Title
Final Report to the EPA on Multilevel Models for Generalization

Authors
de Leeuw, Jan
Berk, Richard

Publication Date
2003

FINAL REPORT TO THE EPA ON MULTILEVEL MODELS FOR GENERALIZATION

JAN DE LEEUW
RICHARD A. BERK
DEPARTMENT OF STATISTICS
UCLA

1. EXECUTIVE SUMMARY

Multilevel statistical models are characterized by analyses undertaken simultaneously at different levels of aggregation or spatial/temporal scales. For example, one might study several reaches in a stream for a number of different research sites. Or one might study several transects in each of several forests. The basic idea in multilevel models is to have a regression equation characterizing relationships at the smaller, or micro, level and then have one or more of the regression coefficients at the micro level a function of predictors at the macro level. At the micro level, for instance, taxa richness may be a function of stream velocity (and other things). Then at the macro level, the regression coefficient linking stream velocity to taxa richness may be a function of proximity of the stream to land used for agriculture. Thus, one can address how the relationship between stream velocity and taxa richness varies (or not) in different locations, here with locale characterized by proximity to land use for agriculture. That is, one can learn when to generalize over sites and when not to generalize over sites. One can also learn how different temporal and/or spatial scales are linked.

These sorts of relationships can also be formulated as interaction effects within conventional regression models. However, the usual estimation procedures used for those models will not properly characterize the uncertainty in the results, so that the usual confidence intervals and hypothesis tests will be wrong. Special estimation procedures are required. Such procedures are well known and widely available in existing software.

Our goal was to extend multilevel models to more complicated and realistic situations. The first extension was to allow for spatial autocorrelation in the residuals of multilevel models. The problem addressed is that more proximate spatial units at the micro level can be expected to have disturbances that are more alike than spatial units at the micro level that are more distant from one another. Thus, transects that are closer together will likely have disturbances that are more alike than transects that are farther apart.

Failing to take this spatial autocorrelation into account will generally lead to biased standard errors, and hence, inaccurate confidence intervals and hypothesis tests.

Formally, a good solution to this problem for linear regression can be found in a classic paper by J. Keith Ord. For the usual sorts of regression models, one constructs a matrix capturing the distance between all micro units within each macro unit (e.g., transects within sites) and builds that information into the estimation process. We initially adopted this approach, introduced it into a multilevel formulation, and applied it to two data sets. One data set was collected to study biodiversity in streams located in Ventura County, California, and the other was collected to study the impact of marine preserves on biodiversity and total fish biomass in coral reefs in the Philippines.

The results were disappointing. First, there was essentially no theory or empirical research in ecology or related disciplines to inform in sufficient detail the construction of the distance matrix. One difficulty was that it was not clear how to measure distance given ocean currents, for example, that transport of nutrients more readily between some locals than others. Another difficulty was that there are a number of different functions of that distance that could have been used in the distance matrix (e.g., exponential decay with increasing distance) and, again no guidance from the scientific literature. It is our sense that similar problems are common for a wide variety of environmental applications.

Second, except for very simple and somewhat unrealistic models, the numerical methods used in the estimation did not perform well. There were several technical reasons, but a key obstacle was that the regression coefficients and the distance matrix were competing for the much the same information. This was because the predictors necessarily also contained spatial information. Micro units that were closer were also likely to be more similar in the values of predictors than micro units that are farther apart. Such predictors could include composition of the streambed and the amount of shading from trees along the banks, for instance. Because of the competition for spatial information, the output from the statistical models tended to be very unstable. Small changes in the model or the data could introduce large changes in the output.

Finally, we planned move beyond multilevel linear models to multilevel generalized linear models. That way, we would be able to include popular procedures such as logistic regression for binary outcomes and Poisson regression for count data. Unfortunately, the Ord approach led to effectively intractable mathematical problems when applied to generalized linear models.

These three difficulties forced us to reconsider the entire project and indeed, the usual philosophy by which spatial modeling is undertaken. To begin, we suspect that for spatial regression models, far too much is made about the exact form of the distance matrix. With scant scientific guidance about how the distance matrix should be formulated, any one of several competing formulations can be applicable. But there is no way to know which. In addition, the distance matrix by itself is rarely of much scientific interest. Its usual role is to allow for more accurate estimates of the regression coefficients that are the real focus of scientific concern. In statistical parlance, the distance matrix represents a set of "nuisance parameters." At a deeper level, George Boxs famous dictum that all models are wrong, but some are useful, applies. Given the current state of subject-matter knowledge, it is native to aim for the "right" model. And in the absence of the right model, many of the usual statistical concern become relatively unimportant. In particular, confidence intervals and tests no longer have much probative value. Rather, one should develop models that are descriptively informative, relatively simple, and that capture in broad-brush strokes the essential features of the empirical world at hand.

These and other considerations led naturally to consider methods by which the distance matrix could be well approximated and in a manner that eliminated much of the instability produced by taking the Ord approach at face value. Two methods seem to be especially effective. One method extracts the eigenvectors of the distance matrix and uses the first few to adjust for spatial autocorrelation. That still requires, however, that a distance matrix be specified. The second method constructs simple functions of the spatial coordinates (e.g., longitude and latitude) and uses these to adjust for spatial autocorrelation. For example, one might include longitude, latitude and their product. Analysis of real data and many simulations indicate that both methods work well work, although the second method is somewhat simpler to implement. Moreover, one can in both cases improve the approximation of the distance matrix as much as desired by using more of its eigenvectors or more complicated functions of the spatial coordinates. That is, one can make the approximations arbitrarily close to the specified distance matrix although at some point the instabilities reappear. Finally, we have developed novel algorithms for estimating multilevel linear models with spatial autocorrelation that have been implemented in our software. The Formal properties of these procedures have also been derived.

The work continues. With our new approach, we can now easily turn to multilevel generalized linear models with spatial autocorrelation. All of the pieces are now in place. It is important to emphasize again, however, that we have in important ways reformulated the manner in which the modeling

is approached; we are no longer seeking the right model but rather, a useful model.

2. INTRODUCTION

This project took at its task to develop and implement multilevel statistical models with which to address whether relationships found between a set of predictors and a response varied across research sites. Could one address directly with statistical tools how best to generalize across sites? The answer is clearly "yes" in principle. But as usual, the devil is in the details.

We built on several existing traditions in statistics. Spatial regression models (Anselin, 1988) are heteroscedastic linear models with correlated disturbances, in which the covariance between the disturbances depends on the spatial distance of the sites. Random coefficient models (Longford, 1993) are heteroscedastic linear models with correlated disturbances, in which the covariance between the disturbances depends on the predictor similarity of the sites. Multilevel models (Kreft and de Leeuw, 1998) are random coefficient models in which the predictor similarity is determined by the fact that sites are grouped into clusters. Disturbances between clusters are uncorrelated, but within clusters the covariance depends on the predictor similarity of the sites. Since distance and similarity are closely related constructs, one would expect a relationship between these two classes of models.

Spatial regression models and random coefficient models both have correlated disturbances, and the size of the correlation depends on the similarity of the sites. Similarity can be defined spatially or, more generally, in terms of similarity of the sites on a number of predictors which may not be spatial. Multilevel models simplify the overall correlation structure by assuming that sites in different clusters are uncorrelated, which means that the covariance matrix of the sites is block-diagonal, and presumably sparse.

Here, we assume *multilevel data*. In the simplest case, that of two levels, the units of level one (which we briefly call the *one-units*) are nested in units of level two (the *two-units*). We will concentrate on spatial examples, so we will often use the terminology of "sites" and "transects" for the units in our levels. Transects are nested in sites.

It should come as no surprise that much of our work relies on the earlier work of many others. But we make four new contributions as follows.

(1) We combine autoregressive models with multilevel models.
(2) We consider spatial effects both as functions of non-spatial covariates with random coefficients and as autocorrelated errors.
(3) We usefully approximate autocorrelated error structures by usung spatial regressors with random coefficients.

(4) We develop augmentation and majorization methods for generalized multilevel autoregressive models.

We also have a single, broad "take-home message." The development of statistical tools for environmental applications and the use of those tools should forego the traditional search for the "correct model" and focus instead on building one or more "useful models."

3. THE MODEL

3.1. **Basics.** In the two-level case, we have m two-units, and within two-unit j, we have n_j one-units. For each two-unit j there is a vector z_j, of length p_j, of regressors describing the two-units. This implies that there will be p_j regression coefficients, excluding the intercept, for two-units. There are also $(n_j \times q)$ matrices $\mathbf{X}_j$ of regressors describing one-units. This implies that there will be q regression coefficients, excluding the intercept, for one-units. The total number of one-units in all m two-units is n.

We usually allow for an intercept in regression models. So, we add a column of $n_j \times 1$ columns of 1's to $\mathbf{X}_j$, and a 1 as the lead element in z_j. Then, the standard two-level model[1] assumes that within each two-unit j we have a *random-coefficient regression model* [2] of the form

$$\underline{y}_{ij} = \sum_{s=0}^{q} x_{ijs}\underline{\beta}_{js} + \underline{\epsilon}_{ij}. \tag{1}$$

Here i is the index used for one-units, which are nested in the two-units. Thus $(i = 1, \cdots, n_j)$. We follow conventional practice and assume that the errors $\underline{\epsilon}_{ij}$ are uncorrelated with the predictors; there are no "omitted variables" at the one-unit level, and the functional forms are appropriate.

The $q + 1$ *random regression coefficients* $\underline{\beta}_{js}$ in equation 1 express the relationship between the *first-level predictors* and the *outcomes*. These random coefficients, of which there are $p+1$ for each two-unit j, are themselves outcomes of a second regression model shown in equation 2

$$\underline{\beta}_{js} = \sum_{r=0}^{p} z_{jr}\gamma_{rs} + \underline{\delta}_{js}, \tag{2}$$

in which the regression coefficients are outcomes predicted by *second-level predictors*. Again following convention, we assume that the errors $\underline{\delta}_{js}$ are uncorrelated with the predictors; there are no "omitted variables" at the two-unit level, and the functional forms are appropriate.

[1] Random variables are always underlined.

[2] We use element-wise notation initially, matrix notation further on.

In the spatial case, the first level predictors describe properties of the transects. They can be spatial, in the sense that they are functions of the coordinates of the transects, or non-spatial. The second level predictors describe properties of the sites, and again they can be spatial or non-spatial.

One can substitute equation 2 into equation 1 to write the model as a single equation.

$$\underline{y}_{ij} = \sum_{s=0}^{q} x_{ijs}\{\sum_{r=0}^{p} z_{jr}\gamma_{rs} + \underline{\delta}_{js}\} + \underline{\epsilon}_{ij} = \tag{3}$$

$$= \sum_{s=0}^{q}\sum_{r=0}^{p} x_{ijs}z_{jr}\gamma_{rs} + \sum_{s=1}^{q} x_{ijs}\underline{\delta}_{js} + \underline{\epsilon}_{ij} \tag{4}$$

Thus we see that the fixed part for two-unit j has the form

$$\mathbf{E}(\underline{y}_{ij}) = \sum_{r=0}^{p}\sum_{s=0}^{q}\gamma_{rs}z_{jr}x_{ijs} \tag{5}$$

with $(p+1)(q+1)$ fixed predictors, each of which is a product of a first-level and a second-level variable, often called "interaction variables," and the random part has the form

$$\underline{y}_{ij} - \mathbf{E}(\underline{y}_{ij}) = \sum_{s=0}^{q} x_{ijs}\underline{\delta}_{js} + \underline{\epsilon}_{ij} \tag{6}$$

We need some additional assumptions on the distribution of the error terms. Some very general ones are

$$\begin{aligned}
\mathbf{E}(\underline{\epsilon}_{ij}) &= 0,\\
\mathbf{E}(\underline{\delta}_{js}) &= 0,\\
\mathbf{C}(\underline{\epsilon}_{ij},\underline{\epsilon}_{k\ell}) &= 0 \text{ if } j\neq\ell,\\
\mathbf{C}(\underline{\delta}_{js},\underline{\delta}_{\ell t}) &= 0 \text{ if } j\neq\ell,\\
\mathbf{C}(\underline{\epsilon}_{ij},\underline{\delta}_{\ell s}) &= 0.
\end{aligned}$$

Thus first-level disturbances for different two-units are uncorrelated, and so are second level disturbances. The dispersion matrices of the first-level disturbances are

$$\mathbf{E}(\underline{\epsilon}_j\underline{\epsilon}_j') = \sigma_j^2\Lambda_j, \tag{7}$$

and those of the second-level disturbances are

$$\mathbf{E}(\underline{\delta}_j\underline{\delta}_j') = \sigma_j^2\Omega_j \tag{8}$$

The dispersion matrix $\sigma_j^2\Lambda_j$ allows the one-unit errors $\underline{\epsilon}_{ij}$ for a given two-unit to have different variances and to be correlated with one another. The dispersion matrix $\sigma_j^2\Omega_j$ allows the errors $\underline{\delta}_{js}$ for a given two-unit to have different variances and to be correlated with one another. The former is where spatial dependence not captured by the regressors is likely to be seen. The latter will reflect dependence between the random coefficients that is

not spatial, but a results of chance processes not captured by the two-unit model.

