

Graphing a Progression of Time Series Plots

Nate Derby, Stakana Analytics, Seattle, WA
Laura Vo, Stakana Analytics, Seattle, WA

ABSTRACT

When graphing an ordered progression of time series plots, it can be difficult to effectively display the progression without looking disorganized and chaotic. Graphing is an essential step for exploratory data analysis and statistical modeling. This paper shows one approach to this problem, with a couple of variations, using the SAS®/GRAPH® package.

KEYWORDS: SAS, graph, PROC GPLOT, time series.

All data sets and SAS code used in this paper are downloadable from <http://nderby.org/docs/WUSS10-TSPlots.zip>.

INTRODUCTION: PROBLEMS WITH TIME SERIES PROGRESSION PLOTS

An essential part of statistical modeling is the *exploratory data analysis* (EDA) required to first look at the data and decide which models may be an appropriate fit for the data. This was initially argued by Tukey (1977) and has since become standard practice in statistical modeling. The goal is to look at the data in ways that are conducive to effectively see how the data are truly progressing through time, including large and/or irregularly spaced times.

Time series analysis is the branch of statistics that deals with data indexed by time, where the order of the observations matters. As an example of time series analysis, we look at bookings versus the number of days out for a daily flight from Greater Seattle (SEA) to San Diego (SAN). In this example the variable `daysleft` acts as a time parameter, `ddate` (departure date) as an independent variable and `bookings` as the response variable. A graph of this quantity is a *cumulative booking curve*, and it is often used in revenue management as a basis for additive and multiplicative pickup models used in statistical forecasting, as explained by Talluri and Van Ryzin (2004, pp. 469-471). However, following the principles of EDA, it can be used to suggest a variety of statistical models.

For our theoretical example, suppose we want to look at the cumulative booking curve for one flight: The one departing on Wednesday, November 3, 2010. This is shown in Figure 1(a), using the following code, within an ODS PDF statement:

```
SYMBOL1 INTERPOL=join MODE=include; ❶

AXIS1 LABEL=( height=1.6 'Days before Departure' ) ORDER=( 0 to 60 by 10 ) ...; ❷
AXIS2 LABEL=( angle=90 height=1.6 'Bookings' ) ORDER=( 0 to 190 by 10 ) ...;

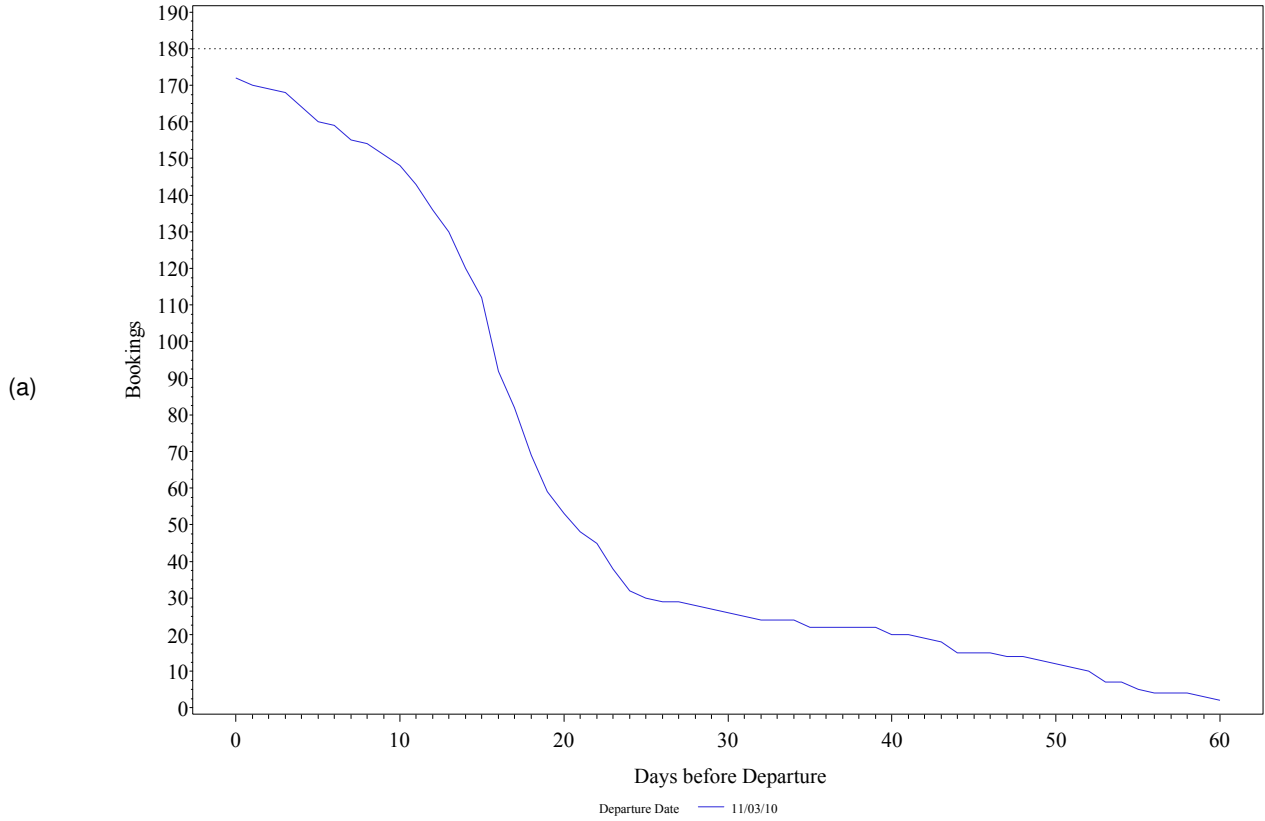
PROC GPLOT DATA=airdata2;
  PLOT bookings*daysleft = ddate / HAXIS=axis1 VAXIS=axis2 VREF=180 LVREF=33 CVREF=black; ❸
RUN;
```

