

Generating Participant Specific Figures Using SAS Graphic Procedures

Carry Croghan and Marsha Morgan, EPA, Research Triangle Park, NC

ABSTRACT

An important part of our research at the U.S. Environmental Protection Agency (EPA) is to effectively communicate the results from observational studies to the study participants. This can be particularly challenging where participants are from different socio-economic backgrounds. For our present study, we developed a short, descriptive report that was easy to understand by over 250 participants from low-, middle-, and high-income backgrounds. In each report, we present the chemical concentrations that were found at the participant's home compared to all homes in the study. The most effective way we found was to present the participant's study results using colorized figures. We will discuss the complex process that involved using SAS procedures gmap and greplay, and macro coding, to develop the figures for the reports. By creating an automated process, this program was an excellent solution that was creative, time efficient, and less prone to human error than the historical method of manually generating the plots.

INTRODUCTION

In observational studies conducted by the EPA, participants are usually provided their study results in paper reports. Because people who participate in these studies can be from various socio-economic backgrounds, the EPA commonly displays them in easy to understand charts or figures within these reports. For a recent study, a figure displaying the participant's results compared to the overall study results was created to be incorporated within a larger report. The percentile score for each participant was calculated and graphed in this figure. To save time and reduce the possibility of error, a SAS program was written to generate the individual figures.

Based on a sketch showing the specifications of the figure that needed to be generated for each participant, the program consisted of three major components. The first component involved re-structuring the data to perform the statistical analysis. The second and most challenging element was to convert a statistical value to a location on a graph. Finally, the graphs were then placed within a template to create a single file for each participant.

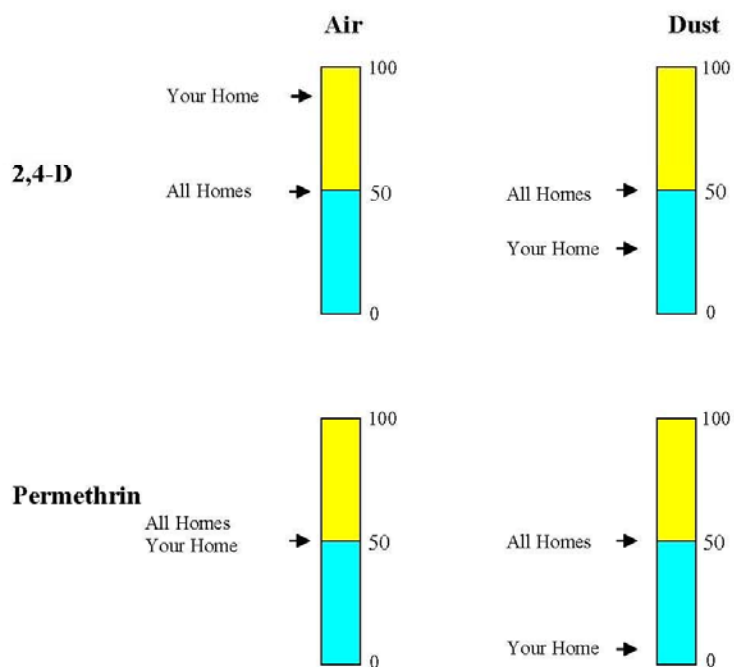
In this paper, we will discuss the different SAS programming components (i.e., data step, SAS graph, and macro programming) that were utilized to construct the requested figures. Some knowledge of SAS programming is assumed. We will point out various efficiency elements that were incorporated within the programming. This paper is not intended to be a complete tutorial on the SAS graphics module. Only elements that were specifically used in generating the figures will be discussed. These elements include proc gmap, proc gslide, and proc greplay.

SPECIFICATIONS

An EPA investigator created a sketch of the figure to be use to report study results to each participant (Figure 1). The sketch provided the structure for presenting the information and details on such things as chemical, matrix (e.g. air, dust, food), and sample types. This figure was designed to list the five chemicals that were most likely to be found in air and dust samples at the participant's home. A copy of the sketch follows:

SKETCH OF FIGURE 1

Figure 1



Repeat info in Figure 1 for the following chemicals:

Dibutylphthalate
Bisphenol-A
Benzo[a]pyrene

* All the info in Figure 1 will be on one page in the final report.

