Paper 131-27

Self Serve Census Data for Neighborhoods and other Custom Aggregations: A SAS/IntrNet® Application

Larry Hoyle, Policy Research Institute, University of Kansas, Lawrence, KS

ABSTRACT

In the past, obtaining Census statistics for aggregates of Census blocks was a somewhat tedious task, which required the involvement of someone skilled in Census Geography and programming, or a GIS package.

The Web application described in this paper allows users to define and name their own collection of counties, census tracts, block groups and blocks. They are then able to display standard Summary File Profiles for the aggregated area. For example, the XYZ Neighborhood Association might want Census statistics for the neighborhood as a whole. Someone from the association could pick the blocks comprising the neighborhood, save the selection as "XYZ", and then select which sets of tabulations to display. He or she could return to the site later and select other tables to display for "XYZ".

This paper shows an example of defining a neighborhood and generating tables. It also describes the SAS techniques used to implement the application. It will also discuss design considerations such as bandwidth, and accessibility. The paper is appropriate for users with intermediate SAS skill levels.

The application uses SAS/IntrNet running on a Digital UNIX system.

INTRODUCTION

The Policy Research Institute, and its earlier incarnations, has been helping to distribute tabulations of US Census data to the public in our state for many years. For the 1990 Census, many inquires were answerable with photocopies from Census Bureau printed reports or microfiche. For more extensive information on a single area we used SAS code written in-house and written by people at other State Data Centers. The 29-page profile of tables from Summary Tape File 3a (STF3a) developed at the California State Data Center was particularly useful. This profile could be generated for any geographic unit down to the tract level. In the mid 90s we generated these profiles in Adobe Acrobat format (pdf) for all counties and cities in the state and made them available on our Kansas Statistical Abstract CDROM. Later we made all of these files available on the Web.

One type of inquiry though, remained something of a problem. Sometimes a neighborhood group, a parent teacher organization from a school, or some other group without much money would want aggregated statistics for their area. From our end this might involve several hours of finding the corresponding Census geographic units and then writing custom code to aggregate them and then generate a report. The cost of this was often more than the requestor was willing to pay.

For the 2000 Census we have developed a Web based solution that allows people to define an area for themselves, and then generate an aggregate profile for that area. The initial profile from Summary File 1 is generated using SAS code developed by volunteers from a group of State Data Centers. The main adaptation needed to generate a profile for an aggregate area was to sum cells in the SF1 file that were aggregable, and to set the other cells to missing. Counts of people, for example can be aggregated, while median age cannot. A further refinement was to print all values when the custom area is a single geographic unit – e.g. a single block.

We wanted to make this facility as accessible as possible, so we made it text based, with links to maps on the Census Bureau's Web site. The custom area can be defined by making selections starting from the county level down, or by entering dimensions of a rectangle surrounding a street address and then refining from a list of all the blocks in that area.

OVERALL DESIGN

In order to allow someone to define an area and then return to the site and produce tabulations of the area, a list of the geographic units selected for the area is saved as a SAS dataset on the server. Here, for example is the list for the Old West Lawrence Neighborhood, in Lawrence, KS.

formval	GeoCode
G20045_0005X02_2	20045-0005.02-2
G20045_0005X02_3001	20045-0005.02-3001
G20045_0005X02_3002	20045-0005.02-3002
G20045_0005X02_3003	20045-0005.02-3003
G20045_0005X02_3004	20045-0005.02-3004
G20045_0005X02_3005	20045-0005.02-3005
G20045_0005X02_3006	20045-0005.02-3006
G20045_0005X02_3007	20045-0005.02-3007
G20045_0005X02_3008	20045-0005.02-3008
G20045_0005X02_3009	20045-0005.02-3009
G20045_0005X02_3010	20045-0005.02-3010
G20045_0005X02_3013	20045-0005.02-3013
G20045_0005X02_3014	20045-0005.02-3014
G20045_0005X02_3015	20045-0005.02-3015
G20045_0005X02_3016	20045-0005.02-3016

The file contains two variables – one, *Geocode*, in the form found in the Census dataset, and the other *formval* in a form usable as a parameter from an HTML form. In this example all units are from state 20, county 045, and tract 0005.02. The first record is all of block group 2. The rest of the records are blocks from block group 3.