Note the following:

- ❶ The `SYMBOL1` statement tells SAS to connect the points via a line and to include lines drawn toward outliers.
- ❷ The `AXIS` statements explicitly label and order the horizontal and vertical axes.
- ❸ This is our plot, using the `AXIS` and `SYMBOL` statements above. We add a dotted horizontal line at the capacity of 180 seats.

A booking curve for one specific flight is not very useful; rather, it is most effective when a series of time series plots are plotted on the same axes, which can be called a *time series progression plot* because it shows the progression of the time series plots. We can produce booking curves for all Wednesday flights (100 of them) by applying the above code to the multi-date data set `airdata` rather than to the single-data data set `airdata2` from before, except that we replace the `SYMBOL` statements with the following:.

Cumulative Bookings, 11/3/10



Cumulative Bookings, Wednesdays, 12/10/08 - 11/3/10

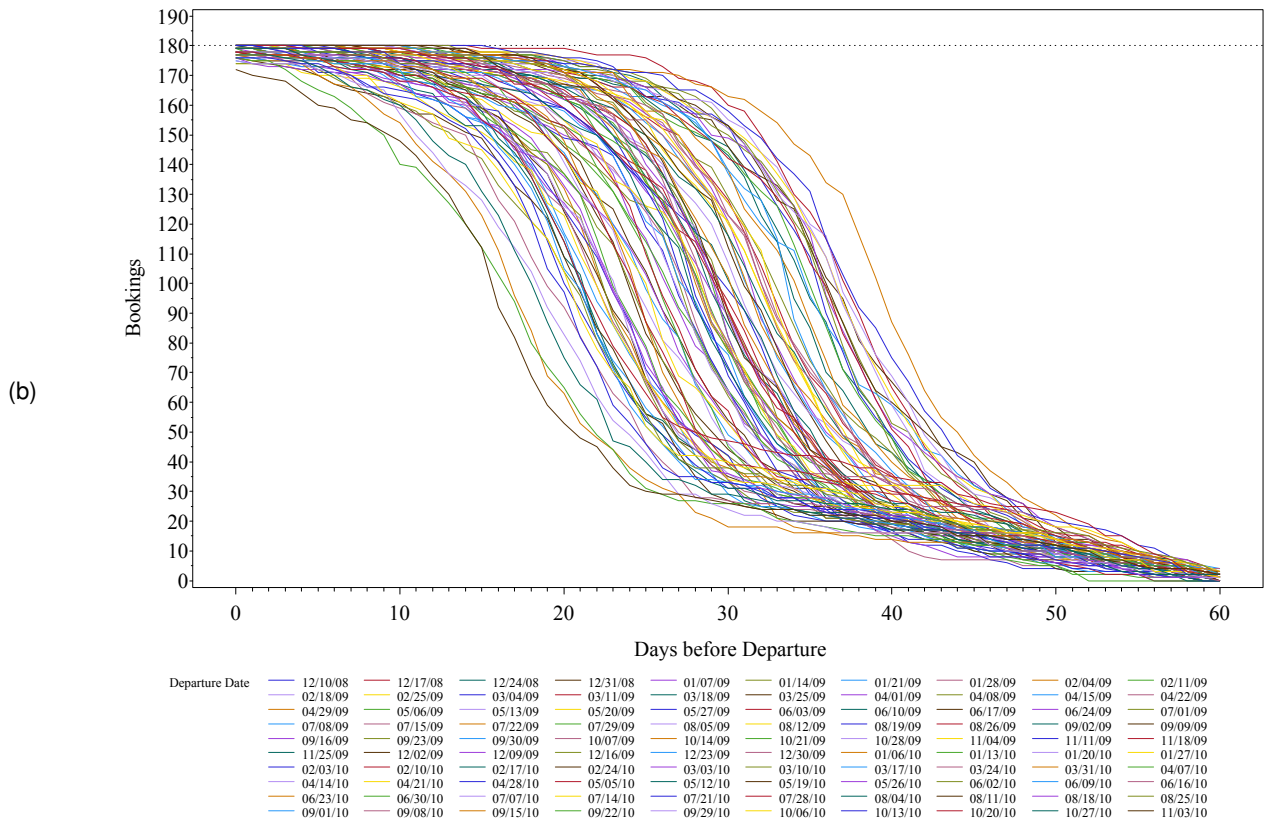


Figure 1: Cumulative bookings curve for the 11/3/10 flight (a) and all Wednesday flights over the past 100 weeks (b), using the default SAS color scheme.

```

%MACRO graphData;

...

%DO j=1 %TO &nseries;
  SYMBOL&j INTERPOL=join MODE=include;
%END;

...

%MEND graphData;

```

This uses the macro language¹ to set the remainder of the `SYMBOL` statements to have the same options as `SYMBOL1`, using the default colors that SAS assigns to each time series. `&nseries` is the number of time series that we have (100). Note that the `SYMBOL` statements are indexed by the order of the departure date (`ddate`) in the `PLOT` statement in step ③. The `SYMBOLi` statement iterates to the next value `i+1` every time the value of `ddate` changes; the actual value of `ddate` is irrelevant. Therefore, the data must be pre-sorted accordingly.

This code produces Figure 1(b). The major problem with this plot, however, is that it fails the central goal of exploratory data analysis: To effectively see what the data are telling us. Indeed, since there is no natural order of the colorings, it is very difficult to visualize the order of the progression. SAS intentionally sets these default colors to maximize contrasts between each progression, since it is designed to make it easy to differentiate the different progressions. However, we are now attempting to minimize the contrast between individual time series plots so that we can visualize general tendencies of the overall progression. Furthermore, with so many time series plots at once, with repeating or very similar colors, the legend is a distraction which serves no useful purpose.

