Paper 10040-2016

Something Old, Something New: Flexible Reporting with DATA Step-based Tools

Pete Lund
Looking Glass Analytics, Olympia, WA

Abstract

The report looks simple enough—a bar chart and a table, like something created with the GCHART and REPORT procedures. But, there are some twists to the reporting requirements that make those procedures not quite flexible enough. The solution was to mix "old" and "new" DATA step-based techniques to solve the problem. Annotate datasets are used to create the bar chart and the Report Writing Interface (RWI) to create the table. Without a whole lot of additional code, an extreme amount of flexibility is gained.

The goal of this paper is to take a specific example of a couple generic principles of programming (at least in SAS®):

1. *The tools you choose are not always the most obvious ones* – So often, other from habit of comfort level, we get zeroed in on specific tools for reporting tasks. Have you ever heard anyone say, "I use TABULATE for everything" or "Isn't PROC REPORT wonderful, it can do anything"? While these tools are great (I've written papers on their use), it's very easy to get into a rut, squeezing out results that might have been done more easily, flexibly or effectively with something else.
2. *It's often easier to make your data fit your reporting than to make your reporting fit your data* – It always takes data to create a report and it's very common to let the data drive the report development. We struggle and fight to get the reporting procedures to work with our data. There are numerous examples of complicated REPORT or TABULATE code that works around the structure of the data. However, the data manipulation tools in SAS (data step, SQL, procedure output) can often be used to preprocess the data in such a way to make the report code significantly simpler and easier to maintain and modify.

The Project

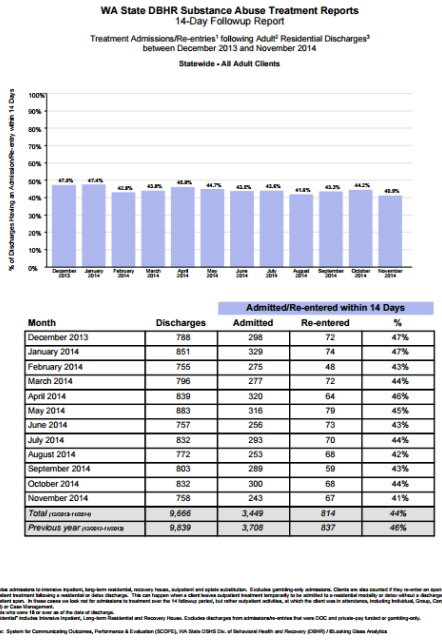
Our company provides an analytic website to the Division of Behavioral Health and Recovery for the state of Washington. A report was requested that showed the percentage of people leaving residential substance abuse treatment that had subsequent follow-up admissions to treatment within 14 days. (That's a good thing, showing that client are being followed after discharge to make sure they get continuing care.) The report was to have a 12-month bar chart, showing the percentage of follow-up admissions, and an accompanying table that also showed the number of discharges and broke down the subsequent admissions into a couple types.

The initial data to support this report was simple: one observation for each month, with just five data columns; the month of discharge, number of discharges, and the number and percentage of follow-up admissions.

A GCHART procedure for the chart and a REPORT procedure for the table would make the reporting code simple as well. Minus the statements and options dealing with the formatting of the results, which is admittedly quite a lot, it could be as easy as this… …and might produce a report like this

GCHART code:

```
proc gchart;
  vbar DischargeMonth / sumvar=PctFollowup
                        discrete;
run;
quit;
```



		Admitted/Re-entered within 14 Days		
Month	Discharges	Admitted	Re-entered	%
December 2013	788	298	72	47%
January 2014	851	329	74	47%
February 2014	755	275	48	43%
March 2014	796	277	72	44%
April 2014	839	320	64	46%
May 2014	883	316	79	45%
June 2014	757	256	73	43%
July 2014	832	293	70	44%
August 2014	772	253	68	42%
September 2014	803	289	59	43%
October 2014	832	300	68	44%
November 2014	758	243	67	41%
Total	*9,666*	*3,449*	*814*	*44%*
Previous year	*9,839*	*3,708*	*837*	*46%*

[1] Includes admissions to intensive inpatient, long-term residential, recovery house, outpatient and opiate substitution. Excludes gambling-only admissions. Clients are also counted if they re-enter an open span of outpatient treatment following a residential or detox discharge. This can happen when a client leaves outpatient treatment temporarily to be admitted to a residential modality or detox without a discharge from their outpatient span. In these cases we look not for admissions to treatment over the 14 followup period, but rather outpatient activities, or which the client was in attendance, including Individual, Group, Conjoint (with client) or Case Management.
[2] Clients who were 18 or over as of the date of discharge.
[3] "Residential" includes Intensive Inpatient, Long-term Residential and Recovery House. Excludes discharges from admissions/re-entries that were DOC and private-pay funded or gambling-only.

