Paper 132-27

Web-Based Tracking of the WA State Medicaid Caseload

Pete Lund, Northwest Crime and Social Research, Olympia, WA
Dave Bendemire, WA State Caseload Forecast Council, Olympia, WA

Introduction

As a separate agency authorized by the Washington state legislature in 1997, the Caseload Forecast Council (CFC) is responsible for the oversight and approval of official caseload forecasts. The CFC staff forecast caseloads three times annually. The forecasts are then presented to the formal CFC for adoption.

The Medical Assistance Administration (MAA) forecast is second only to the Kindergarten – 12th grade forecast in complexity, size, and dollars represented. In FY 2000, over 750,000 Washington residents were eligible for services administered by the MAA. Authorized spending for the MAA in 2001-2003 is $5.9 billion, supporting 961 full-time positions and program priorities for sixteen client categories. Each MAA client category is forecast separately using time series models.

The accuracy of a time series forecast is largely dependent on the accuracy and completeness of the historical data used as a basis for the forecast. The "garbage in, garbage out" rule definitely applies here. Therefore, it is important that CFC staff closely monitor MAA monthly caseload data for accuracy and completeness. Without good historical data to begin with, a forecast can vary unacceptably from actual caseload counts.

Forecast variances can result in an adverse dollar impact to the MAA budget. For this reason, a technical workgroup comprised of financial, legislative, budget, MAA, and CFC staff meet regularly to discuss future trends and forecast assumptions. Workgroup members must be confident that future trends and assumptions are built from complete and accurate caseload information.

For example, CFC staff recently noticed an unusual rise in the Medically Needy (MN) Aged caseload beginning September 2000. By April 2001, the caseload was back to previous levels. Was this a one-time phenomenon, a new trend, or just plain bad data? Without knowing why trends are changing, there is a much greater chance of producing an incorrect forecast. The MAA Program Tracking website was developed to assist CFC staff and other workgroup members in producing forecasts based on sound forecast assumptions.

Let's take a journey through some of the MAA Program Tracking pages used by CFC staff to determine the cause of the rise and fall of the MN Aged caseload counts.

MAA Program Tracking Page

"Welcome to the MAA Program Tracking Page!" This is the greeting each person receives when visiting the home page. The home page provides an introduction stating the goal of the website, describing the sources of the data used to track and monitor the MAA caseload, and describing other information available from the website (Figure 1).

The home page provides hyperlinks to very useful MAA caseload information. There's a link to a "What's New!" page. Another link provides client category titles and descriptions. Links to pages that are very helpful in understanding caseload trends are available from the home page. Most of these pages contain static information that is updated monthly.

So What About the MN Aged Example?

Let's get back to our example where the MN Aged caseload shows a rise and a fall between September 2000 and April 2001. Could this be just bad data?

Fortunately for CFC staff, there are two data sources from which raw (vs.lagged) MAA caseload counts can be pulled. The two data sources are the Incurred Expense Report (IER) and the Eligibility File (EF). The monthly counts pulled from these two sources should equal.

By choosing the "EF – IER Comparison" option from the home page, CFC staff can look at a graphical comparison of the IER vs. EF monthly caseload counts. If the line that represents the EF is not directly on top of the line that represents the IER, then there is a discrepancy between the two data sources. Discrepancies most often occur when the program logic that produces the EF no longer is in sync with the logic that produces the IER. In our MN Aged example, a discrepancy between the two data sources did not exist.

In our example, there was a rise in the caseload counts around September 2000. What caused this rise? Was there an increase in new clients entering (entries) the MN Aged program? Was there a decrease in the number of clients leaving (exits) the program? Did the number of clients moving into MN Aged from other MAA programs (transfers in) increase? Maybe clients moving out of MN Aged to other MAA programs (transfers out) decreased.

Wouldn't it be great if we had a way to view the entries, exits and transfers that occurred in the MN Aged program around September 2000? Well, it just so happens we do have a way to graphically view these counts using the MAA Program Tracking website.

