Paper 8000-2016

A SAS® Macro for Geographically Weighted Negative Binomial Regression

Alan Ricardo da Silva, Universidade de Brasília, Dep. de Estatística, Brazil
Thais Carvalho Valadares Rodrigues, Universidade de Brasília, Dep. de Estatística, Brazil

ABSTRACT

Geographically Weighted Negative Binomial Regression (GWNBR) was developed by Silva and Rodrigues (2014) and it is a generalization of Geographically Weighted Poisson Regression (GWPR) proposed by Nakaya et al. (2005) and of Poisson and negative binomial regressions. This paper aims to show a SAS® macro to estimate GWNBR model encoded in SAS/IML and using PROC GMAP to draw the maps.

INTRODUCTION

Local spatial regression differs from global spatial regression by analyzing the relationship between variables in a specific way for each unit of study instead of combining them. In fact, regions *j* closer region *i* have greater influence in the estimates of the regression coefficients than when those regions are far apart. Having a specific adjustment for each region, the final result is a better representation of the process as a whole. The reason of to use this analysis is the violation of the assumption of stationarity demanded by global models, which allows one to attribute the same relation between variables for all units of study. The nonstationarity or spatial heterogeneity is the process where responses vary with location, area or other characteristics of the spatial units (Anselin, 1988). In this way, Geographically Weighted Regression (GWR) (see Fotheringham et al. (2002)) is used to local modeling.

However, this technique is used when the distribution of the data is Gaussian. In many applications, the dependent variable represents a count, which makes classic GWR inappropriate to model this type of data. Poisson and negative binomial are the most adequate distributions for the modeling of count data, and Geographically Weighted Poisson Regression (GWPR) was developed by Nakaya et al. (2005). Silva and Rodrigues (2014) developed GWR using negative binomial distribution, named GWNBR, and they showed that GWNBR incorporates GWPR. Thus, the main objective of this paper is to show a SAS® macro to estimate the parameters of the GWNBR model.

A BASIC OUTLINE OF GWNBR

The most used global Negative Binomial Regression (NB-2) considers a logarithm link function. Parameterizing this model in terms of μ_j/t_j, where t_j is an off set variable, μ_j is the predicted mean, α is the parameter of overdispersion, β_k is the parameter related to the explicative variable x_k, for $k = 1, \dots, K$, and y_j is the *j*-th dependent variable for $j = 1, \dots, n$ we have.

$$y_j \sim NB\left[\ t_j exp\left(\textstyle\sum_k \beta_k\, x_{jk}\ \right), \alpha\ \right] \tag{1}$$

where NB represents Negative Binomial.

GWNBR is an extension of the global or non-spatial model (1), which allows the spatial variation of the parameters β_k and α. Without limiting the functional form of this variation, GWNBR produces non-parametric surfaces of the parameter estimates. This local model is described as the following:

$$y_j \sim NB\left[t_j\ exp\left(\textstyle\sum_k \beta_k(u_j, v_j)x_{jk}\right), \alpha(u_j, v_j)\right] \tag{2}$$

where (u_j, v_j) are the locations (coordinates) of the data points *j*, for $j = 1, \dots, n$.

The parameter estimation of the global model (1) is performed interactively with the combination of the NR and IRLS methods, i. e., using $\hat{\alpha}$, which is estimated by the NR method, we estimate the vector $\boldsymbol{\beta}$ using the IRLS method. Thus, from this new $\hat{\boldsymbol{\beta}}$, we update $\hat{\alpha}$, and so forth until convergence is obtained. However, modifications in the NR and IRLS algorithms are necessary to incorporate the local variations.

As shown in Silva and Rodrigues (2014), the analytical solution for the local log-likelihood of GWNBR is given by

$$\hat{\boldsymbol{\beta}}(u_i, v_i)^{(m+1)} = \left[\boldsymbol{X}'\boldsymbol{W}(u_i, v_i)\boldsymbol{A}(u_i, v_i)^{(m)}\boldsymbol{X}\right]^{-1}\boldsymbol{X}'\boldsymbol{W}(u_i, v_i)\boldsymbol{A}(u_i, v_i)^{(m)}\boldsymbol{z}(u_i, v_i)^{(m)} \tag{3}$$

where $\boldsymbol{X}$ is an $n \times\ k$ matrix of the explicative variables

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ 1 & x_{12} & \dots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{nk} \end{pmatrix} \tag{4}$$

$\boldsymbol{W}(u_i, v_i)$ is an $n \times n$ GWR diagonal weighting matrix for point *i*

$$\boldsymbol{W}(u_i, v_i) = \begin{pmatrix} w_{i1} & 0 & \dots & 0 \\ 0 & w_{i2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_{in} \end{pmatrix} \tag{5}$$

and $\boldsymbol{A}(u_i, v_i)^{(m)}$ is an $n \times n$ GLM diagonal weighting matrix for iteration *m* and location *i*

$$\boldsymbol{A}(u_i, v_i)^{(m)} = \begin{pmatrix} a_{i1}^{(m)} & 0 & \dots & 0 \\ 0 & a_{i2}^{(m)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{in}^{(m)} \end{pmatrix} \tag{6}$$

$$\boldsymbol{z}(u_i, v_i)^{(m)} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)} + \frac{y_j - \mu_j(\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)})}{a_{ij}^{(m)}\left(1 + \alpha_i \times \mu_j(\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)})\right)} \tag{7}$$

Silva and Rodrigues (2014) used the IRLS method with the observed Fisher Information Matrix, and the elements $a_{ij}^{(m)}$ $(j = 1, \dots, n)$ of GWNBR (6) are the following:

$$a_{ij}^{(m)} = \frac{\mu_j(\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)})}{1 + \alpha_i\, \mu_j(\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)})} + \frac{[y_j - \mu_j(\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)})][\alpha_i \mu_j(\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)})]}{1 + 2\alpha_i \mu_j(\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)}) + \alpha_i^2 \mu_j^2(\widehat{\boldsymbol{\beta}}(u_i, v_i)^{(m)})} \tag{8}$$

As in GWPR (Nakaya et al., 2005), the covariance matrix of the parameter estimates can be estimated by

$$\widehat{Cov}[\widehat{\boldsymbol{\beta}}(u_i, v_i)] = \boldsymbol{C}(u_i, v_i)\boldsymbol{A}(u_i, v_i)^{-1}\boldsymbol{C}'(u_i, v_i) \tag{9}$$

where

$$\boldsymbol{C}(u_i, v_i) = [\boldsymbol{X}'\boldsymbol{W}(u_i, v_i)\boldsymbol{A}(u_i, v_i)\boldsymbol{X}]^{-1} \times \boldsymbol{X}'\boldsymbol{W}(u_i, v_i)\boldsymbol{A}(u_i, v_i) \tag{10}$$

and the elements of $\boldsymbol{W}(u_i, v_i)$ and $\boldsymbol{A}(u_i, v_i)$ are given by (5) and (6), respectively.

After an estimate of $\boldsymbol{\beta}(u_i, v_i)$ is obtained, the parameters α_i will be estimated using the NR method based on the local log-likelihood given by (Silva and Rodrigues, 2014)

$$L(r_i | y_j, \boldsymbol{\beta}(u_i, v_i)) \sum_{j=1}^{n} \{ y_j \log[\mu_j(\boldsymbol{\beta}(u_i, v_i))] - [y_j + r_i] \times log[r_i + \mu_j(\boldsymbol{\beta}(u_i, v_i))]$$
$$+ r_i log[r_i] + log\,[\Gamma(y_j + r_i)] - log\,[\Gamma(r_i)] - log[\Gamma(y_j + 1)]\}\, w(d_{ij}) \tag{11}$$

Maximizing the local log-likelihood (11) using the univariate NR method, we obtain

$$r_i^{(m+1)} = r_i^{(m)} - \left[H_i^{(m)} \right]^{-1} U_i^{(m)} \tag{12}$$

where $U_i^{(m)}$ and $H_i^{(m)}$ are the first and second derivates of the local log-likelihood with respect to $r_i^{(m)}$, i. e.,

$$U_i^m = \frac{d\,L(r_i)}{d\,r_i} = \sum_{j=1}^{n} \left\{ \psi\left[r_i^{\{(m)\}} + y_j \right] - \psi\,[\,r_i^m\,] + log[r_i^m] + 1 - \right.$$
$$\left. log\left[r_i^m + \mu_j(\boldsymbol{\beta}(u_i, v_i))\right] - \frac{r_i^m + y_j}{r_i^m + \mu_j(\boldsymbol{\beta}(u_i, v_i))} \right\} w(d_{ij}) \tag{13}$$

$$H_i^m = \frac{d^2\,L(r_i)}{d\,r_i^2} = \sum_{j=1}^{n} \left\{ \psi'[\,r_i^m + y_j\,] - \psi'[\,r_i^m\,] + \frac{1}{r_i^m} - \frac{2}{r_i^m + \mu_j(\boldsymbol{\beta}(u_i, v_i))} + \right.$$
$$\left. \frac{r_i^m + y_j}{[r_i^m + \mu_j(\boldsymbol{\beta}(u_i, v_i))]^2} \right\} w(d_{ij}) \tag{14}$$

where $\psi(.)$ and $\psi'(.)$ are the digamma and trigamma functions, respectively, which are given by

$$\psi(z) = \frac{d\log\Gamma\{(z)}{dz} \quad \text{and} \quad \psi'(z) = \frac{d\psi(z)}{dz} = \frac{d^2 log\Gamma\{(z)}{dz^2}$$

