The Theory and Practice of Spatial Econometrics

James P. LeSage
Department of Economics
University of Toledo

February, 1999

Preface

This text provides an introduction to spatial econometric theory along with numerous applied illustrations of the models and methods described. The applications utilize a set of MATLAB functions that implement a host of spatial econometric estimation methods. The intended audience is faculty, students and practitioners involved in modeling spatial data sets. The MATLAB functions described in this book have been used in my own research as well as teaching both undergraduate and graduate econometrics courses. They are available on the Internet at http://www.econ.utoledo.edu along with the data sets and examples from the text.

The theory and applied illustrations of conventional spatial econometric models represent about half of the content in this text, with the other half devoted to Bayesian alternatives. Conventional maximum likelihood estimation for a class of spatial econometric models is discussed in one chapter, followed by a chapter that introduces a Bayesian approach for this same set of models. It is well-known that Bayesian methods implemented with a diffuse prior simply reproduce maximum likelihood results, and we illustrate this point. However, the main motivation for introducing Bayesian methods is to extend the conventional models. Comparative illustrations demonstrate how Bayesian methods can solve problems that confront the conventional models. Recent advances in Bayesian estimation that rely on Markov Chain Monte Carlo (MCMC) methods make it easy to estimate these models. This approach to estimation has been implemented in the spatial econometric function library described in the text, so estimation using the Bayesian models require a single additional line in your computer program.

Some of the Bayesian methods have been introduced in the regional science literature, or presented at conferences. Space and time constraints prohibit any discussion of implementation details in these forums. This text describes the implementation details, which I believe greatly enhance understanding and allow users to make intelligent use of these methods in applied settings. Audiences have been amazed (and perhaps skeptical) when I tell them it takes only 10 seconds to generate a sample of 1,000 MCMC draws from a sequence of conditional distributions needed to estimate the Bayesian models. Implementation approaches that achieve this type of speed are described here in the hope that other researchers can apply these ideas in their own work.

I have often been asked about Monte Carlo evidence for Bayesian spatial econometric methods. Large and small sample properties of estimation procedures are frequentist notions that make no sense in a Bayesian setting. The best support for the efficacy of Bayesian methods is their ability to provide solutions to applied problems. Hopefully, the ease of using these methods will encourage readers to experiment with these methods and compare the Bayesian results to those from more conventional estimation methods.

Implementation details are also provided for maximum likelihood methods that draw on the sparse matrix functionality of MATLAB and produce rapid solutions to large applied problems with a minimum of computer memory. I believe the MATLAB functions for maximum likelihood estimation of conventional models presented here represent fast and efficient routines that are easier to use than any available alternatives.

Talking to colleagues at conferences has convinced me that a simple software interface is needed so practitioners can estimate and compare a host of alternative spatial econometric model specifications. An example in Chapter 5 produces estimates for ten different spatial autoregressive models, including maximum likelihood, robust Bayesian, and a robust Bayesian tobit model. Estimation, printing and plotting of results for all these models is accomplished with a 39 line program.

Many researchers ignore sample truncation or limited dependent variables because they face problems adapting existing spatial econometric software to these types of sample data. This text describes the theory behind robust Bayesian logit/probit and tobit versions of spatial autoregressive models and geographically weighted regression models. It also provides implementation details and software functions to estimate these models.

Toolboxes are the name given by the MathWorks to related sets of MATLAB functions aimed at solving a particular class of problems. Toolboxes of functions useful in signal processing, optimization, statistics, finance and a host of other areas are available from the MathWorks as add-ons to the standard MATLAB software distribution. I use the term *Econometrics Toolbox* to refer to my public domain collection of function libraries available at the internet address given above. The MATLAB spatial econometrics functions used to implement the spatial econometric models discussed in this text rely on many of the functions in the *Econometrics Toolbox*. The spatial econometric functions constitute a "library" within the broader set of econometric functions. To use the spatial econometrics function library you need to download and install the entire set of *Econometrics Toolbox* functions. The spatial econometrics function library is part of the *Econometrics Toolbox* and will be available for use along with more traditional econometrics functions. The collection of around 500 econometrics functions and demonstration programs are organized into libraries, with approximately 40 spatial econometrics library functions described in this text. A manual is available for the *Econometrics Toolbox* in Acrobat PDF and postscript on the internet site, but this text should provide all the information needed to use the spatial econometrics library.

A consistent design was implemented that provides documentation, example programs, and functions to produce printed as well as graphical presentation of estimation results for all of the econometric and spatial econometric functions. This was accomplished using the “structure variables” introduced in MATLAB Version 5. Information from estimation procedures is encapsulated into a single variable that contains “fields” for individual parameters and statistics related to the econometric results. A thoughtful design by the MathWorks allows these structure variables to contain scalar, vector, matrix, string, and even multi-dimensional matrices as fields. This allows the econometric functions to return a single structure that contains all estimation results. These structures can be passed to other functions that intelligently decipher the information and provide a printed or graphical presentation of the results.

The *Econometrics Toolbox* along with the spatial econometrics library functions should allow faculty to use MATLAB in undergraduate and graduate level courses with absolutely no programming on the part of students or faculty. Practitioners should be able to apply the methods described in this text to problems involving large spatial data samples using an input program with less than 50 lines.

Researchers should be able to modify or extend the existing functions in the spatial econometrics library. They can also draw on the utility routines and other econometric functions in the *Econometrics Toolbox* to implement and test new spatial econometric approaches. I have returned from conferences and implemented methods from papers that were presented in an hour or two by drawing on the resources of the *Econometrics Toolbox*.

This text has another goal, applied modeling strategies and data analysis. Given the ability to easily implement a host of alternative models and produce estimates rapidly, attention naturally turns to which models best summarize a particular spatial data sample. Much of the discussion in this text involves these issues.

My experience has been that researchers tend to specialize, one group is devoted to developing new econometric procedures, and another group focuses on applied problems that involve using existing methods. This text should have something to offer both groups. If those developing new spatial econometric procedures are serious about their methods, they should take the time to craft a generally useful MATLAB function that others can use in applied research. The spatial econometrics function library provides an illustration of this approach and can be easily extended to include new functions. It would also be helpful if users who produce generally useful functions that extend the spatial econometrics library would submit them for inclusion. This would have the added benefit of introducing these new research methods to faculty and their students.

There are obviously omissions, bugs and perhaps programming errors in the *Econometrics Toolbox* and the spatial econometrics library functions. This would likely be the case with any such endeavor. I would be grateful if users would notify me via e-mail at jpl@jpl.econ.utoledo.edu when they encounter problems. The toolbox is constantly undergoing revision and new functions are being added. If you're using these functions, update to the latest version every few months and you'll enjoy speed improvements along with the benefits of new methods. Instructions for downloading and installing these functions are in an Appendix to this text along with a listing of the functions in the library and a brief description of each.

Numerous people have helped in my spatial econometric research efforts and the production of this text. John Geweke explained the mysteries of MCMC estimation when I was a visiting scholar at the Minneapolis FED. He shared his FORTRAN code and examples without which MCMC estimation might still be a mystery. Luc Anselin with his encylopedic knowledge of the field was kind enough to point out errors in my early work on MCMC estimation of the Bayesian models and set me on the right track. He has always been encouraging and quick to point out that he explored Bayesian spatial econometric methods in 1980. Kelley Pace shared his sparse matrix MATLAB code and some research papers that ultimately lead to the fast and efficient approach used in MCMC estimation of the Bayesian models. Dan McMillen has been encouraging about my work on Bayesian spatial autoregressive probit models. His research in the area of limited dependent variable versions of these models provided the insight for the Bayesian logit/probit and tobit spatial autoregressive methods in this text. Another paper he presented suggested the logit and probit versions of the geographically weighted regression models discussed in the text. Art Getis with his common sense approach to spatial statistics encouraged me to write this text so skeptics would see that the methods really work. Two colleagues of mine, Mike Dowd and Dave Black were brave enough to use the *Econometrics Toolbox* during its infancy and tell me about strange problems they encountered. Their feedback was helpful in making improvements that all users will benefit from. In addition, Mike Dowd the local LaTeX guru provided some helpful macros for formatting and indexing the examples in this text. Mike Magura, another colleague and co-author in the area of spatial econometrics read early versions of my text materials and made valuable comments. Last but certainly not least, my wife Mary Ellen Taylor provided help and encouragement in ways too numerous to mention. I think she has a Bayesian outlook on life that convinces me there is merit in these methods.

Contents

List of Examples

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an overview of the nature of spatial econometrics. An applied approach is taken where the central problems that necessitate special models and econometric methods for dealing with spatial economic phenomena are introduced using spatial data sets. Chapter 2 describes software design issues related to a spatial econometric function library based on MATLAB software from the MathWorks Inc. Details regarding the construction and use of functions that implement spatial econometric estimation methods are provided throughout the text. These functions provide a consistent user-interface in terms of documentation and related functions that provide printed as well as graphical presentation of the estimation results. Chapter 2 describes the function library using simple regression examples to illustrate the design philosophy and programming methods that were used to construct the spatial econometric functions.

The remaining chapters of the text are organized along the lines of alternative spatial econometric estimation procedures. Each chapter discusses the theory and application of a different class of spatial econometric model, the associated estimation methodology and references to the literature regarding these methods.

Section 1.1 discusses the nature of spatial econometrics and how this text compares to other works in the area of spatial econometrics and statistics. We will see that spatial econometrics is characterized by: 1) spatial dependence between sample data observations at various points in space, and 2) spatial heterogeneity that arises from relationships or model parameters that vary with our sample data as we move through space.

The nature of spatially dependent or spatially correlated data is taken up in Section 1.2 and spatial heterogeneity is discussed in Section 1.3. Section 1.4 takes up the subject of how we formally incorporate the locational information from spatial data in econometric models, providing illustrations based on a host of different spatial data sets that will be used throughout the text.

1.1 Spatial econometrics

Applied work in regional science relies heavily on sample data that is collected with reference to location measured as points in space. The subject of how we incorporate the locational aspect of sample data is deferred until Section 1.4. What distinguishes spatial econometrics from traditional econometrics? Two problems arise when sample data has a locational component: 1) spatial dependence between the observations and 2) spatial heterogeneity in the relationships we are modeling.

Traditional econometrics has largely ignored these two issues, perhaps because they violate the Gauss-Markov assumptions used in regression modeling. With regard to spatial dependence between the observations, recall that Gauss-Markov assumes the explanatory variables are fixed in repeated sampling. Spatial dependence violates this assumption, a point that will be made clear in the Section 1.2. This gives rise to the need for alternative estimation approaches. Similarly, spatial heterogeneity violates the Gauss-Markov assumption that a single linear relationship with constant variance exists across the sample data observations. If the relationship varies as we move across the spatial data sample, or the variance changes, alternative estimation procedures are needed to successfully model this variation and draw appropriate inferences.

The subject of this text is alternative estimation approaches that can be used when dealing with spatial data samples. This subject is seldom discussed in traditional econometrics textbooks. For example, no discussion of issues and models related to spatial data samples can be found in Amemiya (1985), Chow (1983), Dhrymes (1978), Fomby et al. (1984), Green (1997), Intrilligator (1978), Kelejian and Oates (1989), Kmenta (1986), Maddala (1977), Pindyck and Rubinfeld (1981), Schmidt (1976), and Vinod and Ullah (1981).

Anselin (1988) provides a complete treatment of many facets of spatial econometrics which this text draws upon. In addition to discussion of ideas set forth in Anselin (1988), this text includes Bayesian approaches as well as conventional maximum likelihood methods for all of the spatial econometric methods discussed in the text. Bayesian methods hold a great deal of appeal in spatial econometrics because many of the ideas used in regional science modeling involve:

1. a decay of sample data influence with distance
2. similarity of observations to neighboring observations
3. a hierarchy of place or regions
4. systematic change in parameters with movement through space

Traditional spatial econometric methods have tended to rely almost exclusively on sample data to incorporate these ideas in spatial models. Bayesian approaches can incorporate these ideas as subjective prior information that augments the sample data information.

It may be the case that the quantity or quality of sample data is not adequate to produce precise estimates of decay with distance or systematic parameter change over space. In these circumstances, Bayesian methods can incorporate these ideas in our models, so we need not rely exclusively on the sample data.

In terms of focus, the materials presented here are more applied than Anselin (1988), providing details on the program code needed to implement the methods and multiple applied examples of all estimation methods described. Readers should be fully capable of extending the spatial econometrics function library described in this text, and examples are provided showing how to add new functions to the library. In its present form the spatial econometrics library could serve as the basis for a graduate level course in spatial econometrics. Students as well as researchers can use these programs with absolutely no programming to implement some of the latest estimation procedures on spatial data sets.

Another departure from Anselin (1988) is in the use of sparse matrix algorithms available in the MATLAB software to implement spatial econometric estimation procedures. The implementation details for Bayesian methods as well as the use of sparse matrix algorithms represent previously unpublished material. All of the MATLAB functions described in this text are freely available on the Internet at http://www.econ.utoledo.edu. The spatial econometrics library functions can be used to solve large-scale spatial econometric problems involving thousands of observations in a few minutes on a modest desktop computer.

1.2 Spatial dependence

Spatial dependence in a collection of sample data means that observations at location i depend on other observations at locations $j \neq i$. Formally, we might state:

$$y_i = f(y_j), i = 1, \ldots, n \quad j \neq i \tag{1.1}$$

Note that we allow the dependence to be among several observations, as the index i can take on any value from $i = 1, \ldots, n$. Why would we expect sample data observed at one point in space to be dependent on values observed at other locations? There are two reasons commonly given. First, data collection of observations associated with spatial units such as zip-codes, counties, states, census tracts and so on, might reflect measurement error. This would occur if the administrative boundaries for collecting information do not accurately reflect the nature of the underlying process generating the sample data. As an example, consider the case of unemployment rates and labor force measures. Because laborers are mobile and can cross county or state lines to find employment in neighboring areas, labor force or unemployment rates measured on the basis of where people live could exhibit spatial dependence.

A second and perhaps more important reason we would expect spatial dependence is that the spatial dimension of socio-demographic, economic or regional activity may truly be an important aspect of a modeling problem. Regional science is based on the premise that location and distance are important forces at work in human geography and market activity. All of these notions have been formalized in regional science theory that relies on notions of spatial interaction and diffusion effects, hierarchies of place and spatial spillovers.

As a concrete example of this type of spatial dependence, we use a spatial data set on annual county-level counts of Gypsy moths established by the Michigan Department of Natural Resources (DNR) for the 68 counties in lower Michigan.

The North American gypsy moth infestation in the United States provides a classic example of a natural phenomena that is spatial in character. During 1981, the moths ate through 12 million acres of forest in 17 Northeastern states and Washington, DC. More recently, the moths have been spreading into the northern and eastern Midwest and to the Pacific Northwest. For example, in 1992 the Michigan Department of Agriculture estimated that more than 700,000 acres of forest land had experienced at least a 50% defoliation rate.

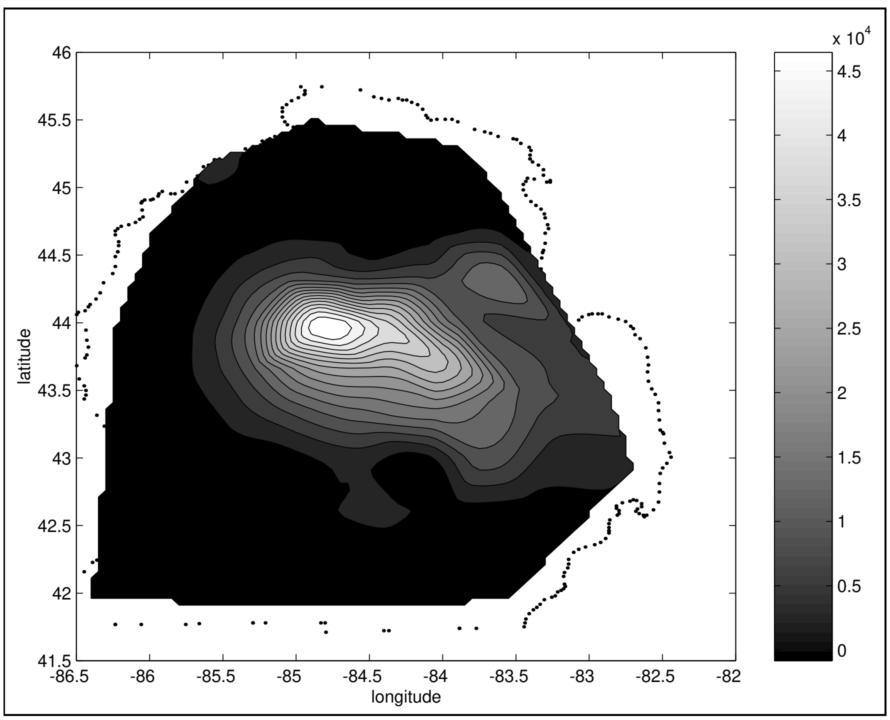

Figure 1.1: Gypsy moth counts in lower Michigan, 1991

Figure 1.1 shows a contour of the moth counts for 1991 overlayed on a map outline of lower Michigan. We see the highest level of moth counts near Midland county Michigan in the center. As we move outward from the center, lower levels of moth counts occur taking the form of concentric rings. A set of k data points $y_i, i = 1, \ldots, k$ taken from the same ring would exhibit a high correlation with each other. In terms of (1.1), y_i and y_j where both observations i and j come from the same ring should be highly correlated. The correlation of $k1$ points taken from one ring and $k2$ points from a neighboring ring should also exhibit a high correlation, but not as high as points sampled from the same ring. As we examine the correlation between points taken from more distant rings, we would expect the correlation to diminish.

Over time the Gypsy moths spread to neighboring areas. They cannot fly, so the diffusion should be relatively slow. Figure 1.2 shows a similarly constructed contour map of moth counts for the next year, 1992. We see some evidence of diffusion to neighboring areas between 1991 and 1992. The circular pattern of higher levels in the center and lower levels radiating out from the center is still quite evident.

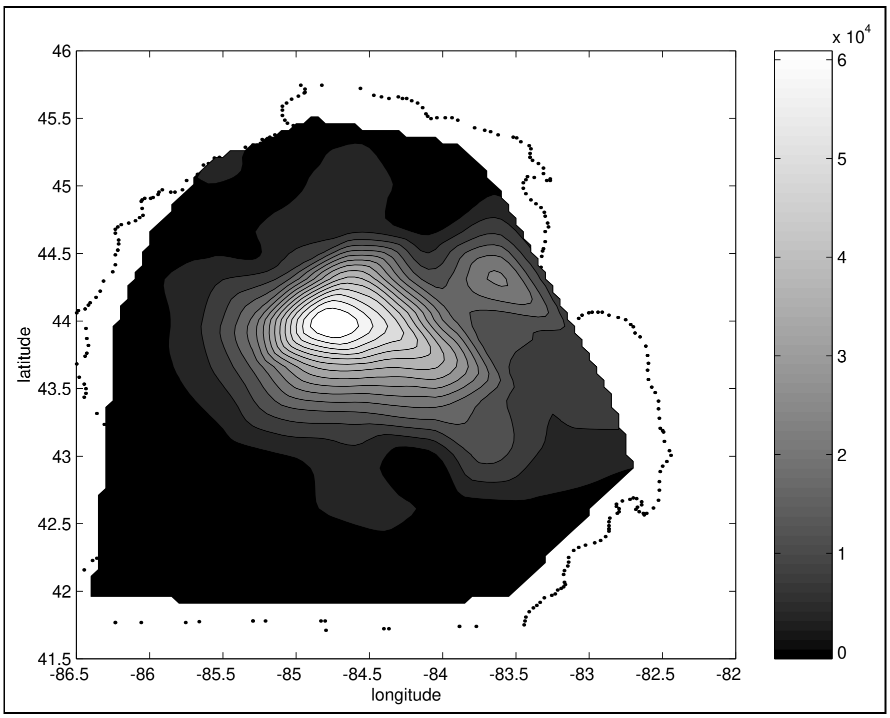

Figure 1.2: Gypsy moth counts in lower Michigan, 1992

Finally, Figure 1.3 shows a contour map of the moth count levels for 1993, where the diffusion has become more heterogeneous, departing from the circular shape in the earlier years. Despite the increasing heterogeneous nature of the moth count levels, neighboring points still exhibit high correlations. An adequate model to describe and predict Gypsy moth levels would require that the function $f()$ in (1.1) incorporate the notion of neighboring counties versus counties that are more distant.

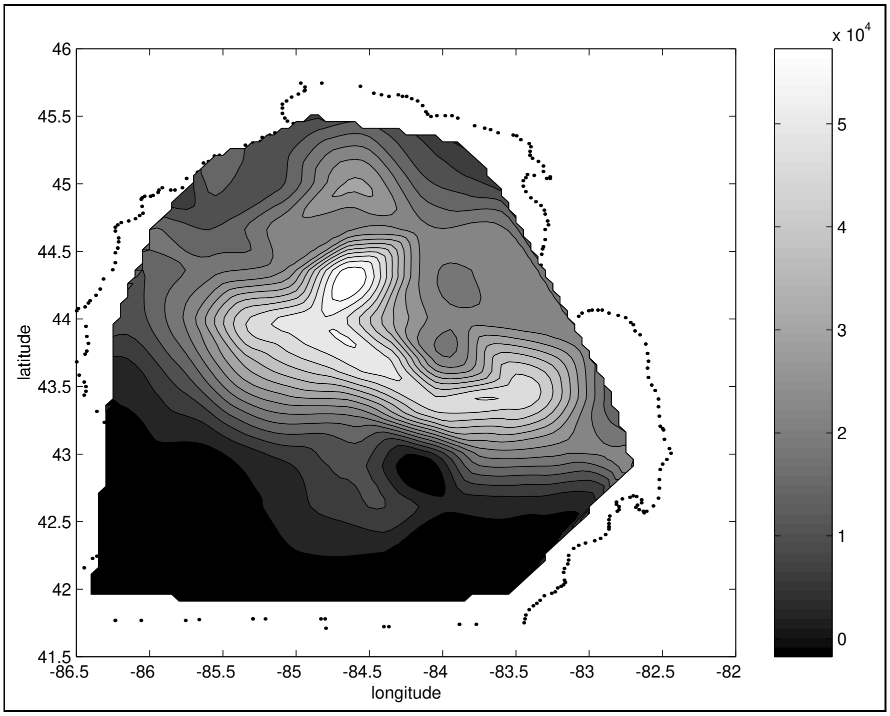

Figure 1.3: Gypsy moth counts in lower Michigan, 1993

How does this situation differ from the traditional view of the process at work to generate economic data samples? The Gauss-Markov view of a regression data sample is that the generating process takes the form of (1.2), where y represent a vector of n observations, X denotes an nxk matrix of explanatory variables, β is a vector of k parameters and ε is a vector of n stochastic disturbance terms.

$$y = X\beta + \varepsilon \tag{1.2}$$

The generating process is such that the X matrix and true parameters β are fixed while repeated disturbance vectors ε work to generate the samples y that we observe. Given that the matrix X and parameters β are fixed, the distribution of sample y vectors will have the same variance-covariance structure as ε. Additional assumptions regarding the nature of the variance-covariance structure of ε were invoked by Gauss-Markov to ensure that the distribution of individual observations in y exhibit a constant variance as we move across observations, and zero covariance between the observations.

It should be clear that observations from our sample of moth level counts do not obey this structure. As illustrated in Figures 1.1 to 1.3, observations from counties in concentric rings are highly correlated, with a decay of correlation as we move to observations from more distant rings.

Spatial dependence arising from underlying regional interactions in regional science data samples suggests the need to quantify and model the nature of the unspecified functional spatial dependence function $f()$, set forth in (1.1). Before turning attention to this task, the next section discusses the other underlying condition leading to a need for spatial econometrics — spatial heterogeneity.