DATA STEPS

The data were organized at the sample - chemical level. The original data set contained 54 chemicals and seven different matrices. The requested figure only required five of these chemicals in two of the matrices. A where statement in a data step efficiently limits the number of records read and processed.

```
data temp;
set &in_file(where=(compound in ("2,4-D","Dibutylphthalate", "Bisphenol-A",
    "Benzo[a]pyrene", "Permethrin") & sampletype= "Home" & matrix in
    ("Air", "Dust")));
. . .
```

```
run;
```

In addition, some participants had multiple samples collected at their homes (e.g. dust samples). To reduce them to a single measurement per participant, matrix, and chemical, a proc means with class and output statements were used. Since proc means generates one record per combination of class levels, the missing function was used to limit the output to only those records of interest.

```
proc means data=temp noprint;
  class matrix compound pid;
  var result;
  output out=temp1 (where=(not(missing(matrix)) & not(missing(pid)) &
                          not(missing(compound)))) mean = result;
run;
```

Next, the participants' percentiles were calculated using the rank procedure with the percent option.

```
proc rank data=temp1 out = temp2 percent;
  by matrix compound;
  var result;
  ranks prt;
run;
```

Since several participants did not have data for air or dust samples, a data shell was created to generate empty records for the missing samples. By ensuring that each participant had the same number of records, the input for the macro loop was uniform. Thus, the programming of the macro loop was simplified.

The data shell was generated in two steps. The first step involved creating a dataset with one record for each participant. There were several methods to generate such a dataset. We chose to use a sql statement.

```
proc sql;
  create table shell as
  select distinct pid
  from &in_file
  where sampletype= "Home";
quit;
```

The second step was to fill-in the combinations of matrix and chemical. As in all programming problems, there were several approaches. We chose the brute force approach within a data step. Cutting and pasting within the editor was viewed as a quicker solution to the problem than the alternatives.

```
data shell;
  set shell;
  length matrix $4 compound $60;

  matrix = 'Air';
  compound = "Permethrin"; output;
  compound = "2,4-D"; output;
  compound = "Dibutylphthalate"; output;
  compound = "Bisphenol-A"; output;
  compound = "Benzo[a]pyrene"; output;

  matrix = 'Dust';
  compound = "Permethrin"; output;
  compound = "2,4-D"; output;
  compound = "Dibutylphthalate"; output;
  compound = "Bisphenol-A"; output;
  compound = "Benzo[a]pyrene"; output;
```

```
run;
```

The dataset created by the proc means was merged with the data shell. The resulting analysis dataset was now uniform with an equal number of records for all participants.

PROC GMAP

To generate the colorized sections within the figure, proc gmap was used. The proc gmap requires three datasets: map, color, and annotation.

MAP AND COLOR

Both the map and color datasets were the same for all participants. The map dataset is a series of x and y coordinates that outline each section of the map. Since the figure had two different rectangular shaped colored sections, the map dataset has eight records, one record for each corner of each section. The unit for the x and y coordinates on the corners was percentage of the total page. The exact values were determined by trial and error. The color dataset, which specifies the pattern to be used in each map section, must have the same number of records as there are sections in the map dataset.

```
data map;
    input section x y @@;
datalines;
1 0 0
1 0 50
1 20 50
1 20 0
2 0 50
2 0 100
2 20 100
2 20 50
;
run;

data color;
    input section level;
datalines;
1 1
2 20
;
run;
```

ANNOTATION

The annotation dataset was data-driven and based upon a single record from the analysis dataset. Annotation allows text or symbols to be placed on a map. The uses of the annotation are numerous. In this particular case, the annotation was used to label the different levels of the figure, and indicate the median of all homes and the participant level. The annotation dataset contains very specific variables including:

Variable	Description
function	command do you want issued, in this case the function was set to 'label'
style	the font used in the label
size	of the font
hsys	coordinate-system for the size variable, set to 3 = percentage of graphics output area
color	of the text
text	of the label
xsys and ysys	coordinate-system for the x and y respectively, set to 2 = data values
x and y	the coordinates of the starting point
position	the position of the text in relation to the starting point

Variable	Description
when	when the function will be performed either b= before the map is generated or a=after the map is generated

Each command is a separate record in the annotation dataset. When any variable is changed, you must output a separate record. Since the arrow generated in the map was a different font than the text, they were each a separated record. The code that was used to produce the annotation dataset is below.

```
data annomap;
  set temp4(where = (matrix = "&med" & pid = "&id" & compound = "&chem"))
    keep = matrix compound pid prt );
length color function style $8 text $20 ;
retain xsys ysys '2' hsys '3' when 'a' position '6';
function='label'; style = 'swissb'; color='black'; size=5;

x = 20; y=5; text = ' 0'; output;
x = 20; y=51; text = ' 50'; output;
x = 20; y=100; position='C'; text = ' 100'; output;

if prt not = 50 & prt > .z then do;
  x = 0; position='4'; y=51;
  text='All Homes'|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
  x = 0; position='4'; y=49; size=10; style='marker'; text='J'; output;
style = 'swissb'; size=5;
  x = 0; y=prt; position='4';
  text='Your Home'|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
  x = 0; position='4'; y=prt-2; size=10; style='marker'; text='J'; output;
style = 'swissb'; size=5;
end;

else if prt <= .z then do;
  x = 0; position='4'; y=51;
  text='All Homes'|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
  x = 0; position='4'; y=49; size=10; style='marker'; text='J'; output;
style = 'swissb'; size=5;
end;

else do;
  x = 0; position='A'; y=51;
  text="All Homes"|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
  x = 0; position='4'; y=49; size=10; style='marker'; text='J'; output;
style = 'swissb'; size=5;
  x = 0; y=prt; position='D';
  text="Your Home"|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
end;

run;
```

The brute force approach was selected again to generate the dataset. There were three possible conditions that had to be addressed: missing data, participant data value was equal to the median, or otherwise. By ordering the if, then, else statements from most likely to least likely, the program was more efficient. Since the majority of the data was neither missing nor equal to the median, this was the first of the if, then, else statements. In this situation, the y coordinate was set to the participant's percentile score. If the participant's score was equal to the median, then the note indicating the median level was shifted up, the participants note was shifted down and the arrow was placed between the two notes. If the participant data was missing, then only the median note was outputted.

GMAP

A proc gmap was used to generate a figure for each combination of chemical, matrix, and participant. The color dataset which defines the pattern in each section is the main dataset called. The map option is used to indicate the name of the map dataset, which outlines the shape of each section. The annotation dataset, the real work horse, is called by using the anno option of the choro statement. Choro is the abbreviation for choropleth, which is a map that uses colors or patterns to indicate different sections of the map.

```
proc gmap data=color map = map gout=graphs.graph;
  id section;
  choro level/ coutline = black nolegend anno =annomap name = "&med._&chem";
  pattern1 v=ms color = blue;
  pattern2 v=ms color = yellow;
run;
quit;
```

This is the code to produce one of map for a participant, chemical, and matrix combination. We needed to place 10 such maps onto a single page. A graphic template is designed especially for this task.