Entries, Exits and Transfers

The MAA Program Tracking website provides comprehensive information that assists in determining causes for changes in caseload trends. Clicking on the "Entries, Exits, and Transfers" option from the home page is a good place to begin. The first screen that appears is the "Components of Growth" page (Figure 2). This page contains a stacked bar chart with "transfers in" stacked on top of "entries" and "transfers out" stacked on top of "exits". If we click on either the bar or legend for entries or exits, then an additional chart appears with 18 months of entries and exits displayed (Figure 3).

In the MN Aged example, we noted that entries began to rise in September 2000 and then fell back to previous levels by April 2001. This behavior in entries corresponded to what we saw in the raw caseload counts taken from the IER and the EF for the same period of time. The exits were also on the rise during that time but not enough to counter balance the rise in entries.

What about transfers in and transfers out? If we return to the "Components of Growth" screen (Figure 2) we can then click on the transfers in bar or legend. On this screen we see the top three MAA programs over the past six months that have contributed to the growth in the MN Aged program via transfers (Figure 4). We can then click on a program bar and view the past 18 months of transfers into MN Aged from that program (Figure 5).

We can also view transfers out in the same way that we view transfers in by going back to the "Components of Growth" screen (Figure 2) and then click on the transfers out bar or legend. By following the same steps as previously for transfers in we can drill down for additional transfers out information.

In our example, the transfers into the MN Aged caseload from other MAA client categories showed an increase between September 2000 and April 2001. Transfers out remained fairly constant during the period in question. This information led to the discovery that the rise and fall of new entries and transfers into the MN Aged program between September 2000 and April 2001 corresponded to the rise and fall of the raw data for the same months in the IER and EF.

With evidence of increases in entries and transfers during the period in question, CFC staff asked workgroup members to see if they could find out the reason for the increases. Upon further investigation, MAA staff discovered that 132,000 mailings had been sent out to Medicare recipients informing them that they may be eligible for cost sharing programs from the MAA. This mass mailing went out between September 2000 and March 2001 and greatly

contributed to the increases. Voila! We found it!

Other Tools Available

Other tools available from the MAA Program Tracking website also assist in the investigation of changes in caseload trends. A screen showing the caseload carry-over from last month is available from the home page. This screen shows 18 months of carry-over percentages for selected programs.

Another screen available from the home page is the "Forecasting Categories – Split by IER Category" accessed by clicking on the "IER Category Counts" option. This screen displays IER categories and the percentage each IER category contributes to the major forecast categories. This screen is very helpful when beginning an investigation of lag factors.

Lag Factors

The goal of the lag adjustment process is to estimate, from observed changes, the value of a particular month of service when it is fully mature. The first step in the lag process is to get an estimate of historical change. This is done by getting the average change for each "look" at a month of service value: how much it changed from the 1st time we saw it to the 2nd, from the 2nd to the 3rd, from the 3rd to the 4th, and so on. The average change ratios, or lag ratios, are used to calculate the lag factors. These lag factors are then applied to the raw caseload data to estimate mature data.

The MAA caseload is reported at the "forecasting category" level. However, the lag adjustment to the caseload is done at the IER category level. When we select "Lag Information" for a particular client category from the home page, a screen appears containing a list of the IER categories that make up the forecasting client category selected (Figure 7).

The first item in the lag tracking drop-down-list displays the change ratios for each "look" at the month of service counts. The initial display will be the ratios that go into the calculation of the average 1st to 2nd look change (Figure 8).

In a perfect world we would see that the lagged data is the same month after month because we are able to perfectly predict how the data will mature. This is never the case, and the lagged data does change. What we would hope to see is that those changes are random, sometimes a little high and sometimes a little low. What we would also hope to see is that the changes get smaller and smaller as the data gets older. When selecting the "Change in Lagged Date – Plot" option a box and whisker plot is displayed to show how the lagged data has changed over the past months (Figure 8).

Other options available from the drop-down box are tables, reports and plots that show how the lag factors influence the monthly client category caseload counts.

Okay, So How Does All Of This Work?

There are over 10 programs in the process, which creates over 1,500 files – both graphics and html. The incoming data files contain millions of observations and yet the entire process takes only a few minutes to run. There are a number of techniques used which are transferable to other applications.

Note: Obviously there is not enough space in ten pages to do any of these techniques justice. Rather than pick one to focus on in detail a number will be highlighted so as to whet your appetite as to what is possible.