As a practical matter, it will be impossible to estimate the values of Λ_j and Ω_j. These matrices contain weights that determine the error variances and covariances and as such, there are far too many parameters to estimate. Often we suppose that the Ω_j are the same for all two-units, and usually the σ_j^2 are supposed to be the same too. That helps. Still, in most cases (see the examples below) the Ω_j and Λ_j are assumed to depend on a small number of parameters θ, which may again be constant over two-units.

3.2. **An Example.** A simple spatial example may help clarify the model. It's not intended to be realistic, but but to illustrate some key concepts. The one-units are observation stations, the two-units are counties. We suppose rainfall at station i in country j depends on altitude (`alt`) and distance to the ocean distance from the ocean (`dfo`).

$$\underline{\texttt{rain}}_{ij} = \underline{\beta}_{0j}\mathbf{1}_{ij} + \underline{\beta}_{1j}\texttt{alt}_{ij} + \underline{\beta}_{2j}\texttt{dfo}_{ij} + \underline{\epsilon}_{ij}, \tag{9}$$

where $\mathbf{1}_{ij}$ is the intercept, which is equal to one for all one-units. We do not assume that the regression coefficients are the same for all counties. In fact they vary according to a second regression model, for which we use indicator variables coding for the counties in the study. Thus, for $s = 0, 1, 2$,

$$\underline{\beta}_{js} = \gamma_{0s}\mathbf{1}_j + \gamma_{1s}\texttt{LA}_j + \gamma_{2s}\texttt{SB}_j + \underline{\delta}_{js}, \tag{10}$$

where again $\mathbf{1}_j$ is the intercept, now equal to one for all two-units. All observation stations in Los Angeles County (`LA`) have the same random coefficient distribution, and so have the observation stations in San Bernadino County (`SB`) and those in neither Los Angeles or San Bernadino County.

If one substitutes the equations at the county level into the equations at the station level, for $i \neq k$, and assuming for notational simplicity that σ_j^2 and Ω_j are the same for all two-units,

$$\mathbf{C}(\underline{\texttt{rain}}_{ij}, \underline{\texttt{rain}}_{kj}) = \sigma^2 \begin{bmatrix} 1 & \texttt{alt}_{ij} & \texttt{doc}_{ij} \end{bmatrix} \begin{bmatrix} \omega_{00} & \omega_{01} & \omega_{02} \\ \omega_{10} & \omega_{11} & \omega_{12} \\ \omega_{20} & \omega_{21} & \omega_{22} \end{bmatrix} \begin{bmatrix} 1 \\ \texttt{alt}_{kj} \\ \texttt{doc}_{kj} \end{bmatrix}$$

Thus, the covariance between the one-units in the same two-unit is determined by the similarity of predictor values of the one-units, where similarity is measured by their inner product in the metric Ω.

3.3. **Matrix Notation.** Define the matrix Z_j as the direct sum of q copies of the row vector z_j'. Thus it is q by qp, and it looks like

$$Z_j = \begin{bmatrix} z_j' & 0 & 0 & \cdots & 0 \\ 0 & z_j' & 0 & \cdots & 0 \\ 0 & 0 & z_j' & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & z_j' \end{bmatrix} \tag{11}$$

Using this matrix, and stacking the γ_{rs} in a single vector γ, we can rewrite (2) as

$$\underline{\beta}_j = Z_j\gamma + \underline{\delta}_j, \tag{12}$$

If we substitute (2) into (1) we find

$$\underline{y}_j = U_j\gamma + X_j\underline{\delta}_j + \underline{\epsilon}_j, \tag{13}$$

with $U_j \triangleq X_j Z_j$, and thus

$$\mathbf{E}(\underline{y}_j) = U_j\gamma, \tag{14a}$$

$$\mathbf{V}(\underline{y}_j) = \sigma_j^2(X_j\Omega_j X_j' + \Lambda_j). \tag{14b}$$

It is convenient to write Σ_j for $X_j\Omega_j X_j' + \Lambda_j$.

Now U_j is of the form

$$U_j = \begin{bmatrix} x_{j1}z_j' & | & \cdots & | & x_{jq}z_j' \end{bmatrix}, \tag{15}$$

where x_{jr} is column r of X_j. Thus, in Equation (14a), the predictors in U_j are products of a first-level predictor from X and a second-level predictor from Z. In principle, all these *cross-level interactions* are part of the model, but we can eliminate some of them by setting the corresponding element of γ equal to zero. Also observe that often the first column of both the X_j and of Z is an *intercept* column with all elements equal to $+1$. If we form all cross-level interactions, this implies that the columns of X and Z themselves also occur as predictors, because they are the intersections with the intercept at the other level.

3.4. **Generalizations.**

3.4.1. *More Than Two Levels.* In a more-than-two-level model, there are one-units, two-units, and three-units, and so on, nested within each other. For instance, we can have transects nested within streams nested within watersheds, and so on. For this case we can adopt a more general notation.

Suppose we have n_r observations on level r, and q_r predictors on that level. Thus we have $n_r \times (q_r + 1)$ matrices $X^{(r)}$ with predictors. We also

use indicator matrices $G^{(r)}$, which are $n_r \times n_{r+1}$, and which indicate how the r-units map into the $(r+1)$-units.

The first two equations defining our multilevel model are

$$\underline{y}^{(1)}_{i_1} = \sum_{s_1=0}^{q_1} x^{(1)}_{i_1 s_1} \sum_{i_2=1}^{n_2} g^{(1)}_{i_1 i_2} \underline{y}^{(2)}_{i_2 s_1} + \underline{\epsilon}^{(1)}_{i_1}, \tag{16a}$$

$$\underline{y}^{(2)}_{i_2 s_1} = \sum_{s_2=0}^{q_2} x^{(2)}_{i_2 s_2} \sum_{i_3=1}^{n_3} g^{(2)}_{i_2 i_3} \underline{y}^{(3)}_{i_3 s_1 s_2} + \underline{\epsilon}^{(s)}_{i_2 s_1}. \tag{16b}$$

Thus we see we have n_1 random variables in $\underline{y}^{(1)}$. These are the observed outcomes. We have $n_2 \times q_1$ unobserved random variables in $\underline{y}^{(2)}$, these are the random regression coefficients from our previous formulation. Then we have $n_3 \times q_1 \times q_2$ unobserved random coefficients in $\underline{y}^{(3)}$, and so on.

In the same way as before we can combine equations to form single equations, which of course rapidly become unwieldy. From (16) we find, for example,

$$\underline{y}^{(1)}_{i_1} = \sum_{s_1=0}^{q_1} x^{(1)}_{i_1 s_1} \sum_{i_2=1}^{n_2} g^{(1)}_{i_1 i_2} [\sum_{s_2=0}^{q_2} x^{(2)}_{i_2 s_1 s_2} \sum_{i_3=1}^{n_3} g^{(2)}_{i_2 i_3} \underline{y}^{(3)}_{i_3 s_1 s_2} + \underline{\epsilon}^{(s)}_{i_2 s_1}] + \underline{\epsilon}^{(1)}_{i_1} \tag{17}$$

3.4.2. *Multivariate Outcomes.* If there is more than one outcome variable, we can use a simple trick to force the model into the multilevel framework. We use *variables* as an additional level, in fact as the first level. Thus variables are nested in transects, transects in sites, and so on. Having multiple outcomes just adds a level to the hierarchy, and it does not really complicate modelling in any essential way. It is also clear that missing data can be incorporated without problems in this way, because this simply means that some transects have fewer units (i.e. variables) than others.

3.4.3. *Non-independent Two-Units.* In our models we usually assume that Ω_j are the same for all sites. If we make this assumption, it is also possible to use a simple model for correlated sites, which has

$$\mathbf{C}(\underline{y}_j, \underline{y}_\ell) = \sigma_{jl}(X_j \Omega X_\ell' + \Lambda_j^{1/2} \Lambda_\ell^{1/2}) \tag{18a}$$

for all $j \neq \ell$, and

$$\mathbf{C}(\underline{y}_j, \underline{y}_j) = \sigma_{jj}(X_j \Omega X_j' + \Lambda_j) \tag{18b}$$

for all j, where the $\sigma_{j\ell}$ are the covariances between sites.

3.4.4. *Generalized MAR Models.* In the same way as linear models are generalized to generalized linear models, we can try to construct generalized mixed linear models from mixed linear models. The trick is simply to condition on the random effects. In generalized linear models first-level observations are independent given the random effects, and thus the conditional distribution is a simply product of univariate Poisson, binomials, or gammas. But in generalized mixed linear models with autocorrelated or spatially correlated first-level errors, we no longer can use independence, and we have to assume that the errors within sites have multivariate Poisson, binomial, or gamma distributions. There is no agreement in statistics how to define such multivariate distributions, and the definitions that are in use do not have many of the simplifying properties of ther univariate versions.

We shall see below, however, that models with correlated first-level errors can be approximated by models with additional random effects and uncorrelated first-level errors. In these approximations conditioning on the random effects makes the observations independent again, and the results developed for generalized mixed linear models apply again.

4. MODELS FOR ERROR DISPERSIONS

The dispersion matrices Λ_j of first-level disturbances can take many different forms. Generally, they are a function of a number of unknown parameters, collected in a vector ρ. Estimation simplifies considerably if the Λ_j are known, and in particular in the homoscedastic case with uncorrelated errors in which $\Lambda_j = I_j$, the identity matrix of order n_j. But in spatial situations the assumption that the errors are uncorrelated often is difficult to defend.

This is why a great deal of attention has been paid to modeling the dependence of spatial observations, taking as the main inspiration the literature on time series models. The key paper in spatial autoregressive (SA) modeling is Ord [1975]. Also compare Griffith [2002b] and Anselin [2001]. There are various forms of these SA models, but the most important ones are one-parameter models, in which the single parameter ρ is interpreted as spatial autocorrelation. It indicates the strength of the spatial effects.

In multilevel models we also often have restrictions on the Ω_j, for instance that they are equal, that specific elements are zero, and so on. We shall discuss these restrictions elsewhere, and concentrate on first-level disturbances in this section.

4.1. **The Spatial Lag Model.** This is also known as the AR, or autoregressive reponse model. It specifies

$$\underline{y}_j = \rho_j W_j \underline{y}_j + X_j \underline{\beta}_j + \underline{\epsilon}_j, \tag{19}$$

where $\underline{\epsilon}_j$ is homoscedastic with variance σ_j^2. Clearly in this AR model

$$\mathbf{E}(\underline{y}_j \mid \beta_j) = (I_j - \rho_j W_j)^{-1} X_j \beta_j, \tag{20a}$$

$$\mathbf{V}(\underline{y}_j \mid \beta_j) = \sigma_j^2 [(I_j - \rho_j W_j)(I_j - \rho_j W_j')]^{-1}. \tag{20b}$$

In this model the autoregression is defined directly in terms of the outcomes. The spatial dependence is built into the model in a structural manner. That is, the data analyst will typically have a subject-matter rationale for why and how values of the outcome variable are related. For example, if air quality is the outcome of interest, there may be diffusion of air pollution from any one location to locations near by. Depending on the value of ρ_j, the diffusion affects might be large or small, or perhaps even be negative. Note also that to isolate their role the predictors, adjustments have to be made for the diffusion process, which links the outcome across locations. A failure to make such adjustments may mean that effects attributed to one or more of the predictors are really just a result of the movement of air pollution from one place to another.

