ODS and Web Enabled Device Drivers: Displaying and Controlling Large Numbers of Graphs

Arthur L. Carpenter and Richard O. Smith
Data Explorations

ABSTRACT

With the advent of the Output Delivery System, ODS, it has become much easier to generate graphs that can be displayed online. ODS works with web enabled device drivers such as GIF, HTML, and WEBFRAME to create graphs that can be browsed and even interactively modified by the user. Coordination becomes problematic however, when a large number of graphs need to be displayed. Tools and techniques are available that permit the programmer to create hot zones that allow the user to drill down through graphs and index tables to find the graphs of interest. Coding techniques, including the use of macros to control the process, can be used to simplify the process of managing, naming, and locating large numbers of graphs. Although primarily directed to graphical displays of the data, many of these same techniques can be applied to reports and data summaries.

KEYWORDS

Output Delivery System, ODS, HTML, WEBFRAME, macro, GIF, GIFANIM

INTRODUCTION

The Output Delivery System, ODS, can be used to produce graphs that can be redisplayed through the use of standard internet browsers. In the production environment this often means the generation of a large number of graphs. It becomes problematic to locate, name, point to, and redisplay all of these graphs in such a way as to be fairly easy for the user and fairly automatic for the programmer.

In the production environment, the primary issue becomes one of management. The graphs and reports must be named and placed in an accessible location. When programs are rerun and graphs are updated or regenerated, the naming conventions must be well enough defined so that the correct graphs are replaced. The longer naming conventions of Version 8 make this easier, however constraints such as limitations within the naming of graphical catalog entries must be dealt with.

Once created, the graphs must then be placed in a location that is automatically determined within the production process, and if the location does not already exist, it must be created. This automated process requires a standardized and well thought out naming convention.

In some of the examples below, a series of survival analyses have been conducted for a variety of different models. Because each model will generate as many as 30 graphs and tables, the naming conventions and locations were determined to a large extent by a model designation code.

CONTROLLING LOCATIONS

The physical location for the various files must be controlled as part of the automated process. You can determine if a specific location (in this case a directory) exists through the use of the FILEEXIST function. Since for discussion purposes here we are operating in the macro environment, %SYSFUNC is used to call FILEEXIST, and if needed %SYSEXEC is used to create the new directory.

```
* Make sure that the hazardratios
* directory exists;
%let rc =
%sysfunc(fileexist("&drive\&project\hazardrat
ios"));
%if &rc=0 %then %do;
   %* Make the directory;
   %sysexec md &drive\&project\hazardratios;
%end;
```

Notice that the location of the directory and portions of the path are controlled by macro variables (&DRIVE and &PROJECT). These are set up by the application to make sure that all files and directories can be located by the application when it is time to retrieve them. These same macro variables are used whenever it is necessary to point to the primary or upper portion of any path within the application.

USING ODS TO CONTROL THE DESTINATION

HTML files will be generated through a variety of methods with an application. Depending on how they are built, these files can be used to point to other HTML files, GIF files, or other tables and graphs. Tables stored as HTML files can be used as indexes, which can be used to point to another level of graphs, tables, reports or even to another level of indexes. HTML files are also built by SAS/GRAPH drivers (primarily HTML and WEBFRAME device drivers) to 'wrap up' graphs in the form of GIF files.

The ODS HTML statement is used to name and point to these HTML files. The PATH= option designates the directory (notice the use of the same macro variables). The URL=NONE is used to build relative links within the HTML file. This allows you to move the files to another location such as a server. The BODY= option specifies the name of the file itself.

