Volume 3 Issue 2

Fall 2001

SOUTHEAST SAS USERS GROUP

The SESUG Informant

From the President

F. Joseph Kelley

Unlike last summer, we seem to be having a good deal of rain. So it is hot and humid instead of hot and dry. Like many of you, I am looking forward to SSU 2001 – The Southern SAS Users Conference. I can assure you the conference co-chairs, Deborah Buck and S. David Riba, have set up some very interesting sections and attracted some outstanding speakers from across the country. Yours truly has had the great pleasure and privilege of working on this conference and working with some of the good folks from both the South Central SAS Users Group (SCSUG) and our own SESUG and it has been a good deal of fun. In my previous column, I mentioned that I would have more to say about SSU in this column, and I certainly do. However, I think I will act with some restraint and talk about only a small portion.

One of the most innovative sections is *Introduction to SAS.* Co-chaired by Imelda Go, Tom Winn, and Andrew Kuligowski, it focuses on the new SAS user. The presenters include some of the best and most skilled in the SAS world, and provide some thoughtful, well-considered advice and concepts. I recommend you go to the web site (http://www.ssu2001.com/) for more complete information. If you are a new SAS user, or work with or instruct new SAS users, this series of presentations could be of immense value.

The former Beginning and Advanced sections for Tutorials have been combined into a sin-

(Continued on page 5)

From the Editor

Randy C. Finch

Welcome once again to The SESUG Informant. I hope you enjoy this issue. Inside you will find articles by some familiar faces such as Ian Whitlock and Kirk Paul Lafler. But you will also see a couple of articles by some newcomers. Jack Shoemaker provides us with our book review this time, and our feature article is by yours truly, Randy Finch. Okay, so I am *not* a newcomer, having ed-

(Continued on page 18)

Inside this issue:

Around and Around With the SAS ROUND Function *by Ian Whitlock*	2
A Ton O' Tips Again *by Kirk Paul Lafler*	4
A Review of "Health Care Data and the SAS System" *by Jack Shoemaker*	8
Southern SAS Users Conference	10
Visualization of Atmospheric Modeling Data Using GPLOT, Annotation, and Macros *by Randy C. Finch*	12
Important Information from SAS	19
Index to Advertisers	20

The SESUG Informant; Fall 2001; Volume 3, Issue 2; ©2001 SouthEast SAS Users Group, Inc.

The SESUG Informant is a publication of the SouthEast SAS Users Group, Inc. It is published twice a year; in the spring before the SAS Users Group International (SUGI) conference and again in the fall before the SESUG conference.

If you would like to help out by advertising or writing articles for this newsletter, please contact the editor, Randy Finch, by calling 256-386-2197 or E-mailing rcfinch@tva.gov.

Around and Around With the SAS® ROUND Function

Let's take a look at the SAS ROUND function. You say, what can one possibly say about rounding? Well there was quite a stir recently in the SAS-L community when someone wanted a weird sort of rounding in SAS. In looking at the problem, it was soon revealed that SAS rounds up when halfway between two rounding points. For example, say we want to round to the nearest hundredth, then what should the rounded value of 8.135 be? In SAS we make the assignment:

```
rounded = round(8.135,.01) ;
```

In this case ROUNDED is 8.14 and we say that the number was rounded up to 8.14 as opposed to down to 8.13. Well what about 8.145? SAS rounds that up also, in fact every number halfway between two points to the nearest hundredth is always rounded up. One big question is, should it?

A number of prominent SAS-L'rs acted surprised and found this fact outrageous. Dale McLerran used a statistical argument that the mean of rounded values would be biased compared with the round of the mean value. He then carefully described how to round to the nearest even integer. Thus rounding up for half of the midpoints and down for half of the points. David Cassell pointed out that this was the IEEE standard and that, moreover, all self respecting languages followed the IEEE standard.

My colleague, Ed Heaton generated an example of a million random numbers between 0 and 100 to argue with Dale. He rounded each number to the nearest integer. His code:

```
/* Generate a million random
numbers and round them into
another variable. */

Data test1 ;
 Do i=1 to 10**6 ;
  val = 100*ranUni(123456);
  roundedVal = round(val,1);
  Output ;
```

Ian Whitlock

```
 End ;
 Drop i ;
Run ;
```

He then used SQL to calculate and print the two means.

```
Proc sql ;
 Title1 "test1" ;
 Select mean(val) as
        meanOfVal ,
        mean(roundedVal) as
        meanOfRoundedVal
   from test1 ;
```

The results showed that the means were very close together. If one thinks about it, this makes sense. The only time it matters is for numbers exactly at the halfway points and random numbers are not going to hit halfway points often enough to matter much. In a way this says that for many practical problems it isn't going to matter.

Dale came back quickly saying that it only mattered when the probability of landing on a midpoint was greater than 0. He then generated an example to make his point.

```
/* Generate y with values 0,
0.5, 1, 1.5, 2, 2.5, 3, 3.5,
4. */

/* Round y in two different
manners, using round function
which rounds up, and rounding
to nearest even integer. */

data test;
 do i=1 to 10000;
  x = rantbl(123456, 1/9,
     1/9, 1/9, 1/9, 1/9, 1/9,
     1/9, 1/9, 1/9);
  y = (x - 1) / 2;
  z1 = round(y);
  if y=.5 or y=2.5 then
     z2 = y - .5;
  else if y=1.5 or y=3.5 then
          z2 = y + .5;
  else z2 = y;
  output;
 end;
run;

proc means;
 var y z1 z2;
run;
```

The line:

```
x = rantbl(123456, 1/9, 1/9,
    1/9, 1/9, 1/9, 1/9, 1/9,
    1/9, 1/9);
```

generates an integer between 1 and 9 so that each integer has a probability of being generated 1/9 of the time. If you are less familiar with the various modifications of RANUNI then you could accomplish the same thing with:

```
x = ceil(9 * ranuni(123456));
```

Multiplication by 9 stretches the interval (0,1) to the interval (0,9). The CEIL function then looks up to the nearest whole number, producing an integer between 1 and 9. Each integer has an equal chance because the probability of being in any one of the 9 possible sub intervals is the same, 1/9.

The next line:

```
y = (x - 1) / 2;
```

shifts the numbers down by one (now 0 to 8) and divides them by 2, thus resulting in 0, .5, 1, 1.5, 2, 2.5, 3, 3.5, and 4. Now Dale has produced whole numbers and midpoints in such a manner that the midpoints are hit 4/9 of the time because there are 4 midpoint values. His report showed:

(Continued on page 9)

THE BASICS

Tip #1 – The SAS Data Set Structure.

A SAS System data set resembles a rectangular structure, similar to a spreadsheet, since it has one or more rows (observations) and one or more columns (variables). When looking at it, it would look like a piece of graph paper consisting of "x-y grids", *x* representing the rows and *y* representing the columns in the data set.

