Paper 2481-2018

Creating a Spatial Weights Matrix for the SPATIALREG Procedure

Alan Ricardo da Silva, Universidade de Brasília, Departamento de Estatística, Brazil

ABSTRACT

The purpose of this paper is to show a macro for creating a spatial weights matrix for the SPATIALREG procedure. The new PROC SPATIALREG estimates spatial regression models by including in its structure the effect of spatial proximity, represented in an $n \times n$ matrix known as **W** matrix. However, this **W** matrix cannot be created directly in the procedure. Thus, this paper will show some ways to do that using Hash Objects and PROC IML.

INTRODUCTION

The new PROC SPATIALREG available in SAS 9.4M4 estimates spatial regression models (and also the classical regression model) proposed by Anselin (1988). The general form of these models is (Anselin, 1988):

$$\begin{aligned} \boldsymbol{y} &= \rho \boldsymbol{W_1 y} + \boldsymbol{X\beta} + \boldsymbol{u} \\ \boldsymbol{u} &= \lambda \boldsymbol{W_2 u} + \boldsymbol{\epsilon} \end{aligned} \qquad (1)$$

where $\boldsymbol{y}$ is an $n \times 1$ vector of dependent variable, $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ are $n \times n$ matrices of spatial dependence, $\boldsymbol{X}$ is an $n \times k$ matrix of explicative variables, $\boldsymbol{\beta}$ is a $k \times 1$ vector of parameters of the model, $\boldsymbol{u}$ is an $n \times 1$ vector of error, $\boldsymbol{\epsilon}$ is an $n \times 1$ vector of the remainder error and ρ and λ are the spatial effects parameters of the model. Note that Equation (1), also known as SAC (Spatial Autoregressive Confused) model is the same classical regression model when ρ and λ are both equal to 0, i.e, when there is no spatial dependence. Others models such as SAR (Spatial AutoRegressive Model) and SEM (Spatial Error Model) are derived from (1) just letting $\lambda = 0$ and $\rho = 0$, respectively. More details about spatial regression models and how to use PROC COUNTREG with spatial components can be found in Wu and Chvosta (2016) and how to use PROC SPATIALREG can be found in Wu and Chvosta (2017).

To estimate these spatial regression models is required to declare in the statement `WMAT=` the spatial weights matrix **W**, which is not created automatically by the SPATIALREG Procedure. Thus, this paper aims to show how creating some possible forms for the spatial weights matrix in order to estimate these spatial regression models.

The paper is organized as follows. In Section 2 is given some aspects of the spatial weights matrix and in Section 3, a SAS® macro is presented. In Section 4 is introduced an illustration and the conclusions are in Section 5.

SPATIAL WEIGHTS MATRIX

The spatial weights matrix, also known as **W** matrix, plays an important part in the spatial modeling. Its purpose is representing, quantitatively, the spatial structure between regions. For n regions, $R_1, R_2, \dots, R_n$, the elements w_{ij} of the $n \times n$ **W** matrix, are representing some proximity measure between regions R_i and R_j, $i, j = 1,2, \dots, n.$, and by definition, its diagonal w_{ii} is zero.

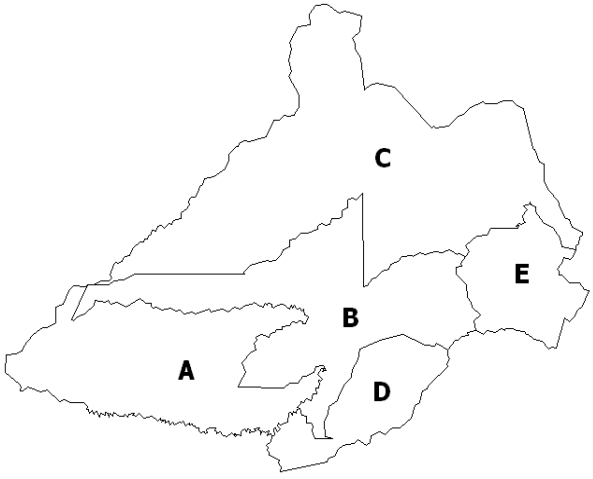
Some choices for measure the spatial proximity are:

1) $w_{ij} = 1$, if R_i shares border with R_j, and $w_{ij} = 0$ otherwise;
2) $w_{ij} = 1$, if R_i is one of the Nth nearest regions of R_j, and $w_{ij} = 0$ otherwise;
3) $w_{ij} = 1$, if R_i is far from R_j by a distance of K miles (or kilometers), and $w_{ij} = 0$ otherwise;
4) $w_{ij} = 1/(1 + d_{ij})$, where d_{ij} is the distance between regions R_i and R_j. In this case it is necessary to force $w_{ii} = 0$; or you can just use $w_{ij} = 1/d_{ij}$, for $d_{ij} > 0$.

