
<td>MFP, EPD Physical Disab Under 65 QMB</td>
</tr>
<tr>
<td>MFP, EPD Physical Disab Under 65</td>
<td>Qualified Medicare Beneficiaries (QMB)</td>
</tr>
</tbody>
</table>

Table 1. Medicaid Fee-for-Service and Dual Eligible Program Groups
Using a DATA step, we grouped program codes by program group using an IN statement embedded in an IF-THEN statement. Below is a partial example of what the DATA step would resemble:

```
Data Enrollment_FFS2;
Set Enrollment_FFS;
   If Program_Code in('PG_1', 'PG_2', 'PG_3', 'PG_4', 'PG_5') then Program_Type = 'Duals_Non_Institutional';
   If Program_Code in('PG_100','PG_6') then Program_Type = "EPD_Waiver";
Run;
```

In the example above we only show two of the IF-THEN statements, but in reality these statements are repeated for each series of program code numbers (variable = Program_Code) until all relevant program code numbers are grouped into the appropriate program group type (variable = Program_Type). We grouped program codes by program group using an IN statement embedded in an IF-THEN statement.

### REFERENCING MACRO VARIABLE USING %SCAN

The MCAC report requires us to group enrollment numbers by month using the following date format: yyyymm (i.e. March 2014 would be formatted as 201403). To this end, we need to refer back to the generic month variables that we previously created in our %LET statements.

In a DATA step, we would create a new variable called "yymm" whose values would be identified by extracting the month and year portion of the Month1 through Month13 variables created by the %LET statement. In the example below, we define the character variable "yymm" by using %SCAN and PUT functions. %SCAN extracts the desired year and month from each previously defined month1, month2, etc. variable. The PUT function turns them into the desired format. Notice the use of the macro variable we created in our %LET statement. The code string below depicts the process we outlined:

```
Data Enrollment_Month;
Set Enrollment2;
   If &Month1 then do;
      yymm=put(year(%scan(&month1,-1)),4.)||put(month(%scan(&month1,-1)),z2.);
      output;
   end;
   If &Month2 then do;
      yymm=put(year(%scan(&month2,-1)),4.)||put(month(%scan(&month2,-1)),z2.);
      output;
   end;
   We repeat the IF-THEN statement for each macro variable through Month 13:
      If &Month13 then do;
         yymm=put(year(%scan(&month13,-1)),4.)||put(month(%scan(&month13,-1)),z2.);
         output;
      end;
Run;
```

The result of this code string is that for each Medicaid beneficiary we will output an observation for each month in which they were enrolled in the program. For example, in our output we see that there is a Medicaid beneficiary who has been enrolled from November 2013 to the present. As we can see, there is an observation for each month of interest and an appropriate value for each month of enrollment under the variable yymm.

Output 1. Creating an Observation for Each Month of Enrollment
PROC FREQ AND PROC SQL IN ENROLLMENT COUNTS

Since we now have observations for each month of enrollment for each Medicaid beneficiary, we can now count the total number of beneficiaries enrolled in each month for total enrollment numbers by using PROC FREQ. The PROC FREQ output will then be followed by a DATA step creating a new variable called “Program_Type” and the output will look like this:

Output 2. PROC FREQ Output of Total Enrollment Numbers

Similarly, we use PROC FREQ to count the number of beneficiaries within each program type for each month of interest. Here is a sample of our partial output to give you a sense of what this looks like:

Output 3. PROC FREQ Output of Enrollment Count by Program Type

From here, we combined the two previous outputs into one dataset, and realized we had a small problem. Since the “yymm” variable is a character variable and not a date variable, we had to develop a method to keep each month of interest in ascending order for our report. In other words, it would not be acceptable to report enrollment numbers for each month out of order.

First, we created a brand new temporary table called “tmp” using PROC SQL. The SELECT statement retrieves each “distinct” output of the variable “yymm” from our previous data set. In our case, each distinct output would be each month of interest. We also place each month in order:
Proc Sql;
   create table tmp as
       select distinct yymm
       from libname.enrollmentcount;
   order by yymm;
quit;

Using a DATA step, we take the output we created in the PROC SQL step and create a new variable called “order,” which adds a value of 1 through 13 for each month:

   data libname.order;
   set tmp;
   order = _N_;
   run;

This step produces the following output:

Output 4. PROC SQL Output Specifying the Correct Order of Each Month

PROC TRANSPOSE AND A MACRO CALLED %TRANSPOSE

The ordered logic that we created using the PROC SQL and DATA step combination in the previous section, needs to be merged back into our dataset containing the enrollment counts for each program type by month. In addition, we also wanted to transpose the variable “yymm,” so that each month of interest is converted from an observation associated with a program type into its own variable. In order to allow us to do this, we used the following macro called %transpose developed by Arthur Tabachneck, Xia Ke Shan, Robert Virgile, and Joe Whitehurst:

   %transpose(data=libname.Enrollment_Merge,
              out=libname.Enrollment_Merge2,
              by=Program_Type,
              id=yymm,
              delimiter=_,
              var=Count,
              use_varname=,
              preloadfmt=libname.order)

Here’s how this macro works:

- The “data=” line refers to our dataset containing the enrollment counts for each program type.
- The “out=” line refers to the dataset name of our macro output.
- The “by=” line refers to the variable by which we want to group the data, in our case “Program_Type”
- The “id=” line refers to the variable that we want to transpose, which in our case is “yymm” – our months of interest
- The “delimiter=” line allows us to place an underscore “_” before the variable name, which is specified in the next line
- The “var=” line allows us to place “Count” or other text before each month. We need to do this because no SAS variable name that we ever want to reference in future DATA steps or PROC steps can start with a number. This way, we allow “Count_” to precede each “yymm” value.
- The “preloadfmt=” line refers to the macro to the dataset containing the ordered month variables.
Our output from this macro is a dataset called “libname.Enrollment_Merge2” and contains each Program_Type and each month of interest in order:

Output 5. Monthly Enrollment Count for Each Program Type in Chronological Order

In our final output we do not want to have “Count_” preceding each of our months of interest. In order to correct the variable names, we must transpose the data using the simple PROC TRANSPOSE step:

Proc Transpose data=libname.Enrollment_Merge2 out=libname.Enrollment_Merge3;
  ID Program_Type;
  Run;

Here is the output from this step:

Output 6. PROC TRANSPOSE Enrollment Months of Interest

This output shows that all program types have been converted from observations into variables, while a new variable called “_NAME_” is automatically created containing each month of interest. We use a DATA step featuring a COMPRESS function to remove “Count_” preceding each month of interest. Essentially, this step allows us to redefine the variable “_NAME_” by setting it equal to the COMPRESS function, which removes the text “Count_” from any value under the variable “_NAME_”:

Data libname.Enrollment_Merge4;
  Set libname.Enrollment_Merge3;
  _NAME_ = Compress(_NAME_,'COUNT_');
  Run;

At this point, we want to convert our months of interest from observations into variables once again. We use PROC TRANSPOSE to transpose our dataset by the variable “_NAME_”:

Proc Transpose data=libname.Enrollment_Merge4 out=libname.Enrollment_Merge5;
  ID _NAME_;
  Run;

Here is the output from this step:

Output 7. PROC TRANSPOSE Output with Finalized Month Values

We will use a DATA step containing the RENAME function to convert the variable _NAME_ to the variable “Program_Type” for our final enrollment count output:
If we remember back to our original data sources, the FFS raw enrollment data was separate from the MCO and Alliance raw enrollment data. Table 2 displays the list of all Managed Care and Alliance program groups:

<table>
<thead>
<tr>
<th>Medicaid Managed Care and Alliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Medicaid Managed Care (TMC)</td>
</tr>
<tr>
<td>Children and Families</td>
</tr>
<tr>
<td>CHIP Children</td>
</tr>
<tr>
<td>HSCSN</td>
</tr>
<tr>
<td>Immigrant Kids Program</td>
</tr>
<tr>
<td>0-133% FPL Expansion SPA</td>
</tr>
<tr>
<td>Over 133-200% Waiver</td>
</tr>
<tr>
<td>Alliance</td>
</tr>
</tbody>
</table>

Table 2. Medicaid Managed Care and Alliance Program Groups

Enrollment associated with Medicaid Managed Care and Alliance program is organized into these groups.

Our next step is to revisit our MCO and Alliance raw dataset and repeat each of the steps that we applied to the FFS dataset, such as referencing the macro variables MonthA through MonthM, PROC FREQ to count the enrollment by MCO and Alliance program type, using PROC SQL and a data step to place the months of interest in ascending order, and using PROC TRANSPOSE steps to put our enrollment numbers in the correct format. Upon completion of these steps, our final enrollment output for the MCO and Alliance enrollment resembles the format of our final FFS enrollment dataset.

FINALIZING THE MCAC REPORT

ENROLLMENT DATA OUTPUT

Once our MCO and FFS enrollment outputs are combined, we use a final DATA step containing a RENAME function to convert the variable _NAME_ to the variable “Program_Type” for our final enrollment count output:

Output 9. Final MCAC Enrollment Count Output
ENROLLMENT TREND REPORTING

In addition to reporting monthly enrollment numbers, the MCAC report also requires that DAPR analysts report enrollment trends. Specifically, MCAC members are interested in enrollment growth in select programs over the past 13 months. We can use a couple of DATA and PROC TRANSPOSE steps to output these figures.