Drill-down Graphics

To create a drill-down graphic in a web page all that is needed is an HTML image map using <map> and <area> tags. The map tag contains the name of the map. There is an area tag for each clickable portion of the the graphic. The area tags contain the shape, the x,y coordinates that define that shape and the link to take when the user clicks inside the defined area. Here's an example of the image map portion for one of the entry/transfer bars in Figure 2.

```
<MAP NAME="my_map">
   <AREA SHAPE="RECT" HREF="to_1005.html"
COORDS="524,108,532,160">
```

```
  <AREA SHAPE="RECT" HREF="EE_1005.html"
COORDS="524,160,532,299">
```

There is an area tag for each bar segment – one that links to transfer information and another that links to entry/exit information. There are similar area tags for each bar segment on the entire graph. To use the image map, simply add the USEMAP= option to the <image> tag for the graphic.

```
<IMG SRC="cog_b.gif" USEMAP="#my_map">
```

This is pretty simple for a single graph, even with a lot of segments. There are numerous software packages that allow you to just point and click at the corners of a graph region and it generates the x,y coordinate list. But remember, this process creates hundreds of graphs and doing the image maps by hand would be a daunting task.

Fortunately, ODS and a couple new SAS/Graph options take care of all of this for us. In the graph in Figure 2 the user can drill down to three different detail graphs: entries and exits, transfers in and transfers out. The dataset contains three variables that denote the type of data. We're going to use this information to create a new variable that contains the name of the page to link to if the user clicks on a bar that contains this observation.

This variable contains the entire file reference portion of the area tag, including the "HREF=." To get the links in the example above, the values would be:

```
"HREF='to_1005.html'"  _ and _
"HREF='ee_1005.html"
```

This new variable is referenced in PROC GCHART with the HTML= option. When the HTML= option is specified, and ODS HTML is used, SAS will automatically generate an image map and place it in the ODS-generated HTML. SAS will use the information in the HTML= variable to create the links in image map. For example,

```
proc gchart data=eet2;
  vbar main / group=mos
              subgroup=sub
              html=COGdetail
              html_legend=COGdetail
              legend=legend1
                     :
```

Obviously, it's important that all the observations in a bar group have the same value for the HTML= variable. Otherwise, you never really know where the drill-down is going to take you.

There is another new SAS/Graph option that works much the same. It is HTML_LEGEND=. This option also references a variable name which contains link information. Using this option causes SAS to generate an image map around the graph legend, making it drillable as well. The HTML_LEGEND= and HTML= options can be used together in the same procedure and can have the same or different values. In the above example, both options reference the same variable (COGdetail), so clicking on either a bar or its corresponding legend entry would link to the same file.

Note: there can be problems if the link file name contains blanks. Some browsers will truncate the file name at the first blank and obviously the hyperlink will not work correctly. An easy way around this is to replace any blanks in the file name with "%20", which the browser will translate to a blank. The single statement used in this process to accomplish this is:

```
COGdetail =
  tranwrd(trim(COGdetail),' ','%20');
```

The file name can be stored or constructed with blank spaces. The above code simply converts all the blanks in the string to "%20".

Clicking on any of the entry or exit bars (or legend entries) in the graph in Figure 2 would take you to Figure 3, showing a slightly different view of just entries and exits. Clicking on any of the "transfer in" bars (or legend entry) would take you to Figure 4, which shows the four major programs transferring into the program in question. Six months of groups bars are shown in the graph in Figure 4. The bars, and the legend, in the graph have been set up to be drillable as well. Clicking on any of the bars, or legend, will take you to a graph like the one shown in Figure 5 which has 18 months of detail for a particular program transfer combination.

So, with just a couple clicks of the mouse the user can go from high-level program growth information to see the long-term trend of a single component of that growth. As noted earlier this information can quickly lead to a determination or confirmation of why the caseload for a program has changed.

Consistency in the Drilldowns

There are two aspects of the graphics, displayed in Figure 4, which need to be maintained as the user drills down. First, though it's difficult to see in the black and white copies of the graphs included in this paper, the bar color. If the user clicks on a green bar the bars in the next graph should be green. Second, the scale of the original graph should be maintained.

