Paper 186-25

Producing Interactive Internet Presentations in SAS® Software

Iza Peszek, Merck & Co., Inc., Rahway, NJ

ABSTRACT

The ODS in SAS v.7 and higher allows users to create HTML files with drill-down functionality. In particular, users can create graphs with “clickable” areas which link to other HTML documents. However, there seem to be some difficulty in getting this to work with PROC GPLOT. We will show how to achieve this functionality – and add more spice– without relying on ODS.

The method presented here creates a series of web-capable displays. For simplicity, we concentrate on 2-level layout: the “main page” and the “child pages”. The “main page” has an embedded interactive graph: when the mouse is paused over a point on the graph, a window pops-up with some information about the point. If user clicks on the point, a “child page” is displayed. This is accomplished via SAS macro, which creates an HTML document with embedded JavaScript. The “child pages” can contain graphics (created via SAS/GRAPH®) or text (output from SAS procedures). The appearance of “child pages” can be enhanced if they are produced in HTML format, either via SAS macro or using ODS feature of SAS 7 or 8. The intended audience should have at least some familiarity with SAS GRAPH software.

INTRODUCTION

The ODS feature in SAS v.7 and later allows users to create HTML files with drill-down functionality. In particular, users can create graphs with “clickable” areas which link to other HTML documents. However, there seem to be some difficulty in getting this to work with densely populated graphs produced by PROC GPLOT. We will show how to prodice drill-down displays – and make them more spicy – without relying on ODS. In fact, the method was developed for SAS v.6.12 and higher. To use this method, your browser must support JavaScript and document layers. Both Netscape 4 and Internet Explorer 4 satisfy these requirements.

The method presented here creates a series of web-capable displays. For simplicity, let us concentrate on 2-level layout: the “main page” and the “child pages”. The “main page” has an embedded interactive graph: when the mouse is paused over a point on the graph, a window pops-up with some information about the point. If user clicks on the point, a “child page” is displayed. The “child page” does not have to contain a graph; it can be a text-based display as well. Of course, you can have several levels of drill-down functionality. If you want the “child pages” to be clickable also, you would produce them in the same way as the main graph.

To produce such an interactive display, you need to follow the following steps:

1. Use PROC GPLOT to create a main graph in GIF format. There are specific requirements for GOPTIONS and AXIS statements; they are explained later in the article. Create two additional variables in the data set used with PROC GPLOT: one with a description of the data point (to appear in the pop-up window), and the other with the file name for the “child page” for the data point. You also need to "translate" the coordinates of the plotted data points into the language which web browser can understand.
2. Create one “child page” for each point on the main graph. The “child pages” can be in any format understood by your browser. The simplest examples include ASCII files, html files, or GIF images.
3. Invoke our macro DOHTML.sas to link the “child pages” to the main graph. This macro will need to access the data set which you prepared in step 1. It will create an html page with embedded JavaScript code, making your display come alive.

Voila! Your web display is ready to use!

Now we present the details of each step.

PRODUCING A CLICKABLE PLOT

The clickable plot has to be produced in GIF format. Later you will need to express the coordinates of each point on the graph as pixels in the GIF file. These pixel coordinates will be used to define so-called "image map". The areas in the image map are interpreted by web browser; we can assign hyperlinks to different areas in the image map.

First of all, you need to specify the goptions vorigin, horigin, vsize, hsize and gunit:

```
goptions
…
device=gif733
rotate=landscape
vorigin=0 in
horigin=0 in
vsize=0 in
hsize=0 in
gunit=pct;
```

To calculate pixel coordinates, the axes have to be positioned very precisely, with specified origin, offset, order and length. We make use of macro variables, so you can see later how these specifications are used to “rescale” the coordinates of the plotted points.

```
axis1
…
origin=(&xorigin pct &yorigin pct)
offset=(0 pct 0 pct)
order = &miny to &maxy by &stepy
length=&ylength pct ;

axis2
…
offset=(0 pct 0 pct)
length=&xlenght pct
order = &minx to &maxx by &stepx;
```

It is important that the offsets for the axes are equal to zero in both directions. If you use different offset, you will need to adjust the formula given in the next section.