First, we will replace each month value with a numerical assignment. Each month will be assigned a number in chronological order of the month’s occurrence. To do this, we will create a new variable called “yymm.” In addition, since the MCAC members are interested in specific programs, we have to use a keep statement to restrict our calculations to the variables of interest. I will only list a few of the variables of interest for this example, instead of providing the full list of variables of interest.

```sas
Data libname.calculation
(keep = yymm Total_Medicaid_Duals TMC Total_DHCF_Enrollment);
Set libname.FinalMCAC;
yymm = _n_;
Run;
```

Output 10. Creation of ‘yymm’ Variable Replacing Each Month Value with Numerical Assignment

Now, we need to transpose our data so that we can calculate the difference in enrollment between month 1 and 13. Using PROC TRANPOSE, we will transpose while keeping the months in ascending order. In addition, we will also use the PREFIX function to place “month” in front of “yymm” observation. In other words yymm = “1” will get converted to yymm = “months1.”

```sas
Proc Transpose data=libname.calculation Out= libname.calculation2
Prefix = months;
ID yymm;
Run;
```

Finally, using a DATA step, we can create a new variable called “Growth,” which will represent the percent change from month 1 to month 13 for each program type. We will use the RENAME option to change the automatically assigned name of the variable “_NAME_” to be “Program_Type.” Using FORMAT option, we will format the “Growth” variable to be a percent rounded to 2 decimal places. In this output, we are only interested in seeing the enrollment growth for each program type of interest, so we can remove the variables months1 through months13 once we have used them to calculate “Growth.”

```sas
Data libname.calculation_final (rename=(_NAME_ = Program_Type));
Set libname.calculation2;
Growth = (months13 - months1)/months1;
Format Growth percent8.2;
Drop months1-months13;
Run;
```

Output 11. Final MCAC Enrollment Growth Calculations
Our last step is to output these values to Microsoft Excel, and paste them into our standard MCAC template. Here is the final version of the MCAC report as it is provided to the MCAC committee members and the public:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Fee-For-Service</td>
<td>11,002</td>
<td>12,102</td>
<td>12,942</td>
<td>13,502</td>
<td>13,923</td>
<td>13,578</td>
<td>13,768</td>
<td>13,768</td>
<td>13,768</td>
<td>13,768</td>
</tr>
<tr>
<td>Total Bill for SS Related Rev</td>
<td>12,246</td>
<td>12,600</td>
<td>13,057</td>
<td>13,401</td>
<td>13,501</td>
<td>13,200</td>
<td>13,502</td>
<td>13,502</td>
<td>13,502</td>
<td>13,502</td>
</tr>
<tr>
<td>SS or SS Related Revs</td>
<td>24,203</td>
<td>25,600</td>
<td>26,957</td>
<td>27,301</td>
<td>27,501</td>
<td>27,200</td>
<td>27,502</td>
<td>27,502</td>
<td>27,502</td>
<td>27,502</td>
</tr>
<tr>
<td>HED Waiver</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>CO Waiver</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>MFP/HD Waiver (Def.)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MFP/HD Waiver (Excl.)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Total Enrollment</td>
<td>17,562</td>
<td>19,102</td>
<td>19,857</td>
<td>20,501</td>
<td>20,701</td>
<td>20,502</td>
<td>20,702</td>
<td>20,702</td>
<td>20,702</td>
<td>20,702</td>
</tr>
</tbody>
</table>

Figure 2. Finalized MCAC Enrollment Report

REVIEW OF METHODOLOGY

Data steps were combined with macro and PROC statements to develop a SAS program for reporting. %LET statements created macro variables to represent each monthly enrollment span, which allowed for the formatting and transposing steps, occurring later in the program, to automatically reference each month. PUT and %SCAN functions were used to reference and redefine variable formats. PROC FREQ was used to obtain total enrollment counts, by month, for each Medicaid program. PROC TRANSPOSE was used to convert each month from an observation into a variable type, which allowed for each row to display enrollment for each Medicaid program type, while the columns served to classify enrollment by month. Finally, we used a series of DATA steps to calculate enrollment growth for Medicaid and Alliance programs of interest.

Using essential macro and PROC statements in SAS, DAPR was able to organize data, calculate trends and output a finalized report related to monthly Medicaid program enrollment, thereby resulting in near-automation in its reporting process.

CONCLUSION

Adopting the use of macro statements alongside data steps and PROC statements in SAS enables greater automation and accuracy in periodic reporting. DAPR has incorporated the combined use of macro and PROC statements in an effort to streamline other recurring reports.

REFERENCES


**ACKNOWLEDGMENTS**

Many thanks to Yiping Zhao and John Wedeles for their contributions in reviewing this paper. Their suggestions and feedback were invaluable.

**CONTACT INFORMATION**

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

- **Name:** Deniz Soyer
- **Enterprise:** District of Columbia Department of Health Care Finance, Division of Analytics and Policy Research
- **Address:** 441 4th Street NW, Suite 900S
- **City, State ZIP:** Washington, DC 20001
- **Work Phone:** 202-442-4625
- **E-mail:** deniz.soyer@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are trademarks of their respective companies.