An entry for the custom area is also made in a dataset containing one record for each custom area. This dataset has these fields:

Field	Contents
areaNum	a unique number for the area
areapw	a password for revising the area description
created	the creation date of the area
creatorip	the IP number from which the area was created
customArea	a name for the area
email	the optional email address of the creator
keepdays	the number of days to keep the definition

The typical custom area definition uses under 20k bytes on the server.

THE USER INTERFACE

The custom area home page (not shown here) allows users to choose between defining new areas, displaying the geography of already defined areas, displaying a profile of Census data for the area, or displaying a population pyramid for the area.

DEFINING AN AREA – ADDRESS LOOKUP

In some cases, it may be easiest to describe an area starting from a rectangle around a particular address. Suppose we are looking for the square mile around the Kansas City SAS office. Entering the address 9401 Indian Creek returns the page on the right.

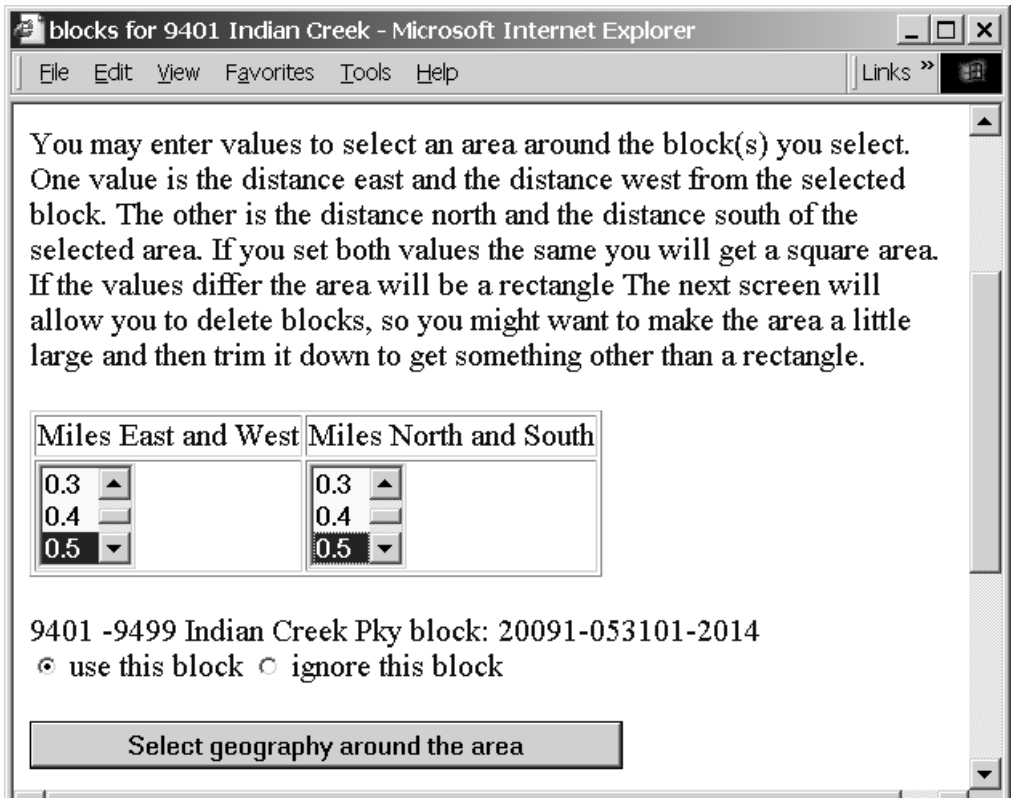

Figure 1

This fuzzy search returned a list with just 1 block. We pick it and also specify that we want all blocks having their "internal point" in a square with sides .5 miles from the internal point of this block. The list can be modified, if needed, and then saved as a custom area definition. Each geographic unit is identified by its Census block number as well as a list of all the features (e.g. streets) bordering it.