5) $w_{ij} = 1/(1 + t_{ij})$, where t_{ij} is the time travel between regions R_i and R_j. In this case it is necessary to force $w_{ii} = 0$; or you can just use $w_{ij} = 1/t_{ij}$, for $t_{ij} > 0$.

The choices 4) and 5) are more realistic if they are computed using the network of highways, in order to capture road and/or traffic conditions, respectively.

An example of the choice 1), also known as contiguity matrix, is in Figure 1. The representation has 5 regions, and it was attributed 1 for those regions which share borders and 0 otherwise. Note that region **B** shares borders with all others regions.



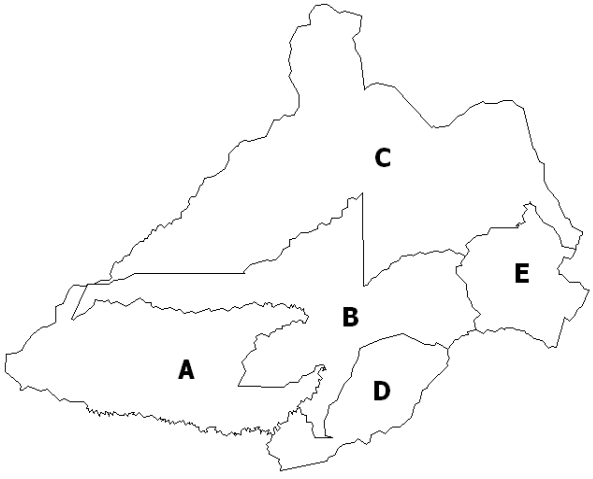
$$\mathbf{W} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 1. Example of a Contiguity Matrix

It is a common practice to use a row-standardized matrix instead of binary matrix, i.e., each row of this new matrix is summing to 1. To do that, just compute each entrance of the matrix as:

$$\boldsymbol{w_{ij}^{*}} = \frac{\boldsymbol{w_{ij}}}{\sum_{j=1}^{n} \boldsymbol{w_{ij}}} \qquad (2)$$

The row-standardized matrix for Figure 1 is in Figure 2. We can see that the region **B** has 4 neighbors and because of that, the row-standardized matrix has values 0.25 for the neighbors, and the other regions has values 0.5 because they share border with only 2 regions.



$$\mathbf{W_s} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 0.5 & 0 & 0.5 & 0 \\ 0.25 & 0 & 0.25 & 0.25 & 0.25 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 2. Example of a Row-Standardized Contiguity Matrix

SAS® MACRO

The SAS® macro ***%WMATRIX*** has been developed using a Hash objects program proposed by Babcock (2010) to find the neighborhood of the regions and PROC IML to construct an $n \times n$ matrix. The general call is:

```
%WMATRIX(map=, id=, lat=, long=, type=contiguity, centroid_lat=,
centroid_long=, distance=, neighbors=, out=neighbor);
```

- `map` = specifies the data set with the map coordinates (latitude, longitude and ID).
- `id` = specifies the variable with the codes for the distinct regions.
- `lat` = specifies the latitude or *y* axis variable.
- `long` = specifies the longitude or *x* axis variable.
- `type` = specifies the type of spatial weights matrix (contiguity or distance). The default value is contiguity.
- `Centroid_lat` = specifies the latitude or *y* axis variable for the centroid of the region.
- `Centroid_long` = specifies the longitude or *x* axis variable for the centroid of the region.
- `distance` = specifies the distance to be considered a neighbor, following choice 3) of second section.
- `neighbors` = specifies the *N*th nearest neighbors, following choice 2) of second section.
- `out` = specifies the output data set.

The variables `map, id, lat` and `long` are mandatory. The others are optional and/or have default values.

The main idea of this macro is creating the spatial weights matrix from the coordinates map. For the contiguity matrix, the program searches for at least one common point between regions, and for distance matrix, the program computes the Euclidian distance between the centroids of the regions. In addition, one can specify other coordinates to compute the Euclidian distance (see Wu and Chvosta (2017)) such as the location of the city hall, or the main plaza by using the variables `Centroid_lat` and `Centroid_long`.

Four data sets are created by ***%WMATRIX*** macro: a data set called NEIGHBOR by default (or `&out` depending of the name assigned above); a data set called WMATRIX, which is the $n \times n$ spatial weights matrix; a data set called WMATRIX_SDR, which is the $n \times n$ row-standardized spatial weights matrix; and a data set called WMATRIX_COMPACT, which is the compact representation of the $n \times n$ spatial weights matrix of size $n \times 3$, supported by PROC SPATIALREG and used to save memory (see Wu and Chvosta (2017)).