1.3 Spatial heterogeneity

The term spatial heterogeneity refers to variation in relationships over space. In the most general case we might expect a different relationship to hold for every point in space. Formally, we write a linear relationship depicting this as:

$$y_i = X_i \beta_i + \varepsilon_i \tag{1.3}$$

Where i indexes observations collected at $i = 1, \ldots, n$ points in space, X_i represents a (1 x k) vector of explanatory variables with an associated set of parameters β_i, y_i is the dependent variable at observation (or location) i and ε_i denotes a stochastic disturbance in the linear relationship.

A slightly more complicated way of expressing this notion is to allow the function $f()$ from (1.1) to vary with the observation index i, that is:

$$y_i = f_i(X_i \beta_i + \varepsilon_i) \tag{1.4}$$

Restricting attention to the simpler formation in (1.3), we could not hope to estimate a set of n parameter vectors β_i given a sample of n data observations. We simply do not have enough sample data information with which to produce estimates for every point in space, a phenomena referred to as a “degrees of freedom” problem. To proceed with the analysis we need to provide a specification for variation over space. This specification must be parsimonious, that is, only a handful of parameters can be used in the specification. A large amount of spatial econometric research centers on alternative parsimonious specifications for modeling variation over space. Questions arise regarding: 1) how sensitive the inferences are to a particular specification regarding spatial variation?, 2) is the specification consistent with the sample data information?, 3) how do competing specifications perform and what inferences do they provide?, and a host of other issues that will be explored in this text.

One can also view the specification task as one of placing restrictions on the nature of variation in the relationship over space. For example, suppose we classified our spatial observations into urban and rural regions. We could then restrict our analysis to two relationships, one homogeneous across all urban observational units and another for the rural units. This raises a number of questions: 1) are two relations consistent with the data, or is there evidence to suggest more than two?, 2) is there a trade-off between efficiency in the estimates and the number of restrictions we use?, 3) are the estimates biased if the restrictions are inconsistent with the sample data information?, and other issues we will explore.

One of the compelling motivations for the use of Bayesian methods in spatial econometrics is their ability to impose restrictions that are stochastic rather than exact in nature. Bayesian methods allow us to impose restrictions with varying amounts of prior uncertainty. In the limit, as we impose a restriction with a great deal of certainty, the restriction becomes exact. Carrying out our econometric analysis with varying amounts of prior uncertainty regarding a restriction allows us to provide a continuous mapping of the restriction's impact on the estimation outcomes.

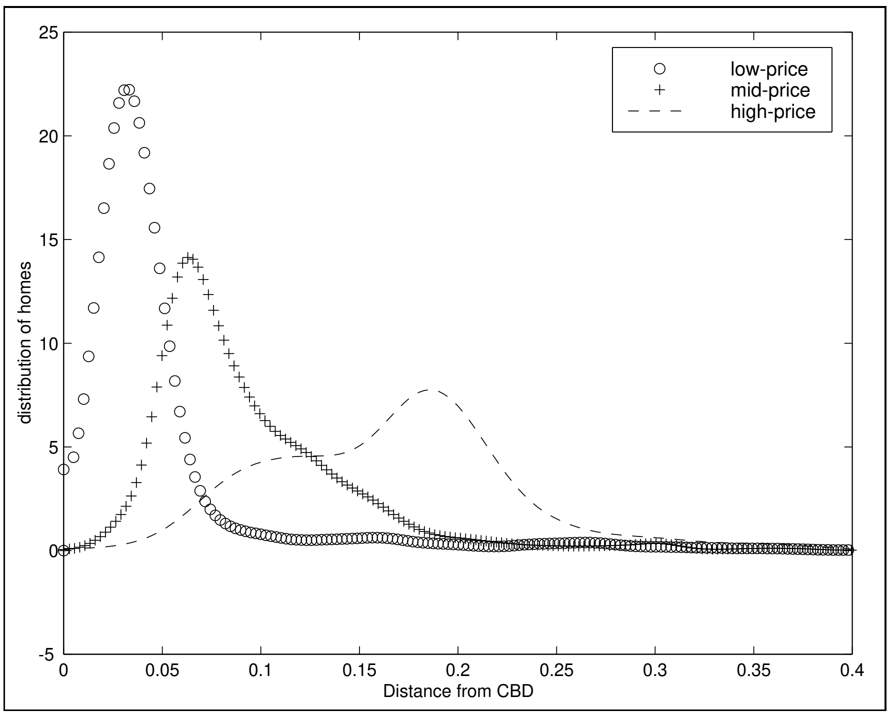

Figure 1.4: Distribution of low, medium and high priced homes versus distance

As a concrete illustration of spatial heterogeneity, we use a sample of 35,000 homes that sold within the last 5 years in Lucas county, Ohio. The selling prices were sorted from low to high and three samples of 5,000 homes were constructed. The 5,000 homes with the lowest selling prices were used to represent a sample of low-price homes. The 5,000 homes with selling prices that ranked from 15,001 to 20,000 in the sorted list were used to construct a sample of medium-price homes and the 5,000 highest selling prices from 30,0001 to 35,000 served as the basis for a high-price sample. It should be noted that the sample consisted of 35,702 homes, but the highest 702 selling prices were omitted from this exercise as they represent very high prices that are atypical.

Using the latitude-longitude coordinates, the distance from the central business district (CBD) in the city of Toledo, which is at the center of Lucas county was calculated. The three samples of 5,000 low, medium and high priced homes were used to estimate three empirical distributions that are graphed with respect to distance from the CBD in Figure 1.4.

We see three distinct distributions, with low-priced homes nearest to the CBD and high priced homes farthest away from the CBD. This suggests different relationships may be at work to describe home prices in different locations. Of course this is not surprising, numerous regional science theories exist to explain land usage patterns as a function of distance from the CBD. Nonetheless, these three distinct distributions provide a contrast to the Gauss-Markov assumption that the distribution of sample data exhibits a constant mean and variance as we move across the observations.

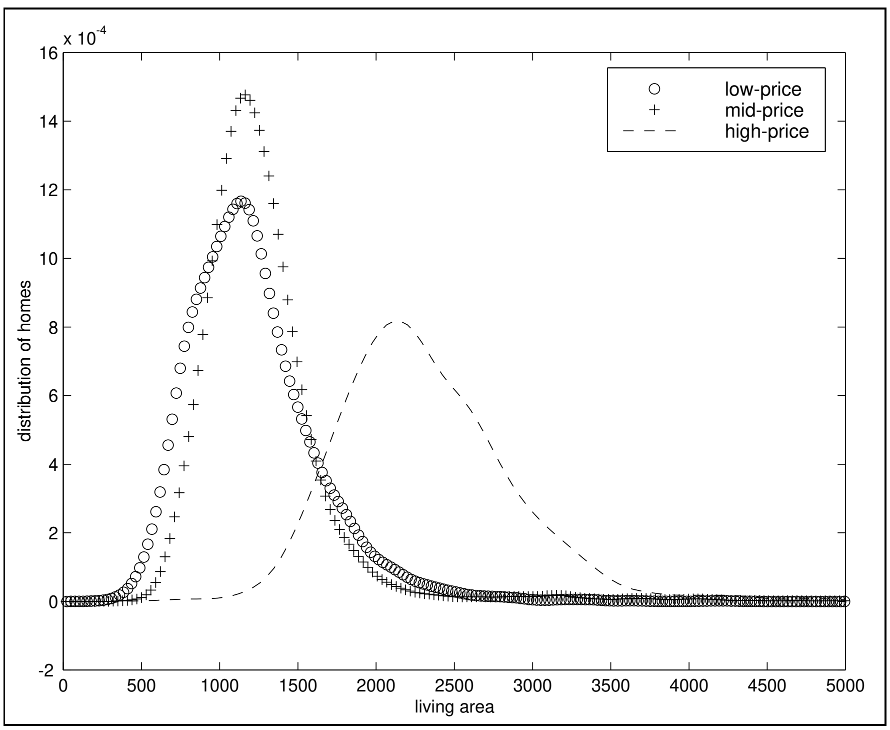

Figure 1.5: Distribution of low, medium and high priced homes versus living area

Another illustration of spatial heterogeneity is provided by three distributions for total square feet of living area of low, medium and high priced homes shown in Figure 1.5. Here we see only two distinct distributions, suggesting a pattern where the highest priced homes are the largest, but low and medium priced homes have roughly similar distributions with regard to living space.

It may be the case that important explanatory variables in the house value relationship change as we move over space. Living space may be unimportant in distinguishing between low and medium priced homes, but significant for higher priced homes. Distance from the CBD on the other hand appears to work well in distinguishing all three categories of house values.

1.4 Quantifying location in our models

A first task we must undertake before we can ask questions about spatial dependence and heterogeneity is quantification of the locational aspects of our sample data. Given that we can always map a set of spatial data observations, we have two sources of information on which to draw.

The location in Cartesian space represented by latitude and longitude is one source of information. This information would also allow us to calculate distances from any point in space, or the distance of observations located at distinct points in space to observations at other locations. Spatial dependence should conform to the fundamental theorem of regional science — distance matters. Observations that are near should reflect a greater degree of spatial dependence than those more distant from each other. This suggests the strength of spatial dependence between observations should decline with the distance between observations.

Distance might also be important for models involving spatially heterogeneous relationships. If the relationship we are modeling varies over space, observations that are near should exhibit similar relationships and those that are more distant may exhibit dissimilar relationships. In other words, the relationship may vary smoothly over space.

The second source of locational information is contiguity, reflecting the relative position in space of one regional unit of observation to other such units. Measures of contiguity rely on a knowledge of the size and shape of the observational units depicted on a map. From this, we can determine which units are neighbors (have borders that touch) or represent observational units in reasonable proximity to each other. Regarding spatial dependence, neighboring units should exhibit a higher degree of spatial dependence than units located far apart. For spatial heterogeneity, relationships may be similar for neighboring units.

It should be noted that these two types of information are not necessarily different. Given the latitude-longitude coordinates of an observation, we can construct a contiguity structure by defining a "neighboring observation" as one that lies within a certain distance. Consider also, given the boundary points associated with map regions, we can compute the centroid coordinates of the regions. These coordinates could then be used to calculate distances between the regions or observations.

We will illustrate how both types of locational information can be used in spatial econometric modeling. We first take up the issue of quantifying spatial contiguity, which is used in the models presented in Chapters 3 4 and 5. Chapters 6 and 7 deal with models that make direct use of the latitude-longitude coordinates, a subject discussed in the Section 1.4.2.

1.4.1 Quantifying spatial contiguity

Figure 1.6 shows a hypothetical example of five regions as they would appear on a map. We wish to construct a 5 by 5 binary matrix W containing 25 elements taking values of 0 or 1 that captures the notion of "connectiveness" between the five entities depicted in the map configuration. We record the contiguity relations for each region in the row of the matrix W. For example the matrix element in row 1, column 2 would record the presence (represented by a 1) or absence (denoted by 0) of a contiguity relationship between regions 1 and 2. As another example, the row 3, column 4 element would reflect the presence or absence of contiguity between regions 3 and 4. Of course, a matrix constructed in such fashion must be symmetric — if regions 3 and 4 are contiguous, so are regions 4 and 3.

It turns out there are a large number of ways to construct a matrix that contains contiguity information regarding the regions. Below, we enumerate some alternative ways to define a binary matrix W that reflects the "contiguity" relationships between the five entities in Figure 1.6. For the enumeration below, start with a matrix filled with zeros, then consider the following alternative ways to define the presence of a contiguity relationship.

Linear contiguity: Define $W_{ij} = 1$ for entities that share a common edge to the immediate right or left of the region of interest. For row 1, where we record the relations associated with region 1, we would have all $W_{1j} = 0, j = 1, \ldots, 5$. On the other hand, for row 5, where we record relationships involving region 5, we would have $W_{53} = 1$ and all other row-elements equal to zero.

Rook contiguity: Define $W_{ij} = 1$ for regions that share a common side with the region of interest. For row 1, reflecting region 1's relations we would have $W_{12} = 1$ with all other row elements equal to zero. As another example, row 3 would record $W_{34} = 1, W_{35} = 1$ and all other row elements equal to zero.

Bishop contiguity: Define $W_{ij} = 1$ for entities that share a common vertex with the region of interest. For region 2 we would have $W_{23} = 1$ and all other row elements equal to zero.

Double linear contiguity: For two entities to the immediate right or left of the region of interest, define $W_{ij} = 1$. This definition would produce the same results as linear contiguity for the regions in Figure 1.6.

Double rook contiguity: For two entities to the right, left, north and south of the region of interest define $W_{ij} = 1$. This would result in the same matrix W as rook contiguity for the regions shown in Figure 1.6.

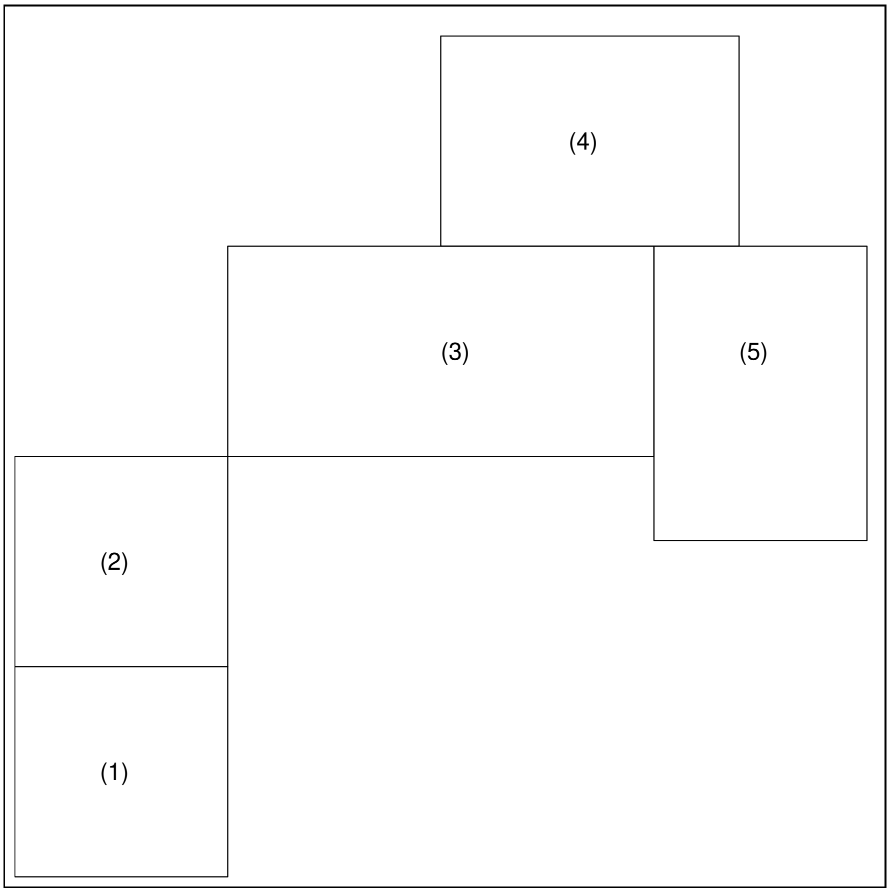

Figure 1.6: An illustration of contiguity

Queen contiguity: For entities that share a common side or vertex with the region of interest define $W_{ij} = 1$. For region 3 we would have: $W_{32} = 1, W_{34} = 1, W_{35} = 1$ and all other row elements zero.

There are of course other ways to proceed when defining a contiguity matrix. For a good discussion of these issues, see Appendix 1 of Kelejian and Robinson (1995). Note also that the double linear and double rook definitions are sometimes referred to as "second order" contiguity, whereas the other definitions are termed "first order". More elaborate definitions sometimes rely on the length of shared borders. This might impact whether we considered regions (4) and (5) in Figure 1.6 as contiguous or not. They have a common border, but it is very short. Note that in the case of a vertex, the rook definition rules out a contiguity relation, whereas the bishop and queen definitions would record a relationship.

The guiding principle is selecting a definition should be the nature of the problem being modeled, and perhaps additional non-sample information that is available. For example, suppose that a major highway connected regions (2) and (3) in Figure 1.6, and we knew that region (2) was a "bedroom community" for persons who work in region (3). Given this non-sample information, we would not rely on the rook definition because it rules out a contiguity relationship between these two regions.

We will use the rook definition to define a first-order contiguity matrix for the five regions in Figure 1.6 as a concrete illustration. This definition is often used in applied work. Perhaps the motivation for this is that we simply need to locate all regions on the map that have common borders with some positive length.

The matrix W in (1.5) shows first-order rook's contiguity relations for the five regions in Figure 1.6.

$$W = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{1.5}$$

Note that W is symmetric, and by convention the matrix always has zeros on the main diagonal. A transformation often used in applied work converts the matrix W to have row-sums of unity. A standardized version of W from (1.5) is shown in (1.6).

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} \tag{1.6}$$

The motivation for the standardization can be seen by considering matrix multiplication of C and a vector of observations y ona variable associated with the five regions. This matrix product, $y^{\star} = Cy$, represents a new variable equal to the mean of observations from contiguous regions as shown in (1.7).

$$\begin{aligned} \begin{pmatrix} y_1^{\star} \\ y_2^{\star} \\ y_3^{\star} \\ y_4^{\star} \\ y_5^{\star} \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} \\ \begin{pmatrix} y_1^{\star} \\ y_2^{\star} \\ y_3^{\star} \\ y_4^{\star} \\ y_5^{\star} \end{pmatrix} &= \begin{pmatrix} y_2 \\ y_1 \\ 1/2y_4 + 1/2y_5 \\ 1/2y_3 + 1/2y_5 \\ 1/2y_3 + 1/2y_4 \end{pmatrix} \end{aligned} \tag{1.7}$$

This is one way of quantifying the notion that $y_i = f(y_j), j \neq i$, expressed in (1.1). Equation (1.8) shows a linear relationship that uses the variable $y^\star$ from (1.7) as an explanatory variable for y in a cross-sectional spatial sample of observations.

$$y = \rho C y + \varepsilon \tag{1.8}$$

The scalar ρ represents a regression parameter to be estimated and ε denotes the stochastic disturbance in the relationship. The parameter ρ would reflect the spatial dependence inherent in our sample data, measuring the average influence of neighboring or contiguous observations on observations in the vector y. If we posit spatial dependence between the individual observations in the data sample y, some part of the total variation in y across the spatial sample would be explained by each observation's dependence on its neighbors. The parameter ρ would reflect this in the typical sense of regression. In addition, we could calculate the proportion of total variation in y explained by spatial dependence using $\hat{\rho}Cy$, where $\hat{\rho}$ is the estimated value of ρ.

We will examine spatial econometric models that rely on this type of formulation in Chapter 3 where we set forth maximum likelihood estimation procedures for a taxonomy of these models known as spatial autoregressive models. Anselin (1988) provided this taxonomy and devised maximum likelihood methods for producing estimates of these models. Chapter 4 provides a Bayesian approach to these models introduced by LeSage (1997) and Chapter 5 takes up limited dependent variable and censored data variants of these models from a Bayesian perspective that we introduce here. As this suggests, spatial autoregressive models have historically occupied a central place in spatial econometrics and they are likely to play an important role in the future.

One point to note is that traditional explanatory variables of the type encountered in regression can be added to the model in (1.8). We can represent these with the traditional matrix notation: $X\beta$, allowing us to modify (1.8) to take the form shown in (1.9).

$$y = \rho C y + X\beta + \varepsilon \tag{1.9}$$

Other extended specifications for these models will be taken up in Chapter 3.

1.4.2 Quantifying spatial position

Another approach to spatial econometric modeling makes direct use of the latitude-longitude coordinates associated with spatial data observations. A host of methods attempt to deal with spatial heterogeneity using locally linear regressions that are fit to sub-regions of space. Given that the relationship in our model varies over space, a locally linear model provides a parsimonious way to estimate multiple relationships that vary with regard to the spatial location of the observations. These models form the basis of our discussion in Chapter 6 where we examine these models from a maximum likelihood perspective and Chapter 7 where Bayesian variants are introduced. These models are also extended to the case of limited dependent variables.

Casetti (1972, 1992) introduced one approach that involves a method he labels "spatial expansion". The model is shown in (1.10), where y denotes an nx1 dependent variable vector associated with spatial observations and X is an nxnk matrix consisting of terms x_i representing kx1 explanatory variable vectors, as shown in (1.11). The locational information is recorded in the matrix Z which has elements $Z_{xi}, Z_{yi}, i = 1, \ldots, n$, that represent latitude and longitude coordinates of each observation as shown in (1.11).

The model posits that the parameters vary as a function of the latitude and longitude coordinates. The only parameters that need be estimated are the $2k$ parameters in β_0 that we denote, β_x, β_y. We note that the parameter vector β in (1.10) represents an nkx1 vector in this model containing parameter estimates for all k explanatory variables at every observation.

$$\begin{aligned} y &= X\beta + \varepsilon \\ \beta &= ZJ\beta_0 \end{aligned} \tag{1.10}$$

Where:

$$\begin{aligned} y &= \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad X = \begin{pmatrix} x_1' & 0 & \ldots & 0 \\ 0 & x_2' & & \\ \vdots & & \ddots & \\ 0 & & & x_n' \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \\ Z &= \begin{pmatrix} Z_{x1} \otimes I_k & Z_{y1} \otimes I_k & 0 & \ldots \\ 0 & \ddots & \ddots & \\ \vdots & & Z_{xn} \otimes I_k & Z_{yn} \otimes I_k \end{pmatrix} J = \begin{pmatrix} I_k & 0 \\ 0 & I_k \\ \vdots & \\ 0 & I_k \end{pmatrix} \\ \beta_0 &= \begin{pmatrix} \beta_x \\ \beta_y \end{pmatrix} \end{aligned} \tag{1.11}$$

Recall that there is a need to achieve a parsimonious representation that introduces only a few additional parameters to be estimated. This approach accomplishes this task by confining the estimated parameters to the $2k$ elements in β_x, β_y. This model can be estimated using least-squares to produce estimates of β_x and β_y. The remaining estimates for individual points in space are derived using $\hat{\beta}_x$ and $\hat{\beta}_y$ in the second equation of (1.10). This process is referred to as the "expansion process". To see this, substitute the second equation in (1.10) into the first, producing:

$$y = XZJ\beta_0 + \varepsilon \tag{1.12}$$

In (1.12) X, Z and J represent available sample data information or data observations and only the $2k$ parameters β_0 need be estimated.

This model would capture spatial heterogeneity by allowing variation in the underlying relationship such that clusters of nearby or neighboring observations measured by latitude-longitude coordinates take on similar parameter values. As the location varies, the regression relationship changes to accommodate a locally linear fit through clusters of observations in close proximity to one another.

Another approach to modeling variation over space is based on the non-parametric locally linear regression literature from exploratory statistics discussed in Becker, Chambers and Wilks (1988). In the spatial econometrics literature, McMillen (1996), McMillen and McDonald (1997) introduced these models and Brundson, Fotheringham and Charlton (1996) labeled these "geographically weighted regression" (GWR) models.

These models use locally weighted regressions to produce estimates for every point in space based on sub-samples of data information from nearby observations. Let y denote an nx1 vector of dependent variable observations collected at n points in space, X an nxk matrix of explanatory variables, and ε an nx1 vector of normally distributed, constant variance disturbances. Letting W_i represent an nxn diagonal matrix containing distance-based weights for observation i that reflects the distance between observation i and all other observations, we can write the GWR model as:

$$W_i y = W_i X \beta_i + \varepsilon_i \tag{1.13}$$

The subscript i on β_i indicates that this kx1 parameter vector is associated with observation i. The GWR model produces n such vectors of parameter estimates, one for each observation. These estimates are produced using least-squares regression on the sub-sample of observations as shown in (1.14).

$$\hat{\beta}_i = (X' W_i^2 X)^{-1} (X' W_i^2 y) \tag{1.14}$$

One confusing aspect of this notation is that $W_i y$ denotes an n-vector of distance-weighted observations used to produce estimates for observation i. The notation is confusing because we usually rely on subscripts to index scalar magnitudes representing individual elements of a vector. Note also, that $W_i X$ represents a distance-weighted data matrix, not a single observation and ε_i represents an n-vector.