We'll save this definition with the name "KC SAS Square Mile".

DEFINING AN AREA – TOP DOWN

An area can also be defined starting from a list of all counties. If a county is marked as "select part" then, on update, the form will show all of the tracts in the county. Tracts can be expanded to show their block groups, and block groups can be expanded to show their blocks. At each level one can select a unit, expand it, or leave it out of the custom area.

The area definition is saved with a name and a password. The password allows the creator to later modify the definition of the area.

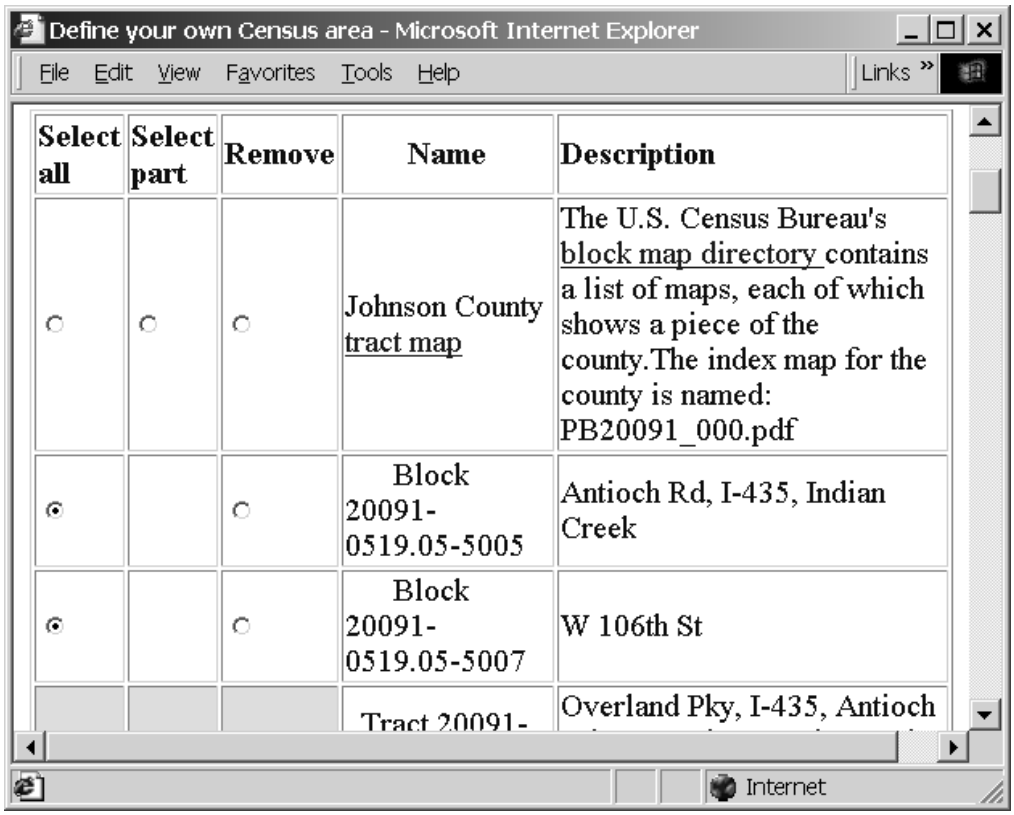

Figure 2

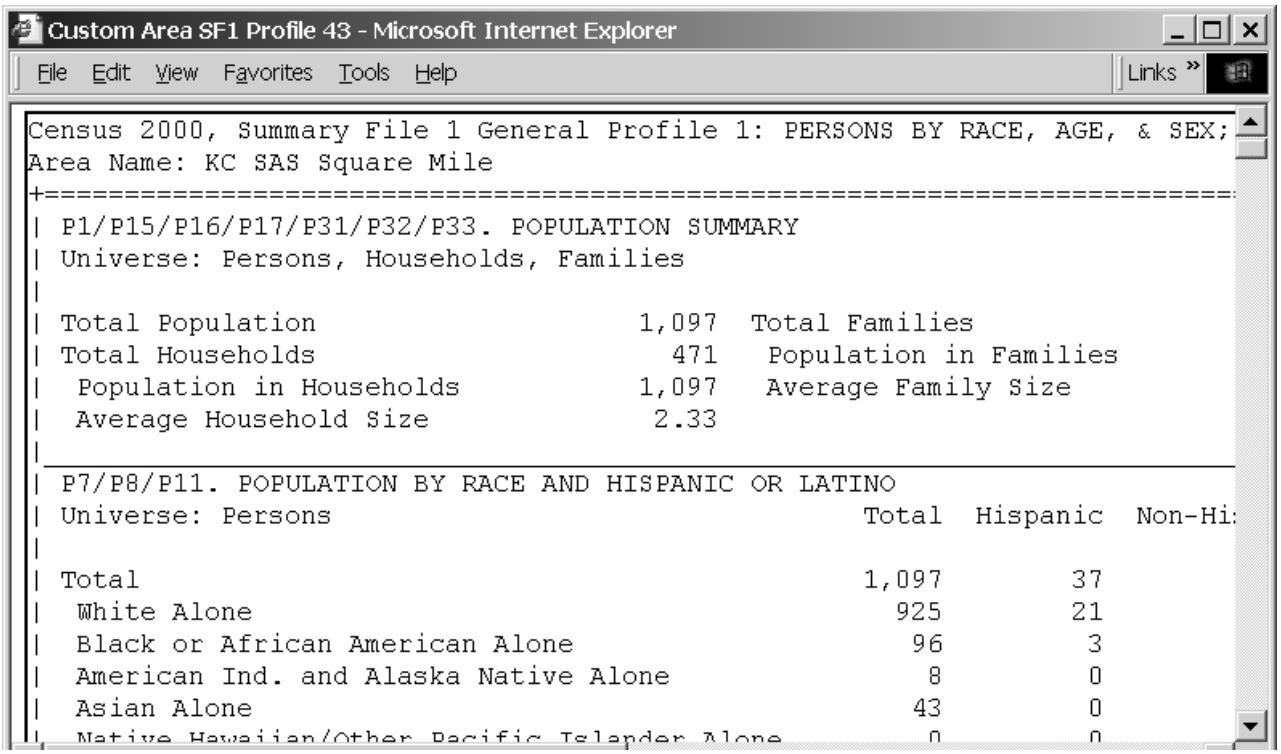

Figure 3

INFORMATION ABOUT A CUSTOM AREA

Once an area definition has been saved, anyone can generate an aggregate SF1 profile or a population pyramid for the area. The SF1 profile can be generated using a font suitable for printing, or one suitable for on-screen viewing. A sample of the latter is at the top of this page.

The full profile is 17 pages of tables from the 2000 Census Summary File 1. Aggregating medians is a problem, so the program sets all medians to missing for any custom area of more than one geographical unit.

We could also choose to display a table of population by age and gender and a population pyramid for the aggregated area.