4.2. **The Spatial Error Model.** This model is also known as the SAR or simultaneous autoregressive model. It has

$$\underline{y}_j = X_j \underline{\beta}_j + \underline{\epsilon}_j, \tag{21a}$$

and it assumes an autoregression structure for the errors terms. Thus

$$\underline{\epsilon}_j = \rho_j W_j \underline{\epsilon}_j + \underline{\zeta}_j, \tag{21b}$$

where the $\underline{\zeta}_j$ are homoscedastic with variance σ_j^2.

This leads to

$$\mathbf{E}(\underline{y}_j \mid \beta_j) = X_j \beta_j, \tag{22a}$$

$$\mathbf{V}(\underline{y}_j \mid \beta_j) = \sigma_j^2 [(I_j - \rho_j W_j)(I_j - \rho_j W_j')]^{-1}. \tag{22b}$$

This formulation implies that the spatial dependence is not potentially confounded with the predictors. It derives solely from dependence among the errors themselves. Errors that are more proximate in space, for instance, may tend to be more alike that errors that are farther apart. The reasons for the dependence are usually not of much interest. As such, the dependence is a mere nuisance and/or beyond current subject matter interest. The goal is to "mop up" the spatial dependence in the errors so that it does not affect the precision of estimates of the β_j or estimates of their standard errors.

4.3. **The Conditional Autoregression Model.** Under this CAR model, we let

$$\underline{y}_j = X_j\underline{\beta}_j + (I_j - \rho_j W_j)^{-1/2}\underline{\epsilon}_j, \tag{23}$$

where W_j is now a symmetric weight matrix, and where $\underline{\epsilon}_j$ is homoscedastic with variance σ_j^2. This implies

$$\mathbf{E}(\underline{y}_j \mid \beta_j) = X_j\beta_j, \tag{24a}$$

$$\mathbf{V}(\underline{y}_j \mid \beta_j) = \sigma_j^2(I_j - \rho_j W_j)^{-1} \tag{24b}$$

If dependence in the errors can be treated as a mere nuisance, the model one uses for the errors is of little importance as long as the dependence is taken into account when the regression coefficients are estimated. In this context, the conditional autoregressive model can be seen as an alternative to the spatial error model, and it has some of the same look and feel. Larger values of ρ_j imply more dependence among the errors. And just as for the spatial error model, the dependence may be a function of distance; closer errors may tend to be more alike. The main advantage of the conditional autoregressive model will be more apparent later. But suffice it to say that it can be as effective in mopping up dependence in the errors as the spatial errors model, but will be far easier to compute.

4.4. **Weight Matrices.** How to choose the W_j has been discussed many times in the geostatistics literature. A good review is Bavaud [1998], see also Cressie [1991]. Although it is possible to give some general indications, choosing a precise and appropriate W_j is difficult, probably even more difficult than choosing a correct set of predictors. The usual problem is that there is too little a priori knowledge to inform the choice and at best some general clues in the data.

4.4.1. *Choice of Weights.* For W_j we assume, in spatial situations, that its elements are similarities of transects in site j. The more similar (the closer) the transects, the larger the corresponding element in W_j. If we do not have a good reason to choose a specific W_j, we can make it some (decreasing) function of the transect distances, but again choosing the function is often disturbingly arbitrary. In many cases, moreover, we even want to replace simple Euclidean distance by other distances (measured along a network or stream, for instance), which take the actual spatial setting into account. Throughout, we suppose the elements of W_j are non-negative.

4.4.2. *Large matrices.* In spatial analysis we often encounter situations in which the order of the W_j is very large, maybe 10^5 or 10^6. Obviously in

such cases, it will generally not be possible to store floating-point matrices of this size, let alone compute their determinants, inverse, or eigendecomposition.

There are several ways around this problem. The first is to use patterned weight matrices of zeroes and ones (coding adjacency or nearest neighbor, for instance), with a determinant or an inverse available in analytical form [Pace and Zou, 2000]. The second is to use sparse matrix techniques for weight matrices with a very large proportions of zeroes [Pace and Barry, 1997a,b,c] (again, adjacency matrices come to mind). We have also seen that multilevel analysis suggests partitioning transects or sites into clusters, and making the between cluster covariance equal to zero. This also introduces a great deal of sparseness. And finally, fast numerical approximations to the loss function are also a possibility. Specifically, techniques for approximating the determinant in the normal log-likelihood for all AR, SAR, and CAR models are in Smirnov and Anselin [2001] and Griffith [2002a].

In the models discussed in this paper, we have the additional complication that the dispersion matrix is made up out of two components: a part based on similarity of the regressors and a part based on spatial information, coded in the weight matrices. This makes patterned weight matrix and sparse matrix techniques more difficult to use, and we have to resort to other types of approximations.

4.4.3. *Normalizing the Weights.* It is computationally convenient if the weight matrices in the SAR and AR models are symmetric. In that case,

$$(I_j - \rho_j W_j)(I_j - \rho_j W_j') = (I_j - \rho_j W_j)^2$$

. This usually simplifies some approximations (see below). Unfortunately in many applications an asymmetric set of weights may make more sense (think of the influence of stream flow or hillside slope on ecological distance, for instance).

Let us indicate briefly why having symmetric matrices is convenient. If the W_j are known symmetric matrices. We can compute the spectral decomposition $W_j = K_j \Phi_j K_j'$, and we find

$$\Lambda_j(\rho_j) = \sum_s \frac{1}{(1 - \rho_j \phi_{js})^2} k_{js} k_{js}' \tag{25a}$$

for SAR and

$$\Lambda_j(\rho_j) = \sum_s \frac{1}{1 - \rho_j \phi_{js}} k_{js} k_{js}' \tag{25b}$$

for CAR. Thus, the eigenvectors of $\Lambda_j(\rho_j)$ are the same as those of W_j, and the eigenvalues are simple functions of the eigenvalues of W_j. If we change ρ_j, only the eigenvalues change, the eigenvectors remain the same.

For interpretation purposes, we often normalize the weights in such a way that the rows of W_j sum to unity. This makes the weight matrix stochastic, and by Frobenius theorem this implies that the largest eigenvalue of W_j is equal to $+1$. This means that the smallest eigenvalue of $I_j - \rho_j W_j$ is $1 - \rho_j$, and thus $I_j - \rho_j W_j$ is positive definite as long as $\rho_j < 1$, which helps in the interpretation of ρ as a type of auto-correlation coefficient.

In some cases, we want W_j to be both symmetric and normalized (i.e. doubly stochastic). This is discussed for CAR models in Page and LeSage [2002]. In our code section we give an algorithm to normalize non-negative symmetric matrices in such a way that they become doubly stochastic.

4.5. **Special Case: Time Series Models.** If the outcomes are one-dimensional (for instance if transects are arranged in lines), then it makes sense to use a time series model for the first-level errors [Hedeker, 1989, Hedeker and Gibbons, 1996]. We discuss these models here briefly because they show where the SA models come from, and because they are more familiar to most statisticians.

A first obvious choice for a time-series model is the *random walk*, which has

$$\underline{\epsilon}_j = W_j \underline{\epsilon}_j + \underline{\zeta}_j, \tag{26}$$

where W_j has all elements equal to zero, except the ones immediately below the main diagonal, which are one. It follows that

$$\underline{\epsilon}_j = T_j \underline{\zeta}_j, \tag{27}$$

where T_j has all elements on and below the main diagonal equal to one and all elements above the main diagonal equal to zero. Thus

$$\Lambda_j = T_j T_j', \tag{28}$$

which means that element (s, t) is equal to $min(s, t)$.

In an AR(p) process we have

$$\underline{\epsilon}_j = W_j \underline{\epsilon}_j + \underline{\zeta}_j, \tag{29}$$

where W_j has a band of width p below the diagonal and zeroes elsewhere. Thus there are p parameters, the autoregression coefficients, in W_j. AR(1) is thus very much like the random walk, except that the element below the diagonal is the single parameter ρ_j.

A MA(q) process also uses a banded matrix with parameter values, but we now have

$$\underline{\epsilon}_j = W_j \underline{\zeta}_j, \tag{30}$$

where W_j has diagonal one, and a band of width q in each row below the diagonal. Thus MA(1) has diagonal one, and ρ_j below the diagonal.

We can easily extend this to ARMA(p,q) and even more complicated processes, but this is comparatively straightforward and it may be overkill in many situations. For our purposes the most interesting models are AR(1) and MA(1), which can be defined in term of the backshift matrix B_j, which has elements equal to one below the diagonal only. Then for AR(1) we have

$$\Lambda_j(\rho_j) = (I_j - \rho_j B_j)^{-1}(I_j - \rho_j B_j')^{-1}, \tag{31a}$$

and for MA(1) we have

$$\Lambda_j(\rho_j) = (I_j + \rho_j B_j)(I_j + \rho_j B_j'). \tag{31b}$$

The random walk is AR(1) with $\rho_j = 1$.

5. Model Approximation

In this section, we discuss two ways to approximate the various AR models. Because the concept of "the true model" is at least obscure and because even if we know how to think about "the true model" we usually do not have very precise information about which W_j produces it, it makes sense to employ an approximation the dispersion matrix that is computationally convenient. We first simplify the model by an approximation that works well for small ρ_j, and then we approximate the model by another model with homoscedastic first-level errors, i.e. a model with $\Lambda_j = I_j$.

5.1. **Simplified AR.** Consider again the SAR model described in Section 4.2. Recall that the variance-covariance matrix of the errors was $\sigma_j^2[(I_j - \rho_j W_j)(I_j - \rho_j W_j')]^{-1}$, where all of the terms to the right of σ_j^2 represent $\Lambda_j(\theta)$. In the *Simplified AR Model* (SIMAR), we assume

$$\Lambda_j(\theta) = I_j + \rho_j W_j, \tag{32}$$

where the off-diagonal elements of the symmetric matrix W_j are again some decreasing function of the Euclidean distances between the transects, or, more generally, of the spatial dissimilarities.

In the CAR model, if ρ_j is small, we have

$$(I_j - \rho_j W_j)^{-1} = I_j + \rho_j W_j + o(\rho_j), \tag{33}$$

and in the SAR and CAR models,

$$\Lambda_j(\rho_j) = (I_j - \rho_j W_j)^{-1}(I_j - \rho W_j')^{-1} = I_j + \rho_j(W_j + W_j') + o(\rho_j), \tag{34}$$

which are both of the SIMAR form.

For both AR(1) and MA(1), and small ρ_j,

$$\Lambda_j = I_j + \rho_j(B_j + B_j') + o(\rho_j), \tag{35}$$

which is again of the required SIMAR form.

5.2. **Spatial Effects as Random Coefficients.** By using random coefficients in appropriate ways we can emulate the covariance structure of the SIMAR without assuming correlated errors for the first-level units. Thus we can maintain $\Lambda_j = I_j$. The trick is really quite simple. We know that in our spatial multilevel models

$$\underline{y}_j = U_j\gamma + X_j\underline{\delta}_j + \underline{\epsilon}_j, \tag{36}$$

where

$$\mathbf{V}(\underline{\epsilon}_j) = \sigma_j^2(I_j + \rho_j W_j). \tag{37}$$

Now suppose $W_j = K_j\Phi_j K_j'$ is the spectral decomposition of W_j. Then we can write

$$\underline{y}_j = U_j\gamma + X_j\underline{\delta}_j + K_j\underline{\eta}_j + \underline{\zeta}_j, \tag{38}$$

where $\underline{\delta}_j$ and $\underline{\eta}_j$ are uncorrelated, and where

$$\mathbf{V}(\underline{\eta}_j) = \sigma_j^2\rho_j\Phi_j, \tag{39a}$$

$$\mathbf{V}(\underline{\zeta}_j) = \sigma_j^2 I_j. \tag{39b}$$

But (38) and (39) can be interpreted as a simple multilevel model in which the covariance matrix of the random effects is of the form

$$\begin{bmatrix}\Omega_j & 0\\ 0 & \rho_j\Phi_j\end{bmatrix}. \tag{40}$$

First-level errors are homoscedastic, and the regression coefficients corresponding with the eigenvector-predictors K_j only have a random part and a vanishing fixed part. Moreover the random parts are uncorrelated, with a diagonal dispersion matrix proportional to the eigenvalues of W_j. In short, one can write the SIMAR model as a multilevel model with restrictions on the covariance matrix of the random effects.

