A Macro that can Create U.S State and U.S County KML Files

Ting Sa, Cincinnati Children's Hospital Medical Center, Cincinnati, OH

ABSTRACT

This paper introduces a macro that can generate the keyhole markup language (KML) files for U.S. states and counties. The generated KML files can be used directly by Google Maps to add customized state and county layers with user-defined colors and transparencies. When someone clicks on the state and county layers in Google Maps, customized information is shown. To use the macro, the user needs to prepare only a simple SAS® input data set. The paper includes all the SAS codes for the macro and provides examples that show you how to use the macro as well as how to display the KML files in Google Maps.

INTRODUCTION

Google map is a good map tool when people want to create interactive maps. KML file is a file that can be used to put different layers onto the google map, like a point, a line or a polygon area. Also inside the KML file, you can define the styles of the layers, like changing the color and transparency of the layers, adding information to the layers etc. To create a KML file, you need to understand its structure, and sometimes it will cost a lot of time to manually create a KML file if a lot of areas are included in it. Using the macro in this paper, the users just need to create a simple SAS input data set for the macro and the macro will automatically create the KML files for the U.S states and counties. For the input SAS data set, the user must provide the state or county name information, also if the user wants to change the color or transparency of the layer or adding information to the layer, the user needs to include those in the input SAS data set as well. The Google Map can import this KML file and put the states and county layers on it. Figure 1 shows the customized Midwest states displaying on the Google map. Figure 2 is the screenshot by further zooming in this map so that user can see the different transparencies among the counties. Also in figure 2, it shows that user can edit the information displayed inside the box, like adding more text information or photos to the layer.
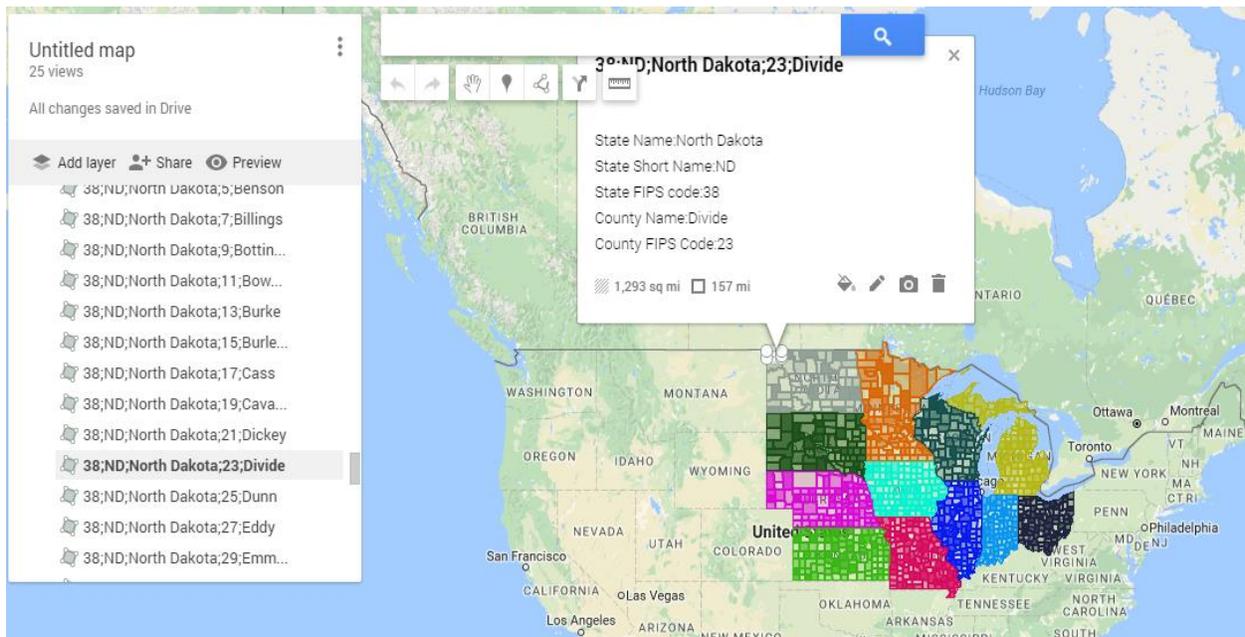