Source: System for Communicating Outcomes, Performance & Evaluation (SCOPE), WA State DSHS Div. of Behavioral Health and Recovery (DBHR) / Bluebird Glass Analytics

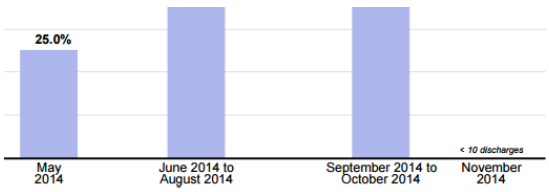
REPORT code:

```
proc report;
  columns DischargeMonth Discharges
          Had14Day_Adm Had14Day_Act PctFollowup;
  define DischargeMonth / order order=internal;
run;
```

It's Not Always the Obvious Tools

But, as we all know, it's often not quite this simple. In this case, there were additional requirements that made the simple approach impossible. This report can be run at the county and agency level and there are often a small number of discharges per month. To protect against volatile rates based on small n's, all rates shown had to be based on at least 10 discharges. Discharges need to be aggregated across months until the necessary 10 discharges are reached and the rate based on that aggregation. This has significant implications for both the chart and the table. Let's look at each…

The chart may now have fewer than 12 bars. If multiple months were aggregated, the bar should be centered in the space the monthly bars would have taken and the label should contain the range of months (e.g., "June 2014 to August 2014"). In addition, if the last month(s) in the report range do not have at least 10 discharges, then the bar should be replaced with the text, "< 10 discharges." This would be a bit more of a challenge to do with GCHART.



It was decided to take total control of the process and use the SAS/Graph Annotate Facility to create the entire graph with an annotate dataset. We'll look at the details after we look at the impact of the month aggregation on the table.

We still want to show the count of discharges and readmissions for each month in the table, but only show the rate for the aggregated months. This means that the final column in the table will span rows if aggregation was necessary. This cell will contain the rate and will show the count of aggregated readmissions and discharges.

Month	Discharges	Admitted/Re-entered within 14 Days		
		Admitted	Re-entered	%[4]
May 2014	12	3	0	25%
June 2014	3	2	0	
July 2014	6	3	0	60% (9/15)
August 2014	6	4	0	
September 2014	8	2	2	
October 2014	9	5	0	53% (9/17)
November 2014	3	1	0	*<10 discharges*

As with the chart, it would be difficult, if not impossible, to use our original plan of attack, PROC REPORT, to create the desired table. Again, we'll use a data step approach that allows us to take total control. In this case, the Report Writing Interface (RWI) will be used to create the table.

Getting the Data Right Can Make the Reporting a Lot Easier

So – principle #1 above comes true: the tools we might have started with on this project aren't the ones we'll use. But before we look at how we'll use other tools (an old one and a new one), let's look at principle #2 – how we can manipulate the data a little bit to make the reporting a bit easier.

The initial data, which we'll still use, has a row for each month with the discharge and follow-up admission counts. Now, we also need the aggregate information. To simplify the reporting process, we're going to compute the necessary information and add it to the original data.

```
retain TotalDischarges 0 StartMonth .;

if StartMonth eq . then StartMonth = DischargeMonth;

TotalDischarges + Discharges; ❶

if TotalDischarges ge 10 or done then ❷
  do;
    EndMonth = DischargeMonth; ❸
    NumMonths = intck('month',StartMonth,EndMonth)+1;
    PctFollowup = TotalFollowup / TotalDischarges;
    output;
    StartMonth = .;
    TotalDischarges = 0;
  end;
```

The code to the left is a snippet of the aggregation code – the number of the two follow-up admission types is done the same way as discharges. ❶ When the count of discharges is greater than 10 (or we reach the end of the dataset) ❷, the end month is set and the aggregate data is output. ❸

We now have four additional pieces of information: the start and end month of the aggregation period and the total discharges and readmissions in the time period. In reality, we'll add a couple more variables – the number of months in the time span and the percent with follow-up admissons. Obviously, these could be calculated in the reporting steps, but then that code would have to be repeated. This way, it's only there once and, with a 12-observation dataset, the extra space is not an issue.

Once the aggregation is done, we just merge the data, where the DischargeMonth in the original data matching the StartMonth in the aggregate data. In cases where there are always 10+ discharges, the discharge month, start month and end month will all be the same and the aggregate totals will be the same as the original monthly counts.

DischargeMonth	Discharges	Had14Day_Adm	Had14Day_Act	PctFollowup
December-13	9	3	0	0.3333
January-14	14	5	0	0.3571
February-14	11	7	1	0.7273
March-14	9	4	0	0.4444
April-14	7	3	0	0.4286
May-14	12	3	0	0.2500
June-14	3	2	0	0.6667
July-14	6	3	0	0.5000
August-14	6	4	0	0.6667
September-14	8	2	2	0.5000
October-14	9	5	0	0.5556
November-14	3	1	0	0.3333

StartMonth	EndMonth	TotalDischarges	NumMonths	PctFollowup
December-13	January-14	23	2	0.3478
February-14	February-14	11	1	0.7273
March-14	April-14	16	2	0.4375
May-14	May-14	12	1	0.2500
June-14	August-14	15	3	0.6000
September-14	October-14	17	2	0.5294
November-14	November-14	3	1	.

