A Dynamic Imagemap Generator

Carol Martell, Highway Safety Research Center, Chapel Hill, NC

ABSTRACT

We learned that our web developers were turning a picture of the state of North Carolina with its one hundred counties into an imagemap by manually outlining each county. Ouch! SAS/GRAPH® does a lovely job via SAS/INTRNET®. The web developers kept changing their minds about the size of the image, so I wrote a little SAS® job that uses Application Dispatcher to return the imagemap to the browser. The user supplies various parameters in a form and in return gets a link to the generated imagemap. The user can then 'save as' and 'save image as' to get both pieces for placement elsewhere. The application is generalized to generate a map of any state.

INTRODUCTION

An imagemap, once obtained, cannot be correctly resized using today's GUI-based html editors. One can resize the gif image, but the map coordinates do not change accordingly. This application makes imagemap recreation a breeze. Beginning with interactively generating a simple map of North Carolina, this paper steps through the process of moving from the interactive environment to the web. At each new step, additional code is shown in bold typeface. Since the intention is that the output generated be incorporated into other html, the map is purposefully devoid of legend, title, footnotes, etc. This paper assumes familiarity with running Application Dispatcher.

PREPARATION

The first step is to prepare input data for PROC GMAP: a map dataset and the data to be mapped. SAS provides various map datasets, one of which is Counties, an unprojected map dataset. The state and county variables in the table are fips codes. SAS has several functions for manipulating fips state codes. We create a new map dataset by subsetting for North Carolina records:

```
data two;
  set maps.counties;
  if state=stfips("NC");
  run;
```

Since Counties is unprojected, we run PROC GPROJECT to ensure the map will be undistorted.

```
proc gproject data=two out=one;
  id county;
  run;
```

The SAS table containing data to be mapped must contain the id variable to link with the map dataset. In this case id is county. GMAP also requires a variable to be mapped onto the state. We call it mapv and assign the value 1 for each county, so every county in the dataset will map as equal. To be mapped, each county must have an observation. The easiest way to get all the county codes is to pull them from the map dataset, because fips county code assignments are not intuitive.

```
proc sql;
create table dummy as
  select distinct county,
  1 as mapv
  from two;
quit;
```

Reset the graphics options and then specify the gif device driver. We've chosen a white background for an image. Pattern statements assign colors for map variable levels. Since there is only one value for mapv, we need only one pattern.

```
goptions reset;
goptions device=gif cback=white;
pattern1 color=blue;
```

GENERATING A MAP INTERACTIVELY

We can produce the map in an interactive session as follows.

```
proc gmap map=one
  data=dummy;
  id county;
choro mapv/ coutline=black nolegend
  name="mymap";
run;
quit;
```

Figure 1 shows the resulting map.

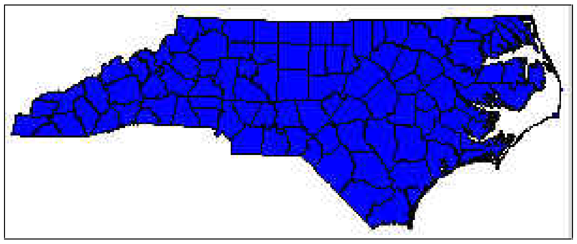


Figure 1

GENERATING A MAP INTERACTIVELY USING ODS

The following additions use the Output Delivery System to create an html file treating the map as a simple gif, not an imagemap. The ODS path parameter gives the hard path for writing both the html and the gif file. Specifying the path in a filename statement rather than directly in the ods statement prevents incorporation of the path into the html gif reference. Without the path, the browser will simply look in the current directory for the gif file.

```
filename o 'directory/path/for/files';
ods html
  path=o
  body='pagename.html';
proc gmap map=one
  data=dummy;
  id county;
```

```
choro mapv/ coutline=black nolegend
  name="mymap";
run;
quit;
ods html close;
```

Browsing to pagename.html, we again see the map of North Carolina. Viewing the html source code reveals the reference <IMG SRC="mymap.gif" border="0">.