5.3. **Positive definite variances.** One problem with this formulation is that it is not guaranteed that the eigenvalues Φ_j of W_j are non-negative. If there are negative eigenvalues, then Equation (39a) becomes hard to interprete.

We can use the fact, however,that $I_j + \rho_j W_j$ must be positive definite. Suppose $\rho_j > 0$, and write ψ_j for the smallest eigenvalue of W_j. Then

$$I_j + \rho_j W_j = (1 + \rho_j\psi_j)I_j + \rho_j K_j(\Phi_j - \psi_j I_j)K_j' \tag{41}$$

and we can rewrite (39) as

$$\mathbf{V}(\underline{\eta}_j) = \sigma_j^2\rho_j(\Phi_j - \psi_j I_j), \tag{42a}$$

$$\mathbf{V}(\underline{\zeta}_j) = \sigma_j^2(1 + \rho_j\psi_j)I_j. \tag{42b}$$

These are somewhat more complicated restrictions, but they always give positive semidefinite dispersion matrices.

5.4. **Using Fewer Eigenvalues.** A second problem with our approximation is that we replace working with a very large spatial error covariance matrix with working with a very large number of random effects. The number of random effects we add is equal to the order of the spatial covariance matrix.

We attack this problem by using only a small number of eigenvectors of W_j, those corresponding with the largest eigenvalues (in modulus). Thus we use a principal component type approximation to the random effects, In the case of spatial information in W_j, using some function of the distances, we can expect that two or three principal components to give a rather good approximation.

5.5. **General Approach.** Instead of approximating the SA models by SIMAR, and then approximating SIMAR by using eigenvectors, we can also follow a more straightforward approach. Consider the following multilevel model for site j

$$\underline{y}_j = X_j\underline{\beta}_j + Z_j\underline{\eta}_j + \underline{\epsilon}_j, \tag{43}$$

where X_j contains regression coordinates and Z_j contains (functions of the) spatial coordinates. For our second level model we use

$$\underline{\beta}_j = A_j\gamma + \underline{\delta}_j, \tag{44a}$$

$$\underline{\eta}_j = B_j\kappa + \underline{\xi}_j. \tag{44b}$$

This implies

$$\underline{y}_j = X_jA_j\gamma + Z_jB_j\kappa + \underline{\nu}_j, \tag{45a}$$

where

$$\underline{\nu}_j = X_j\underline{\delta}_j + Z_j\underline{\xi}_j + \underline{\epsilon}_j, \tag{45b}$$

and thus, with suitable uncorrelatedness assumptions,

$$\mathbf{E}(\underline{y}_j) = X_jA_j\gamma + Z_jB_j\kappa, \tag{46a}$$

$$\mathbf{V}(\underline{y}_j) = \sigma_j^2(X_j\Omega_jX_j' + Z_j\Theta_jZ_j' + I_j). \tag{46b}$$

This becomes an approximate multilevel Ord model if we let $B_j = 0$, i.e. the spatial regression coefficients do not have a fixed part, and we let $\Theta_j = \rho_j^2 I_j$, i.e. the spatial regression coefficients are uncorrelated. Then we get

$$\mathbf{E}(\underline{y}_j) = X_jA_j\gamma, \tag{47a}$$

$$\mathbf{V}(\underline{y}_j) = \sigma_j^2[X_j\Omega_jX_j' + (I_j + \rho_j^2Z_jZ_j')]. \tag{47b}$$

Moreover, if we want to get closer to SA, we can choose Z_j in a clever way, using the results be discussed earlier in this section. If the W_j matrix in the Ord model is a function of the spatial distances, then it obviously is a function of the coordinates, and thus all its eigenvectors are functions of the coordinates. If we choose Z_j as a low-rank (principal component) approximation of W_j, using the eigenvectors, then we can get very close to the Ord model.

6. NORMAL LIKELIHOOD

We do not assume, here or anywhere else, that our data are sampled from a normal distribution. But we do use the normal likelihood to measure the distance between observed and fitted expected values and dispersions [de Leeuw and Kreft, 1986].

6.1. **Log-likelihood.** The normal negative log-likelihood is (except for irrelevant constants)

$$\mathcal{L}(\sigma_j^2,\Omega_j,\Lambda_j,\gamma)=\sum_{j=1}^m n_j\log\sigma_j^2+\sum_{j=1}^m\log\mathbf{det}(\Sigma_j)+\sum_{j=1}^m\frac{(y_j-U_j\gamma)'\Sigma_j^{-1}(y_j-U_j\gamma)}{\sigma_j^2}. \tag{48}$$

We can use the result on partitioned determinants from Appendix A to simplify the log-likelihood, i.e. rewrite it in such a way that it involves less computation and smaller matrices. This gives

$$\begin{aligned}\log\mathbf{det}(\Sigma_j)&=\log\mathbf{det}(X_j\Omega_jX_j'+\Lambda_j)=\\&\log\mathbf{det}(\Lambda_j)+\log\mathbf{det}(\Omega_j)+\log\mathbf{det}(\Omega_j^{-1}+X_j'\Lambda_j^{-1}X_j)=\\&\log\mathbf{det}(\Lambda_j)+\log\mathbf{det}(X_j'\Lambda_j^{-1}X_j)+\log\mathbf{det}(\Omega_j+(X_j'\Lambda_j^{-1}X_j)^{-1})\end{aligned} \tag{49}$$

6.2. **Standard Errors.** Assume all σ_j^2 are the same. Then

$$\hat\gamma=\left(\sum_{j=1}^m U_j'\hat\Sigma_j^{-1}U_j\right)^{-1}\sum_{j=1}^m U_j'\hat\Sigma_j^{-1}y_j, \tag{50}$$

and thus

$$\hat V(\gamma)=\left(\sum_{j=1}^m U_j'\hat\Sigma_j^{-1}U_j\right)^{-1}. \tag{51}$$

7. AUGMENTATION

7.1. **General Idea.** An *augmentation algorithm* to minimize a function $f(x)$ over $x \in X$ constructs an *augmentation function* $g(x,y)$ on $X \otimes Y$, such that

$$\min_{y\in Y} g(x,y) = f(x) \tag{52}$$

for all $x \in X$. We now minimize the function $g(x,y)$ by *block relaxation*, i.e. we start with an initial $x_0 \in X$. We then find

$$y_0 = \underset{y\in Y}{argmin}\, g(x_0,y), \tag{53a}$$

$$x_1 = \underset{x\in X}{argmin}\, g(x,y_0), \tag{53b}$$

$$y_1 = \underset{y\in Y}{argmin}\, g(x_0,y), \tag{53c}$$

and so on, until convergence.

7.2. **Key Result.** We define an augmentation function by introducing the additional variables $\tilde{\mu}_j$ and $\tilde{\Sigma}_j$.

$$\begin{aligned}\mathcal{F}(\sigma_j^2,\Omega_j,\Lambda_j,\gamma,\tilde{\Sigma}_j,\tilde{\mu}_j) &\triangleq \\ \sum_{j=1}^m \left[n_j\log\sigma_j^2 + \log\mathbf{det}(\tilde{\Sigma}_j) + \mathbf{tr}\,\tilde{\Sigma}_j^{-1}(X_j\Omega_jX_j'+\Lambda_j-\tilde{\Sigma}_j)\right] + \\ \sum_{j=1}^m \frac{(y_j-U_j\gamma-X_j\tilde{\mu}_j)'\Lambda_j^{-1}(y_j-U_j\gamma-X_j\tilde{\mu}_j)+\tilde{\mu}_j'\Omega_j^{-1}\tilde{\mu}_j}{\sigma_j^2}\end{aligned} \tag{54}$$

To show that we have a proper augmentation, we need two lemma's.

Lemma 1.

$$\begin{aligned}(y_j-U_j\gamma)'[X_j\Omega_jX_j'+\Lambda_j]^{-1}(y_j-U_j\gamma) = \\ \min_\mu[(y_j-U_j\gamma-X_j\mu)'\Lambda_j^{-1}(y_j-U_j\gamma-X_j\mu)+\mu'\Omega_j^{-1}\mu].\end{aligned}$$

and the minimum is attained for

$$\begin{aligned}\hat{\mu} = [X_j'\Lambda_j^{-1}X_j+\Omega_j^{-1}]^{-1}X_j'\Lambda_j^{-1}(y_j-U_j\gamma) = \\ \Omega_jX_j'[X_j\Omega_jX_j'+\Lambda_j]^{-1}(y_j-U_j\gamma)\end{aligned}$$

Proof. The first expression for $\hat{\mu}$ is obvious. By Sherman-Morrison-Woodbury (Appendix A)

$$[X_j'\Lambda_j^{-1}X_j+\Omega_j^{-1}]^{-1} = \Omega_j - \Omega_jX_j'(\Lambda_j+X_j\Omega_jX_j')^{-1}X_j\Omega_j. \tag{55}$$

If we substitute this in the first expression for $\hat{\mu}$ and simplify, we find the second expression (see Corollary 6 in Appendix A). Now substitute the second expression into the loss function, and we find the final result. □

Lemma 2.

$$\log \mathbf{det}(X_j\Omega_j X_j' + \Lambda_j) = \min_{\Sigma>0} \log \mathbf{det}(\Sigma) + \mathbf{tr}\ \Sigma^{-1}(X_j\Omega_j X_j' + \Lambda_j) - p,$$

and the minimum is attained for $\hat{H} = X_j\Omega X_j' + \Lambda_j$.

Proof. ¿From Appendix D. □

We combine the two lemma's in a theorem.

Theorem 3.

$$\mathcal{L}(\sigma_j^2, \Omega_j, \Lambda_j, \gamma) = \min_{\tilde{\mu}_j} \min_{\tilde{\Sigma}_j} \mathcal{F}(\sigma_j^2, \Omega_j, \Lambda_j, \gamma, \tilde{\Sigma}_j, \tilde{\mu}_j)$$

Proof. From Lemma 1 and Lemma 2. □

7.3. **Augmentation Algorithm.** The theorems in the previous section imply that finding maximum likelihood estimates of the parameters can be done by minimizing $\mathcal{F}$ over all its parameters. We minimize $\mathcal{F}$ by *block relaxation*, i.e. there are six sets of parameters, and we cycle through them, minimizing over each set while keeping the other five fixed at their current values. The minimization gives new values for the active subset of the parameters, and we proceed to the next subset.