Using delta method, Silva and Rodrigues (2014) found that, for a function $g(.)$ that is differentiable in θ, if the distribution $\hat{\theta}_n \rightarrow N(\theta, \sigma_n^2)$ achieves convergence, then the distribution $g(\hat{\theta}_n) \rightarrow N(g(\theta), [g'(\theta)]^2 \times \sigma_2^2)$ also converges (Casella and Berger, 2011). If, under certain conditions, the estimators that maximize the local likelihood are asymptotically normal, unbiased and consistent (Staniswalis, 1989), then

$$\hat{\alpha}_i = \frac{1}{\hat{r}_i} \quad and \quad Var(\hat{\alpha}_i) = \frac{-1}{H_i \hat{r}_i^4} \tag{15}$$

because $Var(\hat{r}_i) = -1/H_i$, where H_i is given in (14), and $[g'(r_i)]^2 = 1/r_i^4$.

Thus, for each regression point *i*, the NR and IRLS algorithms are used alternately until the parameter estimates achieve convergence.

To complete the fitting of the model, it is necessary to estimate the bandwidth of the chosen kernel function, which helps determine the weights $w(d_{ij})$. One possibility could be to estimate it such that it minimizes the corrected AIC criterion (AICc):

$$AIC_c = -2L(\boldsymbol{\beta}, \boldsymbol{\alpha}) + 2k + \frac{2k(k+1)}{n-k-1} \tag{16}$$

where k is the effective number of parameters and $L(\boldsymbol{\beta}, \boldsymbol{\alpha})$ is the log-likelihood of GWNBR shown in Equation (11).

The effective number of parameters of GWNBR can be written as $k = k_1 + k_2$, where k_1 and k_2 are the effective number of parameters due to $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$, respectively. Following the method developed by Nakaya et al. (2005), k_1 is given by the trace of the matrix $\boldsymbol{R}$, which is given by elements

$$\boldsymbol{r}_j = \boldsymbol{X}_j[\boldsymbol{X}'\boldsymbol{W}(u_j, v_j)\boldsymbol{A}(u_j, v_j)\boldsymbol{X}]^{-1}\boldsymbol{X}'\boldsymbol{W}(u_j, v_j)\boldsymbol{A}(u_j, v_j) \tag{17}$$

where $\boldsymbol{X}_j$ is the *j*-th row of $\boldsymbol{X}$.

However, to date, it has not been possible to estimate k_2, i.e., the contribution of the surface of $\boldsymbol{\alpha}$ on the effective number of parameters of the model. Consequently, Silva and Rodrigues (2014) opted to estimate the bandwidth using the cross-validation criterion given by (Fotheringham et al., 2002):

$$CV = \sum_{j=1}^{n}[y_j - \hat{y}_{\neq j}(b)]^2 \tag{18}$$

where $\hat{y}_{\neq j}(b)$ is the estimated value for point *j*, omitting the observation *j* and *b* is the bandwidth.

Note that the indetermination of k_2 does not prevent the adjustment of GWNBR. However, this indetermination makes it difficult to compare models because the complexity of the GWNBR, which is given by the effective number of parameters, is unknown.

GWNBRG MODEL

To avoid the difficult associate with the estimation of k_2, Silva and Rodrigues (2014) proposed the Geographically Weighted Negative Binomial Regression with α global methodology, which is namely GWNBRg. In this model, the spatial variation is allowed only for $\boldsymbol{\beta}(u_i, v_i)$, i. e.,

$$y_j \sim NB\left[\, t_j exp\left(\sum_k \beta_k(u_j, v_j)\, x_{jk}\right), \alpha \,\right] \tag{19}$$

where the parameters are the same as in Equation (2).

In the GWNBRg model, the estimation of the parameter α is made globally, i.e., Silva and Rodrigues (2014) assumed that all of the parameters in the model are stationary, and we estimate a global overdispersion $\hat{\alpha}$ to be used in the local estimates $\boldsymbol{\beta}(u_i, v_i)$. Consequently, they proposed that the estimate of α in the GWNBRg model will be the same as that obtained through non-spatial (or global) negative binomial regression.

The parameters $\boldsymbol{\beta}(u_i, v_i)$ are estimated using the IRLS method, as in the GWNBR model described earlier, assuming that $\hat{\alpha}_i = \hat{\alpha}$ for all *i*. Note that it is not necessary to alternate the NR and IRLS methods, because once α is estimated globally, they estimate $\boldsymbol{\beta}(u_i, v_i)$ for each regression point *i* using only the IRLS method.

Because there is no spatial variation for α, its contribution to the effective number of the parameters in the model is the unity, i.e., $k_2 = 1$. Consequently, the bandwidth can be found using the AIC criterion (16), where

$$L(\boldsymbol{\beta}(u_i, v_i), \alpha | x_{jk}, y_j) = \sum_{j=1}^{n} \Big\{ y_j \, log(\alpha \, \mu_j) - \left(y_j + 1/\alpha \right) log\left(1 + \alpha \, \mu_j\right) + log\left[\Gamma\left(y_j + 1/\alpha\right)\right] - log\left[\Gamma\left(\frac{1}{\alpha}\right)\right] - log\left[\Gamma\left(y_j + 1\right)\right] \Big\} \quad (20)$$

and $k = tr(\boldsymbol{R}) + 1$.

More details about GWR, bandwidth and other topics can be found in Fotheringham et al. (2202), Paez et al. (2011); Leung et al. (2000); Wheeler and Tiefelsdorf (2005); Farber and Paez (2007); McMillen (2010); Cleveland and Devlin (1988).

SAS® MACRO

The SAS® macros use the **IML** (**I**nteractive **M**atrix **L**anguage) Procedure and the parameters are described as follows. SAS® `%golden` macro is used to the Golden Section Search (find the optimal bandwidth) and `%gwnbr` macro is used to estimated GWNBR model.

```
%golden(data=,y=,x=,lat=,long=,method=,type=,gwr=,offset=,out=);
```

```
%gwnbr(data=,y=,x=,lat=,long=,h=,grid=,latg=,longg=,gwr=,method=,alphag=,
offset=,geocod=,out=);
```

Input arguments of `%golden` and `%gwnbr` macros are defined as follows:

- `data` = specifies the data set to be analyzed.
- y = specifies the response or dependent variable.
- x = specifies the independent or explicative variables.
- `lat` = specifies the latitude or *y* axis variable.
- `long` = specifies the longitude or *x* axis variable.
- `method` = specifies the method to be used to find the bandwidth: FIXED, ADAPTIVE1 (1 bandwidth related to the number of neighbors) or ADAPTIVEN (*n* bandwidths).
- `type` = specifies the statistic to be used to find the bandwidth: AIC, CV (Cross Validation) or DEV (Deviance).
- `gwr` = specifies the model: GLOBAL (GWNBRg), LOCAL (GWNBR), POISSON.
- `offset` = specifies the offset variable.
- `out` = specifies output data set.
- h = specifies the bandwidth.
- `grid` = specifies the data set with coordinates to be estimate the parameters.
- `latg` = specifies the latitude or *y* axis variable of the grid data set.
- `longg` = specifies the longitude or *x* axis variable of the grid data set.
- `alphag`= specifies the global alpha value to be used in GWNBRg.
- `geocod` = specifies the geocoding variable.

ILLUSTRATION

To illustrate how GWNBR works, we will use the data set presented by Silva and Rodrigues (2014) about the vehicles used for road freight transportation in Espirito Santo, Brazil. Also we use the Poisson and negative binomial models, in their global and spatial forms. The results of golden section search and of the fit are in Table 1.