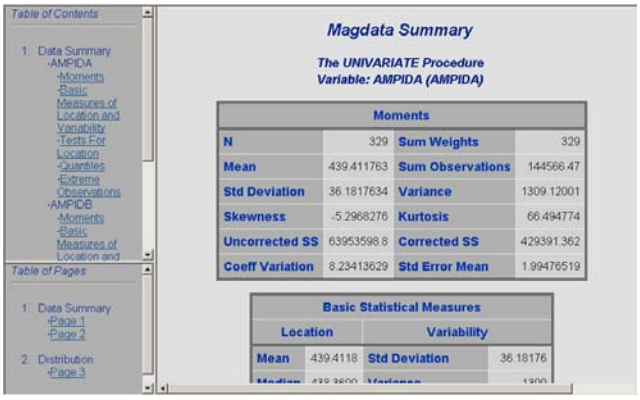
```
ods html path =
       "&drive\&project\hazardratios"
             (url=none)
          body =
       "HR_&&rptgrp&i.._&&hrgrp&i...html";
```

In this example, macro variables in the general form of &&VAR&I have been built based on values within the data. These macro variables in turn are used to determine the name of the HTML document. The ODS statement above resides inside of a macro %DO loop. After macro variable resolution, the name will resolve to something like 'HR_7_3.html'. By making the names of the files data value based, the program will not need modification when new report or data combinations are added. Discussion on how to build and control the macro variables can be found in a number of sources covering advanced macro programming techniques. These include: (Burlew, 1998), (Carpenter and Smith, 2002), and (Carpenter, 1998).

When a series of graphs or tables are to be added to an HTML file, additional options can be used to build a 'Table of Contents' for the various components. In the following example, the FRAME=, CONTENTS=, and PAGE= options are added to the ODS HTML statement.

```
* ODS HTML coordination using FRAME;
ods listing close;
ods html path     = "&drive\&project\tables\"
         body     = 'multiple.html'
         contents= 'multcontents.html'
         page     = 'multpage.html'
         frame    = 'multframe.html'
         (title='Magdata Analysis');

ods proclabel 'Data Summary';
proc univariate data=sasclass.magdata;
   var ampida ampidb;
   title1 'Magdata Summary';
   run;

ods proclabel 'Distribution';
proc gchart
data=sasclass.magdata(where=(ampida>100));
   hbar Ampida;
   title2 'Ampida Distribution';
   run;
   quit;
ods html close;
```

When the file named by the FRAME= option is browsed, the user sees a display such as the one shown below. This allows the individual objects created by the UNIVARIATE and GCHART procedures to be selected directly.

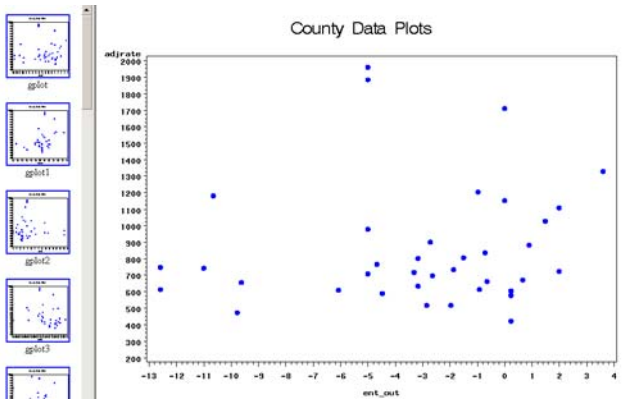


A number of ODS statements and options are available to enhance this type of output file. Of these, the ODS PROCLABEL statement and the TITLE= option are shown.

GRAPHICS DEVICES

Several of the newer graphics device drivers are especially appropriate when you are publishing to the web. Most of these produce GIF files, and indeed there is even a GIF device. The problem with building a series of individual GIF graphics files is that the user must browse them individually. The devices HTML and WEBFRAME solve this problem by wrapping up a series of GIF graphs in a single HTML document. Like one stop shopping the reader now needs to browse only one file to see all the graphs.

The WEBFRAME device creates two GIF files for each graph, full size and thumbnail. An HTML file is then created that 'wraps up' these various GIF files. The user needs to only browse the one HTML file to see any of the graphs and selection of one of the small thumbnail versions brings that graph up for display. A portion of a display generated with the WEBFRAME device is shown below.




```
filename regplt
        "&drive\&project\results\regsctr";

goptions device=webframe gsfname=regplt;
```

```
symbol1 c=blue v=dot r=45;
axis1 label=(a=90 f=simplex c=blue
             'Adjusted Rate');
title2 'County Data Plots';
proc gplot data=sasclass.cnty;
.... more code ....
```

To use this device, use a FILENAME statement to point to a directory (not to a file!). The device will then create a series of HTML and GIF files in this directory. One of these will be named INDEX.HTML, and this is the one that your application should have your user browse.