ILLUSTRATION

To illustrate how the **%WMATRIX** macro can be used, let us use the data set CRIMEOH (Example 32.1: Columbus Crime Data of the SPATIALREG Procedure), which contains data in 49 neighborhoods of Columbus, Ohio, about the number of crimes and possible determinants of crime. This data set is originally obtained from Anselin (1988). Figure 3 shows the shape of Columbus (note that the figure is the same of that presented in Anselin (1988), page 188) and the macro call for creating the contiguity matrix for this data set is:

```
%WMATRIX(map=columbus, id=code, lat=y, long=x, type=contiguity,
out=neighbor_columbus);
```

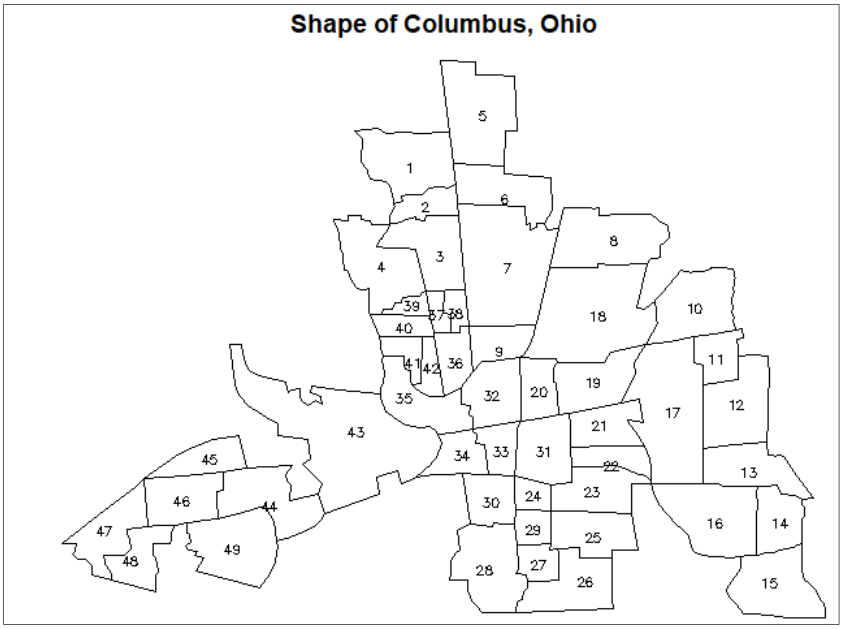


Figure 3. Shape of Columbus, Ohio.

The generated output from **%WMATRIX** macro can be viewed in Output 1. Note that basically the output has two variables (the code and the neighbors of each code - named myid). Comparing Figure 3 and Output 1, we can see that region 1 shares borders with regions 2, 5 and 6, and that region 2 shares borders with regions 1, 3, 6 and 7 (these results are the same of those presented in Anselin (1988), page 190, except for regions 18, 20, 32, 33, 37, 42, 45 and 47).

	code	myid
1	1	2
2	1	5
3	1	6
4	2	1
5	2	3
6	2	6
7	2	7
8	3	2
9	3	4

Output 1. Output from %WMATRIX Macro for the Contiguity Case.

Output 2 shows a piece of row-standardized **W** matrix, and we can see that the matrix is the same presented in Example 32.1 of the SPATIALREG Procedure. A slighted difference is in line 3, where in Example 32.1 is shown 0.1667 instead of the full number 0.1666666667.

	COL1	COL2	COL3	COL4	COL5	COL6
1	0	0.3333333333	0	0	0.3333333333	0.3333333333
2	0.25	0	0.25	0	0	0.25
3	0	0.1666666667	0	0.1666666667	0	0
4	0	0	0.25	0	0	0
5	0.5	0	0	0	0	0.5
6	0.25	0.25	0	0	0.25	0
7	0	0.125	0.125	0	0	0.125
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0

Output 2. A Piece of Row-Standardized W Matrix Generated from %WMATRIX Macro for the Contiguity Case.

The following statements were used to fit a SAR model to the Columbus data using the contiguity matrix:

```
proc spatialreg data=crimeoh Wmat=Wmatrix;
   model crime=income hvalue / type=SAR;
run;

proc spatialreg data=crimeoh Wmat=Wmatrix_sdr NONORMALIZE;
   model crime=income hvalue / type=SAR;
run;

proc spatialreg data=crimeoh Wmat=Wmatrix_compact;
   model crime=income hvalue / type=SAR;
   spatialid code;
run;
```

And Figure 4 shows parameter estimation results from the model, considering the 3 spatial weights matrices generated by **%WMATRIX** macro (the results were the same by using the 3 spatial weights matrices, as expected). Note that results are slightly different from Example 32.1 because in Example 32.1, some regions were not included as a neighbor once they share only one coordinate.

The SPATIALREG Procedure

Model: MODEL 1
Dependent Variable: crime

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	45.603434	7.951848	5.73	<.0001
income	1	-1.048728	0.330430	-3.17	0.0015
hvalue	1	-0.266336	0.088847	-3.00	0.0027
_rho	1	0.423321	0.125586	3.37	0.0007
_sigma2	1	96.857182	19.775801	4.90	<.0001

Figure 4. SAR Model Considering the Contiguity Matrix.