Model	Method	Type	Golden	*b*	$L(\beta,\alpha)$	AICc
GWNBR	Fixed	AIC	892.84	29.4 km	-419.74	892.84
GWNBR	Fixed	CV	67550836	53.2 km	-440.09	899.93
GWNBR	Adaptive	AIC	886.97	18	-416.49	886.97
GWNBR	Adaptive	CV	64188661	49	-446.04	907.01
GWNBRg	Fixed	AIC	908.66	53.1 km	-444.41	908.66
GWNBRg	Fixed	CV	67668447	53.2 km	-444.45	908.66
GWNBRg	Adaptive	AIC	910.08	34	-443.06	910.08
GWNBRg	Adaptive	CV	64303765	49	-447.99	910.91
GWPR	Fixed	AIC	1705.10	9.4 km	-569.77	1705.10
GWPR	Fixed	CV	7309793	28.5 km	-2055.79	4152.17
GWPR	Adaptive	AIC	1549.66	5	-467.89	1549.67
GWPR	Adaptive	CV	14127354	41	-3106.87	6228.27
NBR	Fixed	-	-	-	-458.48	923.29
PR	Fixed	-	-	-	-5001.36	10006.88

Table 1. Golden Section Search and GWNBR Results for Espirito Santo Data, Brazil

The macros calls used for the best models are as bellow:

```
/****** GWR=GWNBR and METHOD=FIXED AND TYPE=CV *********/
%golden(data=data_gwnbr,y=fleet,x=industry,lat=y,long=x,method=fixed,
type=cv,gwr=local,out=band);
%gwnbr(data=data_gwnbr,y=fleet,x=industry,lat=y,long=x,h=53.2,gwr=local,
method=fixed,geocod=geocod,out=gwnbr);

/****** GWR=GWNBRg and METHOD=FIXED AND TYPE=AIC *********/
%golden(data=data_gwnbr,y=fleet,x=industry,lat=y,long=x,method=fixed,
type=aic,gwr=global,out=band);
%gwnbr(data=data_gwnbr,y=fleet,x=industry,lat=y,long=x,h=53.068412,
gwr=global, method=fixed,geocod=geocod,out=gwr);

/****** GWR=GWPR and METHOD=FIXED AND TYPE=AIC *********/
%golden(data=data_gwnbr,y=fleet,x=industry,lat=y,long=x,method=fixed,
type=aic,gwr=poisson,out=band);
%gwnbr(data=data_gwnbr,y=fleet,x=industry,lat=y,long=x,h=9.3796134,
gwr=poisson,method=fixed,geocod=geocod,out=gwr);
```

As shown in Table 1, the PR and GWPR models were the worst models. The bandwidth found for GWPR with the best $L(\boldsymbol{\beta}, \boldsymbol{\alpha})$ is inappropriate because there are only between 1 and 9 points for each regression. The problem detected in the estimation of the bandwidth was a clue of the lack of fit obtained with the Poisson distribution. Thus, we can conclude that the fleet of vehicles used in road freight transportation probably exhibits overdispersion. Therefore, the negative binomial distribution is the best approach for the modeling of these variables.

As an example, the surface of the parameters for GWNBR with bandwidth equal to 53.2 km and their standard errors are shown in Figure 1. The greater values for the intercept (b_0) are concentrated in the Vitoria metropolitan region (southeast side), which is the capital of the state. This finding reflects the greater amount of vehicles located in that place. In contrast, the parameter estimates for b_1 are smaller in that area because there are many industries and the link function is exponential.

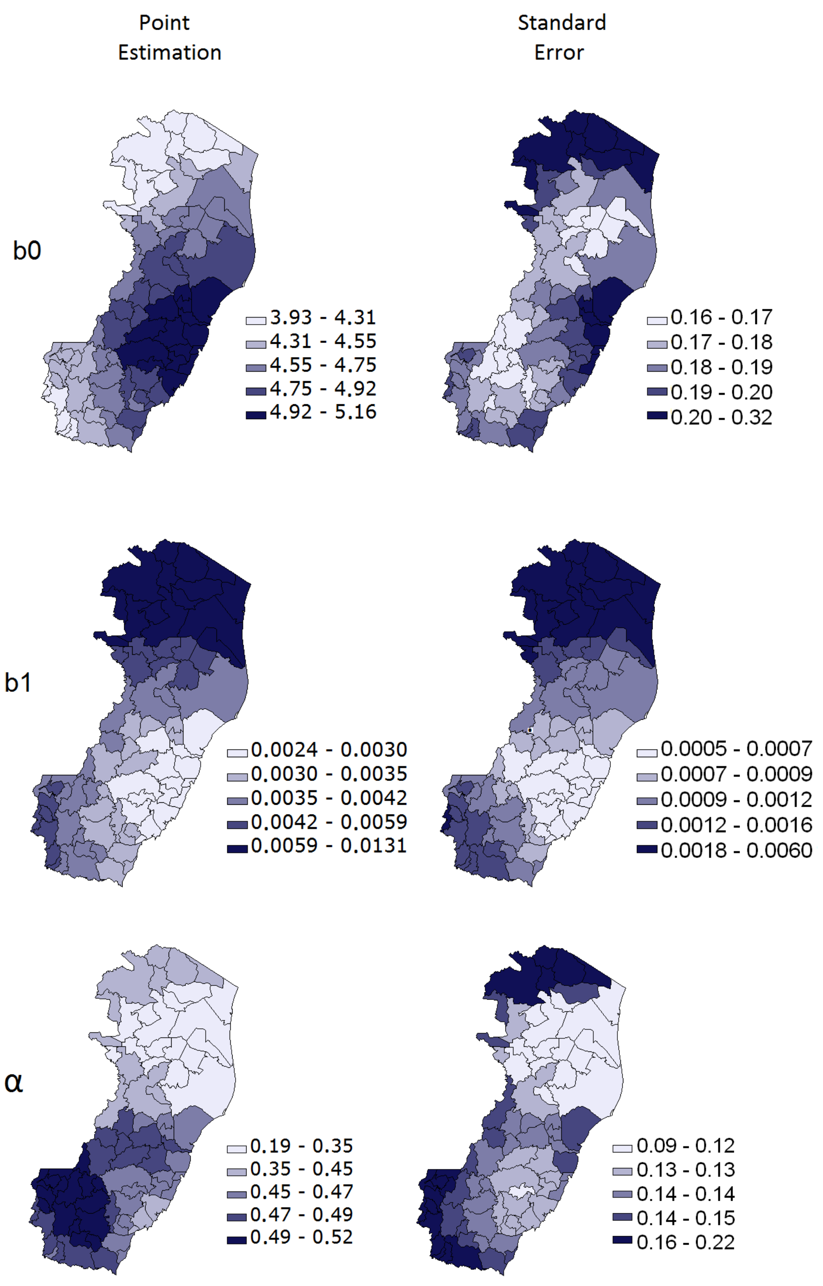
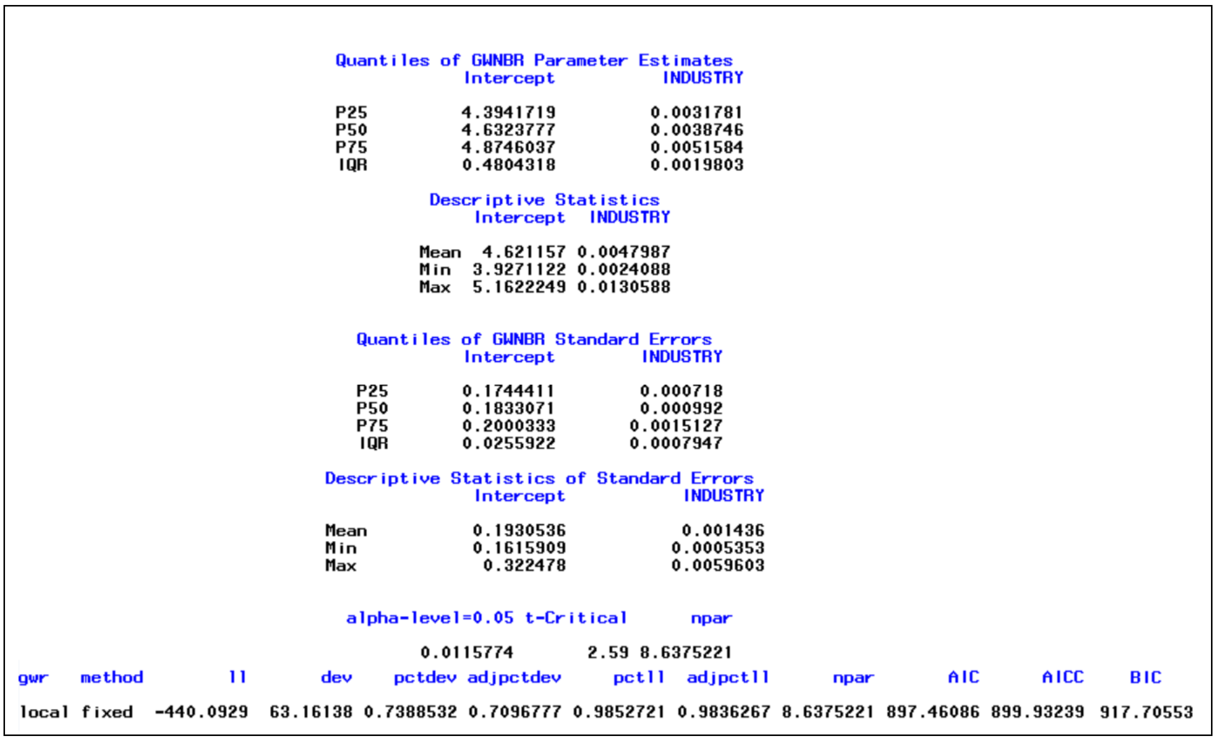