The distance-based weights are specified as a decaying function of the distance between observation i and all other observations as shown in (1.15).

$$W_i = f(\theta, d_i) \tag{1.15}$$

The vector d_i contains distances between observation i and all other observations in the sample. The role of the parameter θ is to produce a decay of influence with distance. Changing the distance decay parameter θ results in a different weighting profile, which in turn produces estimates that vary more or less rapidly over space. Determination of the distance-decay parameter θ using cross-validation estimation methods is discussed in Chapter 5.

Again, note the use of a parsimonious parameterization of the spatially varying relationship. Only a single parameter, θ is introduced in the model. This along with the distance information can be used to produce a set of parameter estimates for every point in the spatial data sample.

It may have occurred to the reader that a homogeneous model fit to a spatial data sample that exhibits heterogeneity will produce residuals that exhibit spatial dependence. The residuals or errors made by a homogeneous model fit to a heterogeneous relationship should reflect unexplained variation attributable to heterogeneity in the underlying relationship over space.

Spatial clustering of the residuals would occur with positive and negative residuals appearing in distinct regions and patterns on the map. This of course was our motivation and illustration of spatial dependence as illustrated in Figure 1.1 showing the Gypsy moth counts in Michigan. You might infer correctly that spatial heterogeneity and dependence are often related in the context of modeling. An inappropriate model that fails to capture spatial heterogeneity will result in residuals that exhibit spatial dependence. This is another topic we discuss in this text.

1.4.3 Spatial lags

A fundamental concept that relates to spatial contiguity is the notion of a spatial lag operator. Spatial lags are analogous to the backshift operator B from time series analysis. This operator shifts observations back in time, where $By_t = y_{t-1}$, defines a first-order lag and $B^p y_t = y_{t-p}$ represents a pth order lag. In contrast to the time domain, spatial lag operators imply a shift over space but are restricted by some complications that arise when one tries to make analogies between the time and space domains.

Cressie (1991) points out that in the restrictive context of regular lattices or grids the spatial lag concept implies observations that are one or more distance units away from a given location, where distance units can be measured in two or four directions. In applied situations where observations are unlikely to represent a regular lattice or grid because they tend to be irregularly shaped map regions, the concept of a spatial lag relates to the set of neighbors associated with a particular location. The spatial lag operator works in this context to produce a weighted average of the neighboring observations.

In Section 1.4.1 we saw that the concept of "neighbors" in spatial analysis is not unambiguous, it depends on the definition used. By analogy to time series analysis it seems reasonable to simply raise our first-order binary contiguity matrix W containing 0 and 1 values to a power, say p to create a spatial lag. However, Blommestein (1985) points out that doing this produces circular or redundant routes, where he draws an analogy between binary contiguity and the graph theory notion of an adjacency matrix. If we use spatial lag matrices produced in this way in maximum likelihood estimation methods, spurious results can arise because of the circular or redundant routes created by this simplistic approach. Anselin and Smirnov (1994) provide details on many of the issues involved here.

For our purposes, we simply want to point out that an appropriate approach to creating spatial lags requires that the redundancies be eliminated from spatial

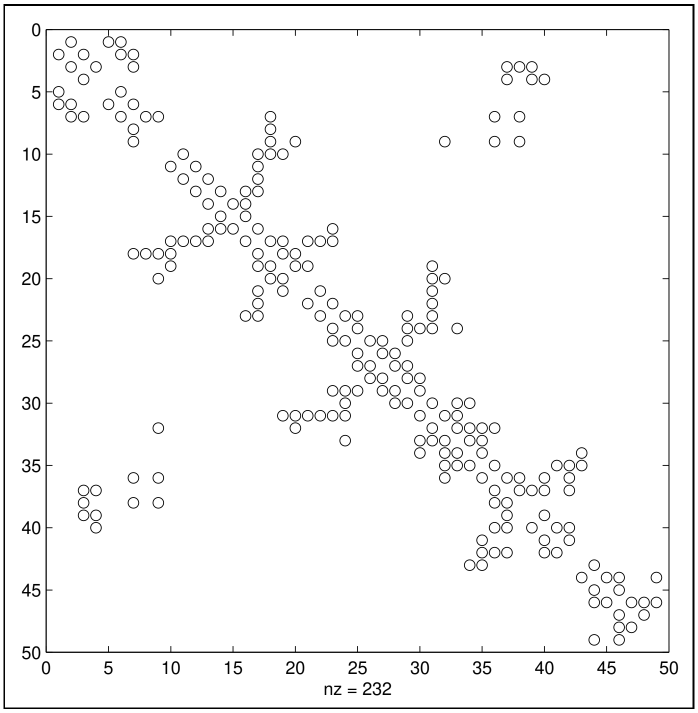

Figure 1.7: First-order spatial contiguity for 49 neighborhoods

weight matrices representing higher-order contiguity relationships. The spatial econometrics library contains a function to properly construct spatial lags of any order and the function deals with eliminating redundancies.

We provide a brief illustration of how spatial lags introduce information regarding "neighbors to neighbors" into our analysis. These spatial lags will be used in Chapter 3 when we discuss spatial autoregressive models.

To illustrate these ideas, we use a first-order contiguity matrix for a small data sample containing 49 neighborhoods in Columbus, Ohio taken from Anselin (1988). This contiguity matrix is typical of those encountered in applied practice as it relates irregularly shaped regions representing each neighborhood. Figure 1.7 shows the pattern of 0 and 1 values in a 49 by 49 grid. Recall that a non-zero entry in row i, column j denotes that neighborhoods i and j have borders that touch which we refer to as "neighbors". Of the 2401 possible elements in the 49 by 49 matrix, there are only 232 are non-zero elements designated on the axis in the figure by 'nz = 232'. These non-zero entries reflect the contiguity relations between the neighborhoods. The first-order contiguity matrix is symmetric which can be seen in the figure. This reflects the fact that if neighborhood i borders j, then j must also border i.

Figure 1.8 shows the original first-order contiguity matrix along with a

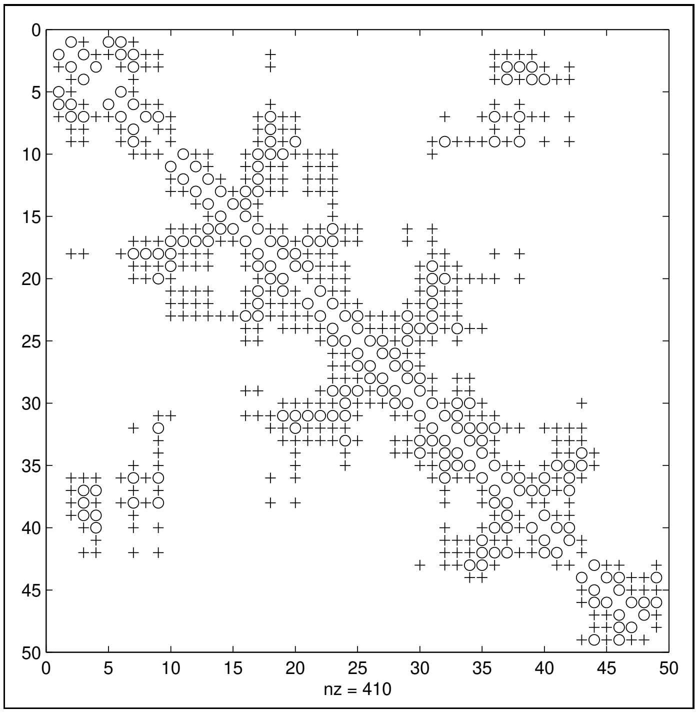

Figure 1.8: A second-order spatial lag matrix

second-order spatially lagged matrix, whose non-zero elements are represented by a '+' symbol in the figure. This graphical depiction of a spatial lag demonstrates that the spatial lag concept works to produce a contiguity or connectiveness structure that represents "neighbors of neighbors".

How might the notion of a spatial lag be useful in spatial econometric modeling? We might encounter a process where spatial diffusion effects are operating through time. Over time the initial impacts on neighbors work to influence more and more regions. The spreading impact might reasonably be considered to flow outward from neighbor to neighbor, and the spatial lag concept would capture this idea.

As an illustration of the redundancies produced by simply raising a first-order contiguity matrix to a higher power, Figure 1.9 shows a second-order spatial lag matrix created by simply powering the first-order matrix. The non-zero elements in this inappropriately generated spatial lag matrix are represented by '+' symbols with the original first-order non-zero elements denoted by 'o' symbols. We see that this second order spatial lag matrix contains 689 non-zero elements in contrast to only 410 for the correctly generated second order spatial lag matrix that eliminates the redundancies.

We will have occasion to use spatial lags in our examination of spatial au-

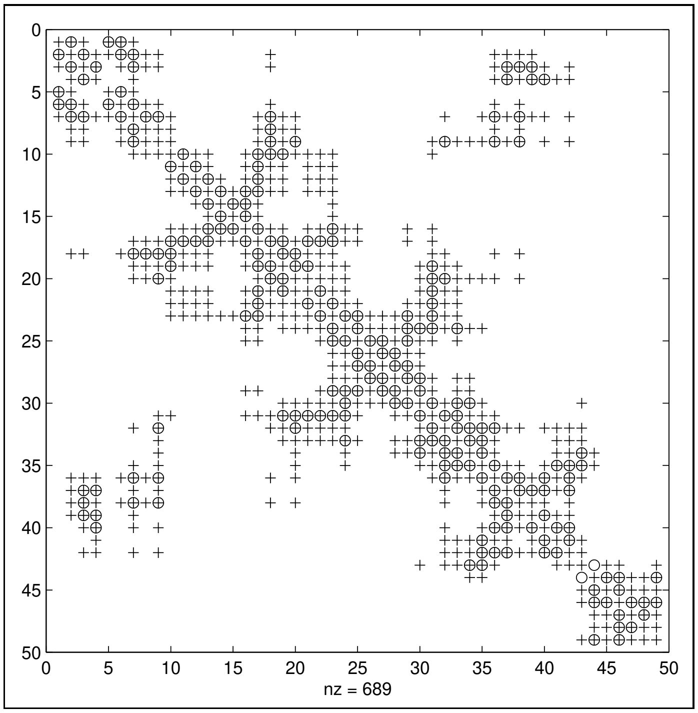

Figure 1.9: A contiguity matrix raised to a power 2

toregressive models in Chapters 3, 4 and 5. The MATLAB function from the spatial econometrics library as well as other functions for working with spatial contiguity matrices will be presented along with examples of their use in spatial econometric modeling.

1.5 Chapter Summary

This chapter introduced two main features of spatial econometric relationships, spatial dependence and spatial heterogeneity. Spatial dependence refers to the fact that sample data observations exhibit within-sample correlation with reference to the location of the sample observations in space. We often observe spatial clustering of sample data observations with respect to map regions. An intuitive motivation for this type of result is the existence of spatial hierarchical relationships, spatial spillovers and other types of spatial interactivity studied in regional science.

Spatial heterogeneity refers to the fact that spatial econometric relationships may vary systematically over space. This creates problems for traditional regression methods that assume a single constant relationship holds for the entire data sample. A host of methods have arisen in spatial econometrics that allow the estimated relationship to vary systematically over space. These methods attempt to achieve a parsimonious specification of systematic variation in the relationship such that only a few additional parameters need be estimated.

A large part of the chapter was devoted to introducing how locational information regarding sample data observations is formally incorporated in spatial econometric models. After introducing the concept of a spatial contiguity matrix, we provided a preview of spatial autoregressive models that rely on the contiguity concept. Chapters 3, and 4 cover this spatial econometric method in detail, and Chapter 5 extends this model to cases where the sample data represent limited dependent variables or variables subject to censoring.

In addition to spatial contiguity, other spatial econometric methods rely on latitude-longitude information to allow variation over space in the relationship being studied. Two approaches to this were introduced, the spatial expansion model and geographically weighted regression, which are the subject of Chapters 6 and 7.

Chapter 2

The MATLAB spatial econometrics library

As indicated in the preface to this text, all of the spatial econometric methods discussed in the text have been implemented using MATLAB software from the MathWorks Inc. All readers should read this chapter as it provides an introduction to the design philosophy that should be helpful to anyone using the functions. A consistent design was implemented that provides documentation, example programs, and functions to produce printed as well as graphical presentation of estimation results for all of the econometric functions. This was accomplished using the "structure variables" introduced in MATLAB Version 5. Information from econometric estimation is encapsulated into a single variable that contains "fields" for individual parameters and statistics related to the econometric results. A thoughtful design by the MathWorks allows these structure variables to contain scalar, vector, matrix, string, and even multi-dimensional matrices as fields. This allows the econometric functions to return a single structure that contains all estimation results. These structures can be passed to other functions that can intelligently decipher the information and provide a printed or graphical presentation of the results.

In Chapter 3 we will see our first example of constructing MATLAB functions to carry out spatial econometric estimation methods. Here, we discuss some design issues that affect all of the spatial econometric estimation functions and their use in the MATLAB software environment. The last section in this chapter discusses sparse matrices and functions that are used in the spatial econometrics library to achieve fast and efficient solutions for large problems with a minimum of computer memory.

2.1 Structure variables in MATLAB

In designing a spatial econometric library of functions, we need to think about organizing our functions to present a consistent user-interface that packages all of our MATLAB functions in a unified way. The advent of 'structures' in MATLAB version 5 allows us to create a host of alternative spatial econometric functions that return 'results structures'.

A structure in MATLAB allows the programmer to create a variable containing what MATLAB calls 'fields' that can be accessed by referencing the structure name plus a period and the field name. For example, suppose we have a MATLAB function to perform ordinary least-squares estimation named **ols** that returns a structure. The user can call the function with input arguments (a dependent variable vector y and explanatory variables matrix x) and provide a variable name for the structure that the **ols** function will return using:

```
result = ols(y,x);
```

The structure variable 'result' returned by our **ols** function might have fields named 'rsqr', 'tstat', 'beta', etc. These fields would contain the R-squared statistic, $t-$statistics for the $\hat{\beta}$ estimates and the least-squares estimates $\hat{\beta}$. One virtue of using the structure to return regression results is that the user can access individual fields in the structure that may be of interest as follows:

```
bhat = result.beta;
disp('The R-squared is:');
result.rsqr
disp('The 2nd t-statistic is:');
result.tstat(2,1)
```

There is nothing sacred about the name 'result' used for the returned structure in the above example, we could have used:

```
bill_clinton = ols(y,x);
result2      = ols(y,x);
restricted   = ols(y,x);
unrestricted = ols(y,x);
```

That is, the name of the structure to which the **ols** function returns its information is assigned by the user when calling the function.

To examine the nature of the structure in the variable 'result', we can simply type the structure name without a semi-colon and MATLAB will present information about the structure variable as follows:

```
result =
     meth: 'ols'
        y: [100x1 double]
     nobs: 100.00
     nvar: 3.00
     beta: [  3x1 double]
     yhat: [100x1 double]
    resid: [100x1 double]
     sige: 1.01
    tstat: [  3x1 double]
     rsqr: 0.74
     rbar: 0.73
       dw: 1.89
```

Each field of the structure is indicated, and for scalar components the value of the field is displayed. In the example above, 'nobs', 'nvar', 'sige', 'rsqr', 'rbar', and 'dw' are scalar fields, so there values are displayed. Matrix or vector fields are not displayed, but the size and type of the matrix or vector field is indicated. Scalar string arguments are displayed as illustrated by the 'meth' field which contains the string 'ols' indicating the regression method that was used to produce the structure. The contents of vector or matrix strings would not be displayed, just their size and type. Matrix and vector fields of the structure can be displayed or accessed using the MATLAB conventions of typing the matrix or vector name without a semi-colon. For example,

```
result.resid
result.y
```

would display the residual vector and the dependent variable vector y in the MATLAB command window.

Another virtue of using 'structures' to return results from our regression functions is that we can pass these structures to another related function that would print or plot the regression results. These related functions can query the structure they receive and intelligently decipher the 'meth' field to determine what type of regression results are being printed or plotted. For example, we could have a function **prt** that prints regression results and another **plt** that plots actual versus fitted and/or residuals. Both these functions take a structure returned by a regression function as input arguments. Example 2.1 provides a concrete illustration of these ideas.

The example assumes the existence of functions **ols**, **prt**, **plt** and data matrices y, x in files 'y.data' and 'x.data'. Given these, we carry out a regression, print results and plot the actual versus predicted as well as residuals with the MATLAB code shown in example 2.1. We will discuss the **prt** and **plt** functions in Section 2.2.

```
% ----- Example 1.1 Demonstrate regression using the ols() function
load y.data;
load x.data;
result = ols(y,x);
prt(result);
plt(result);
```

2.2 Constructing estimation functions

Now to put these ideas into practice, consider implementing an **ols** function. The function code would be stored in a file 'ols.m' whose first line is:

```
function results=ols(y,x)
```

The keyword 'function' instructs MATLAB that the code in the file 'ols.m' represents a callable MATLAB function.

The help portion of the MATLAB 'ols' function is presented below and follows immediately after the first line as shown. All lines containing the MATLAB comment symbol '%' will be displayed in the MATLAB command window when the user types 'help ols'.

```
function results=ols(y,x)
% PURPOSE: least-squares regression
%---------------------------------------------------
% USAGE: results = ols(y,x)
% where: y = dependent variable vector (nobs x 1)
%        x = independent variables matrix (nobs x nvar)
%---------------------------------------------------
% RETURNS: a structure
%        results.meth  = 'ols'
%        results.beta  = bhat
%        results.tstat = t-stats
%        results.yhat  = yhat
%        results.resid = residuals
%        results.sige  = e'*e/(n-k)
%        results.rsqr  = rsquared
%        results.rbar  = rbar-squared
%        results.dw    = Durbin-Watson Statistic
%        results.nobs  = nobs
%        results.nvar  = nvars
%        results.y     = y data vector
% --------------------------------------------------
% SEE ALSO: prt(results), plt(results)
%---------------------------------------------------

if (nargin ~= 2); error('Wrong # of arguments to ols');
else
 [nobs nvar] = size(x); nobs2 = length(y);
 if (nobs ~= nobs2); error('x and y not the same # obs in ols'); end;
end;
results.meth = 'ols';   results.y = y;
results.nobs = nobs;    results.nvar = nvar;
[q r] = qr(x,0);        xpxi = (r'*r)\eye(nvar);
results.beta = xpxi*(x'*y);
results.yhat = x*results.beta;
results.resid = y - results.yhat;
sigu = results.resid'*results.resid;
results.sige = sigu/(nobs-nvar);
tmp = (results.sige)*(diag(xpxi));
results.tstat = results.beta./(sqrt(tmp));
ym = y - mean(y);
rsqr1 = sigu; rsqr2 = ym'*ym;
results.rsqr = 1.0 - rsqr1/rsqr2; % r-squared
rsqr1 = rsqr1/(nobs-nvar); rsqr2 = rsqr2/(nobs-1.0);
results.rbar = 1 - (rsqr1/rsqr2); % rbar-squared
ediff = results.resid(2:nobs) - results.resid(1:nobs-1);
results.dw = (ediff'*ediff)/sigu; % durbin-watson
```

All functions in the spatial econometrics library present a unified documentation format for the MATLAB 'help' command by adhering to the convention of sections entitled, 'PURPOSE', 'USAGE', 'RETURNS', 'SEE ALSO', and perhaps a 'NOTES' and 'REFERENCES' section, delineated by dashed lines.

The 'USAGE' section describes how the function is used, with each input argument enumerated along with any default values. A 'RETURNS' section portrays the structure that is returned by the function and each of its fields. To keep the help information uncluttered, we assume some knowledge on the part of the user. For example, we assume the user realizes that the '.residuals' field would be an (nobs x 1) vector and the '.beta' field would consist of an (nvar x 1) vector.

The 'SEE ALSO' section points the user to related routines that may be useful. In the case of our **ols** function, the user might what to rely on the printing or plotting routines **prt** and **plt**, so these are indicated. The 'REFERENCES' section would be used to provide a literature reference (for the case of our more exotic spatial estimation procedures) where the user could read about the details of the estimation methodology. The 'NOTES' section usually contains important warnings or requirements for using the function. For example, some functions in the spatial econometrics library require that if the model includes a constant term, the first column of the data matrix should contain the constant term vector of ones. This information would be set forth in the 'NOTES' section. Other uses of this section would be to indicate that certain optional input arguments are mutually exclusive and should not be used together.

As an illustration of the consistency in documentation, consider the function **sar** that provides estimates for the spatial autoregressive model that we presented in Section 1.4.1. The documentation for this function is shown below. It would be printed to the MATLAB command window if the user typed 'help sar' in the command window.

```
 PURPOSE: computes spatial autoregressive model estimates
          y = p*W*y + X*b + e, using sparse matrix algorithms
---------------------------------------------------
 USAGE: results = sar(y,x,W,rmin,rmax,convg,maxit)
 where:  y = dependent variable vector
         x = explanatory variables matrix
         W = standardized contiguity matrix
      rmin = (optional) minimum value of rho to use in search
      rmax = (optional) maximum value of rho to use in search
     convg = (optional) convergence criterion (default = 1e-8)
     maxit = (optional) maximum # of iterations (default = 500)
---------------------------------------------------
 RETURNS: a structure
        results.meth  = 'sar'
        results.beta  = bhat
        results.rho   = rho
        results.tstat = asymp t-stat (last entry is rho)
        results.yhat  = yhat
        results.resid = residuals
        results.sige  = sige = (y-p*W*y-x*b)'*(y-p*W*y-x*b)/n
        results.rsqr  = rsquared
        results.rbar  = rbar-squared
        results.lik   = -log likelihood
        results.nobs  = # of observations
        results.nvar  = # of explanatory variables in x
        results.y     = y data vector
        results.iter  = # of iterations taken
```

```
        results.romax = 1/max eigenvalue of W (or rmax if input)
        results.romin = 1/min eigenvalue of W (or rmin if input)
  --------------------------------------------------
  SEE ALSO: prt(results), sac, sem, far
 ---------------------------------------------------
 REFERENCES: Anselin (1988), pages 180-182.
```

Now, we turn attention to the MATLAB code for estimating the ordinary least-squares model, which appears after the user documentation for the function. We begin processing the input arguments to carry out least-squares estimation based on a model involving y and x. First, we check for the correct number of input arguments using the MATLAB 'nargin' variable.

```
if (nargin ~= 2); error('Wrong # of arguments to ols');
else
 [nobs nvar] = size(x); [nobs2 junk] = size(y);
 if (nobs ~= nobs2); error('x and y not the same # obs in ols'); end;
end;
```

If we don't have two input arguments, the user has made an error which we indicate using the MATLAB **error** function. In the face of this error, the error message will be printed in the MATLAB command window and the **ols** function will return without processing any of the input arguments. Another error check involves the number of rows in the y vector and x matrix which should be equal. We use the MATLAB **size** function to implement this check in the code above.

Assuming that the user provided two input arguments, and the number of rows in x and y are the same, we can proceed to use the input information to carry out a regression.

The 'nobs' and 'nvar' returned by the MATLAB **size** function are pieces of information that we promised to return in our results structure, so we construct these fields using a '.nobs' and '.nvar' appended to the 'results' variable specified in the function declaration. We also fill in the 'meth' field and the 'y' vector fields.

```
results.meth = 'ols'; results.y = y;
results.nobs = nobs;  results.nvar = nvar;
```

The decision to return the actual y data vector was made to facilitate the **plt** function that will plot the actual versus predicted values from the regression along with the residuals. Having the y data vector in the structure makes it easy to call the **plt** function with only the structure returned by a regression function.