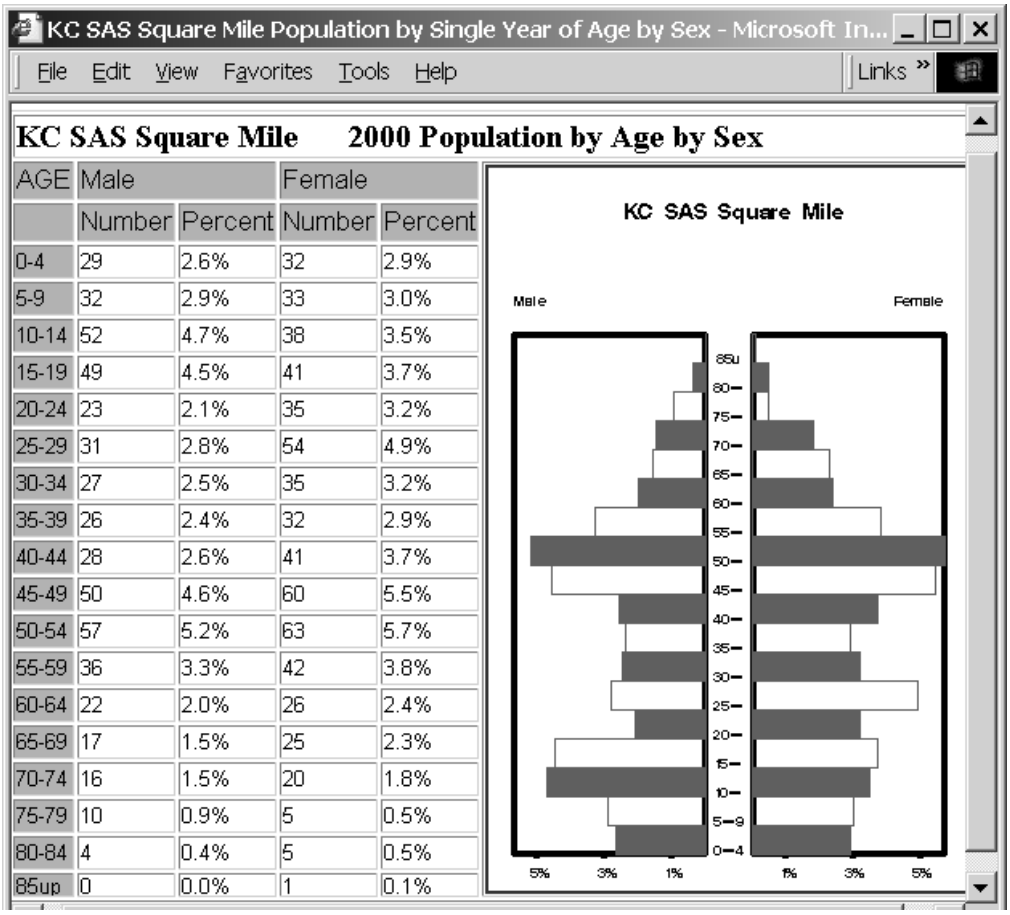

Figure 4

PREPARATION

THE GEOGRAPHIC SELECTION DATASET

This project required the preparation of a dataset (geoselect) containing a record for each possible geographic unit – county, tract, block-group, and block. For those units, these fields were extracted from the SF1 file:

AreaLand	*land area*
AreaName	*90 character area name*
AreaWatr	*water area*
BG	*block group code*
Block	*block code*
County	*county code*
GeoCode	*geographic code*
HU100	*number of housing units*
IntPtLat	*latitude of an internal point*
IntPtLon	*longitude of an internal point*
Pop100	*number of people*
State	*state code*
SumLev	*summary level (e.g. 140 is a tract)*
Tract	*tract code*
geo_id	SF1 file record key
formval	*macro variable compatible geocode*
desc	*unique list of bordering feature names*

The field *formval* is GeoCode recoded as

```
'G'||translate(geoCode,'_X','-.') as formval
```

This is a form acceptable as a macro variable. The field *desc* contains a unique list of all the feature names (e.g. streets) surrounding the geographic unit. The latter field was created from the Census Tiger file corresponding to the SF1 file and is used for the "Description" column in the table in figure 2.

