

Getting the Job Done With Macros and Annotation
Randy C. Finch, Tennessee Valley Authority, Muscle Shoals, AL 35662-1010

Introduction

How often have you been given an assignment, and as soon as it was completed, more was needed? And after that, even more was needed, and so on ad managerium? I was recently given such an assignment. Luckily, this work was a fun learning experience rather than a how-do-I-get-out-of-this experience.

Background

My work involves processing data from atmospheric computer models and presenting it in an easily visualized way. Since there are tons of data spewing forth from the models, the only way to assimilate it (Borg terminology for making something an integral part of yourself) is to condense it into summary tables, graphs, or plots. The data typically consist of the concentrations of various chemical species in the atmosphere over a geographical region. This region is divided into grid cells that have a width (East-West), height (North-South), and depth (Up-Down). Each grid cell has a concentration, typically in parts-per-billion (PPB), for each chemical species.

I was given ASCII files containing the daily maximum ozone concentrations for five overlapping regions in the northeast United States known collectively as the Northeast Corridor (NEC). Figure 1 shows these regions. The data spanned an eight day period, July 10-17, 1995. There were six modeling runs, one being the base case and the other five being sensitivity runs, each representing progressively wider spread emission controls. There was a separate file for each day in each region for each run for a total of 240 files. Each file looked like Listing 1. Ultimately, all of this data needed to be merged and processed to create a box and whisker plot as shown in Figure 2. This plot shows how large an effect the differing emission controls had over the entire NEC for all days at different levels of base concentration.

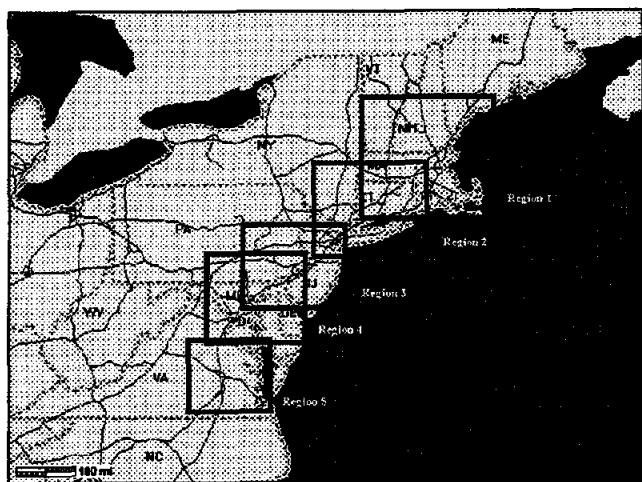


Figure 1 - The Northeast Corridor

To create Figure 2, all of the base case ozone concentrations for each day in each cell of the NEC were categorized into ranges of 20 (0-20, 20-40, etc.). The concentration differences between each sensitivity run and the base case were calculated. The differences for each run and each range of base concentrations were summarized with a box and whisker plot showing the median, the 25th and 75th percentile, and the maximum and minimum. Since a difference of zero represents no change in ozone concentration between a sensitivity run and the base case, a Y reference line was drawn at zero. Also, to aid in visually categorizing the base case concentrations, X reference lines were drawn at 0, 20, 40, etc.

[illegible]

Listing 1 - ASCII File Format

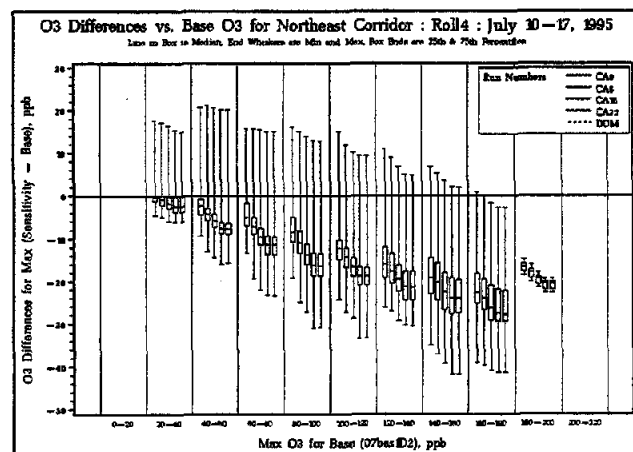


Figure 2 - The Original Box and Whisker Plot

Getting Looped

Not wanting to write 240 data steps, a macro with nested %do loops was in order. Fortunately, the files used a strict naming convention that included the region and the date. This made it easy to use macro variables. The SAS program to create a data set with all the information necessary for creating Figure 2 is shown in Listing 2. The code has a MACROS section and a PROGRAM section.