Figure 1. Surface of the parameter estimates and standard errors obtained with the GWNBR model

Output 1 shows the output of `%gwnbr` macro, where we can see, besides descriptive statistics, the actual alpha-level, *t*-critical and the number of parameter estimated by GWNBR model, following Silva and Fotheringham (2015), and pseudo R^2 and adjusted R^2 measures, considering Deviance (`pctdev` and `adjpctdev`, respectively) and considering likelihood (`pctll` and `adjpctll`, respectively), following Cameron and Windmeijer (1996).



```
                          Quantiles of GWNBR Parameter Estimates
                                      Intercept              INDUSTRY

                          P25         4.3941719             0.0031781
                          P50         4.6323777             0.0038746
                          P75         4.8746037             0.0051584
                          IQR         0.4804318             0.0019803

                                   Descriptive Statistics
                                       Intercept  INDUSTRY

                                 Mean   4.621157 0.0047987
                                 Min   3.9271122 0.0024088
                                 Max   5.1622249 0.0130588


                            Quantiles of GWNBR Standard Errors
                                      Intercept            INDUSTRY

                            P25       0.1744411           0.000718
                            P50       0.1833071           0.000992
                            P75       0.2000333          0.0015127
                            IQR       0.0255922          0.0007947

                         Descriptive Statistics of Standard Errors
                                       Intercept               INDUSTRY

                         Mean          0.1930536               0.001436
                         Min           0.1615909              0.0005353
                         Max            0.322478              0.0059603


                             alpha-level=0.05 t-Critical       npar

                                    0.0115774        2.59 8.6375221
gwr    method        ll        dev     pctdev adjpctdev      pctll  adjpctll       npar        AIC       AICC        BIC

local fixed   -440.0929  63.16138 0.7388532 0.7096777 0.9852721 0.9836267 8.6375221 897.46086 899.93239 917.70553
```

Output 1. Output from `%gwnbr` **macro**

CONCLUSION

Geographically Weighted Negative Binomial Regression (GWNBR) is an important tool to incorporate overdispersion to the local model. The equations proposed by Silva and Rodrigues (2014) were encoded in SAS/IML language and now this model can be estimated, as well as, it is possible to estimate GWPR proposed by Nakaya et al. (2005) from GWNBR. The illustration showed that when the overdispersion is present the Poisson distribution is not adequate to model the data, and the quality of fit of the negative binomial distribution is superior, mainly GWNBR.

REFERENCES

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

Cameron, A. C. and Windmeijer, F. A. G. (1996). "*R*-Squared Measures for Count Data Regression Models with Applications to Health-Care Utilization". *Journal of Business and Economic Statistics*. 14(2): 209-220.

Casella, G. and Berger, R. L. 2001. *Statistical Inference*. 2nd ed. Duxbury.

Cleveland, W. S. and Devlin, S. J. 1988. "Locally-weighted Regression: An Approach to Regression Analysis by Local Fitting". *Journal of the American Statistical Association.* 83: 596-610.

Farber, S. and Paez, A. 2007. “A Systematic Investigation of Cross-Validation in GWR Model Estimation: Empirical Analysis and Monte Carlo Simulations”. *Journal of Geographical Systems* 9(4): 371-396.

Fotheringham, A. S., Brunsdon, C. and Charlton, M. 2002. *Geographically Weighted Regression*. Wiley.

Leung, Y., Mei, C.-L. and Zhang, W.-X. 2000. “Statistical Tests for Spacial Nonstationarity Based on the Geographically Weighted Regression Model”. *Environment and Planning A* 32: 9-32.

McMillen, D. P. 2010. “Issues in Spatial Data Analysis”. *Journal of Regional Science* 50(1): 119-141.

Nakaya, T., Fotheringham, A. S., Brunsdon, C. and Charlton, M. 2005. “Geographically Weighted Poisson Regression for Disease Association Mapping”. *Statistics in Medicine* 24: 2695-2717.

Paez, A., Farber, S. and Wheeler, D. 2011. “A Simulation-based Study of Geographically Weighted Regression as a Method for Investigating Spatially Varying Relationships”. *Environment and Planning A* 43(12): 2992-3010.

Silva, A. R. and Rodrigues, T. C. V. 2014. “Geographically Weighted Negative Binomial Regression - Incorporating Overdispersion”. *Statistics and Computing* 24: 769-783.

Silva, A. R. and Fotheringham, A. S. 2015. “The Multiple Testing Issue in Geographically Weighted Regression”. *Geographical Analysis*. Forthcoming.

Staniswalis, J. G. 1989. “The kernel Estimate of a Regression Function in Likelihood-based Models”. *Journal of the American Statistical Association* 84: 276-283.

Wheeler, D. and Tiefelsdorf, M. 2005. “Multicollinearity and Correlation among Local Regression Coefficients in Geographically Weighted Regression”. *Journal of Geographical Systems* 7(2): 161-187.

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

Name: Thais Carvalho Valadares Rodrigues
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: tatahcv@yahoo.com.br
Web: www.est.unb.br