Figure 5 (left) shows the locations where there are differences between **%WMATRIX** macro and Anselin (1988). Note in point 1 that regions 2 and 4 visually share borders (only 1 coordinate in common), but with zoom in (Figure 5 (right)), we can see that they are not connected, as predicted by **%WMATRIX** macro and Anselin (1988). However, in Figure 6 we can see that points 2, 3, 4 and 5 share 1 coordinate, and because of that they are connected (using bishop or queen neighbors), as predicted by **%WMATRIX** macro. The difference can be explained or because Anselin (1998) assigned neighborhood by eye or he was using a shape with a different system of coordinates of it is used in this paper.

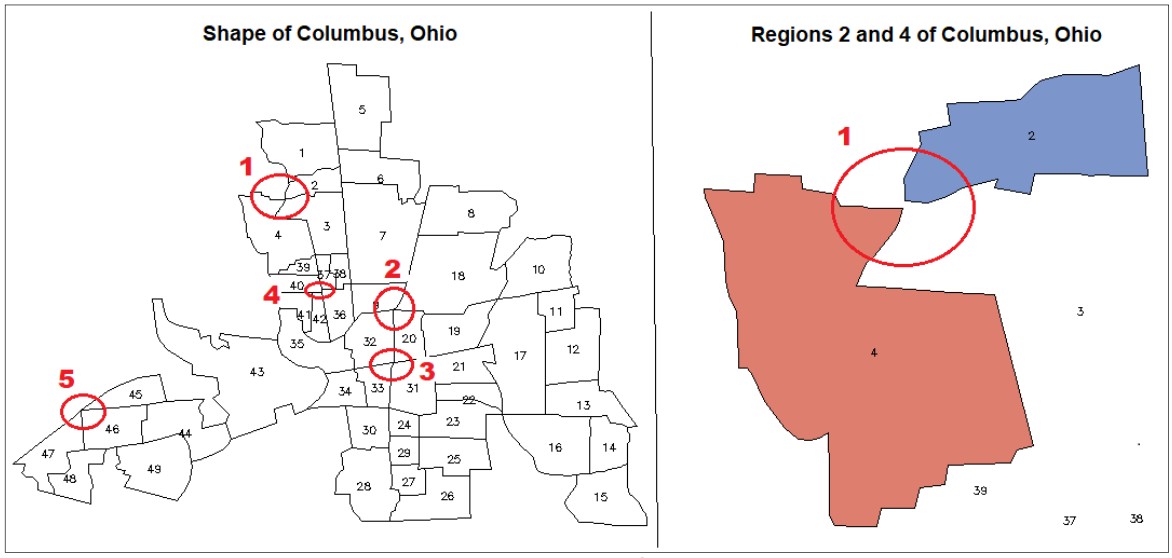


Figure 5. Regions that Share Borders.

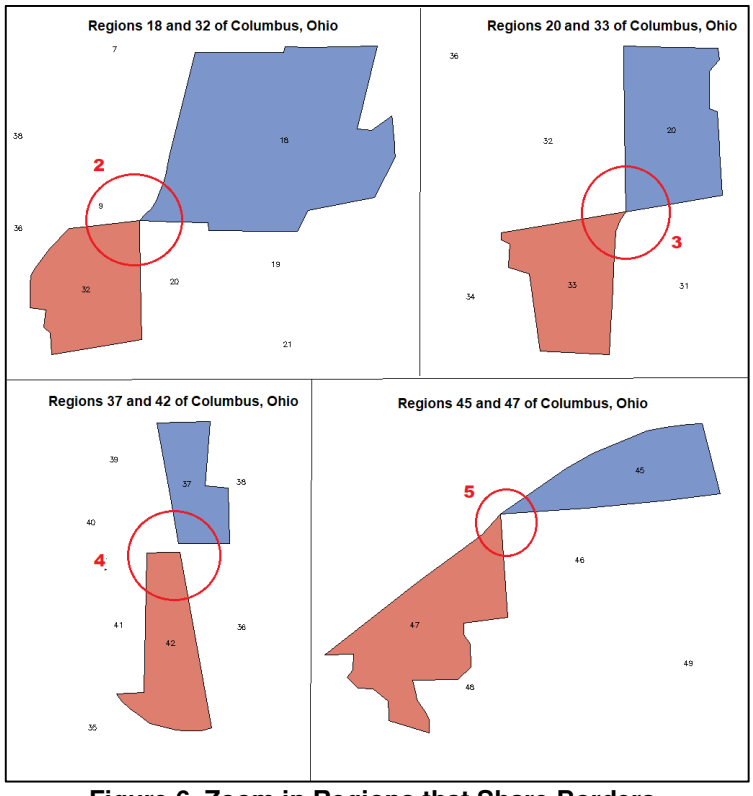


Figure 6. Zoom in Regions that Share Borders.

