Paper SAS 3214-2019

Identifying Seasonality and Trend with a Cycle Plot in SAS®

Lingxiao Li, SAS Institute Inc.

ABSTRACT

Can you easily tell the day of the week on which the most sales for a store item occurred, and how its sales varied over time for each day? The cycle plot (also known as month plot or seasonal subseries plot) is a popular tool for answering these types of questions. It is an effective graph for analyzing seasonal patterns and long-term trends. Initially developed by Cleveland, et al. in the 1970s, it has gained great interest in the data visualization community lately. Although the Graph Template Language (GTL) does not provide a statement for this plot, you can easily produce one in SAS® with a combination of the DATA step, the SQL and SUMMARY procedures, and ODS Graphics. This paper uses examples to show you ways to generate cycle plots. The same techniques can also be used to produce similar visuals for other types of data.

INTRODUCTION

The ODS Graphics system in SAS® can be used to create a wide-range of graphs from the simple scatter plot, to the more complicated multi-cell paneled graphs. It provides you with a rich set of tools that you can use to creatively assemble a specialized graph that helps you visualize hidden gems in the data from a unique perspective. In this paper, we will show you how to use ODS Graphics to create one of the highly customized graphs: the cycle plot.

In time series analysis, analysts are interested in finding seasonality and trend in the data. Let us look at the sale of snacks in a store over time. Figure 1 shows the time series plot of snack sales over a thirteen-week period.

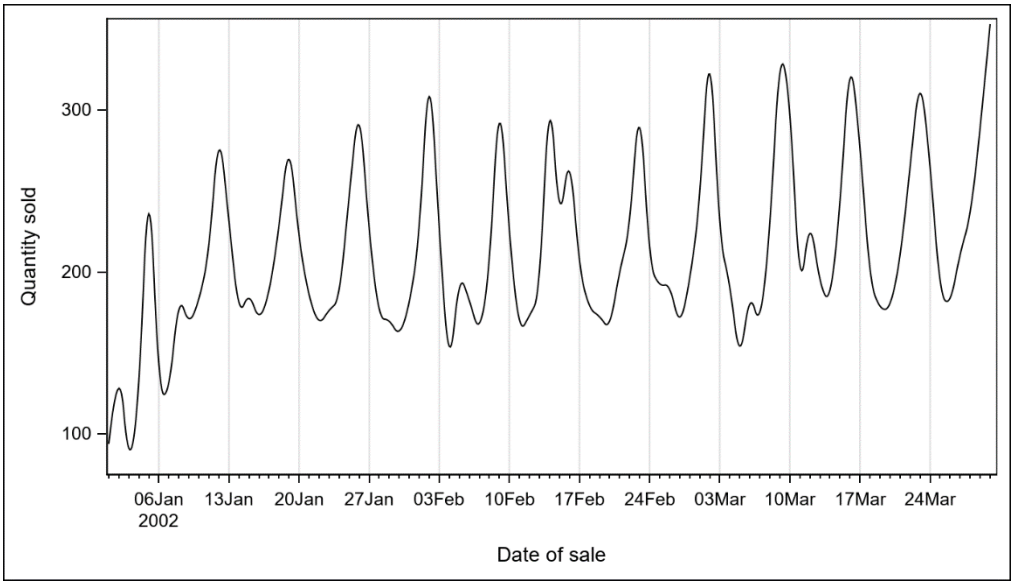

Figure 1. Time Series Plot of Snack Sales

Although the time series in Figure 1 reveals some weekly seasonality and trend in the data, it is very hard to tell which day-of-the-week has the highest average sales and the long-term trends for each of the seven days in a week.

The cycle plot was originally developed by Cleveland et al. (1978) to detect seasonality and trend in a time series. In recent years, industry experts such as Naomi Robbins (2008) and Stephen Few (2013) have further raised its awareness in the data visualization community. The cycle plot can identify the seasonality and trend in the day-of-the-week effect, the month-of-the-year effect, or the quarter-of-the-year effect. Other well-known names for the cycle plot are seasonal subseries plot and month plot.