Bar Color

The bars in the grouped bar charts, as shown in Figure 4, will always have the same four colors: magenta, blue, green and red.

```
%let color1 = cx840052;
%let color2 = cx006b84;
%let color3 = cx218429;
%let color4 = cxad0000;
```

PATTERN statements then reference the assigned colors for the grouped bar chart, for example:

```
pattern1 c=&color1;
pattern2 c=&color2;
```

There is a macro variable, &Keepers, that contains a list of the top three program codes in the order that was displayed in the grouped bar chart. For the graph shown in Figure 4, the value of &Keepers would be:

```
1020,1140,1100
```

Now, we can loop through and create the appropriate graph and set the correct color. Here is a greatly abbreviated section of code:

```
%do i = 1 %to 3;
  %let ThisProg =
    %scan(&keepers,&i,%str(,));

  pattern1 c=&&color&i v=solid;

  < GCHART here using &ThisProg to
    subset the correct program values >

%end;
```

In the example above, when &i is 2 &ThisProg will be 1140 and the color will be set to &Color2 (cx006b84). This is the same color that the 1140 bars had in the grouped bar chart. This simple method assures that the bars in the drilled graph will always be the same as the bar that was clicked.

Graph Scale

The other aspect of the drilled graph that was maintained was the vertical axis scale. This means that we have set the axis order. A little macro was used that calculates a round number that is about 10% greater than the maximum value and a by value that divides that value into six to nine segments. These values are stored in macro variables &RndMax and &ByVal respectively. By using these in an AXIS statement

```
axis1 value=(f=Arial) label=none
      order=0 to &RndMax by &ByVal;
```

we can control the scale of the axis. Then, the same axis is referenced in the RAXIS= option is both the grouped bar GCHART and the drilled GCHARTs and the vertical axes are the same.

User-Generated HTML and JavaScript

The goal of the entire process is to create a set of files that can be accessed from any network accessible drive. There is no server-side scripting used and there is no need for the files to be "hosted" on a machine running a web server.

As noted earlier, there are hundreds of HTML files produced by this process. The pages that contain graphs are, for the most part, generated by SAS using ODS HTML. There are a few pages that contain a mixture of tables and text (Figure 7). There are also a couple places where web pages generated by the SAS process need to have some interactive functionality (Figure 8). This is all client-side scripting done with JavaScript that is generated by the SAS programs.

Generating HTML

In these cases a working prototype of an HTML page and any necessary scripts was

created. When everything was working satisfactorily the code was placed into PUT statements in a DATA _NULL_ datastep. For example, the following is part of the HTML file shown in Figure 7:

```
<table> <tr>  <td> </td>
<td>IER Category</td>
<td>Average<br>Eligibles</td>
<td>Percent of<br>Total Eligibles</td>
</tr>  <tr>
<td><form method="POST" name="SelIER">
<input type="radio" name="IERcats"
value="07" checked name="R1"></td>
<td>(07) Medicaid only</td>
<td>271</td><td>2.92%</td>  </tr>
```

Notice that the page shown in Figure 7 is too much of a hodge-podge to be generated with ODS. There is a table of data with embedded radio buttons, a drop-down list box and a number of text entries. The items in bold in the example above are specific to this Medicaid program and will change each month. This is where we can take advantage of the SAS language to create this HTML file.

The static portions of the page are simply replaced as written in PUT statements. The text for the portions that are variable are created with simple programming logic. To create part of the above

```
  put '<tr>';
  put '<td>';

  if _n_ eq 1 then put '<form
method="POST" name="SelIER">';

  put '<input type="radio" name="IERcats"
value=' IERcat;

  if _n_ eq 1 then put ' checked ';

  put 'name="R1"> </td>';
  put '<td>' IERttl '</td>';
  put '<td>' AvgElig comma7. '</td>';
  put '<td>' PctElig percent'9.2 '</td>';
```

In this way the values for each program component are loaded into the HTML table. The values in the table will always be the current values. It is very helpful to work from a prototype, rather than simply coding the HTML straight into PUT statements. This way you know that the layout is what you want and later comparing the SAS-generated HTML to the prototype makes it much easier to debug if it comes out looking a little different.