GENERATING A MAP FROM THE WEB

Leaving the interactive session behind, we place our SAS code in a program library defined to the Application Dispatcher. We then generate the map using either a URL or a web form. With Application Dispatcher, one usually supplies an ODS html body parameter of _webout (dynamic) to send the output to the browser window. In this instance, however, there must still exist a gif file for the page to reference. When we change the body parameter to _webout, we find that the reference to the gif file (in the current directory) no longer works. Moving the path out of the filename statement and into the ODS html statement will not work because SAS is writing to a hard path while the html reference to the gif is URL-based. The solution is to use a base parameter. The base parameter provides the URL prefix to correctly locate the gif file from an html page.

Whenever running graphics using Application Dispatcher, it is important to first close the ODS listing destination. Otherwise, SAS will attempt to write a copy of the image where the broker application resides.

```
ods listing close;
filename o 'directory/path/for/files';
ods html
  path=o
  body=_webout (dynamic)
  base="complete-url-to-directory-o/";

proc gmap map=one
  data=dummy;
  id county;
choro mapv/ coutline=black nolegend
name="mymap";
run;
quit;
ods html close;
```

There is another complication involving browser caching with the above code. Even with browser preferences set to always obtain files from the server rather than use a cached version, the gif may still be pulled from cache. Manually refreshing the _webout page simply reruns the program, which again uses the cached gif. One can 'open image in new window' and then successfully refresh, but this is not particularly acceptable.

One workaround is to write the html to a file rather than to _webout. A simple data step can, using put statements, compose an intermediate page with a link to the html file containing the map. The user can click that link to bring the map page into the browser. Since the page showing the gif is no longer _webout, it can be refreshed without resubmitting the job. The base parameter is not needed in this workaround. Instead, we revert to using a path defined in a filename statement, causing the browser to look for the gif in the current directory.

```
ods listing close;
filename o 'directory/path/for/files';
ods html
  path=o
  body="myfile.html";

proc gmap map=one
data=dummy;
id county;
choro mapv/ coutline=black nolegend
name="mymap";
run;
quit;
ods html close;

data _null_;
file _webout;
put 'Content-type: text/html';
put;
put 'Click below to see your map';
put
'<a href="url-to-myfile.html">map</a>';
run;
```

SPECIFYING THAT THE MAP BE AN IMAGEMAP

For the examples thus far, the original input data was sufficient. For an imagemap, however, SAS requires a third variable in the data to be mapped, the link variable. This variable should hold the html link value to use with the current observation. This is the variable that allows different counties to link to different web pages. To keep our example simple, we assume there exists an html page for each county named *fipscode*.html. In other words, for a fips county code of 25, to get a link pointing to 25.html the variable value should be **a href="25.html"**. The imagemap parameter on the PROC GMAP statement causes all the polygon coordinates for each county to be html-encoded. The html parameter on the CHORO statement indicates the link variable.

```
proc sql;
create table dummy as
  select distinct county,
  1 as mapv,
  'a href="'||
    trim(left(put(i,3.)))||
    '.html"' as linkvar
  from two;
quit;

ods listing close;
filename o 'directory/path/for/files';
ods html
  path=o
  body="myfile.html";

proc gmap map=one
data=dummy imagemap=m;
id county;
choro mapv/ coutline=black nolegend
name="mymap" html=linkvar;
run;
quit;
ods html close;

data _null_;
file _webout;
put 'Content-type: text/html';
put;
put 'Click below to see your map';
put
'<a href="url-to-myfile.html">map</a>';
run;
```