Figure 2 shows a cycle plot of the same data used to produce Figure 1. The vertical positions of the inserted subseries plots indicate the average snack sales per day of the week. The subseries plot is made up with a spline fit of the y-variable (response variable), its confidence band, and a horizontal reference line that shows the mean y-value over the entire time. We can see that Saturday has the most sales on average. In addition, the subseries plots on most of the days show a similar upward trend in sales from week to week. That information is difficult to tell from Figure 1.

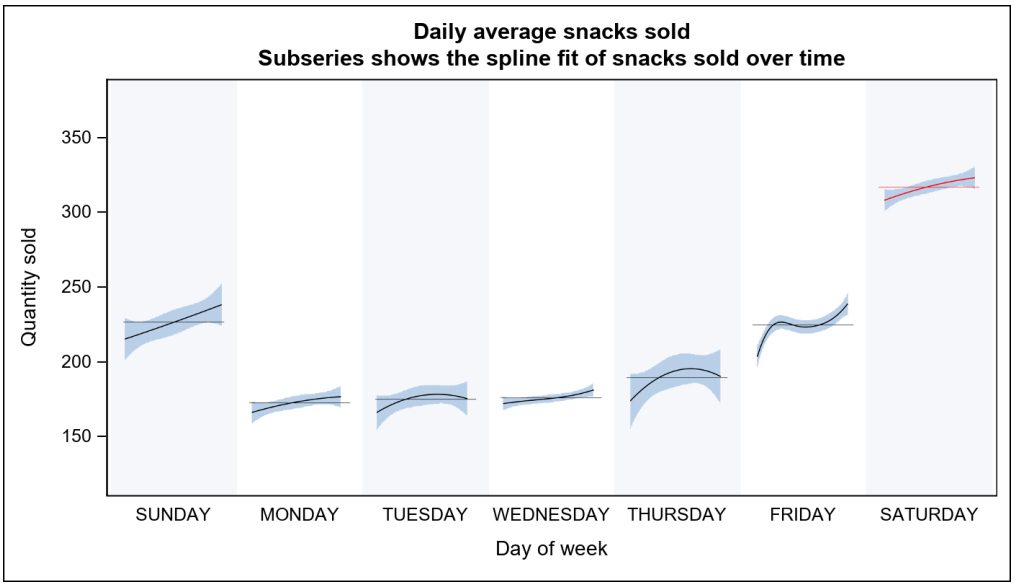

Figure 2. Cycle Plot of Snack Sales

In this paper, you will learn how to use the DATA step, the SQL procedure, the SUMMARY procedure, and ODS Graphics to produce a cycle plot. The code that generates the cycle plot can be easily extended to handle different data and different seasonal periods.

We will break down the creation of the cycle plot into three major steps:

- Data preparation.
- Generating the subseries plots as images.
- Using these images as inserted custom markers in a scatter plot.

DATA PREPARATION

The sample data that we used to produce Figure 2 is SASHELP.SNACKS as partially shown in Table 1. For better illustration of the cycle and trend, we only plot the data for the first three months in 2002. We are mainly interested in the QTYSOLD and DATE columns.

Obs	QtySold	Price	Advertised	Holiday	Date	Product
774	4.00	1.99	0	1	13FEB2004	Baked potato chips
775	8.00	1.99	0	1	14FEB2004	Baked potato chips
777	5.00	1.99	0	1	16FEB2004	Baked potato chips
778	5.00	1.99	0	1	17FEB2004	Baked potato chips
781	8.00	1.99	0	0	20FEB2004	Baked potato chips
782	10.00	1.99	0	0	21FEB2004	Baked potato chips
783	7.00	1.99	0	0	22FEB2004	Baked potato chips

Table 1. Sample of the SASHELP.SNACKS Data Set

Several data preparation steps are needed prior to making the cycle plot.

SUMMARIZING THE DATA BY DATE

We first use PROC SUMMARY to compute the sum of all snacks sold for each day in time:

```
proc summary data=sashelp.snacks
             (where=(date >= '01jan2002'd && date < '31mar2002'd)) nway;
  var qtysold;
  class date;
  output out=snacks_all sum=;
run;
```

Then we add two additional columns (DAY and DAYNAME) for the day-of-the-week (DoW):

```
data mysnacks;
  set snacks_all;
  day=weekday(date);
  dayname=put(date, downame.);
run;
```

As it will become more obvious later in the paper, the DAY column will help us generate the cycle plot with the desirable category order (from Sunday to Saturday) and the DAYNAME column will help us define the list of custom marker symbols.