Generating JavaScript Functions

The graph shown in Figure 8 is actually one of 22 graphs showing the lag ratios for IER Category 22 (see above discussion on lags for more detail). The HTML page is generated in a DATA _NULL_ step and contains JavaScript functions that make the navigation bar at the bottom of the page cycle through all the different graphs.

As in the previous example a working prototype of the page was created and used as a template. The following JavaScript function controls the changing of the graphic image as the left or right arrows are clicked.

```
function scroll(increment) {
  if (increment == -1) {current --}
  else {current ++}
  if (current > 21) {current = 21}
  if (current <= 0) {
    current = 0
   source = 'images\\i22_g.gif'}
  else {
   source =
    'images\\i22_g'+current+'.gif'}
  document.images[0].src = source}
```

Note that the IER category (22) is part of the file name that is loaded. The SAS code that generates this HTML is very similar to the code we looked at earlier.

```
put 'function scroll(increment) {';
put 'if (increment == -1) {current --}';
put 'else {current ++}';
put "if (current > %eval(&maxgap-1))
 {current = %eval(&maxgap-1)}";
put 'if (current <= 0) {';
put 'current = 0';
put "source = images\\&prefix._g.gif'}";
put "else {source =
 'images\\&prefix._g'+current+'.gif'}";
```

The IER category (22 in the above example) is passed in a macro variable, &prefix, where it is inserted into the JavaScript statements. In this way a single prototype template can be used to generate any number of functional HTML pages without any manual intervention.

ODS and DATA _NULL_ Reporting

In addition to capturing procedure output and rendering it for a multitude of destinations, including HTML, ODS can also be used to greatly simplify DATA _NULL_ reporting. The table in Figure 6 looks like the result of PROC REPORT. In fact, it was produced by

a DATA _NULL_ - a very short and simple one at that.

First, notice the structure of the table in Figure 6. There are four columns of data with headers in the first row. Also, repeating information is suppressed in the first column. Finally, the first two columns are left-justified and the last two columns are right-justified. This would present some challenges with traditional DATA _NULL_ reporting. Things have gotten much easier with ODS.

The DATA Step

There are a number of things to notice about the following datastep. First, there is no "logic" to determine the first row or the justification or anything else for that matter. In fact, the whole program is made up of a FILE statement and a PUT statement.

```
 data _null_;
   set ier.FcsByIER;
   by fcstcat;

   file print ods=(
     template='table1'

     columns=(
      num_var=fcstcat
         (generic=on format=cat2ttl.
dynamic=(colhd='Forecast Category))

      num_var=category
         (generic=on format=ier2ttl.
dynamic=(colhd='IER Category'))

      num_var_right=avgelig
         (generic=on format=comma12.
dynamic=(colhd='Average Eligibles'))

      num_var_right=pctelig
         (generic=on format=percent9.2
dynamic=(colhd='Percent of Total'))
           )
        );

     put _ods_;
run;
```

But, they are special FILE and PUT statements! Let's take a quick look at these two statements.

Generally, when we see FILE PRINT we assume that the output is going to the output window. However, with ODS= option FILE PRINT will send our DATA _NULL_ output to the current ODS destination(s). Also, the FILE statement is usually pretty short – in this case it makes up most of the program.

There are two components to the ODS= option. First, almost all ODS output makes use of a table template to describe how the output will be laid out and FILE PRINT ODS= is no exception. The first thing coded is the name of the template to use, in this case TABLE1. We'll look at this template later in this section. Second, there is a definition of the columns that make up the table. This is done with the COLUMNS= suboption. Notice that there are four columns (variables) listed: FcstCat, Category, AvgElig, PctElig. These are the variable names in the dataset used in the datastep, ier.FcsByIER.