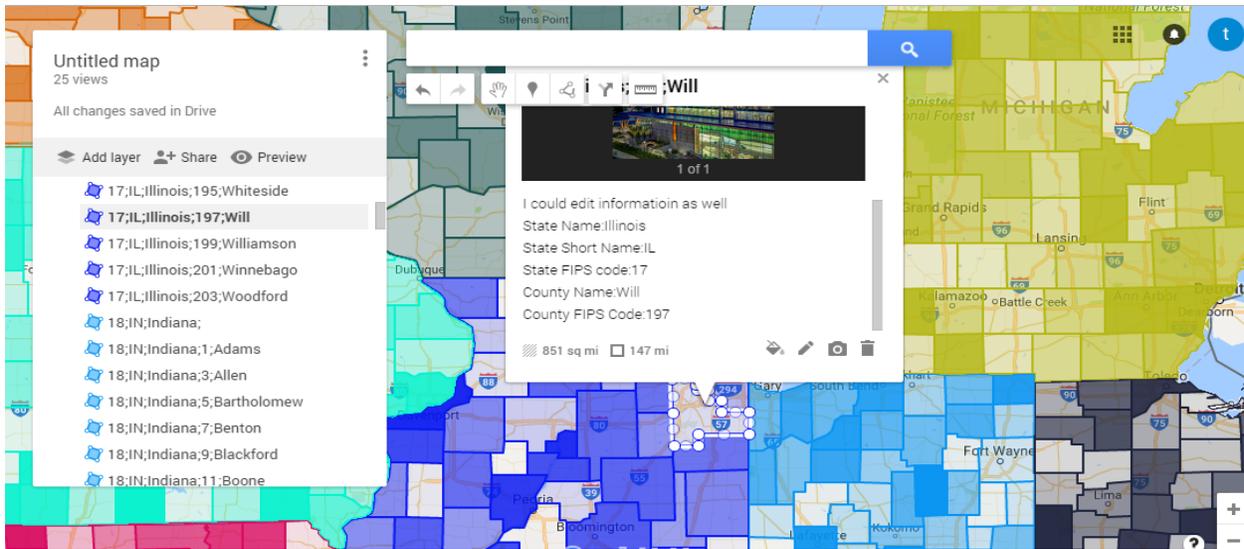**Figure 1. The Customized Midwest States**

Figure 2. Different Transparency on the Layer and Adding Photos to the Message Box

In this paper, you will get all the information you need to create the customized map shown in Figure 1 and Figure 2, including all the SAS codes for the macro and the input SAS data set. Besides I will show you how to add the KML file to the Google map.

THE INPUT DATA SET FOR THE MACRO

Figure 3 shows the structure of the input data set that can be used by the macro. The column "STATE" contains the state FIPS codes. The "STATECODE" column contains the state abbreviations. The "STATENAME" column contains the state name information. The column "COUNTY" contains the county FIPS codes. The "COUNTYNAME" column contains the county name information. The "description" column contains the text information for the layer, you can include the html tags into the text information, for e.g, in Figure 3, I've used the <br> tag to break the line. The "bgcolor" column contains the background color for the layer, you can search online to find the 6 digits hex color code for the background color. The "opacity" column contains the transparency percentage for each layer, the value range for this column is an integer number from 0 to 100.

	STATE	STATECODE	STATENAME	COUNTY	COUNTYNAME	description	bgcolor	opacity	
1	.		Illinois	.		State Name:Illinois State Short Name:IL State FIPS code:17	f21909	1	
2	18			.		State Name:Indiana State Short Name:IN State FIPS code:18	f29509	2	
3	.	IA		.		State Name:Iowa State Short Name:IA State FIPS code:19	cff209	3	
4	.		Kansas	.		State Name:Kansas State Short Name:KS State FIPS code:20	14b73a	4	
5	26			.		State Name:Michigan State Short Name:MI State FIPS code:26	14b7b4	5	
6	.	MN		.		State Name:Minnesota State Short Name:MN State FIPS code:27	0a67d8	6	
7	.		Missouri	.		State Name:Missouri State Short Name:MO State FIPS code:29	600ad8	7	
8	31			.		State Name:Nebraska State Short Name:NE State FIPS code:31	d50ad8	8	
9	.	ND		.		State Name:North Dakota State Short Name:ND State FIPS code:38	959E8F	9	
10	.		Ohio	.		State Name:Ohio State Short Name:OH State FIPS code:39	3E231E	10	
11	46			.		State Name:South Dakota State Short Name:SD State FIPS code:46	174f1a	11	
12	.	WI		.		State Name:Wisconsin State Short Name:WI State FIPS code:55	4e4f17	12	
13	.		Illinois	.	Adams	State Name:Illinois State Short Name:IL State FIPS code:17 County Name:Adams County FIPS Code:1	f21909	13	
14	17			3	Alexander	State Name:Illinois State Short Name:IL State FIPS code:17 County Name:Alexander County FIPS Code:3	f21909	14	