Tip #2 – The SAS Supervisor Controls All Activities in a Job.

The Supervisor program is the controlling program that directs all activities within the SAS System. It directs what program modules (DATA, PROC, or Macro) will be accessed according to the program statements.

Additionally, the Supervisor controls the following activities:

- Log message handling
- External file data management
- Memory management
- Initiation and termination of program steps
- Global statements and their settings
- Macro processing.

Tip #3 – Control Flow of a Traditional (Non-Macro) Program.

The control flow of a traditional (non-macro) program consists of the steps shown to the right.

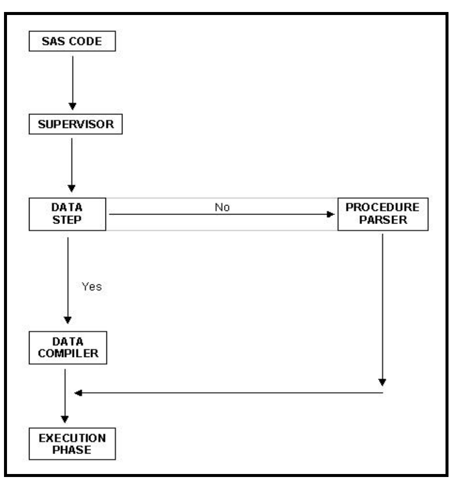
Tip #4 – Step Boundaries.

A step boundary is a condition where a step begins or a step ends. Although knowledge of this concept is not essential for the average user, it can help enable users to write better and more efficient programs. Because the SAS System proceeds in a top-down fashion processing one step before processing the next, an understanding of a step boundary is important to know.

Specific statements trigger a step boundary to occur. The DATA and PROC statements, acting as a step boundary, inform the SAS System to begin a new step. The RUN statement informs the SAS System that the current step is complete. An ENDSAS statement informs the SAS System that the current step is complete and to terminate the session.

When the statements RUN, DATA, or PROC are not specified, certain conditions such as an end-of-file condition can trigger a step boundary. A step boundary can also occur when a semicolon after data lines is processed.

Kirk Paul Lafler



SQL

Tip #5 – SQL Statements and Clauses Are Case-Insensitive.

It is important to note that SQL statements and clauses are ***case-insensitive***. So typing SELECT is the same as Select or select. Although there are no established guidelines, many SQL programmers type statements and clauses in uppercase characters, and table and column names in lowercase characters in an attempt to make program code more readable.

Tip #6 – PROC SQL Remains Active Until a QUIT; Statement is Issued.

PROC SQL is designed as an ***interactive procedure*** – meaning that multiple statements (e.g., CREATE TABLE, or RUN groups) can be processed without having to reissue the SQL procedure for each task. This is useful when more than one task is planned during a single session and can reduce the number of keystrokes. RUN groups are determined by specifying a semicolon (;) at the end of an SQL statement. A QUIT; statement is specified to terminate the procedure.

Tip #7 – Parentheses and Order of Evaluation in SQL.

Any number of AND and OR operators can be used in a WHERE clause. Consequently, it is advisable to use parentheses to explicitly group operators even if the default order of evaluation is the way you want it in the WHERE clause. Since parentheses have a higher order of evaluation than

(Continued on page 6)

From the President (Continued from page 1)

gle large *Tutorials* section. There are still beginning and advanced selections, but the single section allows for a greater coordination of the presentations. This is a very large section and co-chairs Carla Mast, Joy Smith, and Keith Cranford have assembled an impressive array of presentations ranging from "Basic Functions" to ODS, Macros, SQL, XML, and beyond. It is a powerful section.

The *Hands-On Workshops* is chaired by Heidi Markovitz and has sections on writing Java-based applications, ODS, Enterprise Guide, and others. Attendance in these is on a first-come, first-served basis, so if you wish to attend one, please be sure to review the *Conference program*, and arrive early. I'm sorry I can't point out highlights in each section, but with 2-½ days of presentations, this is quite a schedule.

And of course, there is the city itself: New Orleans, the Crescent City, the Big Easy. A most remarkable place.

And what next? Well, next year, SCSUG and SESUG go their separate ways: SCSUG to Richardson, TX; SESUG to Savannah, GA. The SESUG 2002 Conference co-chairs, Heidi Markovitz and David Maddox, are already planning that, and you will hear more about it both at SSU 2001 and in the months that follow.

Now I am getting ready to go to New Orleans and ***spice up my SAS skills!***

(Continued from page 4)

AND and OR, care should be exercised to construct correct WHERE clause operators.

To emphasize the importance of using parentheses, suppose you wanted to list PG and PG-13 movies having a length less than 121 minutes. The first query below uses a WHERE clause without any parentheses while the second query uses parentheses to group operators.

```
PROC SQL;
 SELECT TITLE, LENGTH
  FROM libref.MOVIES
   WHERE RATING = "PG" OR
      RATING = "PG-13" AND
      LENGTH <= 120;
QUIT;
```

The results from this query shows that nine movies were selected with three of them having a length greater than 120 minutes. Obviously the rows were not filtered as intended.

Title	Length
Casablanca	103
Jaws	125
Rocky	120
Star Wars	124
Poltergeist	115
The Hunt for Red October	135
National Lampoon's Vacation	98
Christmas Vacation	97
Michael	106

To fix the problem identified above, parentheses will need to be used to explicitly group related operators as follows.

```
PROC SQL;
 SELECT TITLE, LENGTH
  FROM libref.MOVIES
   WHERE (RATING = "PG" OR
     RATING = "PG-13") AND
     LENGTH <= 120;
QUIT;
```

The results from this query shows that six movies were selected and all have a length less than 121 minutes. This time the rows were filtered as intended.

Title	Length
Casablanca	103
Rocky	120
Poltergeist	115
National Lampoon's Vacation	98
Christmas Vacation	97
Michael	106

Tip #8 – Using the SQL Procedure to Summarize Data

Although the SQL procedure is frequently used to display or extract detailed information from tables in a database, it is also a wonderful tool for summarizing (or aggregating) data. By constructing simple queries, data can be summarized down rows (observations) as well as across columns (variables). This flexibility gives SAS users an incredible range of power, and the ability to take advantage of several SAS-supplied (or built-in) summary functions. For example, it may be more interesting to see the average of some quantities rather than the set of all quantities. Without the ability to summarize data in SQL, users would be forced to write complicated formulas and/or routines, or even write and test DATA step programs to summarize data. To see how an SQL query can be constructed to summarize data, two examples will be illustrated: 1) Summarizing data down rows and 2) Summarizing data across rows.