With this distorted view of the data, a statistician might choose to fit a random intercept model, similar to that illustrated in Diggle et al. (2002, p. 136),² since it looks like the intercepts are random. But this kind of model would be inappropriate, since the intercepts actually follow an ordered progression, as we'll see in the next section.

A BETTER TIME SERIES PROGRESSION PLOT

For exploratory data analysis, the key to making a better time series progression plot is to put a visual order to the colorings. We can do this by picking a color and making the shade lighter for older time series and darker for more recent ones. Then the plot would look similar to a contour plot, except that their contours could cross. For this example, blue is used, since that color can easily be seen from both black-and-white and color printers. However, the ideas in this paper can be applied to any color.

We can use the exact same code we used before, but with different shades of blue chosen for the `COLOR` option in the `SYMBOL` statements. For coding up those shades of blue, we need to learn a little about how SAS codes colors. Watts (2001, 2002, 2003, 2004) has written a series of detailed papers about working with colors within SAS ODS. As best explained in her 2004 paper and briefly summarized here, a color is uniquely identified by the percentage of the primary additive colors red, green and blue. Each color is given a number between 0 and 255, which is the full range of an 8-bit byte (since $2^8 = 256$ different values). Many computing applications and formats such as JPG and TIFF code colors as a triplet of these values. Thus,³

Blue is (0,0,255), White is (255,255,255), Black is (0,0,0).

SAS codes these colors as hexadecimal values, so that each character represents an integer from 1 to 16 (integers 10 to 16 are represented by the letters A through F, respectively). Each hexadecimal digit is multiplied by a power of 16, with the rightmost digit being multiplied by 16^0 , the second rightmost digit being multiplied by 16^1 , and so on. For example, a value of 213 would be coded as D5, since D represents the integer 13 and

$$(13 \times 16^1) + (5 \times 16^0) = 208 + 5 = 213.$$

Fortunately, SAS has a hexadecimal format for us, `.hex2`. Therefore, within SAS, to find the hexadecimal representation of 213, we would simply write `PUT(213, .hex2)`. A full hexadecimal code for a color is `CXrrggbb`, where `rr`, `gg` and `bb` represent the two-digit hexadecimal codes for the percentages of red, green and blue, respectively. Thus,

Blue is CX0000FF, White is CXFFFFFF, Black is CX000000.

¹This macro language won't work in open code; it must be within a macro.

²The model illustrated on that page is a logistic regression model, which would be inappropriate here. However, the random intercept component is comparable.