PREPARING THE DATA SET TO DEFINE TARGETS FOR USER ACTIONS

In this step you need to manipulate the data set you used in

Step 1. Assume that each record in the data set represents a point on the graph. First, you need to create a description to appear in the pop-up window when the mouse pauses over a point on the main graph. For example, define

```
pt_desc="Age:"||put(age,2.)||"<br>Gender:
"||sex||"<br>Race: "||race;
```

The tag "<br>" is used to create a carriage return in the pop-up display.

Next, define the URL for the "child page". You will need to name your "child pages" carefully, because you need to link each record to the correct "child page". If "child pages" are located in the same folder as the "main page", you only need to specify the filename of the "child page"; otherwise, use the relative path:

```
pt_url = "file"||put(pat_id, 4.)||".gif";

or

pt_url = "childs/file"||
put(pat_id, 4.)||".gif";
```

In this examples, we used patient id (pat_id) as an identification of the record. Later, we can use the same variable to name our "child pages".

Finally, translate the coordinates of the points plotted on the main graph into pixel language. The formula below assumes that you used device=GIF733 and that the offset for both axes is (0 pct, 0 pct). If your offset is different, you need to adjust the formula accordingly. Note that if you used gif733 device, the upper left corner of your graph has pixel coordinates (0,0) and the lower right corner has pixel coordinates (733,550):

```
xpixel= &xlength*(xvar - &minx)/(&maxx-
&minx)+&xorigin;
ypixel= &ylength*(yvar- &miny)/(&maxy-
&miny)+&yorigin;
* point position as percentage of the
graphic area;
xpixel = round(7.33*(xpixel),1);
ypixel = round(5.50*(100-ypixel),1);
* rescale to get pixels;
```

You can see now why you had to carefully define the exact position and length of the axes. This allowed you to express the position of the plotted points in terms of the percentage of the graphic area, and then to re-scale to get pixel coordinates.

PRODUCING "CHILD PAGES"

If your drill-down displays are to be static (non-interactive), you need only to produce them in the format which web browser can understand. If you are satisfied with plain-vanilla textual display, you can save the output of any SAS procedure in ASCII or HTML format. However, users of SAS v.7 or 8 can make the child pages much more attractive using ODS and PROC TEMPLATE. More round-about approach creates HTML files using DATA _NULL_ and PUT statement to add HTML tags. If the child pages contain only graphics, it suffices to produce them in GIF format. Modern browsers, like Netscape 4 or Internet Explorer 4, can display GIF files directly, so you do not have to embed them in HTML documents.

You can also make your "child pages" clickable if you create them in the same way as the main graph and use the macro described in the next section.

It is very important to name your child pages in a meaningful way, so you can tell which point on the main graph corresponds to which "child page". For example, you can use patient number (pat_id) in the file names as we did in our examples.

CREATING AN HTML FILE WITH CLICKABLE GRAPH

The final step calls macro DOHTML.SAS to create the HTML file with the graph produced in Step 1. A sample call is given below:

```
 %dohtml (
device=gif733,
plotname=theplot.gif,
datain=mydata,
htmlname=c:\html_test\theplot.htm,
title=The main page,
url_var=pt_url,
desc_var=pt_desc,
xpixels=xpixels,
ypixels=ypixels,
radius=2) ;
```

In this call, it is assumed that all files you created are placed in the same folder, c:\html_test. The file theplot.gif is the graph you produced in Step 1. The file theplot.htm will be created by the macro.
The data set mydata contains the targets for user actions in the form of the variables pt_url, pt_desc, xpixels and ypixels. The macro variable radius needs a little explanation. The clickable "areas" on the graph are circles centered around the point on the graph and the variable radius defines the radius of these circles. You need to take care so that the clickable areas do not overlap. With little work, you can calculate the value of the variable radius programmatically.

The code for the macro %dohtml is given in the appendix. This macro uses DATA _null_ to print HTML tags and other elements of the HTML document. This particular version prints embedded JavaScript code with some fairly simple functions which take care of the "pop-up" windows. The html tags insert the graph you created and define an image map for this graph. The areas in the image map correspond to the plotted points and their targets are the "child pages" you linked to each point.

It is quite easy to substitute your own JavaScript functions for the ones we used. You can design some very elaborated responses to user actions according to you imagination and needs.

CONCLUSION

The method described here requires some extra programming effort, since you need to calculate the pixel coordinates of the points on your graph. We do not intend to compete with ODS, which can help you produce interactive displays with much less effort. Our purpose is to fill the gaps in ODS and to show how you can extend the interactivity of your presentation beyond the limits of ODS. We encourage you to capitalize on all available tools: use ODS when it is convenient to easily produce attractive, static pages and use the proposed method to add more sophisticated features.