Obviously WEBFRAME is most useful when a series of graphs are to be displayed. One of its limitations, however, is that it cannot be used across PROC step boundaries. This means that you would be unable to combine the results of a PROC GPLOT with a PROC GCHART within the same display.

Fortunately we can get around this problem by utilizing NODISPLAY and saving all the graphs of interest as graphics entries (entry type of GRSEG) in a catalog. These graphs can then all be redisplayed using PROC GREPLAY in a single step. This process is described in detail in the following section.

USING WEBFRAME WITH GRSEG CATALOG ENTRIES

Whenever a SAS/GRAPH procedure creates a graph, an entry is also created in a catalog. Unless otherwise specified, the catalog is WORK.GSEG and the entry type is GRSEG. The name of the entry can be specified by the user, but otherwise it takes on the value of the name of the procedure that generated the graph. A PROC GPLOT, therefore would create an entry named WORK.GSEG.GPLOT.GRSEG. If a second plot is generated by GPLOT, the entry name becomes GPLOT1 and so on.

When the objective is to display a number of graphs from different procedures using a single WEBFRAME, each graph needs to be stored as a catalog entry. Since the storage is temporary, a WORK catalog can be used. If each graph in the work catalog is to be redisplayed, the catalog should be cleared first. The following PROC DATASETS will delete the catalog so that it can be recreated fresh.

```
*clear the default graph catalog;
proc datasets library=work mt=cat nolist;
  delete gseg nofs ;
  quit;
```

WEBFRAME utilizes the name of the graphics entry as the label for the thumbnail, so the default naming behavior is less than desirable, *e.g.* gplot, gplot1, gplot2, etc. Fortunately the entry name can be specified by the user through the use of the NAME and DESCRIPTION options. In the following example, one plot is generated for each call to GPLOT. The macro variables used in the NAME= option provide a unique name for the graph.

```
goptions nodisplay;
proc gplot data=pltdat ;
   plot hrplt*Q=pltvar/
       skipmiss nolegend
       vaxis=axis1
       name = "H&&cmod&i&&regim&i"
       des  = "&&model&i &&regim&i";
   run;
   quit;
```

Since there is no reason to display the graph at this time, the graphics option NODISPLAY is used. Rather the display is saved in a catalog for redisplay later when the WEBFRAME device can be used. Also notice that although Version 8 allows longer variable names, the catalog entry can still have no more than 8 characters, so care must be taken when constructing a value for the NAME= option. In the following step, all of the graphics entries with names starting with 'H&&cmod&i' are selected for plotting by placing a list of their names into the macro variable &SUMPLOTS.

```
* create macro containing list of the plots;
data _null_;
   set sashelp.vcatalg;
   where libname = 'WORK'
       and memname = 'GSEG'
       and objname =: "H&&cmod&i";
   length sumplots $20000;
   retain sumplots ' ';
   sumplots = trim(sumplots)||
                ' '||trim(objname);
   call symput('sumplots',
                    trim(left(sumplots)));
   run;
```

If the 8 characters in the name are not enough, the DESCRIPTION= option can also be used to store information. This information can also be used to determine subsets in much the same way as the name was used in the previous DATA _NULL_ step. In this example, the names of the selected plots are stored in a macro variable (&SUMPLOTS).

The procedure GREPLAY is used to redisplay the selected graphs. The REPLAY statement specifically names the plots to be redisplayed, in this case the list is stored in the macro variable &SUMPLOTS.

```
filename webloc
"&drive\&project\hazardratios\HR&&cmod&i";
 * Wrap the various graphs with html;
goptions dev=webframe display gsfname=webloc;
proc greplay igout=gseg nofs;
   replay &sumplots;
   run;
   quit;
```

Remember that among all the files generated within the designated directory, there will always be one named INDEX.HTML. Have your user browse this file to see all

of the individual graphs and thumbnails.