Just for comparison, using the following statements, we get the same results of Example 32.1, as shown in Figure 7. The difference in 5th decimal place is because the rounding from 0.1666666667 to 0.1667 discussed before.

```
data Wmatrix_compact2;set Wmatrix_compact;
  if code=18 and ccode=32 then delete;
  if code=32 and ccode=18 then delete;
  if code=20 and ccode=33 then delete;
  if code=33 and ccode=20 then delete;
  if code=45 and ccode=47 then delete;
  if code=47 and ccode=45 then delete;
run;
proc sql;
 insert into Wmatrix_compact2
  values (37,42,1) values (42,37,1);
quit;

proc spatialreg data=crimeoh Wmat= Wmatrix_compact2;
   model crime=income hvalue / type=SAR;
   spatialid code;
run;
```

The SPATIALREG Procedure

Model: MODEL 1
Dependent Variable: crime

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	45.079467	7.871241	5.73	<.0001
income	1	-1.031616	0.328416	-3.14	0.0017
hvalue	1	-0.265927	0.088222	-3.01	0.0026
_rho	1	0.431018	0.123619	3.49	0.0005
_sigma2	1	95.494608	19.507896	4.90	<.0001

Figure 7. SAR Model Considering the Adjusted Contiguity Matrix.

The macro call for generating a distance **W** matrix is:

```
%WMATRIX(map=columbus, id=code, lat=y, long=x, type=distance,
out=neighbor_columbus);
```

In this case, a piece of **W** matrix considering the Euclidian distance between the regions is given in Output 3 and a piece of of row-standardized **W** matrix is in Output 4. Note that now we have all distances between *i*th and *j*th regions, for $i, j = 1, \ldots, n$.

	idi	idj	d
1	1	2	0.3314341976
2	1	3	0.5884988411
3	1	4	0.8459335234
4	1	5	0.6075082642
5	1	6	0.7108644483
6	1	7	1.0264098389
7	1	8	1.5449828583
8	1	9	1.4411494383
9	1	10	2.2574718176
10	1	11	2.6139783905
11	1	12	2.9334658733

Output 3. Output from %WMATRIX macro for the Distance Case.

	COL1	COL2	COL3	COL4	COL5	COL6
1	0	0.0998944193	0.0562591196	0.0391383315	0.0544987264	0.0465748804
2	0.0809269614	0	0.0973520822	0.0439900659	0.036892634	0.0462355779
3	0.0404220196	0.0863412464	0	0.0622249918	0.024283861	0.0329933381
4	0.027874709	0.0386732472	0.0616804887	0	0.0176411383	0.0213877647
5	0.0630741573	0.0527051142	0.0391162997	0.0286670942	0	0.0772304481
6	0.0434439524	0.0532355644	0.0428330265	0.0280114083	0.062244507	0
7	0.0240081371	0.0338169159	0.0387357855	0.0268900315	0.0234010905	0.0433848325
8	0.0206937799	0.0239374195	0.0238932942	0.0193385442	0.0254369684	0.0377729674
9	0.0139486939	0.0180910508	0.0223861423	0.020689604	0.0126667236	0.0180584351
10	0.0142101875	0.0159227648	0.0163756709	0.0145152127	0.0160865258	0.0203513755
11	0.0117313109	0.0130572465	0.0135812612	0.0124410964	0.0128028194	0.0156723323

Output 4. A Piece of Row-Standardized W Matrix Generated from %WMATRIX Macro for the Distance Case.

The following statements were used to fit a SAR model to the Columbus data using the distance matrix:

```
proc spatialreg data=crimeoh Wmat=Wmatrix;
   model crime=income hvalue / type=SAR;

proc spatialreg data=crimeoh Wmat=Wmatrix_sdr NONORMALIZE;
   model crime=income hvalue / type=SAR;

proc spatialreg data=crimeoh Wmat=Wmatrix_compact;
   model crime=income hvalue / type=SAR;
   spatialid code;
run;
```

And Figure 8 shows parameter estimation results from the model, considering the 3 spatial weights matrices generated by **%WMATRIX** macro (the results were the same by using the 3 spatial weights matrices, as expected).

The SPATIALREG Procedure

Model: MODEL 1
Dependent Variable: crime

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	34.317683	8.905110	3.85	0.0001
income	1	-1.289958	0.303518	-4.25	<.0001
hvalue	1	-0.281368	0.091201	-3.09	0.0020
_rho	1	0.802891	0.184000	4.36	<.0001
_sigma2	1	102.085711	20.929371	4.88	<.0001

Figure 8. SAR Model Considering the Distance Matrix.