The nested %do loops appear in the %loopregs macro. The outer loop is for the date, and the inner loop is for the region. Since the date and region both consist of consecutive numbers, only the beginning and ending dates and regions are needed. These, along with the path/filename and a short symbolic name representing the run, are passed to the macro via the %loopregs calls in the PROGRAM section. There is a separate call for each model run.

Each path/filename contains two macro variables: ® and &date. Since these macro variables cannot be resolved at this point, the path/filename is enclosed in single quotes. This prevents the substitution of the values of the macro variables for the macro variables themselves.

The short symbolic names for each run are BAS, CA0, CA8, CA16, CA32, and DOM. BAS represents the base case, while the others represent the sensitivity runs. DOM represents a run where emission controls were applied domain-wide.

Listing 2 - Program to Create Data Set

This creates a new data set with all the regional data for a given run and date. The `®sets` macro variable is a space delimited list of the five regional files created by the `%readdata` macro. This list is created in the `%let` statement following the `%readdata` call.

Remember that the five regions are *overlapping*, meaning there is some duplicate data between the regional files. These duplicates have to be eliminated so no grid cells are used more than once. This is accomplished with the SORT procedure using the NODUPKEY option, which eliminates any observations that duplicate previous values of the variables on the BY line. To remove any observations with duplicate dates and (X,Y) grid positions, the data set is sorted by DATE, Y, and X.

Now, all daily data sets for a given run need to be combined into an overall run data set as follows:

```
DATA &dsname;  
  SET &datssets;  
RUN;
```

A data set with a name consisting of only the symbolic name is created. It contains the data for all days and regions for that run. The `&datasets` macro variable is a space delimited list of the eight daily data sets created earlier. This list is created in the `%let` statement at the end of the outer loop. After all six calls to the `%looppregs` macro, there exists six data sets named BAS, CA0, CA8, CA16, CA32, and DOM. These have to be merged, but let's first look at the `%readdata` macro.

Reading Data

The %readdata macro is called 240 times, once for each ASCII file. Four pieces of information are passed to the macro: the path/filename, the name of the data set to create, the name of the variable to contain the ozone concentrations, and the region number. A different name is needed for each run's ozone concentrations because the data will eventually be merged, and the ozone concentrations for each run need to be represented by different variable names. The name given is O3&dsname, where &dsname is the symbolic name for the run. For example, the variable name for the ozone concentrations in the DOM run is O3DOM.

The `FILENAME` statement assigns `FILEIN` a value of “&fnin”, where &fnin is the macro variable containing the path/filename of the ASCII file to be read. Here, the path/filename appears in double quotes. This causes the macro variables to be replaced with their values. Therefore, ® is substituted with the current region, and &date is substituted with the current date.

The data in the ASCII files do not appear until line 10; the first nine lines contain header information. The `FIRSTOBS=10` option on the `INFILE` statement informs SAS of this. Only the (X,Y) grid position, date, and ozone concentration are read with the `INPUT` statement. The `@` operator is used to specify the column in which each piece of data begins.

The ASCII files for each region contain (X,Y) grid positions starting with (1,1). This is not good because cell (1,1) in one region is not in the same geographical location as cell (1,1) in the other regions. This would not be a problem if the regions did not overlap. However, in order to eliminate duplicate cells using the NODUPKEY option in the SORT procedure, the grid cells for each region need a common reference point. This is the purpose of the XOFFST and YOFFST arrays. They provide offset values for each region that give all cells in all regions a common reference point.

I Love It When a Data Set Comes Together

After all of the calls to %loopregs, the six overall run data sets are sorted by DATE, Y, and X. This makes it possible to merge them into one data set named ALLNEC in a data step.