BUILDING AN INDEX WITH PROC PRINT

In the previous example, a series of graphs were built using the WEBFRAME device. Each execution of the GREPLAY wrote another series of graphs to another directory. The user can now display all of the graphs in a given subdirectory simply by going to that directory, finding the 'index.html' file, and browsing it. This is still quite a challenge if there are a lot of directories and/or if there are many graphs within any given directory. What we next need is a single file that, when browsed, will direct the user to the correct index file / directory combination by a simple click of the mouse. We will create this file with a simple PROC PRINT.

The data set to be printed must contain a variable whose value is a valid HTML anchor tag that references the name of another file. The form of this link (GRPREF in the following DATA step) includes the value to be displayed (the report group in this case which is stored in CRPTGRP), and the name/location of the file to branch to branch to (the link is stored in the variable MYLINK).

```
data prep2;
   set prep1;
   length grpref mylink $150 crptgrp $3;
   *rptgrp+1;
   crptgrp= trim(left(put(rptgrp,3.)));
   myLink = "\\&project\hazardratios\hr"||
            trim(crptgrp)||'\index.html';
   grpref =  "<a HREF="||trim(myLink)||
             '>'||trim(crptgrp)||'</a>'
   label grpref = 'Report Group';
   run;

* Define location and index name for the
* new index;
ods html path="&drive\&project\&outloc"
             (url=none)
         body="&indexname..html";
%put going to "&drive\&project\&outloc";
ods listing close;
proc print data=prep2 label noobs;
   var grpref groupdesc;
   title1 "&title  Control File: &indsn";
   run;
ods html close;
```

When the file generated by the PROC PRINT is displayed, only the portion of the variable GRPREF that came from the variable CRPTGRP will be displayed. When the user selects one of these values, the corresponding INDEX.HTML file, which was built by WEBFRAME, will be displayed.

In a more complex study or analysis presenting more than one level of PROC PRINT indexes may be needed. Perhaps the first level selects general types of reports or general areas of interest. A selection here then points to a secondary index that allows the reader to further select the set of graphs of interest. Effectively this process creates a Table of Contents that passes the reader directly to the graphs of interest.

CREATING DRILL-DOWN GRAPHS AND CHARTS

Not only do we want to be able to use the tools discussed above to locate and display a specific graph, we may also want to use the graph itself to point to more information. As we browse the graph created using the techniques shown above, we want to be able to create 'hot zones' that allow us to drill down through a portion of a graph by clicking on a bar, line, or symbol, so that we can then display and browse another related graph or table.

In the example, below both a histogram and a scatter plot are generated as GIF files. Each of these graphs contains information that is STATION specific. When browsing, if the cursor is moved over a station specific portion of the graph, a 'hot zone', it changes to a pointer. The pointer indicates that clicking on that hot zone will cause a branch to another file or graph. In this case the branch will be to a table created by PROC PRINT of that station's data.

First we create a PROC PRINT for each value of the variable STATION. Notice that the name of the HTML file, the TITLE, and the value of the variable STATION in the WHERE clause all contain the name of the station (AZU in this example).

```
* List the AZUSA data;
ods html path="&drive\&project\figures"
                      (url=none)
          body='azu.html';
proc print data=sasclass.ca88air
              (where=(station='AZU'))
             noobs;
   var month co o3 no3 tem;
   title1 'AZU Pollution';
   footnote;
   run;
ods html close;
```

In the production environment it is likely that the list of stations will be stored in a macro array. The PROC PRINT can then be placed in a %DO loop. This allows the automated generation of the secondary files as shown below.

```
%do i = 1 %to &stacnt;
* List the &&sta&i data;
ods html path="&drive\&project\figures"
             (url=none)
          body="&&sta&i...html";
proc print data=sasclass.ca88air
       (where=(station="&&sta&i"))
       noobs;
   var month co o3 no3 tem;
   title1 "&&sta&i Pollution";
   footnote;
   run;
ods html close;
%end;
```

The key to pointing to the secondary files (created by PROC

PRINT above) is a character variable that contains a reference file pointer. The form of the value of this variable will be "href=*xxxxxxx*.html", where *xxxxxxx* is data dependent (in this case the station name).