COMBINING ALL ELEMENTS

A graphic template is a structure into which graphics elements can be placed. Maps were replayed into a graphic template thus generating a single graphic output for each subject. In addition to the maps, the graphic template contains slides that were used to print titles, labels, and footnotes. Using proc gslide, a general title and footnote was generated and saved. The chemical labels were created and saved into their own files. A graphics catalog and file name were used to control where the slide was saved. A proc gslide requires either a title, footnote, or note statement. For the chemical labels, we used the note statement. For example, this was the proc gslide code that created the label for the permethrin figure.

```
proc gslide gout=graphs.graph name = "gslide5";
  note f = swiss h=15pct "Permethrin";
run;
quit;
```

The title and footnote statements were used to generate the slide that had the title and footnote.

```
proc gslide gout=graphs.graph name = "gslide";
  title f= swiss h=4pct j=l "Figure 1";
  footnote f= swiss h=.9pct j=r "&id";
run;
```

We generated a graphic template that was a 3 by 5 grid in which the different elements were placed. A graphic template can be created interactively within the SAS graphics editor or batch mode. In this case, batch mode was used. The graphic template definition is similar to the map dataset. Sections were defined by inputting the coordinates of the four corners. The exact values were determined by trial and error. Since this element did not require modification, the graphic template was created in a separate program. The basic design was a 3 by 5 grid with an additional section that encompasses the entire page for the title and footnote slide. Therefore, there were 16 defined sections. The coordinates were scaled as a percentage of the total page. The width page was divided into three equal sections across the page minus space for the margins. Likewise, the length of the page was divided into five equal sections minus a portion of the page for the margins. The proc greplay was used to generate the graphic template with a tc option stating where the graphic template catalog was located. The statement tdef was used to define a new graphic template. Then, each section was enumerated and defined by setting the values of the lower left, upper left, upper right and lower right coordinates. This was a section of the code that was used:

```
proc greplay nofs tc=graphs.templt;
tdef newl2r5
  1 / llx=26.0 lly=77.0 ulx=26.0 uly= 95.0 urx= 58.0 ury= 95.0 lrx= 58.0 lry=77.0
  2 / llx=58.0 lly=77.0 ulx=58.0 uly= 95.0 urx= 90.0 ury= 95.0 lrx= 90.0 lry=77.0
  ...
```

```

16 / llx= 5.0 lly= 5.0 ulx= 5.0 uly= 14.0 urx= 24.0 ury= 14.0 lrx= 24.0 lry= 5.0
;
run;
quit;

```

A proc greplay was used to combine all of the individual elements together. The input and output graphics catalogs were specified as options. The treplay statement specifies which graphic element was to be placed in each section of the template.

```

proc greplay nofs igout=graphs.graph gout=graphs.participantreport;
  tc graphs.templs;
  template newl2r5;
  treplay      1:"Air_2_4_"
               2:"Dust_2_4_"
               ...
               16:gslide5;
run;
quit;

```

The last element of code was one more greplay to rename the plot to include the participant identification number.

```

proc greplay nofs igout=graphs.participantreport;
  modify Template / name=fig&id;
run;
quit;

```

All of this programming was necessary to create a single figure to be incorporated into each participant's report. The data re-structuring and statistical analysis were performed on all participants at one time. The generation of the figures needed to be done individually for each participant.

MACRO

To efficiently run this program for all of the participants, a macro loop was used. The first step in designing a macro loop was to determine whether an element was static or dynamic. The static elements included the label slides, map, and color datasets. These elements needed to be performed only once. They can be performed outside of the macro. Only the dynamic elements that changed with every participant were included within the macro. These elements included the annotation dataset, the title and footnote slide, and the individual compound and matrix maps. To make the macro procession easier, a macro variable was generated using proc sql to hold all unique participant identification numbers.

```

proc sql noprint;
  select distinct(pid)
  into :pidlist separated by ' '
  from temp4;
quit;
run;

```

Separate macro variables were created to hold the list of chemicals and matrices. The main participant loop was an until loop. An until loop was selected to make the programming easier by having the data determine the stopping point of the loop. Nested until loops were used to run through all the combinations of chemicals and matrices. Since an until loop was used, the pointing variable must be re-initialized between each nested loop run. In addition, all of the previous participant's graphs must be deleted.

```

%macro loopit;
%let i_count = 1;
%let m_count = 1;
%let c_count = 1;

%do %until(%scan(&pidlist,&i_count)= );