To understand the purpose of the assigned variables in the data step, look again at Figure 2. The Y-axis represents the difference in ozone concentrations of the sensitivity runs and the base case. This explains the last five assignments in the data step. But what the heck is this?

[illegible]

Upon calling %loopregs, the looping can begin. Before each loop, a macro variable is assigned a null value. The names of the data sets created within the loop are appended to this macro variable. The inner loop calls the %readdata macro which contains the data step that reads the ASCII files. It creates a data set for a given run on a given date for each region. The resulting data sets have names consisting of the symbolic name, the date, and the region number. For example, the data set for the CA0 run on the 10th in region 3 has a name of CA0103. The five regional files are combined together as follows:

```
DATA &dsname&date;  
  SET &regsets;  
RUN;
```

$$\text{O3BASC} = 20 * \text{INT}(\text{O3BAS} / 20) + 4$$

And why are those other variables that are offset from O3BASC by increments of three being created? The formula $20 * INT(O3BAS/20)$ is used to categorize the base case data. If $0 \leq O3BAS < 20$, then O3BASC is zero; if $20 \leq O3BAS < 40$, then O3BASC is 20; and so on. Four is added to this value to prevent the box and whisker plot from falling on top of the X reference lines. The GLOT procedure does not provide a way to offset X-axis values for multiple overlaid plots. Thus, if the differences for each sensitivity run are plotted against a common base case value (O3BASC), the box and whisker symbols would overlap. To overcome this problem, separate X-axis values (O3BASC, O3BASP1 - O3BASP4) that are offset from each other are created for plotting against the five sensitivity run differences.

All of the data needed to create Figure 2 are now in place. However, the data set does need to be sorted by the X-axis values so the GLOT procedure will work properly. I chose to sort by O3BASC, but any of the other X-axis values (O3BASP1 - O3BASP4) could have been used.

The Plot Thickens

The program to create Figure 2 is shown in Listing 3. The code is fairly straightforward, but there are a few points that need to be made. The `GOPTIONS` statement uses a handy little value named `LFACTOR`. It causes the width of all lines in a plot to be increased by the specified factor, in this case three. Setting `LFACTOR` to one while testing the program allows the plot to look good on a computer monitor, but the lines are sometimes too thin to look good when printed. When it is time for printing, `LFACTOR` can be set to a larger value and the plot regenerated. Now, the plot looks weird on screen, but the printout looks great.

A SYMBOL statement is used for each sensitivity run. The main thing to point out here is the I=BOXT00 option. BOX specifies that a box and whisker symbol should be generated. The T specifies that the tops and bottoms of the whiskers should be drawn. The 00 specifies that the ends of the whiskers should be the maximum and minimum values rather than a percentile.

The **AXIS1** statement contains some interesting options. The X-axis values are not the typical numbers-under-the-tick-marks variety. There is actually text specifying a range of values. To make these values appear, the **ORDER** of the values to use for the X-axis (-10 to 230 by 20) is specified, but no tick marks are needed as indicated by the **MAJOR** and **MINOR** options. The **VALUE** option is used to center range labels under the **ORDER** values. The first and last range labels are null, preventing labels from being printed at -10 and 230.

The order of the **LEGEND** labels has to be the same as the order of the plots in the **GLOT** procedure so the colors will match properly.

Now for some action using the Gplot procedure! Five plots are specified: each sensitivity run difference versus its matching offset base case value. The OVERLAY option is used so all five plots will appear on one graph. FRAME causes a box to be drawn around the entire plot area. The VREF=0 option tells SAS to create the Y reference line. The HREF option specifies where the X reference lines should be drawn. Since no annotate data set is being used at this time, the ANNO=ANNO option is commented. Later, as more features are added to the plot via annotation, this line will be un-commented.

That's it! When the GPLOT procedure completes execution, Figure 2 is produced.

Listing 3 - Program to Create Figure 2

Addition #1 - Number of Observations For Data Groups

After creating the plot in Figure 2, an addition was requested. It was to print the number of observations used to create each group of box and whisker symbols (where the X-axis values were 0-20, 20-40, etc.). To accomplish this task, the code in Listing 4 was added to the code in Listing 3. The resulting plot is shown in Figure 3.