COMPUTING THE EXTREME Y-VALUES

The y-variable that we choose for the cycle plot in Figure 2 is the QTYSOLD column. We want to normalize the y-values of all subseries plots with a uniform scale so that we can better compare trends across categories or seasons. We use PROC SQL to compute the minimum and maximum values for the summarized QTYSOLD column and put them into the macro variables QTYSOLD_MIN and QTYSOLD_MAX:

```
proc sql noprint;
  select min(qtysold), max(qtysold) into :qtysold_min, :qtysold_max
         from mysnacks;
  select distinct dayname, day into
         :symbols separated by ' ', :values separated by ' '
         from mysnacks order by day;
quit;
```

In the code above, we put the distinct day-of-the-week names (SUNDAY, MONDAY, and so

on) into the macro variable SYMBOLS as a space separated list. We also put the distinct day-of-the-week values (1-7) into the macro variable VALUES as another space separated list. Later, we will use the VALUES variable to subset the data for each subseries plot and the SYMBOLS variable to define the custom marker symbols.

COMPUTING THE MEAN Y-VALUE ON EACH DAY-OF-THE-WEEK

To display the inserted subseries plots vertically at the average values of the snacks sold on each day in Figure 2, we use PROC SUMMARY again to compute the mean QTYSOLD value:

```
proc summary data=mysnacks nway;
  var qtysold;
  class day;
  output out=mysnacks_mean mean=;
run;
```

Finally, we use PROC SQL one more time to find the day-of-the-week value that has the maximum average snacks sold. We put this value into the macro variable DAYMAX so that we can highlight the subseries plot with the maximum average:

```
proc sql noprint;
  select day into :daymax
         from mysnacks_mean having qtysold=max(qtysold);
quit;
```

In the following sections, we will use the two data sets MYSNACKS and MYSNACKS_MEAN as well as the five macro variables QTYSOLD_MIN, QTYSOLD_MAX, SYMBOLS, VALUES, and DAYMAX to make the cycle plot.

GENERATING THE SUBSERIES PLOTS AS IMAGES

THE SUBSERIES PLOT TEMPLATE

To derive the subseries plot for each day in Figure 2, we define the following GTL template:

```
%macro subseries_template(x=,    /* x role of the subseries */
                          y=,    /* y role of the subseries */
                          ymin=, /* min value of y */
                          ymax=, /* max value of y */
                          size=  /* image size */);
  proc template;
  define statgraph subseries;
  dynamic linecolor;
  begingraph / pad=0px border=false opaque=false
               designwidth=&size designheight=&size;
    layout overlay / walldisplay=none xaxisopts=(display=none)
                     yaxisopts=(display=none
                                linearopts=(viewmin=&ymin viewmax=&ymax));
      modelband 'spline';
      pbsplineplot x=&x y=eval(&y+0.5*(&ymin + &ymax)- mean(&y)) /
                   clm='spline' nknots=10 lineattrs=(color=linecolor);
      referenceline y=eval(0.5*(&ymin + &ymax)) /
                    lineattrs=(color=linecolor) ;
    endlayout;
  endgraph;
  end;
  run;
%mend subseries_template;
```

With the opaque=false option in the BEGINGRAPH statement, the subseries plots are generated as PNG images with a transparent background.

You can customize the subseries GTL template to fit your own needs or preference. For example, you can replace the PBSPLINEPLOT and MODELBAND statements with a simple SERIESPLOT statement so that the plot looks like most of the cycle plot examples on the web.

The EVAL expressions are used to shift the y-values in the subseries plots to line up the center of the y-axis range between QTYSOLD_MIN and QTYSOLD_MAX with the mean value. That way, the final output in Figure 2 will show the mean-value line in the subseries plot exactly at the mean-value position for each day-of-the-week.

The dynamic LINECOLOR in the template is used to highlight the subseries with the highest average.