```

```

** Deleting previous participants graphes **;
proc greplay igout=graphs.graph nofs;
  delete "gslide" ;
  delete "Air_perm" ;
  delete "Air_2_4_" ;
  delete "Air_dibu" ;
  delete "Air_bisp" ;
  delete "Air_benz" ;
  delete "Dust_perm";
  delete "Dust_2_4_";
  delete "Dust_dibu";
  delete "Dust_bisp";
  delete "Dust_benz";
quit;
run;

  %let id = %scan(&pidlist,&i_count);

title;
footnote;

%let c_count = 1;

%do %until(%qscan(&chemlist, &c_count," ")= );
%let m_count = 1;
  %let chem = %qscan("&chemlist", &c_count," ");

%do %until(%scan(&medialist,&m_count)= );
  %let med = %scan(&medialist,&m_count);

***** ,

data annomap;
  .
  .
  .
run;

proc gmap data=color map = map gout=graphs.graph;
  .
  .
  .
run;
quit;

title;
footnote;

%let m_count=%eval(&m_count+1);
%end;
%let c_count=%eval(&c_count+1);
%end;

proc gslide gout=graphs.graph name = "gslide";

```

```

        title f= swiss h=4pct j=l "Figure 1";
        footnote f= swiss h=.9pct j=r "&id";
run;

title;
footnote;

proc greplay nofs igout=graphs.graph gout=graphs.report;
    tc templ.templs;
    template newl2r5;
    treplay          1:"Air_2_4_"
                    2:"Dust_2_4_"
                    .
                    .
                    .
                    16:gslide5;
run;
quit;

proc greplay nofs igout=graphs.report;
    modify Template / name=fig&id;
run;
quit;

%let i_count=%eval(&i_count+1);

%end;

%mend;

```

The looping creates an efficient way to process the data for all the participants. Also, during testing, the loop was run for a small subset of participants allowing for quicker runs. Particular participants with known data issues could be selected for debugging the program. These elements combined to create an effective method of performing a repetitive process.