THE ADDRESS LOOKUP FILES

The address lookup facility required the creation of a file for each county with the following fields:

HIaddL, HIaddR, LOaddL, LOaddR,	*(*High and low ranges)
fedirp, fedirs, fename, fetype,	(feature identifiers)
geocodeL, geocodeR	*(*area on left and right*)*

This file has a record for each Tiger line segment associated with each block in the county. Note that the ranges are high and low rather than the beginning and ending numbers that appear in the Tiger file. The feature identifiers include a name, a type, a prefix, and a suffix.

IMPLEMENTATION

ADDRESS LOOKUP

Consider the address 123 S. MyStreet. The Census Tiger file might include the "S." in the feature name or might put it in the feature prefix. This necessitates some sort of fuzzy match for address lookup. The fuzzy match is also convenient for users. The application uses this match:

```
     where (((LOaddL <= &num) and (&num <=
HIaddL)) OR
           ((LOaddR <= &num) and (&num <=
HIaddR))) AND

(spedis(upcase(fename),"&street")<=25 OR

spedis("&street",upcase(fename))<=25  OR
        upcase(fename)=*"&street") ;
```

Code like this determines which side of the street to use:

```
          /* number even */

          /*left even  number even, use LEFT */
    if  mod(LOaddL,2)=0 AND  mod(&num.,2)=0
then do;
       geocode=geocodeL;
       low=LOaddL;
       high=HIaddL;
    end;
      /* right even number even, use RIGHT */
    if   mod(LOaddR,2)=0 AND  mod(&num,2)=0
then do;
      geocode=geocodeR;
       low=LOaddR;
       high=HIaddR;
    end;
```

The GeoCode field can then be matched to the geoselect dataset. The following code finds the units in the specified rectangle:

```
/*  geographic units which are rectangle
centers */
 create table centers as
 select 1 as center,
        min( (IntPtLat+(&nsRadius./69.1)),
   (IntPtLat-(&nsRadius./69.1))  ) as minLAt,
   max( (IntPtLat+(&nsRadius./69.1)),
   (IntPtLat-(&nsRadius./69.1))  ) as maxLAt,
   min(
(IntPtLon+(&ewRadius./(69.1*cos(IntPtLat*&rPe
rDeg.)))),
   IntPtLon-
(&ewRadius./(69.1*cos(IntPtLat*&rPerDeg.))))
) as minLon,
   max(
(IntPtLon+(&ewRadius./(69.1*cos(IntPtLat*&rPe
rDeg.)))),
   (IntPtLon-
(&ewRadius./(69.1*cos(IntPtLat*&rPerDeg.))))
) as maxLon,
        *  from sf12000.geoselect
 where formval in (select name from mvars
where upcase(value)='R');

 create table rects as
 select 1 as keep, g.*
 from sf12000.geoselect as g, work.centers as
c
 where g.IntPtLat>=c.minLat AND
       g.IntPtLat<=c.maxLat AND
       g.IntPtLon>=c.minLon AND
       g.IntPtLon<=c.maxLon;
```

SELECTING AND EXPANDING AREAS

Once the blocks for the area around an address are located, or when starting from the list of counties, a form like figure 2 allows the user to select or expand areas. The HTML for the first row of that form looks like:

```
<tr>
<td>
 <input type="radio" name="G20091" value="K">
</td>
<td>
 <input type="radio" name="G20091" value="S">
</td>
<td>
 <input type="radio" name="G20091" value="D"
CHECKED >
</td>
<td>Johnson County
 <A
href="http://ftp2.census.gov/plmap/pl_trt/st2
0_Kansas/c20091_Johnson/CT20091_001.pdf"
    target="Census_map_window">
 tract map </A>
</td>
<td>The U.S. Census Bureau's
 <A
href="http://ftp2.census.gov/plmap/pl_blk/st2
```

```
0_Kansas/c20091_Johnson/"
    target="Census_map_window"> block map
directory </A>
 contains a list of maps, each of which shows
a piece of the county.
 The index map for the county is named:
PB20091_000.pdf
</td>
</tr></tr>
```

