Paper 8540-2016

Working with PROC SPP, PROC GMAP and PROC GINSIDE to Produce Nice Maps

Alan Ricardo da Silva, Universidade de Brasília, Dep. de Estatística, Brazil

ABSTRACT

This paper aims to show how to create maps from the output of new PROC SPP using the same layer of the data. This cut can be done by PROC GINSIDE and the final plot can be drawn by PROC GMAP. The result is a nice map and nicer than the plot produced by PROC SPP.

INTRODUCTION

The new PROC SPP (Spatial Point Pattern) deal with spatial data, which are a collection of locations of single events of a spatial process (SAS, 2014). It is possible to use PROC SSP to create a surface of the intensity of the point pattern process. The problem is that PROC SPP generates data only for a squared area, even the data are bordered by an irregular area. This paper describes how to use PROC SPP, PROC GMAP and PROC GINSIDE to produce an intensity map for a non-squared area.

SPATIAL POINT PATTERN

The first analysis in order to characterizing the intensity of the data points in an area can be done by a kernel estimator of the intensity function. The general form of this kind of estimator is given by (Cressie, 1991):

$$\hat{\lambda}_h(\boldsymbol{s}) = \frac{1}{\rho_h(s)}\{\sum_{i=1}^{n} k_h(\boldsymbol{s} - \boldsymbol{s_i})\} \quad (1)$$

where $k_h(.)$ is a probability density (kernel) function symmetric about the origin, $h > 0$ determines the amount of smoothing (also known as bandwidth), $\boldsymbol{s}$ is the spatial locations of *n* events and $\rho_h(\boldsymbol{s})$ is an edge correction factor.

PROC SPP compute the nonparametric intensity estimate as (SAS, 2014)

$$\hat{\lambda}_h(\boldsymbol{s}) = \frac{1}{\rho_h(\boldsymbol{s})}\left\{\sum_{i=1}^{n} h^{-2} k_h\left(\frac{s-s_i}{h}\right)\right\} \quad (2)$$

The distance between points can be computed by Euclidian distance as:

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (3)$$

where x_i, x_j are the x-coordinates (longitude) and y_i, y_j are the y-coordinates (latitude).

Some options for the kernel function are Gaussian, uniform, rectangular and quartic (Silverman, 1986).

The Gaussian kernel function is given by:

$$K(d) = \frac{e^{\frac{-d^2}{2h^2}}}{\sqrt{2\pi}} \quad (4)$$

ILLUSTRATION

To illustrate how to use PROC SPP and PROC GINSIDE, let us use an irregular shape from the Canchim farm (EMBRAPA) in São Carlos, São Paulo, Brazil, as shown by Figure 1. There are 85 data referring to the clay content. The shape file (*.shp) can be imported by PROC MAPIMPORT.

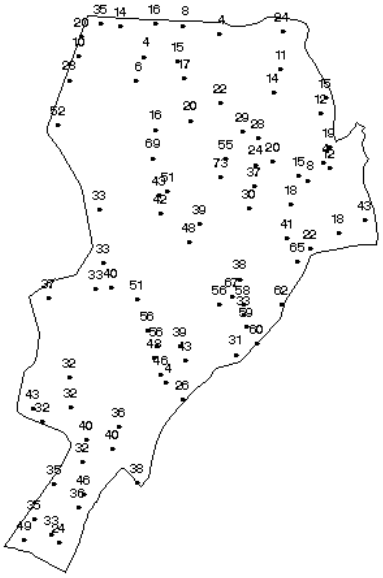

Figure 1. Clay content in Canchim farm (EMBRAPA) in São Carlos, São Paulo, Brazil.