³For a color map, see http://en.wikipedia.org/wiki/Rgb#The_24-bit_RGB_representation.

Therefore, we need to code up the intermediate shades between white (CXFFFFFF) and blue (CX0000FF). Assuming we have fewer than 256 time series sequences that are equally spaced in time, we can plot the first time series in white, the last one in blue, and the intermediate ones in the equidistantly calculated color values. Assuming we have n time series progressions, in pseudocode we have

```
FOR j = 1, 2, ..., n,
  LET k =  $\left\lfloor 255 \cdot \left( \frac{n-j}{n-1} \right) \right\rfloor$  in Hexadecimal ❶
  SYMBOLj COLOR = CXkkFF ❷
END
```

A couple notes:

- ❶ The mathematical symbol $\lfloor \cdot \rfloor$ means the *floor* of a number, which is the greatest integer less than or equal to that number. For instance, $\lfloor 5.5 \rfloor = 5$ and $\lfloor 6 \rfloor = 6$. We use this to ensure that we have an integer between 0 and 255. The $\frac{n-j}{n-1}$ term tells us what percentage of the maximal value of 255 to use; as j increases from 1 to n , this expression goes from 255 (giving us CXFFFFFF (white) in step ❷) to 0 (giving us CX0000FF (blue) in step ❷).
- ❷ Here we code the color in hexadecimal format.

This is coded into SAS as

```
%MACRO graphData;
...
%DO j=1 %TO &nseries;
  %LET k = %SYSFUNC( PUTN( %SYSFUNC( FLOOR( 255*(&nseries-&j)/(&nseries-1) ) ), hex2. ) );
  SYMBOL&j INTERPOL=join MODE=include COLOR=CX&k.&k.FF;
%END;
...
%MEND graphData;
```

Note the use of PUTN here instead of PUT, as we need to compute this in run time rather than in compile time, since it isn't inside of a DATA step. Furthermore, recall from our earlier discussion that the the SYMBOL*i* statements iterate whenever a new value of ddate is reached in the PLOT statement (step ❸ on page 1). Intuitively, in order for this to work, the data must be sorted with regards to the variable ddate.

The result is shown in Figure 2. Here we see something we were unable to see with our multicolored plot in Figure 1(b): As the departure date increases, customers wait longer before buying tickets (i.e., it shifts to the left, so that customers generally make their bookings on a smaller number of days before departure). Specifically, our random intercept model that we considered using in our previous exploratory data analysis (Figure 1(b)) would be inappropriate, since we have a *deterministic* intercept (depending on departure date) rather than a *random* intercept.

MORE THAN 256 PROGRESSIONS

The algorithm mentioned in the previous section applies to up to 256 time series progressions. If we have more than that, we can change the REPEAT= parameter in the code from 1 to a larger integer, depending on how many progressions we have. This tells SAS to repeat a certain SYMBOL statement a given number of times before moving on to the next one:⁴

```
%MACRO graphData;
...
%DO j=1 %TO &nseries;
  %LET k = %SYSFUNC( PUTN( %SYSFUNC( FLOOR( 255*(&nseries-&j)/(&nseries-1) ) ), hex2. ) );
  SYMBOL&j INTERPOL=join MODE=include REPEAT=X COLOR=CX&k.&k.FF;
%END;
...
%MEND graphData;
```

⁴This is only in effect if a COLOR option is given, which is definitely true in this case.

