# Map in an App: The Power of GIS software in SAS Applications

by Dave Jeffreys, Aaron Hill & Lisa Weber
SAS Institute Inc.

**ABSTRACT:**

In today's competitive business environment, effective business decisions are more critical than ever. The rapid development and use of Geographic Information Systems (GIS) technology allows organizations to improve business processes and enhance decision support by combining leading GIS technology with business data.

With map (spatial) and demographic (attribute) data becoming less expensive and more accurate, more organizations are looking at business geographics applications to gain a competitive advantage. Business geographics applications provide a powerful interface for exploring business data such as sales, competition, and demographic information through an intuitive mapping environment.

This paper will demonstrate examples of how to integrate SAS/GIS software into organizational applications and link the SAS System's data analysis tools directly to the maps, providing a more intuitive understanding of the data.

**INTRODUCTION:**

FACT: The power of SAS's graphics, statistical, forecasting and additional diverse analytical tools provides the mechanism for creating sophisticated decision support applications which analyze and present critical business data to our users.

Frequently, the users of these decision support applications are not SAS programmers, but business end-users. These end-users need visual tools for investigating, analyzing and presenting data in order to reinforce business decisions. With the release of SAS/GIS software, application developers can now provide powerful desktop mapping capabilities to these business end-users.

SAS/GIS, as an integrated component of the SAS System, can be linked to all SAS/EIS and SAS/AF applications. Moreover, these applications can be launched from within SAS/GIS. This allows application developers to tap into 'out-of-the-box' SAS/EIS objects or customized SAS/AF screens, thus linking them directly to the mapping environment.

There is no doubt that the concept of 'Map in an App' is catching the attention of SAS users world wide. To assist in learning more about the concept of a "Map in an App', the following sections will outline the steps of how to integrate SAS/GIS into your SAS/EIS or SAS/AF applications. In addition, linking the SAS System's powerful analytical tools into this environment will be highlighted and discussed.

**INTEGRATING SAS/GIS WITH SAS APPLICATIONS:**

There are two primary methods employed to achieve SAS/GIS integration with SAS applications. The first method consists of using the program action within SAS/GIS. SAS/GIS program actions can execute SAS programs based on tables created from features selected from the map. The second method is the employment of a SAS/EIS or SAS/AF application to control the session and utilize SAS/GIS as a viewer component. SAS/GIS sessions which are driven from SAS/EIS or SAS/AF applications provide users with powerful SCL and data step processing capabilities for manipulating data such as the theme data sets utilized by the map. Application specific criteria can be used to modify a map's coverage or layer partitioning. A resulting map would display information more pertinent to the application subject matter.

**HOW TO SECTIONS:**

Overview on linking your SAS/GIS map to external data.

Map features are associated with external data by key variables common to both an external table and the spatial data. The SAS/GIS program action is linked to the table and will automatically create a subset of the table with rows which contain matching key variables for the features selected in the map. Thematic layers are tied to external tables using the same method, but include an extra "theme" variable. Color or size of the layer's features are determined by the theme variable. SAS/GIS uses SAS/SHARE to open these data sets with record level locking, allowing GIS actions or applications to simultaneously read or update them.