A Quick Tour of the SAS/GRAPH Annotate Facility

Even though it's over 15 years old, Art Carpenter's little book, *Annotate: Simply the Basics* (1999), is a great resource for starting with the SAS/GRAPH Annotate facility and he offers a great description: "The Annotate facility allows the use to create customized modifications to the graphic output. These modifications can be either predetermined or data-driven. This means that, through the use of Annotate, you can greatly extend the already powerful capabilities of SAS/GRAPH…. The power of the Annotate facility is accessed through the use of a specialized dataset. When using this dataset Annotate looks for variables with specific names, and the values taken on by these variables let Annotate know what your intentions are." (pp 2,3) In addition to adding information to SAS/GRAPH output, Annotate datasets can be used to create complete graphs and charts – and this is what we'll do. An Annotate dataset is no different than any other dataset in anyway other than the use of the prescribed variable names – much like CNTLIN datasets used with PROC FORMAT.

There are three required variables in an Annotate dataset: FUNCTION (what to do), X and Y (where to do it). With Annotate, I've always found it helpful to think back to the days of pen plotters. Each observation is a command to the "plotter." The FUNCTION variable has values like "MOVE", "DRAW" and "LABEL" and you can imagine what they mean in the plotter analogy. The X and Y variables define where to do that function. There are multiple coordinate systems available, set with the XSYS and YSYS variables. In our example report, the X and Y values reference the percentage of the entire graphics area. As can be seen below, there are many other variable that may or may not be used with different functions. See the documentation listed in the References section for details.

- Attribute variables
  - COLOR, CBORDER, CBOX
  - ANGLE, ROTATE, POSITION
  - HTML,IMGPATH, LINE
  - SIZE, STYLE
  - TEXT
  - WHEN
- Coordinate value-related variables
  - Z, XC and YC
  - CHART, MIDPOINT and SUBGROUP
- Internal coordinate variables (read-only)
  - XLAST, YLAST
  - XLSTT, YLSTT
- Coordinate system variables
  - HSYS, XSYS, YSYS, ZSYS

With over 20 values of the FUNCTION variable and over 30 other variables used by Annotate, it can be daunting to remember what and when things are needed. Fortunately, there are little shortcuts that can often be used. SAS supplies almost 30 macros that perform common annotate functions. In this report we'll use three of them.

1. %bar – writes two observations to the dataset, a MOVE function and a BAR function, with the following parameters: *x1*, *y1*, *x2*, *y2*, *color*, *line*, *style*
2. %label – writes a single observation with a LABEL function, with the following paramters: *x*, *y*, *text-string*, *color*, *angle*, *rotate*, *size*, *style*, *position*
3. %line – writes two observations to the dataset, a MOVE function and a DRAW function with the following parameters: *x1*, *y1*, *x2*, *y2*, *color*, *line*, *size*

The macro parameters all correspond to the variables in the dataset (see the variable list above). Notice that the same variables will be used in different ways by the different "functions." For example, STYLE in the LABEL function is the font of the label text, whereas STYLE in the BAR function is the type of fill to use for the bar. The macros do make it a lot easier to remember which pieces of information are necessary for the different Annotate functions.

Note: unlike other SAS-supplied macros, the Annotate macros are <u>not</u> available by default. You must submit the %annomac macro to make use of them.

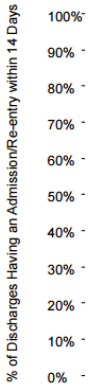
Creating the Chart – With a Data Step Instead of PROC GCHART

There are six steps to creating the bar chart and getting it on the page – all using less than 50 lines of code. Remember that our manipulated dataset has one row for each month, with StartMonth/EndMonth values attached to the first month of any set of aggregated months.

<u>Step 1</u> – the first iteration of a datastep (_N_ eq 1) is often used to process things that only need to be done once. ❶ In this case we'll set up a block of code to create the vertical axis label, ticks and values.

```
if _n_ eq 1 then ❶
  do;
    %label(2,52.5,'% of Discharges ... within 14 Days',black,90,0,.9,'Helvetica',5); ❷
    do i = 0 to 100 by 10; ❸
      ypos = (i*.95)+5; ❹
      %label(6,ypos,catt(i,'%'),black,0,0,.9,'Helvetica',6); ❺
      %line(9.5,ypos,10,ypos,black,1,1.5); ❻
      %line(10,ypos,90,ypos,cxe0e0e0,1,1); ❼
    end;
  end;
```

The %LABEL macro ❷ is used to create the axis label. The X,Y (2,52.5), ANGLE (90) and Position (5) parameters are used to place the label where we want it.