Step 1: $\tilde{\Sigma}_j$**:** We already know how to solve for $\tilde{\Sigma}_j$. This is just

$$\tilde{\Sigma}_j = X_j\Omega_j X_j' + \Lambda_j. \tag{56}$$

Step 2: $\tilde{\mu}_j$**:** For $\tilde{\mu}_j$ we find

$$\tilde{\mu}_j = [X_j'\Lambda_j^{-1}X_j + \Omega_j^{-1}]^{-1}X_j'\Lambda_j^{-1}[y_j - U_j\gamma] = \Omega_j X_j'\tilde{\Sigma}_j^{-1}(y_j - U_j\gamma) \tag{57}$$

Step 3: σ_j^2**:** Solving for σ_j^2 is easy. Let $r_j \triangleq y_j - U_j\gamma - X_j\tilde{\mu}_j$. Then

$$\sigma_j^2 = \frac{1}{n_j}\{r_j'\Lambda_j^{-1}r_j + \tilde{\mu}_j'\Omega_j^{-1}\tilde{\mu}_j\}, \tag{58}$$

and if all σ_j^2 are required to be the same

$$\sigma^2 = \frac{1}{n}\sum_{j=1}^m\{r_j'\Lambda_j^{-1}r_j + \tilde{\mu}_j'\Omega_j^{-1}\tilde{\mu}_j\}. \tag{59}$$

Step 4: γ: By weighted least squares,

$$\gamma = (\sum_{j=1}^{m} U_j' \Lambda_j^{-1} U_j)^{-1} \sum_{j=1}^{m} U_j' \Lambda_j^{-1} (y_j - X_j \tilde{\mu}_j). \tag{60}$$

Step 5: Ω_j: Define

$$A_j \triangleq X_j' \tilde{\Sigma}_j^{-1} X_j, \tag{61}$$

$$B_j \triangleq \frac{1}{\sigma_j^2} \tilde{\mu}_j \tilde{\mu}_j'. \tag{62}$$

Then

$$\frac{\partial \mathcal{F}}{\partial \Omega_j} = A_j - \Omega_j^{-1} B_j \Omega_j^{-1}. \tag{63}$$

If all Ω_j are required to be the same, we find

$$\frac{\partial \mathcal{F}}{\partial \Omega} = A - \Omega^{-1} B \Omega^{-1}. \tag{64}$$

Setting the partials to zero gives an equation is of the form discussed in Appendix C. The solution for all Ω the same is

$$\Omega = B^{1/2} (B^{1/2} A B^{1/2})^{-1/2} B^{1/2}. \tag{65}$$

Step 6: Λ_j: For Λ_j we find similarly

$$\frac{\partial \mathcal{F}}{\partial \Lambda_j} = \tilde{\Sigma}_j^{-1} - \Lambda_j^{-1} C_j \Lambda_j^{-1}, \tag{66a}$$

where

$$C_j \triangleq \frac{1}{\sigma_j^2} r_j r_j'. \tag{66b}$$

In this case certainly the Λ_j will have to be restricted to some parametric form.

7.4. **Restrictions on the variance parameters.**

7.4.1. Ω. In multilevel analysis we often have restrictions of Ω, however, in which case this solution does not apply any more. If there are restrictions, we may have to use a numerical optimization method.

Diagonal: If we require Ω to be diagonal, then the solution for diagonal element ω_{ss} is simply

$$\omega_{ss} = \sqrt{\frac{b_{ss}}{a_{ss}}} \tag{67}$$

Almost Diagonal: Alternatively, we may be in a situation where we require $\Omega = K\Phi K'$, with K orthonormal and known and Φ unknown. Then

$$\phi_{ss} = \sqrt{\frac{\{K'BK\}_{ss}}{\{K'AK\}_{ss}}} \tag{68}$$

Linear: If the model is of the form $\Omega = \sum \omega_s T_s$, then

$$\frac{\partial \mathcal{F}}{\partial \omega_s} = \mathbf{tr}\ X_j' H_j^{-1} X_j T_s - \frac{1}{\sigma^2}\mathbf{tr}\ V'\Omega^{-1}T_s\Omega^{-1}V, \tag{69a}$$

with V the $q \times m$ matrix with the v_j as columns. Also

$$\frac{\partial^2 \mathcal{F}}{\partial \omega_s \partial \omega_t} = \frac{2}{\sigma^2}\mathbf{tr}\ V'\Omega^{-1}T_s\Omega^{-1}T_t\Omega^{-1}V. \tag{69b}$$

Simultaneously Diagonalizable: If there is an orthonormal K and diagonal Φ_s such that $T_s = K\Phi_s K'$

7.4.2. Λ. In the spatial case, discussed above, the Λ_j depend on a single parameter θ, which we could find by some univariate minimization method. For ease of reference we compute the derivatives with respect to θ. Clearly

$$\frac{\partial \mathcal{F}}{\partial \theta} = \mathbf{tr}\ H_j^{-1}\frac{\partial \Lambda_j}{\partial \theta} - \frac{1}{\sigma^2} r_j' \Lambda_j^{-1}\frac{\partial \Lambda_j}{\partial \theta}\Lambda_j^{-1} r_j, \tag{70}$$

and

$$\frac{\partial^2 \mathcal{F}}{\partial \theta^2} = \mathbf{tr}\ H_j^{-1}\frac{\partial^2 \Lambda_j}{\partial \theta^2} - \frac{1}{\sigma^2} r_j' \Lambda_j^{-1}\frac{\partial^2 \Lambda_j}{\partial \theta^2}\Lambda_j^{-1} r_j + 2\frac{1}{\sigma^2} r_j' \Lambda_j^{-1}\frac{\partial \Lambda_j}{\partial \theta}\Lambda_j^{-1}\frac{\partial \Lambda_j}{\partial \theta}\Lambda_j^{-1} r_j, \tag{71}$$

In the Ord model, where $\Lambda_j = (I - \theta W_j)^{-1}(I - \theta W_j')^{-1}$, then we can perhaps most easily minimize $\mathcal{F}$ by grid search.

The derivatives are also fairly easy to compute. Let $P(\epsilon) = I - (\theta + \epsilon)W$ and wite P for $P(0)$. Then

$$P(\epsilon)^{-1} = P^{-1} + \epsilon P^{-1}WP^{-1} + \epsilon^2 P^{-1}WP^{-1}WP^{-1} + o(\epsilon^2) \tag{72a}$$

and

$$P(\epsilon)^{-T} = P^{-T} + \epsilon P^{-T}W'P^{-T} + \epsilon^2 P^{-T}W'P^{-T}W'P^{-T} + o(\epsilon^2), \tag{72b}$$

where P^{-T} is short for $(P^{-1})' = (P')^{-1}$. Now

$$\Lambda(\epsilon) = P(\epsilon)^{-1}P(\epsilon)^{-T} = \Lambda + \epsilon(\Lambda W'P^{-T} + P^{-1}W\Lambda) + \epsilon^2(\Lambda W'P^{-T}W'P^{-T} + P^{-1}WP^{-1}W\Lambda + P^{-1}W\Lambda W'P^{-T}) + o(\epsilon^2) \tag{73}$$

Thus

$$\frac{\partial \Lambda_j}{\partial \theta} = \Lambda_j W_j' P_j^{-T} + P_j^{-1} W_j \Lambda_j, \tag{74a}$$

$$\frac{\partial^2 \Lambda}{\partial \theta^2} = \Lambda_j W_j' P_j^{-T} W_j' P_j^{-T} + P_j^{-1} W_J P_j^{-1} W_j \Lambda_j + P_j^{-1} W_j \Lambda_j W_j' P_j^{-T}. \tag{74b}$$

7.5. **Another Augmentation Algorithm.** Instead of Lemma 2 we can also use

Lemma 4.

$$\log \mathbf{det}(X_j \Omega_j X_j' + \Lambda_j) = \log \mathbf{det}(\Lambda_j) + \log \mathbf{det}(\Omega_j) + \min_{\Psi > 0} \log \mathbf{det}(\Psi) + \mathbf{tr} \Psi^{-1}(\Omega_j^{-1} + X_j' \Lambda_j^{-1} X_j - \Psi),$$

and the minimum is attained for $\hat{\Psi} = \Omega_j^{-1} + X_j' \Lambda_j^{-1} X_j$.

Proof. Same as before. □

This lemma leads to the augmentation function

$$\begin{aligned} \mathcal{G}(\sigma_j^2, \Omega_j, \Lambda_j, \gamma, \tilde{\Psi}_j, \tilde{\mu}_j) \triangleq & \\ + \sum_{j=1}^m \Big[n_j \log \sigma_j^2 + \log \mathbf{det}(\Lambda_j) + \log \mathbf{det}(\Omega_j) + \log \mathbf{det}(\tilde{\Psi}_j) + & \\ \mathbf{tr} \tilde{\Psi}_j^{-1}(\Omega_j^{-1} + X_j' \Lambda_j^{-1} X_j - \tilde{\Psi}_j) \Big] + & \\ + \sum_{j=1}^m \frac{(y_j - U_j \gamma - X_j \tilde{\mu}_j)' \Lambda_j^{-1} (y_j - U_j \gamma - X_j \tilde{\mu}_j) + \tilde{\mu}_j' \Omega_j^{-1} \tilde{\mu}_j}{\sigma_j^2}. & \end{aligned} \tag{75}$$

and to an algorithm in which the updates for $\sigma_j^2, \tilde{\mu}_j$, and γ are the same as before, but in which

$$\tilde{\Psi}_j = \Omega_j^{-1} + X_j' \Lambda_j^{-1} X_j. \tag{76a}$$

$$\frac{\partial \mathcal{G}}{\partial \Omega_j} = \Omega_j^{-1} - \Omega_j^{-1}(\tilde{\Psi}_j^{-1} + B_j)\Omega_j^{-1} \tag{76b}$$

$$\frac{\partial \mathcal{G}}{\partial \Lambda_j} = \Lambda_j^{-1} - \Lambda_j^{-1}(X_j \tilde{\Psi}_j^{-1} X_j' + C_j)\Lambda_j^{-1} \tag{76c}$$

Thus, if there are no restrictions on Ω, the update is

$$\Omega_j = \Psi_j^{-1} + B_j, \tag{77}$$

and if all Ω_j are restricted to be the same

$$\Omega = \frac{1}{m}\sum_{j=1}^{m}(\Psi_j^{-1} + B_j). \tag{78}$$

8. JENSEN MAJORIZATION

8.1. **Majorization.** A *majorization algorithm* to minimize a function $f(x)$ over $x \in X$ constructs a *majorization function* $g(x, y)$ on $X \otimes X$, such that

$$f(x) \leq g(x, y) \qquad \forall x, y \in X \tag{79a}$$

$$f(x) = g(x, x) \qquad \forall x \in X \tag{79b}$$

Clearly a majorization function defines an augmentation function, so augmentation is the more general process. If we apply block relaxation to a majorization function with

$$y_0 = \underset{y\in X}{argmin}\, g(x_0, y), \tag{80a}$$

$$x_1 = \underset{x\in X}{argmin}\, g(x, y_0), \tag{80b}$$

$$y_1 = \underset{y\in X}{argmin}\, g(x_0, y), \tag{80c}$$

we find $y_0 = x_0$, $y_1 = x_1$, and so on. Thus we can also write more briefly

$$x_1 = \underset{x\in X}{argmin}\, g(x, x_0), \tag{81a}$$

$$x_2 = \underset{x\in X}{argmin}\, g(x, x_1), \tag{81b}$$

8.2. **Jensen Majorization.** In Jensen majorization we use Jensen's inequality to get the majorization function. We use this in the situation where we are maximizing a function of the form

$$f(x) = \log \int h(x, y)dy, \tag{82}$$

where $h(x, y)$ is positive everywhere. Write

$$\log \frac{\int h(x, y)dy}{\int h(z, y)dy} = \log \int h(y \mid z)\frac{h(x, y)}{h(z, y)}dy, \tag{83a}$$

where

$$h(y \mid z) \triangleq \frac{h(z, y)}{f(z)}. \tag{83b}$$

Now, by Jensen's inequality,

$$\log \int h(y|z)\frac{h(x,y)}{h(z,y)}dy \geq \int h(y|z)\log\frac{h(x,y)}{h(z,y)}dy = \int h(y|z)\log h(x,y)dy - \int h(y|z)\log h(x,z)dy \tag{84}$$

and thus, summarizing, we have $f(x) = g(x,x) = \max_{z\in X} g(x,z)$, where

$$g(x,z) \stackrel{\Delta}{=} k(x,z) - k(z,z) + f(z). \tag{85a}$$

and

$$k(x,z) \stackrel{\Delta}{=} \int h(y \mid z)\log h(x,y)dy \tag{85b}$$

Maximizing $g(x,z)$ over $x \in X$ can be done by maximizing $k(x,z)$, which is the only term depending on x.