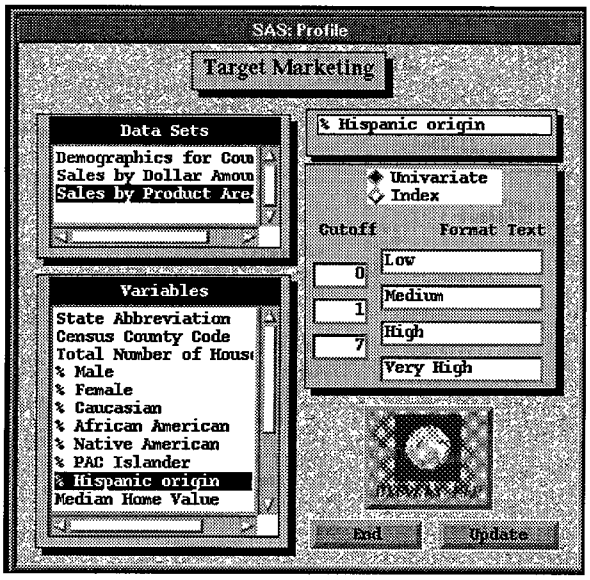
**Specific examples:**

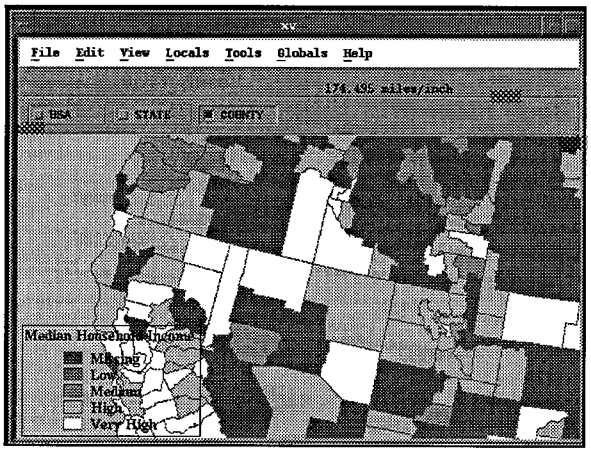
**Example 1**: Download and display of web data via the program action.

```
/*-----------------------------------------------------+
| Set data linked data set, keyed on COUNTY.
| Don't bother with multiple selections; although,
| multiple selections could be handled too.
+-----------------------------------------------------*/
 data _null_;
    set;
    call symput( 'st', put(STATE,z2.) );
    call symput( 'co', put(COUNTY, z3.) );
    stop;
    end;

filename xx URL
"http://venus.census.gov/cdrom/lookup/DB=C90STF3A/CMD
=RET/FMT=HTML/LEV=COUNTY90/F0=FIPS.STATE/F1=
FIPS.COUNTY90/SEL=&st,&co/T=P2/T=P60"
```

```
  proxy='http://inetgw.unx.sas.com' ;

data _null_;
  infile xx;
  input;
  file 'test.html';
  put _infile_;
run;

dm "wbrowse test.html";
```

**Example 2**: Changing the theme data set of a layer dynamically from application code. In this application,SAS/GIS is used as a viewer.




Default breaks are supplied in percentages for use with the 'Index' radio selection. The list box showing the available data sets is initialized, etc.

```
init:
  /*--- Initialize the widgets, user array to get ds names.....*/
  control asis;
  %let DSLIB    = SUGI22;
  %let STATAREA = COUNTY;
  %let LBLDS    = &DSLIB.CNTY_LBL;
  %let BRK1     = 80;
  %let BRK2     = 120;
  %let BRK3     = 200;

  ...
```

The map is initially invoked from the application. The launching icon is then grayed.

```
map:
  /*--- Invoke GIS and gray icon .....*/
  call notify( 'map','_gray_' );
  call notify('update','_ungray_');
  call execcmdi( 'gis "map=sugi22.dbm.county";');
return;
```

This label is driven when a new variable is selected for the current data set. The county layer has already been themed on a numeric summary variable, which takes on values of 1 to 4. A variable has been formatted for display in the GIS legend.

The code below will either choose indexed break points for the variable or will run the univariate procedure to compute them. These breaks will be used to replace the summary variable when the map is updated from the update label, driven by 'Update' button.

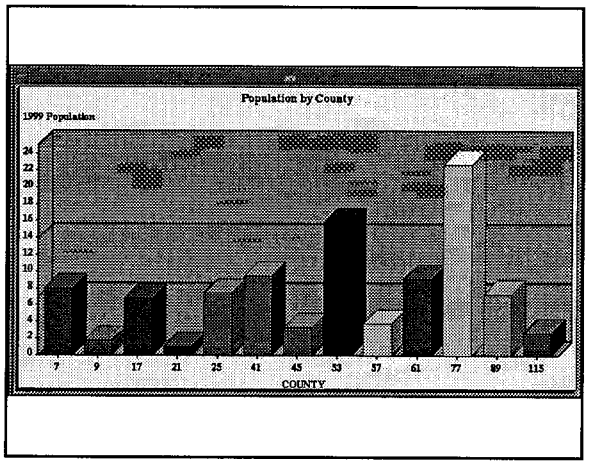
```
vars:
  call notify( 'vars', '_get_last_sel_', row, sel, label );
  call notify( 'varlabel', '_set_text_', label );
  var = varname( dsid, row );

  /*--- Unprotect the variable label and format range fields
        also uprotect the update button and radio box.....*/
  call notify( 'varlabel','_unprotect_');
  call notify( 'val1','_unprotect_');
  call notify( 'val2','_unprotect_');
  call notify( 'val3','_unprotect_');
  call notify( 'radiobin','_ungray_');
  call notify( 'update','_ungray_');

  /*--- check the radio box buttons for the univariate button.*/
  call notify( 'radiobin','_get_station_','Univariate',butnum );
  call notify( 'radiobin','_issel_',butnum,unival );
  indxval = not unival;

  /*--- if the univariate button is pushed then use
  univariate.....*/
```

```
  if unival eq 1 then do;
    submit continue;
      proc univariate data=&dsname noprint;
      var &var;
      output out=work.themeuni mean=mean q1=cut1
      median=cut2 q3=cut3;
      run;
    endsubmit;

    /* reassign the cut off values */
    udsid = open( 'work.themeuni' );
    if udsid ne 0 then do;
      call set(udsid);
      rc = fetch( udsid );
      rc = close( udsid );
      udsid = 0;
    end;

  end;
  else
    link dfltbrks;
  call notify( 'val1','_set_value_', cut1 );
  call notify( 'val2','_set_value_', cut2 );
  call notify( 'val3','_set_value_', cut3 );
return;

dfltbrks:
  cut1 = &BRK1;
  cut2 = &BRK2;
  cut3 = &BRK3;
return;
```