Figure 3. A Sample Input Data Set for the Macro

As you can see from the sample data set, you don't need to provide information for all the 3 columns "STATE", "STATECODE" and "STATENAME". As long as you provide at least one value for these 3 columns, the macro will know the state you want to use. The same rule applies to the column "COUNTY" and "COUNTYNAME", as long as at least one of them has value. For the values in the "STATECODE", "STATENAME", "COUNTY" and "COUNTYNAME" columns, they are case insensitive.

Below are the SAS codes I've used to create the sample input data set in Figure 3. In this input data set, it asks the macro to generate the state layers for all the Midwest states and all the county layers within those states.

```
proc sql;
create table samplestate as
select distinct state,statecode,statename from SASHELP.ZIPCODE where
STATECODE IN ("IL","IN","IA","KS","MI","MN","MO","NE","ND","OH","SD","WI");
create table samplecounty as
select distinct state,STATECODE,statename,county,COUNTYNM as COUNTYNAME
from SASHELP.ZIPCODE
where STATECODE IN
("IL","IN","IA","KS","MI","MN","MO","NE","ND","OH","SD","WI");
quit;

data sampledata;
set samplestate samplecounty;
length description $500 bgcolor $20.;
if COUNTYNAME ^="" then description=cats("State Name:",STATENAME,"<br>State
Short Name:",statecode,"<br>State FIPS code:",state, "<br>County Name:",
countyname,"<br>County FIPS Code:",county);
else description=cats("State Name:",STATENAME,"<br>State Short
Name:",statecode,"<br>State FIPS code:",state);
if statecode="IL" then bgcolor="f21909";
else if statecode="IN" then bgcolor="f29509";
else if statecode="IA" then bgcolor="cff209";
else if statecode="KS" then bgcolor="14b73a";
else if statecode="MI" then bgcolor="14b7b4";
else if statecode="MN" then bgcolor="0a67d8";
else if statecode="MO" then bgcolor="600ad8";
else if statecode="NE" then bgcolor="d50ad8";
else if statecode="ND" then bgcolor="959E8F";
else if statecode="OH" then bgcolor="3E231E";
else if statecode="SD" then bgcolor="174f1a";
else if statecode="WI" then bgcolor="4e4f17";
if mod(_n_,3)=1 then do;call missing(state,statecode);end;
else if mod(_n_,3)=2 then do;call missing(STATECODE,statename);end;
else if mod(_n_,3)=0 then do;call missing(state,statename);end;
if mod(_n_,3)=1 then do;call missing(county);end;
else if mod(_n_,3)=0 then do;call missing(countyname);end;
opacity=mod(_n_,100);
run;
```

THE CREATESTATECOUNTYKML MACRO

You can find the all the macro codes at the end of the paper. Immediately below is the structure of the macro:

```
%macro createStateCountyKML(libnm=work,datanm=,outputkml=);
```

- The "libnm" is used to indicate the library name for the input dataset.
- The "datanm" is used to indicate the input SAS dataset names.
- The "outputkml" is used to indicate the location to save the output KML file.

Below is an example showing you how to call the macro, the result KML file will be saved into "c:\sample.kml" file.

```
%createStateCountyKML(libnm=work,datanm=sampledata,outputkml=%str(c:\sample
.kml));
```

ADD KML FILE TO THE GOOGLE MAP

You can use the "Google My Maps" application to import your KML file to the Google map. Below are the steps to import the KML file:

1. Go to website https://www.google.com/mymaps

2. Use your gmail account to log in, if you don't have one or want to share the maps with others, you can create a new gmail account.