[illegible]

A copy of the ALLNEC data set is created. A new variable, NUM, with a value of one for each observation is added to the data set. The MEANS procedure is used to sum up the NUM variable for each value of O3BASC, which is the base value used for categorizing the data. The MEANS procedure creates a data set named STATSUM. The NUMSUM variable in this data set contains the number of observations for each group of data.

An annotated data set named ANNO1 is created. It contains the standard annotated variable names. The LABEL function is to be used to print PINK text using the ZAPF-style font with a size of 3. A POSITION of 2 indicates that the text should be centered about the designated X value and set above the designated Y value. An HSYS value of 3 indicates that the SIZE value is in units of percent of plot area. XSYS and YSYS values of 2 indicate that the X and Y values are in units matching that of the plot itself.

```

Add these lines to the beginning of the program section:

*****
DATA NAME,
  GET NUMBER(10),
  SUM = 1,
  MIN,
*****
*****
PROC NAME= SUM GROUPS,
  TWO WAY,
  BY COLOR,
  CATCH GET-STATSUM CUM-MINIMUM,
  MIN,
*****
*****
DATA LABEL,
  LENGTH TEXT = 4 COLOR POSITION STYLE = 0,
  GETACH POSITION LABEL COLOR 'RED' POSITION 'R',
  MOVE '1' TO '1' MOVE '2' TO '2' MOVE '3' TO '3',
  GET CATCH,
  TEXT = MINIMUM,
  X = COLOR - 1,
  Y = 10,
  END,
*****
*****
DATA NAME,
  GET CATCH,
  END,
*****
*****
Add these lines to the titles section:

TITLE 1= 'Sum of 1'
          Numbers at Bottom are Total Number of Values for Each Group'

Un-comment this line in PROC SELECT:

MIN = MIN*
```

Listing 4 - Code to Add to Listing 3 to Print Number of Observations (Figure 3)

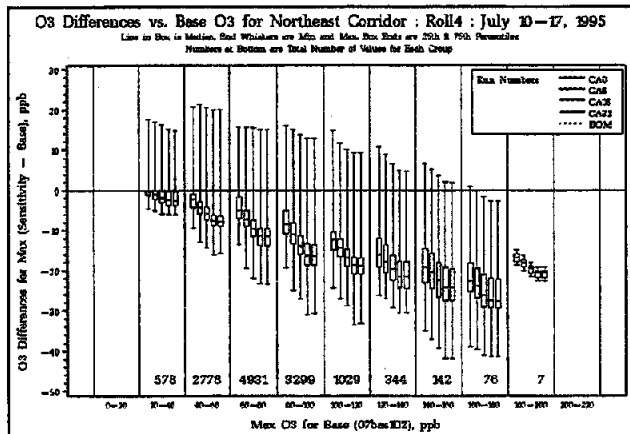


Figure 3 - Plot Showing Number of Observations

TEXT is assigned the value of NUMSUM. X is the center of the data group ranges (10, 30, 50, etc.), which is O3BASC plus six. To place the text at the bottom of the plot area, Y is set equal to the smallest value on the Y axis, -50.

Finally, the ANNO1 data set is copied to the ANNO data set. What sense does this make? Well, this allows for additional annotate data sets to be created later and appended to ANNO without messing up ANNO1. The ANNO=ANNO statement in the GPLOT procedure is un-commented so the annotation will be applied to the plot.

Addition #2 - High and Low Tick Marks

The next addition was to add tick marks to the top and bottom of each box and whisker symbol. These tick marks were to represent the highest five Y

values, other than the maximum, and the lowest five Y values, other than the minimum. The final result was to look like Figure 4. To create these tick marks, the code in Listing 5 was combined with the code in Listings 3 and 4.