APPENDIX I – SAS® MACRO

```
/************************************************/
/********* SAS MACROS  ***********************/
/**********************************************/

/**********************************************/
/********** CREATING PARAMETERS ESTIMATES********/
/************************************************/
%macro beta(par);
proc iml;
use _beta_;
        read all into b;
close _beta_;
n=nrow(b);
npar=&par+1;
%do i=0 %to &par;
        b&i=j(1,8,0);
        nome={"id" "geocod" "x" "y" "b" "sebi"
"tstat" "probtstat"};
        create b&i from b&i[colname=nome];
        do i=1 to (n/npar);      * from i=1 to N;
                b&i[1,]=b[(i-1)*npar+&i+1,];
                append from b&i;
        end;
%end;
quit;
%mend beta;



/**********************************************/
/***** PARAMETERS FOR STATIONARITY TEST *********/
/************************************************/
%macro vk(par);
proc iml;
use _beta_;
        read all into b;
close _beta_;

use _alpha_;
        read all var {alphai} into alpha;
close _alpha_;

n=nrow(b);
npar=&par+1;
n=n/npar;
vk=0;
%do i=0 %to &par;
        b&i=j(n,1,0);
        do i=1 to n;
                b&i[i,1]=b[(i-1)*npar+&i+1,5];
        end;
        vk&i=sum((b&i - b&i[:] )##2)/n ;
        vk=vk||vk&i;
%end;
vka= sum((alpha - alpha[:] )##2)/n ;
vk=vk||vka;
idx = setdif(1:(npar+2),1);
vk = vk[,idx];
create vk from vk;
append from vk;
quit;
%mend vk;

/************************************************/
/***** PERMUTATION FOR STATIONARITY TEST ********/
/***********************************************/


%macro perm(data=,geocod=,x=,y=);
proc iml;
        use &data;
        read all var{&geocod &x &y} into tab;
        close &data;
        n=nrow(tab);
    u = 1:n;
    call randgen(u, "Uniform");
    _u_=rank(u);
        create perm var{_u_};
        append;
quit;
data perm; merge perm &data(drop= &geocod &x y) ;
run;
proc sort data=perm; by _u_; run;
data perm; merge perm &data(keep=&geocod &x &y);
run;
%mend perm;

/*********************************************/
/*********** STATIONARITY TEST *****************/
/************************************************/


%macro
estac(data=,y=,x=,lat=,long=,h=,grid=,latg=,longg=,
gwr=,method=,alphag=,offset=,geocod=,rep=);
%let nvar=0;
```

```
%do %while(%scan(%str(&x),&nvar+1)~=);

    %let nvar=%eval(&nvar+1);

%end;

%gwnbr(data=&data,y=&y,x=&x,lat=&lat,long=&long,h=&
h,grid=&grid,

latg=&latg,longg=&longg,gwr=&gwr,method=&method,alp
hag=&alphag,

offset=&offset,geocod=&geocod);

%vk(&nvar);

data vk2; set vk; i=1; run;

%do it=2 %to (&rep+1);

         %perm(data=&data,geocod=&geocod,x=&long,y=
&lat);

         %gwnbr(data=perm,y=&y,x=&x,lat=&lat,long=&
long,h=&h,grid=&grid,

         latg=&latg,longg=&longg,gwr=&gwr,method=&m
ethod,alphag=&alphag,

         offset=&offset,geocod=&geocod);

         %vk(&nvar);

         data vk; set vk; i=&it; run;

         proc append base=vk2 data=vk force; run;

%end;


proc iml;

use vk2;

         read all into x;

close vk2;

nvar=ncol(x)-1;

n=nrow(x);

count=j(1,nvar,0);

do v=1 to nvar;

         do i=1 to n;

                  if x[i,v]>=x[1,v] then
count[v]=count[v]+1;

         end;

end;

count=count/n*100;

print count;

varnames="b0":"b&nvar"||"alpha";

create pvalor_est from count [colname=varnames];

append from count;

quit;

%mend estac;




/************************************************/

/************ GOLDEN SECTION SEARCH ***********/

/******************************************/
```

```
%macro
golden(data=,y=,x=,lat=,long=,method=,type=,gwr=,of
fset=,out=);

proc iml;

use &data;

         read all var {&y} into y;

         read all var {&x} into x;

         read all var{&long &lat} into COORD;

         n=nrow(y);

         %if &offset= %then %do; offset=j(n,1,0);
%end;

         %else %do; read all var {&offset} into
offset; %end;

close &data;

x=j(n,1,1)||x;

method= "&method"; *fixed, adaptive1 ou adaptiven;

type="&type"; *aic , cv ou dev ;

gwr="&gwr"; *global, local, poisson;

print method type gwr;


start dist(coord,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;

                 if abs(coord[,1])<180 then do;

                     dif=abs(COORD[i,1]-
COORD[j,1]);

                     raio=arcos(-1)/180;


         ang=sin(COORD[i,2]*raio)*sin(COORD[j,2]*ra
io)+cos(COORD[i,2]*raio)*cos(COORD[j,2]*raio)*cos(d
if*raio);

                                arco=arcos(ang);

                  d[1]=i;

                  d[2]=j;

                  d[3]=arco*6371 /*Earth's Radius =
6371 (approximately)*/;

                  append from d;

                 end;

                 else do;

                     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;

         end;

         close _dist_;
```

```
finish dist;
run dist(coord,n);
use _dist_;
        read all into d;
        maxd=int(max(d[,3])+1);
        free d;
close _dist_;
if      method= "adaptive1" then do;
        h0= 5 ; h3= n;
end;
else if method= "adaptiven" | method= "fixed" then
do;
        h0= 0 ; h3= maxd;
end;
r=0.61803399; c=1-r;
if method= "adaptive1" then tol=0.9; else tol=0.1;
h1=h0+(1-r)*(h3-h0);
h2=h0+r*(h3-h0);
print h0 h1 h2 h3;

start cv(h) global(method, n, coord, x, y, type,
maxd, gwr, offset);
        alphaii= j(n,2,0);
        yhat=j(n,1,0);
        S=j(n,n,0);
        if gwr="global" then do;
                ym=sum(y)/nrow(y);
                u=(y+ym)/2;
                n=log(u);
                par=1; ddpar=1; j=0; aux2=0;
                do while (abs(ddpar)>0.00001);
                        aux1=0; dpar=1;
parold=par;
                        do while
(abs(dpar)>0.001);
                                aux1=aux1+1;
                                if par<0 then do;

        par=0.00001;
                                end;

        par=choose(par<1E-10,1E-10,par);
                        g=sum(digamma(par+y)-
digamma(par)+log(par)+1-log(par+u)-
(par+y)/(par+u));
                        hess=sum(trigamma(par+y)-
trigamma(par)+1/par-
2/(par+u)+(y+par)/((par+u)#(par+u)));

        hess=choose(abs(hess)<1E-23,sign(hess)*1E-
23,hess);
        hess=choose(hess=0,1E-23,hess);
                        par0=par;
                        par=par0-inv(hess)*g;
                                if aux1>50 &
par>1E5 then do;
                                        dpar=
0.0001;

        aux2=aux2+1;
                                        if
aux2=1 then par=2 ;
                                        else if
aux2=2 then par=1E5;
                                        else if
aux2=3 then par=0.0001;
                                end;
                                else dpar=par-
par0;
                        end;
                        a=1/par; dev=0; ddev=1;
i=0;
                        do while
(abs(ddev)>0.00001);
                                i=i+1;
                        w=(u/(1+a*u))+(y-
u)#(a*u/(1+2*a*u+a*a*u#u));
                        z=n+(y-u)/(w#(1+a*u)) -
offset;
                        b=inv((x#w)`*x)*(x#w)`*z;
                        n=x*b + offset;
                        u=exp(n);
                        olddev=dev;
                                tt=y/u;

        tt=choose(tt=0,1E-10,tt);
                        dev=2*sum(y#log(tt)-
(y+1/a)#log((1+a*y)/(1+a*u)));
                        ddev=dev-olddev;
                        end;
                        if aux2>4 then ddpar=1E-
9;
                        else ddpar=par-parold;
                end;
                alpha=a;
        end;
        n=nrow(y);
        aux2=0;
        do i=1 to n;
                d=j(1,3,0);
                dist=d;
       do j=1 to n;
               if abs(coord[,1])<180 then do;
```

```
                dif=abs(COORD[i,1]-COORD[j,1]);

                raio=arcos(-1)/180;


        ang=sin(COORD[i,2]*raio)*sin(COORD[j,2]*ra
io)+cos(COORD[i,2]*raio)*cos(COORD[j,2]*raio)*cos(d
if*raio);

                                  if i=j then
arco=0;

                else arco=arcos(ang);

                d1=arco*6371;

                 end;

                 else d1=sqrt((COORD[i,1]-
COORD[j,1])**2+(COORD[i,2]-COORD[j,2])**2);

                          d[1]=i; d[2]=j; d[3]=d1;

                          if j=1 then dist=d;

                          else dist=dist//d;

                 end;

                 u=nrow(dist);

                 w=j(u,1,0);

                 if method= "fixed" then do;

                          if type="cv" then do;

                          do jj=1 to u;

                                           if
dist[jj,3]<=maxd*0.8 & dist[jj,3]^=0 then