The Table Template

Note also that the columns contain another reference, NUM_VAR and NUM_VAR_RIGHT. These name the sections in the template that define how the variable values will be displayed. The base procedure PROC TEMPLATE is used to create or edit ODS templates. Here is the code that defines the template TABLE1.

```
proc template;
   define table Table1;
      dynamic colhd;
      mvar sysdate9;
      column num_var num_var_right;

      define num_var;
         header = colhd;
         style = cellcontents;
         generic;
         blank_dups;
      end;
      define num_var_right;
         header = colhd;
         just = r;
         style = cellcontents;
         generic;
      end;
      classlevels;
   end;
run;
```

This is where the definitions for NUM_VAR and NUM_VAR_RIGHT are stored. There are two types of columns in the Table 6, left-justified (the first two columns, defined by NUM_VAR) and right-justified (the last two columns, defined by NUM_VAR_RIGHT). Notice that the COLUMNS= suboption in the datastep passes the column header to the template as COLHD (set to the HEADER= template parameter). The NUM_VAR definition contains the directive to blank

duplicates and the NUM_VAR_RIGHT definition contains the directive to right justify.

There are many options that can be set for table layout. By defining the styles you want your output to have and referencing those in the COLUMNS= option of the FILE statement DATA _NULL_ reporting takes on a whole new world of possibilities.

Conclusion

The goal of this paper has been to give an overview of our web-based tracking process and give an introduction to some of the techniques used to create the output files. We hope that this will spur you on to investigate what you can accomplish using basic SAS tools. The sky's the limit!

Author Contact Information

Pete Lund
Northwest Crime and Social Research
215 Legion Way SW
Olympia, WA 98501
(360) 528-8970
pete.lund@nwcsr.com

Dave Bendemire
WA State Caseload Forecast Council
515 15th Ave SE
Olympia, WA 98504-0962
(360) 902-0086
dave.Bendemire@cfc.wa.gov