Cumulative Bookings, Wednesdays, 12/10/08 - 11/3/10

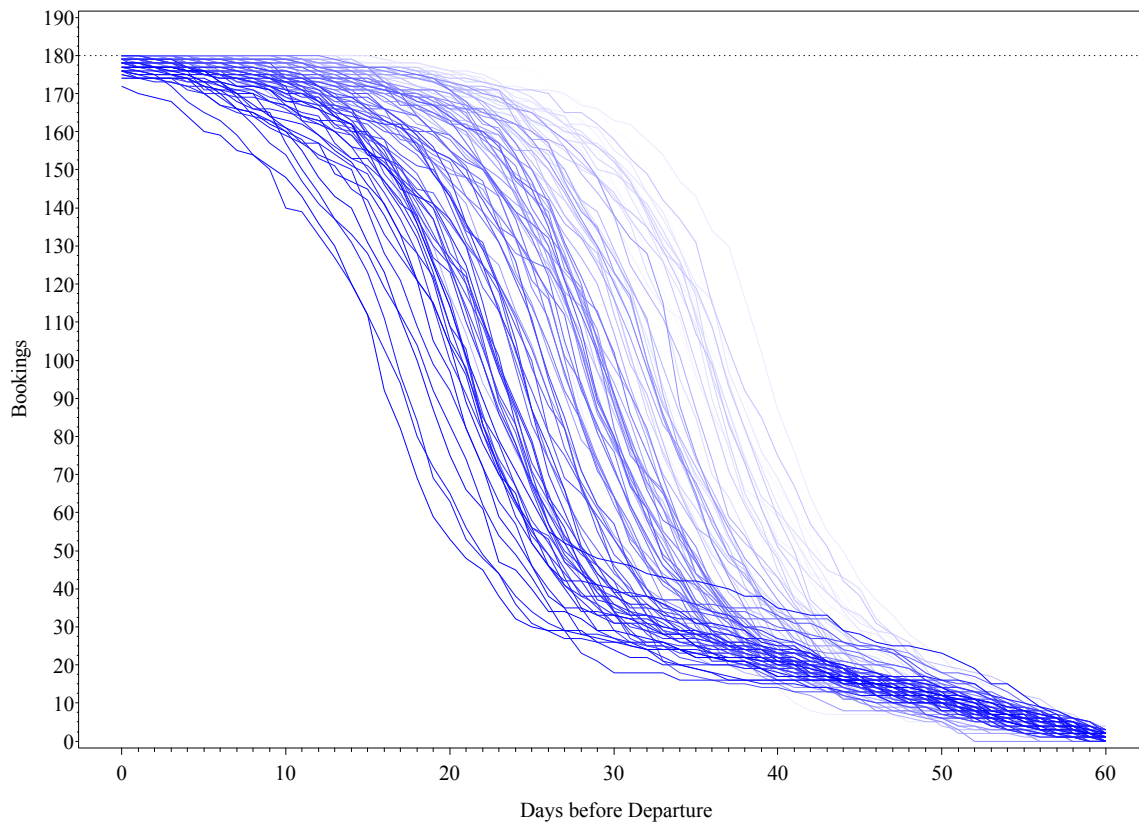


Figure 2: Cumulative bookings curves for all Wednesday flights over the past 100 weeks, using the improved color scheme.

For instance, if we have between 256 and 512 (256×2) progressions, we have $repeat=2$. If we have between 512 and 768 (256×3) progressions, we have $repeat=3$. Overall, if we have between $(256 \times (i-1))$ and $(256 \times i)$ progressions, we should have $repeat=i$.

Utilizing this style of repetition we repeat the color shading for n (number of repeats) times. In this example it would shade the first n departure dates white, then the next n departure dates the next shade and so on. While this may seem to create confusion due to repeating colors, keep in mind that this is for looking at more than 256 progressions, so that at most a given pair (or triple, or quadruple) of progressions that are the same shade is at most $\frac{1}{256}$ of the total data. And overall, this is meant to make it easier to visualize the progression of the entire data set.

PROGRESSION PLOTS WITH IRREGULARLY SPACED DATES

Frequently in data analysis there is missing data, when performing EDA this issue likely arises due to missing dates or times within the data set, leading to irregular spacing. Since the `SYMBOL` statement is determined by the value of the response variable (i.e., `bookings`) and not by the independent date variable (i.e., `ddate`), there is no way to extend this technique to irregularly spaced progressions. However, we can change our irregularly spaced progression into a regularly spaced one by filling in the gaps at regular intervals with unknown data. We can complete, or fill out, our data set by creating another data set that will fill in these gaps. Ideally this additional data set would contain both the dates for which we have data and the dates for which we are missing data that will create a regular spacing of dates, with the response variable (`bookings`) set to missing.

Cumulative Bookings, Wednesdays, 12/10/08 - 11/3/10 (missing departure dates)