GENERATING THE SUBSERIES PLOTS

Using the GTL template defined in the previous section, the following macro function runs the SGRENDER procedure with a WHERE clause to generate the image of the subseries plot for each day:

```
%macro subseries_plot(symbol=,      /* symbol name like DoW */
                      data=,        /* dataset */
                      whereclause=, /* where clause for this symbol */
                      linecolor=    /* line color to highlight the max */);
ods graphics / reset imagename="&symbol";
proc sgrender data=&data(where=(&whereclause)) template=subseries;
dynamic linecolor="&linecolor";
run;
%mend subseries_plot;
```

We define another macro function that generates subseries plots for all days with the WHERE expressions that are built by traversing the list in the SYMBOLS macro variable:

```
%macro all_subseries_plots(symbols=,     /* list of symbols */
                           data=,        /* dataset */
                           wherevar=,    /* variable used for subsetting */
                           wherevalues=, /* where values */
                           maxvalue=     /* value with the max mean */);
ods _all_ close;
ods listing gpath="&gpath";
%let word_cnt=%sysfunc(countw(%superq(symbols)));
  %do i = 1 %to &word_cnt;
    %let var&i=%qscan(%superq(symbols),&i,%str( ));
    %let val&i=%qscan(%superq(wherevalues),&i,%str( ));
    %subseries_plot(symbol=&&var&i, data=&data,
                    whereclause=&wherevar.=&&val&i,
                    linecolor=%if(&maxvalue = &&val&i) %then red;
                    %else black;);
  %end;
ods listing close;
%mend all_subseries_plots;
```

The following code runs the SAS macro functions and produces the subseries plots as images for all seven days of the week:

```
/* put the generated images in the WORK directory */
%let gpath= %sysfunc(getoption(WORK));
%subseries_template(x=date, y=qtysold, ymin=&qtysold_min,
                    ymax=&qtysold_max, size=200px);
%all_subseries_plots(symbols=&symbols, data=mysnacks, wherevar=day,
                     wherevalues=&values, maxvalue=&daymax);
```

The resulting seven subseries plots are shown in Figure 3.

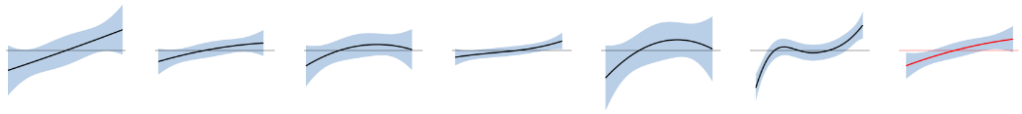

Figure 3. Subseries Plots from Sunday to Saturday

In the next section we will introduce the SYMBOLIMAGE statement in GTL and use it to define a list of custom image markers in the cycle plot of Figure 2.

SYMBOLIMAGE IN GTL

The key feature in GTL that is used to create the cycle plot in this paper is the ability to use images to define custom markers with the SYMBOLIMAGE statement. This statement was introduced in SAS® 9.4M1:

```
SymbolImage name=symbol-name image="file-name" </ options>;
```

We can use the SYMBOLIMAGE statement to define a custom symbol with a name specified by the name=symbol-name option, using an image file on the file system specified in the image="file-name" option. Then, we can use this symbol to draw image markers in the graph using its assigned name just like any other built-in symbol name in GTL such as CIRCLE.

Normally, images used for the SYMBOLIMAGE statement are taken from your smart phone camera or found on the web (Matange 2017). You can go a different route and use images from the ODS Graphics outputs, such as the ones in Figure 3, to define custom marker symbols and insert them in a scatter plot or any plot that supports markers. This is the key idea behind the paper that enables us to generate the cycle plot.

Let us introduce another macro function that traverses through a list of named symbols and their pre-defined file paths (like the SAS work directory) to define a series of SYMBOLIMAGE statements:

```
%macro symbol_images(symbols /* list of symbols */);
%let word_cnt=%sysfunc(countw(%superq(symbols)));
  %do i = 1 %to &word_cnt;
    %let var&i=%qscan(%superq(symbols),&i,%str( ));
   symbolimage name=&&&var&i image="&gpath/&&&var&i...png";
  %end;
%mend symbol_images;
```

Now we can proceed to the final step to put everything together and produce the cycle plot.

PUTTING EVERYTHING TOGETHER

If we can insert the customized image markers shown in Figure 3 at the corresponding mean-value positions of the snacks sold on each day, then that would put the finishing touches to the cycle plot shown in Figure 2.