3. Click the "+ CREATE A NEW MAP" on the left top of the page, see the screenshot in Figure 4.
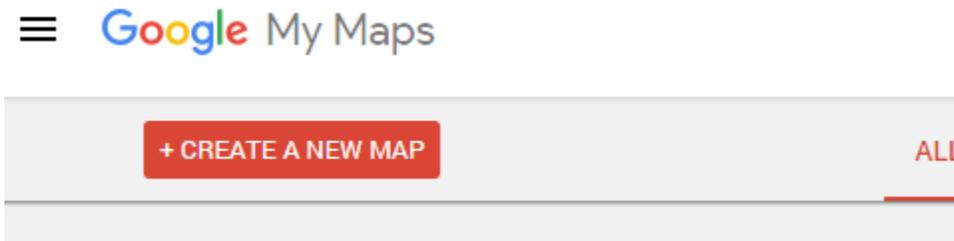

Figure 4. The Screenshot for the Main Page of the "Google My Maps"

4. Then it will link to a new Google map web page. On this page, click the "Import" link. See the screenshot in Figure 5.
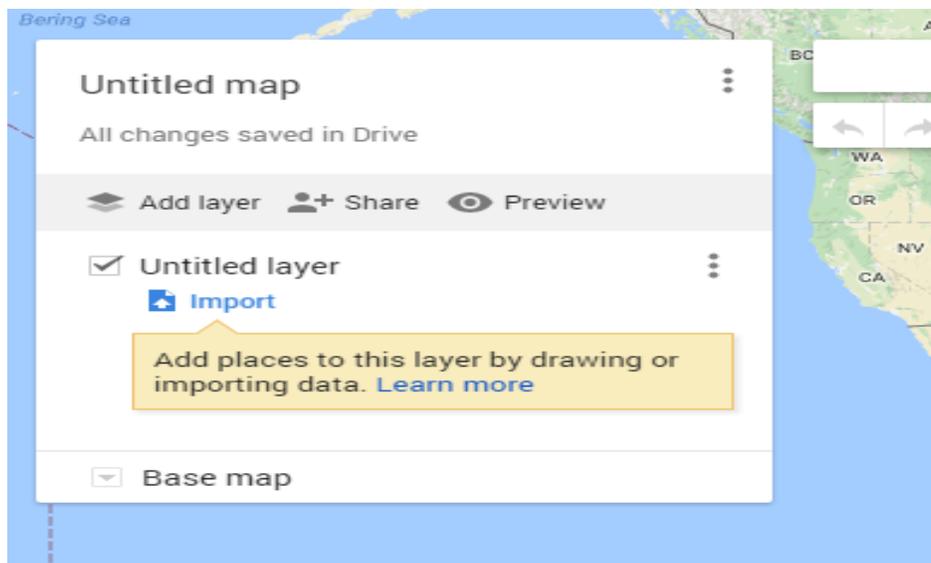