CONCLUSION

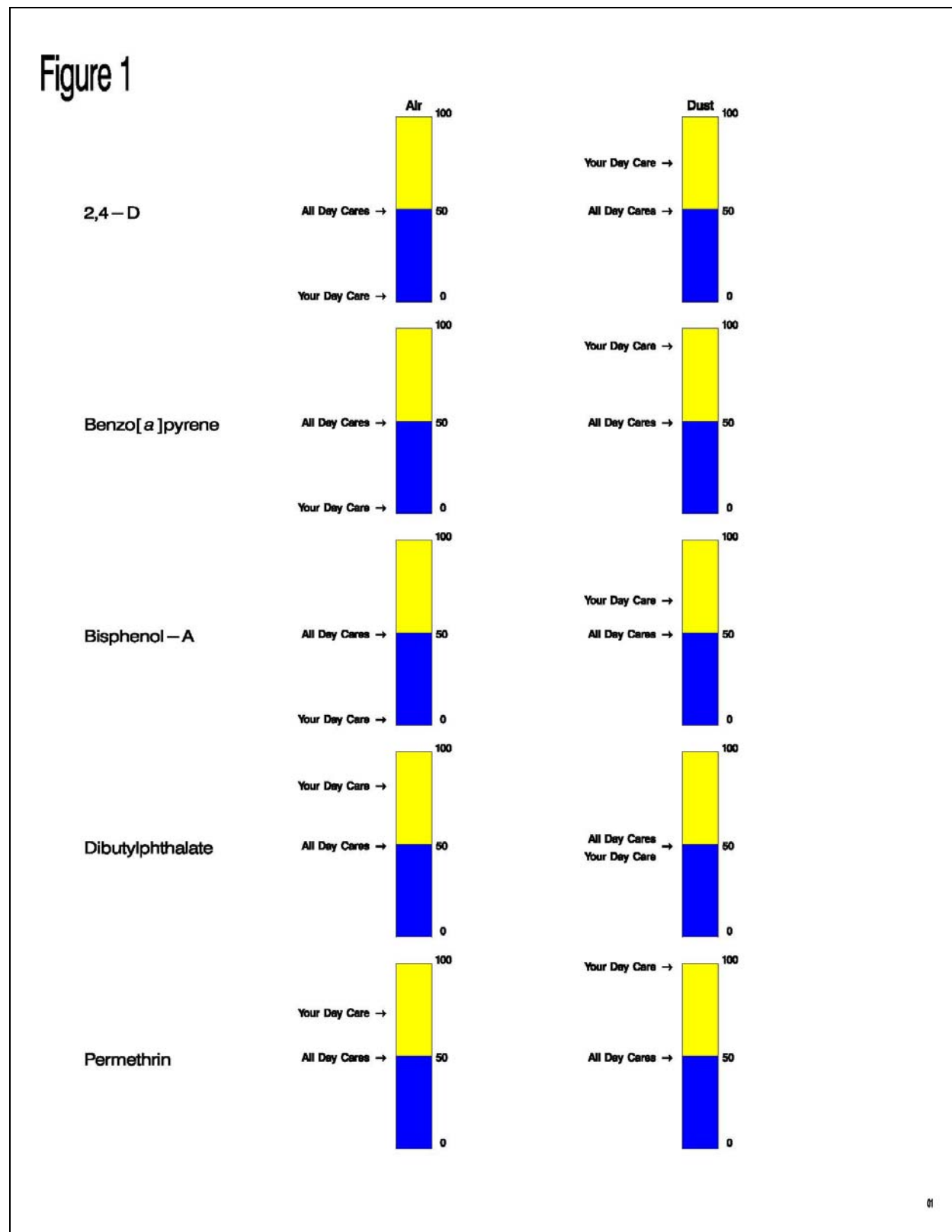
The goal was to efficiently generate figures to display study results in participant reports. This was accomplished by writing a SAS program that generated data-driven figures. This program consisted of three main steps:

1. data reduction and calculation,
2. generating participant, chemical, and matrix specific figures, and
3. combining the figures into a single file for each participant.

The participant results were displayed as percentile scores. The percentiles were calculated with ease because of proc rank and a single option, percent. A proc gmap generated the individual figures using three datasets: map, color, and annotation. The map and color datasets were simple and the same for all participants. The annotation dataset was data-driven changing for each participant, chemical, and matrix combination. It converted a statistical value to a location on a figure. Using a graphic template, proc greplay was used to place all of a participant's figures into a single figure which included title, footnote, and labels. Thus, a single file was generated for each participant displaying their percentiles scores for selected chemicals and matrices.

These figures were successfully incorporated into the participant reports. Although the programming was time consuming, it was more efficient and precise than the alternative method of hand generating the figures. Below is one of the figures that was generated using this SAS program.