The macro call for generating a **W** matrix for the *N*th (*N*=4) nearest neighbor, following choice 2) of second section is:

```
%WMATRIX(map=columbus, id=code, lat=y, long=x, type=contiguity,
neighbors=4, out=neighbor_columbus);
```

Note that in this case, Output 5 differs from Output 1 by adding two more columns (the ordering (seq) of the neighbors and the distance (d) from the region *j* to the region *i*). We can see that all regions have exactly 4 neighbors.

	seq	idi	idj	d
1	1	1	2	0.3314341976
2	2	1	3	0.5884988411
3	3	1	5	0.6075082642
4	4	1	6	0.7108644483
5	1	2	3	0.2755150368
6	2	2	1	0.3314341976
7	3	2	6	0.5801152209
8	4	2	4	0.609727718
9	1	3	2	0.2755150368
10	2	3	4	0.3822951358
11	3	3	39	0.4412382277
12	4	3	37	0.4883577954

Output 5. Output from %WMATRIX Macro for the Contiguity Case, Considering the *4*th Nearest Neighbor.

Output 6 shows a piece of row-standardized **W** matrix for the data of Output 5.

	COL1	COL2	COL3	COL4	COL5	COL6
1	0	0.25	0.25	0	0.25	0.25
2	0.25	0	0.25	0.25	0	0.25
3	0	0.25	0	0.25	0	0
4	0	0	0.25	0	0	0
5	0.25	0.25	0.25	0	0	0.25
6	0.25	0.25	0	0	0.25	0
7	0	0	0	0	0	0.25
8	0	0	0	0	0	0.25
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0

Output 6. A Piece of Row-Standardized W Matrix Generated from %WMATRIX Macro for the *4*th Nearest Neighbor.

The following statements were used to fit a SAR model to the Columbus data using the 4th nearest neighbor matrix:

```
proc spatialreg data=crimeoh Wmat=Wmatrix;
   model crime=income hvalue / type=SAR;
run;

proc spatialreg data=crimeoh Wmat=Wmatrix_sdr NONORMALIZE;
   model crime=income hvalue / type=SAR;
run;

proc spatialreg data=crimeoh Wmat=Wmatrix_compact;
   model crime=income hvalue / type=SAR;
   spatialid code;
run;
```

And Figure 9 shows parameter estimation results from the model, considering the 3 spatial weights matrices generated by **%WMATRIX** macro (the results were the same by using the 3 spatial weights matrices, as expected).

The SPATIALREG Procedure

Model: MODEL 1
Dependent Variable: crime

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	43.454511	6.891382	6.31	<.0001
income	1	-1.046073	0.298473	-3.50	0.0005
hvalue	1	-0.252344	0.083803	-3.01	0.0026
_rho	1	0.450553	0.102469	4.40	<.0001
_sigma2	1	85.931177	17.485915	4.91	<.0001

Figure 9. SAR Model Considering the 4th Nearest Neighbor Matrix.

Finally, the macro call for generating a **W** matrix considering a distance *d=1* to be a neighbor, for example, following choice 3) of second section is:

```
%WMATRIX(map=columbus, id=code, lat=y, long=x, type=distance,
distance=1, out=neighbor_columbus);
```

Output 7 shows a piece of row-standardized **W** matrix for the data considering a distance less than 1 to be a neighbor:

	COL1	COL2	COL3	COL4	COL5	COL6
1	0	0.2	0.2	0.2	0.2	0.2
2	0.1	0	0.1	0.1	0.1	0.1
3	0.0714285714	0.0714285714	0	0.0714285714	0.0714285714	0.0714285714
4	0.0769230769	0.0769230769	0.0769230769	0	0	0
5	0.25	0.25	0.25	0	0	0.25
6	0.1428571429	0.1428571429	0.1428571429	0	0.1428571429	0
7	0	0.0625	0.0625	0.0625	0	0.0625
8	0	0	0	0	0	0.25
9	0	0	0.0526315789	0.0526315789	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0

Output 7. A Piece of Row-Standardized W Matrix Generated from %WMATRIX Macro for a Distance Less than 1 to be a Neighbor.

The following statements were used to fit a SAR model to the Columbus data using a distance less than 1 to be a neighbor matrix:

```
proc spatialreg data=crimeoh Wmat=Wmatrix;
   model crime=income hvalue / type=SAR;
run;

proc spatialreg data=crimeoh Wmat=Wmatrix_sdr NONORMALIZE;
   model crime=income hvalue / type=SAR;
run;

proc spatialreg data=crimeoh Wmat=Wmatrix_compact;
   model crime=income hvalue / type=SAR;
   spatialid code;
run;
```

And Figure 10 shows parameter estimation results from the model, considering the 3 spatial weights matrices generated by **%WMATRIX** macro (the results were the same by using the 3 spatial weights matrices, as expected). Note that changing the spatial weight matrix, the results also change based on the new space configuration represented by the spatial weights matrix.