1. Summarizing data down rows

The first example shows a single aggregate result value being produced when movie-related data is summarized down rows (or observations). The advantages of using a summary function in SQL is that it will generally compute the aggregate quicker than if a user-defined equation were constructed and it saves the effort of having to construct and test a program containing the user-defined equation in the first place. Suppose you wanted to know the average length of all PG and PG-13 movies in a database table containing a variety of movie categories. The following query computes the average movie length and produces a single aggregate value using the AVG function.

```
PROC SQL;
 SELECT AVG(LENGTH) AS
      Average_Movie_Length
  FROM libref.MOVIES
   WHERE RATING IN
         ("PG", "PG-13");
QUIT;
```

The result from executing this query shows that the average movie length rounded to the hundredths position is 124.08 minutes.

Average_Movie_Length
124.0769

2. Summarizing data across columns

Being able to summarize data across columns often comes in handy, when a computation is required on two or more columns in each row. Suppose you wanted to know the difference in minutes between each PG and PG-13 movie's running length with trailers (add-on specials for your viewing pleasure) and without trailers.

```
PROC SQL;
 SELECT TITLE,
      RANGE(LENGTH_TRAIL,
              LENGTH) AS
          Extra_Minutes
```

(Continued on page 9)

A Review of "Health Care Data and the SAS® System" by Marge Scerbo, Craig Dickstein, and Alan Wilson

Jack N. Shoemaker

There are hundreds, perhaps thousands, of SAS consultants and contractors scattered about the United States who owe their livelihood in large part to being able to unravel the tangled web of health-care data. In a single, easy-to-read book full of useful and relevant examples, Scerbo, Dickstein, and Wilson, have placed the future employment prospects of these folks in serious jeopardy.

The first three chapters cover the data proper - what to expect, how they are organized, some of the pitfalls and gotchas to avoid. These chapters should become required reading for any user of health-care data regardless of analytic tool. A large part of the challenge for anyone new to health-care data is getting the lingo straight. Terms like "claims" and "encounters" are carefully explained and placed in the context of the health-care-delivery business. Chapter 3 contains samples of the primary data collection forms as well as pictures of how those data might manifest themselves in record descriptions and data files.

Chapter 4 is a tutorial on using SAS to read complex data structures. The narrative is peppered with code samples to amplify what they have written. The chapter builds to a rather elaborate data step that includes pointer controls, conditional output, arrays, and iterative input. Although the example at hand is a UB92 record, the techniques explained could just as easily apply to any complex record with repeating sets of fields. Garbage in; garbage out. If the data are noisy, the slickest program in the world won't help.

In Chapter 5, the authors describe how to use the tools available in SAS to test the validity of the data. As with chapter 4, the narrative contains many code examples showing how to use FREQ, MEANS, SORT, UNIVARIATE, and PRINT to explore and vet the data.

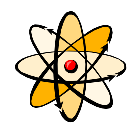
Chapter 6 is another tutorial on using Base SAS. Strip away the health-care theme, and this chapter could stand on its own merits as a sweeping survey of the many functions that SAS has at the ready to manipulate character strings, numbers, and dates. There's even a short section on using the SAS macro facility to accomplish simple, repetitive tasks.

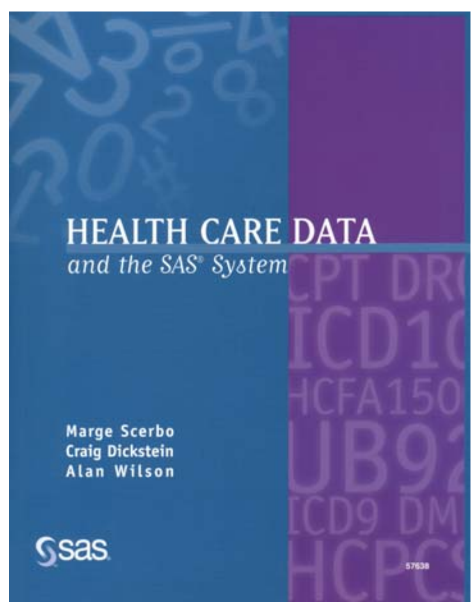
> "In a single, easy-to-read book full of useful and relevant examples, Scerbo, Dickstein, and Wilson, have placed the future employment prospects of these folks in serious jeopardy."

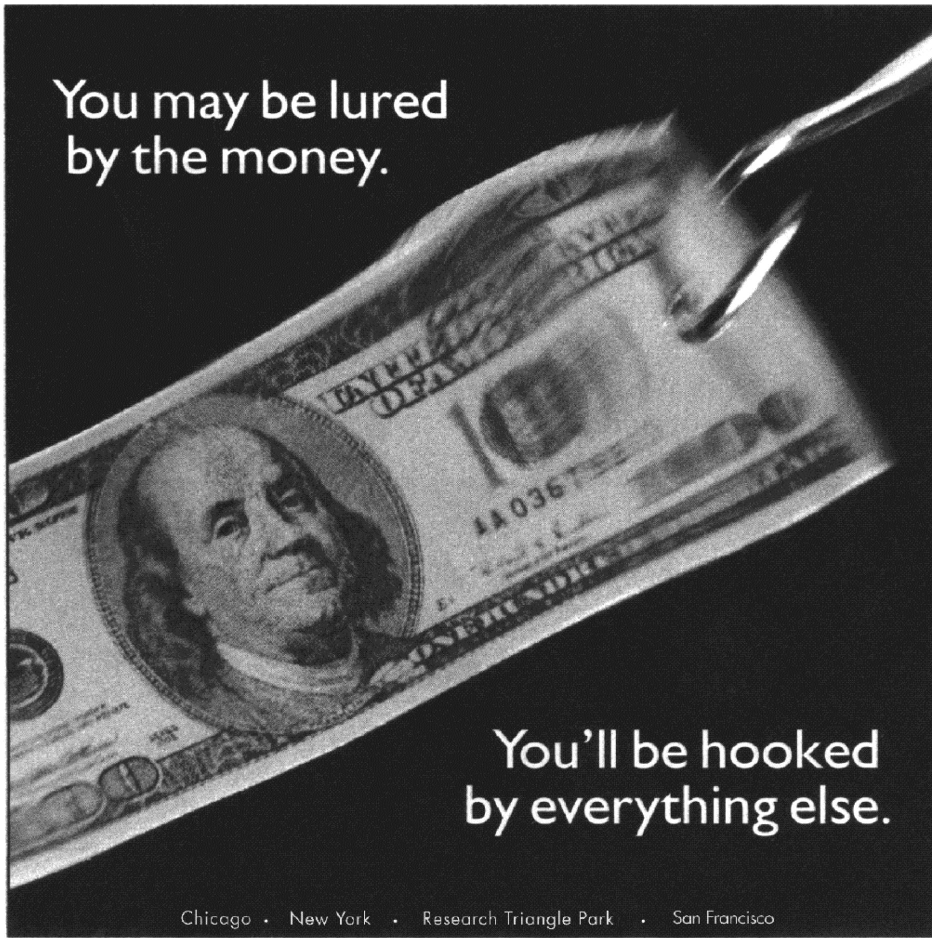
Continuing with the tutorial theme of the middle chapters, chapter 7 provides sanguine advice about the most powerful, yet most error-prone, part of SAS - the tools used to combine data. The MERGE statement is explained in wonderful detail complete with all the caveats you would come to expect at this point. All sorts of merges are discussed - one-to-one, one-to-many, and the notorious many-to-many. The MERGE statement is contrasted with interleaving SET statements, PROC APPEND, and PROC SQL. Each tool has its place and the authors do an excellent job of laying out the reasons why you would use one technique over another.