We proceed to estimate the least-squares coefficients $\hat{\beta} = (X'X)^{-1}X'y$, which we solve using the QR matrix decomposition. A first point to note is that we require more than a simple solution for $\hat{\beta}$, because we need to calculate $t-$statistics for the $\hat{\beta}$ estimates. This requires that we compute $(X'X)^{-1}$ which is done using the MATLAB 'slash' operator to invert the $(X'X)$ matrix. We represent $(X'X)$ using $(r'r)$, where r is an upper triangular matrix returned by the QR decomposition.

```
[q r] = qr(x,0);
xpxi = (r'*r)\eye(nvar);
results.beta = xpxi*(x'*y);
```

Given these solutions, we are in a position to use our estimates $\hat{\beta}$ to compute the remaining elements of the **ols** function results structure. We add these elements to the structure in the '.yhat, .resid, .sige,' etc., fields.

```
results.yhat = x*results.beta;
results.resid = y - results.yhat;
sigu = results.resid'*results.resid;
results.sige = sigu/(nobs-nvar);
tmp = (results.sige)*(diag(xpxi));
results.tstat = results.beta./(sqrt(tmp));
ym = y - mean(y);
rsqr1 = sigu; rsqr2 = ym'*ym;
results.rsqr = 1.0 - rsqr1/rsqr2; % r-squared
rsqr1 = rsqr1/(nobs-nvar);
rsqr2 = rsqr2/(nobs-1.0);
results.rbar = 1 - (rsqr1/rsqr2); % rbar-squared
ediff = results.resid(2:nobs) - results.resid(1:nobs-1);
results.dw = (ediff'*ediff)/sigu; % durbin-watson
```

2.3 Using the results structure

To illustrate the use of the 'results' structure returned by our **ols** function, consider the associated function **plt_reg** which plots actual versus predicted values along with the residuals. The results structure contains everything needed by the **plt_reg** function to carry out its task. Earlier, we referred to functions **plt** and **prt** rather than **plt_reg**, but **prt** and **plt** are "wrapper" functions that call the functions **prt_reg** and **plt_reg** where the real work of printing and plotting regression results is carried out. The motivation for taking this approach is that separate smaller functions can be devised to print and plot results from all of the spatial econometric procedures, making it easier to develop new estimation methods. The wrapper functions eliminate the need for the user to learn the names of different printing and plotting functions associated with each group of spatial econometric procedures — all results structures can be printed and plotted by simply invoking the **prt** and **plt** functions. All spatial econometric results are printed by: **prt_cas**, **prt_spat**, or **prt_gwr** and all plotting of results is carried out by the function **plt_spat**. Although you need not remember these function names, you may wish to examine the functions to see what information is being printed, change the information being printed, or add new printing methods for additional functions you construct.

A portion of the **plt_spat** function is shown below. This function is also called by the wrapper function **plt**, so we need not remember the name **plt_spat**. A check that the user supplied a regression results structure can be carried out using the MATLAB **isstruct** function that is true if the argument represents a structure. After this error check, we rely on a MATLAB programming construct called the 'switch-case' to provide the remaining error checking for the function.

```
function plt_spat(results,vnames)
% PURPOSE: Plots output using spatial regression results structures
%---------------------------------------------------
% USAGE: plt_spat(results,vnames)
% Where: results = a structure returned by a spatial regression
%        vnames  = an optional vector of variable names
%                  e.g. vnames = strvcat('y','constant','x1','x2');
%---------------------------------------------------
%  RETURNS: nothing, just plots the spatial regression results
% --------------------------------------------------
% NOTE: user must supply pause commands, none are in plt_spat function
%       e.g. plt_spat(results);
%            pause;
%            plt_spat(results2);
% --------------------------------------------------
% SEE ALSO: prt, plt
%---------------------------------------------------

if ~isstruct(results)
 error('plt_spat requires structure argument');
end;
nobs = results.nobs;
switch results.meth

case {'sac','sar','far','sem'}
tt=1:nobs;
clf; % plot actual vs predicted and residuals
subplot(2,1,1), plot(tt,results.y,'-',tt,results.yhat,'--');
title([upper(results.meth), '   Actual vs. Predicted']);
subplot(2,1,2), plot(tt,results.resid)
title('Residuals');

case {'casetti'}
.
.
.
otherwise
error('method not recognized by plt_spat');
end;
```

The 'switch' statement examines the 'meth' field of the results structure passed to the **plt_spat** function and executes the plotting commands if the 'meth' field is one of the spatial methods implemented in our function library. In the event that the user passed a result structure from a function other than one of our spatial functions, the 'otherwise' statement is executed which prints an error message.

The switch statement also helps us to distinguish the special case of the **casetti** spatial expansion method where the estimated parameters are plotted for every point in space. A similar approach was used to extend the **plt_spat** function to accommodate other special spatial estimation functions where additional or specialized plots were desired.

A decision was made not to place the 'pause' command in the **plt** function, but rather let the user place this statement in the calling program or function. An implication of this is that the user controls viewing regression plots in 'for loops', or in the case of multiple invocations of the **plt** function. For example, only the second 'plot' will be shown in the following code.

```
result1 = sar(y,x1,W);
plt(result1);
result2 = sar(y,x2,W);
plt(result2);
```

If the user wishes to see the spatial autoregressive model plots associated with the first model, the code would need to be modified as follows:

```
result1 = sar(y,x1,W);
plt(result1);
pause;
result2 = sar(y,x2,W);
plt(result2);
```

The 'pause' statement would force a plot of the results from the first spatial autoregression and wait for the user to strike any key before proceeding with the second regression and accompanying plot of these results.

Our **plt_spat** function would work with new spatial estimation functions that we add to the library provided that the function returns a structure containing the fields '.y', '.yhat', '.nobs' and '.meth'. We need simply add this method to the switch-case statement.

A more detailed example of using the results structure is **prt_spat** that provides a formatted printout of estimation results. The function relies on the 'meth' field to determine what type of estimation results are being printed, and uses the 'switch-case' statement to implement specialized methods for different types of models.

A small fragment of the **prt_spat** function showing the specialized printing for the **sar** and **far** spatial autoregression estimation methods is presented below:

```
function prt_spat(results,vnames,fid)
% PURPOSE: Prints output using spatial econometric results structures
%---------------------------------------------------
% USAGE: prt_spat(results,vnames,fid)
% Where: results = a structure returned by a spatial regression
%        vnames  = an optional vector of variable names
%        fid     = optional file-id for printing results to a file
%                  (defaults to the MATLAB command window)
%---------------------------------------------------
%  NOTES: e.g. vnames = strvcat('y','const','x1','x2');
%          e.g. fid = fopen('ols.out','wr');
%  use prt_spat(results,[],fid) to print to a file with no vnames
% --------------------------------------------------
%  RETURNS: nothing, just prints the regression results
% --------------------------------------------------
% SEE ALSO: prt, plt
%---------------------------------------------------

if ~isstruct(results) error('prt_spat requires structure argument');
elseif nargin == 1, nflag = 0; fid = 1;
```

```
elseif nargin == 2, fid = 1; nflag = 1;
elseif nargin == 3, nflag = 0;
 [vsize junk] = size(vnames); % user may supply a blank argument
   if vsize > 0, nflag = 1; end;
else, error('Wrong # of arguments to prt_spat');
end;
nvar = results.nvar; nobs = results.nobs;
Vname = 'Variable'; % handling of vnames
 for i=1:nvar
    tmp = ['variable ',num2str(i)]; Vname = strvcat(Vname,tmp);
 end;
Vname = strvcat(Vname,'rho'); % add spatial rho parameter name
if (nflag == 1) % the user supplied variable names
Vname = 'Variable'; [tst_n nsize] = size(vnames);
 if tst_n ~= nvar+1
 warning('Wrong # of variable names in prt_spat -- check vnames argument');
 fprintf(fid,'will use generic variable names \n'); nflag = 0;
 else,
 for i=1:nvar; Vname = strvcat(Vname,vnames(i+1,:)); end;
    Vname = strvcat(Vname,'rho'); % add spatial rho parameter name
 end; % end of if-else
end; % end of nflag issue
switch results.meth
case {'sar'} % <=================== spatial autoregressive model
fprintf(fid,'\n'); fprintf(fid,'Spatial autoregressive Model Estimates \n');
if (nflag == 1)
fprintf(fid,'Dependent Variable = %16s \n',vnames(1,:));
end;
fprintf(fid,'R-squared       = %9.4f \n',results.rsqr);
fprintf(fid,'Rbar-squared    = %9.4f \n',results.rbar);
fprintf(fid,'sigma^2         = %9.4f \n',results.sige);
fprintf(fid,'log-likelihood  = %16.8g \n',results.lik);
fprintf(fid,'Nobs, Nvars     = %6d,%6d \n',results.nobs,results.nvar-1);
fprintf(fid,'# of iterations = %6d   \n',results.iter);
fprintf(fid,'min and max rho = %9.4f,%9.4f \n',results.romin,results.romax);
fprintf(fid,'*******************************************************\n');
bout = [results.beta
        results.rho];
nvar = results.nvar; nobs = results.nobs;

case {'far'} % <=================== first-order autoregressive model
.
.
.
otherwise
error('results structure not known by prt_spat function');
end;

% now print coefficient estimates, t-statistics and probabilities
tout = norm_prb(results.tstat); % find Aymptotic t-stat z-probabilities
tmp = [bout results.tstat tout];  % matrix to be printed
% column labels for printing results
bstring = 'Coefficient'; tstring = 'Asymptot t-stat'; pstring = 'z-probability';
cnames = strvcat(bstring,tstring,pstring);
in.cnames = cnames; in.rnames = Vname; in.fmt = '%16.6f'; in.fid = fid;
mprint(tmp,in);
```

The function **mprint** is a utility function from the *Econometrics Toolbox* to produce formatted printing of a matrix with column and row-labels. All printing of matrix results for the library of spatial econometric functions is done using the **mprint** function.

The **prt_spat** function allows the user an option of providing a vector of fixed width variable name strings that will be used when printing the regression coefficients. These can be created using the MATLAB **strvcat** function that produces a vertical concatenated list of strings with fixed width equal to the longest string in the list. We can also print results to an indicated file rather than the MATLAB command window. Three alternative invocations of the **prt_spat** (or **prt**) function illustrating these options are shown below:

```
vnames = strvcat('crime','const','income','house value');
res = sar(y,x,W);
prt(res);                     % print with generic variable names
prt(res,vnames);              % print with user-supplied variable names
fid = fopen('sar.out','wr'); % open a file for printing
prt(res,vnames,fid);          % print results to file `sar.out'
```

The first use of **prt** produces a printout of results to the MATLAB command window that uses 'generic' variable names:

```
Spatial autoregressive Model Estimates
R-squared       =    0.6518
Rbar-squared    =    0.6366
sigma^2         =   95.5033
Nobs, Nvars     =     49,     3
log-likelihood  =       -165.41269
# of iterations =     12
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable        Coefficient      Asmptot t-stat    z-probability
variable 1        45.056481         6.186275        0.000000
variable 2        -1.030647        -3.369256        0.001533
variable 3        -0.265970        -3.004718        0.004293
rho                0.431377         3.587351        0.000806
```

The second use of **prt** uses the user-supplied variable names. The MATLAB function **strvcat** carries out a vertical concatenation of strings and pads the shorter strings in the 'vnames' vector to have a fixed width based on the longer strings. A fixed width string containing the variable names is required by the **prt** function. Note that we could have used:

```
    vnames = ['crime      ',
              'const      ',
              'income     ',
              'house value'];
```

but, this takes up more space and is slightly less convenient as we have to provide the padding of strings ourselves. Using the 'vnames' input in the **prt** function would produce the following output in the MATLAB command window.

```
Spatial autoregressive Model Estimates
Dependent Variable =      crime
```

```
R-squared       =    0.6518
Rbar-squared    =    0.6366
sigma^2         =   95.5033
Nobs, Nvars     =     49,     3
log-likelihood  =       -165.41269
# of iterations =     12
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable        Coefficient      Asymptot t-stat   z-probability
const             45.056481         6.186275        0.000000
income            -1.030647        -3.369256        0.001533
house value       -0.265970        -3.004718        0.004293
rho                0.431377         3.587351        0.000806
```

The third case specifies an output file opened with the command:

```
fid = fopen('sar.out','wr');
```

The file 'sar.out' would contain output identical to that from the second use of **prt**. It is the user's responsibility to close the file that was opened using the MATLAB command:

```
fclose(fid);
```

Next, we turn to details concerning implementation of the **prt_spat** function. The initial code does error checking on the number of input arguments, determines if the user has supplied a structure argument, and checks for variable names and/or an output file id. We allow the user to provide a file id argument with no variable names using the call: prt_spat(result,[],fid) where a blank argument is supplied for the variable names. We check for this case by examining the size of the vnames input argument under the case of nargin == 3 in the code shown below.

```
if ~isstruct(results) error('prt_spat requires structure argument');
elseif nargin == 1, nflag = 0; fid = 1;
elseif nargin == 2, fid = 1; nflag = 1;
elseif nargin == 3, nflag = 0;
 [vsize junk] = size(vnames); % user may supply a blank argument
   if vsize > 0, nflag = 1; end;
else, error('Wrong # of arguments to prt_spat');
end;
```

Variable names are constructed and placed in a MATLAB fixed-width string-array named 'Vname', with the first name in the array being the row-label heading 'Variable' used by the function **mprint**. For the case where the user supplied variable names, we simply transfer these to a MATLAB 'string-array' named 'Vname', again with the first element 'Variable' used by **mprint**. We do error checking on the number of variable names supplied which should equal the number of explanatory variables plus the dependent variable (nvar+1). If the user supplied the wrong number of variable names, we print a warning and switch to reliance on the generic variable names. This decision was made because it is often the case that students working with a data set know the variable names and would prefer to see printed output with generic names than receive an error message and no printed output.

```
nvar = results.nvar; nobs = results.nobs;
Vname = 'Variable'; % handling of vnames
 for i=1:nvar
    tmp = ['variable ',num2str(i)]; Vname = strvcat(Vname,tmp);
 end;
Vname = strvcat(Vname,'rho'); % add spatial rho parameter name
if (nflag == 1) % the user supplied variable names
Vname = 'Variable'; [tst_n nsize] = size(vnames);
 if tst_n ~= nvar+1
 warning('Wrong # of variable names in prt_spat -- check vnames argument');
 fprintf(fid,'will use generic variable names \n'); nflag = 0;
 else,
 for i=1:nvar; Vname = strvcat(Vname,vnames(i+1,:)); end;
    Vname = strvcat(Vname,'rho'); % add spatial rho parameter name
 end; % end of if-else
end; % end of nflag issue
```

After constructing variable names, the 'switch-case' takes over sending our function to the appropriate customized segment of code for printing part of the spatial estimation results depending on the 'meth' field of the results structure.

```
switch results.meth
case {'sar'} % <=================== spatial autoregressive model
fprintf(fid,'\n'); fprintf(fid,'Spatial autoregressive Model Estimates \n');
if (nflag == 1) fprintf(fid,'Dependent Variable = %16s \n',vnames(1,:)); end;
fprintf(fid,'R-squared       = %9.4f \n',results.rsqr);
fprintf(fid,'Rbar-squared    = %9.4f \n',results.rbar);
fprintf(fid,'sigma^2         = %9.4f \n',results.sige);
fprintf(fid,'log-likelihood  = %16.8g \n',results.lik);
fprintf(fid,'Nobs, Nvars     = %6d,%6d \n',results.nobs,results.nvar-1);
fprintf(fid,'# of iterations = %6d   \n',results.iter);
fprintf(fid,'min and max rho = %9.4f,%9.4f \n',results.romin,results.romax);
fprintf(fid,'*******************************************************\n');
bout = [results.beta
        results.rho];
nvar = results.nvar; nobs = results.nobs;
```

A point to note is that use of the MATLAB 'fprintf' command with an input argument 'fid' makes it easy to handle both the case where the user wishes output printed to the MATLAB command window or to an output file. The 'fid' argument takes on a value of '1' to print to the command window and a user-supplied file name value for output printed to a file.

Finally, after printing the specialized output, the coefficient estimates, t-statistics and marginal z-probabilities that are in common to all spatial regressions are printed. The marginal probabilities are calculated using a function **norm_prb** that determines these probabilities given a vector of asymptotic $t-$ratios that are normally distributed random variates. This function is part of the distributions library in the *Econometrics Toolbox*, which illustrates how functions in the spatial econometrics library rely on the broader set of functions included in the *Econometrics Toolbox*.

The code to print coefficient estimates, asymptotic t-statistics and marginal probabilities is common to all regression printing procedures, so it makes sense to move it to the end of the 'switch-case' code and execute it once as shown below. We rely on the function **mprint** to do the actual printing of the matrix of regression results with row and column labels specified as fields of a structure variable 'in'. Use of structure variables with fields as input arguments to functions is a convenient way to pass a large number of optional arguments to MATLAB functions, an approach we will rely on in other functions we develop in the spatial econometrics library. Again, the function **mprint** is part of the *Econometrics Toolbox* and is used by all printing routines in the toolbox to print matrices formatted with row and column labels.

```
tout = norm_prb(results.tstat); % find t-stat z-probabilities
tmp = [bout results.tstat tout];  % matrix to be printed
% column labels for printing results
bstring = 'Coefficient'; tstring = 'Asymptot t-stat'; pstring = 'z-probability';
cnames = strvcat(bstring,tstring,pstring);
in.cnames = cnames; in.rnames = Vname;
in.fmt = '%16.6f'; in.fid = fid;
mprint(tmp,in); % print estimates, t-statistics and probabilities
```

In the following chapters that present various spatial estimation methods, we will provide details concerning computational implementation of the estimation procedures in MATLAB. A function exists in the spatial econometrics library to implement each estimation procedure. Our focus is on the estimation functions, but of course there are accompanying functions to provide printed and graphical presentation of the results. These functions follow the format and methods described in this section, so we don't provide details. Of course, you will see the format of printed and graphical results when we provide applied examples, and you're free to examine the code that produces printed and graphical output.

2.4 Sparse matrices in MATLAB

Sparse matrices are those that contain a large proportion of zeros. In spatial econometrics the first-order contiguity matrix W discussed in Chapter 1 represents a sparse matrix.

A first point to note regarding sparsity is that large problems involving thousands of spatial observations will inevitably involve a sparse spatial contiguity weighting matrix. We will provide an illustration in Chapter 3 that uses a sample of 3,107 U.S. counties. When you consider the first-order contiguity structure of this sample, individual counties exhibited at most 8 first-order (rook definition) contiguity relations. This means that the remaining 2,999 entries in this row of W are zero. The average number of contiguity relationships between the sample of counties was 4, so a great many of the elements in the matrix W are zero, which is the definition of a sparse matrix.

To understand how sparse matrix algorithms conserve on storage space and computer memory, consider that we need only record the non-zero elements of a sparse matrix for storage. Since these represent a small fraction of the total 3107x3107 = 9,653,449 elements in our example weight matrix, we save a tremendous amount of computer memory. In fact for this case of 3,107 counties, only 12,429 non-zero elements were found in the first-order spatial contiguity matrix, representing a very small fraction (far less than 1 percent) of the total elements.

MATLAB provides a function **sparse** that can be used to construct a large sparse matrix by simply indicating the row and column positions of non-zero elements and the value of the matrix element for these non-zero row and column elements. Continuing with our county data example, we could store the first-order contiguity matrix in a single data file containing 12,429 rows with 3 columns that take the form:

```
row column value
```

This represents a considerable savings in computational space when compared to storing a matrix containing 9,653,449 elements. A handy utility function in MATLAB is **spy** which allows one to produce a specially formatted graph showing the sparsity structure associated with sparse matrices. We demonstrate by executing **spy(W)** on our weight matrix W from the Pace and Barry data set, which produced the graph shown in Figure 2.1. As we can see from the figure, most of the non-zero elements reside near the diagonal.

An interesting point about Figure 2.1 is that the blocks along the diagonal represent states and the off-diagonal scatters reflect counties on the borders of states.

As an example of storing a sparse first-order contiguity matrix, consider example 2.1 below that reads data from the file 'ford.dat' in sparse format and uses the function **sparse** to construct a working spatial contiguity matrix W. The example also produces a graphical display of the sparsity structure using the MATLAB function **spy**.

```
% ----- Example 2.1 Using sparse matrix functions
load ford.dat; % 1st order contiguity matrix
               % stored in sparse matrix form
ii = ford(:,1);
jj = ford(:,2);
ss = ford(:,3);
clear ford;                   % clear out the matrix to save RAM memory
W = sparse(ii,jj,ss,3107,3107);
clear ii; clear jj; clear ss; % clear out these vectors to save memory
spy(W);
```

MATLAB does not automatically treat a matrix as sparse, users must provide a declaration using the **sparse** command. If we know a matrix is sparse and want to use this to our advantage when executing matrix operations, we convert the matrix to sparse form using: 'b = **sparse**(a)'. Subsequent MATLAB matrix operations such as multiplication, addition, subtraction, division and the MATLAB 'slash' operator (discussed below) involving 'b' and other sparse matrices will produce sparse matrix results. That is, operating on two sparse matrices will create a new sparse matrix as a result. Other operations that produce sparse matrix results when given a sparse matrix argument are

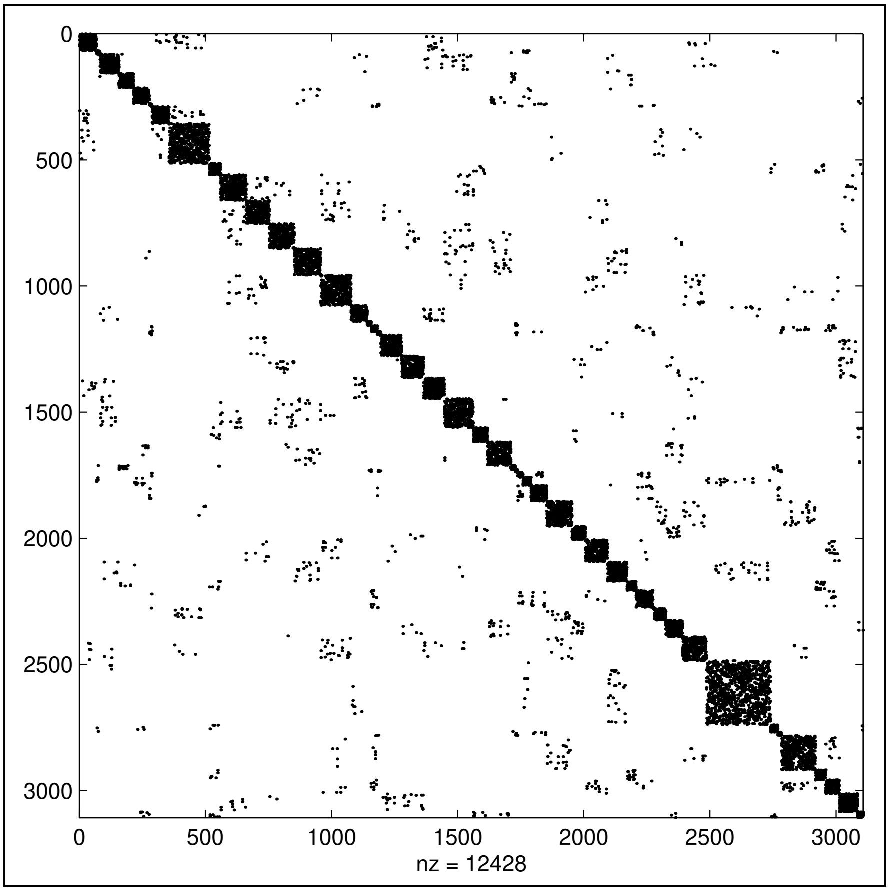

Figure 2.1: Sparsity structure of W from Pace and Barry

the cholesky decomposition **chol**, the householder QR decomposition **qr** and the LU decomposition **lu**.