THE SCATTER PLOT TEMPLATE

To produce the scatter plot with the inserted subseries plots in Figure 2, we come up with the following GTL template that uses marker symbols defined with the SYMBOLIMAGE statements:

```
%macro cycle_plot_template(x=,       /* category or x role such as DoW */
                           y=,       /* mean values of Y */
                           symbols=, /* list of symbols */
                           ticks=    /* list of displayed tick values */);
proc template;
define statgraph cycle_plot_graph;
dynamic title1 title2 footnote; /* for two titles and a footnote */
begingraph / subpixel=on
             attrpriority=none       /* each group gets a new symbol */
             datasymbols=(&symbols); /* override the group markers */
  entrytitle title1;
  entrytitle title2;
  entryfootnote halign=left footnote;
  %symbol_images(&symbols); /* define the custom image markers */

  layout overlay / xaxisopts=(type=discrete display=(tickvalues label)
                              discreteopts=(colorbands=odd
                              colorbandsattrs=(transparency=0.75)
                              tickdisplaylist=(&ticks)))
                   yaxisopts=(linearopts=(thresholdmin=1 thresholdmax=1
                              tickvalueformat=(extractscale=true)));
    scatterplot x=&x y=&y / group=&x usediscretesize=true
                            discretemarkersize=0.85;
  endlayout;
endgraph;
end;
run;
%mend cycle_plot_template;
```

In the template, we use the same role for X and GROUP on the scatter plot. Along with the ATTRPRIORITY=NONE option and the DATASYMBOLS option, we will have a different marker from the DATASYMBOLS list for each X value. If you want a connect line through the subseries plots, you can add a SERIESPLOT statement before the SCATTERPLOT statement and use the same X and Y variables. There is an example later in the paper that shows the extra connect line in the graph.

We also use other GTL features such as the COLORBANDS and EXTRACTSCALE options to make the graph look and read better. Finally, the DISCRETEMARKERSIZE option is put in to help with rescaling the size of the subseries plot appropriately for other seasonal periods such as the month-of-the-year effect.

Since the GTL syntax expects a space-separated list of quoted strings for the list of displayed tick values such as SUNDAY on the X axis in Figure 2, we use the following SAS function to generate the list from the macro variable SYMBOLS:

```
%let ticks=%sysfunc(catq('1a',%sysfunc(translate(%upcase(&symbols),
                    %str(,),%str( )))));
```

We run the following code to define the GTL template:

```
%cycle_plot_template(x=day, y=qtysold, symbols=&symbols, ticks=&ticks);
```

GENERATING THE FINAL CYCLE PLOT

As mentioned earlier, the use of the DAY column as the category variable, as opposed to the DAYNAME column, helps the cycle plot display the category axis with the desirable order of Sunday to Saturday. The following PROC SGRENDER code generates Figure 2 with the data set MYSNACKS_MEAN:

```
proc sgrender data=mysnacks_mean template=cycle_plot_graph;
label day='Day of week';
dynamic title1="Daily average snacks sold"
        title2="Subseries shows the spline fit of snacks sold over time";
run;
```

In the next section, we will show examples of the cycle plot with other data.

OTHER EXAMPLES

Instead of a day-of-the-week cycle plot, we can also produce the month-of-the-year cycle plot as shown in Figure 4.