8.3. **Normal Likelihood.** We use an integral representation of the normal log-likelihood, which merely says that we get the marginal density of the observables by integrating the product of the conditional density of the observables given the second-order disturbances with the density of these second-order disturbances. Thus

$$f(\sigma_j^2,\Omega_j,\Lambda_j,\gamma) = \sum_{j-1}^{m}\log\int p_{\sigma_j^2,\Omega_j\Lambda_j,\gamma}(y_j \mid \delta_j)p_{\sigma_j^2,\Omega_j\Lambda_j,\gamma}(\delta_j)d\delta_j \tag{86}$$

Using Jensen Majorization means that we actually have to maximize

$$k(\sigma_j^2,\Omega_j,\Lambda_j,\gamma;\tilde{\sigma}_j^2,\tilde{\Omega}_j,\tilde{\Lambda}_j,\tilde{\gamma}) = \sum_{j-1}^{m}\int p_{\tilde{\sigma}_j^2,\tilde{\Omega}_j,\tilde{\Lambda}_j,\tilde{\gamma}}(\delta_j \mid y_j)\log p_{\sigma_j^2,\Omega_j,\Lambda_j,\gamma}(y_j,\delta_j)d\delta_j = \sum_{j-1}^{m}\mathbf{E}_{\tilde{\sigma}_j^2,\tilde{\Omega}_j,\tilde{\Lambda}_j,\tilde{\gamma}}\left\{\log p_{\sigma_j^2,\Omega_j,\Lambda_j,\gamma}(y_j,\delta_j) \mid y_j\right\}. \tag{87}$$

In order to simplify this, we first write the joint density as a product of the condtional and marginal densities. The conditional mean and variance of $\underline{y}_j$, given the second-order disturbances δ_j, is

$$\mathbf{E}(\underline{y}_j|\delta_j) = U_j\gamma + X_j\delta_j, \tag{88a}$$

$$\mathbf{V}(\underline{y}_j|\delta_j) = \sigma^2\Lambda_j. \tag{88b}$$

Thus two times the negative logarithm of the joint density of $\underline{y}_j$ and $\underline{\delta}_j$, assuming normality, and using (88), can be written as

$$\log\mathbf{det}(\sigma^2\Lambda_j)+\frac{1}{\sigma^2}(y_j-U_j\gamma-X_j\delta_j)'\Lambda_j^{-1}(y_j-U_j\gamma-X_j\delta_j)+\log\mathbf{det}(\sigma^2\Omega)+\frac{1}{\sigma^2}\delta_j'\Omega^{-1}\delta_j \tag{89}$$

To compute the majorization function, we need the conditional expectation of the logarithm given by (89). This can be expressed most simply by defining

$$\mu_j\triangleq\mathbf{E}(\underline{\delta}_j|y_j)=\Omega X_j'(X_j\Omega X_j'+\Lambda_j)^{-1}(y_j-U_j\gamma)=[X_j'\Lambda_j^{-1}X_j+\Omega^{-1}]^{-1}X_j'\Lambda_j^{-1}(y_j-U_j\gamma), \tag{90a}$$

and

$$\Sigma_j\triangleq\mathbf{V}(\underline{\delta}_j|y_j)=\sigma^2[\Omega-\Omega X_j'(X_j\Omega X_j'+\Lambda_j)^{-1}X_j\Omega]=\sigma^2[X_j'\Lambda_j^{-1}X_j+\Omega^{-1}]^{-1}. \tag{90b}$$

The part of the majorization function we have to minimize turns out to be

$$\begin{aligned}2k(\sigma_j^2,\Omega_j,\Lambda_j,\gamma;\tilde\sigma_j^2,\tilde\Omega_j,\tilde\Lambda_j,\tilde\gamma)=&\\ \log\mathbf{det}(\sigma_j^2\Lambda_j)+\log\mathbf{det}(\sigma_j^2\Omega_j)+&\\ \frac{1}{\sigma_j^2}(y_j-U_j\gamma-X_j\tilde\mu_j)'\Lambda_j^{-1}(y_j-U_j\gamma-X_j\tilde\mu_j)+&\\ \frac{1}{\sigma_j^2}\tilde\mu_j'\Omega_j^{-1}\tilde\mu_j+\frac{1}{\sigma_j^2}\mathbf{tr}\,(\Omega_j^{-1}+X_j'\Lambda_j^{-1}X_j)\tilde\Sigma_j.&\end{aligned} \tag{91}$$

The tilde above the symbols μ and Σ indicates they are evaluated at the current values of the parameters, which are $\tilde\sigma_j^2,\tilde\Omega_j,\tilde\Lambda_j,\tilde\gamma$.

8.4. **Jensen Majorization Algorithm.** The majorization function $g(x,z)$ is a function of the eight sets of parameters $\sigma_j^2,\Omega_j,\Lambda_j,\gamma$ and $\tilde\sigma_j^2,\tilde\Omega_j,\tilde\Lambda_j,\tilde\gamma$. This means that we could use block relaxation procedures that cycles over the eight blocks. Instead, we will use the fact that the minimum of $g(x,z)$ over z is attained at $z=x$, and thus we update by minimizing (91) by block relation of the four blocks $\sigma_j^2,\Omega_j,\Lambda_j,\gamma$ before computing a new superblock $\tilde\sigma_j^2,\tilde\Omega_j,\tilde\Lambda_j,\tilde\gamma$. In fact, we shall only carry our one cycle of upgrades of the four blocks, before computing the new superblock, although many variations are possible in which we use more than one cycle.

In the EM algorithm [McLachlan and Krishnan, 1997] updating the superblock is known as the E-step, and updating the four parameter blocks is the M-step.

8.4.1. *E-Step 1:* μ_j.

$$\hat{\mu}_j = \Omega_j X_j'(X_j\Omega_j X_j' + \Lambda_j)^{-1}(y_j - U_j\gamma) = [X_j'\Lambda_j^{-1}X_j + \Omega_j^{-1}]^{-1}X_j'\Lambda_j^{-1}(y_j - U_j\gamma), \tag{92}$$

8.4.2. *E-Step 2:* Σ_j.

$$\hat{\Sigma}_j = \sigma_j^2[\Omega_j - \Omega_j X_j'(X_j\Omega_j X_j' + \Lambda_j)^{-1}X_j\Omega_j] = = \sigma_j^2[X_j'\Lambda_j^{-1}X_j + \Omega_j^{-1}]^{-1}. \tag{93}$$

8.4.3. *M-Step 3:* γ.

$$\hat{\gamma} = (\sum_{j=1}^{m} U_j'\Lambda_j^{-1}U_j)^{-1}\sum_{j=1}^{m} U_j'\Lambda_j^{-1}(y_j - X_j\mu_j). \tag{94}$$

8.4.4. *M-Step 4:* σ_j^2.

$$\hat{\sigma}_j^2 = \frac{r_j'\Lambda^{-1}r_j + \mu_j'\Omega_j^{-1}\mu_j + \mathbf{tr}\,\Sigma_j(X_j'\Lambda_j^{-1}X_j + \Omega_j^{-1})}{n_j + q}. \tag{95}$$

If we require all σ_j^2 to be the same, this becomes

$$\hat{\sigma}^2 = \frac{\sum_{j=1}^{m} r_j'\Lambda^{-1}r_j + \mu_j'\Omega_j^{-1}\mu_j + \mathbf{tr}\,\Sigma_j(X_j'\Lambda_j^{-1}X_j + \Omega_j^{-1})}{n + mq}. \tag{96}$$

8.4.5. *M-Step 5:* Ω_j. We find

$$-2\frac{\partial k}{\partial \Omega_j} = \Omega_j^{-1} - \frac{1}{\sigma_j^2}\Omega_j^{-1}[\tilde{\mu}_j\tilde{\mu}_j' + \tilde{\Sigma}_j]\Omega_j^{-1}. \tag{97}$$

If all Ω_j are the same, and if they are unrestricted, this gives

$$\hat{\Omega} = \sum_{j=1}^{m}\frac{\tilde{\mu}_j\tilde{\mu}_j' + \tilde{\Sigma}_j}{\sigma_j^2} \tag{98}$$

8.4.6. *M-Step 6:* Λ_j. In much the same way as in the Ω_j step

$$-2\frac{\partial k}{\partial \Lambda_j} = \Lambda_j^{-1} - \frac{1}{\sigma_j^2}\Lambda_j^{-1}[\tilde{r}_j\tilde{r}_j' + X_j\tilde{\Sigma}_j X_j']\Lambda_j^{-1}. \tag{99}$$

But the model in which Λ_j are the same and unretricted is not of much interest.

APPENDIX A. PARTITIONED INVERSE AND DETERMINANT

Theorem 5. *The inverse of* $\begin{bmatrix} A & B \\ B' & C \end{bmatrix}$ *is*

$$\begin{bmatrix} A^{-1} - A^{-1}B(C - B'A^{-1}B)^{-1}B'A^{-1} & -A^{-1}B(C - B'A^{-1}B)^{-1} \\ -(C - B'A^{-1}B)^{-1}B'A^{-1} & (C - B'A^{-1}B)^{-1} \end{bmatrix} =$$
$$= \begin{bmatrix} (A - BC^{-1}B')^{-1} & -(A - BC^{-1}B')^{-1}BC^{-1} \\ -C^{-1}B'(A - BC^{-1}B')^{-1} & C^{-1} - C^{-1}B'(A - BC^{-1}B')^{-1}BC^{-1} \end{bmatrix},$$

provided all the relevant inverses exist.

Proof. We must have

$$\begin{bmatrix} A & B \\ B' & C \end{bmatrix} \begin{bmatrix} P & Q \\ Q' & R \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}, \tag{100}$$

i.e.

$$AP + BQ' = I, \tag{101a}$$
$$AQ + BR = 0, \tag{101b}$$
$$B'P + CQ' = 0, \tag{101c}$$
$$B'Q + CR = I. \tag{101d}$$

We see from (101b) that $Q = -A^{-1}BR$. Use this in (101d) to get $CR - B'A^{-1}BR = I$, i.e.

$$R = (C - B'A^{-1}B)^{-1}.$$

This gives

$$Q = -A^{-1}B(C - B'A^{-1}B)^{-1},$$

and finally, from (101a),

$$P = A^{-1} - A^{-1}BQ' = A^{-1} - A^{-1}B(C - B'A^{-1}B)^{-1}B'A^{-1}.$$

On the other hand, from (101c), $Q' = -C^{-1}B'P$, and thus, from (101a), $AP - BC^{-1}B'P = I$, i.e.

$$P = (A - BC^{-1}B')^{-1}.$$

This gives

$$Q = -(A - BC^{-1}B')^{-1}BC^{-1},$$

and finally, from (101d),

$$R = C^{-1} - C^{-1}B'(A - BC^{-1}B')^{-1}BC^{-1}.$$

□

Corollary 6 (Sherman-Morrison-Woodbury)**.** *If the relevant inverse exist then*

$$(A - BC^{-1}B')^{-1} = A^{-1} - A^{-1}B(C - B'A^{-1}B)^{-1}B'A^{-1}$$

Proof. This is just the upper left hand corner of the partitioned inverse from the previous theorem. □

Theorem 7. *The determinant of* $\begin{bmatrix} A & B \\ B' & C \end{bmatrix}$ *is*

$$\mathbf{det}(A)\mathbf{det}(C - B'A^{-1}B) = \mathbf{det}(C)\mathbf{det}(A - BC^{-1}B'),$$

provided the relevant inverses exist.

Proof. Clearly

$$\begin{bmatrix} A & B \\ B' & C \end{bmatrix} \begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & 0 \\ B' & C - B'A^{-1}B \end{bmatrix} \tag{102}$$

The rest follows by symmetry. □

APPENDIX B. OTHER MODIFIED INVERSE FORMULAS

A *modifed inverse formula* gives an expression for $(A + XBX')^{-1}$ in terms of its component matrices. We have already seen one such results, in Corollary 6. In multilevel models, the following result is even more useful.