First, one can use PROC SQL to select the borders of the area named MINX (the lower left limit for the x-coordinate), MINY (the lower left limit for the y-coordinate), MAXX (the upper right limit for the x-coordinate), MAXY (the upper right limit for the y-coordinate).

```
proc sql noprint;
 select min(x) into:minx from sao_carlos;
 select min(y) into:miny from sao_carlos;
 select max(x) into:maxx from sao_carlos;
 select max(y) into:maxy from sao_carlos;
quit;
%put minx=&minx maxx=&maxx miny=&miny maxy=&maxy;
```

After that, one can use that information about the borders of the area in the AREA= option of the PROCESS statement of PROC SPP. The b= option referred to the kernel bandwidth parameter of the kernel first-order intensity estimates and GRID= specifies a reference grid for computing the kernel estimate.

```
proc spp data=sao_carlos_pt plots(equate)=(trends observations);
   process AVG_Z = (x, y /area=(&minx,&miny,&maxx,&maxy) Event=AVG_Z) /
   kernel(type=gaussian b=500 out=kernel grid(90,90));
run;
```

To plot the results, one can use the ANNOTATE Facility from the dataset generated by KERNEL sub-option OUT= and PROC GMAP with ANNO= option in the CHORO statement. Just remember to rename the variables

GXC and GYC to X and Y, respectively. Using SIZE=0.5 (small squares) we can see how the coordinates are distributed on the area (Figure 2 - left) and using SIZE=1 (large squares) we can see these coordinates in the “continuous way” on the area (Figure 2 - right). The squares are so large that appears to fill the entire area as one.

```
data anno;set kernel(rename=(GXC=x GYC=y));
 length function style $10. color $8.;
 retain line 1 xsys ysys '2' hsys '3' color 'red';
 function='label';text='U';position='5';style='marker';
 size=1;
run;
proc gmap data=a map=sao_carlos all;
 id segment;
 choro v / anno=anno nolegend;
run;
quit;
```

Figure 2. Plot of the coordinates generated by PROC SPP.

Finally, to plot the continuous surface one can use the program below to color each coordinate (square created by the ANNOTATE Facility) and to create a continuous bar. This job can be done with %colorscale macro (SAS, 2003) with some adaptations. This macro is on Appendix I.

```
proc sql noprint;
 select count(*) into:n from anno;
 select min(ESTIMATE) into:min from anno;
```

```
 select max(ESTIMATE) into:max from anno;
quit;

/*defining the number of classes*/
data _null_;
 nc=floor(1+3.3*log10(&n))-1;
 call symput('nc',nc);
run;
%put &nc;
/*checking the number of coordinates, the min value, the maximum value and the
amplitude of each class*/
%put &n &min &max %sysevalf((&max-&min)/&nc);

%include 'TYPE-YOUR-PATH\colorscaleMACRO.sas';

%colorscale(FFFFFF,,FF0000,&nc,clist,no);%patt;

/*creating a variable macro for the color of each class*/
data _null_;set clist;
 call symput('color'||trim(left(_n_)),'cx'||rgb);
run;

/*assigning a color for each class*/
%macro cl;
data anno;set anno;
 if ESTIMATE<=&min+(&max-&min)/&nc then color="&color1";
 %do i=2 %to &nc;
  else if ESTIMATE<=&min+&i*(&max-&min)/&nc then   color="&&color&i";
 %end;
 if ESTIMATE=0 then color="white";
run;
%mend cl;
%cl;

/*defining the continuous bar*/
%bar(FF3333,FFFFFF,&min,&max,vertical,y_i=44,x_i=80);

data anno;set _a_ anno;
```

```
proc gmap data=a map=sao_carlos all anno=anno_points;
 id segment;
 choro v / anno=anno nolegend;
run;
quit;
```

Figure 3 (left) shows the result of the Kernel first-order intensity estimates for the squared area and Figure 3 (right) shows the final result of the intensity estimates only for the coordinates which are inside the polygon (Figure 1). PROC GINSIDE generates these results as follows:

```
proc ginside data=anno map=Sao_carlos out=anno2 insideonly;
 id segment;
 run;

data anno2;set _a_ anno2;

proc gmap data=a map=sao_carlos all anno=anno_points;
 id segment;
 choro v / anno=anno2 nolegend;
run;
quit;
```