We then want to create 11 tick marks ❸, every 10% from 0 to 100 on the vertical axis. We want 0 line to be 5% from the bottom of the graphics area, so that there is room for the bar labels. Thus the vertical position of the ticks is offset by 5 and the gap is really 9.5 rather than 10. ❹ We'll use the same adjustment when we create the bars. The labels (0%, 10%, etc) are created with the %LABEL macro ❺ and then we draw two lines. The first line is very short, from X position 9.5 to 10 – this is the tick mark. ❻ The next line goes all the way from X position 10 to 90 and is gray in color (cxe0e0e0). ❼ This serves as a horizontal reference line at each of the decile values.

Step 2 – some quick logic is used to determine the center of each bar – the gap between the center of the bars is 6.5% of the graphics area. We'll see in the next section that each bar is 5% of the area wide. Without showing the four lines of math, we end up with a variable, C, that contains the center value of the bar on the X axis. The initial value is offset from the left edge, to leave room for the what we just did in Step 1, and, if all 12 bars were drawn, the values of C would be 13.75, 20.25, 26.75,…, 85.25.

Step 3a – now we'll check to see if there is a percentage value on the record ❶, meaning that we can actually draw the bars.

```
if not missing(PctFollowup) then ❶
  do;
    %bar(c-2.5,5,c+2.5,(95*PctFollowup)+5,cxADB6ED,0,'s'); ❷
    %label(c,(95*PctFollowup)+5,),put(PctFollowup,percent8.1),black,0,0,.7,'Helvetica/bold',2); ❸
  end;
```

A couple things to remember here: the variable C contains the center position of the bar (step 2) and the height of the bars will be from 5 to 100 (step 1). The %BAR macro ❷ draws a blue (cxADB6ED) bar that's 5% of graphics are wide, with X values of C-2.5 and C+2.5. The lower Y value is set at 5 and the top of the bar is based on the PctFollowup value. It has a value of 0-1, so multiplying by 95 and adding 5 gives a range of 5 to 100.

The %LABEL macro ❸ puts the value of the percentage on top of the bar. The label is placed at the center and top of the bar (C and the same maximum vertical position). The Position parameter value of 2 puts the text above the defined Y value.

Step 3b – if at the end of the reporting period the aggregate ten discharges has not been reached ❶ there will be a null value in the PctFollowup variable. In that case, instead of a bar a "<10 discharges" note will be displayed.

```
else ❶
  do;
    %label(c,5,'< 10 discharges',black,0,0,.5,'Helvetica/italic',2); ❷
  end;
```

< 10 discharges

The %LABEL macro puts the note on the graph, in an italic font. ❷ Its X position is the bar center value (C). The Position parameter value of 2 specifies that the text is centered above the Y position, which is 5

– the Y value of the horizontal axis. *Note: there are 15 different values of the position variable, defining the vertical placement (above, below, centered) and alignment (left, right, centered) of the text relative to the X,Y coordinate.*

Step 4a – there are two potential label styles for the bars, either a single month and year or a range of month/year values. To facilitate the desired display of the dates we'll always put the information into

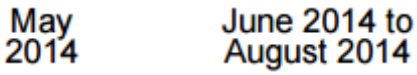
```
if NumMonths eq 1 then ❶
  do;
    DateText1 = put(StartMonth,MonName.);
    DateText2 = put(StartMonth,year.);
  end;
else ❷
  do;
    DateText1 = catx(' ',put(StartMonth,MonName.),put(StartMonth,year.),'to');
    DateText2 = catx(' ',put(EndMonth,MonName.),put(EndMonth,year.));
  end;
```

two character variables. In cases with a one-month bar, the month name goes into the first variable and the year goes into the second. ❶ In cases with a multi-month bar, the month and year of the start month, plus the word "to", goes into the first variable and the month and year of the end month goes into the second variable.❷

Step 4b – two %LABEL macros will be used to display the dates, one for each of the variables set in Step 4a. As expected, both have the same X value – the center of the bar (C). What might not be expected is

```
%label(c,2,trim(left(DateText1)),black,0,0,.7,'Helvetica',B);
%label(c,2,trim(left(DateText2)),black,0,0,.7,'Helvetica',E);
```

that they both have the same Y value as well (2). The position values of B and E place the text a half-cell above and below the X,Y position, respectively. The effect is a shown here.