In the data set to be plotted, we create a variable, DRILLSTA, which contains the name of the station as part of the file pointer.

```
data ca88air;
   set sasclass.ca88air;
   length drillsta $15;
   drillsta = 'href='||trim(left(station))||
              '.html';
   run;
```

This variable is utilized by the PROC GCHART in the VBAR statement through the HTML= option. Notice also that we are using the GIF device, although other devices such as WEBFRAME, which was shown above, could also be used.

```
goptions device=gif;
ods html path="&drive\&project\figures"
                    (url=none)
         body='chart.html';
PROC GCHART DATA=ca88air;
    VBAR station  / type=mean sumvar=o3
                    patternid=midpoint
                    subgroup=station
                    html=drillsta
                    raxis=axis1 maxis=axis2;
    run;
    quit;
ods html close;
```

For reasons that are not entirely clear to us, the SUBGROUP= option is also sometimes required with VBAR and HBAR chart types in order to make use of the HTML= option. This is true for this chart even though the SUBGROUP= option is not otherwise needed.

It is also possible to create hot zones in a scatter plot. In the following example, the same data that was used above is also plotted. In this case there is one line per station. Because the DRILLSTA variable is a constant for each station, each line becomes a separate hot zone.

```
goptions device=gif;
ods html path="&drive\&project\figures"
             (url=none)
         body='gplot.html';
proc gplot DATA=ca88air;
    plot o3*month=station /
              html=drillsta
              htmllegend=drillsta
              vaxis=axis1;
    run;
    quit;
ods html close;
```

The HTMLLEGEND= option creates hot zones in the legend as well. In this example, the variable DRILLSTA is constant for each station. If instead it had been unique for each plotted point, then each point could have been a separate hot zone.

USING NON-STATIC DRIVERS

Device drivers such as ACTIVEX and JAVA allow you to create graphs that are not static. These graphs can be manipulated by the user of the graph as it is browsed.

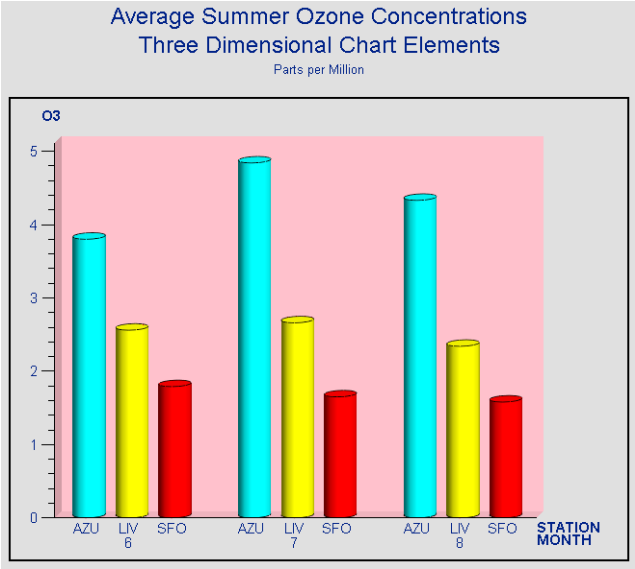
ACTIVEX

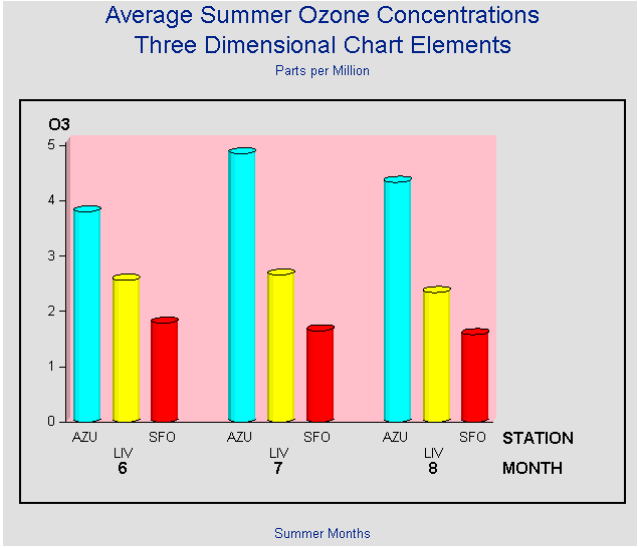
The ACTIVEX driver allows the developer to build graphs that can be changed and adapted by the user outside of SAS/GRAPH. Unlike the GIF, WEBFRAME, and HTML devices this device creates an HTML file, not a GIF file.