The SPATIALREG Procedure

Model: MODEL 1
Dependent Variable: crime

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	45.813343	9.424640	4.86	<.0001
income	1	-1.133655	0.345212	-3.28	0.0010
hvalue	1	-0.257386	0.092819	-2.77	0.0056
_rho	1	0.424624	0.156626	2.71	0.0067
_sigma2	1	105.319777	21.376814	4.93	<.0001

Figure 10. SAR Model Considering a Distance Less than 1 to be a Neighbor Matrix.

CONCLUSIONS

The new SPATIALREG Procedure estimates spatial regression models by including in its structure the effect of spatial proximity, named **W** matrix. However, this **W** matrix cannot be created directly in the procedure. This paper showed how to create this matrix from a shape file (or a dataset with the coordinates of the region), considering the contiguity case and the distance case. Now it is easy to use the new SPATIALREG Procedure considering different ways to measure spatial proximity. A suggestion is also make available other `WMAT` statement to be used for SAC models, which require 2 spatial weights matrices.

REFERENCES

Anselin, L. 1988. *Spatial Econometrics: Methods and Models.* Kluwer Academic Publishers, USA.

Babcock, G. 2010. "Aggregation without aggravation: determining spatial contiguity and joining geographic areas using hashing". *SAS Global Forum 2010, Seattle, Washington.*

SAS Institute, Inc. 2017, *SAS/STAT 9.4 User's Guide, Cary, NC: SAS Institute, Inc.*

Wu, G. and Chvosta, J. 2016. "How Do My Neighbors Affect Me? SAS/ETS® Methods for Spatial Econometric Modeling". *SAS Global Forum 2016, Las Vegas, Nevada.*

Wu, G. and Chvosta, J. 2017. "Big Value from Big Data: SAS/ETS® Methods for Spatial Econometric Modeling in the Era of Big Data". *SAS Global Forum 2017, Orlando, Florida.*

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: +55 61 3107-3672
E-mail: alansilva@unb.br
Web: www.est.unb.br