w[jj]=exp(-0.5*(dist[jj,3]/h)**2);

                                           else
w[jj]=   0;

                          end;

                          end;

                          else do;

                          do jj=1 to u;

                                           if
dist[jj,3]<=maxd*0.8 then w[jj]=exp(-
0.5*(dist[jj,3]/h)**2);

                                           else
w[jj]=   0;

                          end;

                          end;

                 end;

                 else if method= "adaptiven" then
do;

                          if type="cv" then do;

                          do jj=1 to u;

                                           if
dist[jj,3]<=h & dist[jj,3]^=0 then w[jj]=(1-
(dist[jj,3]/h)**2)**2;

                                           else
w[jj]=   0;

                          end;

                          end;

                          else do;

                          do jj=1 to u;

                                           if
dist[jj,3]<=h then w[jj]=(1-(dist[jj,3]/h)**2)**2;

                                           else
w[jj]=   0;

                         end;

                          end;

                 end;

                 else if method= "adaptive1" then
do;

                          call sort(dist,{3});

                          dist=dist||(1:n)`;

                          w=j(n,2,0);

                          hn=dist[h,3];

                          if type="cv" then do;

                                  do jj=1 to n;

                                           if
dist[jj,4]<= h & dist[jj,3]^=0 then w[jj,1]=(1-
(dist[jj,3]/hn)**2)**2;

                                           else
w[jj,1]=0;


        w[jj,2]=dist[jj,2];

                                  end;

                          end;

                          else do;

                                  do jj=1 to n;

                                           if
dist[jj,4]<=h then w[jj,1]=(1-
(dist[jj,3]/hn)**2)**2;

                                           else
w[jj,1]=0;


        w[jj,2]=dist[jj,2];

                                  end;

                          end;

                          call sort(w,{2});

                 end;

                 wi=w[,1];

                 ym=sum(y)/nrow(y);

                 uj=(y+ym)/2;

                 nj=log(uj);

                 if i=1 | aux2=5 then par=1; else
par=alphaii[i-1,2];

                 ddpar=1; jj=0; count=0; aux2=0;

                 do while (abs(ddpar)>0.000001);

                          aux1=0;

                          dpar=1;

                          parold=par;

                          if gwr="global" |
gwr="poisson" then do;

                                  dpar=0.00001;
```

```
                                if gwr=  "global"
then par=1/a;
                        end;
                        /* computing alpha=1/par,
where par=theta */
                        do while
(abs(dpar)>0.001);
                                aux1=aux1+1;
                                if gwr="local"
then do;

        par=choose(par<1E-10,1E-10,par);
                        g=sum((digamma(par+y)-
digamma(par)+log(par)+1-log(par+uj)-
(par+y)/(par+uj))#w[,1]);

        hess=sum((trigamma(par+y)-
trigamma(par)+1/par-
2/(par+uj)+(y+par)/((par+uj)#(par+uj)))#w[,1]);
                                end;

        hess=choose(abs(hess)<1E-23,sign(hess)*1E-
23,hess);

        hess=choose(hess=0,1E-23,hess);
                par0=par;
                par=par0-inv(hess)*g;
                                if par<=0 then
do;

        count=count+1;
                                        if
count<10 then par=0.000001;
                                        else
par=abs(par);
                                end;
                                if aux1>50 &
par>1E5 then do;
                                        dpar=
0.0001;

        aux2=aux2+1;
                                        if
aux2=1 then par=2 ;
                                        else if
aux2=2 then par=1E5;
                                        else if
aux2=3 then par=0.0001;
                                end;
                                else do;

        dpar=par-par0;
                                        if
par<1E-3 then dpar=dpar*100;
                                end;
                        end;
                        if gwr=  "poisson" then
alpha=0;
```

```
                        else alpha=1/par;
                        dev=0; ddev=1; cont=0;
                        /* computing beta */
                        do while
(abs(ddev)>0.000001);
                                cont=cont+1;

        uj=choose(uj>1E100,1E100,uj);
                                aux=
(alpha*uj/(1+2*alpha*uj+alpha*alpha*uj#uj));
                Ai=(uj/(1+alpha*uj))+(y-uj)#aux;

        Ai=choose(Ai<=0,1E-5,Ai);
                zj=nj+(y-uj)/(Ai#(1+alpha*uj)) -
offset;
                                if
det(x`*(wi#Ai#x))=0 then bi=j(ncol(x),1,0);
                                else
bi=inv(x`*(wi#Ai#x))*x`*(wi#Ai#zj);
                nj=x*bi + offset;

        nj=choose(nj>1E2,1E2,nj);
                uj=exp(nj);
                olddev=dev;
                                uj=choose(uj<1E-
150,1E-150,uj);
                                tt=y/uj;

        tt=choose(tt=0,1E-10,tt);
                                if gwr=
        "poisson" then dev=2*sum(y#log(tt)-(y-
uj));
                                else
dev=2*sum(y#log(tt)-
(y+1/alpha)#log((1+alpha*y)/(1+alpha*uj)));
                                if cont>100 then
ddev= 0.0000001;
                else ddev=dev-olddev;
                        end;
                        jj=jj+1;
                        if gwr="global" |
gwr="poisson" | aux2>4 | jj>50 | ddpar=0.0000001
then ddpar=1E-9;
                        else do;
                                ddpar=par-parold;
                                if par<1E-3 then
ddpar=ddpar*100;
                        end;
                end;
                Ai2=(uj/(1+alpha*uj))+(y-
uj)#(alpha*uj/(1+2*alpha*uj+alpha*alpha*uj#uj));
                if Ai2[><,]<1E-5 then
Ai2=choose(Ai2<1E-5,1E-5,Ai2);
                Ai=Ai2;
                if det(x`*(wi#Ai#x))=0 then
S[i,]=j(1,n,0);

                else S[i,]=
x[i,]*inv(x`*(wi#Ai#x))*(x#wi#Ai)`;

                yhat[i]=uj[i];

                alphaii[i,1]=i;

                alphaii[i,2]= alpha;

        end;

        alpha= alphaii[,2];

        yhat=choose(yhat<1E-150,1E-150,yhat);

        tt=y/yhat;

        tt=choose(tt=0,1E-10,tt);

        if gwr=  "poisson" then
dev=2*sum(y#log(tt)-(y-yhat));

        else dev=2*sum(y#log(tt)-
(y+1/alpha)#log((1+alpha#y)/(1+alpha#yhat)));

        if gwr ^=        "poisson" then do;

        a2=y+1/alpha; b2=1/alpha; c2=y+1;

        end;

        else do;

        a2=y; b2=1/(alpha+1e-8); c2=y+1;

        end;

        algamma=j(n,1,0); blgamma=j(n,1,0);
clgamma=j(n,1,0);

        do i=1 to nrow(y);

                algamma[i]=lgamma(a2[i]);
blgamma[i]=lgamma(b2[i]); clgamma[i]=lgamma(c2[i]);

        end;

        if gwr^="poisson" then do;

                ll=sum(y#log(alpha#yhat)-
(y+1/alpha)#log(1+alpha#yhat)+ algamma - blgamma -
clgamma );

                npar=trace(S)+1;

        end;

        else do;

                ll=sum(-yhat+y#log(yhat)-clgamma);

                npar=trace(S);

        end;

        /*AIC= 2*npar + dev;*/

        AIC= 2*npar -2*ll;

        AICC= AIC +(2*npar*(npar+1))/(n-npar-1);

        CV=(y-yhat)`*(y-yhat);

        res=cv||aicc||npar||dev;

        return (res);

finish;


if type="cv" then do;

        pos=1;

        create &out var{h1 res1 h2 res2};

end;

else do;

        if type="aic" then pos=2;

        else pos=4;

        create &out var{h1 res1 npar1 h2 res2
npar2};

end;

res1=cv(h1); npar1=res1[3]; res1=res1[pos];

res2=cv(h2); npar2=res2[3]; res2=res2[pos];

append;

do while(abs(h3-h0) > tol*2);

    if res2<res1 then do;

        h0=h1;

                h1=h2;

        h2=c*h1+r*h3;

        res1=res2;

                npar1=npar2;

                res2=cv(h2);

                npar2=res2[3];

                res2=res2[pos];

    end;

    else do;

        h3=h2;

        h2=h1;

        h1=c*h2+r*h0;

        res2=res1;

                npar2=npar1;

                res1=cv(h1);

                npar1=res1[3];

                res1=res1[pos];

    end;

        append;

end;

if method= "adaptive1" then do;

        xmin = (h3+h0)/2;

        h2=ceil(xmin);

        h1=floor(xmin);

        golden1 = cv(h1);

        g1= golden1[pos];