In order to control the size of the graph and to prevent it from not fitting on the browser screen, the XPIXELS= and YPIXELS= options are often needed. The values taken on by these two options will depend on the display resolution used by the person browsing the resulting HTML file.

```
goptions reset=all border ftext=simplex;
goptions device=activex
         xpixels = 640
         ypixels = 480;

pattern1  c=cyan;
pattern2  c=yellow;
pattern3  c=red;

ods html
body="&drive\&project\figures\SummerO3.html";

PROC GCHART DATA=sasclass.ca88air;
    where month in(6 7 8);
    VBAR3D station  / type=mean sumvar=o3
                      shape=cylinder
                      cframe=pink
                      group=month
                      patternid=midpoint;
    TITLE1 'Average Summer Ozone
Concentrations';
    title2 'Three Dimensional Chart
Elements';
    title3 a=90 f=simplex h=1 'Parts per
Million';
    footnote1 h=1 f=simplex 'Summer Months';
    run;
    quit;
ods html close
```

This code generates the following graph.





JAVA

The JAVA driver builds the JAVA components so that when the HTML file is browsed, the appropriate JAVA Applets construct the graph. Like the ACTIVEX graphs, these graphs are not static to the user and can be reshaped, resized, and otherwise greatly modified. As in the example for the ACTIVEX device, notice the use of the XPIXELS= and YPIXELS= options to control the initial size of the graph.

The PARAMETERS= option in the ODS HTML statement is used to place parameters and associated values into the JAVA code. In this case, the DRILLDOWNMODE parameter receives a value of LOCAL.

```
goptions reset=all border ftext=simplex;
goptions device=java
         xpixels = 600
         ypixels = 600;

pattern1  c=cyan;
pattern2  c=yellow;
pattern3  c=red;

ods html file=
  "&drive\&project\figures\SummerMonth.html"
       parameters=("DRILLDOWNMODE"="LOCAL");

PROC GCHART DATA=sasclass.ca88air;
    where month in(6 7 8);
    VBAR3D station  / type=mean sumvar=o3
                      shape=cylinder
                      cframe=pink
                      group=month
                      patternid=midpoint;
    TITLE1 'Average Summer Ozone
Concentrations';
    title2 'Three Dimensional Chart
Elements';
    title3 a=90 f=simplex h=1 'Parts per
Million';
    footnote1 h=1 f=simplex 'Summer Months';
    run;
    quit;
ods html close;
```

ANIMATED GIF GRAPHS

The device GIFANIM can be used to create a graph, which when viewed, appears to be animated. In fact, a series of distinct graphs are produced and then displayed in succession. This might show a process through time or a plot from different perspectives.

This driver is a bit more advanced than the others and requires a bit more care on the part of the programmer. Although a single GIF file is created, internally it is composed of three parts that enable the animation process. The parts of the file are built through the use of the GSFMODE option and a short DATA step.