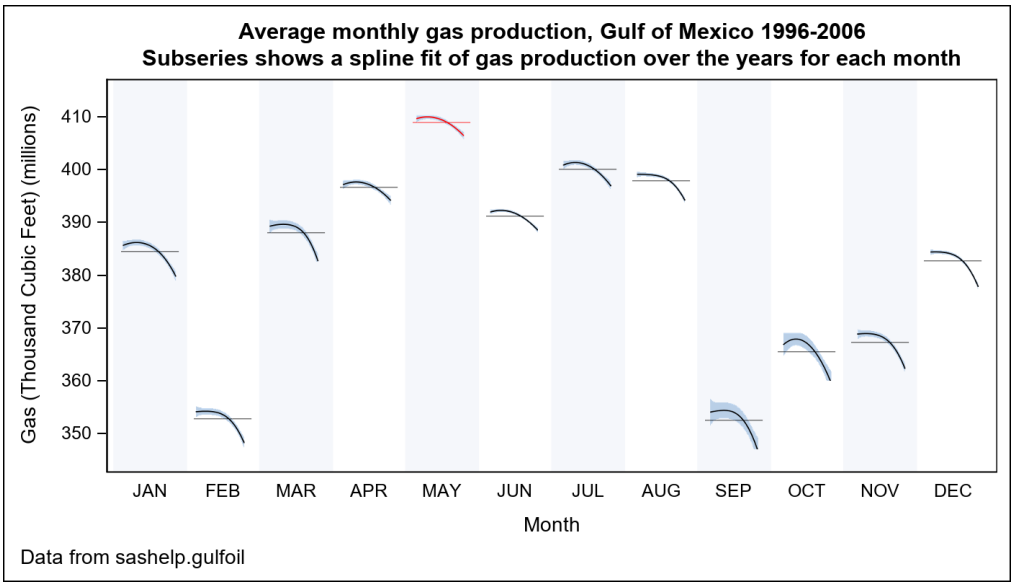

Figure 4. Cycle Plot of Gas Production

The cycle plot in Figure 4 shows that the month of May has the highest average gas production. It also shows a declining trend from year to year for all months.

To get the month-of-the-year cycle plot, we follow the steps outlined earlier in the paper for the day-of-the-week cycle plot. The one major modification to the steps is to swap out the day variables with the month variables. Due to the limited space, we will not show the code here. The complete source code for all examples in this paper will be available at https://github.com/sascommunities/sas-global-forum-2019.

Similarly, we can also produce a cycle plot with the quarter-of-the-year effect as shown in Figure 5.

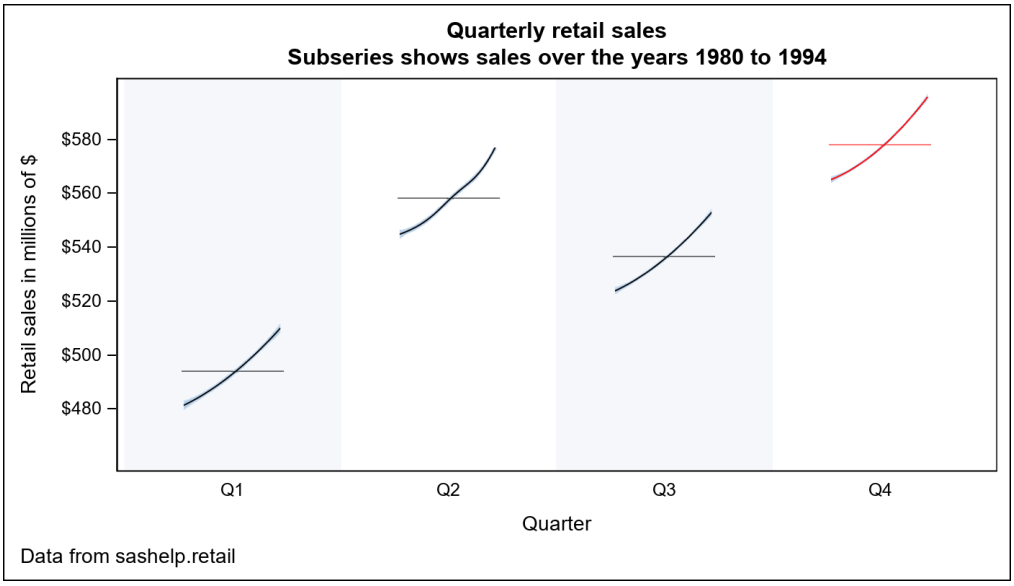

Figure 5. Cycle Plot of Retail Sales

If we go one step further beyond the traditional cycle plot, we can apply the same techniques explained in this paper and produce a custom graph that inserts the ODS Graphics outputs as markers in another plot. Figure 6 has the SYMBOLIMAGE markers on a scatter plot overlaid on top of a series plot. It shows the breakdown of the (average) flight delay to the day-of-the-week and part-of-the-day effects.