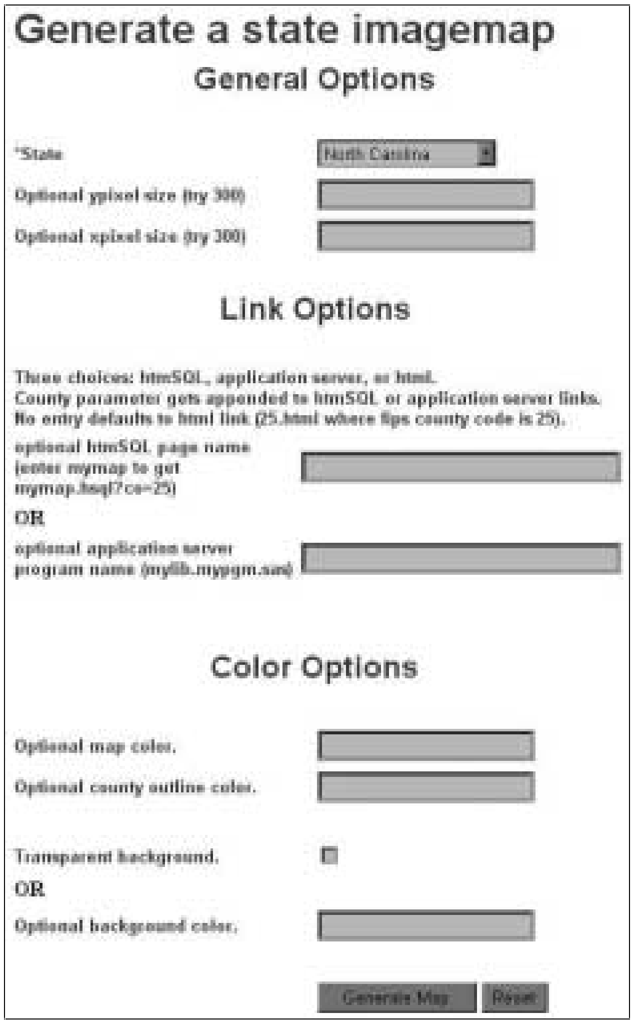
Browsing to myfile.html reveals a map looking exactly like those already created. This time, however, when the mouse is passed over the state, links are visible and each county links to a different page.

LETTING THE USER CUSTOMIZE THE MAP

The application incorporates the following as variables:

NAME	DESCRIPTION
St	State
Yp	Ypixels – goptions size parameter. Determines height of image.
Xp	Xpixels – goptions size parameter. Determines width of image
hsqname	A base for an htmSQL reference. If, for example, mybase is entered, the link for county 25 will be mybase.hsql?co=25
progname	A program name for an Application Dispatcher link. If a.b.sas were entered: url?_program=a.b.sas&_service=default&co=25
mapcolor	Color used for all counties. Any valid SAS color name may be used. For example, in addition to named colors, HTML hex specs like #FFCC00 are used in SAS with a CX prefix, as in CXFFCC00.
Co	County outline color
Transp	Specifies a transparent background
cb	Image background color

Table 1



Generate a state imagemap

General Options

*State North Carolina

Optional ypixel size (try 300)

Optional xpixel size (try 300)

Link Options

Three choices: htmSQL, application server, or html.
County parameter gets appended to htmSQL or application server links.
No entry defaults to html link (25.html where fips county code is 25).

optional htmSQL page name (enter mymap to get mymap.hsql?co=25)

OR

optional application server program name (mylib.mypgm.sas)

Color Options

Optional map color.

Optional county outline color.

Transparent background.

OR

Optional background color.