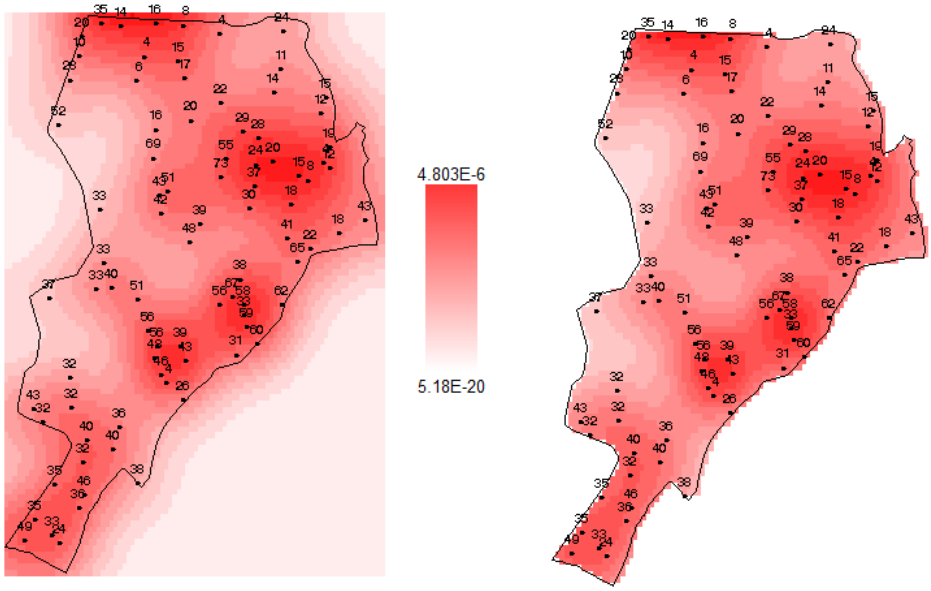

Figure 3. Kernel first-order estimates for the squared area (left) and for the polygon area (right).

Note that Figure 3 presents the same result of the plot generated by PROC SPP (Figure 4) but the former is really nicer than the last.

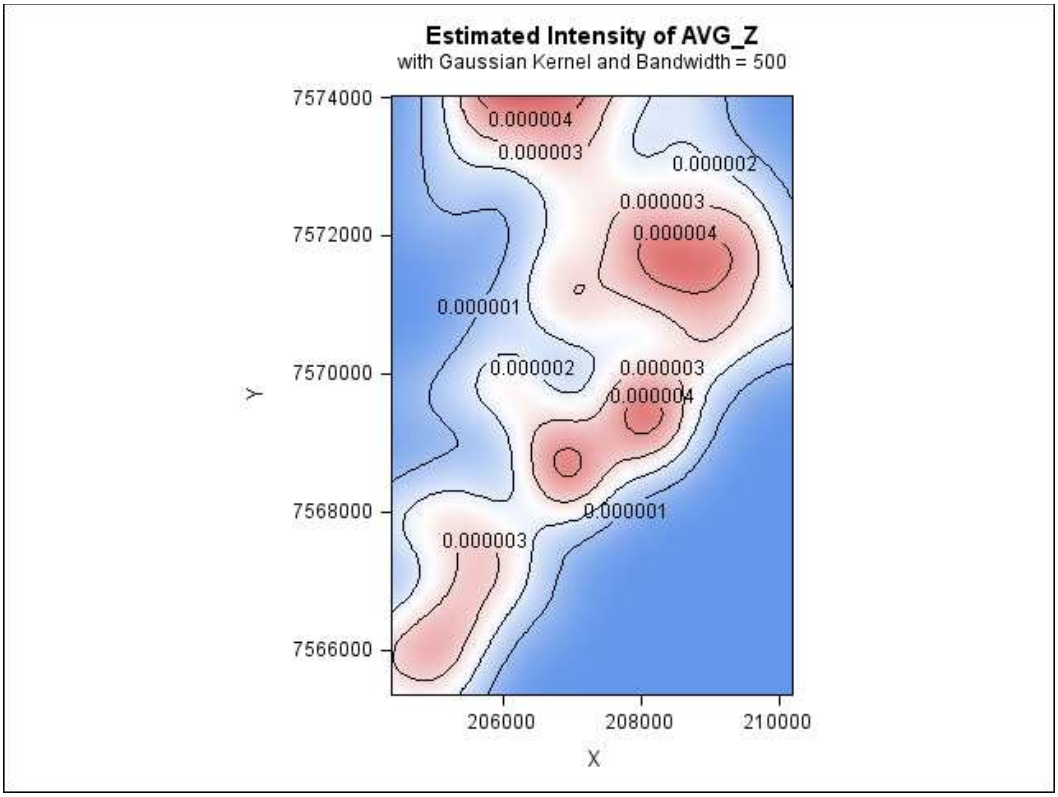

Figure 4. PROC SPP plot for the Kernel first-order estimates.