If one matrix in a binary operation is sparse and the other a full matrix, the result will be a full matrix. When you are uncertain about the nature of a matrix, the function **issparse** will provide an answer. This function returns a value of 1 if the matrix is sparse and 0 otherwise. Another useful function for dealing with sparse matrices is **nnz**, which returns the number of non-zero elements in a matrix.

To illustrate working with sparse matrices, example 2.2 generates a sparse matrix system which we solve using the MATLAB 'slash' operator that carries out Gaussian elimination to solve the problem. A full matrix solution is also provided, and we compare the time needed using the **tic** and **toc** commands.

```
% ----- Example 2.2 Solving a sparse matrix system
n = 500; density = 0.1; knum=0.5;
```

```
for i=1:5;
density = density*0.5;
X = sprandsym(n,density,knum);
y = randn(n,1);
tic; b = X\y; time1 = toc;
xf = full(X);
tic; b = xf\y; time2 = toc;
fprintf('# of non-zero elements in X = %6d \n',nnz(X));
fprintf('sparse matrix solution time = %8.3f \n',time1);
fprintf('full matrix solution   time = %8.3f \n',time2);
end;
```

The example relies on the sparse matrix function **sprandsym** to generate a symmetric matrix of dimension 'n=500' with random normal numbers distributed with a density provided by an input argument to the function. This 'density' argument represents the fraction of elements that will be non-zero. We produce a sequence of five matrix systems with increasing sparsity by varying the 'density' input argument each time through the 'for i=1:5' loop. A third argument allows us to specify the condition number for the generated matrix using a fraction 'knum' between zero and one such that the condition number will be 1/knum. We use this option to prevent ill-conditioning in our matrix system as the number of non-zero elements becomes smaller.

The results are shown below where we see that the full matrix operation takes roughly 3.3 seconds to solve all five systems. In contrast, the sparse matrix solution is around 10 times faster when we have a density of 0.05, taking only .356 seconds. As the sparsity increases, the time required to solve the problem decreases dramatically to 0.010 seconds, which is around 300 times faster. For the U.S counties the density is: 12,429/9,653,449 = 0.00128. By comparison, the smallest density in example 2.2 is 0.003125, so it should be clear that one could spend weeks of computer time solving large problems without sparse matrix algorithms or a few seconds with these algorithms.

```
# of non-zero elements in X =  11878
sparse matrix solution time =    0.356
full matrix solution   time =    3.260
# of non-zero elements in X =   5970
sparse matrix solution time =    0.080
full matrix solution   time =    3.370
# of non-zero elements in X =   2982
sparse matrix solution time =    0.020
full matrix solution   time =    3.284
# of non-zero elements in X =   1486
sparse matrix solution time =    0.013
full matrix solution   time =    3.310
# of non-zero elements in X =    744
sparse matrix solution time =    0.010
full matrix solution   time =    3.261
```

The subject of sparse matrix computational efficiency is beyond the scope of this text, but we can provide some insights that will be useful in Chapter 3. Gaussian elimination includes a special preordering of the columns of the matrix labeled "minimum degree ordering", which reduces "fill-in" that occurs with matrix operations. The term "fill-in" refers to the introducion of non-zero elements as a result of matrix manipulation. As an illustration of fill-in, we carry out matrix multiplication of the sparse first-order contiguity matrix taken from Anselin's (1988) Columbus neighborhood crime data set. A comparison of the sparsity associated with the original first-order contiguity matrix W and the matrix product $W * W * W$ is shown in Figure 2.2.

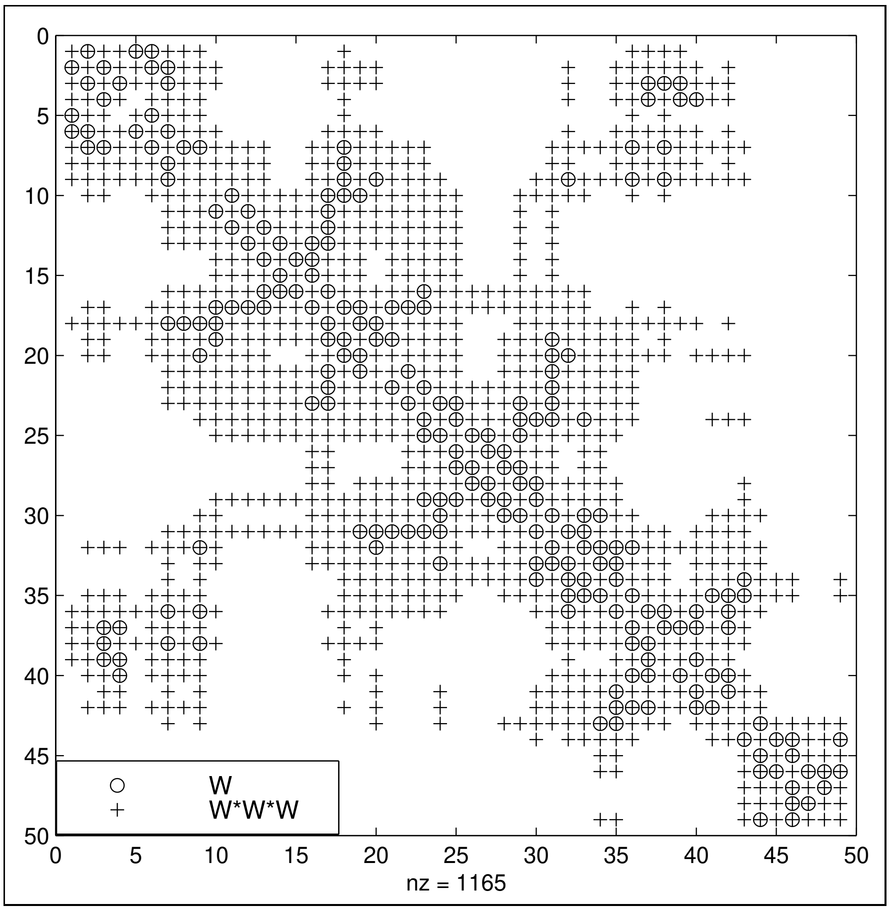

Figure 2.2: An illustration of fill-in from matrix multiplication

To illustrate the importance of preordering in matrix computations, we carry out matrix multiplication on a symmetric sparse random matrix generated using **sprandsym** from example 2.2. We produce a symmetric minimum degree ordering, using the MATLAB function **symmmd** and carry out matrix multiplication based on this ordering. We also rely on MATLAB to automatically do this for us and carry out matrix multiplication involving the sparse matrix. A comparison to the case of a full matrix multiplication is provided in terms of both floating point operations (flops) and the time required.

```
% ----- Example 2.3 Symmetric minium degree ordering operations
n = 500; density = 0.1; knum=0.5;
for i=1:5;
density = density*0.5;
X = sprandsym(n,density,knum);
% produce symmetric minimum degree ordering
mo = symmmd(X);
tic; flops(0); result = X(mo,mo)*X(mo,mo); time0 = toc; flops0 = flops;
tic; flops(0); result = X*X; time1 = toc; flops1 = flops;
xf = full(X);
tic; flops(0); result = xf*xf; time2 = toc; flops2 = flops;
fprintf('# of non-zero elements in X = %6d \n',nnz(X));
fprintf('ordered matrix solution time = %8.3f flops = %16d \n',time0,flops0);
fprintf('sparse matrix solution  time = %8.3f flops = %16d \n',time1,flops1);
fprintf('full matrix solution    time = %8.3f flops = %16d \n',time2,flops2);
end;
```

From the results presented below we see that the minimum degree ordering accounts for the greatly reduced number of floating point operations during sparse matrix multiplication. The time is slightly faster for the straightforward sparse matrix multiplication because the internal ordering is faster than our explicit ordering.

```
# of non-zero elements in X =  11922
ordered matrix solution time =    0.179 flops =          1228634
sparse matrix solution  time =    0.169 flops =          1228634
full matrix solution    time =    8.320 flops =        250000018
# of non-zero elements in X =   5956
ordered matrix solution time =    0.058 flops =           296022
sparse matrix solution  time =    0.047 flops =           296022
full matrix solution    time =    8.533 flops =        250000018
# of non-zero elements in X =   2976
ordered matrix solution time =    0.018 flops =            71846
sparse matrix solution  time =    0.010 flops =            71846
full matrix solution    time =    8.448 flops =        250000018
# of non-zero elements in X =   1502
ordered matrix solution time =    0.008 flops =            17238
sparse matrix solution  time =    0.006 flops =            17238
full matrix solution    time =    8.323 flops =        250000018
# of non-zero elements in X =    742
ordered matrix solution time =    0.004 flops =             2974
sparse matrix solution  time =    0.002 flops =             2974
full matrix solution    time =    8.470 flops =        250000018
```

As a graphical illustration of symmetric minimum degree ordering, Figure 2.3 shows the 3,107 county contiguity matrix sparsity structure in the original unordered form along with this structure for a symmetric minimum degree ordering of this matrix.

MATLAB provides other orderings for non-symmetric matrices using the function **colmmd** and a Cuthill MacGee ordering can be generated using the function **symrcm**. We will have an opportunity to demonstrate these ideas a bit more in Chapter 3.

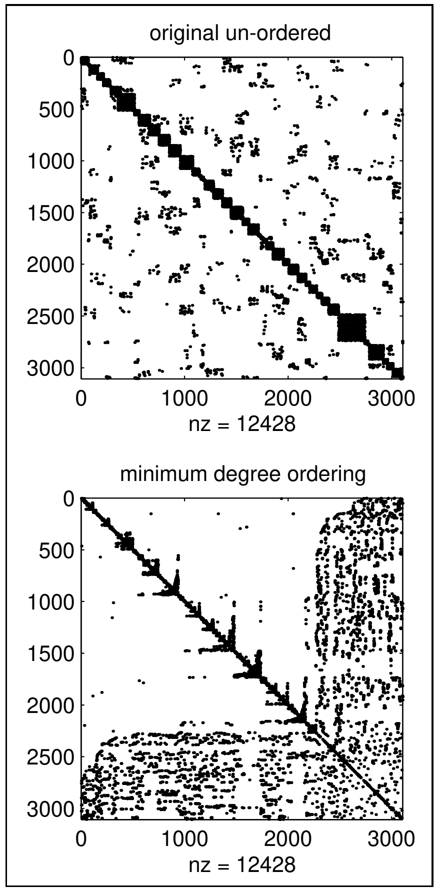

Figure 2.3: Minimum degree ordering versus unordered Pace and Barry matrix

As a final example of sparse matrix functionality, we examine some code from a function **normw** in the spatial econometrics library that normalizes contiguity matrices to have row-sums of unity. One might approach this problem inefficiently using a loop over the rows as shown below.

```
rsum = sum(win');
wout = zeros(n1,n1);
for i=1:n1
wout(i,:) = win(i,:)/rsum(1,i);
end;
```

This code is inefficient because we need to allocate space for wout, which could be a very large but sparse matrix containing millions of elements, most of which are zero. Rather than allocate this memory, we can rely on the **find** command to extract only non-zero elements on which the standardization operates as shown below.

```
[i1,j1,s1] = find(win);
rsum = sum(win');
for i=1:n;
  ind = find(i1 == i);
s1(ind,1) = s1(ind,1)/rsum(1,i);
end;
wout = sparse(i1,j1,s1);
```

We loop over all rows and find a vector of elements from the row index 'i1' equal to the non-zero elements in each row. The index variable 'ind' is then used to standardize only the non-zero elements in 's1', operating on the entire vector of these elements. On exit, the sparse matrix is reconstructed based on the standardized values in the variable 's1' using the **sparse** function.

2.5 Chapter Summary

This chapter described a software design for implementing the spatial econometric estimation methods that we will discuss in this text. Our estimation methods will be implemented using MATLAB software from the MathWorks Inc. A design based on MATLAB structure variables was set forth. This approach to developing a set of spatial econometric estimation functions can provide a consistent user-interface for the function documentation and help information as well as encapsulation of the estimation results in a MATLAB structure variable. This construct can be accessed by related functions that provide printed as well as graphical presentation of the estimation results.

We also demonstrated the computational gains from using the sparse matrix functionality in MATLAB. Further evidence of these gains will be illustrated when implementing our estimation procedures. For the most part, MATLAB will operate intelligently on matrices that are declared sparse without any effort on our part.

Chapter 3

Spatial autoregressive models

This chapter discusses spatial autoregressive models briefly introduced in Chapter 1. These models are used to model cross-sectional spatial data samples and Anselin (1988) provides a relatively complete treatment of these models from a maximum likelihood perspective. In chapter 4 we examine these same models from a Bayesian perspective, where the model is extended to handle cases involving non-constant variance or outliers, a situation that often arises in practice.

The most general statement of a spatial autoregressive model is shown in (3.1).

$$\begin{aligned} y &= \rho W_1 y + X\beta + u \\ u &= \lambda W_2 u + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.1}$$

Where y contains an nx1 vector of cross-sectional dependent variables and X represents an nxk matrix of explanatory variables. W_1 and W_2 are known nxn spatial weight matrices, usually containing contiguity relations or functions of distance. As explained in Section 1.4.1, a first-order contiguity matrix has zeros on the main diagonal, rows that contain zeros in positions associated with non-contiguous observational units and ones in positions reflecting neighboring units that are (first-order) contiguous based on one of the contiguity definitions set forth in Chapter 1.

From the general model in (3.1) we can derive special models by imposing restrictions. For example, setting $X = 0$ and $W_2 = 0$ produces a first-order spatial autoregressive model shown in (3.2).

$$y = \rho W_1 y + \varepsilon \tag{3.2}$$

$$\varepsilon \sim N(0, \sigma^2 I_n)$$

This model attempts to explain variation in y as a linear combination of contiguous or neighboring units with no other explanatory variables. It represents a spatial analogy to the first order autoregressive model from time series analysis, $y_t = \rho y_{t-1} + \varepsilon_t$, where total reliance is on past period observations to explain variation in y_t.

Setting $W_2 = 0$ produces a mixed regressive-spatial autoregressive model shown in (3.3). This model is analogous to the lagged dependent variable model in time series. Here we have additional explanatory variables in the matrix X to explain variation in y over the spatial sample of observations.

$$\begin{aligned} y &= \rho W_1 y + X\beta + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.3}$$

Letting $W_1 = 0$ results in a regression model with spatial autocorrelation in the disturbances shown in (3.4).

$$\begin{aligned} y &= X\beta + u \\ u &= \lambda W_2 u + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.4}$$

A related model known as the spatial Durbin model is shown in (3.5), where a "spatial lag" of the dependent variable as well as a spatial lag of the explanatory variables matrix X are added to a traditional least-squares model.

$$\begin{aligned} y &= \rho W_1 y + X\beta_1 + W_1 X\beta_2 + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.5}$$

This chapter is organized into sections that discuss and illustrate each of the special cases as well as the most general version of the spatial autoregressive model from (3.1).

Section 3.1 deals with the first-order spatial autoregressive model from (3.2) which we label the FAR model. The mixed regressive-spatial autoregressive model from (3.3) that we refer to as the SAR model is taken up in Section 3.2. Section 3.3 takes up the SEM model containing spatial autocorrelation in the disturbances shown in (3.4). In addition, this section illustrates various tests for spatial dependence using least-squares regression residuals and other approaches. These tests can help in specification of an appropriate model from the alternatives available. The model from (3.5) is discussed in Section 3.4. This model is often called the spatial Durbin model, and we reference it as the SDM model. The most general model from (3.1) which we label SAC is the focus of Section 3.5.

Computational details necessary to produce maximum likelihood estimates for each of the models are provided in subsections entitled "computational details". Spatial econometrics library functions that utilize MATLAB sparse matrix algorithms allow us to estimate models with over 3,000 observations in around 100 seconds on an inexpensive desktop computer.

Applied examples are presented in subsections entitled "applied examples" for each of the models. These examples rely on a host of spatial data sets ranging from small to large and even very large.

3.1 The first-order spatial AR model

This model is seldom used in applied work, but it serves to motivate ideas we use in later sections of the chapter. Perhaps the most frequent use for this model is checking residuals for spatial autocorrelation, which we illustrate in the next chapter. The model which we label FAR throughout the text, takes the form:

$$\begin{aligned} y &= \rho W y + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.6}$$

Where the spatial contiguity matrix W has been standardized to have row sums of unity and the variable vector y is expressed in deviations from the means form to eliminate the constant term in the model.

To illustrate the problem with least-squares estimation of spatial autoregressive models, consider applying least-squares to the model in (3.6). This would produce an estimate for the parameter ρ in the model equal to:

$$\hat{\rho} = (y'W'Wy)^{-1} y'W'y \tag{3.7}$$

Can we show that this estimate is unbiased? If not, is it consistent? Taking the same approach as in least-squares, we substitute the expression for y from the model statement and attempt to show that $E(\hat{\rho}) = \rho$, to prove unbiasedness.

$$\begin{aligned} E(\hat{\rho}) &= E[(y'W'Wy)^{-1} y'W'(\rho W y + \varepsilon)] \\ &= \rho + E[(y'W'Wy)^{-1} y'W'\varepsilon] \end{aligned} \tag{3.8}$$

Note that the least-squares estimate is biased, since we cannot show that $E(\hat{\rho}) = \rho$. The usual argument that the explanatory variables matrix X in least-squares is fixed in repeated sampling allows one to pass the expectation operator over terms like $(y'W'Wy)^{-1}y'W'$, and argue that $E(\varepsilon) = 0$, eliminating the bias term. Here however, because of spatial dependence, we cannot make the case that Wy is fixed in repeated sampling. This is not unlike the case with autoregressive time-series models where the observations are generated sequentially and therefore not independent. In this case, the observations are generated by a spatial process. We argued in Chapter 1 that this produces spatial dependence between the observations in the vector y.

This also rules out consistency for the least-squares estimate of ρ, because the probability limit (plim) for the term $y'W'\varepsilon$ is not zero. In fact, Anselin (1988) establishes that:

$$\text{plim}N^{-1}(y'W'\varepsilon) = \text{plim}N^{-1}\varepsilon'W(I-\rho W)^{-1}\varepsilon \tag{3.9}$$

Expression (3.9) is equal to zero only in the trivial case when ρ equals zero and there is no spatial dependence in the data sample.

Given that least-squares will produce biased and inconsistent estimates of the spatial autoregressive parameter ρ in this model, how do we proceed to estimate ρ? The maximum likelihood estimator for ρ requires that we find a value of ρ that maximizes the likelihood function shown in (3.10).

$$L(y|\rho,\sigma^2) = \frac{1}{2\pi\sigma^{2(n/2)}}|I_n - \rho W| \exp\{-\frac{1}{2\sigma^2}(y-\rho Wy)'(y-\rho Wy)\} \tag{3.10}$$

To simplify the maximization problem, we obtain a concentrated log likelihood function based on eliminating the disturbance variance parameter σ^2. This is accomplished by substituting $\hat{\sigma}^2 = (1/n)(y-\rho Wy)'(y-\rho Wy)$ in the likelihood (3.10) and taking logs, which yields the expression in (3.11).

$$\text{Ln}(L) = -\frac{n}{2}ln(\pi) - \frac{n}{2}\ln(y-\rho Wy)'(y-\rho Wy) + \ln|I_n - \rho W| \tag{3.11}$$

This expression can be maximized with respect to ρ using a simplex univariate optimization routine that is part of the MATLAB function library. The estimate for the parameter σ^2 is obtained using the maximum likelihood value of ρ which we label $\tilde{\rho}$ in: $\hat{\sigma}^2 = (1/n)(y-\tilde{\rho}Wy)'(y-\tilde{\rho}Wy)$. In the next section, we demonstrate a sparse matrix approach to evaluating this likelihood function that allows us to solve problems involving thousands of observations quickly with small amounts of computer memory.

Two implementation details arise with this approach to solving for maximum likelihood estimates. First, there is a constraint that we need to impose on the parameter ρ. Anselin and Florax (1994) point out that this parameter can take on feasible values in the range:

$$1/\lambda_{min} < \rho < 1/\lambda_{max}$$

Where λ_{min} represents the minimum eigenvalue of the standardized spatial contiguity matrix W and λ_{max} denotes the largest eigenvalue of this matrix. This requires that we constrain our optimization search to values of ρ within this range.

The second implementation issue is that the numerical hessian matrix that would result from a gradient-based optimization procedure and provide estimates of dispersion for the parameters is not available with simplex optimization. We can overcome this problem in two ways. For problems involving a small number of observations, we can use our knowledge of the theoretical information matrix to produce estimates of dispersion. An asymptotic variance matrix based on the Fisher information matrix shown below for the parameters $\theta = (\rho, \sigma^2)$ can be used to provide measures of dispersion for the estimates of ρ and σ^2. Anselin (1980, page 50) provides the analytical expressions needed to construct this information matrix.

$$[I(\theta)]^{-1} = -E[\frac{\partial^2 L}{\partial\theta\partial\theta'}]^{-1} \tag{3.12}$$

This approach is computationally impossible when dealing with large scale problems involving thousands of observations. The expressions used to calculate terms in the information matrix involve operations on very large matrices that would take a great deal of computer memory and computing time. In these cases we evaluate the numerical hessian matrix using the maximum likelihood estimates of ρ and σ^2 and our sparse matrix function for the likelihood. Given the ability to evaluate the likelihood function rapidly, numerical methods can be used to compute approximations to the gradients shown in (3.12). We will demonstrate both of these approaches in the next section.

3.1.1 Computational details

Building on the software design set forth in Chapter 2 for our spatial econometrics function library, we will implement a function **far** to produce maximum likelihood estimates for the first-order spatial autoregressive model. Since this is our first spatial econometrics estimation function, we start by constructing a function **fars** that doesn't take advantage of sparse matrix functionality available in MATLAB. This keeps the **fars** function simple and makes the entire development process easier to understand. After constructing this simplistic function (that is not useful for large-scale realistic problems), we show how to develop an analogous routine **far** that relies on the sparse matrix functionality of MATLAB. We demonstrate this function which is part of the spatial econometrics library in action on a data set involving 3,107 U.S. counties.

To solve the univariate optimization problem involving the single parameter ρ in the model, we rely on a simplex optimization function **fmin** that is part of the MATLAB function library. This function takes three input arguments: 1) a function name provided as a string argument, 2) a single parameter over which the optimization takes place, and 3) a range of values over which to search for the optimal parameter value. Our first task is to create a function representing the concentrated log likelihood function for the first-order spatial autoregressive model from (3.11). This function named **fs_far** is shown below:

```
function llike = fs_far(rho,nobs,y,W)
% PURPOSE: evaluate the concentrated log-likelihood
% for the first-order spatial autoregressive model
% ---------------------------------------------------
%  USAGE:llike = fs_far(rho,nobs,y,W)
%  where: rho  = spatial autoregressive parameter
```

```
%        nobs = # of observations in the model
%        y    = dependent variable vector
%        W    = standardized contiguity matrix
% ---------------------------------------------------
%  RETURNS: a scalar equal to the concentrated log-likelihood
%           function value at the parameter value rho
%  --------------------------------------------------
%  SEE ALSO: far, f_far f_sar, f_sac, f_sem
% ---------------------------------------------------
IN = eye(nobs); B = IN - rho*W;
dB = det(B); epe = y'*B'*B*y;
llike = (nobs/2)*log(pi) + (nobs/2)*log(epe) - log(dB);
```

One point to note regarding the function **fs_far** is that we return the negative of the concentrated log-likelihood function. This is because the optimization function **fmin** that we use to solve for a maximum likelihood value of the parameter ρ works by minimizing the function. Minimizing the negative of the log-likelihood is equivalent to maximizing the log-likelihood function. A second point is that the likelihood function equations from expression (3.11) were easy to code in MATLAB, which we will find is often the case.