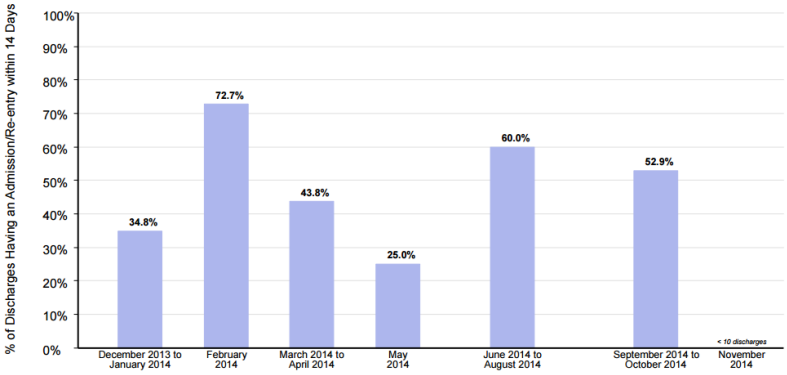
Step 5 – the last step in creating the dataset is adding the axis lines. This is done when we're done processing the data (an END=DONE option is on the SET statement). ❶ A %LINE macro is called for the

```
if done then ❶
  do;
    %line(10,5,10,100,black,1,2); ❷
    %line(10,5,90,5,black,1,2); ❸
  end;
```

vertical axis ❷ and another for the horizontal axis. ❸ There is a reason for doing this at the end rather than in the _N_ eq 1 block of code, at least for the horizontal axis. The lower Y value of the bars has a value of 5 – the same as we're making the axis line. If the line was already there, the bottom of the bars would be over the axis line, making it appear to go back and forth between black (the line) and blue (the bars). By drawing the line after the bars are there, it gives a solid black line.

Step 6 – finally we're ready to put the instructions to use. The GANNO procedure simply runs though the observations in the Annotate dataset and executes them. In this case, we're creating a PDF document, and the graph is placed in that document.

```
proc ganno anno=FollowupChart;
run;  quit;
```



Note: most SAS/GRAPH procedures have an ANNO= option which references an Annotate dataset. In this case the annotation is added to the graphics procedure output.

A Quick Tour of the Report Writing Interface (RWI)

Finally part of the production software in SAS 9.4, the Report Writing Interface allows for all the power and flexibility of the DATA step to create a report. The Report Writing Interface is just a fancy way of saying you're using the ODSOUT object in a data step. This object allows you to layout the page, create tables, embed images, add titles and footnotes and more – all from within a data step, using whatever data step logic you need. Also, all the style capabilities of ODS are available to you so that your data step created output can have fonts, sizes, colors, backgrounds and borders to make your report look just like you want.

The RWI uses a data step object called ODSOUT. Once the object has been declared, there are "methods" (like functions) of that object that will create tables, rows, cells, text, page breaks, lines, etc. Most methods have attributes that allow us to control the appearance of the output, such as fonts, colors and borders. We'll only use a handful of methods in our report.

1. table_start() and table_end() – lets us create a table
2. row_start () and row_end() – insert rows into the table
3. format_cell () – puts columns into the rows

There are a number of other methods that won't be used here – see the documentation listed in the Reference section for details.

- Table methods (more)
  - cell_start and cell_end
  - head_start and head_end
  - body_start and body_end
  - foot_start and foot_end
- Text methods
  - format_text
  - note
- General purpose methods
  - line
  - href
  - image
- Layout methods
  - layout_gridded or layout_absolute
  - layout_end
  - region
- Page methods
  - page
  - title and footnote

The Table – DATA Step Instead of PROC REPORT

Using the same dataset that was used to create the chary, there are five steps we'll take to create our table.

Step 1 – as in our chart-creation data step, there are some thing for our table that only need to be done once and will be placed in an _N_ eq 1 block of code. ❶ The first is to DECLARE the ODSOUT object.

```
if _n_ eq 1 then ❶
  do;
    declare odsout p1();❷
```

The object can be given any valid SAS name, in this case P1. ❷ Once declared, the object can be used in subsequent code. *Note: The object here is called P1 because this is part of the first page of the real report.*

Step 2 – the following code is still part of the _N_ eq 1 block. It's very common to have the methods to start the table and create the column headers in this section of code. The TABLE_START() method call starts a table ❶ and will expect row and cell calls to follow. We then have two header rows, with a total of five columns. Let's look at the second row first. ❷ The row is started with the ROW_START() method call and it contains five cells, or columns. The five FORMAT_CELL calls place the column headers on the table.

We only have one piece of text needed in the first header row ❸ that needs to span the last three columns. We can use the COLSPAN parameter on each of the FORMAT_CELL method calls to put blank

```
  p1.table_start();❶
    p1.row_start();❸
      p1.format_cell(text: ' ',colspan:2);
      p1.format_cell(text: 'Admitted/Re-entered within 14 Days',colspan:3);
    p1.row_end();
    p1.row_start();❷
      p1.format_cell(text: 'Month');
      p1.format_cell(text: 'Discharges');
      p1.format_cell(text: 'Admitted');
      p1.format_cell(text: 'Re-entered');
      p1.format_cell(text: '%');
    p1.row_end();
end;  ❹
```

text over the first two columns and the "Admitted/Re-entered…" text over the last three columns. The END statement ❹ closes our _N_ eq 1 processing. Note that the table is not ended here – that will be done when all the data has been processed. The header rows look like this.