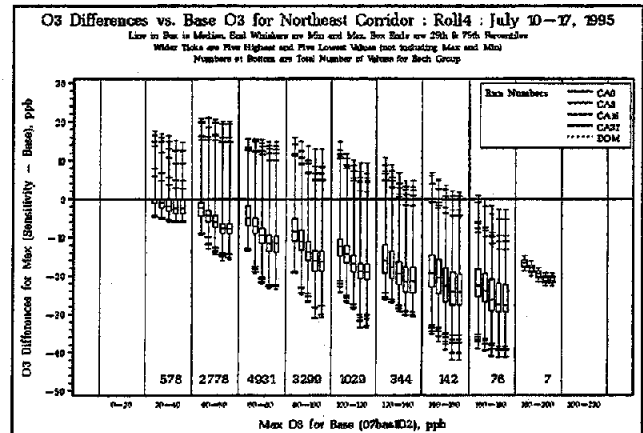


Figure 4 - Plot with High and Low Tick Marks

Two macros, %hilo5 and %doanno2, are used. The %hilo5 macro is called once for each set of run differences. The macro loops through the base X values, each time extracting from the ALLNEC data set only the data for the current X value. Once extracted, the data is sorted (in ascending order, the default) by the designated Y variable, and %doanno2 is called. This macro creates an annotate data set for drawing small tick marks onto the box and whisker symbols. It only does this for observations 2 through 6, which are the lowest five values, other than the minimum. Once %doanno2 completes execution, the %hilo5 macro continues. The same data is sorted in descending order, and %doanno2 is called once again. This time, observations 2 through 6 represent the highest five values, other than the maximum.

The %doanno2 macro does its thing with functions POINT and DRAW. The POINT function moves the annotate cursor position to (X,Y) on the plot. In this case, X is equated to the X value of the box and whisker symbol *minus* 1.2, or slightly to the left of the stem. Y is equated to the value of Y of the current observation in the sorted data set. The DRAW function draws a line from the current annotate cursor position (which was just set with the POINT function) to a different (X,Y) position. Here, X is equated to the X value of the box and whisker symbol *plus* 1.2, or slightly to the right of the stem. Y is again equated to the value of Y of the current observation. These two functions result in a small horizontal tick mark being drawn across the stem of the box. Each time through the %doanno2 macro, a two-observation annotate data set, ANNO2, is created and appended to the master ANNO data set. After all iterations of %doanno2 are complete, the ANNO data set will have a pair of POINT/DRAW observations for all the tick marks on all the box and whisker symbols.

Addition #3 - Number of Observations Greater Than Zero

The third and final addition was to print the number of observations where the Y value was greater than zero above each box and whisker symbol. Adding the code shown in Listing 6 took care of this request. The resulting plot is shown in Figure 5.

A macro named %numgt0 is called five times, once for each sensitivity run. This macro extracts from the ALLNEC data set all observations where the specified Y value is greater than zero. For each of these observations a new variable named NUM is set equal to one. The MEANS procedure is used to sum up NUM for each group of X values. Also, it determines the maximum value of Y for each group.

[illegible]

Listing 5 - Code to Add to Listings 3 & 4 to Draw High and Low Tick Marks (Figure 4)

An annotated data set named ANNO3 is created with a bunch of LABEL functions, similar to the ANNO1 data set created earlier. However, in this case the text size is smaller (SIZE=1.4), left aligned (POSITION=6), and rotated 90 degrees counterclockwise (ANGLE=90). The X position for the label is the X value for the box and whisker symbol. The Y position is the maximum value of Y determined in the MEANS procedure. Before the %numgt0 macro ends, the observations in ANNO3 are appended to the master ANNO data set.

Conclusions

Macros and annotation are very powerful tools in SAS. The use of macros can greatly reduce the amount of code that needs to be written, while annotation allows you to add just about anything to a plot. When properly structured, it is very easy to add macros and annotation to an existing program.

Listing 6 - Code to Add to Listings 3, 4, & 5 to Print Number of Observations Greater Than Zero (Figure 5)

[illegible]

References

- SAS Language: Reference, Version 6, First Edition*; SAS Institute, Inc.; 1990.
- SAS Procedures Guide, Version 6, Third Edition*; SAS Institute, Inc.; 1990.
- SAS Guide to Macro Processing, Version 6, Second Edition*; SAS Institute, Inc.; 1990.
- SAS/Graph Software: Usage, Version 6, First Edition*; SAS Institute, Inc.; 1990.
- SAS/Graph Software: Reference, Version 6, First Edition, Volume 1*; SAS Institute, Inc.; 1990.
- SAS/Graph Software: Reference, Version 6, First Edition, Volume 2*; SAS Institute, Inc.; 1990.