APPENDIX I – WMATRIX MACRO

```
%macro Wmatrix(map=,id=,lat=,long=,type=contiguity,centroid_lat=,
centroid_long=, distance=,neighbors=,out=neighbor);

%if %upcase(&type)=CONTIGUITY %then %do;
%if &neighbors = %then %do;
/********Defining the Neighborhood for contiguity ***********/
proc sort data=&map out=&map.2 nodupkey; by &long &lat &id;run;
data _hashmap_ (keep=&long &lat z &id);
retain z;
set &map.2(where=(&long NE . and &lat NE .));
by &long &lat;
if first.&long or first.&lat then z=1;
z=z+1;
run;
proc means data=_hashmap_ noprint;
output out=maxz max(z)=mz; run;
/*put the maximum value of z into macro variable maxnb*/
data _null_;
set maxz;
call symputx("maxnb",mz);
run;
%put &maxnb;
proc sort data=&map.2; by &id;run;
data nonb (keep=myid);
/*length &long &lat z 8; format &id 16.;*/
if _n_=1 then do;
/*this hash object holds a second copy of the entire map for comparison*/
declare hash fc(dataset: "_hashmap_");
fc.definekey("&long","&lat","z");
fc.definedata("&long","&lat","z","&id");
fc.definedone();
call missing (&long,&lat,z,&id);
/*this hash object will hold the rook neighbors for each area: they have two
points in common*/
declare hash rook();
rook.definekey("&id","myid");
rook.definedata("&id","myid");
rook.definedone();
end;
/*this hash object holds the bishop neighbors for each area: they have a
point
in common*/
declare hash bishop();
bishop.definekey("&id","myid");
bishop.definedata("&id","myid");
bishop.definedone();
foundnb="N";
do until (last.myid);
set &map.2 (keep=&id &long &lat rename=(&id=myid &long=myx &lat=myy)
where=(myx NE .
and myy NE .)) end=eof;
by myid;
do n=1 to &maxnb.; /*this is max number of points in common =max z*/
rc=fc.find(key:myx, key:myy, key:n);
if rc=0 and myid NE &id then do;
```

```
nbrc=rook.check(key:&id, key:myid);
if nbrc=0 then do;
rc2=rook.add(key:&id, key:myid, data:&id, data:myid);
foundnb="Y";
end;
else rc1=rook.add(key:&id, key:myid, data:&id, data:myid);
end;
end;*do &maxnb.;
end;*end DOW loop;
if foundnb="N" then output nonb;
if eof then rook.output(dataset:"&out");
run;
proc sort data=&out;by &id myid;run;

proc sort data=&out(keep=&id) nodupkey out=_tab_;by &id;run;
data _tab_;set _tab_;idi=_n_;run;
proc sort data=&out;by &id;run;
data _tab2_;merge &out _tab_;by &id;run;
proc sort data=_tab2_;by myid;run;
data _tab2_;merge _tab2_ _tab_(rename=(idi=idj &id=myid));by myid;run;
proc sort data=_tab2_;by &id;run;
proc sql;drop table _tab_,_hashmap_,Nonb,Maxz,&map.2;quit;

data Wmatrix_compact;set &out(rename=myid=c&id);
value=1;
run;
%end;
%else %do;
%if &centroid_lat= or &centroid_long= %then %do;
%annomac;
proc sort data=&map(keep=&id &lat &long rename=(&long=y &lat=x)) out=_&map;by
&id;run;
%centroid(_&map,&out,&id);
proc sql;drop table _&map;quit;
%end;
%else %do;
data &out;set &map(keep=&id &centroid_lat &centroid_long
rename=(&centroid_long=y &centroid_lat=x));run;
%end;

proc iml;
use &out;
read all var{x y} into COORD;
n=nrow(coord[,1]);
*print n;
d=j(1,3,0);
nome={"idi" "idj" "d"};
create _dist_ from d[colname=nome];
do i=1 to n;
      do j=1 to n;
            d[1]=i;
            d[2]=j;
            d[3]=sqrt((COORD[i,1]-COORD[j,1])**2+(COORD[i,2]-COORD[j,2])**2);
            append from d;
      end;
end;
close _dist_;
```

```
quit;
data _dist_;set _dist_;if d=0 then delete;run;
proc sort data=_dist_;by idi d;run;
data _dist_;retain seq 0;set _dist_;by idi;
if first.idi then seq=1;
else seq+1;
run;
data _tab2_;set _dist_(where=(seq<=&neighbors));run;
data &out;set _tab2_;run;
data Wmatrix_compact;set &out;
rename idi=&id idj=c&id;value=1;
drop seq d;
run;
proc sql;drop table _dist_;quit;
%end;

/********creating W Matrix ***********/

proc iml;
use _tab2_ var{idi idj};
read all;
n=max(idj);
w=j(n,n,0);
do h=1 to nrow(idi);
w[idi[h],idj[h]]=1;
end;
wpdr=j(n,n,0);
soma=j(n,1,0);
do i=1 to n;
do j=1 to n;
soma[i]=soma[i]+w[i,j];
end;
end;
do i=1 to n;
do j=1 to n;
if soma[i]=0 then wpdr[i,j]=0;
else wpdr[i,j]=w[i,j]/soma[i];
end;
end;
create Wmatrix from w;
append from w;
create Wmatrix_sdr from wpdr;
append from wpdr;
quit;
proc sql;drop table _tab2_;quit;
%end;
%else %if %upcase(&type)=DISTANCE %then %do;

%if &centroid_lat= or &centroid_long= %then %do;
%annomac;
proc sort data=&map(keep=&id &lat &long rename=(&long=y &lat=x)) out=_&map;by
&id;run;
%centroid(_&map,&out,&id);
proc sql;drop table _&map;quit;
%end;
%else %do;
```

```
data &out;set &map(keep=&id &centroid_lat &centroid_long
rename=(&centroid_long=y &centroid_lat=x));run;
%end;

proc iml;
use &out;
read all var{x y} into COORD;
n=nrow(coord[,1]);
*print n;
d=j(1,3,0);
nome={"idi" "idj" "d"};
create _dist_ from d[colname=nome];
do i=1 to n;
      do j=i+1 to n;
            d[1]=i;
            d[2]=j;
            d[3]=sqrt((COORD[i,1]-COORD[j,1])**2+(COORD[i,2]-COORD[j,2])**2);
            append from d;
      end;
end;
close _dist_;
quit;
data &out;set _dist_;run;
proc sql;drop table _dist_;quit;

data Wmatrix_compact;
set &out(rename=(idi=&id idj=c&id)) &out(rename=(idi=c&id idj=&id));
%if &distance= %then %do;
value=1/d;
%end;
%else %do;
if d<=&distance then value=1;
if value=. then delete;
%end;
keep &id c&id value;
run;
proc sort data=Wmatrix_compact;by &id c&id;run;

/********creating W Matrix ***********/
proc iml;
use &out var{idi idj d};
read all;
n=max(idj);
w=j(n,n,0);
do h=1 to nrow(idi);
%if &distance= %then %do;
w[idi[h],idj[h]]=1/d[h];
w[idj[h],idi[h]]=1/d[h];
%end;
%else %do;
if d[h]<=&distance then do;
w[idi[h],idj[h]]=1;
w[idj[h],idi[h]]=1;
end;
%end;
end;
```

```
wpdr=j(n,n,0);
soma=j(n,1,0);
do i=1 to n;
do j=1 to n;
soma[i]=soma[i]+w[i,j];
end;
end;
do i=1 to n;
do j=1 to n;
if soma[i]=0 then wpdr[i,j]=0;
else wpdr[i,j]=w[i,j]/soma[i];
end;
end;
create Wmatrix from w;
append from w;
create Wmatrix_sdr from wpdr;
append from wpdr;
quit;
%end;

%mend Wmatrix;
```