		Admitted/Re-entered within 14 Days		
Month	**Discharges**	**Admitted**	**Re-entered**	**%[4]**

Note: in the interest of space-savings and simplicity a number of formatting attributes are not shown in these examples. For instance, the blue background of the first header row, the bold font and the alignment of the text are all controlled with style attributes. One of those statements actually looks like this, setting the style template element to use (HeaderCells) and overriding other attributes (background color):

```
p1.format_cell(text: 'Admitted/Re-entered within 14 Days',
               style_elem: 'HeaderCells', style_attr: 'background=#ADB6ED',
               colspan:3);
```

Step 3 – now we can put the data in the table by creating a row ❶ for each observation in the datastep. The discharge date, number of discharges and number of readmissions and reentries will always be

```
p1.row_start();❶
  p1.format_cell(text: DischargeMonth); ❷
  p1.format_cell(text: Discharges);
  p1.format_cell(text: Had14Day_Adm);
  p1.format_cell(text: Had14Day_Act);
```

present in the table and the FORMAT CELL calls for these variables are on each row. ❷ As noted in Step 2 above, there are style attributes in the FORMAT_CELL call controlling the fonts, formats and alignment of the text. One of these statements actually looks like this:

May 2014	12	3	0
June 2014	3	2	0
July 2014	6	3	0
August 2014	6	4	0

```
p1.format_cell(text: Discharges, format: 'BlankComma6.',
               style_elem: Page1Cellstyle,
               style_attr: 'rightmargin=10mm');
```

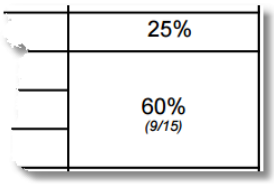
Notice that the row is still "open" (no ROW_END) – in the next steps we'll put the last data element from each row: the rate of readmission.

Step 4a – again, we're stiil in the row that was started in step 3 as we prepare to display the rate. The rate column can contain one of three values: a rate for a single month, a rate aggregated over multiple months or the "< 10 discharges" note. This step will deal with the first two of these conditions.

```
if not missing(PctFollowup) then ❶
  do;
    if NumMonths eq 1 then PercentToDisplay = PctFollowup;   ❷
    else
      do;
        CountsToDisplay = catt('(',FollowupToUse,'/',TotalDischarges,')}');❸
        PercentToDisplay = catx('~{newline}', PctFollowup,CountsToDisplay);❹
      end;
    p1.format_cell(text: PercentToDisplay,rowspan:NumMonths);❺
  end;
```

If there is a followup percentage, then there are numbers to be displayed. ❶ We'll store the text to display in the column in the variable PercentToDisplay. If there's only one month in the "span" of aggregation, this is simply the percent of followup admissions in the month (PctFollowup). ❷

If there are multiple months in the span, we'll display a couple things: the total number of followup admissions and discharges are put together in the variable CountsToDispay. ❸ This is concatenated with the PctFollowup variable, with a line break (~{newline}) between them. ❹



Finally, the FORMAT_CELL call is made for the fifth data column in the table. It will contain either of the two types of counts create above and will span rows, if necessary, using the ROW_SPAN parameter. ❺

Step 4b – if there is no percentage to display, the same note that's placed on the chart is put into the

```
else
  do;
    p1.format_cell(text: '<10 discharges', rowspan:NumMonths);❶
  end;
```

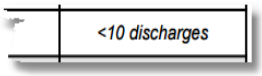


table. Instead of a variable reference in the TEXT parameter, the constant value is displayed. ❶

Step 5 – when all the observations in the dataset have been processed, the table can be closed. The

```
if done then p1.table_end();
```

TABLE_END method closes the table and, like the TABLE_START, is only done once. Below we can see the entire table.

		Admitted/Re-entered within 14 Days		
Month	**Discharges**	**Admitted**	**Re-entered**	**%[4]**
December 2013	9	3	0	35% (8/23)
January 2014	14	5	0	
February 2014	11	7	1	73%
March 2014	9	4	0	44% (7/16)
April 2014	7	3	0	
May 2014	12	3	0	25%
June 2014	3	2	0	60% (9/15)
July 2014	6	3	0	
August 2014	6	4	0	
September 2014	8	2	2	53% (9/17)
October 2014	9	5	0	
November 2014	3	1	0	*<10 discharges*

The appendices that follow contain a couple examples of the complete report – one with data for all months and the one that has been used in most of the examples.

CONCLUSION

Hopefully this simple report has piqued your interest in using DATA step-based tools for both graphical and tabular reporting. The reference sections below contain a number of resources for more information on both the Annotate Facility and the Report Writing Interface.

REFERENCES - ANNOTATE

Carpenter, Art, *Annotate: Simply the Basics*, Cary, NC: SAS Institute Inc., 1999, 110 pp.

Lund, Pete, *More to it than Meets the Eye: Creating Custom Legends that Really Tell a Story*, Proceedings of the 2010 Midwest SAS Users Group Conference, 2010.