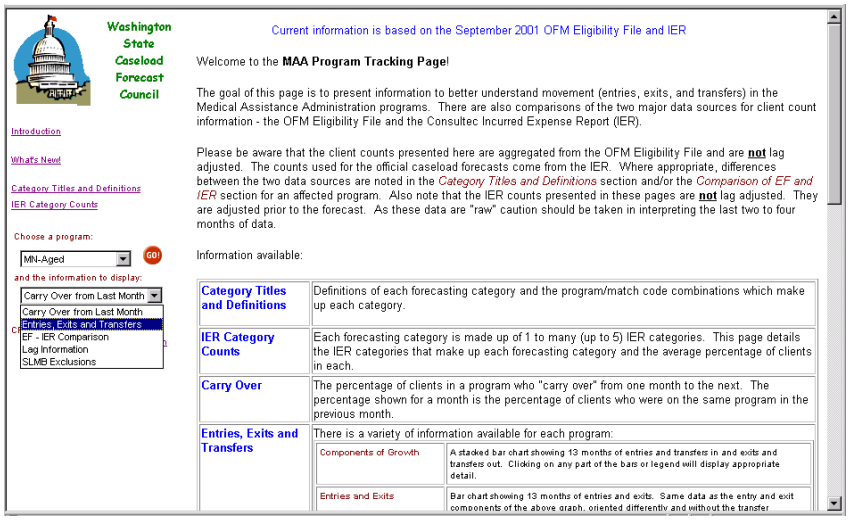

Figure 1

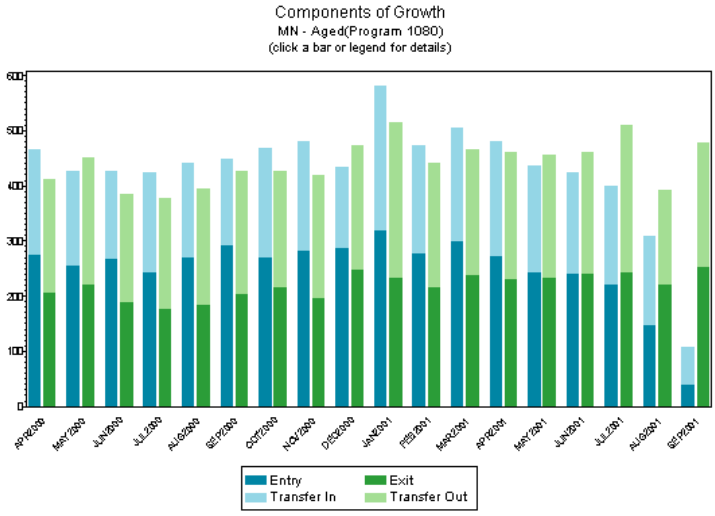

Figure 2

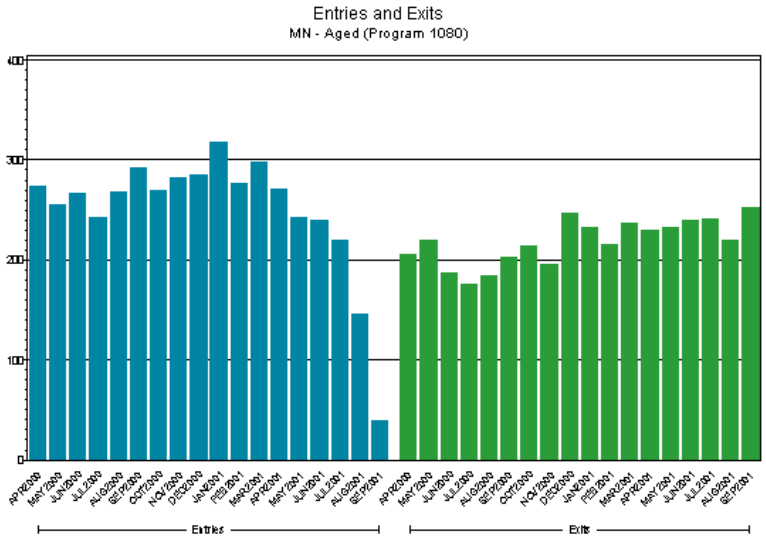

Figure 3

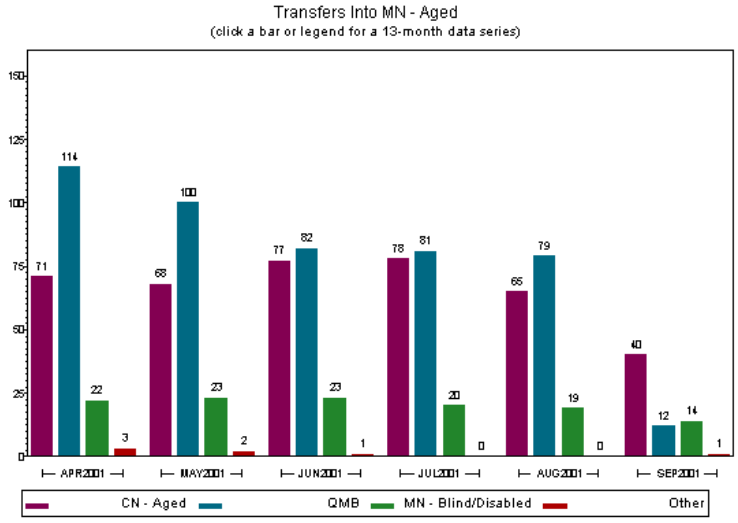

Figure 4

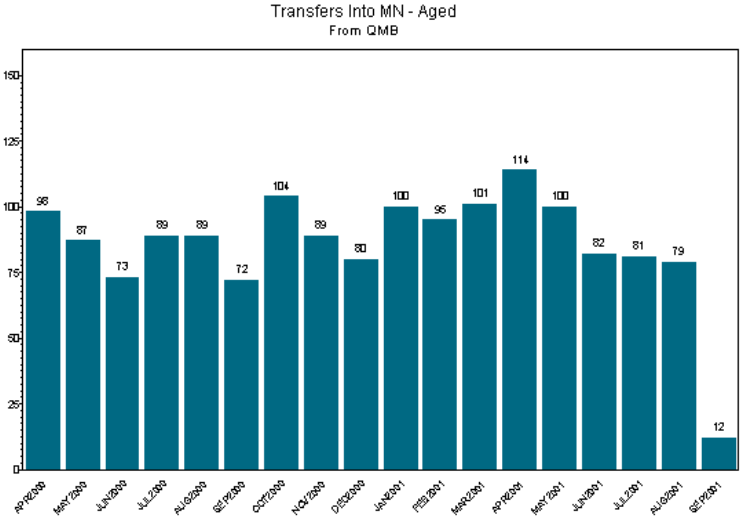

Figure 5

Forecasting Categories - Split by IER Category
From OCT2001 IER

(average eligibles from MAR2001 to AUG2001)