TRADEMARK INFORMATION

SAS, SAS/GRAPH are registered trademarks or trademarks

of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

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

Iza Peszek
Merck & Co., Inc.
Clinical Biostatistics, P.O. Box 2000
Rahway NJ 07065
Work Phone: (732) 594 3623
Fax: (732) 594 6075
Email: izabella_peszek@merck.com

APPENDIX

The code for macro %DOHTML:

```
%macro dohtml (
device=gif733,
plotname=theplot.gif,
datain=,
htmlname=c:\theplot.htm,
title=,
url_var=,
desc_var=,
xpixels=xpixels,
ypixels=ypixels,
radius=2) ;

/*----------------------------------------

this macro produces an html file with a
clickable graph;

Macro parameters:
device
  graphic device. Either gif733 of
gif570.
plotname
  name of the graph file, without quotes.
If graph             is in the same
folder as html file,
   no need to add path. Can use relative
path (e.g., plotname=../images/plot.gif)
datain
  sas data set used to produce graph
&plotname
htmlname
  name and path of the html file, without
quotes
title
  title to show in browser bar, without
quotes
url_var
  variable with url for drill-down
display
desc_var
  variable with description to show in
text area
xpixels
  variable with pixel coordinate (x) for
a point
ypixels
  variable with pixel coordinate (y) for
a point
radius
  length of the radius of the clickable
area
----------------------------------------*/
%local gifx gify qt pixels;

%let gifx=733;
%let gify=550;

%let qt = %str(%");
%let pixels=570;

%if %index(&device, &pixels) >0 %then %do;
  %let gifx=570;
  %let gify=429;
%end;
%put gifx=&gifx gify=&gify;

data _null_;
file "&htmlname";
put "<HTML>";
  put "<HEAD>" ;
  put "<TITLE>&title</TITLE>" ;
  put "<SCRIPT
LANGUAGE=&qt.JavaScript&qt.>";
  put "<!-- begin hiding";
  put "var PointDesc = new Array();";
run;

data _null_;
file "&htmlname" mod;
set &datain;
put "PointDesc[" _n_ "]= &qt." &desc_var
"&qt.";
run;

data _null_;
file "&htmlname" mod;
set &datain end=eof;
if _n_=1 then do;
  put "var
isNS4=(navigator.appName.indexOf(&qt.Netsc
ape&qt.)!= -1)";
  put "function clearEl() {}";
  put "function init() {";
  put "setTimeout(&qt.window.onresize =
redo&qt., 1000);}";
  put "function redo()
{window.location.reload();}";
  put "function makeEl(id, width, code)
{";
  put "if (!style) return;";
  put "var str = &qt.<STYLE
TYPE='text/css'>&qt.;";
  put "str += &qt.#&qt. + id + &qt.
{&qt.;";
  put "str += &qt.width: &qt. + width +
&qt.;&qt.;";
  put "str += &qt.}&qt.;";
  put "str += &qt.</STYLE>&qt.;";
  put "str += &qt.<DIV CLASS='tooltip'
ID='&qt. + id + &qt.'>&qt. + code +
&qt.</DIV>&qt.;";
  put "document.write(str);}";
  put "function displayEl(left, top) {";
  put "if (NS4)
{document.releaseEvents(Event.MOUSEMOVE)};
";
  put "document.onmousemove = null;";
  put "var whichEl = (NS4) ?
document[active] :
document.all[active].style;";
  put "whichEl.left = left;";
  put "whichEl.top = top;";
  put "whichEl.visibility = (NS4) ?
&qt.show&qt. : &qt.visible&qt.;}";
  put "function clearEl() {";
  put "if (!style) return;";
  put "var whichEl = (NS4) ?
document[active] :
document.all[active].style;";
  put "whichEl.visibility = (NS4) ?
&qt.hide&qt. : &qt.hidden&qt.;";
  put "active = null;";