Given this function, we can turn attention to using the MATLAB simplex optimization function **fmin** to find a value for the parameter ρ that minimizes the function. We call this function from within a function **fars** that implements maximum likelihood estimation of the first-order spatial autoregressive model. The **fars** function is shown below:

```
function results = fars(y,W,convg,maxit)
%  PURPOSE: computes 1st-order spatial autoregressive estimates
%           model: y = p*W*y + e
% ---------------------------------------------------
%  USAGE: results = fars(y,W,convg,maxit)
%  where:  y = dependent variable vector
%          W = standardized contiguity matrix
%      convg = (optional) convergence criterion (default = 1e-8)
%      maxit = (optional) maximum # of iterations (default = 500)
% ---------------------------------------------------
%  RETURNS: a structure
%        results.meth  = 'fars'
%        results.rho   = rho
%        results.tstat = asymptotic t-stat
%        results.yhat  = yhat
%        results.resid = residuals
%        results.sige  = sige = (y-p*W*y)'*(y-p*W*y)/n
%        results.rsqr  = rsquared
%        results.lik   = log likelihood
%        results.nobs  = nobs
%        results.nvar  = nvar = 1
%        results.y     = y data vector
%        result.iter   = # of iterations taken
%  --------------------------------------------------
%  SEE ALSO: prt(results), far, sar, sem, sac
% ---------------------------------------------------
% REFERENCES: Anselin (1988), pages 180-182.
% ---------------------------------------------------
options = zeros(1,18);
```

```
if nargin == 2 % set default optimization options
options(1) = 0; options(2) = 1.e-8; options(14) = 500;
elseif nargin == 3 % set user supplied convergence option
options(1) = 0; options(2) = convg; options(14) = 500;
elseif nargin == 4 % set user supplied convg and maxit options
options(1) = 0; options(2) = convg; options(14) = maxit;
else
 error('Wrong # of arguments to fars');
end;
[n junk] = size(y); results.y = y;      results.nobs = n;
results.nvar = 1;   results.meth = 'fars';
weig = eig(W); lmin = min(weig); lmax = max(weig);
wy = W*y; IN = eye(n);
% step 1) maximize concentrated likelihood function;
[p opts] = fmin('fs_far',1/lmin,1/lmax,options,n,y,W);
if opts(10) == options(14),
fprintf(1,'far: convergence not obtained in %4d iterations',options(14));
end;
results.iter = opts(10);
epe = (y - p*wy)'*(y-p*wy); sige = epe/n; % step 2) find sige
results.rho = p; results.yhat = p*wy;
results.resid = y - results.yhat; results.sige = sige;
% asymptotic t-stats (page 50 Anselin, 1980)
B = speye(n) - p*sparse(W);  xpxi = zeros(2,2); WB = W*inv(B);
pterm = trace(WB*WB + WB*WB');          % rho,rho term
term1 = trace(inv(B'*B)*(W'*W)); xpxi(1,1) = term1;
xpxi(2,2) = n/(2*sige*sige);            % sige,sige term
xpxi(1,2) = -(1/sige)*(p*term1 - trace(inv(B'*B)*W));
xpxi(2,1) = xpxi(1,2);                  % sige,rho term
xpxi = inv(xpxi);
results.tstat = results.rho./(sqrt(xpxi(1,1)));
ym = y - mean(y);                  % r-squared, rbar-squared
rsqr1 = results.resid'*results.resid;
rsqr2 = ym'*ym;
results.rsqr = 1.0-rsqr1/rsqr2;   % r-squared
results.lik = -fs_far(p,n,y,W);   % log likelihood
```

This function like all of our spatial estimation routines returns a 'results' structure containing information pertaining to the estimation results. Information regarding the 'fields' of the returned structure is provided in the function documentation that would appear if the user typed 'help fars' in the MATLAB command window.

The function first checks for the proper number of input arguments by the user. The 'options' vector controls the behavior of some optional arguments available when using the MATLAB **fmin** function that carries out simplex optimization. We set default options so that: 1) intermediate results are not printed, 2) the maximum number of iteration allowed are 500, and 3) the convergence criterion is 1e-8. Optional input arguments to the function **fars** allow the user to provide values for these aspects of the optimization problem different from the default values.

After error checking on the inputs and processing any user input options regarding optimization, we fill-in some fields of the 'results structure' that will be returned by the function. In the function documentation we promised to return: 1) the values of the vector y input to the function, 2) the number of observations, 3) the number of variables, and 4) a 'meth' field that allows users as well as other functions to decipher which econometric method produced the results structure. Returning the vector of y observations facilitates plotting results implemented in our **plt_spat** function. The other values in the results fields are used when printing estimation results.

After error checking and initialization of the results structure, we compute the eigenvalues of the spatial weight matrix W using the MATLAB function **eig**. We need the maximum and minimum eigenvalues to constrain our search for the maximum likelihood value of ρ. The function **eig** returns a vector of n eigenvalues associated with the nxn spatial weight matrix, so we use the MATLAB **max** and **min** functions to find the maximum and minimum eigenvalues.

At this point, we call the simplex optimization function **fmin** to solve for a maximum likelihood estimate of ρ. The call takes the form:

```
[p opts] = fmin('fs_far',1/lmin,1/lmax,options,n,y,W);
```

Where: the first argument is a string containing the name of the concentrated log likelihood function we wish to minimize; the next two arguments contain the minimum and maximum of the range over which to search for the optimal parameter value of ρ; the options vector contains our optional arguments that control optimization behavior of the simplex algorithm, and the last three input arguments will be passed to the function **fs_far** as input arguments. Note that our function **fs_far** was written to accept these last three arguments as inputs.

Outputs from the function **fmin** take the form of a scalar argument 'p' that contains the optimal parameter value and a vector 'opts' containing information regarding the optimization process. We use the information in 'opts(10)' to check against the maximum number of iterations set as an input option to **fmin**. If the actual number of iterations taken equals the maximum allowed, we have a situation where convergence has not taken place. We provide a printed warning for the user in this case, but still return a results structure containing estimates based on the non-optimal value of ρ. This allows the user to examine the printed output and attempt another solution to the problem using a different convergence criterion or an increased number of iterations. The number of iterations taken by **fmin** are also returned as a field in the results structure and printed with other output information.

Having obtained a maximum likelihood estimate ($\hat{\rho}$) of the parameter ρ, we use the matrix relationships to obtain an estimate of the parameter σ^2 based on the residuals from the model, $e = y - \hat{\rho}Wy$. Finally, we construct $t-$statistics based on the theoretical information matrix provided by Anselin (1988).

The code segment from **prt_spat** that prints the output associated with the **fars** model estimates might look as follows:

```
case {'fars'} % <=================== first-order autoregressive model
nvar = 1; nobs = results.nobs;
% special handling of vnames
Vname = 'Variable';
Vname = strvcat(Vname,'rho'); % add spatial rho parameter name
```

```
fprintf(fid,'\n');
fprintf(fid,'First-order spatial autoregressive model Estimates \n');
if (nflag == 1)
fprintf(fid,'Dependent Variable = %16s \n',vnames(1,:));
end;
fprintf(fid,'R-squared       = %9.4f \n',results.rsqr);
fprintf(fid,'sigma^2         = %9.4f \n',results.sige);
fprintf(fid,'Nobs, Nvars     = %6d,%6d \n',results.nobs,results.nvar);
fprintf(fid,'log-likelihood  = %16.8g \n',results.lik);
fprintf(fid,'# of iterations = %6d   \n',results.iter);
fprintf(fid,'*******************************************************\n');
bout = results.rho;
% <================== end of far case
```

Of course, the code at the end of **prt_spat** that prints the estimates would also be executed based on the MATLAB variable 'bout' and the variable names vector 'Vname' set in the code above. As pointed out in Chapter 2, this occurs at the end of the **prt_spat** function, since all of the spatial regression functions have this task in common. For the sake of completeness, we show this code below. Note that we would not have to write this code as it was already part of the spatial econometrics library — only the code shown above would be needed to implement the **fars** model and add it to the library. One point to note is that we compute the marginal probability associated with the asymptotic $t-$ratio using a normal or z-probability in place of the $t-$distribution probability. Unlike the case of least-squares regression the parameters in spatial autogressive models do not take the on a $t-$distribution, but represent asymptotic $t-$statistics so the marginal probabilities are calculated using the normal distribution. This is accomplished using the function **norm_prb** which was written for this purpose since asymptotic $t-$ratios often arise in econometric analysis. This function is part of the *Econometrics Toolbox* and is described in Chapter 8 of the manual for the toolbox.

```
% now print coefficient estimates, t-statistics and probabilities
tout = norm_prb(results.tstat); % find asymptotic t-stat z-probabilities
tmp = [bout results.tstat tout];  % matrix to be printed
% column labels for printing results
bstring = 'Coefficient'; tstring = 'Asymptot t-stat'; pstring = 'z-probability';
cnames = strvcat(bstring,tstring,pstring);
in.cnames = cnames; in.rnames = Vname;
in.fmt = '%16.6f'; in.fid = fid;
mprint(tmp,in);
```

As indicated at the outset, the function **fars** is not part of the spatial econometrics library, it was merely used to motivate our approach to estimating this model. It has some computational shortcomings when it comes to solving large sample problems. The computational difficulties are: 1) we require eigenvalues for the large n by n matrix W, 2) we need to find the determinant of the n by n related matrix $(I_n - \rho W)$, and 3) matrix multiplications involving the large matrices W and $(I_n - \rho W)$ are required to compute the information matrix to produce estimates of dispersion.

We construct an alternative function **far** that relies on sparse matrix algorithms available in MATLAB and is part of the spatial econometrics library. We will demonstrate this function with a sample data set containing observations for 3,107 U.S. counties. The function can produce estimates for the first-order spatial autoregressive model in this large problem in 95 seconds on a moderately fast but inexpensive desktop computer. These MATLAB algorithms for dealing with sparse matrices make the software ideally suited for spatial modeling because spatial weight matrices are almost always sparse.

Our first task is to construct a function to evaluate the concentrated log likelihood based on the sparse matrix algorithms. This function named **f_far** is shown below.

```
function llike = f_far(rho,y,W)
% PURPOSE: evaluate the concentrated log-likelihood for the first-order
% spatial autoregressive model using sparse matrix algorithms
% ---------------------------------------------------
%  USAGE:llike = f_far(rho,y,W)
%  where: rho  = spatial autoregressive parameter
%         y    = dependent variable vector
%         W    = spatial weight matrix
% ---------------------------------------------------
%  RETURNS: a  scalar equal to minus the log-likelihood
%           function value at the parameter rho
%  --------------------------------------------------
%  SEE ALSO: far, f_sar, f_sac, f_sem
% ---------------------------------------------------
n = length(y); spparms('tight');
z = speye(n) - 0.1*sparse(W);
p = colmmd(z);
z = speye(n) - rho*sparse(W);
[l,u] = lu(z(:,p));
detval = sum(log(abs(diag(u))));
epe = y'*z'*z*y;
llike = (n/2)*log(pi) + (n/2)*log(epe) - detval;
```

The function solves the determinant of $(I - \rho W)$ using the LU decomposition implemented by the MATLAB **lu** command. This algorithm operates on sparse matrices in an intelligent way. The command **spparms** sets options for operation of the sparse matrix algorithms. The option 'tight' sets the minimum degree ordering parameters to particular settings, which lead to sparse matrix orderings with less fill-in. As we saw in Section 2.4, this has a great impact on the execution time. Some experimentation on my part with the various options that can be set has led me to believe this is an optimal setting for spatial autoregressive models. For the reader interested in further details regarding sparse matrix algorithms in this context, see George and Liu (1981). Pace and Barry (1998) provide a good discussion of the alternative MATLAB sparse matrix settings in the context of spatial econometric models along with timing results and operation counts for various settings.

The command **sparse** informs MATLAB that the matrix W is sparse and the command **speye** creates an identity matrix in sparse format. We set up an initial matrix based on $(I_n - 0.1W)$ from which we construct a column vector of minimum degree orderings for this sparse matrix. Note that we rely on the function **colmmmd** rather than **symmmd** which determines a minimum degree ordering for symmetric matrices. This allows users to rely on non-symmetric spatial weight matrices, making the function more generally useful. By executing the **lu** command with the minimum degree ordering, we manage to operate on a sparser set of LU factors than if we operated on the matrix $z = (I - \rho W)$. As we saw in Chapter 2, this improves execution time.

Given this function to evaluate the log likelihood for very large spatial weight matrices W, we can now rely on the same **fmin** simplex optimization algorithm as in the case of the function **fars**. Another place where we can rely on sparse matrix functions is in determining the minimum and maximum eigenvalues of the matrix W. Recall we use these to set the feasible range of values for ρ in our call to the simplex search function **fmin**. The code for carrying this out is:

```
opt.tol = 1e-3; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
lmin = 1/lambda(2);
lmax = 1/lambda(1);
```

The MATLAB function **eigs** works to compute selected eigenvalues for a sparse matrix, and we use the option 'BE' to compute only the maximum and minimum eigenvalues which we need to determine the feasible range for ρ. The options are set as fields in the 'opt' structure variable which is input as an argument to the **eigs** function. A tolerance value is one option we use and the other option is set to prohibit display of the iterative solution results. The default tolerance is 1e-10, but using a tolerance of 1e-3 or 1e-4 speeds up the solution by a factor of four times. Since the function **eigs** works iteratively, we are trading off some accuracy by truncating the number of iterations with the larger tolerance values. This seems a reasonable trade-off because we are not overly concerned about loosing a few decimal digits of accuracy. It is unlikely that the maximum likelihood values of ρ are near the upper and lower bounds on the feasible range. If we find a maximum likelihood estimate near these limits, we can always change the tolerance option used as an input argument to the **eigs** function.

Another point is that for the case of standardized weight matrices W, the maximum eigenvalue will always take on a value of unity, so we could save time by only computing one eigenvalue. However, not all researchers will use a row-standardized matrix W, so we make the function more generally useful by computing both eigenvalues.

We face a design decision regarding whether to provide an optional input to the function **far** for the tolerance used by the **eigs** function. A decision to keep things simple ruled out an input argument for this option. Readers however should be aware that for some spatial weight matrices inaccurate or even imaginary eigenvalues may be found due to the large tolerance value used. In these cases there are two remedial actions that can be taken. First, you can change the tolerance value in the function **far**. A second solution is provided by input options for user-supplied maximum and minimum values over which to search for estimates of ρ. These values wil be used in lieu of the computed eigenvalues.

An option for user input of the search range is provided for all of the spatial autoregressive models discussed in this chapter. It allows users to save computing time when estimating several different specifications of a spatial autoregressive model. In cases where all of the models rely on the same spatial weight matrix, you can compute the limits once and then input these limits for the other models.

The function **far** is shown below, where the optional inputs 'rmin' and 'rmax' are documented. This function was written to perform on both large and small problems. If the problem is small (involving less than 500 observations), the function **far** computes measures of dispersion using the theoretical information matrix. If more observations are involved, the function determines these measures by computing a numerical hessian matrix. (Users may need to decrease the number of observations to less than 500 if they have computers without a large amount of RAM memory.)

```
function results = far(y,W,rmin,rmax,convg,maxit)
% PURPOSE: computes 1st-order spatial autoregressive estimates
%          y = p*W*y + e, using sparse matrix algorithms
% ---------------------------------------------------
% USAGE: results = far(y,W,rmin,rmax,convg,maxit)
% where:  y = dependent variable vector
%         W = standardized contiguity matrix
%      rmin = (optional) minimum value of rho to use in search
%      rmax = (optional) maximum value of rho to use in search
%     convg = (optional) convergence criterion (default = 1e-8)
%     maxit = (optional) maximum # of iterations (default = 500)
% ---------------------------------------------------
% RETURNS: a structure
%        results.meth  = 'far'
%        results.rho   = rho
%        results.tstat = asymptotic t-stat
%        results.yhat  = yhat
%        results.resid = residuals
%        results.sige  = sige = (y-p*W*y)'*(y-p*W*y)/n
%        results.rsqr  = rsquared
%        results.lik   = -log likelihood
%        results.nobs  = nobs
%        results.nvar  = nvar = 1
%        results.y     = y data vector
%        results.iter  = # of iterations taken
%        results.rmax  = 1/max eigenvalue of W (or rmax if input)
%        results.rmin  = 1/min eigenvalue of W (or rmin if input)
%  --------------------------------------------------
% SEE ALSO: prt(results), sar, sem, sac
% ---------------------------------------------------
options = zeros(1,18); rflag = 0;
options(1) = 0; options(2) = 1.e-6; options(14) = 500;
if nargin == 2,               % use default options
elseif nargin == 4, rflag = 1; % set user lmin,lmax
elseif nargin == 5, rflag = 1; % set user convergence option
options(1) = 0; options(2) = convg; options(14) = 500;
elseif nargin == 6, rflag = 1; % set user convg, maxit options
```

```
options(1) = 0; options(2) = convg; options(14) = maxit;
else, error('Wrong # of arguments to far');
end;
[n junk] = size(y); results.y = y; results.nobs = n; results.nvar = 1;
results.meth = 'far';
if rflag == 0, opt.tol = 1e-3; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
rmin = 1/lambda(2);       rmax = 1/lambda(1);
end;
results.rmax = rmax;      results.rmin = rmin;
% step 1) maximize concentrated likelihood function;
[p options] = fmin('f_far',rmin,rmax,options,y,W);
if options(10) == options(14),
fprintf(1,'far: convergence not obtained in %4d iterations',options(14));
else, results.iter = options(10);
end;
Wy = sparse(W)*y; epe = (y - p*Wy)'*(y-p*Wy); sige = epe/n;
results.rho = p; results.yhat = p*Wy;
results.resid = y - results.yhat; results.sige = sige;
parm = [p
        sige];
if n > 500, % asymptotic t-stats using numerical hessian
hessn = hessian('f2_far',parm,y,W); xpxi = inv(hessn);
if xpxi(1,1) < 0, xpxi(1,1) = 1;
fprintf(1,'far: negative or zero variance from numerical hessian \n');
fprintf(1,'far: replacing t-stat with 0 \n');
end;
results.tstat = results.rho/sqrt(xpxi(1,1));
if xpxi(1,1) < 0, results.tstat(1,1) = 0; end;
else, % asymptotic t-stats based on information matrix (page 50 Anselin, 1980)
B = speye(n) - p*sparse(W);  xpxi = zeros(2,2); WB = W*inv(B);
pterm = trace(WB*WB + WB*WB');            % rho,rho term
term1 = trace(inv(B'*B)*(W'*W)); xpxi(1,1) = term1;
xpxi(2,2) = n/(2*sige*sige);              % sige,sige term
xpxi(1,2) = -(1/sige)*(p*term1 - trace(inv(B'*B)*W));
xpxi(2,1) = xpxi(1,2); xpxi = inv(xpxi); % sige,rho term
results.tstat = results.rho./(sqrt(xpxi(1,1)));
end;
ym = y - mean(y); rsqr1 = results.resid'*results.resid;
rsqr2 = ym'*ym; results.rsqr = 1.0-rsqr1/rsqr2;   % r-squared
results.lik = -f2_far(parm,y,W);                 % -log likelihood
```

The function follows our earlier function **fars** in terms of design, with the added sparse matrix code. Another change regards computing measures of dispersion. A subtle point in the function is that sparse matrix operations must involve two sparse matrices, so we rely on the function **speye** to create a sparse version of the identity matrix when computing the matrix $I_n - \rho W$. If we used the identity function **eye**, MATLAB would not carry out sparse matrix operations.

As already noted, we cannot always rely on the information matrix approach used in the function **fars** because this would involve matrix operations on very large matrices when the sample size is large. In these cases, we produce measures of dispersion by numerically evaluating the hessian matrix using the maximum likelihood estimates of ρ and σ^2. A function **hessian** computes the hessian matrix given maximum likelihood estimates for ρ and σ and a non-concentrated version of the log likelihood function named **f2_far** shown below.

```
function llike = f2_far(parm,y,W)
% PURPOSE: evaluate the log-likelihood for ML rho,sigma values
% for the first-order spatial autoregressive model
% ---------------------------------------------------
%  USAGE: llike = f2_far(parm,y,W)
%  where: parm  = 2x1 vector with rho,sigma values
%          y    = dependent variable vector
%          W    = spatial weight matrix
% ---------------------------------------------------
%  RETURNS: a scalar equal to minus the log-likelihood
%           function value at the ML parameters rho,sigma
%  --------------------------------------------------
%  SEE ALSO: far, f2_sar, f2_sac, f2_sem
% ---------------------------------------------------
n = length(y); rho = parm(1,1); sige = parm(2,1);
spparms('tight'); z = speye(n) - 0.1*sparse(W);
p = colmmd(z);    z = speye(n) - rho*sparse(W);
[l,u] = lu(z(:,p));
detval = sum(log(abs(diag(u))));
epe = y'*z'*z*y;
llike = (n/2)*log(pi) + (n/2)*log(epe) +(n/2)*log(sige) - detval;
```

We do not present the function **hessian**, as it is a bit off the subject. The interested reader is free to examine the function which numerically approximates expression (3.12). A key to using this approach is the ability to evaluate the log likelihood function in **f2_far** using the sparse algorithms to handle large matrices.

A point to note regarding the numerical hessian approach for computing measures of dispersion is that negative variances might result in cases where the computational approach encounters difficulty determining accurate gradients. Users can often avoid these problems by transforming the data to deviations from the means form, which improves the scaling so that likelihood function evaluations are more numerically accurate. As a rule, adequately scaled data is often important when solving non-linear optimization problems and spatial autoregressive models are no exception to this rule.

If negative variances are encountered, users would see error messages indicating an attempt to take the square root of negative numbers when the $t-$statistics are computed. These messages would be quite confusing, so we rely on the code below to circumvent these error messages.

```
if n > 500, % asymptotic t-stats using numerical hessian
hessn = hessian('f2_far',parm,y,W); xpxi = inv(hessn);
if xpxi(1,1) <= 0, xpxi(1,1) = 1;
fprintf(1,'far: negative or zero variance from numerical hessian \n');
fprintf(1,'far: replacing t-stat with 0 \n');
end;
results.tstat = results.rho/sqrt(xpxi(1,1));
if xpxi(1,1) < 0, results.tstat(1,1) = 0; end;
```

The code checks for a negative or zero variance, which results in a printed warning message. Results are still returned and can be printed using the **prt** function, but the $t-$statistics associated with negative variances will be replaced with zero values.

It should be noted that Pace and Barry (1997) when confronted with the task of providing measures of dispersion for spatial autoregressive estimates based on sparse algorithms suggest using likelihood ratio tests to determine the significance of the parameters. The approach taken here may suffer from some numerical inaccuracy relative to measures of dispersion based on the theoretical information matrix, but has the advantage that users will be presented with traditional $t-$statistics on which they can base inferences.

We will have more to say about how our approach to solving large spatial autoregressive estimation problems using sparse matrix algorithms in MATLAB compares to one proposed by Pace and Barry (1997), when we apply the function **far** to a large data set in the next section.

3.1.2 Applied examples

Given our function **far** that implements maximum likelihood estimation of small and large first-order spatial autoregressive models, we turn attention to illustrating the use of the function with some spatial data sets. In addition to the estimation functions, we have functions **prt_spat** and **plt_spat** that are called by **prt** and **plt** to provide printed and graphical presentation of the estimation results.

Example 3.1 provides an illustration of using these functions to estimate a first-order spatial autoregressive model for a spatial data sample involving 49 neighborhoods in Columbus, Ohio from the Anselin (1988). The variable vector y in this model is neighborhood crime incidents.