Figure 5. The Screenshot for the "Import" Link

5. Then you will see a box pop out, see the screenshot in Figure 6. You can just drag your KML file from your computer to this box area, the KML file will be imported to the Google map. See the screenshot in Figure 6. You can import more than one KML file to the Google map by click the "Add layer" link, see the link in Figure 7. You can also check or uncheck the layer checkbox to show or hide the layer.
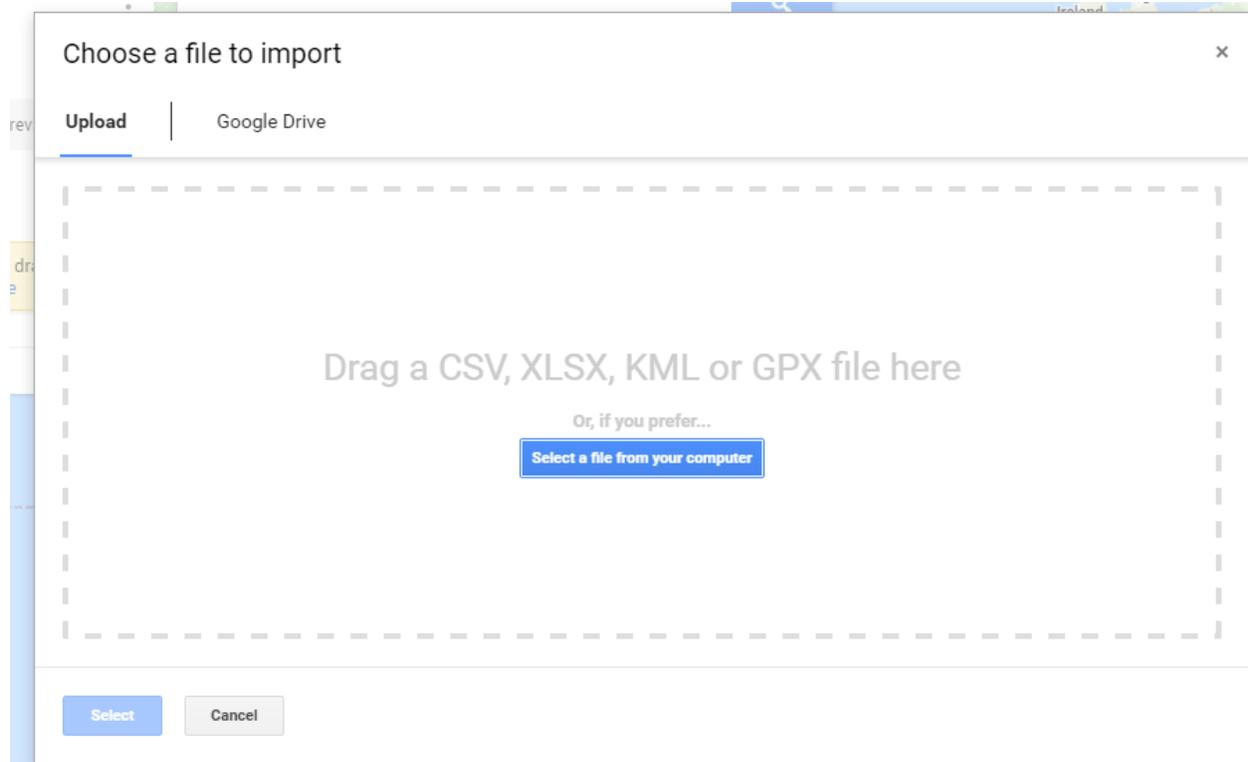

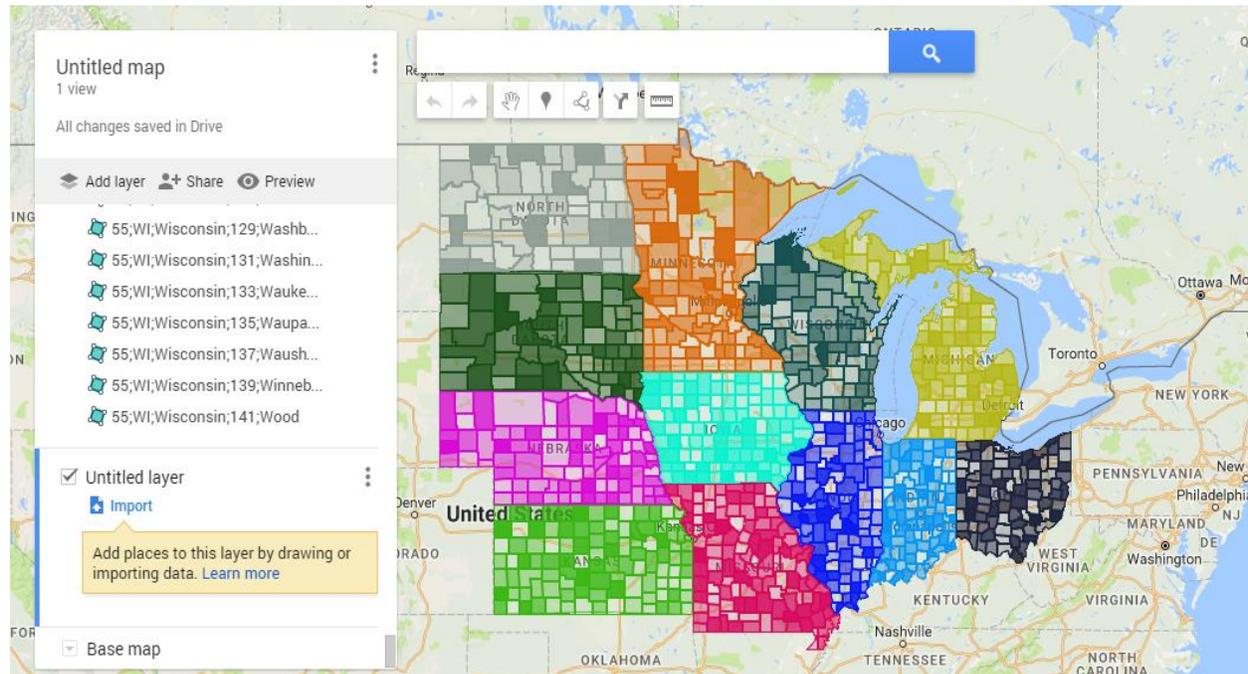**Figure 6. The Screenshot for the Box that Imports your KML File**