Theorem 8. *Suppose* A *is positive definite,* B *is positive semi-definite, and* X *is of full column-rank. Then*

$$\begin{aligned}(A + XBX')^{-1} = \\ A^{-1}X(X'A^{-1}X)^{-1}[(X'A^{-1}X)^{-1} + B]^{-1}(X'A^{-1}X)^{-1}X'A^{-1} + \\ + [A^{-1} - A^{-1}X(X'A^{-1}X)^{-1}X'A^{-1}]\end{aligned}$$

Proof.

$$(A + XBX')^{-1} = A^{-1/2}[I + \tilde{X}B\tilde{X}']^{-1}A^{-1/2},$$

where $\tilde{X} = A^{-1/2}X$. Now

$$I + \tilde{X}B\tilde{X}' = \tilde{X}[(\tilde{X}'\tilde{X})^{-1} + B]\tilde{X}' + [I - \tilde{X}(\tilde{X}'\tilde{X})^{-1}\tilde{X}'],$$

and thus

$$\begin{aligned}(I + \tilde{X}B\tilde{X}')^{-1} = \\ \tilde{X}(\tilde{X}'\tilde{X})^{-1}[(\tilde{X}'\tilde{X})^{-1} + B]^{-1}(\tilde{X}'\tilde{X})^{-1}\tilde{X}' + [I - \tilde{X}(\tilde{X}'\tilde{X})^{-1}\tilde{X}']\end{aligned}$$

Combining these results gives the formula in the Theorem. □

By straightforward multiplication it actually follows that the formula is true for all non-singular A and for all B such that $(X'A^{-1}X)^{-1}+B$ has an inverse. There is no need for B to be definite, in fact the result even remains true for $B=0$.

APPENDIX C. THE MATRIX EQUATION $A=XBX$

Theorem 9. *Suppose A is positive semi-definite, and B is positive definite. Then the unique positive semi-definite solution of $A=XBX$ is $X=B^{-1/2}(B^{1/2}AB^{1/2})^{1/2}B^{-1/2}$.*

Proof. We rewrite the equation as

$$B^{1/2}AB^{-1/2}=(B^{1/2}XB^{1/2})(B^{1/2}XB^{1/2}),$$

which shows that $B^{1/2}XB^{1/2}$ is the symmetric square root of $B^{1/2}AB^{1/2}$. Thus $B^{1/2}XB^{1/2}=(B^{1/2}AB^{1/2})^{1/2}$, which leads to the result in the theorem. □

Theorem 10. *Suppose A is positive semi-definite, and B is positive semi-definite.*

(1) *if A is positive definite and B is singular, then $A=XBX$ does not have a solution.*
(2) *If $A=XBX$ is solvable, then a solution is*

$$X=B^{-1/2}(B^{1/2}AB^{1/2})^{1/2}B^{-1/2},$$

where $B^{-1/2}$ is now defined as the square root of the Moore-Penrose inverse.

Proof. If A is positive definite, then the solution X cannot be singular. If X was singular, then there is a nonzero z such that $Xz=0$, and thus $Az=XBXz=0$, contradicting non-singularity of a.

If A is positive definite, B is singular, and X is non-singular, then there is a nonzero z such that $Bz=0$. Let $y\overset{\Delta}{=}X^{-1}z$. Then $Ay=XBXX^{-1}z=0$, again contradicting that A is non-singular. This proves the first part.

Because square roots of positive semidefinite matrices are uniquely defined, we can stiil conclude that $B^{1/2}XB^{1/2}=(B^{1/2}AB^{1/2})^{1/2}$.

Suppose $B=K\Lambda^2K'$, with Λ, or order r, where r is the rank of B. Also K_0, is an orthonormal basis for the null space of B. Now $B^{1/2}=K\Lambda K'$ is still uniquely defined, and thus we can still conclude that □

APPENDIX D. LINEAR MAJORIZATION OF THE DETERMINANT

Theorem 11. *Suppose A and B are positive definite. Then*

$$\log\mathbf{det}(A)\leq\log\mathbf{det}(B)+\mathbf{tr}\ B^{-1}(A-B)$$

Moreover we have equality if and only if $A = B$.

Proof. Because A and B are positive definite, there exists an S such that $B = SS'$ and $A = S\Phi S'$, with Φ diagonal with positive diagonal elements ϕ_s. After substituting these expressions for A and B the result we want to prove becomes

$$\sum \log \phi_s \leq \sum (\phi_s - 1),$$

with equality if and only if $\phi_s = 1$ for all s. This follows trivially from the strict concavity of the logarithm. □

Theorem 12. *Suppose* A, B *and* C*are positive definite. Then*

$$\mathbf{tr}\ A^{-1}C \geq \mathbf{tr}\ B^{-1}C - \mathbf{tr}\ B^{-1}(A-B)B^{-1}C$$

Moreover we have equality if and only if $A = B$.

Proof. We proceed in the same way as in the proof of the previous theorem. The result we have to prove becomes

$$\sum \frac{d_s}{\phi_s} \geq \sum d_s - (1 - \phi_s)d_s,$$

where d_s are the diagonal elements of $S^{-1}C(S^{-1})'$. This amounts to showing $(\phi_i - 1)^2 \geq 0$, which is obviously true. □

(103) $\mathbf{tr}\ A^{-1}C = \mathbf{tr}\ [B + (A-B)]^{-1}C =$
$\mathbf{tr}\ B^{-1}C - \mathbf{tr}\ B^{-1}(A-B)B^{-1}C + \mathbf{tr}\ D^{-1}(A-B)D^{-1}(A-B)D^{-1}C,$

where D is on the line connecting A and B. Let us look at the last term in detail, using Δ for $A - B$, and E for $D^{-1}CD^{-1}$. Then

$$\text{(104)}\quad \mathbf{tr}\ \Delta D^{-1}\Delta E = \sum_i \sum_j \delta_{ij}(D^{-1}\Delta E)_{ji} =$$
$$\sum_i \sum_j \delta_{ij} \sum_k d^{jk}(\Delta E)_{ki} = \sum_i \sum_j \delta_{ij} \sum_k d^{jk} \sum_\ell \delta_{k\ell} e_{\ell i} =$$
$$\sum_i \sum_j \sum_k \sum_\ell \delta_{ij}\delta_{kl} d^{jk} e_{\ell i}.$$