You understand the data content; you have sufficiently beaten them into submission; now it's time for analysis. Chapter 8 provides a review of the many tools SAS supplies for reporting and analysis. Careful consideration is given to data-reduction tools like PROC SUMMARY as well as data-display tools like PROC REPORT and PROC GPLOT. In keeping with the prescription-for-use thread which weaves its way through the whole book, the chapter concludes with a no-nonsense table suggesting what PROC to use when.

Chapter 9 brings all these concepts together as a case-study of emergency-room utilization in Maryland. The book then concludes with a perspicacious view of the future. There are new coding initiatives under foot; there are emerging technologies and treatment modalities; there are complex regulatory requirements and changing business climate. Perhaps the legion of SAS health-care consultants needn't search for new work just yet.







Around and Around... (Continued from page 2)

```
Experiment Mean Y Mean Z1 Mean Z2
---------- ------ ------- -------
1          1.9794 2.2200  1.9763
```

Notice that Mean Z1 is high compared with Mean Y. This is how SAS currently rounds. Mean Z2 is what following the IEEE standard would produce.

Both Ed and Dale are correct. For the most part, real examples are not rich in midpoints and Ed is correct that it usually doesn't matter. Dale, on the other side, is correct that one can construct sets of numbers where it will matter. Someone reported that version 9 of SAS will switch to the IEEE standard.

Most people are not even aware that one can write things like

```
z = round ( x , 1/3 ) ;
```

You do not have to round to decimal places. There are times when this is quite handy, but there is too little room left to describe them.

Before SAS version 8.1, one could write:

```
z = round ( x , m/n ) ;
```

and there would be no error message, but the calculation would be wrong unless M was 1. Fortunately this includes the standards, .1, .01, .001, etc. Howard Schreier pointed out that this bug has been fixed in 8.1.

A Ton 'O Tips Again (Continued from page 6)

```
  FROM libref.MOVIES
   WHERE RATING IN
         ("PG", "PG-13");
QUIT;
```

This query computes the difference between the length of the movie and its trailer in minutes and once computed displays the range value for each row as Extra_Minutes.

Title	Extra_Minutes
Casablanca	0
Jaws	0
Rocky	0
Star Wars	0
Poltergeist	0
The Hunt for Red October	15
National Lampoon's Vacation	7
Christmas Vacation	6
Ghost	0
Jurassic Park	33
Forrest Gump	0
Michael	0
Titanic	36

If you would like more information or have any questions about these tips, please contact: Kirk Lafler, Software Intelligence Corporation at KirkLafler@cs.com or visit his web site at: http://www.software-intelligence.com.

SSU 2001

The Conference for Southern SAS Users

August 19 - 22, 2001
New Orleans, Louisiana

Spice Up Your SAS® Skills!

***SESUG** and **SCSUG** invite you to join us in New Orleans for a very special conference –*
***SSU 2001**, the first ever joint conference of the SouthEast and South Central SAS Users Groups.*

SSU 2001 will feature presentations and workshops for every SAS user, from the beginner to the expert. Planned concurrent sessions for Monday, Tuesday, and Wednesday will include:

- *Applications Development*
- *Data Warehousing*
- *Internet, Intranet & the Web*
- *SAS Solutions*
- *Systems Architecture*
- *Tutorials*

It is not too late to Register !
Register online at WWW.SSU2001.COM

That is not all !!!

This is an activity-filled conference. **SSU 2001** will feature 140 presentations in 2-½ jam-packed days. Attendees will receive a bound 900+ page Proceedings of the presentations and a CD-ROM with those presentations. There will also be courses and workshops before and after the conference, and many opportunities to meet your fellow SAS users in a relaxed and fun setting.

Since New Orleans is also a great destination city for the entire family, we are planning a number of optional events suitable for your spouse and children. Check our Web site for more information.

Join us at **SSU 2001** and be prepared to leave New Orleans with more SAS software ideas and techniques than you thought possible.

Don't miss this conference !!!

Although early registration is over, there is still time to reserve your place at SSU 2001.

Conference Registration: $425.00 (after July 20, 2001)
$300.00 (Academic -- after July 20, 2001)
$125.00 (Student -- full time w/ current Student ID)
$ 85.00 (Guest -- after July 20, 2001)

Conference Hotel: **Hotel Inter-Continental New Orleans**
444 St. Charles Avenue
New Orleans, LA 70130
Phone: 504-525-5566
Room Rate: **$140.00**
If rooms are still available after July 18th, they will be honored at this rate.

Please visit our Web site at: **WWW.SSU2001.COM**

Volunteers !!!

Since SSU 2001 is a SAS **user** conference, there are many opportunities for you to volunteer. Participate as a Registration Assistant, Session Coordinator, or assist in a number of other roles. Help make this conference an exceptional experience for yourself and your fellow SAS users. Volunteer!

You may get more information and volunteer online at WWW.SSU2001.COM or by contacting the SSU 2001 volunteer coordinator:

Heidi Markovitz
305-365-0439 (voice)
simplysyst@aol.com

We hope to see you in New Orleans in August, 2001 for 4 exciting days of learning, camaraderie and fun! Join us at SSU 2001 where you will be able to *Spice Up Your SAS Skills!*

Visualization of Atmospheric Modeling Data Using GPLOT, Annotation, and Macros

Randy C. Finch

[NOTE: The graphics presented in this article are dependent on color for proper visualization; however, this is not a critical factor in understanding the article. If you want to see the graphics in color, please view the PDF version of this newsletter on our Web site: www.sesug.org.]

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? This article documents one such adventure I personally experienced.