  put "if (timerID)
clearTimeout(timerID);";
  put " if (NS4)
document.releaseEvents(Event.MOUSEMOVE);"
;
  put "document.onmousemove = null;}";

  put "function activateEl(id, e) {";
  put "if (!style) return;";
  put "active = id;";

  put "if (NS4)
document.captureEvents(Event.MOUSEMOVE);"
;
  put "document.onmousemove = checkEl;";
  put "checkEl(e);}";
  put "function checkEl(e) {";
  put "if (timerID)
clearTimeout(timerID);";
  put "var left = (NS4) ? e.pageX :
event.clientX +
document.body.scrollLeft;";
  put "var top = (NS4) ? e.pageY + 20 :
event.clientY + document.body.scrollTop +
20;";
  put "timerID =
setTimeout(&qt.displayEl(&qt. + left +
&qt., &qt. + top + &qt.)&qt., 300);}";

  put "function make_el(id)
{makeEl(&qt.aaa&qt. + id+&qt. &qt., 300,
PointDesc[id])}";

  put "var timerID=null;";
  put "var NS4 = (document.layers) ? 1 :
0;";
  put "var IE4 = (document.all) ? 1 :
0;";
  put "// end -->";
  put "</SCRIPT>";

  put "</HEAD>";
  put "<BODY>";
  put "<p><span ID=&qt.test&qt.
STYLE=&qt.position:
absolute;&qt.></span></p>";
* load image;
  put "<IMG SRC = &qt.&plotname.&qt.
WIDTH=&gifx. HEIGHT=&gify.
border=&qt.0&qt USEMAP=&qt.#graph&qt. >";
  put "<MAP NAME=&qt.graph&qt.>";
end;
* body statements;
put "<AREA HREF=&qt." &url_var "&qt.";
put "COORDS=&qt." &xpixels. "," &ypixels.
", &radius.&qt. SHAPE=&qt.circle&qt.";
put "onMouseOver=&qt.if(NS4||IE4)
{activateEl('aaa" _n_ "',event)}; return
true&qt.";
put "OnMouseOut=&qt.clearEl(); return
true;&qt.>";

if eof then do;
  put "</MAP>";
  put "<script
LANGUAGE=&qt.JavaScript1.1&qt.>";
  put "<!-- ";
  put "var style = ((NS4 &" "&
document.test) || IE4) ? 1 : 0;";
  put "var timerID = null;";
  put "var padding = 3;";
  put "var bgcolor = &qt.beige&qt.;";
  put "var borWid = 1;";
  put "var borCol = &qt.#0000cc&qt.;";
  put "var borSty = &qt.solid&qt.;";
  put "var str = &qt.<STYLE
TYPE='text/css'>&qt.;";
  put "str += &qt..tooltip {position:
absolute;visibility: hidden;&qt.;";
  put "str += &qt.left: 0; top: 0;&qt.;";
  put "if (borWid > 0) { ";
  put "str += &qt.border-width: &qt. +
borWid + &qt.;&qt.;";
  put "str += &qt.border-color: &qt. +
borCol + &qt.;&qt.;";
  put "str += &qt.border-style: &qt. +
borSty + &qt.;&qt.;}";
  put "if (NS4) {";
  put "if (borWid > 0 &" "& padding <= 3)
{";
  put "str += &qt.padding: 0;&qt.;";
  put "str += &qt.layer-background-color:
&qt. + bgcolor + &qt.;&qt.;";
  put "} else if (borWid > 0 &" "& padding
> 3) {";
  put "  str += &qt.padding: &qt. +
(padding - 3) + &qt.;&qt.;";
  put "  str += &qt.background-color: &qt.
+ bgcolor + &qt.;&qt.;";
  put "} else if (borWid == 0) {";
  put "  str += &qt.padding: &qt. +
padding + &qt.;&qt.;";
  put "  str += &qt.layer-background-
color: &qt. + bgcolor + &qt.;&qt.;}";
  put "} else {";
  put "str += &qt.padding: &qt. + padding
+ &qt.;&qt.;";
  put "str += &qt.background-color: &qt. +
bgcolor + &qt.;&qt.;}";
  put "str += &qt.}</STYLE>&qt.;";
  put "if (style) {";
  put "document.write(str);";
  put "if (NS4) window.onload = init;}";
end;
run;

data _null_;
file "&htmlname" mod;
set &datain end=eof;

put "make_el(" _n_  ");";
if eof then do;
  put "// end -->";
  put "</script>";
  put "</BODY>";
  put "</HTML>";
end;
run;
quit;

%mend;
```