FIGURE 1 AS GENERATED BY SAS



COMPLETE CODE

```
*****,  
***   Program Name:      Figures  
***   Author:           Carry Croghan  
***   Version:          1  
***  
***   Purpose           Creating the figures of results for participant reports.  
***  
*****,  
                *****,  
***   Set-up           ***;  
                *****,  
                ;  
  
options ps=45;  
  
                *****,  
***   libraries       ***;  
                *****,  
                ;  
  
libname in_put "C:\sasdata";  
libname graphs "C:\graphs";  
libname templ "C:\templates";  
  
*****,  
  
%let in_file = in_put.results;  
  
*****,  
  
%let chemlist = %nrstr(2,4-D Permethrin Dibutylphthalate Bisphenol-A  
Benzo[a]pyrene) ;  
%let medialist = Air Dust;  
  
*****,  
  
proc catalog c=graphs.report kill force;  
run;  
  
proc catalog c=graphs.graph kill force;  
run;  
  
*****,  
proc sql;  
    create table template as  
    select distinct substr(pid,1,2) as pid  
    from &in_file  
    where sampletype= "Home";  
quit;  
  
data template;  
    set template;  
    length matrix $4 compound $60;  
  
    matrix = 'Air';  
compound = "Permethrin"; output;  
compound = "2,4-D"; output;
```

```

compound = "Dibutylphthalate"; output;
compound = "Bisphenol-A"; output;
compound = "Benzo[a]pyrene"; output;

matrix = 'Dust';
compound = "Permethrin"; output;
compound = "2,4-D"; output;
compound = "Dibutylphthalate"; output;
compound = "Bisphenol-A"; output;
compound = "Benzo[a]pyrene"; output;
run;

proc sort data=template;
    by matrix compound pid;
run;

*****;

proc sort data=set &in_file(where=(compound in ("2,4-D","Dibutylphthalate",
"Bisphenol-A", "Benzo[a]pyrene", "Permethrin") & samplotype= "Home" & matrix in
("Air", "Dust")))
    out=temp;
    by matrix compound pid detected;
run;

data temp0(keep = matrix compound pid detected);
    set temp;
    by matrix compound pid;
    if last.pid;
run;

proc means data=temp noprint;
    class matrix compound pid;
    var result qc_flag;
    output out=temp1 (where=(not(missing(matrix)) & not(missing(pid)) &
not(missing(compound))) drop = dflag dresult)
        mean = result dflag min = dresult qc_flag ;
run;

*****;

proc rank data=temp1 out = temp2 percent;
    by matrix compound;
    var result;
    ranks prt;
run;
*****;

data temp4;
    merge temp2
          temp0
          template;
    by matrix compound pid;

    if detected = 0 then prt = 5;
    else if prt > .z & prt < 5 then prt = 5;
    else if prt <= 55 & prt >= 45 then prt=50;

```

```

run;

proc sql noprint;
    select distinct(pid)
    into :pidlist separated by ' '
    from temp4 ;
quit;
run;

*****;
options reset=all ROTATE=PORTRAIT;
options ps=60;

data map;
    input section x y @@;
cards;
1 0 0 1 0 50 1 20 50 1 20 0
2 0 50 2 0 100 2 20 100 2 20 50
;
run;

data color;
    input section level;
cards;
1 1
2 20
;
run;

*****;
proc greplay igout=graphs.graph nofs;
quit;
run;

goption nodisplay;

proc gslide gout=graphs.graph name = "gslide1";
    note f = swiss h=15pct "2,4-D";
run;
quit;

proc gslide gout=graphs.graph name = "gslide2";
    note f = swiss h=15pct "Benzo[" f=swissi "a" f=swiss "]pyrene";
run;
quit;

proc gslide gout=graphs.graph name = "gslide3";
    note f = swiss h=15pct "Bisphenol-A";
run;
quit;

proc gslide gout=graphs.graph name = "gslide4";
    note f = swiss h=15pct "Dibutylphthalate";
run;
quit;

```

```

proc gslide gout=graphs.graph name = "gslide5";
    note f = swiss h=15pct "Permethrin";
run;
quit;

*****;
%macro loopit;
%let i_count = 1;
%let m_count = 1;
%let c_count = 1;

%do %until(%scan(&pidlist,&i_count)= );

proc greplay igout=graphs.graph nofs;
    delete "gslide" ;
    delete "Air_perm" ;
    delete "Air_2_4_" ;
    delete "Air_dibu" ;
    delete "Air_bisp" ;
    delete "Air_benz" ;
    delete "Dust_perm";
    delete "Dust_2_4_" ;
    delete "Dust_dibu";
    delete "Dust_bisp";
    delete "Dust_benz";
quit;
run;

    %let id = %scan(&pidlist,&i_count);

%let c_count = 1;

%do %until(%qscan(&chemlist, &c_count," ")= );
%let m_count = 1;
    %let chem = %qscan("&chemlist", &c_count," ");

%do %until(%scan(&medialist,&m_count)= );
    %let med = %scan(&medialist,&m_count);

*****;

data annomap;
    set temp4(where = (matrix = "&med" & pid = "&id" & compound = "&chem")
               keep = matrix compound pid prt );
length color function style $8 text $20 ;
retain xsys ysys '2' hsys '3' when 'a' position '6';
function='label'; style = 'swissb'; color='black'; size=5;

x = 20; y=5; text = ' 0'; output;
x = 20; y=51; text = ' 50'; output;
x = 20; y=100; position='C'; text = ' 100'; output;

if prt not = 50 & prt > .z then do;
    x = 0; position='4'; y=51;
    text='All Homes'|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
    x = 0; position='4'; y=49; size=10; style='marker'; text='J'; output;
    style = 'swissb'; size=5;