In order to use radio buttons in the form, each possible geographic unit needs to have its own name and the SAS program accepting input from the form as macro variables needs to be able to identify those names as being for geographic areas quickly. The trick here is to make a rule that only geographic area radio buttons use a name beginning with "G" in this form. The SAS program can easily find all of those form variables with this query against the "sashelp.vmacro" view of all macro variables:

```
proc sql;
  create table mvars as
  select name,value
  from sashelp.vmacro
  where scope='GLOBAL' and
upcase(substr(name,1,1))="G";
```

The radio buttons can have one of these values:

K - keep the area
S - subset the area
D - delete the area

Other values are used in hidden tags in forms sent back by the program.
H - hide controls but display the area - previously subsetted
E - show the area for re-editing - drop selected
R - keep everything in a radius around the area the area

Values of "R" are sent by the address lookup form.

UNACCEPTABLE AREA NAMES?

We had some concern about the possibility that someone would create an area with a name we didn't want displaying in a drop down box on our server. Consequently, each time a new area is created, the SAS program sends an email with the area name to one of our staff – just in case.

```
filename mymail email "pri@ku.edu"
   subject="new custom area";

data _null_;
   file mymail;
 set areas.areaPWs;
 where areaNum=&newAnum ;

 put areaNum= / customArea= / created= /
keepdays= / creatorIP=/ email=;
run;
```

MODIFYING THE PROFILE CODE

The SF1 profile code, which is available on the State Data Center Clearinghouse Web site (http://www.sdcbidc.iupui.edu/), produces tables using pointer control in DATA steps. In order produce HTML which prints reasonably to landscape pages we used the following ODS code:

```
ods path work.template(update)
sashelp.tmplmst(read);

 proc template;
define style Styles.myHtm;
   parent = styles.default;

   replace color_list
      "Colors used in the default style" /
      'fgB2' = cx0066AA
      'fgB1' = cx004488
      'fgA4' = cxAAFFAA
      'bgA4' = cxFFFFFF
      'bgA3' = cxFFFFFF
      'fgA2' = cx0033AA
      'bgA2' = cxFFFFFF
      'fgA1' = cx000000
      'bgA1' = cxFFFFFF
      'fgA' = cx002288
      'bgA' = cxFFFFFF;

    replace colors
      "Abstract colors used in the default
style" /
      'headerfgemph' = color_list('fgA2')
      'headerbgemph' = color_list('bgA2')
      'headerfgstrong' = color_list('fgA2')
      'headerbgstrong' = color_list('bgA2')
      'headerfg' = color_list('fgA2')
      'headerbg' = color_list('bgA2')
      'datafgemph' = color_list('fgA1')
      'databgemph' = color_list('bgA3')
      'datafgstrong' = color_list('fgA1')
      'databgstrong' = color_list('bgA3')
      'datafg' = color_list('fgA1')
      'databg' = color_list('bgA3')
      'batchfg' = color_list('fgA1')
      'batchbg' = color_list('bgA3')
      'tableborder' = color_list('fgA1')
      'tablebg' = color_list('bgA1')
      'notefg' = color_list('fgA')
      'notebg' = color_list('bgA')
      'bylinefg' = color_list('fgA2')
      'bylinebg' = color_list('bgA2')
      'captionfg' = color_list('fgA1')
      'captionbg' = color_list('bgA')
      'proctitlefg' = color_list('fgA')
      'proctitlebg' = color_list('bgA')
      'titlefg' = color_list('fgA')
      'titlebg' = color_list('bgA')
      'systitlefg' = color_list('fgA')
      'systitlebg' = color_list('bgA')
      'Conentryfg' = color_list('fgA')
      'Confolderfg' = color_list('fgA')
      'Contitlefg' = color_list('fgA')
      'link2' = color_list('fgB2')
      'link1' = color_list('fgB1')
      'contentfg' = color_list('fgA2')
      'contentbg' = color_list('bgA2')
      'docfg' = color_list('fgA')
      'docbg' = color_list('bgA');


   replace fonts
      "Fonts used in my HTML style" /
      'TitleFont2' = ("Arial, Helvetica,
Helv",4,Bold Italic)
      'TitleFont' = ("Arial, Helvetica,
Helv",5,Bold Italic)
      'StrongFont' = ("Arial, Helvetica,
Helv",4,Bold)
      'EmphasisFont' = ("Arial, Helvetica,
Helv",3,Italic)
      'FixedEmphasisFont' =
("Courier",&fontSize.,Italic)
      'FixedStrongFont' =
("Courier",&fontSize.,Bold)
      'FixedHeadingFont' =
```

```
("Courier",&fontSize.)
      'BatchFixedFont' =
("Courier",&fontSize.)
      'FixedFont' = ("Courier",&fontSize.)
      'headingEmphasisFont' = ("Arial,
Helvetica, Helv",4,Bold Italic)
      'headingFont' = ("Arial, Helvetica,
Helv",4,Bold)
      'docFont' = ("Arial, Helvetica,
Helv",3);

   replace Output from Container
      "Abstract. Controls basic output
forms." /
     font = fonts('FixedFont')
      background = colors('tablebg')
      rules = NONE
      frame = void
      cellpadding = .5pt
      cellspacing = 0.25pt
      bordercolor = colors('tableborder')
      borderwidth = 1;


   replace Batch from Output
      "Controls batch mode output." /
      font = fonts('BatchFixedFont')
       cellheight=8.5pt
      cellpadding = .5pt
      cellspacing = 0.25pt
       frame=void
      rules = NONE
      foreground = colors('batchfg')
      background = colors('batchbg');
  end;
run;


ods html file=_webout( NO_Bottom_Matter
title="Custom Area SF1 Profile &areaNum")
style=myHtm   CSS ;
ods listing close;
ods noproctitle;
```