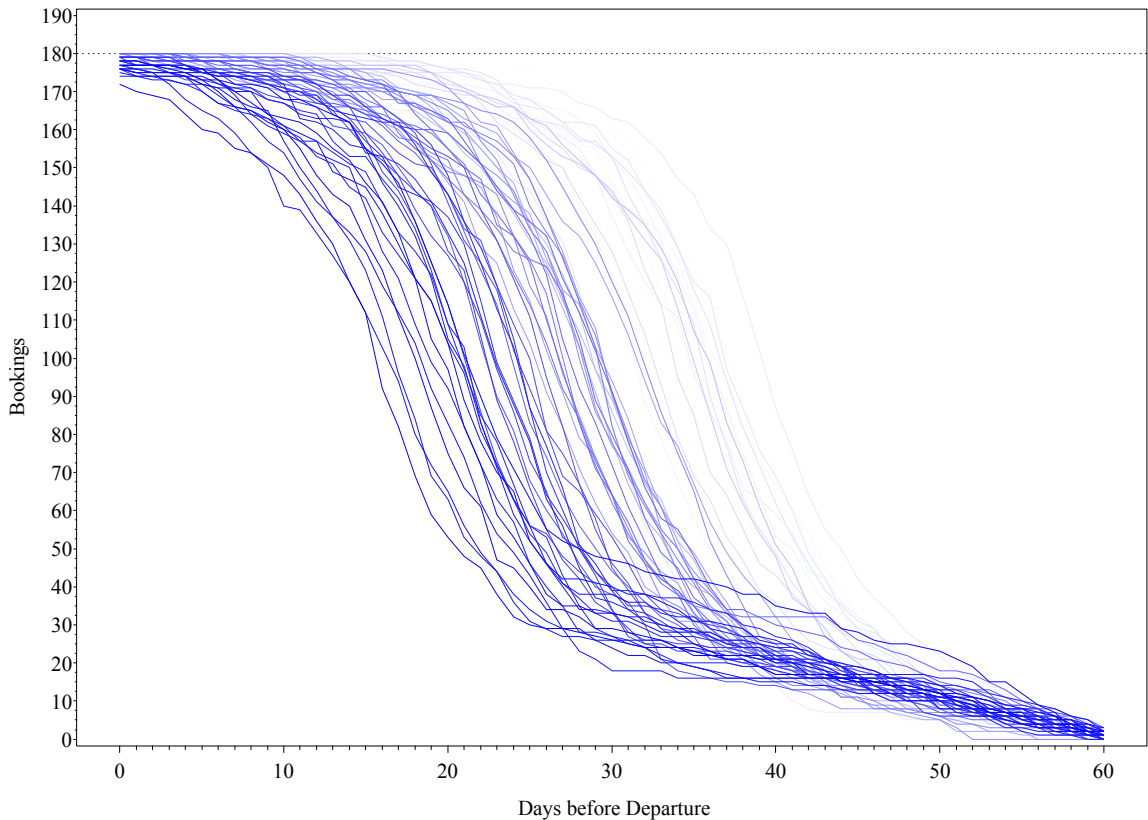


Figure 3: Cumulative bookings curves for Wednesday flights over the past 100 weeks but with some departure dates randomly removed, using the improved color scheme.

Let's suppose that in our data set (`airdata3`), we have some departure dates that are missing at random. We thus create a complete data set with all missing values for `bookings`, then `UPDATE`⁵ that data set with our data set:

```
DATA filler;
  FORMAT ddate MMDDYY8. daysleft 4. bookings 3.;
  DO i=99 TO 0 BY -1;
    DO daysleft=0 TO 60;
      ddate = MDY( 11, 3, 2010 ) - 7*i;
      bookings = .;
      OUTPUT;
    END;
  END;
RUN;

DATA airdata3;
  UPDATE filler airdata3;
  BY ddate daysleft;
RUN;
```

Graphing the data under the coloring scheme described here is shown in Figure 3. Comparing that to Figure 2, we can easily see that some progressions are missing. Nonetheless, each individual progression plot in Figure 3 uses the same shade of blue as in Figure 2 (which may be difficult to see, but it's true). As before, the earlier and later departure dates are lighter/darker, respectively, so the effect is the same as in Figure 2 in terms of exploratory data analysis. That is, we can still clearly see that the data is shifting to the left over time.

⁵Here we make use of the `UPDATE` statement which updates a data set from other data sets, thereby enabling a combined data set: In our case, one that has regularly spaced dates. Note that the data sets used need to have the same variables.

This technique can be extended to many different situations with irregularly spaced dates, as long as the data can be extended to a situation where there *are* regularly spaced dates.

PROGRESSION PLOTS ALIGNED TO THE RIGHT

There are times when we may wish to have our graphs aligned on the right – meaning that the vertical axis is labeled on the right, and the horizontal axis might be reordered as well. For example, the booking curve as described in Talluri and Van Ryzin (2004, p. 470) is aligned on the right with the days before departure in descending rather than ascending order. This can be done via the `PLOT2` statement, which can only be issued after a `PLOT` statement, which plots a quantity on the left horizontal axis. To do this, we use the following code, where we essentially plot the data twice, but with the first one (with the `PLOT` statement) completely in white, with no tick marks or labels on the left vertical axis:

```
SYMBOL1 INTERPOL=join MODE=include COLOR=white; ❶
%DO j=2 %TO %EVAL( &nseries + 1 ); ❷
  %LET k = %SYSFUNC( PUTN( %SYSFUNC( FLOOR( ( &nseries+1-&j ) * 255 / &nseries ) ), hex2. ) );
  SYMBOL&j INTERPOL=join MODE=include COLOR=CX&k.&k.FF;
%END;

AXIS1 LABEL=( height=1.6 'Days before Departure' ) ORDER=( 60 to 0 by -10 ) ...; ❸
AXIS2 LABEL=none ORDER=( 0 to 190 by 10 ) MINOR=none MAJOR=none VALUE=none; ❹
AXIS3 LABEL=( angle=90 height=1.6 'Bookings' ) ORDER=( 0 to 190 by 10 ) ...; ❺

PROC GPLOT DATA=airdata;
  PLOT bookings*daysleft = 1 / NOLEGEND HAXIS=axis1 VAXIS=axis2; ❻
  PLOT2 bookings*daysleft = ddate / VAXIS=axis3 VREF=180 LVREF=33 CVREF=black; ❼
RUN;
```