Note that we convert the variable vector containing crime incidents to deviations from the means form. To illustrate the difference in accuracy between measures of dispersion based on the theoretical information matrix and the numerical hessian approach, we present estimation results from the **far** function tricked into computing estimates of dispersion based on the numerical hessian approach for this small problem.

```
% ----- Example 3.1 Using the far() function
load Wmat.dat;    % standardized 1st-order contiguity matrix
load anselin.dat; % load Anselin (1988) Columbus neighborhood crime data
y = anselin(:,1);
ydev = y - mean(y);
W = wmat;
vnames = strvcat('crime','rho');
res = far(ydev,W);  % do 1st-order spatial autoregression
prt(res,vnames);    % print the output
plt(res,vnames);    % plot actual vs predicted and residuals
```

This example produced the following printed output with the graphical output presented in Figure 3.1. Based on the $t-$statistic of 4.25 for the parameter ρ shown in the output, we would infer that spatial dependence exists among the neighborhood crime incidents. We interpret this statistic in the typical regression fashion to indicate that the estimated ρ lies 4.25 standard deviations away from zero. The R^2 statistic indicates that this model explains nearly 44% of the variation in crime incidents (in deviations from the means.)

```
% Dispersion estimates based on theoretical information matrix
First-order spatial autoregressive model Estimates
Dependent Variable =            crime
R-squared       =    0.4390
sigma^2         =  153.8452
Nobs, Nvars     =     49,     1
log-likelihood  =       -373.44669
# of iterations =     12
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable      Coefficient  Asymptot t-stat    z-probability
rho              0.669775         4.259172         0.000021
% Dispersion estimates based on numerical hessian
First-order spatial autoregressive model Estimates
Dependent Variable =            crime
R-squared       =    0.4390
sigma^2         =  153.8452
Nobs, Nvars     =     49,     1
log-likelihood  =       -373.44669
# of iterations =     12
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable      Coefficient  Asymptot t-stat    z-probability
rho              0.669775         6.078831         0.000000
```

From the example, we see that the inaccuracy from using the numerical hessian to determine the dispersion of ρ would not lead to inferences very different from those based on dispersion calculated using the theoretical information matrix. We provide additional comparisons in Section 5.7 for a larger problem with 506 observations and 14 explanatory variables.

Another more challenging example involves a large sample of 3,107 observations representing counties in the continental U.S. from Pace and Barry (1997). They examine presidential election results for this large sample of observations covering the U.S. presidential election of 1980 between Carter and Reagan. The variable we wish to explain using the first-order spatial autoregressive model is the proportion of total possible votes cast for both candidates. Only persons 18 years and older are eligible to vote, so the proportion is based on those voting for both candidates divided by the population over 18 years of age.

Pace and Barry (1997) suggest an alternative approach to that implemented here in the function **far**. They propose overcoming the difficulty we face in evaluating the determinant $(I - \rho W)$ by computing this determinant once over a grid of values for the parameter ρ ranging from $1/\lambda_{min}$ to $1/\lambda_{max}$ prior to estimation. They suggest a grid based on 0.01 increments for ρ over the feasible range. Given these pre-determined values for the determinant $(I - \rho W)$, they point out that one can quickly evaluate the log-likelihood function for all values of ρ in the grid and determine the optimal value of ρ as that which maximizes the likelihood function value over this grid. Note that their proposed approach would involve evaluating the determinant around 200 times if the feasible range

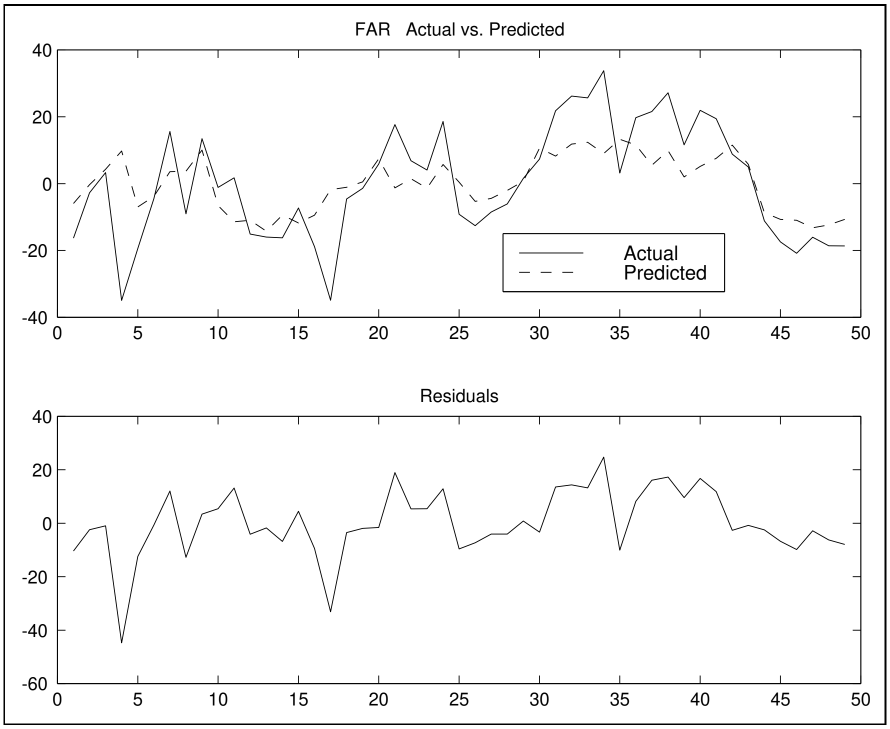

Figure 3.1: Spatial autoregressive fit and residuals

of ρ was -1 to 1. In many cases the range is even greater than this and would require even more evaluations of the determinant. In contrast, our function **far** reports that only 17 iterations requiring log likelihood function evaluations involving the determinant were needed to solve for the estimates in the case of the Columbus neighborhood crime data set.

In addition, consider that one might need to construct a finer grid around the approximate maximum likelihood value of ρ determined from the initial grid search, whereas our use of the MATLAB simplex algorithm produces an estimate that is accurate to a number of decimal digits.

To compare our function **far** with the approach proposed by Pace and Barry, we implement their approach in example 3.2. We take a more efficient approach to the grid search over values of the parameter ρ than suggested by Pace and Barry. Rather than search over a large number of values for ρ, we based our search on a large increment of 0.1 for an initial grid of values covering ρ from $1/\lambda_{min}$ to $1/\lambda_{max}$. Given the determinant of $(I - \rho W)$ calculated using sparse matrix algorithms in MATLAB, we evaluated the negative log likelihood function values for this grid of ρ values to find the value that minimized the likelihood function. We then make a second pass based on a tighter grid with increments of 0.01 around the optimal ρ value found in the first pass. A third and final pass is based on an even finer grid with increments of 0.001 around the optimal estimate from the second pass.

```
% ----- Example 3.2 Using sparse matrix functions and Pace-Barry approach
%       to estimate 1st order autoregressive model rho parameter
load ford.dat; % 1st order contiguity matrix stored in sparse matrix form
ii = ford(:,1); jj = ford(:,2); ss = ford(:,3);
nnzip = length(ii); n = 3107; options.tol = 1e-3;
clear ford; % clear ford matrix to save RAM memory
W = sparse(ii,jj,ss,n,n);  I = speye(n,n);
clear ii; clear jj; clear ss; % clear to save RAM memory
tic; lambda = eigs(W,I,2,'BE',options); toc;
z = I-.1*W;                  p = colmmd(z);
lmin = 1/lambda(2);          lmax = 1/lambda(1);
lgrid = lmin:.1:lmax;        iter = length(lgrid);
like = zeros(iter,2);
load elect.dat;               % load data on votes
y = elect(:,7)./elect(:,8); % proportion of voters casting votes
ydev = y - mean(y);           % deviations from the means form
spparms('tight');             % set sparse matrix options
tic; for i=1:iter;
rho = lgrid(i); z=(I-rho*W); [l,u]=lu(z(:,p));
detval =sum(log(abs(diag(u))));
epe = ydev'*z'*z*ydev;
like(i,1) = -detval + (n/2)*log(epe); like(i,2) = rho;
end; toc;
% now find rho value that maximized the -log likelihood function
[likemin index] = min(like(:,1)); rhoapprox = like(index,2);
% form another grid around this value
lgrid = rhoapprox-0.1:0.01:rhoapprox+0.1;
iter = length(lgrid); like = zeros(iter,2);
tic; for i=1:iter;
rho = lgrid(i); z=(I-rho*W); [l,u]=lu(z(:,p));
detval =sum(log(abs(diag(u))));
epe = ydev'*z'*z*ydev;
like(i,1) = -detval + (n/2)*log(epe); like(i,2) = rho;
end; toc;
% now find rho value that maximized the -log likelihood function
[likemin index] = min(like(:,1)); rhonext = like(index,2);
% form another grid around this value
lgrid = rhonext-0.01:0.001:rhonext+0.01;
iter = length(lgrid); like = zeros(iter,3);
tic; for i=1:iter;
rho = lgrid(i); z=(I-rho*W); [l,u]=lu(z(:,p));
detval =sum(log(abs(diag(u))));
epe = ydev'*z'*z*ydev;
like(i,1) = -detval + (n/2)*log(epe); like(i,2) = rho;
like(i,3) = epe;
end; toc;
[likemin index] = min(like(:,1));
rhofinal = like(index,2); sigfinal = like(index,3);
fprintf(1,'estimate of rho = %8.4f \n',rhofinal);
fprintf(1,'estimate of sigma = %8.4f \n',sigfinal/n);
fprintf(1,'-log likelihood function value = %16.8f \n',like(index,1));
```

Note that in example 3.2 we set the tolerance for solving the eigenvalues of the contiguity matrix W to 1e-3, rather than rely on the default tolerance of 1e-

10. This speeds the calculation, requiring around 60 seconds to solve this part of the problem as shown in the output below. The time necessary to perform each pass over the grid of 21 values for ρ was around 10 seconds. With a total of 3 passes to produce an estimate of ρ accurate to 3 decimal digits, we have a total elapsed time of 1 minute and 30 seconds to solve for the maximum likelihood estimate of ρ. This is certainly a reasonable amount of computational time for such a large problem on a reasonably inexpensive desktop computing platform. Of course, there is still the problem of producing measures of dispersion for the estimates that Pace and Barry address by suggesting the use of likelihood ratio statistics.

The output from the program in example 3.3 was as follows:

```
elapsed_time = 59.8226 % computing min,max eigenvalues
elapsed_time = 10.5280 % carrying out 1st 21-point grid over rho
elapsed_time = 10.3791 % carrying out 2nd 21-point grid over rho
elapsed_time = 10.3747 % carrying out 3rd 21-point grid over rho
estimate of rho =   0.7220
estimate of sigma =   0.0054
log likelihood function value =    4606.75550074
```

How does our approach compare to that of Pace and Barry? Example 3.3 shows a program to estimate the same model as that in example 3.2 using our **far** function. The first point is that our function is much easier to use than the approach of Pace and Barry, but their approach could also be encapsulated in a MATLAB function to make it easier to implement.

```
% ----- Example 3.3 Solving for rho using the far() function
%       with very large data set from Pace and Barry

load elect.dat;             % load data on votes
y = elect(:,7)./elect(:,8); % proportion of voters casting votes
ydev = y - mean(y);         % deviations from the means form
clear y;     % conserve on RAM memory
clear elect; % conserve on RAM memory
load ford.dat; % 1st order contiguity matrix stored in sparse matrix form
ii = ford(:,1); jj = ford(:,2); ss = ford(:,3);
n = 3107;
clear ford; % clear ford matrix to save RAM memory
W = sparse(ii,jj,ss,n,n);
clear ii; clear jj; clear ss; % conserve on RAM memory
tic; res = far(ydev,W); toc;
prt(res);
lmin = res.rmin; lmax = res.rmax;
tic; res = far(ydev,W,lmin,lmax); toc;
prt(res);
```

In terms of time needed to solve the problem, our use of the simplex optimization algorithm takes only 10.6 seconds to produce a more accurate estimate than that based on the grid approach of Pace and Barry. Their approach which we modified took 30 seconds to solve for a ρ value accurate to 3 decimal digits. Note also in contrast to Pace and Barry, we compute a conventional $t-$statistic using the numerical hessian estimates which required only 1.76 seconds. The total time required to compute not only the estimates and measures of dispersion for ρ and σ, but the R−squared statistics and log likelihood function was around 100 seconds. One point to note is that the log likelihood function values are not comparable because: 1) an inessential constant was left out of the log likelihood in example 3.2 and 2) the function **far** reports a likelihood function value based on the non-concentrated likelihood, whereas example 3.2 relies on the concentrated log likelihood.

```
elapsed_time = 59.8226 % computing min,max eigenvalues
elapsed_time = 10.6622 % time required for simplex solution of rho
elapsed_time =  1.7681 % time required for hessian evaluation
elapsed_time =  1.7743 % time required for likelihood evaluation
total time   = 74.01   % comparable time to Pace and Barry

First-order spatial autoregressive model Estimates
R-squared       =    0.5375
sigma^2         =    0.0054
Nobs, Nvars     =   3107,     1
log-likelihood  =        1727.9824
# of iterations =     12
min and max rho =   -1.0710,   1.0000
***************************************************************
Variable      Coefficient  Asymptot t-stat    z-probability
rho              0.721474        59.495145         0.000000
```

Another point worth noting is that setting a minimum and maximum value for the parameter ρ of -1.071 and 1 saves computational time needed to compute the eigenvalues of the matrix W. Using this option in this application reduced the time required to 37 seconds, a factor of three improvement in speed. Setting the limits for the search to the range 0 to 1 speeds up the computation slightly as it reduces the search range facing the simplex algorithm. This produced estimates in 10 iterations rather than 12, taking 33 seconds.

For a truly spectacular example of the power of MATLAB sparse matrix algorithms, we estimated a FAR model using the sample of 35,702 home selling prices for Lucas county Ohio discussed in Chapter 1. Solving this estimation problem took 72 minutes when the feasible range for ρ was computed based on the eigenvalues and 51 minutes when the computations were avoided by setting a 0,1 range for ρ. The estimation results are presented below for this very large model.

```
First-order spatial autoregressive model Estimates
R-squared       =    0.6518
sigma^2         =    0.3256
Nobs, Nvars     =  35702,     1
log-likelihood  =        -173518.8
# of iterations =     19
min and max rho =    -2.0892,   1.0000
***************************************************************
Variable      Coefficient  Asymptot t-stat    z-probability
rho              0.996304      3942.332568         0.000000
```

This solution was produced on an Apple PowerPC computer with a 266 Mhz. G3 chip. Monitoring actual memory usage is a bit difficult, but indications were that only around 32 megabytes of RAM were used to solve this problem. The amount of computational gain from using the sparse matrix algorithms depends heavily on the amount of sparsity that exists in the spatial weight matrix. For this example, there were 414,168 non-zero elements in the 35,702 by 35,702 spatial weight matrix, or a fraction, 0.000168 of the total elements. The average number of neighbors was very close to 6. This means that on average only 6 of the 35,702 elements in each row of the spatial weight matrix were non-zero.

For extremely large problems such as this, Pace and Barry (1998) discuss clever approaches for computing the determinant of $(I_n - \rho W)$ that rely on polynomial and cubic spline interpolation. Intuitively, the idea is to solve the determinant of a smaller problem over a rough grid of values for the parameter ρ and then interpolate these solutions for the larger problem. As demonstrated by our problem involving 35,702 observations, the **far** function developed here can solve these very large problems given adequate time and computer RAM memory.

Many of the ideas developed in this section regarding MATLAB sparse matrix algorithms will apply equally to the estimation procedures we develop in the next four sections for other members of the spatial autoregressive model family.

3.2 The mixed autoregressive-regressive model

This model extends the first-order spatial autoregressive model to include a matrix X of explanatory variables such as those used in traditional regression models. Anselin (1988) provides a maximum likelihood method for estimating the parameters of this model that he labels a 'mixed regressive - spatial autoregressive model'. We will refer to this model as the spatial autoregressive model (SAR). The SAR model takes the form:

$$\begin{aligned} y &= \rho W y + X\beta + \varepsilon \qquad (3.13) \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned}$$

Where y contains an nx1 vector of dependent variables, X represents the usual nxk data matrix containing explanatory variables and W is our spatial contiguity matrix. The parameter ρ is a coefficient on the spatially lagged dependent variable, Wy, and the parameters β reflect the influence of the explanatory variables on variation in the dependent variable y. The model is termed a mixed regressive - spatial autoregressive model because it combines the standard regression model with a spatially lagged dependent variable, reminiscent of the lagged dependent variable model from time-series analysis.

Maximum likelihood estimation of this model is based on a concentrated likelihood function as was the case with the FAR model. A few regressions are carried out along with a univariate parameter optimization of the concentrated likelihood function over values of the autoregressive parameter ρ. The steps are enumerated in Anselin (1988) as:

1. perform OLS for the model: $y = X\beta_0 + \varepsilon_0$
2. perform OLS for the model $Wy = X\beta_L + \varepsilon_L$
3. compute residuals $e_0 = y - X\hat{\beta}_0$ and $e_L = Wy - X\hat{\beta}_L$
4. given e_0 and e_L, find ρ that maximizes the concentrated likelihood function: $L_C = -(n/2)\ln(\pi) - (n/2)\ln(1/n)(e_0 - \rho e_L)'(e_0 - \rho e_L) + \ln|I - \rho W|$
5. given $\hat{\rho}$ that maximizes L_C, compute $\hat{\beta} = (\hat{\beta}_0 - \rho\hat{\beta}_L)$ and $\hat{\sigma}_\varepsilon^2 = (1/n)(e_0 - \rho e_L)'(e_0 - \rho e_L)$

As in the case of the FAR model, the univariate simplex optimization algorithm leaves us with no estimates of dispersion for the parameters. We can overcome this using the theoretical information matrix for small problems and the numerical hessian approach introduced for the FAR model in the case of large problems.

Since this model is quite similar to the FAR model which we already presented, we will turn to computational implementation issues.

3.2.1 Computational details

Our first task is to construct a function to evaluate the concentrated log-likelihood function for this model, which is shown below:

```
function llike = f_sar(rho,eo,el,W)
% PURPOSE: evaluates concentrated log-likelihood  for the spatial
%   autoregressive model using sparse matrix algorithms
% ---------------------------------------------------
% USAGE:llike = f_sar(rho,eo,el,W)
% where: rho  = spatial autoregressive parameter
%        eo   =  y - X b, (least-squares residuals)
%        el   = Wy - X b, (residuals from a regression)
%        W    = spatial weight matrix
% ---------------------------------------------------
% RETURNS: a  scalar equal to minus the log-likelihood
%          function value at the parameter rho
% ---------------------------------------------------
n = length(eo); spparms('tight');
z = speye(n) - 0.1*sparse(W); p = colmmd(z);
z = speye(n) - rho*sparse(W);
[l,u] = lu(z(:,p));
detval = sum(log(abs(diag(u))));
epe = (eo-rho*el)'*(eo-rho*el);
llike = (n/2)*log(pi) + (n/2)*log(epe/n) - detval;
```

The function **sar** is fairly similar to our **far** function, but we present part of the code below. Code for the information matrix and numerical hessian estimates was eliminated to conserve on space. As in the case of the FAR model, information matrix estimates are computed for cases with less than 500 observations and the hessian approach is used in larger problems. Note the use of the MATLAB 'slash' operation on 'x' and 'y' to solve the least-squares estimation problems. This operator invokes Gaussian elimination to solve the problem: $(X'X)\beta = X'y$.

```
function results = sar(y,x,W,lmin,lmax,convg,maxit)
% PURPOSE: computes spatial autoregressive model estimates
%          y = p*W*y + X*b + e, using sparse matrix algorithms
% ---------------------------------------------------
% USAGE: results = sar(y,x,W,rmin,rmax,convg,maxit)
% where:  y = dependent variable vector
%         x = explanatory variables matrix
%         W = standardized contiguity matrix
%      rmin = (optional) minimum value of rho to use in search
%      rmax = (optional) maximum value of rho to use in search
%     convg = (optional) convergence criterion (default = 1e-8)
%     maxit = (optional) maximum # of iterations (default = 500)
% ---------------------------------------------------
% RETURNS: a structure
%        results.meth  = 'sar'
%        results.beta  = bhat
%        results.rho   = rho
%        results.tstat = asymp t-stat (last entry is rho)
%        results.yhat  = yhat
%        results.resid = residuals
%        results.sige  = sige = (y-p*W*y-x*b)'*(y-p*W*y-x*b)/n
%        results.rsqr  = rsquared
%        results.rbar  = rbar-squared
%        results.lik   = -log likelihood
%        results.nobs  = # of observations
%        results.nvar  = # of explanatory variables in x
%        results.y     = y data vector
%        results.iter  = # of iterations taken
%        results.rmax  = 1/max eigenvalue of W (or rmax if input)
%        results.rmin  = 1/min eigenvalue of W (or rmin if input)
%  ---------------------------------------------------
options = zeros(1,18); rflag = 0;
options(1,1) = 0; options(1,2) = 1.e-6; options(14) = 500;
if nargin == 3,                % use default options
elseif nargin == 5, rflag = 1; % set user supplied lmin,lmax
elseif nargin == 6, options(1,2) = convg; % set converg option
elseif nargin == 7 % set user supplied convg and maxit options
options(1,2) = convg; options(1,14) = maxit;
else, error('Wrong # of arguments to sar');
end;
[n nvar] = size(x); [n1 n2] = size(W);
if n1 ~= n2,    error('sar: wrong size weight matrix W');
elseif n1 ~= n, error('sar: wrong size weight matrix W');
end;
results.y = y; results.nobs = n; results.nvar = nvar;
results.meth = 'sar';
if rflag == 0,
opt.tol = 1e-3; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
lmin = 1/lambda(2);        lmax = 1/lambda(1);
end;
results.rmax = lmax;      results.rmin = lmin;
% step 1) do regressions
Wy = sparse(W)*y;
bo = x\y; bl = x\Wy; eo = y - x*bo; el = Wy - x*bl;
```

```
% step 2) maximize concentrated likelihood function;
[p options] = fmin('f_sar',lmin,lmax,options,eo,el,W);
if options(10) == options(14),
fprintf(1,'sar: convergence not obtained in %4d iterations',options(14));
else, results.iter = options(10);
end;
% step 3) find b,sige maximum likelihood estimates
results.beta = bo - p*bl; results.rho = p; bhat = results.beta;
results.sige = (1/n)*(eo-p*el)'*(eo-p*el); sige = results.sige;
results.yhat = p*Wy + x*results.beta;
results.resid = y - results.yhat;
```

The other point to note is that we have a function named **f2_sar** (not presented) that evaluates the non-concentrated log likelihood function for our **hessian** routine to compute the numerical measures of dispersion.

3.2.2 Applied examples

As an illustration of using the **sar** function, consider the program in example 3.4, where we estimate a model to explain variation in votes casts on a per capita basis in the 3,107 counties using the Pace and Barry (1997) data set. The explanatory variables in the model were: the proportion of population with high school level education or higher, the proportion of the population that are homeowners and the income per capita. Note that the population deflater used to convert the variables to per capita terms was the population 18 years or older in the county.

```
% ----- Example 3.4 Using the sar() function with a large data set
load elect.dat;             % load data on votes in 3,107 counties
y =  log(elect(:,7)./elect(:,8)); % convert to per capita variables
x1 = log(elect(:,9)./elect(:,8)); % education
x2 = log(elect(:,10)./elect(:,8));% homeownership
x3 = log(elect(:,11)./elect(:,8));% income
n = length(y); x = [ones(n,1) x1 x2 x3];
clear x1; clear x2; clear x3;
clear elect;                % conserve on RAM memory
load ford.dat; % 1st order contiguity matrix stored in sparse matrix form
ii = ford(:,1); jj = ford(:,2); ss = ford(:,3);
n = 3107;
clear ford; % clear ford matrix to save RAM memory
W = sparse(ii,jj,ss,n,n);
clear ii; clear jj; clear ss; % conserve on RAM memory
vnames = strvcat('voters','const','educ','homeowners','income');
rmin = -1.0710; rmax = 1;
to = clock;
res = sar(y,x,W,rmin,rmax);
etime(clock,to)
prt(res,vnames);
```

Since we already determined the minimum and maximum eigenvalues for the spatial weight matrix in this problem when estimating the FAR model, we can enter these as inputs to the function **sar** to speed the computations. We use the MATLAB **clock** function as well as **etime** to determine the overall execution time needed to solve this problem, which was 150 seconds. Setting the search range for ρ to 0,1 reduced this time to 103 seconds. The estimation results are presented below:

```
Spatial autoregressive Model Estimates
Dependent Variable =       voters
R-squared       =    0.6356
Rbar-squared    =    0.6352
sigma^2         =    0.0143
Nobs, Nvars     =   3107,     4
log-likelihood  =        3159.4467
# of iterations =     11
min and max rho =   -1.0710,   1.0000
***************************************************************
Variable        Coefficient  Asymptot t-stat    z-probability
const              0.649079        15.363781         0.000000
educ               0.254021        16.117196         0.000000
homeowners         0.476135        32.152225         0.000000
income            -0.117354        -7.036558         0.000000
rho                0.528857        36.204637         0.000000
```

We see from the results that all of the explanatory variables exhibit a significant effect on the variable we wished to explain. The results also indicate that the dependent variable y exhibits strong spatial dependence even after taking the effect of the explanatory variables in X into account. Since all variables are in log form, the coefficient estimates can be interpreted as elasticities. The estimates indicate that homeownership exerts the greatest influence on voter turnout. The elasticity estimate suggests a 10% increase in homeownership would lead to a 4.7% increase in voter turnout.