The update label replaces the summary variable from the univariate results computed above or uses the default break points supplied in the init section. Finally a single command is issued to GIS, which will update map based on the current data set and theme variable chosen. In this way GIS is used as a viewer component for the application.

```
update:
  /*--- Update the reusable HML categorization variable....*/
  call notify( 'radiobin','_get_station_','Univariate',butnum );
  call notify( 'radiobin','_issel_',butnum,unival );
  indxval = not unival;

  if indxval eq 1 then do;
    rc = varstat( dsid, var, 'MEAN', mean );
    submit continue;
      data &dsname; modify &dsname;
        index = &vars / &mean * 100;
    endsubmits;
  end;
  else do;
    submit continue;
      data &dsnamess; modify &dsname;
        index = &var;
    endsubmit;
  end;
  submit continue;
      if index <= &Val1 then hml = 1;
      else if index <= &Val2 then hml = 2;
      else if index <= &Val3 then hml = 3;
      else hml = 4;
      format hml hml.;
      run;
  endsubmit;

  /*--- Retheme the map with the new summary.....*/
  submit continue;
    dm gis "Layer ReTheme &STATAREA data=&dsname";
  endsubmit;
return;
```

**Example 3**: Using SAS/GIS to launch and interact with other components of the SAS system.




```
/*--- App SCL, driven by GIS from program  action.....*/
init:
  control always;
  length name $35 title $16;

/*--- Create a global list item with the name of the
    frame and a value of the window title.....*/
  call send( _self_, '_get_name_', name );
   title = getnitemc( _self_, 'name' );
  if title = '' then _status_ = 'H';

  env = envlist( 'g' );
  rc  = setnitemc( env, title, name );

  /*--- Initialize frame contents.....*/
  link update;
return;

main:
  /*--- Subsequent action envocations will send a command,
   which will be processed here.  Ignore the command
   text, but update the contents based on the SAS/GIS
   program action output.....*/
 link update;
  call nextcmd();
```

```
return;

term:
  /*--- Ensure that the global list item is removed.....*/
  if name ^= " & nameditem( env, name ) then
    rc = delnitem( env, name );
return;

update:
  /*--- Application specific widget updating goes here
      This is where the bar chart is populated and updated
      with each SAS/GIS action invocation.....*/
    ...
return;



/*.SCL program action for initializing and then updating the
  application window. The global list item stored by the
  application's init section is used to target the update
  command to the correct window.....*/
%let appname = &APPLIB..&APPLCAT..APP1.FRAME;
length app $16;
init:
/*---Send Update command to the application if running.....*/
  env = envlist( 'g' );
  app = getnitemc( env, "&appname", 1, 1, "" );
  if app ^= " then do;
    submit;
      dm 'Next &app; Update';
    endsubmit;
    end;
  else
    /*--- Start the application.....*/
    call display( "&appname" );
return;
```

**APPLICATION EXAMPLES:**

Following the guidelines above, many SAS/GIS customers have taken advantage of adding mapping capabilities to their decision support applications and the results have been powerful!

For example, SAS users in the Health Care industry are using SAS/GIS for a variety of activities such as mapping their market share, pinpointing their hospital locations and the locations of their competitors, and displaying other key variables. Resulting maps visually establish the areas from which their patients as well as their competitor's patients emanate, the areas containing high concentrations of specific medical occurrances, and other customized requests. Color coding the map based on particular criteria demonstrates where profitable new clinics or offices could be established or where focused marketing efforts can reacquire customers. SAS/GIS provides an interactive map which allows the user to launch reports, graphics and further analysis, providing them a key advantage in making sound business decisions.

In the Bank Industry, SAS users have integrated SAS/GIS into large decision support applications to improve lending practices and to investigate branch placement based on demographics and market share. SAS/GIS can describe where a bank's customers are in relation to bank branches, ATMS and competition as well as represent the demographics of borrowers. This visual representation of information can help banks quickly interpret the business occuring in their territories and make more informed decisions. Linking this mapping environment to their current analytical methods provides banks with an effective tool that surpasses traditional graphics and reports.

Similar applications have been seen in areas of retail, insurance, government and other industries where geographics play a key role. Desktop mapping is exploding into the business community as businesses find and see the value of adding mapping technology to their decision support applications.

**FUTURE:**

In the near future, look for the release of a SAS/GIS Frame object to expand SAS/GIS's integration with SAS/EIS applications. The goal is to provide SAS/EIS and SAS/AF users with a mapping product that is fully integratable with new or existing applications, exacting ease-of-use as an essential characteristic. Additional goals for the product include, but are not limited to, enhancing PROC GIS, providing a more powerful batch tool and refining the geocoding capabilities.

**Acknowledgments**

The following SAS/GIS team members contributed to the preparation of this paper:

Barry Hicks Sandi Smith
Jack Bulkley



| | | |
|---|---|---|
| Dave Jeffreys | sasdkj@unx.sas.com | 677-8000 |
| Aaron Hill | sascyh@unx.sas.com | 677-8000 |
| Lisa Weber | saslny@unx.sas.com | 677-8000 |