        golden2= cv(h2);

        g2= golden2[pos];

        npar1=golden1[3];

        res1=golden1[pos];

        npar2=golden2[3];

        res2=golden2[pos];

        append;

        if g1<g2 then do;

                xmin=h1;
                npar=golden1[3];
                golden=g1;
        end;
        else do;
                xmin=h2;
                npar=golden2[3];
                golden=g2;
        end;
end;
else do;
        xmin = (h3+h0)/2;
        golden = cv(xmin);
        npar=golden[3];
        golden=golden[pos];
end;
h1 = xmin;
res1 = golden;
npar1=npar;
h2 = .;
res2 = .;
npar2=.;
append;
if type="cv" then print golden xmin;
else print golden xmin npar;
quit;
%mend golden;




/**********************************************/
/*************** GWNBR  **********************/
/**********************************************/


%macro
gwnbr(data=,y=,x=,lat=,long=,h=,grid=,latg=,longg=,
gwr=,method=,alphag=,offset=,geocod=,out=);
proc iml;
use &data;
        read all var {&y} into y;
        read all var {&x} into x;
        read all var{&long &lat} into COORD;
        n=nrow(y);
        %if &offset= %then %do; offset=j(n,1,0);
%end;
        %else %do; read all var {&offset} into
offset; %end;
        %if &grid= %then %do;
                read all var{&long &lat} into
POINTS;
                read all var{&geocod} into
geocod_;
        %end;
close &data;
%if &grid^= %then %do;
        use &grid;
        read all var{&longg &latg} into POINTS;
        close &grid;
        geocod_=nrow(points,1,0);
%end;
x=j(n,1,1)||x;
yhat=j(n,1,0);
h=&h;
gwr="&gwr"; *global,local, poisson;
method="&method"; *fixed, adaptive1, adaptiven;
m=nrow(POINTS);
bii=j(ncol(x)*m,2,0); alphaii= j(m,2,0);
xcoord=j(ncol(x)*m,1,0); ycoord=j(ncol(x)*m,1,0);
&geocod= j(ncol(x)*m,1,0);
sebi=j(ncol(x)*m,1,0); sealphai= j(m,1,0);
S=j(n,n,0);
yp=y-sum(y)/n;
probai=j(m,1,0); probbi=j(m,1,0);
yhat=j(m,1,0);
res= j(m,1,0);
if gwr^="poisson" then do;
        ym=sum(y)/nrow(y);
        u=(y+ym)/2;
        n=log(u);
        par=1; ddpar=1; j=0; aux2=0;
        do while (abs(ddpar)>0.00001);
                aux1=0;
                dpar=1;
                parold=par;
                do while (abs(dpar)>0.001);
                        aux1=aux1+1;
                        if par<0 then
par=0.00001;
                        par=choose(par<1E-10,1E-
10,par);
                g=sum(digamma(par+y)-
digamma(par)+log(par)+1-log(par+u)-
(par+y)/(par+u));
                hess=sum(trigamma(par+y)-
trigamma(par)+1/par-
2/(par+u)+(y+par)/((par+u)#(par+u)));
                        hess=choose(abs(hess)<1E-
23,sign(hess)*1E-23,hess); *CONFERIR!!!;
                        hess=choose(hess=0,1E-
23,hess);
```

```
                  par0=par;
                  par=par0-inv(hess)*g;
                            if aux1>50 & par>1E5 then
do;
                                    dpar= 0.0001;
                                    aux2=aux2+1;
                                    if aux2=1 then
par=2 ;
                                    else if aux2=2
then par=1E5;
                                    else if aux2=3
then par=0.0001;
                            end;
                            else dpar=par-par0;
                  end;
                  a=1/par; dev=0; ddev=1; i=0;
                  do while (abs(ddev)>0.00001);
                            i=i+1;
                  w=(u/(1+a*u))+(y-
u)#(a*u/(1+2*a*u+a*a*u#u));
                            w=choose(w<=0,1E-5,w);
                            z=n+(y-u)/(w#(1+a*u)) -
offset;
                  b=inv((x#w)`*x)*(x#w)`*z;
                  n=x*b + offset;
                            n=choose(n>1E2,1E2,n);
                  u=exp(n);
                  olddev=dev;
                            tt=y/u;
                            tt=choose(tt=0,1E-10,tt);
                  dev=2*sum(y#log(tt)-
(y+1/a)#log((1+a*y)/(1+a*u)));
                  ddev=dev-olddev;
                  end;
                  if aux2>4 then ddpar=1E-9;
                  else ddpar=par-parold;
         end;
         %if &alphag= %then %do; alphag=a;%end;
         %else %if &alphag=0 %then %do; alphag=1e-
8;%end;
         %else %do; alphag=&alphag;%end;
         bg=b;
         parg=par;
end;
if gwr="global" then print alphag aux2;
n=nrow(y);
aux2=0;
do i=1 to m;
         d=j(1,3,0);
         do j=1 to n;
                  if abs(COORD[,1])<180 then do;
         dif=abs(POINTS[i,1]-COORD[j,1]);
         raio=arcos(-1)/180;

         ang=sin(POINTS[i,2]*raio)*sin(COORD[j,2]*r
aio)+cos(POINTS[i,2]*raio)*cos(COORD[j,2]*raio)*cos
(dif*raio);
                           if
round(ang,0.000000001)=1 then arco=0;
         else arco=arcos(ang);
         d1=arco*6371 /*Earth's Radius = 6371
(approximately)*/;
        end;
        else d1=sqrt((POINTS[i,1]-
COORD[j,1])**2+(POINTS[i,2]-COORD[j,2])**2);
                  d[1]=i; d[2]=j; d[3]=d1;
                  if j=1 then dist=d;
         *cleaning dist where i value changes;
                  else dist=dist//d;
         end;
         w=j(n,1,0);
         if method= "fixed" then do;
        do jj=1 to n;
                           w[jj]=exp(-
0.5*(dist[jj,3]/h)**2);
        end;
         end;
         else if method= "adaptiven" then do;
        do jj=1 to n;
                           if dist[jj,3]<=h then
w[jj]=(1-(dist[jj,3]/h)**2)**2;
                           else w[jj]=      0;
        end;
         end;
         else if method= "adaptive1" then do;
                  w=j(n,2,0);
                  call sort(dist,{3});
                  dist=dist||(1:n)`;
                  hn=dist[h,3]; *bandwith for the
point i;
                  do jj=1 to n;
                           if dist[jj,4]<=h then
w[jj,1]=(1-(dist[jj,3]/hn)**2)**2;
                           else w[jj,1]=0;
                           w[jj,2]=dist[jj,2];
                  end;
                  call sort(w,{2});
         end;
         wi=w[,1];
         ym=sum(y)/nrow(y);
         uj=(y+ym)/2;
        nj=log(uj);

        ddpar=1; jj=0; count=0; aux2=0;

        if i=1 | aux2=5 | count=4 then par=1; else
par=alphaii[i-1,2];

        do while (abs(ddpar)>0.000001);

                dpar=1;

                if ddpar=1 then parold=1.8139;

                else parold=par;

                aux1=0;

                if gwr="global" | gwr="poisson"
then do;

                        dpar=0.00001;

                        if gwr=  "global" then
par=1/alphag;

                end;

                /* computing alpha=1/par, where
par=theta=r */

                do while (abs(dpar)>0.001);

                        aux1=aux1+1;

                        if gwr="local" then do;


        par=choose(par<1E-10,1E-10,par);

                g=sum((digamma(par+y)-
digamma(par)+log(par)+1-log(par+uj)-
(par+y)/(par+uj))#w[,1]);

                hess=sum((trigamma(par+y)-
trigamma(par)+1/par-
2/(par+uj)+(y+par)/((par+uj)#(par+uj)))#w[,1]);

                        end;

        par0=par;

                        hess=choose(abs(hess)<1E-
23,sign(hess)*1E-23,hess);

                        hess=choose(hess=0,1E-
23,hess);

        par=par0-inv(hess)*g;

                        if par<=0 then do;

                                count=count+1;

                                if count=1 then
par=0.000001;

                                else if count=2
then par=0.0001;

                                else
par=1/alphag;

                        end;

                        if aux1>100 & par>1E5
then do; *MAXINTA;

                                dpar= 0.0001;

                                if aux2=0 then
par=1/alphag + 0.0011;

                                if aux2=1 then
par=2 ;

                                else if aux2=2
then par=1E5;

                                else if aux2=3
then par=0.0001;

                                aux2=aux2+1;

                        end;

                        else do;

                                dpar=par-par0;

                                if par<1E-3 then
dpar=dpar*100;

                        end;

                end;

                if gwr=  "poisson" then alpha=0;


                else alpha=1/par;

                dev=0; ddev=1; cont=0;

                /* computing beta */

                do while (abs(ddev)>0.000001);

                        cont=cont+1;

        Ai=(uj/(1+alpha*uj))+(y-
uj)#(alpha*uj/(1+2*alpha*uj+alpha*alpha*uj#uj));

                        Ai=choose(Ai<=0,1E-5,Ai);

        zj=nj+(y-uj)/(Ai#(1+alpha*uj))-offset;

                        if det(x`*(wi#Ai#x))=0
then bi=j(ncol(x),1,0);