The GIF file is named using a FILENAME statement, and the resulting *fileref* is tied to the graph through the use of the GSFNAME option. First you should initialize the file with a GSFMODE=REPLACE. Then each graph produced by successive procedures, BY statements, and graph definitions results in an animated image. After the first procedure, you will need to change the GSFMODE to APPEND. When the last graph has been added to the file, a short DATA step is executed to add a character to 'close' the file.

```
* Name the Animation file (GIF);
filename animate
"&drive\&project\figures\animate.gif";

goptions reset=all;
goptions dev=gifanim
         gsfmode=replace
         gsfname=animate
         iteration=1
         delay=60;

proc sort data=sasclass.ca88air
           out=ca88air;
   by month station;
   run;
```

```
* One Bar chart for each of 12 months;
title h=1 'Average Ozone';
axis1 order=(0 to 6 by 2)
      label=none;
proc gchart data=ca88air;
   by month;
   vbar station / sumvar=o3 type=mean
                  raxis=axis1;
   run;

* Prepare for another set of graphs;
goptions gsfmode=append;

* One Bar chart for each station;
proc sort data=ca88air;
   by station month;
   run;
proc gchart data=ca88air;
   by station;
   vbar month/ sumvar=o3 type=mean
               raxis=axis1;
   run;

* Complete the Animation file;
data _null_;
   file animate recfm=n mod;
   put '3B'x;
run;

* Reset the options;
goptions reset=all;
```

The ITERATION= option is supposed to be used to control the number of times that the animation is to loop through the file. This option does not always work as advertised, and all values (not just 0) often result in continuous loops. The amount of time that each image is to be displayed is controlled by the DELAY option, which takes on values in tenths of seconds. You will need to experiment with this value as the actual time of display will vary due to a number of factors (and may not even be consistent among the images).

SUMMARY

The production and tracking of large numbers of graphs that are interrelated with tables and other data displays can be difficult in the production environment. Not only is coordination required between linked graphs and tables, but just locating and pointing to the myriad of combinations can be problematic. Through the use of index tables and graphs with drill-down hot zones, it is possible to set up an environment that is much more easily navigated by the user.

Macros and strict naming conventions can be used by the developer to control the names of the various graphs and files. This allows a degree of control for the entire process.

A number of graphics device drivers have been developed which enable the programmer to create a variety of types of graphs that can be browsed by the user. When these are coupled with ODS and controlled through the use of macros, a very robust and expandable system can be created.

Tie these techniques together and you can generate a series of interconnected tables and graphs in an automated user friendly environment.

REFERENCES

Bessler, LeRoy and Francesca Pierri, 2002, "%TREND: A Macro to Produce Maximally Informative Trend Charts with SAS/GRAPH ®, SAS®, and ODS for the Web or Hardcopy", Proceedings of the Twenty-Seventh Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., paper 93.

Burlew, Michele M., 1998, *SAS® Macro Programming Made Easy*, Cary, NC: SAS Institute, Inc., pp. 280.

Carpenter, Arthur L., 1998, *Carpenter's Complete Guide to the SAS Macro Language*, Cary, NC: SAS Institute Inc., pp. 242.

Carpenter, Arthur L. and Richard O. Smith, 2002, "Library and File Management: Building a Dynamic Application", Proceedings of the Twenty-Seventh Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., paper 21.

Watts, Perry, 2002, *Multiple Plot Displays: Simplified with Macros*, Cary, NC: SAS Institute Inc., pp.125.

AUTHORS

Richard Smith and Art Carpenter are senior partners at Data Explorations, a SAS Alliance Silver Member™. Both are SAS Certified Professionals™ and they provide data management, analysis, and SAS programming services nationwide.

Arthur L. Carpenter
Art Carpenter's publications list includes three books on SAS topics (*Annotate: Simply the Basics*, *Quick Results with SAS/GRAPH® Software*, and *Carpenter's Complete Guide to the SAS® Macro Language*), two chapters in *Reporting from the Field*, and numerous papers and posters presented at various user group conferences. Art has been using SAS since 1976 and has served in a variety of positions in user groups at the local, regional, and national level.

Richard O. Smith
Richard Smith has a masters in Biology/Ecology and has provided complete data management and analysis services for numerous environmental research projects as a senior biologist, SAS programmer, and project manager. He has also provided programming and management services for the health related industries. He has been using SAS extensively since 1981.

AUTHOR CONTACT

Data Explorations

2270 Camino Vida Roble, Suite L
Carlsbad, CA 92009

Arthur L. Carpenter
(760) 945-0613
art@caloxy.com

Richard O. Smith
(760) 438-1336
ROSmith@SciX.com