The output is in Figure 4. Note the following:

- ❶ The `SYMBOL1` states that the data in the `PLOT` statement (step ❻) will be in white, to be invisible. To avoid crossing reference lines (see below), the `INTERPOL` and `MODE` options should be the same as in the `SYMBOL` statements in step ❷.
- ❷ As before, these `SYMBOL` statements tell SAS to connect the points via a line, include lines drawn toward outliers. Note that `&j` now ranges from 2 to `&nseries + 1`, and that the ratio in the `FLOOR` function is modified to reflect that.
- ❸ The `ORDER` statement tells SAS to order the days before departure in numbers counting down to zero, to have the curves oriented up toward the right axis than toward the left one (for 0 days left).
- ❹ Here we tell SAS not to put anything on this axis, to be used in step ❸. Although we are putting nothing on this axis, we use the same `ORDER` statement as in step ❸, to keep this “invisible” graph on the same scale as our visible one, to avoid crossing any reference lines (as described below).
- ❺ We explicitly set the order of the vertical axis, to be used in step ❼.
- ❻ This is the “invisible” plot, using axes 1 and 2 defined in ❸ and ❹. Again, the only purpose of this step is to put something onto a `PLOT` statement, which is necessary in order to use the `PLOT2` statement to put our vertical axis on the right.
- ❼ This is our main plot. As before, we set a dotted horizontal line at the capacity of 180 seats.

Technically we could plot any variable in the vertical axis of the `PLOT` statement; since it’s in white and consequently invisible, why should it matter? It only matters if we have reference lines in the output, as we do here. If any of our white lines from the `PLOT` statement cross this reference line, it blocks out that part of the reference line. This can be considerable if there are either many reference lines, or plotted white lines, or both.⁶ Superimposing a graph onto itself assures that reference lines will only be blocked when they are also blocked by the main graph – in which case it doesn’t matter. There are flaws or inconveniences with other potential workarounds with this issue:

- If we wish to plot a dummy variable (made up just to use in the `PLOT` statement), we have to add that dummy variable to the code.
- If we plot a variable with any missing values, we get a note in the log; we get a warning in the log if all of the values are missing.
- If we plot a variable any missing values or values outside of the range mentioned in the `ORDER` option of the `AXIS` statement, we get a note in the log; if we do not include an `ORDER` option, we will always be within the range of the graph.

⁶For an example of this with just one line, in the last example define a new variable in the `airdata` data set, `blah`, which is always equal to 180. Then modify the `PLOT` statement in step ❻ to `PLOT blah*daysleft=1 / ... VAXIS=axis3`. The horizontal reference line at 180 will be almost completely blocked out.