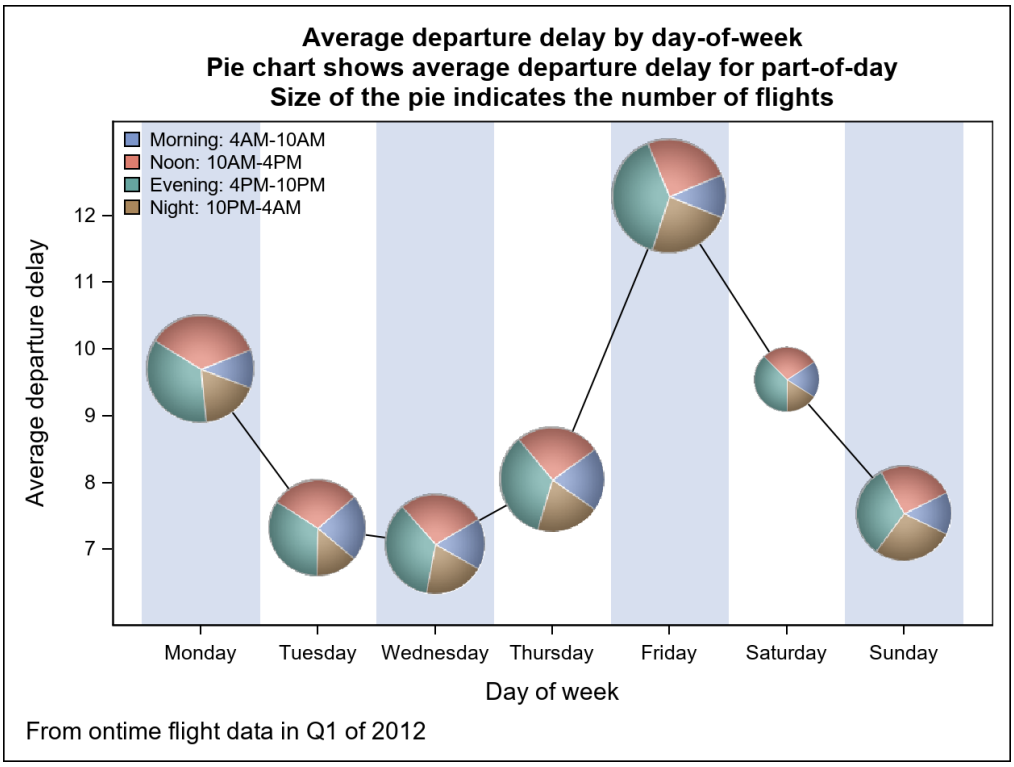

Figure 6. Inserted Pie Charts for Flight Delay

As you can see in the Figure 6, the average delay is the largest on Fridays and during the late-afternoon to evening hours in each day. That conclusion seems to match with our personal experience as airline passengers.

CONCLUSION

The introduction of the SYMBOLIMAGE statement in SAS® 9.4M1 essentially allows us to use any image output generated by ODS Graphics as a marker symbol in a subsequent graph. The techniques shown in this paper demonstrated that, with a combination of SAS language features and the ODS Graphics system, you can come up with an automatic way to produce highly customized graphs such as the cycle plot. This opens new grounds to the SAS user community of ODS Graphics and gives you the power to be more creative!

REFERENCES

Cleveland, William, Douglas Dunn, and Irma Terpenning. 1978. "The SABL Seasonal Analysis Package – Statistical and Graphical Procedures." Bell Laboratories, Murray Hill, NJ: Computing Information Service.

Few, Stephen. 2013. "A Template for Creating Cycle Plots in Excel." Visual Business Intelligence – A blog by Stephen Few. Available at https://www.perceptualedge.com/blog/?p=1780

Matange, Sanjay. 2017. "Decorative Infographs Using SAS®." *Proceedings of the SAS Global 2017 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings17/SAS0315-2017.pdf

Robbins, Naomi B. 2008. "Introduction to Cycle Plots." Perceptual Edge. Available at https://www.perceptualedge.com/articles/guests/intro_to_cycle_plots.pdf

ACKNOWLEDGMENTS

Thanks to my colleagues Prashant Hebbar, Dan Heath, Chris Noto, and Sanjay Matange, SAS Institute, for their help and comments on the paper.

CONTACT INFORMATION

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

Lingxiao Li
SAS Institute Inc.
919-531-5858
L.LI@SAS.COM