CONCLUSION

This paper showed how to generate the Kernel first-order intensity maps using the new PROC SPP, PROC GMAP and PROC GINSIDE. The `%colorscale` macro also has been used to color the continuous map and to create a continuous bar as a legend, and the final plot can be considered a nicer map than that produced by PROC SPP, because the final map is created using the same layer of the data, i.e., using an irregular area instead of a squared area.

REFERENCES

SAS 2003. “Determine a list of colors in a gradient - colorscale macro”. Accessed July 10, 2009. ftp://ftp.sas.com/techsup/download/sample/graph/other-color.html https://github.com/friendly/SAS-macros/blob/master/colorscale.sas.

SAS (2014). SAS Institute Inc. 2014. Base SAS® 9.4 Procedures Guide. Cary, NC: SAS Institute Inc.

Cressie, N. A. C. 1991. *Statistics for Spatial Data*. USA:Wiley-Interscience Inc.

Silverman, B. W. 1986. *Density Estimation for Statistics and Data Analysis*. New York: Chapman & Hall.

CONTACT INFORMATION

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

Name: Alan Ricardo da Silva
Enterprise: Universidade de Brasília
Address: Campus Universitário Darcy Ribeiro, Departamento de Estatística, Prédio CIC/EST sala A1 35/28
City, State ZIP: Brasília, DF, Brazil, 70910-900
Work Phone: +5561 3107 3672
E-mail: alansilva@unb.br
Web: www.est.unb.br