Cumulative Bookings, Wednesdays, 12/10/08 - 11/3/10

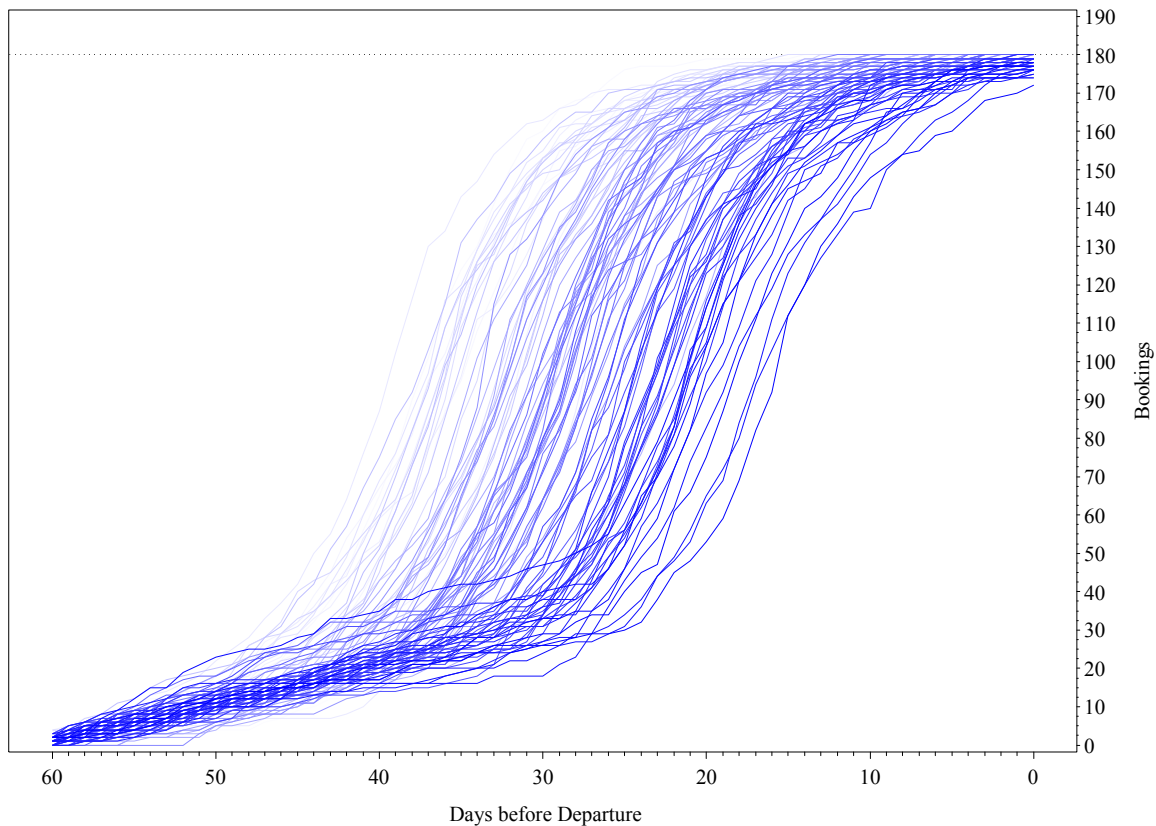


Figure 4: Cumulative bookings curves for Wednesday flights over the past 100 weeks but with a year in the middle removed, using the improved color scheme.

Many of these are just minor inconveniences if done only once, but they can become a burden if it's used for many different data sets, such as when used within a macro that's frequently used.

Lastly, note that to make sure that the main graph (from the `PLOT2` statement) is superimposed onto the white one (from the `PLOT` statement), their respective `VAXIS` designations must have identical `ORDER` and `OFFSET` options, as we see in steps 4 and 5 in our code.

CONCLUSIONS

Using SAS/GRAPH, exploratory data analysis can be made more effective for a progression of time series plots by utilizing a better choice of colors, for which there is a visual natural order to the colors. This is done by adding some simple code to the `COLOR` option to the `SYMBOL` statements. These techniques can be used for both large data sets and irregularly spaced data (when modifications have been made), enabling a more intuitive view of the progressions.

REFERENCES

- Diggle, P. J., Heagerty, P., Liang, K.-Y. and Zeger, S. L. (2002), *Analysis of Longitudinal Data*, second edn, Oxford University Press, Oxford, UK.
- Talluri, K. T. and Van Ryzin, G. J. (2004), *The Theory and Practice of Revenue Management*, Kluwer Academic Publishers, Boston.
- Tukey, J. (1977), *Exploratory Data Analysis*, Addison-Wesley, Boston.
- Watts, P. (2001), Defining colors with precision in your SAS/GRAPH application, *Proceedings of the Fourteenth Northeast SAS*

Users Group Conference.

<http://www.nesug.org/proceedings/nesug01/cc/cc4023.pdf>

Watts, P. (2002), Using ODS and the macro facility to construct color charts and scales for SAS software applications, *Proceedings of the Twenty-Seventh SAS Users Group International Conference*, paper 125-27.

<http://www2.sas.com/proceedings/sugi27/p125-27.pdf>

Watts, P. (2003), Working with RGB and HLS color coding systems in SAS software, *Proceedings of the Twenty-Eighth SAS Users Group International Conference*, paper 136-28.

<http://www2.sas.com/proceedings/sugi28/136-28.pdf>

Watts, P. (2004), Advanced programming techniques for working with color in SAS software, *Proceedings of the Twenty-Ninth SAS Users Group International Conference*, paper 091-29.

<http://www2.sas.com/proceedings/sugi29/091-29.pdf>

ACKNOWLEDGMENTS

We thank our clients for working with Nate on this topic. We also thank Perry Watts for her insights on color, via her papers referenced here as well as via personal correspondence. We also thank our respective partners/spouses Charles and Hoa for their patience and support.

CONTACT INFORMATION

Comments and questions are valued and encouraged. Contact one of the authors:

Nate Derby
Stakana Analytics
815 First Ave., Suite 287
Seattle, WA 98104-1404
206-973-2403
nderby@stakana.com
<http://nderby.org>

Laura Vo
Stakana Analytics
815 First Ave., Suite 287
Seattle, WA 98104-1404
206-973-2403
lvo@stakana.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.