The macro variable *fontsize* is previously set from the form via the following code:

```
fs=upcase(symget("fontSize"));
 select (fs);
     when('M') call
symput("fontSize","10.0pt");
     when('L') call
symput("fontSize","14.0pt");
     otherwise call
symput("fontSize","6.7pt");
  end;
```

CLEANUP

Custom Area definitions are defined with a number of days to keep the definition. A SAS program is scheduled to run daily (via a crontab entry) to delete any expired entries. The deletion code looks like:

```
create table dellist as
  select "area"||trim(left(put(areanum,6.)))
as togo
  from areas.areapws
  where (created+(keepdays*86400))<datetime()
;
delete from areas.areapws
 where (created+(keepdays*86400))<datetime()
 ;
title 'After Deletions';
```

```
proc sql;
  select areanum,
         customArea, created, keepdays
  from areas.areapws
;
quit;
      /*  delete the individual files */
data _null_;
 set dellist end=last;
 length dlist $ 2000;
 retain dlist " ";
 dlist=trim(dlist)||" "||togo;

 if last then do;
 call symput("dlist",trim(dlist));
 end;
 run;
%put &dlist;

proc datasets library=areas;
 delete &dlist.;
 run;
```


ACKNOWLEDGMENTS

Larry Hoyle and Xanthippe Stevens did design and programming for the custom area Web pages.

The team led by John Blodgett of the Missouri State Data Center created the code for the SF1 profile. Credits for that code can be seen at:

http://www.ku.edu/pri/ksdata/census/2000#sf1credits

John also created the SAS code to make SAS datasets out of the files distributed by the Census Bureau.

CONTACT INFORMATION

You are welcome to contact the author at:

Larry Hoyle
Policy Research Institute, Univ. of Kansas
Blake Hall
1541 Lilac Lane suite 607
Lawrence, KS 66044-3177
LarryHoyle@ku.edu
http://www.ku.edu/pri

The application discussed in this paper can be found at:
http://www.ku.edu/pri/ksdata/census/2000/
An extended version of the paper can be found at:
http://www.ku.edu/pri/ksdata/sashttp/sugi27