APPENDIX I – %COLORSCALE MACRO

```
/*************************************************************/
/*     The COLORSCALE macro can be used to determine a list of      */
/*     colors in a gradient.  The TOP and BOTTOM colors are         */
/*     required; a middle color is optional.  The value N sets the  */
/*     desired number of intermediate colors.  For example,  if N   */
/*     is 10 and no middle color is specified, 12 colors are shown  */
/*     in the output.  If a middle color is specified, 13 colors    */
/*     would be shown in the output.                                */
/*                                                               */
/*     The macro takes the following parameters:                    */
/*                                                               */
/*      TOP: color displayed on top of the output                */
/*   MIDDLE: optional middle color; the gradient is              */
/*           forced through this color                           */
/*   BOTTOM: color displayed on the bottom of the output         */
/*        N: the number if intermediate colors                   */
/*      DSN: name of the dataset that stores the colors.         */
/*           The variable RGB contains the color values,         */
/*           the variable NUMCOL contains the number             */
/*           of colors.                                          */
/*   SWATCH: if "Y", display a sample of the colors.             */
/*                                                               */
/*   Colors should be represented as RGB hex values, such as     */
/*   FFFFFF for white or 000000 for black.  See Technical        */
/*     Support document TS-688 for more information.                */
/*                                                               */
/*     This macro uses the INCR macro, below, to calculate the      */
/*     intermediate color values.                                   */
/*                                                               */
/*     Because values must be rounded, slightly different results   */
/*     may occur if the values for the top and bottom colors are    */
/*     reversed.  If the last intermediate color seems to 'jump'    */
/*     from the top or bottom color, try reversing the values for   */
/*     the top and bottom colors.                                   */
/*                                                               */
/*     When invoking the macro, remember that the parameters are    */
/*     positional. If no middle color is specified, the comma       */
/*     should remain:  %colorscale(000000,,FFFFFF,3,anno);          */
```

```
/*                                                                  */
/*                                                Revised 20SEP02   */
/*******************************************************************/

%macro colorscale(top,middle,bottom,n,dsn,swatch);
/* If there is a middle color,                                      */
%if "&middle" NE "" %then %do;
/* set the number of intermediate colors between each section       */
/* (top and middle or bottom and middle). If N is even, N/2         */
/* intermediate colors are calculated; if N is odd, (N-1)/2         */
/* intermidiate colors are calculated. The customer is warned       */
/* of this change below.                                            */
       data _null_;
             if mod(&n,2)=0 then call symput('halfn',left(put(&n/2,5.)));
             else call symput('halfn',left(put((&n-1)/2,5.)));
       run;
/* Calculate the top to middle and middle to bottom                 */
/* ranges, and combine these into the T2B data set.                 */
  %incr(&top,&middle,t2m,&halfn,n);
       %incr(&middle,&bottom,m2b,&halfn,y);
       data t2b;
         set t2m m2b;
       run;
/* Determine the size of the bars for the annotation,               */
/* and warn the user if N-1 values were calculated instead of       */
/* N values.                                                        */
       data _null_;
       if mod(&n,2) EQ 0 then call symput('draw',left(put(&n+3,6.)));
       else do;
       call symput('draw',left(put(&n+2,6.)));
             put "Warning:  You have entered an odd value: &n..";
    put "          %eval(&n-1) colors were created instead.";
       end;
run;
%end;
/* If there is no middle color,                                     */
%else %do;
/* Calculate the N intermediate colors between TOP and BOTTOM,      */
  %incr(&top,&bottom,t2b,&n,y);
/* and determine the size of the bars for the annotation.           */
```

```
       data _null_;
       call symput('draw',left(put(&n+2,6.)));
       run;
%end;

%if %upcase("&swatch")="Y" or %upcase("&swatch")="YES" %then %do;
/* The T2B dataset now contains the original and intermediate  */
/* colors, in order of creation.  This annotation will display */
/* both a color swatch and the RGB hex value for each color.   */
data swatch;
 length color function $8. style $10.;
 retain xsys ysys '1' when 'b' ;
 set t2b;
 if _n_=1 then y=98;
 function='move';x=0;output;
 function='bar';x=60;y+-floor(100/(&draw));style='solid';color="cx"||rgb;output;
function='label';x=62;position='3';text=rgb;size=1;color='black';style='SWISS';output;
run;

/* Create an image with the annotation.  To export this image, */