Figure 7. The Screenshot of the Imported KML File

There are many good benefits of using the "Google My Maps" application. One thing is that you can save the map file and share with other people. People can view your interactive maps online without installing any other software. You can learn more about how to use "Google My Maps" by doing the Google search.

THE CREATESTATECOUNTYKML MACRO

Presented below are the SAS codes for the createStateCountyKML macro.

```
%macro createStateCountyKML(libnm=work,datanm=,outputkml=);
*get county longtitude and latitude information;
data sascounty;
set maps.counties;
oldrowno=_n_;
run;
*get state longtitude and latitude information;
data sasstate;
set maps.states;
oldrowno=_n_;
run;

proc sql;
*get state name and county name information from SASHELP.ZIPCODE;
create table us_state_county_names as
select distinct STATE,STATECODE,STATENAME,COUNTY,COUNTYNM as COUNTYNAME
from SASHELP.ZIPCODE
order by STATE,STATECODE,STATENAME,COUNTY,COUNTYNM;

*create "state" data set which contains the state name information, the
coordinates
information for the each state, transform the x,y unprojected radian values
to the
degree values and save them into longitude and latitude because google maps
use
degree values;
create table state(drop=density) as
select s.*,STATECODE,STATENAME,
round((x*-(180/(gamma(0.5)**2))),0.000000001) as longitude,
round((y*(180/(gamma(0.5)**2))),0.000000001) as latitude,
catx(",",calculated longitude,calculated latitude,cats('0'," ")) as
coordinate
from sasstate(where=(x ^=. and y^=. and density<4)) as s left join
(select distinct STATE,STATECODE,STATENAME from us_state_county_names) as u
on s.STATE=u.STATE
order by oldrowno;

*add the state name information to the county;
create table county1 as
select c.*,STATECODE,STATENAME
from sascounty(where=(x ^=. and y^=. and density<4) ) as c left join
(select distinct STATE,STATECODE,STATENAME from us_state_county_names) as u
on c.STATE=u.STATE
order by c.state;

*add the county name information and transform the radian values to the
degree values;
create table county as
```

```
select c.*,COUNTYNAME,
round((x*-(180/(gamma(0.5)**2))),0.000000001) as longitude,
round((y*(180/(gamma(0.5)**2))),0.000000001) as latitude,
catx(",",calculated longitude,calculated latitude,cats('0'," ")) as
coordinate
from county1 as c left join
us_state_county_names as u
on c.STATE=u.STATE and c.COUNTY=u.COUNTY
order by oldrowno;

drop table county1;
quit;

*combine the state and county information together,
if no values for both county FIPS code and county name, it means this is a
state,
assing -999 values to these two fields;
data state_county_info;
set state(keep=state STATECODE statename)
county(keep=state STATECODE statename county countyname);
if cmiss(county,countyname)=2 then do;county=-999;countyname="-999";end;
run;
proc sort data=state_county_info nodupkey;by state STATECODE statename
county countyname;run;

*read the input data set, if no state information has been provided, will
delete this record
from the input data set. If no county information, then the macro will
consider this as a state
and adding -999 to both county and countyname columns;
data tmp;
set &libnm..&datanm.;
if cmiss(state,STATECODE,statename)=3 then delete;
if cmiss(county,countyname)=2 then do;county=-999;countyname="-999";end;

if opacity=0 then opacity=opacity + .0001;
opacity=((opacity/100)*256)- 1;
OP16=substr(put(opacity, hex.),7,2);
run;

*add the complete state name, state short name, state FIPS codes, county
name, county FIPS
codes to the input data set;
proc sql;
create table tmp1 as
select s.state,s.STATECODE,s.statename,s.county,s.countyname,