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

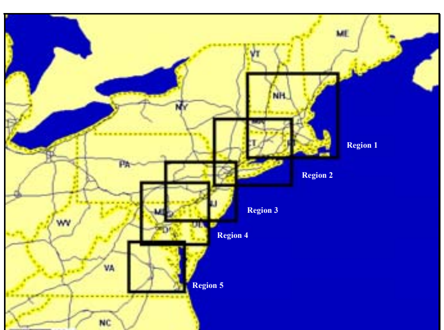


Figure 1 - The Northeast Corridor

```
Output from get_uamconc program for 07bas1D2 on 10jul95 for subregion 1
SW corner of grid: ( -73.500,  41.111) Deg
NE corner of grid: ( -69.500,  44.000) Deg
West-East (X) cell size: 0.167 Deg, South-North (Y) cell size: 0.111 Deg
Full grid: 192 x 189.  Portion used: ( 154, 137) to ( 177, 162)
Number of X values (west-east): 24, number of Y values (south-north): 26
There are data for  1 levels (Z):  1
Concentrations scaled by:  1.0000E+03, giving: (ppb)
   X    Y    Z  DATE-AND-HOUR  O3
   1    1    1 07/10/95:24:00  6.3681E+01
   2    1    1 07/10/95:24:00  5.2253E+01
   3    1    1 07/10/95:24:00  5.9229E+01
   4    1    1 07/10/95:24:00  6.7159E+01
…
[more data]
```

Listing 1 - ASCII File Format

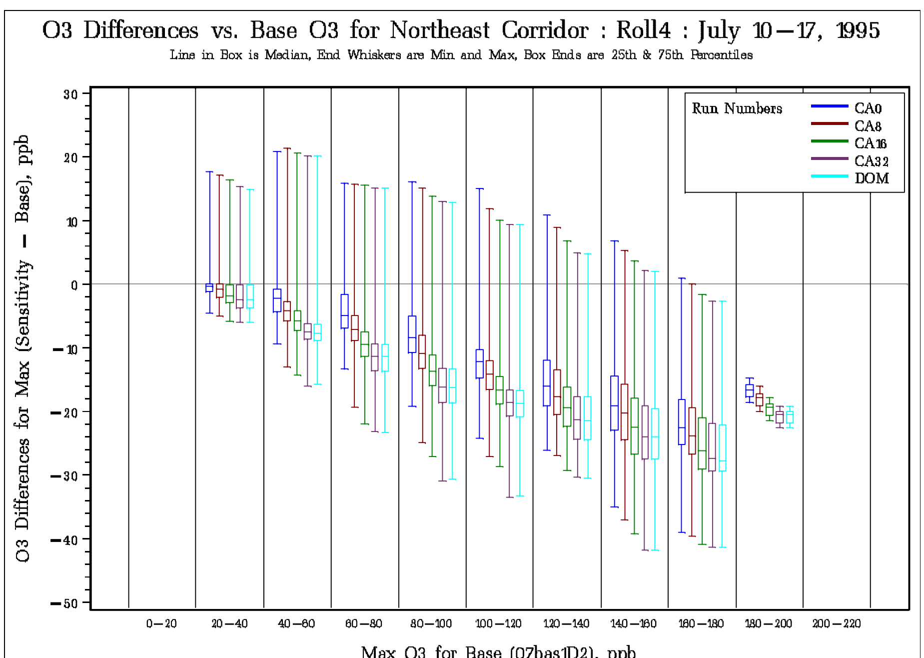


Figure 2 - The Original Box and Whisker Plot

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.

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.

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: &reg and &date. Since these macro variables cannot be

```
OPTIONS MPRINT;
******************************************************************************;
* MACROS                                                                     ;
******************************************************************************;

%MACRO readdata(fnin, dsname, o3name, reg);
FILENAME FILEIN "&fnin";
DATA &dsname;
  INFILE FILEIN FIRSTOBS=10;
  INPUT @1 X 4.
        @5 Y 4.
        @17 DATE 2.
        @29 &o3name E11.;
  ARRAY XOFFST{18} (153 144 132 126 123);
  ARRAY YOFFST{18} (136 128 117 110  95);
  X = X + XOFFST{&reg};
  Y = Y + YOFFST{&reg};
  KEEP X Y DATE &o3name;
RUN;
%MEND readdata;

******************************************************************************;

%MACRO loopregs(fnin, dsname, begdate, enddate, begreg, endreg);
%let datasets =;
%do date = &begdate %to &enddate;
  %let regsets =;
  %do reg = &begreg %to &endreg;
    %readdata(&fnin, &dsname&date&reg, O3&dsname, &reg);
    %let regsets = &regsets &dsname&date&reg;
  %end;
  DATA &dsname&date;
    SET &regsets;
  RUN;
  PROC SORT NODUPKEY;
    BY DATE Y X;
  RUN;
  %let datasets = &datasets &dsname&date;
%end;
DATA &dsname;
  SET &datasets;
RUN;
%MEND loopregs;

******************************************************************************;
*  PROGRAM                                                                   ;
******************************************************************************;

%loopregs('e:\jul95\guam.max.&reg..&date.jul95-07.tv.07bas1D2',BAS,10,17,1,5)
%loopregs('e:\jul95\guam.max.&reg..&date.jul95-07.tv.07roll4.CA0',CA0,10,17,1,5)
%loopregs('e:\jul95\guam.max.&reg..&date.jul95-07.tv.07roll4.CA8',CA8,10,17,1,5)
%loopregs('e:\jul95\guam.max.&reg..&date.jul95-07.tv.07roll4.CA16',CA16,10,17,1,5)
%loopregs('e:\jul95\guam.max.&reg..&date.jul95-07.tv.07roll4.CA32',CA32,10,17,1,5)
%loopregs('e:\jul95\guam.max.&reg..&date.jul95-07.tv.07bas1D2-tst5c',DOM,10,17,1,5)

******************************************************************************;

DATA SASUSER.ALLNEC;
  MERGE BAS CA0 CA8 CA16 CA32 DOM;
  BY DATE Y X;
  * group O3BAS by ranges of 20, offset by 4;
  O3BASC   = 20 * INT(O3BAS / 20) + 4;
  * create offsets for B&Ws of other runs;
  O3BASP1  = O3BASC + 3;
  O3BASP2  = O3BASC + 6;
  O3BASP3  = O3BASC + 9;
  O3BASP4  = O3BASC + 12;
  * calculate difference of sens runs and the base run;
  O3CA0D   = O3CA0  - O3BAS;
  O3CA8D   = O3CA8  - O3BAS;
  O3CA16D  = O3CA16 - O3BAS;
  O3CA32D  = O3CA32 - O3BAS;
  O3DOMD   = O3DOM  - O3BAS;
RUN;

******************************************************************************;

PROC SORT;
  BY O3BASC;
RUN;

******************************************************************************;

QUIT;
```

Listing 2 - Program to Create Data Set

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.

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;
```

This creates a new data set with all the regional data for a given run and date. The &regsets 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 &datasets;
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 %loopregs 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, &reg 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?

O3BASC = 20*INT(O3BAS / 20) + 4

And why are those other variables that are offset from O3BASC by increments of three being created? The fomula 20*INT(O3BAS/20) is used to categorize the base case data. If 0 <= O3BAS < 20, then O3BASC is zero; if

20 <= 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 GPLOT 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 GPLOT 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 GPLOT 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.