Generate Map Reset

Figure 2

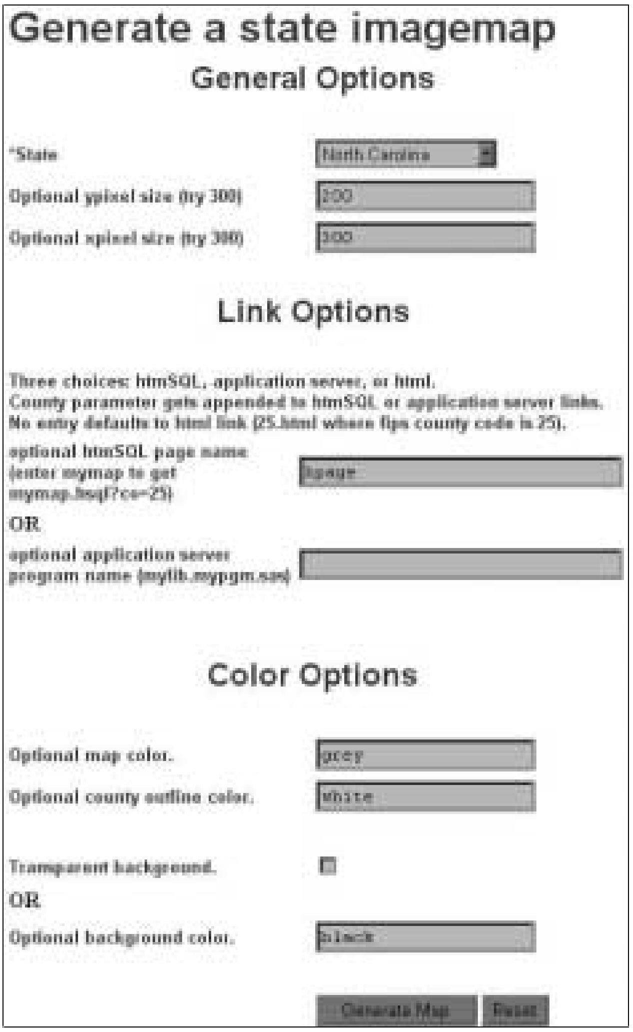
The fields listed in table 1 are used in the web form shown in figure 2. When using Application Dispatcher, form field values are passed to the SAS program as macro variables. Placing the program code inside a macro enables the use of macro programming language. The following code makes use of all the options listed in the form.

```
%global st transp xp yp cb co hsqname
progname mapcolor;

%macro amap;

data two;
  set maps.counties;
  if state=stfips("&st");
  run;

proc gproject data=two out=one;
  id county;
  run;

proc sql;
create table dummy as select
distinct county,
1 as mapv,
'a href="'||
  %if &hsqname ne %then
   "%trim(&hsqname)"||
   '.hsql?co='||
   left(put(county,3.))||
   '"'
   ;
  %else %if &progname ne  %then
   "%trim(&_URL)"||
   '?service=_default&_program='||
   "%trim(&progname)"||
   '&co='||
   left(put(county,3.))||
   '"'
   ;
  %else
   trim(left(put(county,3.)))||
   '.html"'
   ;
 as linkvar
from two;
quit;

goptions reset;
ods listing close;
filename o '/mypath';
ods html body='amap.html' path=o ;
goptions device=gif
  %if "&transp"="yes" %then transparency;
  %if &cb ne %then cback=&cb;
  %if &xp ne %then xpixels=&xp;
  %if &yp ne %then ypixels=&yp;
  ;
%if &pattern1 ne %then
  pattern1 color=&mapcolor;;
proc gmap map=one
 data=dummy imagemap=m;
 id county;
choro mapv/
 coutline=
  %if &co eq %then black;
  %else &co;
 html=linkvar name='forweb' nolegend ;
run;
ods html close;

data _null_;
```

```
file _webout;
put ’Content-type: text/html’;
put;
put ‘<title>Imagemap Results</title>’;
put ‘<font size=+3 align=center>’
 ‘Click the link below to see your map.’
 ‘</font><p>’;
put '<font size=+2 align=center>’
 ‘You must perform’
 ‘<font color=red>two</font>'
 ‘saves to the same directory.</font><p>';
put '<table><font align=center>’
 ‘<tr><td>'
 ‘"save as" for the html’
 ‘<td>’
 ‘<font size=+2 color=orange>and</font>’
 ‘<td>’
 ‘"save image as" for the gif file.’
 ‘</tr><tr></tr>’;
put '<tr><td>’
 ‘<font size=+2 color=green>ok</font>’
 ‘to rename html<td>’
 ‘<font size=+2 color=orange>but</font><td>’
 ‘<font size=+2 color=red>do not</font>’
 ‘rename the gif file’;
put '</tr><tr><td>';
put
 ‘<a href="myurlpath/amap.html">’
 ‘<font size=+3>map</font></a>’
 '</font></table>';
put '<font size=+3 color=orange>NOTE:</font>’
‘Make sure your browser refreshes both html ’
‘and image for repetitive map generations.’;
run;
%mend amap;

%amap
```

A CUSTOMIZED EXAMPLE

The user requests a map of North Carolina 200 pixels tall by 300 pixels wide. Links are to be htmSQL links. The htmSQL page is hpage.hsql. The map should be gray with counties outlined in white on a black background. Figure 3 shows this request.