Forecast Category	IER Category	Average Eligibles	Percent of Total
(1005) CN - TANF	(04) TANF-R	273,476	99.71%
	(05) TANF-E	794	0.29%
(1006) TANF Reinstatements	(33) TANF Reinstatements	33	100.00%
(1020) CN - Aged	(01) Medicaid only	7,592	14.49%
	(19) Dual eligible	44,800	85.51%
(1040) CN - Blind/Disabled	(02) Medicaid only	129	0.12%
	(03) Medicaid only	69,167	64.87%
	(20) Dual eligible	29,256	27.44%
	(21) Dual eligible	0	0.00%
	(38) GA-X	8,076	7.57%
(1050) CN - Other Women	(36) Pregnant women	26,373	60.45%
	(39) Pregnant women	17,254	39.55%
(1055) CN - Other Kids	(06) Foster Care	79,936	25.81%

Figure 6

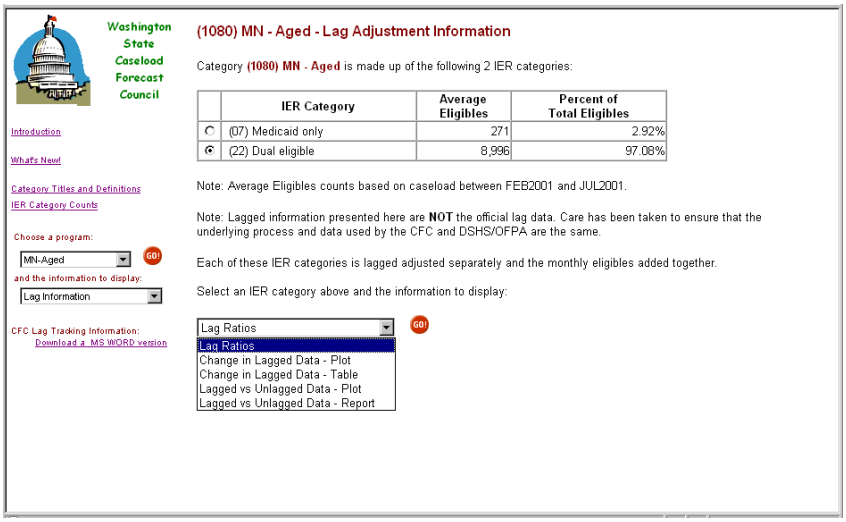

Figure 7

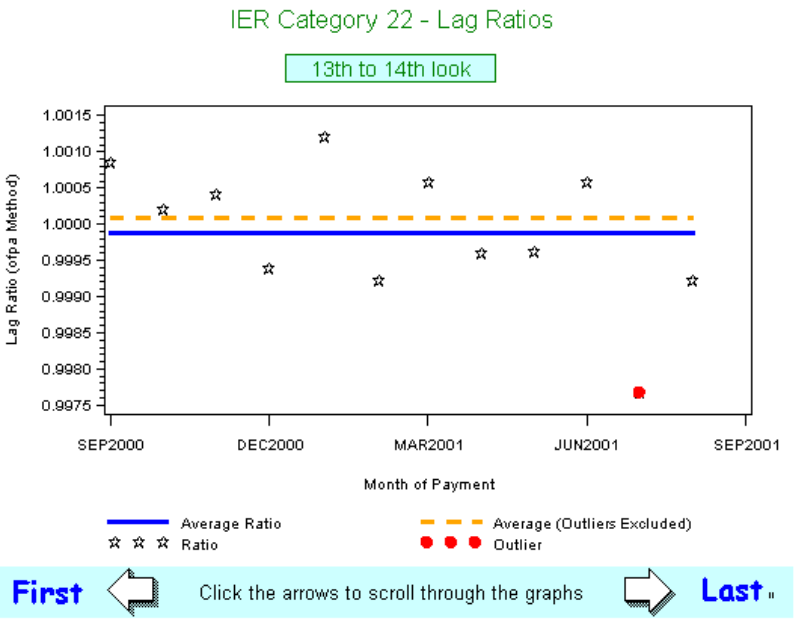

Figure 8