APPENDIX E. CODE

Our programs are written in R, the statistical environment also known as GNU S. For additional information about R we refer to Dalgaard [2002], for use of R in the geosciences and geography see Bivand and Gebhardt [2000], Grunsky [2002].

```
#####################################################################
#### need for input:
#### 1. first level predictors (organized in a matrix)
#### 2. response (organized in a vector)
#### 3. second level predictors (organized in a matrix)
#### 4. coordinates (a n by 2 matrix)
#### 5. a vector indicating number of transects within each site
#### 6. a vector indicating which first level predictors are related
#### 7. a vector indicating which fistt level predictors have random
#### 8. a real number. when the change of log-likehood after one ite
#### 9. a binary variable indicating whether we should take spatial
#### 10. a variable indicating which kind of form the omiga matrix i
#####################################################################


## input a weight matrix, first row normalize then column normalize,
weight.normalize<-function(w, error=0.000001)
{
  w.cur<-w
  repeat
  {
    w.pre<-w.cur
    for(j in 1:nrow(w.cur))
    {
      s<-sum(w.cur[j,])
      w.cur[j,]<-w.cur[j,]/s
    }
    for(j in 1:ncol(w.cur))
    {
      s<-sum(w.cur[,j])
      w.cur[,j]<-w.cur[,j]/s
    }
    w.cur<-(w.cur+t(w.cur))/2
    if(max(abs(w.pre-w.cur))<error) break
   }
  return (w.cur)
}

## get the Euclidean distance
```

```
dist<-function(a,b)
{
  result<-sum((a-b)^2)
  result<-sqrt(result)
  return(result)
}

## helper function
l<-0.00001
f<-function(x)
{ return(exp(-l*x*x))}
f1<-function(x)
{ return(1/x)}


spatial<-function(x,y,z,coor,lev2.index,index.gamma,index.omiga,erro
{
  site.n<-nrow(lev2.index)

  x.exp<-cbind(1,x)

  #calculate all w_j
  for (i in 1:site.n)
  {
    temp.w<-matrix(0,lev2.index[i,1],lev2.index[i,1])
    if(i==1) previous.end<-0
    else     previous.end<-sum(lev2.index[1:(i-1),1])
    for (j in 1:lev2.index[i,1])
    {
      corj<-previous.end+j
      for(k in 1:j)
      {
        cork<-previous.end+k
        if (j!=k)
        {
          temp.w[j,k]<-f1(dist(c(coor[corj,1],coor[corj,2]),c(coor[c
          temp.w[k,j]<-temp.w[j,k]
        }
      }
```

```
  }
  temp.w<-weight.normalize(temp.w)
  if(i==1) w1<-list(temp.w)
  else w1[[i]]<-temp.w
}


#calculate all w_j
for (i in 1:site.n)
{
  w.temp<-diag(rep(1,lev2.index[i]-1))
  temp.w<-matrix(0,lev2.index[i],lev2.index[i])
  temp.w[2:lev2.index[i],1:(lev2.index[i]-1)]<-w.temp
  if(i==1) w2<-list(temp.w)
  else w2[[i]]<-temp.w
}

#w1<-w2

#calculate the Z_j
z.exp<-cbind(1,z)
for (i in 1:site.n)
{
  temp.z<-matrix(0,ncol(x.exp),ncol(x.exp)*ncol(z.exp))
  if(i==1) previous.end<-0
  else     previous.end<-sum(lev2.index[1:(i-1),1])
  for (j in 1:ncol(x.exp))
  {
    temp.z[j,((j-1)*ncol(z.exp)+1):(j*ncol(z.exp))]<-z.exp[previou
  }
  if(i==1) Z<-list(temp.z)
  else Z[[i]]<-temp.z
}

#make x,y into lists
for(i in 1:site.n)
{
  if(i==1)
  {
```

```
    X<-list(x.exp[1:lev2.index[i],])
    Y<-list(y[1:lev2.index[i]])
  }
  else
  {
    X[[i]]<-x.exp[(sum(lev2.index[1:(i-1)])+1):(sum(lev2.index[1:i
    Y[[i]]<-y[(sum(lev2.index[1:(i-1)])+1):(sum(lev2.index[1:i]))]
  }
}

index.seq<-c(1,2,3,4,5,0)
#index.seq<-c(1,0,2,5,3,4)
get.index<-function(ind)
{ return (index.seq[ind]) }

get.R<-function(gamma.cur,v.cur)
{
  R<-matrix(0,nrow(x.exp),1)
  for (i in 1:site.n)
  {
    if(i==1) R<-list(Y[[i]]-(X[[i]]%*%Z[[i]])[,index.gamma]%*%gamn
    else R[[i]]<-Y[[i]]-(X[[i]]%*%Z[[i]])[,index.gamma]%*%gamma.cu
  }
  return(R)
}

get.sigma<-function(theta.cur,gamma.cur,v.cur,omiga.cur)
{
  R<-get.R(gamma.cur,v.cur)
  temp<-0
  for (i in 1:site.n)
  {
    A.i<-diag(1,lev2.index[i],lev2.index[i])-theta.cur*w1[[i]]
    clumda.i<-solve(A.i)
    temp<-temp+t(R[[i]])%*%clumda.i%*%R[[i]]+t(v.cur[[i]])%*%solve
  }
  sigma.square.cur<-(temp/nrow(x.exp))[1,1]
  return(sigma.square.cur)
}
```

```
get.H<-function(theta.cur,omiga.cur)
{
  for (i in 1:site.n)
  {
    A.i<-diag(1,lev2.index[i],lev2.index[i])-theta.cur*w1[[i]]
    H.cur[[i]]<-X[[i]][,index.omiga]%*%omiga.cur%*%t(X[[i]][,index
  }
  return(H.cur)
}

get.v<-function(theta.cur,omiga.cur,gamma.cur)
{
  for (i in 1:site.n)
  {
    A.i<-diag(1,lev2.index[i],lev2.index[i])-theta.cur*w1[[i]]
    clumda.i<-solve(A.i)
    v.cur[[i]]<-solve(t(X[[i]][,index.omiga])%*%clumda.i%*%X[[i]][
  }
  return(v.cur)
}

get.gamma<-function(dimension,theta.cur,v.cur)
{
  temp1<-matrix(0,dimension,dimension)
  temp2<-matrix(0,dimension,1)
  for (i in 1:site.n)
  {
    A.i<-diag(1,lev2.index[i],lev2.index[i])-theta.cur*w1[[i]]
    clumda.i<-solve(A.i)
    temp1<-temp1+t((X[[i]]%*%Z[[i]])[,index.gamma])%*%clumda.i%*%(
    temp2<-temp2+t((X[[i]]%*%Z[[i]])[,index.gamma])%*%clumda.i%*%(
  }
  gamma.cur<-solve(temp1)%*%temp2
  return(gamma.cur)
}

get.omiga<-function(v.cur,sigma.square.cur,H.cur)
{
```

```
  dimension<-nrow(as.matrix(index.omiga))
  A<-matrix(0,dimension,dimension)
  B<-matrix(0,dimension,dimension)
  for (i in 1:site.n)
  {
    A<-A+t(X[[i]][,index.omiga])%*%solve(H.cur[[i]])%*%X[[i]][,ind
    B<-B+v.cur[[i]]%*%t(v.cur[[i]])
  }
  B<-B/sigma.square.cur
  B.u<-as.matrix(eigen(B)$vectors)
  B.lumda<-eigen(B)$values
  B.half<-B.u%*%diag(sqrt(B.lumda),nrow(as.matrix(index.omiga)),nr
  temp<-B.half%*%A%*%B.half
  temp.value<-eigen(temp)$values
  temp.vec<-eigen(temp)$vectors
  temp.half<-temp.vec%*%diag(sqrt(temp.value),nrow(as.matrix(index
  omiga.temp<-solve(solve(B.half)%*%temp.half%*%solve(B.half))
  return(list(omiga=omiga.temp,B.lumda=B.lumda,half=temp.value))
}

get.omiga.2<-function(v.cur,sigma.square.cur,H.cur)
  {
    temp1<-0
    temp2<-0
    for(i in 1:site.n)
    {
      temp1<-temp1+sum(diag(solve(H.cur[[i]])%*%X[[i]][,index.omig
      temp2<-temp2+t(v.cur[[i]])%*%v.cur[[i]]
    }
    temp2<-temp2/sigma.square.cur
    theta.omiga<-sqrt(temp2[1,1]/temp1)
    omiga.cur<-theta.omiga*diag(1,nrow(as.matrix(index.omiga)),nro
    return(omiga.cur)
  }



get.ml.theta<-function(theta.temp,H.cur,R,sigma.square.cur)
{
```

```
  temp<-0
  for(i in 1:site.n)
  {
    A.i.temp<-diag(1,lev2.index[i],lev2.index[i])-theta.temp*w1[[i
    clumda.i.temp<-solve(A.i.temp)
    temp1<-sum(diag(solve(H.cur[[i]])%*%A.i.temp))
    temp2<-(1/sigma.square.cur)*(t(R[[i]])%*%clumda.i.temp%*%R[[i]
    temp<-temp+temp1+temp2
  }
  return(temp[1,1])
}

get.theta.search<-function(sigma.square.cur,gamma.cur,v.cur,omiga.
{
  R<-get.R(gamma.cur,v.cur)
  par(mfrow=c(2,3))
  num.search<-10
  star.p<-theta.cur
  index.s<-star.p
  ml<-get.ml.theta(star.p,H.cur,R,sigma.square.cur)
  for (k in 1:3)
  {
    gh.ml<-rep(ml,2*num.search-1)
    step<-1/(num.search^k)
    i<-1
    repeat
    {
      theta.temp<-star.p+i*step
      #print(theta.temp)
      if(theta.temp>=1) break
      temp<-get.ml.theta(theta.temp,H.cur,R,sigma.square.cur)
      gh.ml[i]<-temp
      if(ml>=temp)
      {
        index.s<-star.p+i*step
        ml<-temp
        i<-i+1
      }
      else {break}
```

```
      }
      plot(gh.ml)
      gh.ml<-rep(ml,2*num.search-1)
      i<-1
      repeat
      {
        theta.temp<-star.p-i*step
        #print(theta.temp)
        if(theta.temp<0) break
        temp<-get.ml.theta(theta.temp,H.cur,R,sigma.square.cur)
        gh.ml[i]<-temp
        if(ml>=temp)
        {
          index.s<-star.p-i*step
          ml<-temp
          i<-i+1
        }
        else {break}
      }
      plot(gh.ml)
      theta.k<-index.s
      #print(theta.k)
      star.p<-theta.k
    }
    theta.cur<-theta.k
    return(theta.cur)
}

get.mle<-function(sigma.square.cur,H.cur,omiga.cur,theta.cur,gamma
{
    R<-get.R(gamma.cur,v.cur)
    temp1<-0
    temp2<-0
    for(j in 1:site.n)
    {
      A.j<-diag(1,lev2.index[j],lev2.index[j])-theta.cur*w1[[j]]
      clumda.j<-solve(A.j)
      temp1<-temp1+log(det(H.cur[[j]]))+sum(diag(solve(H.cur[[j]])%*
      temp2<-temp2+t(R[[j]])%*%clumda.j%*%R[[j]]+t(v.cur[[j]])%*%sol
```

```
    }
    result<-sum(lev2.index)*log(sigma.square.cur)+temp1+temp2/sigma.
    return(result)
  }

  #debug1<-1:9
  #initialize parameters

  d.omiga<-nrow(as.matrix(index.omiga))
  omiga.cur<-diag(1,d.omiga,d.omiga)
  theta.cur<-0
  for(i in 1:site.n)
  {
    if(i==1)
    {
      v.cur<-list(as.matrix(rep(1,nrow(as.matrix(index.omiga)))))
      v.0<-list(as.matrix(rep(0,nrow(as.matrix(index.omiga)))))
      H.cur<-list(diag(rep(1,lev2.index[i])))
    }
    else
    {
      v.cur[[i]]<-as.matrix(rep(1,nrow(as.matrix(index.omiga))))
      v.0[[i]]<-as.matrix(rep(0,nrow(as.matrix(index.omiga))))
      H.cur[[i]]<-diag(rep(1,lev2.index[i]))
    }
  }
  dimension<-nrow(as.matrix(index.gamma))
  sigma.square.cur<-1
  gamma.cur<-get.gamma(dimension,theta.cur,v.0)

    sign<-TRUE
  mle.cur<-get.mle(sigma.square.cur,H.cur,omiga.cur,theta.cur,gamma.

  #do loop to implement CCA
  ind<-0
  count<-0

  repeat
  {
```

```
    count<-count+1
    print("new loop begins")
    print(count)

##########These pre's may not be needed except mle.pre
     mle.pre<-mle.cur

    ml.pre<-mle.pre

    for (integer in 1:6)
    {
      ind<-ind%%6+1
      index<-get.index(ind)

      if (index==3)
      {
        sigma.square.cur<-get.sigma(theta.cur,gamma.cur,v.cur,omiga.

      }
      else if (index==1)
      {
       H.cur<-get.H(theta.cur,omiga.cur)

      }
      else if (index==2)
      {
        v.cur<-get.v(theta.cur,omiga.cur,gamma.cur)
      }
      else if(index==4)
      {
        gamma.cur<-get.gamma(dimension,theta.cur,v.cur)

      }
      else if(index==5)
      {
        temp<-get.omiga(v.cur,sigma.square.cur,H.cur)
        omiga.cur<-temp$omiga

      }
```

```
      else if(index==0)
      {
        if(effect.spatial==TRUE){
        theta.cur<-get.theta.search(sigma.square.cur,gamma.cur,v.cur
        #R<-get.R(gamma.cur,v.cur)          #theta.cur<-optimize(f=get
        }
      }
      ml.cur<-get.mle(sigma.square.cur,H.cur,omiga.cur,theta.cur,gam
      if(ml.cur>ml.pre) { print(index)
       sign<-FALSE
       print("error")
       print(temp$B.lumda)
       print(temp$half)
       }
      ml.pre<-ml.cur
    }
    mle.cur<-get.mle(sigma.square.cur,H.cur,omiga.cur,theta.cur,gamm
    if(abs(mle.cur-mle.pre)<error) break

    print(mle.cur[1,1])
    #debug1[count]<-theta.cur
    #if(count==7) break
  }
  variance.gamma<-matrix(0,dimension,dimension)
  for(i in 1:site.n)
  {
    variance.gamma<-variance.gamma+t((X[[i]]%*%Z[[i]])[,index.gamma]
  }
  variance.gamma<-solve(variance.gamma)*sigma.square.cur

  ## AIC & BIC
  RSS<-0
  for(i in 1:site.n)
  {
    residual.i<-Y[[i]]-(X[[i]]%*%Z[[i]])[,index.gamma]%*%gamma.cur
    RSS<-RSS+(t(residual.i)%*%residual.i)[1,1]
  }
  para.num.gamma<-nrow(as.matrix(index.gamma))
  if(omiga.form=="general") para.num.omiga<-(nrow(as.matrix(index.or
```

```
  else if(omiga.form=="diag") para.num.omiga<-nrow(as.matrix(index.c
  else para.num.omiga<-1
  K<-1+para.num.gamma+para.num.omiga+1
  AIC<-log(RSS/nrow(x.exp))+2*K/nrow(x.exp)
  BIC<-log(RSS/nrow(x.exp))+K*log(nrow(x.exp))/nrow(x.exp)

return(list(sigma=sigma.square.cur,gamma=gamma.cur,omiga=omiga.cur,t
}
```

REFERENCES

L. Anselin. Spatial Regression. URL `http://geog55.gis.uiuc.edu/~luc/talks/spreg.pdf`. 2001.

F. Bavaud. Models for Spatial Weights: a Systematic Look. *Geographical Analysis*, 30:153–171, 1998.

R. Bivand and A. Gebhardt. Implementing funtions for spatial statistical analysis using the R language. *Journal of Geographical Systems*, 2:307–317, 2000.

N. Cressie. *Statistics for Spatial Data*. Wiley, 1991.

P. Dalgaard. *Introductory Statistics with R*. Springer, 2002.

J. de Leeuw and I. G. G. Kreft. Random Coefficient Models for Multilevel Analysis. *Journal of Educational Statistics*, 11:57–85, 1986.

D.A. Griffith. Quick but no so Dirty. ML Estimation of Spatial Autoregressive Models. Technical report, Department of Geography, Syracuse University, 2002a.

D.A. Griffith. Spatial Autoregression. Technical report, Department of Geography, Syracuse University, 2002b.

E.C. Grunsky. R: a data analysis and statistical processing environment – an emerging tool for the geosciences. *Computers and Geosciences*, 28: 1219–1222, 2002.

D. R. Hedeker and R.D. Gibbons. MIXREG: a Computer Program for Mixed-Effects Regression Analysis with Autocorrelated Errors. *Computer Methods and Programs in Biomedicine*, 49:229–252, 1996.

D.R. Hedeker. *Random Regression Models with Autocorrelated Errors: Investigating Drug Plasma Levels and Clinical Response*. PhD thesis, University of Chicago at Illinois, 1989.

F.R. De Hoog, T.P. Speed, and E.R. Williams. On a Matrix Indentity Associated with Generalized Least Squares. *Linear Algebra and Its Appplications*, 127:449–456, 1990.

G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New York, 1997.

K. Ord. Estimation Methods for Models of Spatial Interaction. *Journal of the American Statistical Association*, 70:120–126, 1975.

R. K. Pace and R. Barry. Fast CARs. *Journal of Statistical Computation and Simulation*, 59:123–147, 1997a.

R. K. Pace and R. Barry. Quick Computation of Spatial Autoregressive Estimators. *Geographic Analysis*, 29:232–246, 1997b.

R. K. Pace and R. Barry. Sparse Spatial Autoregressions. *Statistics and Probability Letters*, 33:291–297, 1997c.

R.K. Pace and D. Zou. Closed-form maximum likelihood estimates of nearest neighbor spatial dependence. *Geographic Analysis*, 32, 2000.

R.K. Page and J. P. LeSage. Conditional Autoregressions with Doubly Stochastic Weight Matrices. *(under review)*, 2002.

O. Smirnov and L. Anselin. Fast maximum likelihood estimation of very large spatial autoregressive models: a characteristic polynnomial approach. *Computational Statistics and Data Analysis*, 35:301–319, 2001.