```
OPTIONS MPRINT;
**********************************************************;
*  MACROS                                                ;
**********************************************************;

* None yet;

**********************************************************;
*  PROGRAM                                               ;
**********************************************************;

GOPTIONS LFACTOR=3;

**********************************************************;

SYMBOL1  C=DEFAULT I=BOXT00 CI=BLUE   CO=BLUE   W=4 V=NONE;
SYMBOL2  C=DEFAULT I=BOXT00 CI=RED    CO=RED    W=4 V=NONE;
SYMBOL3  C=DEFAULT I=BOXT00 CI=GREEN  CO=GREEN  W=4 V=NONE;
SYMBOL4  C=DEFAULT I=BOXT00 CI=PURPLE CO=PURPLE W=4 V=NONE;
SYMBOL5  C=DEFAULT I=BOXT00 CI=CYAN   CO=CYAN   W=4 V=NONE;

**********************************************************;

AXIS1 COLOR=BLACK WIDTH=3.0 MAJOR=NONE MINOR=NONE
   ORDER = (-10 TO 230 BY 20)
   VALUE = (FONT=ZAPF H=1.0 J=CENTER "" "0-20" "20-40"
                   "40-60" "60-80" "80-100" "100-120"
                   "120-140" "140-160" "160-180"
                   "180-200" "200-220" "")
   LABEL = (FONT=ZAPF H=1.5 'Max O3 for Base (07bas1D2), ppb')
   ;
AXIS2 COLOR=BLACK WIDTH=3.0 MAJOR=(HEIGHT=1.0 WIDTH=2.5)
                            MINOR=(HEIGHT=0.7 WIDTH=2.0)
   VALUE = (FONT=ZAPF H=1.2)
   LABEL = (ANGLE=90 FONT=ZAPF H=1.5
          'O3 Differences for Max (Sensitivity - Base), ppb')
   ;

**********************************************************;

LEGEND1 ACROSS=1 FRAME
   POSITION = (TOP RIGHT INSIDE)
   LABEL = (F=ZAPF H=1.2 'Run Numbers')
   VALUE = (F=ZAPF H=1.2 'CA0' 'CA8' 'CA16' 'CA32' 'DOM')
   ;

**********************************************************;

TITLE1 J=C F=ZAPF
  'O3 Differences vs. Base O3 for Northeast Corridor :
          Roll4 : July 10-17, 1995';
TITLE2 J=C F=ZAPF H=1.1
  'Line in Box is Median, End Whiskers are Min and Max,
          Box Ends are 25th & 75th Percentiles';

**********************************************************;

PROC GPLOT DATA=SASUSER.ALLNEC ;
   PLOT O3CA0D*O3BASC O3CA8D*O3BASP1 O3CA16D*O3BASP2
        O3CA32D*O3BASP3 O3DOMD*O3BASP4 /
      OVERLAY
      HAXIS = AXIS1
      VAXIS = AXIS2
      LEGEND = LEGEND1
      FRAME
      VREF = 0
      HREF = 0 20 40 60 80 100 120 140 160 180 200 220
      /* ANNO = ANNO */
      ;
RUN;
```

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 val-

Add these lines to the beginning of the program section:

```
******************************************;

DATA TEMP;
  SET SASUSER.ALLNEC;
  NUM = 1;
RUN;

******************************************;

PROC MEANS SUM NOPRINT;
  VAR NUM;
  BY O3BASC;
  OUTPUT OUT=STATSUM SUM=NUMSUM;
RUN;

******************************************;

DATA ANNO1;
  LENGTH TEXT $ 4 COLOR FUNCTION STYLE $ 8;
  RETAIN FUNCTION 'LABEL' COLOR 'PINK'
          POSITION '2' HSYS '3' YSYS '2'
          XSYS '2' SIZE 3 STYLE 'ZAPF';
  SET STATSUM;
  TEXT = NUMSUM;
  X = O3BASC + 6;
  Y = -50;
RUN;

******************************************;

DATA ANNO;
  SET ANNO1;
RUN;

******************************************;
```

Add these lines to the titles section:

```
TITLE3 J=C F=ZAPF H=1.1
      'Numbers at Bottom are Total Number
          of Values for Each Group';
```

Un-comment this line in PROC GPLOT:

Listing 4 - Code to Add to Listing 3 to Print Number of Obs (Fig 3)

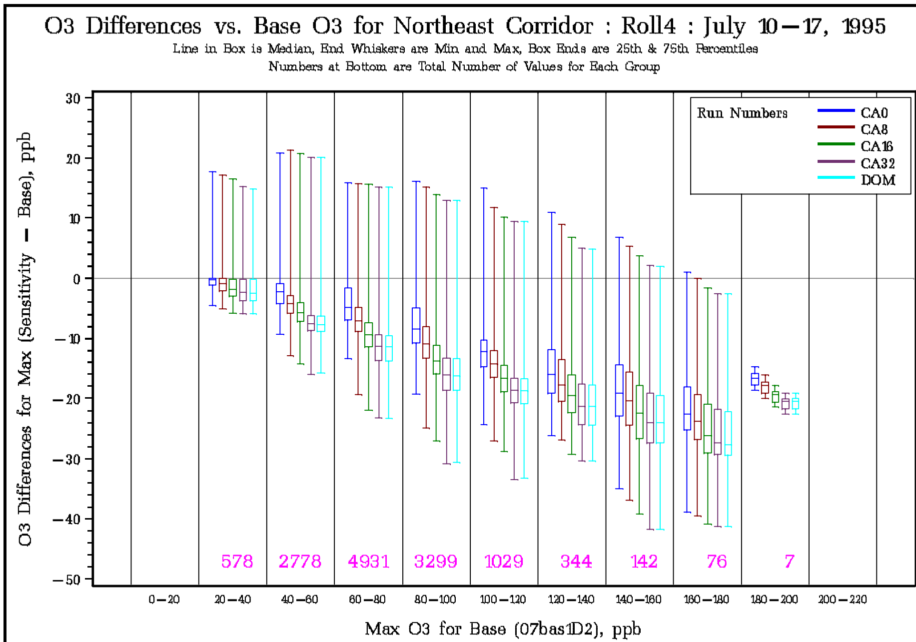


Figure 3 - Plot Showing Number of Observations

ues 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.

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 annotate data set named ANNO1 is created. It contains the standard annotate 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.

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.

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

Visualization of Atmospheric Modeling Data Using GPLOT, Annotation, and Macros (continued)

```
Add these lines to the macros section:

*********************************************;

%MACRO doanno2(xvar, yvar, color);
DATA ANNO2;
  LENGTH X Y 4. COLOR FUNCTION $ 8;
  RETAIN HSYS '3' YSYS '2' XSYS '2' SIZE .1;
  KEEP X Y COLOR FUNCTION COLOR HSYS YSYS
         XSYS SIZE;
  SET TEMP;
  IF _N_ >= 2 AND _N_ <= 6 THEN DO;
    FUNCTION = 'POINT';
    COLOR = &color;
    X = &xvar - 1.2;
    Y = &yvar;
    OUTPUT;
    FUNCTION = 'DRAW';
    COLOR = &color;
    X = &xvar + 1.2;
    Y = &yvar;
    OUTPUT;
  END;
RUN;
DATA ANNO;
  SET ANNO ANNO2;
RUN;
%MEND doanno2;

*********************************************;

%MACRO hilo5(from, to, by, xvar, yvar, color);
%do basval = &from %to &to %by &by;
  DATA TEMP;
    SET SASUSER.ALLNEC;
    IF O3BASC = &basval;
  RUN;
  PROC SORT DATA=TEMP;
    BY &yvar;
  RUN;
  %doanno2(&xvar, &yvar, &color);
  PROC SORT DATA=TEMP;
    BY DESCENDING &yvar;
  RUN;
  %doanno2(&xvar, &yvar, &color);
%end;
%MEND hilo5;

*********************************************;


Add these lines to the program section above
the GOPTIONS statement:

%hilo5(4, 204, 20, O3BASC,  O3CA0D,  'BLUE'  );
%hilo5(4, 204, 20, O3BASP1, O3CA8D,  'RED'   );
%hilo5(4, 204, 20, O3BASP2, O3CA16D, 'GREEN' );
%hilo5(4, 204, 20, O3BASP3, O3CA32D, 'PURPLE');
%hilo5(4, 204, 20, O3BASP4, O3DOMD,  'CYAN'  );

Put these lines in the titles section:

TITLE3 J=C F=ZAPF H=1.1
  'Wider Ticks are Five Highest and Five
   Lowest Values (not including Max and Min)';
TITLE4 J=C F=ZAPF H=1.1
```

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

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.

An annotate 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

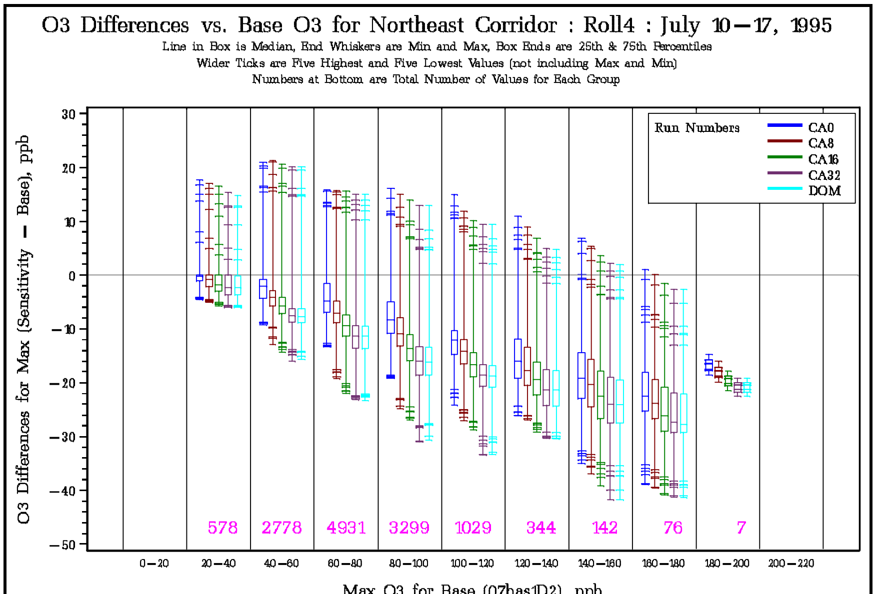


Figure 4 - Plot with High and Low Tick Marks

Visualization of Atmospheric Modeling Data Using GPLOT, Annotation, and Macros (continued)

Add these lines to the macro section:

```
%MACRO numgt0(xvar, yvar, color);
DATA TEMP;
  SET SASUSER.ALLNEC;
  IF &yvar > 0;
  NUM = 1;
RUN;
PROC MEANS SUM MAX NOPRINT;
  VAR NUM &yvar;
  BY &xvar;
  OUTPUT OUT=STATSUM SUM=NUMSUM MAX=DUMMY MAX;
RUN;
DATA ANNO3;
  LENGTH TEXT $ 4 COLOR FUNCTION STYLE $ 8;
  RETAIN FUNCTION 'LABEL' COLOR &color
         POSITION '6' ANGLE 90 HSYS '3'
         YSYS '2' XSYS '2' SIZE 1.4
         STYLE 'ZAPF';
  SET STATSUM;
  TEXT = NUMSUM;
  X = &xvar;
  Y = MAX;
RUN;
DATA ANNO;
  SET ANNO ANNO3;
RUN;
%MEND numgt0;
```

Add these lines to the program section above the GOPTIONS statement:

```
%numgt0(O3BASC,  O3CA0D,  'BLUE'  );
%numgt0(O3BASP1, O3CA8D,  'RED'   );
%numgt0(O3BASP2, O3CA16D, 'GREEN' );
%numgt0(O3BASP3, O3CA32D, 'PURPLE');
%numgt0(O3BASP4, O3DOMD,  'CYAN'  );
```

Substitute this line in the titles section:

```
TITLE4 J=C F=ZAPF H=1.1
       'Numbers at Bottom are Total Number of
        Values for Each Group, Numbers Above
        B&Ws are Number of Values > 0';
```

Listing 6 - Code to Add to Listings 3-5 to Print Number of Obs > 0 (Fig 5)

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.

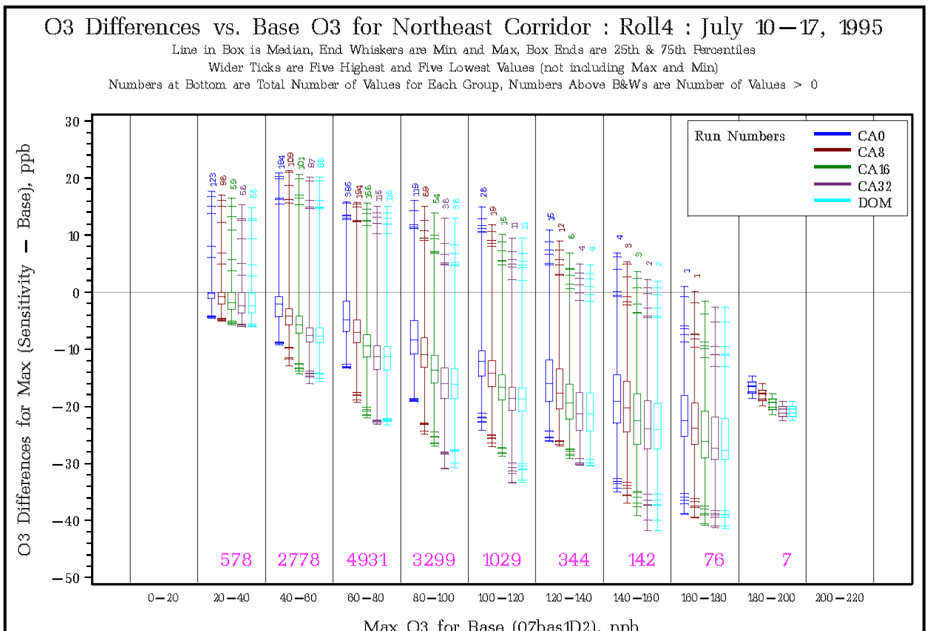


Figure 5 - Final Plot with Number of Obs > 0

From the Editor (continued)

(Continued from page 1)

ited this newsletter now for three years. However, this is the first article of any length I have written for the Informant. "Why?" you ask. Well, basically I am a shy person and I am just now trying to break out of my shell. And if you believe that, I have some seaside property in Arizona I would like sell you!! Actually, I was having a hard time getting enough content for this issue. In the past, I have always had more than enough material and would have to delay publication of some articles. Not true this time. So, I did what any good editor working for a salary of $0.00 per hour would do. *I started begging!* When that did not work, I did the second thing any good editor would do. I published one of my own articles. Problem solved, case closed.

I said all that so I can say this. If you have a SAS article that you would like to see in print in this newsletter, please contact me and let me know. I will happily consider it for publication. If you would like to advertise with us, let me know that also. We have reasonable rates and offer discounts to people who provide articles for the newsletter.

Well, I am all geared up to travel to New Orleans in August for the Southern SAS Users conference. I am registered and have my airline and hotel reservations. If you have not signed up yet, you are a bit late. But we can accommodate you anyway. You don't want to miss that Cajun Fais Do-Do on Tuesday night. I hear it's going to be a blast. The conference chairs tell me that Do-Do is pronounced "dough-dough", which I am relieved to hear since I originally thought it was pronounced "dew-dew". With a name like that I didn't know what to expect.

Oh, and there'll be a lot of good SAS papers at the conference, also. Be there!

Important Information From SAS

SESUG Liaison
Lisa Brugh
Phone: 919-531-2320
E-mail: lisa.brugh@sas.com

SESUG Local Users Group Liaison
Patsy Harbour
Phone: 919-531-2855
E-mail: patsy.harbour@sas.com

SUGI 27 News

Make plans now for SUGI 27, scheduled for 14-17APR02 in Orlando, Florida!

Conference Chair Cyndie Gareleck will issue the Call for Papers and Participation in July. In addition to the brochure, the Call for Papers information will be available online at:

http://www.sas.com/usergroups/sugi/sugi27

Subscribe to UGNEWS

If you'd like to receive periodic information about the Users Group Support program, subscribe to SAS' electronic newsletter, UGNEWS. Sent via e-mail and posted on the Web at:

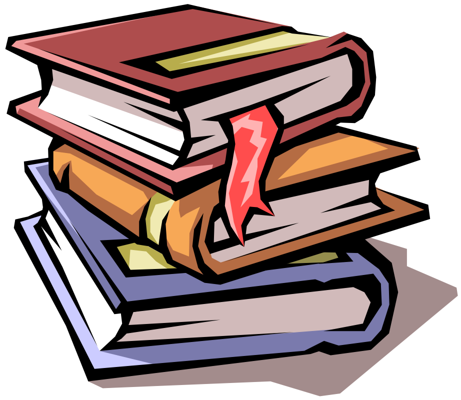
http://www.sas.com/usergroups/ugnews ,

the newsletter:

- announces the formation of new users groups and new services from the Executive and User Events
- contains features about SAS users group services
- includes reminders for upcoming users group conferences
- lists SAS and BBU speakers scheduled for upcoming users group meetings
- provides tips and news from leaders in the SAS users group community.

To start receiving UGNEWS: send e-mail to sugweb@sas.com saying "SUBSCRIBE UGNEWS-L" and include your full name or use the form on our Web site at:

http://www.sas.com/usergroups/ugnews-form.html



Local Users Groups in the Southeast

ATLANTA SAS USERS GROUP

BIRMINGHAM USERS GROUP FOR SAS

FL GULF COAST SAS USERS GROUP

MARYLAND SAS USERS GROUP

MEMPHIS AREA SAS USERS GROUP

NORTH FL USERS OF SAS SOFTWARE

NORTHEAST GEORGIA SAS USERS GROUP

NORTHERN VA SAS USERS GROUP

RESEARCH TRIANGLE SAS USERS GROUP

SOUTH CAROLINA SAS USERS GROUP

SOUTH FLORIDA SAS USERS GROUP

TRIAD SAS USERS GROUP

VIRGINIA SAS USERS GROUP

WASHINGTON DC SAS USERS GROUP

WILMINGTON AREA SAS USERS GROUP

Index to Advertisers

Company Name	Page #
Abt Associates	Back Page
ASG	9
DataLogic Consulting	5
Destiny Corporation	3
Qualex Consulting Services	5
Research Triangle Institute	7
Smith Hanley	7

SOUTHEAST SAS USERS GROUP

Serving SAS Users in the Southeast Since 1993

We're on the Web
www.sesug.org
www.ssu2001.com

The SESUG Informant
POB 8345
Clearwater, FL 33758

SESUG EXECUTIVE COUNCIL MEMBERS	TELEPHONE	E-MAIL
Gregory Barnes Nelson (1999 co-chair)	919-465-0322 x351	greg.barnesnelson@statprobe.com
Dan Bruns, Vice President (1997 co-chair)	423-751-6430	debruns@tva.gov
Philip J. d'Almada, Secretary (2000 co-chair)	404-639-6120	pdalmada@cdc.gov
Frank C. Dilorio (1994 & 1996 co-chair)	919-942-2028	fcd1@mindspring.com
Randy C. Finch, Treasurer, Editor (1998 co-chair)	256-386-2197	rcfinch@tva.gov
Maribeth H. Johnson (2000 co-chair)	706-721-3785	maribeth@stat.mcg.edu
F. Joseph Kelley, President (1998 co-chair)	706-542-5359	jkelley@arches.uga.edu
Andrew T. Kuligowski (1997 co-chair)	727-738-3000 x6545	kuligoat@tvratings.com
David Maddox (2002 co-chair)	205-977-0415	david.maddox1@bridge.bellsouth.com
Heidi Markovitz (2002 co-chair)	305-365-0439	simplysyst@aol.com
George Matthews (1999 co-chair)	706-542-5359	gmatthew@arches.uga.edu
S. David Riba (1993 & 2001 co-chair)	727-726-6099	dave@jadetek.com