Generate a state imagemap

General Options

*State North Carolina

Optional ypixel size (try 300) 200

Optional xpixel size (try 300) 300

Link Options

Three choices: htmSQL, application server, or html.
County parameter gets appended to htmSQL or application server links.
No entry defaults to html link (25.html where fips county code is 25).

optional htmSQL page name (enter mymap to get mymap.hsql?co=25) hpage

OR

optional application server program name (mylib.mypgm.sas)

Color Options

Optional map color. grey

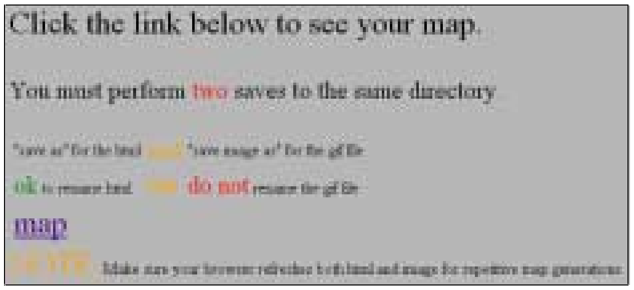
Optional county outline color. white

Transparent background. ☐

OR

Optional background color. black

Figure 3



Click the link below to see your map.

You must perform two saves to the same directory

"save as" for the html and "save image as" for the gif file

ok to rename html but do not rename the gif file

map

NOTE: Make sure your browser refreshes both html and image for repetitive map generations.

Figure 4

Figure 4 shows the message returned to the browser. The user clicks on the word ‘map’ to reveal figure 5.

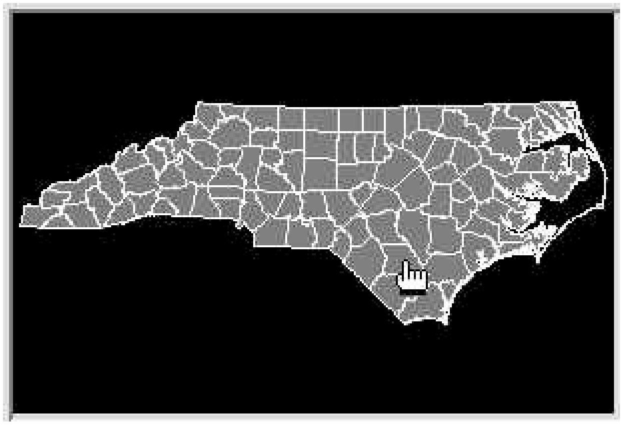
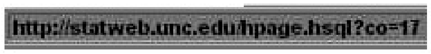


Figure 5

The mouse is positioned over the state, and the URL appears in figure 6.



http://statweb.unc.edu/hpage.hsql?co=17

Figure 6

SUMMARY

This application uses the Version 8.0 SAS Output Delivery System and SAS/GRAPH with Version 2.0 Application Dispatcher to process graph customizations from a web form. The resulting graph is an imagemap of any state in the US with the associated links taking one of three forms: Application Dispatcher calls, htmSQL calls, or html page references. Size and colors are to the user's specifications. Using an intermediate page for output that links to the requested imagemap circumvents problems related to browser caching.

CONTACT INFORMATION

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

Carol Martell
UNC Highway Safety Research Center
730 Airport Rd, CB# 3430
Chapel Hill, NC 27599-3430
Work Phone: 919-962-8713
Fax: 919-962-8710
Email: carol_martell@unc.edu