tranwrd(catx(";",s.state,s.STATECODE,s.statename,s.county,s.countyname),"-
999;-999","") as name,
description,bgcolor,OP16
from tmp as t,state_county_info as s
where (t.state=s.state or lowcase(t.STATECODE)=lowcase(s.STATECODE) or
lowcase(t.statename)=lowcase(s.statename))
and (t.county=s.county or lowcase(t.countyname)=lowcase(s.countyname))
order by s.state,s.STATECODE,s.statename,s.county,s.countyname;
quit;
```

```
*select the states from tmp1 and add the state coordinates info to those
states;
data tmp2state;
merge tmp1(in=a where=(county=-999)) state;
by state;
if a;
run;

*select the states from tmp1 and add the county coordinates info to those
states;
data tmp3county;
merge tmp1(in=a where=(county^=-999)) county;
by state county;
if a;
run;

data tmp2state;
set tmp2state;
by state segment;
length kmlcodes $700.;
retain count;
if first.segment then do;
kmlcodes=cats("<Placemark><name>",name,"</name>",

"<Style><PolyStyle><color>",cats(OP16,bgcolor),"</color></PolyStyle></Style
>",
  "<description>",
  "<![CDATA[<center><table>",
  "<tr bgcolor='#E3E3F3'><td>",description,"</td></tr>",
  "</table></center>]]>",

"</description><Polygon><tessellate>1</tessellate><outerBoundaryIs><LinearR
ing><coordinates>");
count=1;
output;
end;
kmlcodes=coordinate;count=count+1;output;
if last.segment then do;
count=count+1;
kmlcodes=cats("","</coordinates></LinearRing></outerBoundaryIs></Polygon></
Placemark>");
output;
end;
keep state STATECODE statename segment kmlcodes count;
attrib _all_ label=' ';
run;

data tmp3county;
set tmp3county;
by state COUNTY segment;
length kmlcodes $700.;
retain count;
if first.segment then do;
kmlcodes=cats("<Placemark><name>",name,"</name>",

"<Style><PolyStyle><color>",cats(OP16,bgcolor),"</color></PolyStyle></Style
>",
```

```
  "<description>",
  "<![CDATA[<center><table>",
  "<tr bgcolor='#E3E3F3'><td>",description,"</td></tr>",
  "</table></center>]]>",

"</description><Polygon><tessellate>1</tessellate><outerBoundaryIs><LinearR
ing><coordinates>");
count=1;
output;
end;
kmlcodes=coordinate;count=count+1;output;
if last.segment then do;
count=count+1;
kmlcodes=cats("","</coordinates></LinearRing></outerBoundaryIs></Polygon></
Placemark>");
output;
end;
keep state STATECODE statename county countyname segment kmlcodes count;
attrib _all_ label=' ';
run;

*add all the coordinates together;
data tmp4;
set tmp2state tmp3county;
rowno=_n_;
run;
proc sort data=tmp4;by STATE COUNTY segment count;run;

*the top part of the kml file header;
data topkml;
length kmlcodes $700.;
kmlcodes='<?xml version="1.0" encoding="UTF-8"?>';
output;
kmlcodes='<kml xmlns="http://earth.google.com/kml/2.0">';
output;
kmlcodes='<Document>';
output;
run;

*the ending part of the kml file;
data endkml;
length kmlcodes $700.;
kmlcodes='</Document></kml>';
run;

data _null_;
set topkml tmp4 endkml;
file "&outputkml." notitles noprint;
put kmlcodes;
run;
%mend;
```

CONCLUSION

The macro presented in this paper can be used as a helpful tool to create the KML file for any US states

and counties. Once you understand this macro, you can further extend its function to create other KML files for certain areas.

REFERENCES

1. Azimaee,Mahmoud. April 2010. "KML Macro: Integrating SAS® and Google API and Its Application in Mapping Manitoba's Health Data on Google Earth and Google Map". Proceedings of the SAS Global 2010 Conference, Seattle, WA. Available at

   http://support.sas.com/resources/papers/proceedings10/220-2010.pdf

ACKNOWLEDGMENTS

CONTACT INFORMATION

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

Ting Sa
Division of Biostatistics and Epidemiology, Cincinnati Children's Hospital Medical Center
513-636-3674
Ting.Sa@CCHMC.ORG