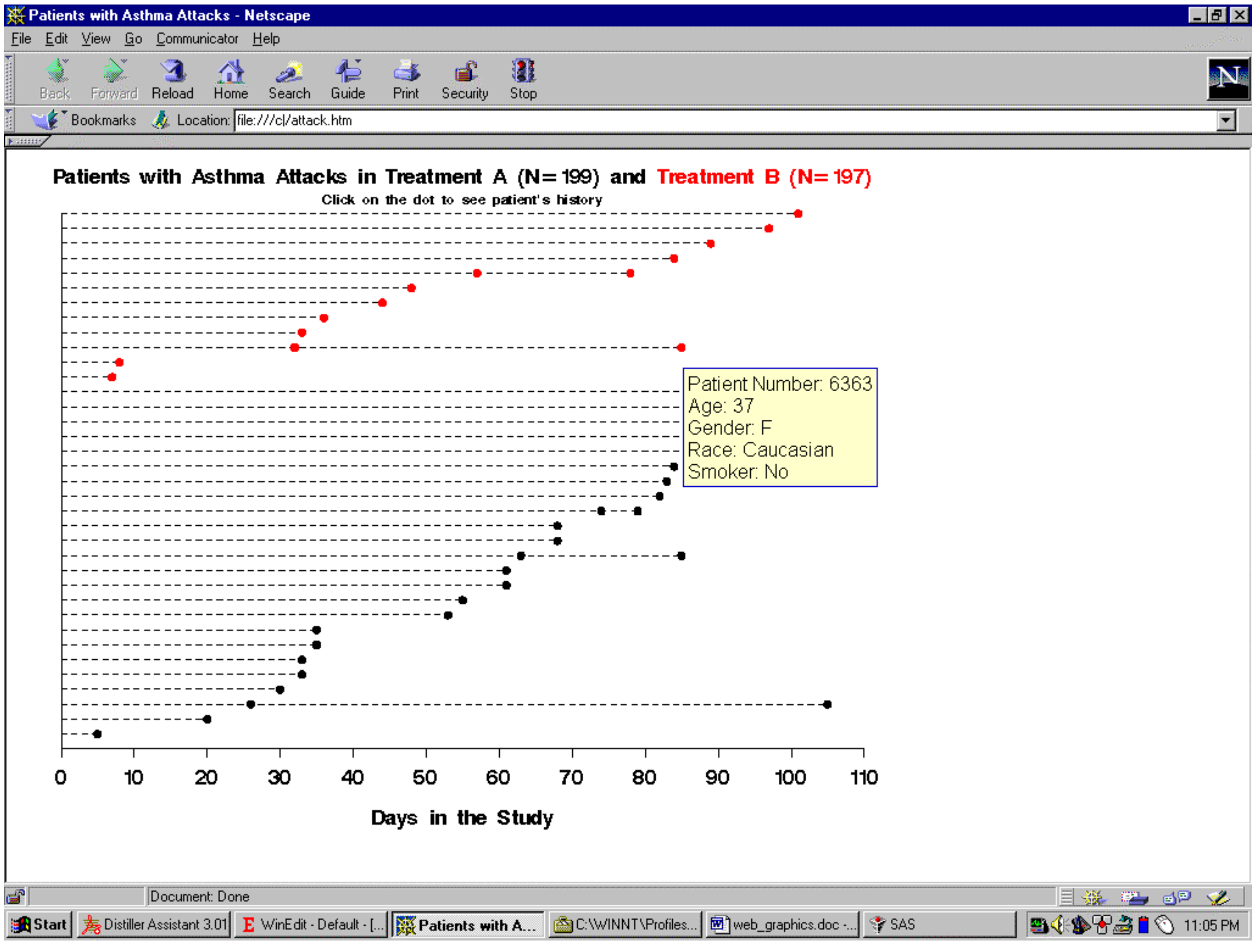
Patients with Asthma Attacks - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: file:///cl/attack.htm
Patients with Asthma Attacks in Treatment A (N=199) and Treatment B (N=197)
Click on the dot to see patient's history
Patient Number: 6363
Age: 37
Gender: F
Race: Caucasian
Smoker: No
0 10 20 30 40 50 60 70 80 90 100 110
Days in the Study
Document: Done
Start Distiller Assistant 3.01 WinEdit - Default - [... Patients with A... C:\WINNT\Profiles... web_graphics.doc -... SAS 11:05 PM

Figure 1: A snapshot of a pop-up window which appears when the cursor pauses over a point