Lund, Pete, *Make Your Tables Pop: Embedding Micrographics in PROC REPORT Output*, Proceedings of the 2009 SAS Global Forum Conference, SAS Institute Inc. (Cary, NC), 2009.

A web search of "SAS 9.4 Annotate Facility overview" will find, usually first in the list, the appropriate section in the SAS/GRAPH documentation on support.sas.com, as well as numerous papers and presentations on the topic.

REFERENCES – REPORT WRITING INTERFACE

Huff, Gina, *Simple ODS Tips to Get RWI (Really Wonderful Information) from your RWI (Report Writing Interface)*, Proceedings of the 2014 SAS Global Forum Conference, SAS Institute Inc. (Cary, NC), 2014.

Lund, Pete, *Have it Your Way: Creating Reports with the Data Step Report Writing Interface*, Proceedings of the 2014 SAS Global Forum Conference, SAS Institute Inc. (Cary, NC), 2014.

O'Connor, Daniel, *Take Home the ODS Crown Jewels: Master the New Production Features of ODS LAYOUT and Report Writing Interface Techniques*, Proceedings of the 2013 SAS Global Forum Conference, SAS Institute Inc. (Cary, NC), 2013.

A web search of "SAS 9.4 Output Delivery System Advanced Topics" will find both an HTML-based and PDF-based version of the Report Writing Interface documentation. There is a section in the ODS User's Guide on "Output Delivery System and the DATA Step" but, unfortunately it has nothing to say about the RWI.

ACKNOWLEDGEMENTS

AUTHOR CONTACT INFORMATION

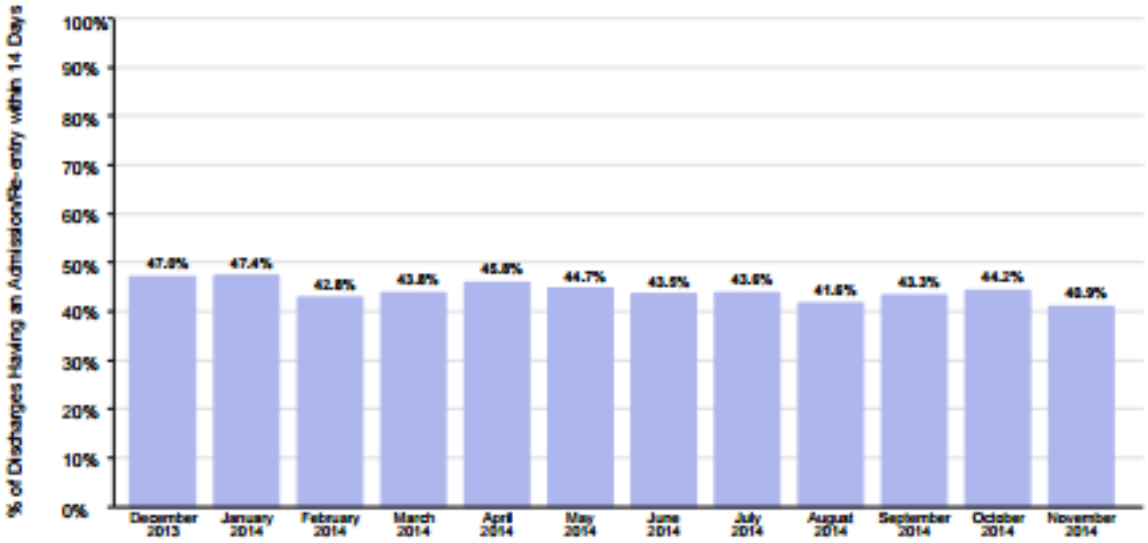
Pete Lund
Looking Glass Analytics
215 Legion Way SW
Olympia, WA 98501
(360) 528-8970
pete.lund@lgan.com

Appendix 1 – Report with Sufficient Data for Each Month

WA State DBHR Substance Abuse Treatment Reports
14-Day Followup Report

Treatment Admissions/Re-entries[1] following Adult[2] Residential Discharges[3]
between December 2013 and November 2014

Statewide - All Adult Clients



		Admitted/Re-entered within 14 Days		
Month	**Discharges**	**Admitted**	**Re-entered**	**%**
December 2013	788	298	72	47%
January 2014	851	329	74	47%
February 2014	755	275	48	43%
March 2014	796	277	72	44%
April 2014	839	320	64	46%
May 2014	883	316	79	45%
June 2014	757	256	73	43%
July 2014	832	293	70	44%
August 2014	772	253	68	42%
September 2014	803	289	59	43%
October 2014	832	300	68	44%
November 2014	758	243	67	41%
Total (12/2013-11/2014)	*9,666*	*3,449*	*814*	*44%*
Previous year (12/2012-11/2013)	*9,839*	*3,708*	*837*	*46%*

[1] Includes admissions to intensive inpatient, long-term residential, recovery house, outpatient and opiate substitution. Excludes gambling-only admissions. Clients are also counted if they re-enter an open span of outpatient treatment following a residential or detox discharge. This can happen when a client leaves outpatient treatment temporarily to be admitted to a residential modality or detox without a discharge from their outpatient span. In these cases we look not for admissions to treatment over the 14 followup period, but rather outpatient activities, at which the client was in attendance, including Individual, Group, Conjoint (with client) or Case Management.
[2] Clients who were 18 or over as of the date of discharge.
[3] "Residential" includes Intensive Inpatient, Long-term Residential and Recovery House. Excludes discharges from admissions/re-entries that were DOC and private-pay funded or gambling-only.