```

```

        x = 0; y=prt; position='4';
        text='Your Home'|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
        x = 0; position='4'; y=prt-2; size=10; style='marker'; text='J'; output;
style = 'swissb'; size=5;
end;

else if prt <= .z then do;
    x = 0; position='4'; y=51;
    text='All Homes'|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
    x = 0; position='4'; y=49; size=10; style='marker'; text='J'; output;
style = 'swissb'; size=5;
end;

else do;
    x = 0; position='A'; y=51;
    text="All Homes"|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
    x = 0; position='4'; y=49; size=10; style='marker'; text='J'; output;
style = 'swissb'; size=5;
    x = 0; y=prt; position='D';
    text="Your Home"|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x|| '00'x; output;
end;

run;

proc print data=annomap;
run;

proc gmap data=color map = map gout=graphs.graph;
    id section;
    choro level/ coutline = black nolegend anno =annomap name = "&med._&chem";
    pattern1 v=ms color = blue;
    pattern2 v=ms color = yellow;
    %if (&c_count = 1 ) %then
        title f = swissb h=6pct c=black "&med" ;
    %else
        title f = swissb h=6pct c=black " " ;
;
run;
quit;

title;
footnote;

%let m_count=%eval(&m_count+1);
%end;
%let c_count=%eval(&c_count+1);
%end;

proc gslide gout=graphs.graph name = "gslide";
    title f= swiss h=4pct j=l "Figure 1";
    footnote f= swiss h=.9pct j=r "&id";
run;

title;
footnote;

proc greplay nofs igout=graphs.graph gout=graphs.report;

```

```

tc templ.templs;
template newl2r5;
treplay      1:"Air_2_4_"
              2:"Dust_2_4_"
              3:"Air_benz"
              4:"Dust_benz"
              5:"Air_bisp"
              6:"Dust_bisp"
              7:"Air_dibu"
              8:"Dust_dibu"
              9:"Air_perm"
             10:"Dust_perm"
             11:gslide
             12:gslide1
             13:gslide2
             14:gslide3
             15:gslide4
             16:gslide5;

run;
quit;

proc greplay nofs igout=graphs.report;
    modify Template / name=fig&id;
run;
quit;

%let i_count=%eval(&i_count+1);

%end;

%mend;
*****;
%loopit;
*****;
*** End of program ***;

```

REFERENCES

Aster, R., *Professional SAS Programmer's Pocket Reference*, 5th edition, Paoli, PA: Breakfast Communication Corporation, 2004.

Powers, B., Rakes, S., Wagoner, B., and Waldbauer, E., *SAS Color Graphics Course Notes*, Cary, NC: SAS Institute Inc., 2002.

SAS Institute Inc., *SAS Guide to Macro Processing, Version 6, Second Edition*, Cary, NC: SAS Institute Inc., 1990.

DISCLAIMER

The United States Environmental Protection Agency through its Office of Research and Development funded and managed the research described here. It has been subjected to Agency administrative review and approved for publication. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

CONTACT INFORMATION

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

Carry Croghan
U.S. Environmental Protection Agency
MD-E205-04, 109 TW Alexander Drive
Research Triangle Park, NC 27709
Work Phone: 919 541 3184
Fax: 919 541 0905
Email: Croghan.Carry@EPA.gov

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.