                        else
bi=inv(x`*(wi#Ai#x))*x`*(wi#Ai#zj);

        nj=x*bi + offset;

                        nj=choose(nj>1E2,1E2,nj);

        uj=exp(nj);

        olddev=dev;

                        uj=choose(uj<1E-150,1E-
150,uj);

                        tt=y/uj;

                        tt=choose(tt=0,1E-10,tt);

                        if gwr=  "poisson" then
dev=2*sum(y#log(tt)-(y-uj));

                        else dev=2*sum(y#log(tt)-
(y+1/alpha)#log((1+alpha*y)/(1+alpha*uj)));

                        if cont>100 then ddev=
0.0000001; *MAXINTB;

        else ddev=dev-olddev;

                end;

                jj=jj+1;

                *print jj bi;

                if gwr="global" | gwr="poisson" |
aux2>4 | count>3 | jj>200 then ddpar=1E-9;

                else do;

                        ddpar=par-parold;

                        if par<1E-3 then
ddpar=ddpar*100;

                end;

    /* print j aux1 cont aux2 count parold par
ddpar;*/

        end;

        if aux2>4 then probai[i]=1;
        if count>3 then probai[i]=2;

    Ai2=(uj/(1+alpha*uj))+(y-
uj)#(alpha*uj/(1+2*alpha*uj+alpha*alpha*uj#uj));

        if Ai2[><,]<1E-5 then do;

                probbi[i]=1;

                Ai2=choose(Ai2<1E-5,1E-5,Ai2);

        end;

        Ai=Ai2;

        %if &grid= | &grid=&data %then %do;

                if det(x`*(wi#Ai#x))=0 then
S[i,]=j(1,n,0);

                else S[i,]=
x[i,]*inv(x`*(wi#Ai#x))*(x#wi#Ai)`;

        %end;

        C=inv(x`*(wi#Ai#x));

        varb= C;

        seb=sqrt(vecdiag(varb));

        if gwr^="poisson" then do;

                ser=sqrt(1/abs(hess));

                r=1/alpha;

                sealpha=ser/(r**2);

                sealphai[i,1]=sealpha;

                alphaii[i,1]=i;

                alphaii[i,2]= alpha;

        end;

        m1=(i-1)*ncol(x)+1;

        m2=m1+(ncol(x)-1);

        sebi[m1:m2,1]=seb;

        bii[m1:m2,1]=i;

        bii[m1:m2,2]=bi;

        xcoord[m1:m2,1]= POINTS[i,1];

        ycoord[m1:m2,1]= POINTS[i,2];

        &geocod[m1:m2,1]= geocod_[i,1];

        %if &grid= | &grid=&data %then %do;

                yhat[i]=uj[i];

        %end;

end;

tstat= bii[,2]/sebi;

probtstat=2*(1-probnorm(abs(tstat)));

if gwr^="poisson" then do;

        atstat= alphaii[,2]/sealphai;

        aprobtstat=2*(1-probnorm(abs(atstat)));
        *check for normality;

end;

else do;

        atstat=j(n,1,0);

        aprobtstat=j(n,1,1);

end;

b=bii[,2];

alphai=alphaii[,2];

_id_=   bii[,1];

_ida_=alphaii[,1];


_beta_=shape(bii[,1:2],n);

i=do(2,ncol(_beta_),2);

_beta_=_beta_[,i];

call qntl(qntl,_beta_);

qntl=qntl//(qntl[3,]-qntl[1,]);

descriptb=_beta_[:,]//_beta_[><,]//_beta_[<>,];


print qntl[label="Quantiles of GWNBR Parameter
Estimates"

rowname={"P25", "P50", "P75", "IQR"}
colname={'Intercept' &x}],,

descriptb[label="Descriptive Statistics"
rowname={"Mean", "Min", "Max"}

colname={'Intercept' &x}];


_stdbeta_=shape(sebi,n);

call qntl(qntls,_stdbeta_);

qntls=qntls//(qntls[3,]-qntls[1,]);

descripts=_stdbeta_[:,]//_stdbeta_[><,]//_stdbeta_[
<>,];


print qntls[label="Quantiles of GWNBR Standard
Errors"

rowname={"P25", "P50", "P75", "IQR"}
colname={'Intercept' &x}],,

descripts[label="Descriptive Statistics of Standard
Errors" rowname={"Mean", "Min", "Max"}

colname={'Intercept' &x}];


%if &grid= | &grid=&data %then %do;

        yhat=choose(yhat<1E-150,1E-150,yhat);

        tt=y/yhat;

        tt=choose(tt=0,1E-10,tt);

        if gwr=  "poisson" then do;

                dev=2*sum(y#log(tt)-(y-yhat));

                devnull=2*sum(y#log(y/y[:])-(y-
y[:]));

                pctdev=1-dev/devnull;

        end;

        else do;

                dev=2*sum(y#log(tt)-
(y+1/alphai)#log((1+alphai#y)/(1+alphai#yhat)));

                devnull=2*sum(y#log(y/y[:])-
(y+1/alphai)#log((1+alphai#y)/(1+alphai#y[:])));

                pctdev=1-dev/devnull;

        end;
        if gwr^="poisson" then do;

                a2=y+1/alphai; b2=1/alphai;

                algamma=j(n,1,0);
blgamma=j(n,1,0);

                do i=1 to nrow(y);

                        algamma[i]=lgamma(a2[i]);

                        blgamma[i]=lgamma(b2[i]);

                end;

        end;

        c2=y+1;

        clgamma=j(n,1,0);

        do i=1 to nrow(y);

                clgamma[i]=lgamma(c2[i]);

        end;

        if gwr^="poisson" then do;

                ll=sum(y#log(alphai#yhat)-
(y+1/alphai)#log(1+alphai#yhat)+ algamma - blgamma
- clgamma );

                if gwr="global" & alphai^=1/parg
then npar=trace(S);

                else npar=trace(S)+1;

                ll1=sum(y#log(y/(alphai#yhat))-
y+(y+1/alphai)#log(1+alphai#yhat)-algamma+blgamma);

                llnull=sum(y#log(y/y[:]));

                pctll=1-ll1/llnull;

        end;

        else do;

                ll=sum(-yhat+y#log(yhat)-clgamma);

                npar=trace(S);

                pctll=pctdev;

        end;

        adjpctdev=1-((nrow(y)-1)/(nrow(y)-
npar))*(1-pctdev);

        adjpctll=1-((nrow(y)-1)/(nrow(y)-
npar))*(1-pctll);

        resord=y-yhat;

        sigma2=  (resord`*resord)/(n-npar);

        sii=vecdiag(S);

        res=resord/sqrt(sigma2#(1-sii));

        res=unique(_id_)`||COORD[,1]||COORD[,2]||y
||yhat||res||resord;

        /*AIC= 2*npar + dev;*/

        AIC= 2*npar - 2*ll;

        AICC= AIC +(2*npar*(npar+1))/(n-npar-1);

        BIC= npar*log(n) - 2*ll ;

        _malpha_=0.05*(ncol(x)/npar);

        _t_critical_=abs(tinv(_malpha_/2,n-npar));


        print _malpha_[label="alpha-level=0.05"]
_t_critical_[format=comma6.2 label="t-Critical"]
npar;

        print gwr method ll dev pctdev adjpctdev
pctll adjpctll npar aic aicc bic;

        create _res_ from res[colname={"_id_"
"xcoord" "ycoord" "yobs" "yhat" "res" "resraw"}];

        append from res;

        stat=ll|| dev|| pctdev || adjpctdev||
pctll || adjpctll || npar|| aic|| aicc|| bic;

        create _stat_ from stat[colname={"l1"
"dev" "pctdev" "adjpctdev" "pctll" "adjpctll"
"npar" "aic" "aicc" "bic"}];

        append from stat;

%end;

%else %do; print gwr method; %end;


create _beta_ var{_id_ &geocod xcoord ycoord b sebi
tstat probtstat}; * _beta_ has beta vector for each
point i;

append;

xcoord=COORD[,1];ycoord=COORD[,2];

&geocod=unique(&geocod)`;

sig_alpha=j(n,1,"not significant at 90%");

v1=npar;

do i=1 to n;

if aprobtstat[i]<0.01*(ncol(x)/v1) then
sig_alpha[i]="significant at 95%";

else if aprobtstat[i]<0.1*(ncol(x)/v1) then
sig_alpha[i]="significant at 90%";

else sig_alpha[i]="not significant at 90%";

end;

create _alpha_ var{_ida_ &geocod xcoord ycoord
alphai sealphai atstat aprobtstat sig_alpha probai
probbi}; * _alpha_ has alpha vector for each point
i;

append;

_tstat_=_beta_/_stdbeta_;

_probt_=2*(1-probnorm(abs(_tstat_)));

_bistdt_=geocod_||COORD||_beta_||_stdbeta_||_tstat_
||_probt_;

_colname1_={"Intercept" &x};

_label_=repeat("std_",ncol(x))//repeat("tstat_",nco
l(x))//repeat("probt_",ncol(x));

_colname_={"&geocod" "x"
"y"}||_colname1_||concat(_label_,repeat(_colname1_`
,3))`;

call change(_colname_, "_ ", "_");

call change(_colname_, "_ ", "_");

create _parameters_ from
_bistdt_[colname=_colname_];

append from _bistdt_;

close _parameters_;


_sig_=j(n,ncol(x),"not significant at 90%");

v1=npar;

do i=1 to n;
do j=1 to ncol(x);

if _probt_[i,j]<0.01*(ncol(x)/v1) then
_sig_[i,j]="significant at 99%";

else if _probt_[i,j]<0.05*(ncol(x)/v1) then
_sig_[i,j]="significant at 95%";

else if _probt_[i,j]<0.1*(ncol(x)/v1) then
_sig_[i,j]="significant at 90%";

else _sig_[i,j]="not significant at 90%";

end;

end;

_colname1_={"Intercept" &x};

_label_=repeat("sig_",ncol(x));

_colname_=concat(_label_,repeat(_colname1_`,1))`;

create _sig_parameters2_ from
_sig_[colname=_colname_];

append from _sig_;

/*

%let nvar=0;

%do %while(%scan(%str(&x),&nvar+1)~=);

    %let nvar=%eval(&nvar+1);

%end;

use _beta_;

         read all into b;

close _beta_;

n=nrow(b);

npar=&nvar+1;

%do i=0 %to &nvar;

         b&i=j(1,8,0);

         nome={"_id_" "&geocod" "xcoord" "ycoord"
"b" "sebi" "tstat" "probtstat"};

         create &out._b&i from b&i[colname=nome];

         do i=1 to (n/npar);

                  b&i[1,]=b[(i-1)*npar+&i+1,];

                  append from b&i;

         end;

%end;

*/

quit;

%mend gwnbr;
```

```
%macro map(data=,var=, map=, geocod=);

goptions reset=all;

proc gmap data=&data map=&map all;

id &geocod;

choro &var / legend=legend1;

legend1 position=(middle right) across=1
mode=reserve label=(position=top j=c);

run;

quit;

%mend map;
```