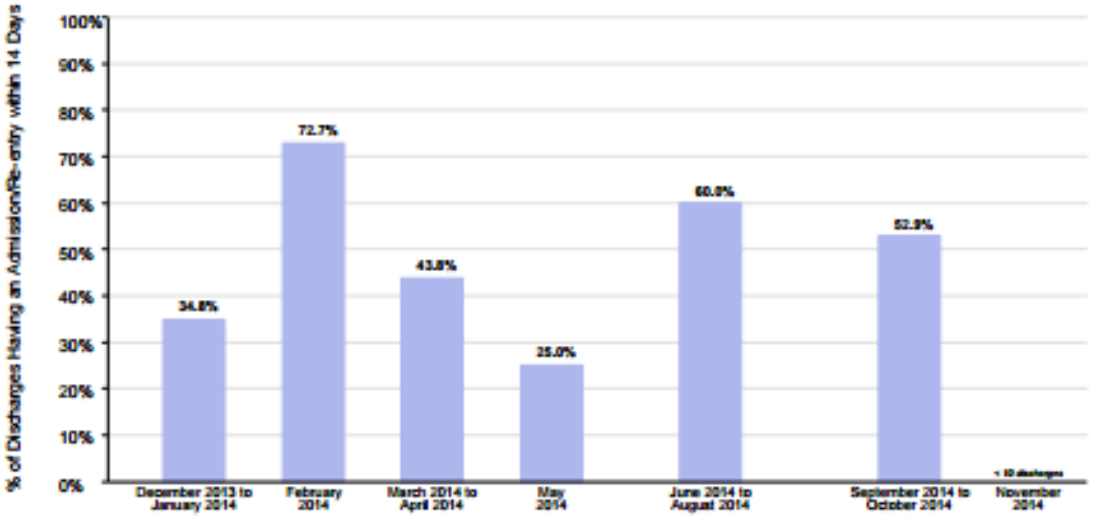
Source: System for Communicating Outcomes, Performance & Evaluation (SCOPE), WA State DSHS Div. of Behavioral Health and Recovery (DBHR) / ©Looking Glass Analytics

Appendix 2 – Report with Insufficient Data for Each Month

WA State DBHR Substance Abuse Treatment Reports
14-Day Followup Report

Treatment Admissions/Re-entries[1] following Adult[2] Residential Discharges[3]
between December 2013 and November 2014

Thurston County - All Adult Clients



Month	Discharges	Admitted/Re-entered within 14 Days: Admitted	Re-entered	%[4]
December 2013	9	3	0	35% (8/23)
January 2014	14	5	0	
February 2014	11	7	1	73%
March 2014	9	4	0	44% (7/16)
April 2014	7	3	0	
May 2014	12	3	0	25%
June 2014	3	2	0	60% (9/15)
July 2014	6	3	0	
August 2014	6	4	0	
September 2014	8	2	2	53% (9/17)
October 2014	9	5	0	
November 2014	3	1	0	*<10 discharges*
Total (12/2013-11/2014)	*97*	*42*	*3*	*46%*
Previous year (12/2012-11/2013)	*100*	*42*	*2*	*44%*

[1] Includes admissions to intensive inpatient, long-term residential, recovery house, outpatient and opiate substitution. Excludes gambling-only admissions. Clients are also counted if they re-enter an open span of outpatient treatment following a residential or detox discharge. This can happen when a client leaves outpatient treatment temporarily to be admitted to a residential modality or detox without a discharge from their outpatient span. In these cases we look not for admissions to treatment over the 14 followup period, but rather outpatient activities, at which the client was in attendance, including Individual, Group, Conjoint (with client) or Case Management.

[2] Clients who were 18 or over as of the date of discharge.

[3] "Residential" includes Intensive Inpatient, Long-term Residential and Recovery House. Excludes discharges from admissions/re-entries that were DOC and private-pay funded or gambling-only.

[4] In order to present more stable rates, a minimum of ten discharges are required to calculate percentage of follow-up admissions. Discharges are accumulated across months until the minimum is met. The table presents counts for each month, but only shows rates when the minimum is reached. The summed discharges and follow-up admissions are shown below the percentage in the aggregated cells.

Source: System for Communicating Outcomes, Performance & Evaluation (SCOPE), WA State DSHS Div. of Behavioral Health and Recovery (DBHR) / ©Looking Glass Analytics