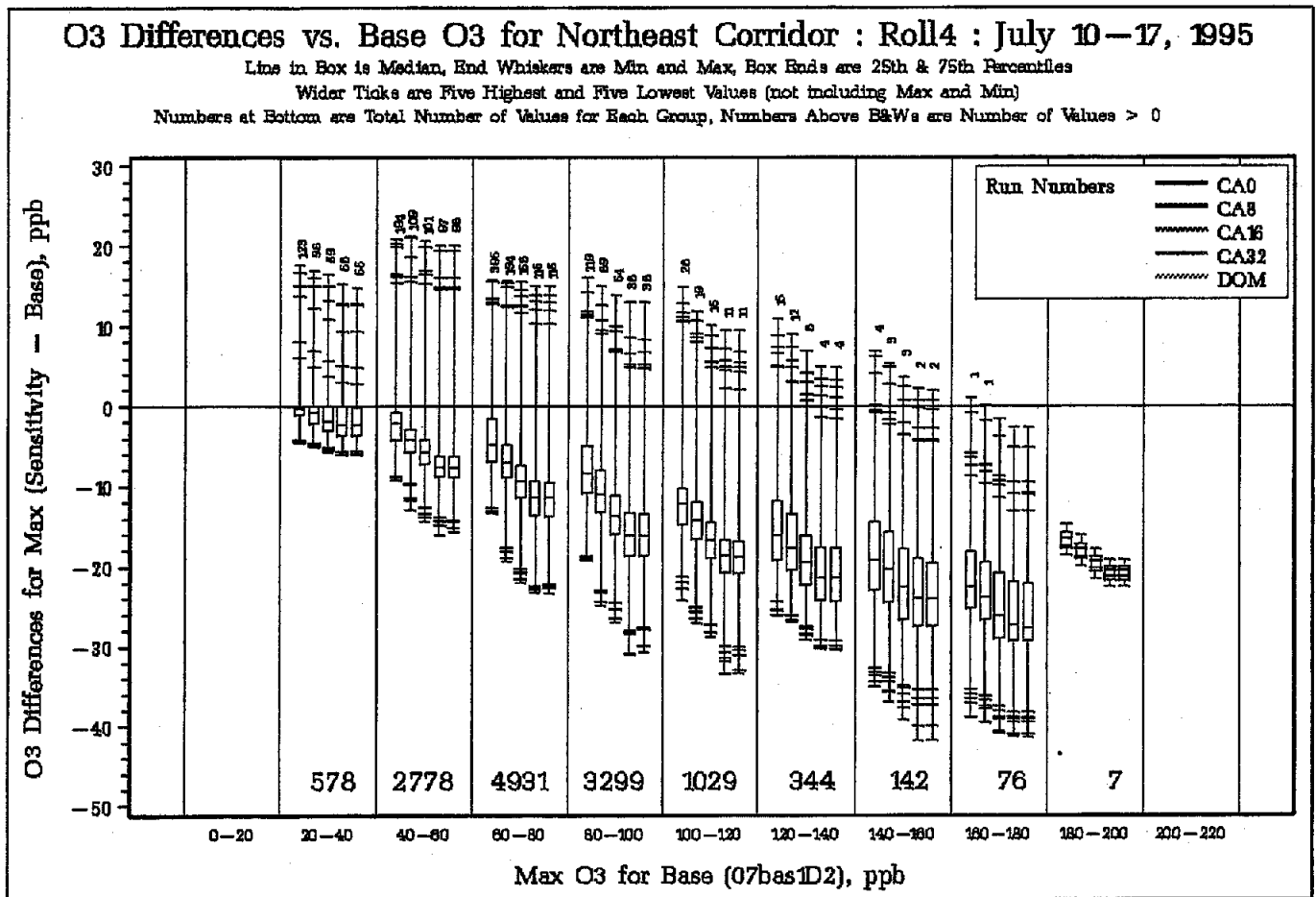


Figure 5 - Final Plot with Number of Observations Greater Than Zero