/* use a GOPTIONS and FILENAME statement before invoking       */
/* the macro.                                                  */
goptions cback=white;
proc gslide anno=swatch;
title1 j=r "Sample" j=r "of" j=r "Colors";
run;quit;
title1;
%end;
data &dsn;
 set t2b;
 retain numcol;
 numcol=&draw;
 keep rgb numcol;
 if rgb='FFFFFF' then delete;
 run;
%mend;



%macro incr(start,end,dsn,div,lc);
```

```
data &dsn;
length start end sred ered sblue eblue sgreen egreen $6.;
/* Get the colors.                                              */
start=upcase("&start");
end=upcase("&end");
/* Find the starting and ending values for Red, Green, and Blue. */
 sred=substr(start,1,2);
 ered=substr(end,1,2);
 sgreen=substr(start,3,2);
 egreen=substr(end,3,2);
 sblue=substr(start,5,2);
 eblue=substr(end,5,2);
/* Calculate the increment for Red, Green, and Blue.            */
incrn=(input(ered,hex2.)-input(sred,hex2.))/(&div+1);
incgn=(input(egreen,hex2.)-input(sgreen,hex2.))/(&div+1);
incbn=(input(eblue,hex2.)-input(sblue,hex2.))/(&div+1);

/* Determine the direction of the value, either increasing      */
/* increasing or decreasing.                                    */
if incrn<0 then multr=-1;else multr=1;
incr=left(put(incrn*multr,hex2.));
if incgn<0 then multg=-1;else multg=1;
incg=left(put(incgn*multg,hex2.));
if incbn<0 then multb=-1;else multb=1;
incb=left(put(incbn*multb,hex2.));
/* Add the start color to the output.                          */
rgb=start;output;
/* Calculate the incremental values for red, green, and blue   */
/* and combine them into a single value, RGB.                  */
do i=1 to &div;
 red=input(sred,hex2.)+input(incr,hex2.)*i*multr;
 green=input(sgreen,hex2.)+input(incg,hex2.)*i*multg;
 blue=input(sblue,hex2.)+input(incb,hex2.)*i*multb;
 rgb=put(red,hex2.)||put(green,hex2.)||put(blue,hex2.);
 output;
end;
/* Add the last color to the output.  This step is             */
/* not performed for the T2M dataset; otherwise T2M and        */
/* M2B would each write out the color, creating two bands      */
/* of the middle color in the final output.                    */
```

```
if upcase("&lc")="Y" then do;
       rgb=end;
       output;
end;
run;
%mend;

*green
%colorscale(FFFFFF,,078800,2,clist,yes);
*blue
%colorscale(FFFFFF,,000888,8,clist,yes);
*blue-red
%colorscale(FF0000,FFFFFF,0000FF,8,clist,yes);
*red
%colorscale(FFFFFF,,FF3333,8,clist,yes);
*gray
%colorscale(FFFFFF,,333b33,8,clist,yes);
*orange
%colorscale(FFFFFF,,FF8000,8,clist,yes);
*yellow
%colorscale(FFFFFF,,FFFF57,8,clist,yes);

/* Create the PATTERN statements. */
%macro patt;
data _null_;
set clist;
call symput('color'||left(put(_n_,3.)),'cx'||rgb);
call symput('total',left(put(numcol,3.)));
run;
%do i=1 %to &total-1;
  pattern&i v=s c=&&color&i;
%end;
%mend;

%macro bar(end,start,tinicial,tfinal,direcao,y_i=,x_i=);
data _a_;
   length color function style $ 8 text $ 20;
   retain xsys ysys '3' when 'A' style 'S';
   start=upcase("&start");
   end=upcase("&end");
```

```
   /* Find the starting and ending values for Red, Green, and Blue. */
   sred=substr(start,1,2);
   ered=substr(end,1,2);
   sgreen=substr(start,3,2);
   egreen=substr(end,3,2);
   sblue=substr(start,5,2);
   eblue=substr(end,5,2);
   /* Calculate the increment for Red, Green, and Blue.            */
   incrn=(input(ered,hex2.)-input(sred,hex2.))/100;
   incgn=(input(egreen,hex2.)-input(sgreen,hex2.))/100;
   incbn=(input(eblue,hex2.)-input(sblue,hex2.))/100;
   /* Determine the direction of the value, either increasing      */
   /* increasing or decreasing.                                    */
   if incrn<0 then multr=-1;else multr=1;
   incr=left(put(incrn*multr,hex2.));
   if incgn<0 then multg=-1;else multg=1;
   incg=left(put(incgn*multg,hex2.));
   if incbn<0 then multb=-1;else multb=1;
   incb=left(put(incbn*multb,hex2.));
   %if %upcase(&direcao)=HORIZONTAL %then %do;
x =  25; xinc = 0.3;
   %end;
   %if %upcase(&direcao)=VERTICAL %then %do;
Y =  &y_i; yinc = 0.3;
   %end;
/* Add the start color to the output.                           */
rgb=start;output;
/* Calculate the incremental values for red, green, and blue    */
/* and combine them into a single value, RGB.                   */
do i=1 to 99;
red=input(sred,hex2.)+input(incr,hex2.)*i*multr;
green=input(sgreen,hex2.)+input(incg,hex2.)*i*multg;
blue=input(sblue,hex2.)+input(incb,hex2.)*i*multb;
rgb=put(red,hex2.)||put(green,hex2.)||put(blue,hex2.);
color='cx'||rgb;
function = 'MOVE';
%if %upcase(&direcao)=HORIZONTAL %then %do;
      y = 2;
      output;
      function = 'BAR';
```

```
      x + xinc;
      y = 7;
%end;
%if %upcase(&direcao)=VERTICAL %then %do;
      x = &x_i;%let xi=&x_i;
      output;
      function = 'BAR';
      y + yinc;
      x = &x_i+4;%let xf=%sysevalf(&x_i+4);
%end;
output;
end;
   function = 'LABEL';
   Style = "Arial";
   Text = "&tfinal";
   color='black';
   Position = '5';
%if %upcase(&direcao)=HORIZONTAL %then %do;
   x + xinc;
   y = 4.5;
%end;
%if %upcase(&direcao)=VERTICAL %then %do;
   y + yinc + 2;
   x = (&xi+&xf)/2;
%end;
   output;
   Text = "&tinicial";
   Position = '5';
%if %upcase(&direcao)=HORIZONTAL %then %do;
   x = 24.5;
   y = 4.5;
%end;
%if %upcase(&direcao)=VERTICAL %then %do;
   x = (&xi+&xf)/2;
   y = &y_i-2;
%end;
   output;
run;
%put &xi &xf;
%mend bar;
```