In Section 3.1 we discussed the fact that ordinary least-squares estimates are biased and inconsistent in the face of sample data that exhibits spatial dependence. As an illustration of the bias associated with least-squares estimation of spatial autoregressive models, we present the following example based on a spatial sample of 88 observations for counties in the state of Ohio. A sample of average housing values for each of 88 counties in Ohio will be related to population per square mile, the number of households and unemployment rates in each county. This regression relationship can be written as:

$$HOUSE_i = \alpha + \beta POP_i + \gamma HOUSEHOLDS_i + \delta UNEMPLOY_i + \varepsilon_i \quad (3.14)$$

The motivation for the regression relationship is that population and household density as well as unemployment rates work to determine house values in each county. Suburban sprawl and the notion of urban rent gradients suggest that housing values in contiguous counties should be related. The least-squares relationship in (3.14) ignores the spatial contiguity information, whereas the SAR model would allow for this type of variation in the model.

The first task is to construct a spatial contiguity matrix for use with our spatial autoregressive model. This could be accomplished by examining a map of the 88 counties and recording neighboring tracts for every observation, a very tedious task. An alternative is to use the latitude and longitude coordinates to construct a contiguity matrix. We rely on a function **xy2cont** that carries out this task. This function is part of Pace and Barry's Spatial Statistics Toolbox for MATLAB, but has been modified to fit the documentation conventions of the spatial econometrics library. The function documentation is shown below:

```
PURPOSE: uses x,y coord to produce spatial contiguity weight matrices
         with delaunay routine from MATLAB version 5.2
------------------------------------------------------
USAGE: [w1 w2 w3] = xy2cont(xcoord,ycoord)
where:     xcoord = x-direction coordinate vector (nobs x 1)
           ycoord = y-direction coordinate vector (nobs x 1)
------------------------------------------------------
RETURNS: w1 = W*W*S, a row-stochastic spatial weight matrix
         w2 = W*S*W, a symmetric spatial weight matrix (max(eig)=1)
         w3 = diagonal matrix with i,i equal to 1/sqrt(sum of ith row)
------------------------------------------------------
References: Kelley Pace, Spatial Statistics Toolbox 1.0
------------------------------------------------------
```

This function essentially uses triangles connecting the x-y coordinates in space to deduce contiguous entities. As an example of using the function, consider constructing a spatial contiguity matrix for the Columbus neighborhood crime data set where we know both the first-order contiguity structure taken from a map of the neighborhoods as well as the x-y coordinates. Here is a program to generate the first-order contiguity matrix from the latitude and longitude coordinates and produce a graphical comparison of the two contiguity structures shown in Figure 3.2.

The errors made by **xy2cont** are largely a result of the irregular shape of the Columbus neighborhood data set. For a state like Ohio that is relatively square, we would expect to see fewer errors.

```
% ----- Example 3.5 Using the xy2cont() function
load anselin.data;  % Columbus neighborhood crime
xc = anselin(:,5);  % longitude coordinate
yc = anselin(:,4);  % latitude coordinate
load Wmat.data;     % load standardized contiguity matrix
% create contiguity matrix from x-y coordinates
[W1 W2 W3] = xy2cont(xc,yc);
% graphically compare the two
spy(W2,'ok'); hold on; spy(Wmat,'+k');
legend('generated','actual');
```

Example 3.6 reads in the data from two files containing a database for the 88 Ohio counties as well as data vectors containing the latitude and longitude information needed to construct a contiguity matrix. We rely on a log transformation of the dependent variable house values to provide better scaling for the data. Note the use of the MATLAB construct: 'ohio2(:,5)./ohio1(:,2)', which divides every element in the column vector 'ohio(:,5)' containing total households in each county by every element in the column vector 'ohio1(:,2)', which contains the population for every county. This produces the number of households per capita for each county as an explanatory variable measuring household density.

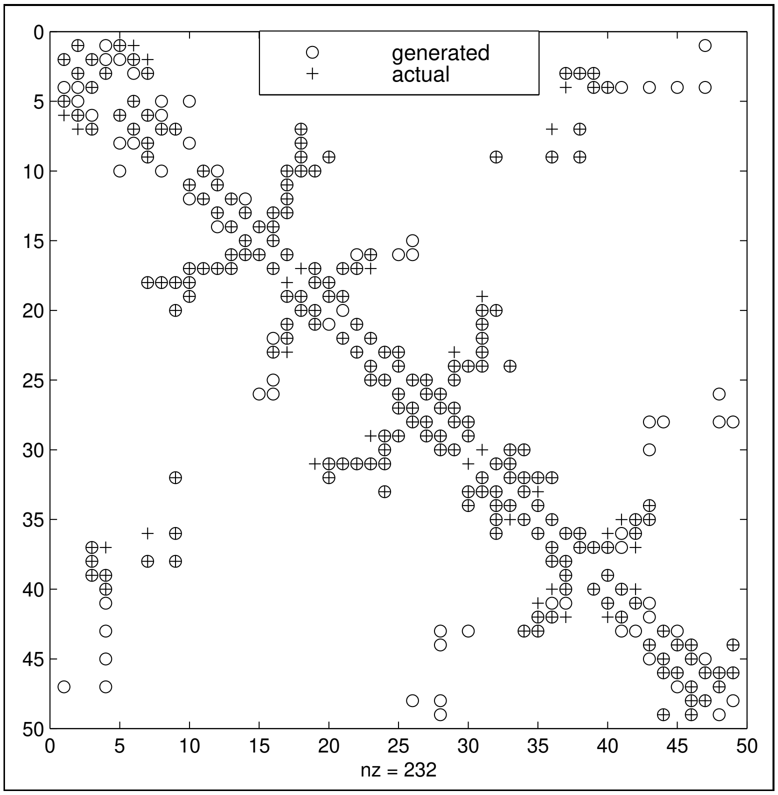

Figure 3.2: Generated contiguity structure results

```
% ----- Example 3.6 Least-squares bias
%       demonstrated with Ohio county data base
load ohio1.dat; % 88 counties (observations)
% 10 columns
% col1  area in square miles
% col2  total population
% col3  population per square mile
% col4  black population
% col5  blacks as a percentage of population
% col6  number of hospitals
% col7  total crimes
% col8  crime rate per capita
% col9  population that are high school graduates
% col10 population that are college graduates
load ohio2.dat; % 88 counties
% 10 columns
% col1  income per capita
% col2  average family income
% col3  families in poverty
% col4  percent of families in poverty
% col5  total number of households
```

```
% col6  average housing value
% col7  unemployment rate
% col8  total manufacturing employment
% col9  manufacturing employment as a percent of total
% col10 total employment
load ohio.xy; % latitude-longitude coordinates of county centroids
[junk W junk2] = xy2cont(ohio(:,1),ohio(:,2)); % make W-matrix
y = log(ohio2(:,6)); n = length(y);
x = [ ones(n,1) ohio1(:,3) ohio2(:,5)./ohio1(:,2) ohio2(:,7) ];
vnames = strvcat('hvalue','constant','popsqm','housedensity','urate');
res = ols(y,x);   prt(res,vnames);
res = sar(y,x,W); prt(res,vnames);
```

The results from these two regressions are shown below. The first point to note is that the spatial autocorrelation coefficient estimate for the SAR model is statistically significant, indicating the presence of spatial autocorrelation in the regression relationship. Least-squares ignores this type of variation producing estimates that lead us to conclude all three explanatory variables are significant in explaining housing values across the 88 county sample. In contrast, the SAR model indicates that population density (popsqm) is not statistically significant at conventional levels. Keep in mind that the OLS estimates are biased and inconsistent, so the inference of significance from OLS is likely to be incorrect.

```
Ordinary Least-squares Estimates
Dependent Variable =     hvalue
R-squared      =    0.6292
Rbar-squared   =    0.6160
sigma^2        =    0.0219
Durbin-Watson  =    2.0992
Nobs, Nvars    =     88,     4
***************************************************************
Variable          Coefficient      t-statistic    t-probability
constant            11.996858        71.173358         0.000000
popsqm               0.000110         2.983046         0.003735
housedensity        -1.597930        -3.344910         0.001232
urate               -0.067693        -7.525022         0.000000

Spatial autoregressive Model Estimates
Dependent Variable =     hvalue
R-squared       =    0.7298
Rbar-squared    =    0.7201
sigma^2         =    0.0153
Nobs, Nvars     =     88,     4
log-likelihood  =        87.284225
# of iterations =     11
min and max rho =   -2.0159,   1.0000
***************************************************************
Variable          Coefficient  Asymptot t-stat    z-probability
constant             6.300142         6.015811         0.000000
popsqm               0.000037         1.170500         0.241800
housedensity        -1.251435        -3.081162         0.002062
urate               -0.055474        -6.933049         0.000000
rho                  0.504132         5.557773         0.000000
```

A second point is that incorporating spatial variation using contiguity improves the fit of the model, raising the R-squared statistic for the SAR model.

Finally, the magnitudes of the OLS parameter estimates indicate that house values are more sensitive to the household density and unemployment rate variables than the SAR model. For example, the OLS estimates imply that a one percentage point increase in the unemployment rate leads to a decrease of 6.76 percent in house values whereas the SAR model places this at 5.54 percent. Similarly, the OLS estimate for household density is considerably larger in magnitude than that from the SAR model.

The point of this illustration is that ignoring information regarding the spatial configuration of the data observations will produce different inferences that may lead to an inappropriate model specification. Anselin and Griffith (1988) also provide examples and show that traditional specification tests are plagued by the presence of spatial autocorrelation, so that we should not rely on these tests in the presence of significant spatial autocorrelation.

3.3 The spatial autoregressive error model

Here we turn attention to the spatial errors model shown in (3.15), where the disturbances exhibit spatial dependence. Anselin (1988) provides a maximum likelihood method for this model which we label SEM here.

$$\begin{aligned} y &= X\beta + u \\ u &= \lambda W u + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.15}$$

The nx1 vector y contains the dependent variable and X represents the usual nxk data matrix containing explanatory variables. W is the spatial weight matrix and the parameter λ is a coefficient on the spatially correlated errors, analogous to the serial correlation problem in time series models. The parameters β reflect the influence of the explanatory variables on variation in the dependent variable y.

We introduce a number of statistical tests that can be used to detect the presence of spatial autocorrelation in the residuals from a least-squares model. Use of these tests will be illustrated in the next section.

The first test for spatial dependence in the disturbances of a regression model is called Moran's $I-$statistic. Given that this test indicates spatial correlation in the least-squares residuals, the SEM model would be an appropriate way to proceed.

Moran's $I-$statistic takes two forms depending on whether the spatial weight matrix W is standardized or not.

1. W not standardized
$$I = (n/s)[e'We]/e'e \tag{3.16}$$

2. W standardized
$$I = e'We/e'e \tag{3.17}$$

Where e represent regression residuals. Cliff and Ord (1972, 1973, 1981) show that the asymptotic distribution for Moran's I based on least-squares residuals correspond to a standard normal distribution after adjusting the $I-$statistic by subtracting the mean and dividing by the standard deviation of the statistic. The adjustment takes two forms depending on whether W is standardized or not. (Anselin, 1988, page 102).

1. W not standardized, let $M = (I - X(X'X)^{-1}X')$ and tr denotes the trace operator.

$$\begin{aligned} E(I) &= (n/s)\mathrm{tr}(MW)/(n-k) \\ V(i) &= (n/s)^2[\mathrm{tr}(MWMW') + \mathrm{tr}(MW)^2 + (\mathrm{tr}(MW))^2]/d - E(I)^2 \\ d &= (n-k)(n-k+2) \\ Z_I &= [I - E(I)]/V(I)^{1/2} \end{aligned} \tag{3.18}$$

2. W standardized

$$\begin{aligned} E(I) &= \mathrm{tr}(MW)/(n-k) \\ V(i) &= [\mathrm{tr}(MWMW') + \mathrm{tr}(MW)^2 + (\mathrm{tr}(MW))^2]/d - E(I)^2 \\ d &= (n-k)(n-k+2) \\ Z_I &= [I - E(I)]/V(I)^{1/2} \end{aligned} \tag{3.19}$$

We implement this test in the MATLAB function **moran**, which takes a regression model and spatial weight matrix W as input and returns a structure variable containing the results from a Moran test for spatial correlation in the residuals. The **prt** function can be used to provide formatted output of the test results. The help documentation for the function is shown below.

```
 PURPOSE: computes Moran's I-statistic for spatial correlation
          in the residuals of a regression model
---------------------------------------------------
 USAGE: result = moran(y,x,W)
 where: y = dependent variable vector
        x = independent variables matrix
        W = contiguity matrix (standardized or unstandardized)
---------------------------------------------------
 RETURNS: a  structure variable
        result.morani = e'*W*e/e'*e (I-statistic)
        result.istat  = [i - E(i)]/std(i), standardized version
        result.imean  = E(i),   expectation
        result.ivar   = var(i), variance
        result.prob   = std normal marginal probability
        result.nobs   = # of observations
        result.nvar   = # of variables in x-matrix
---------------------------------------------------
NOTE: (istat > 1.96, => small prob,
                     => reject H0: of no spatial correlation
---------------------------------------------------
See also: prt(), lmerrors, walds, lratios
---------------------------------------------------
```

A number of other asymptotic approaches exist for testing whether spatial correlation is present in the residuals from a least-squares regression model. Some of these are the likelihood ratio test, the Wald test and a lagrange multiplier test, all of which are based on maximum likelihood estimation of the SEM model.

The likelihood ratio test is based on the difference between the log likelihood from the SEM model and the log likelihood from a least-squares regression. This quantity represents a statistic that is distributed $\chi^2(1)$. A function **lratios** carries out this test and returns a results structure that can be passed to the **prt** function for presentation of the results. Documentation for the function is:

```
 PURPOSE: computes likelihood ratio test for spatial
          correlation in the errors of a regression model
---------------------------------------------------
 USAGE: result = lratios(y,x,W)
    or: result = lratios(y,x,W,sem_result);
 where:   y = dependent variable vector
          x = independent variables matrix
          W = contiguity matrix (standardized or unstandardized)
 sem_result = a results structure from sem()
---------------------------------------------------
 RETURNS: a  structure variable
        result.meth   = 'lratios'
        result.lratio = likelihood ratio statistic
        result.chi1   = 6.635
        result.prob   = marginal probability
        result.nobs   = # of observations
        result.nvar   = # of variables in x-matrix
---------------------------------------------------
NOTES: (lratio > 6.635,  => small prob,
                         => reject HO: of no spatial correlation
       calling the function with a results structure from sem()
       can save time for large models that have already been estimated
---------------------------------------------------
```

Note that we allow the user to supply a 'results' structure variable from the **sem** estimation function, which would save time if the SEM model has already been estimated. This could represent a considerable savings for large problems.

Another approach is based on a Wald test for residual spatial autocorrelation. This test statistic, shown in (3.20), is distributed $\chi^2(1)$. (Anselin, 1988, page 104).

$$\begin{aligned} W &= \lambda^2[t_2 + t_3 - (1/n)(t_1^2)] \sim \chi^2(1) \\ t_1 &= \mathrm{tr}(W. * B^{-1}) \\ t_2 &= \mathrm{tr}(WB^{-1})^2 \\ t_3 &= \mathrm{tr}(WB^{-1})'(WB^{-1}) \end{aligned} \tag{3.20}$$

Where $B = (I_n - \lambda W)$, with the maximum likelihood estimate of λ used, and $.*$ denotes element-by-element matrix multiplication.

We have implemented a MATLAB function **walds**, that carries out this test. The function documentation is shown below:

```
 PURPOSE: Wald statistic for spatial autocorrelation in
          the residuals of a regression model
---------------------------------------------------
 USAGE: result = walds(y,x,W)
 where: y = dependent variable vector
        x = independent variables matrix
        W = contiguity matrix (standardized)
---------------------------------------------------
 RETURNS: a structure variable
        result.meth = 'walds'
        result.wald = Wald statistic
        result.prob = marginal probability
        result.chi1 = 6.635
        result.nobs = # of observations
        result.nvar = # of variables
---------------------------------------------------
NOTE: (wald > 6.635,  => small prob,
                      => reject HO: of no spatial correlation
---------------------------------------------------
See also:  lmerror, lratios, moran
---------------------------------------------------
```

A fourth approach is the Lagrange Multiplier (LM) test which is based on the least-squares residuals and calculations involving the spatial weight matrix W. The LM statistic takes the form: (Anselin, 1988, page 104), where e denote least-squares residuals.

$$\begin{aligned} LM &= (1/T)[(e'We)/\sigma^2]^2 \sim \chi^2(1) \\ T &= \mathrm{tr}(W + W') * W \end{aligned} \tag{3.21}$$

This test is implemented in a MATLAB function **lmerror** with the documentation for the function shown below.

```
 PURPOSE: LM error statistic for spatial correlation in
          the residuals of a regression model
---------------------------------------------------
 USAGE: result = lmerror(y,x,W)
 where: y = dependent variable vector
        x = independent variables matrix
        W = contiguity matrix (standardized)
---------------------------------------------------
 RETURNS: a structure variable
        result.meth = 'lmerror'
        result.lm   = LM statistic
        result.prob = marginal probability
        result.chi1 = 6.635
        result.nobs = # of observations
        result.nvar = # of variables
---------------------------------------------------
NOTE: (lm > 6.635,  => small prob,
                    => reject HO: of no spatial correlation
---------------------------------------------------
See also:  walds, lratios, moran
---------------------------------------------------
```

Finally, a test based on the residuals from the SAR model can be used to examine whether inclusion of the spatial lag term eliminates spatial dependence in the residuals of the model. This test differs from the four tests outlined above in that we allow for the presence of the spatial lag variable in the model. The test for spatial dependence is conditional on having a ρ parameter not equal to zero in the model, rather than relying on least-squares residuals as in the case of the other four tests.

One could view this test as based on the following model:

$$\begin{aligned} y &= \rho C y + X\beta + u \\ u &= \lambda W u + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.22}$$

Where the focus of the test is on whether the parameter $\lambda = 0$. This test statistic is also a Lagrange Multiplier statistic based on: (Anselin, 1988, page 106).

$$\begin{aligned} (e'We/\sigma^2)[T_{22} &- (T_{21})^2 \mathrm{var}(\rho)]^{-1} \sim \chi^2(1) \\ T_{22} &= \mathrm{tr}(W. * W + W'W) \\ T_{21} &= \mathrm{tr}(W. * CA^{-1} + W'CA^{-1}) \end{aligned} \tag{3.23}$$

Where W is the spatial weight matrix shown in (3.22), $A = (I_n - \rho C)$ and $\mathrm{var}(\rho)$ is the maximum likelihood estimate of the variance of the parameter ρ in the model.

We have implemented this test in a MATLAB function **lmsar** with the documentation for the function shown below.

```
 PURPOSE: LM statistic for spatial correlation in the
          residuals of a spatial autoregressive model
---------------------------------------------------
 USAGE: result = lmsar(y,x,W1,W2)
 where: y = dependent variable vector
        x = independent variables matrix
       W1 = contiguity matrix for rho
       W2 = contiguity matrix for lambda
---------------------------------------------------
 RETURNS: a structure variable
        result.meth = 'lmsar'
        result.lm   = LM statistic
        result.prob = marginal probability
        result.chi1 = 6.635
        result.nobs = # of observations
        result.nvar = # of variables
---------------------------------------------------
NOTE: (lm > 6.635,  => small prob,
                    => reject HO: of no spatial correlation
---------------------------------------------------
See also:  walds, lratios, moran, lmerrors
---------------------------------------------------
```

It should be noted that a host of other methods to test for spatial dependence in various modeling situations have been proposed. In addition, the small sample properties of many alternative tests have been compared in Anselin and Florax (1994) and Anselin and Rey (1991).

3.3.1 Computational details

To implement estimation of the spatial error model (SEM) we can draw on the sparse matrix approach we used for the FAR and SAR models. One approach to estimating this model is based on an iterative approach that: 1) constructs least-squares estimates and associated residuals, 2) finds a value of λ that maximizes the log likelihood conditional on the least-squares β values, 3) updates the least-squares values of β using the value of λ determined in step 2). This process is continued until convergence in the residuals.

First, we construct a function **f_sem** to evaluate the concentrated log likelihood function at values of λ, given values for residuals from the least-squares estimation of β. This function **f_sem** is shown below:

```
function lik = f_sem(lam,eD,W)
% PURPOSE: evaluates concentrated log-likelihood for the
%  spatial error model using sparse matrix algorithms
% ---------------------------------------------------
%  USAGE:llike = f_sem(lam,eD,W)
%  where: lam  = spatial error parameter
%         eD   = begls residuals
%         W    = spatial weight matrix
% ---------------------------------------------------
%  RETURNS: a  scalar equal to minus the log-likelihood
%           function value at the parameter lambda
%  --------------------------------------------------
%  SEE ALSO: sem, f_far, f_sac, f_sar
% ---------------------------------------------------
n = length(eD); spparms('tight');
z = speye(n) - 0.1*sparse(W); p = colmmd(z);
z = speye(n) - lam*sparse(W);
[l,u] = lu(z(:,p));
detval = sum(log(abs(diag(u))));
tmp = speye(n) - lam*sparse(W);
epe = eD'*tmp'*tmp*eD;
sige = epe/n; tmp2 = 1/(2*sige);
lik = (n/2)*log(pi) + (n/2)*log(sige) - detval + tmp2*epe;
```

Next, we examine the function **sem** that carries out the iterative estimation process. This is quite similar in approach to the functions **far** and **sar** already described, so we eliminate the code for computing measures of dispersion.

```
function results = sem(y,x,W,lmin,lmax,convg,maxit)
%  PURPOSE: computes spatial error model estimates
%           y = XB + u,  u = L*W*u + e, using sparse algorithms
% ---------------------------------------------------
%  USAGE: results = sem(y,x,W,lmin,lmax,convg,maxit)
%  where: y = dependent variable vector
%         x = independent variables matrix
```

```
%        W = contiguity matrix (standardized)
%     lmin = (optional) minimum value of lambda to use in search
%     lmax = (optional) maximum value of lambda to use in search
%    convg = (optional) convergence criterion (default = 1e-8)
%    maxit = (optional) maximum # of iterations (default = 500)
% ---------------------------------------------------
%  RETURNS: a structure
%        results.meth  = 'sem'
%        results.beta  = bhat
%        results.lam   = L (lambda)
%        results.tstat = asymp t-stats (last entry is lam)
%        results.yhat  = yhat
%        results.resid = residuals
%        results.sige  = sige = e'(I-L*W)'*(I-L*W)*e/n
%        results.rsqr  = rsquared
%        results.rbar  = rbar-squared
%        results.lik   = log likelihood
%        results.nobs  = nobs
%        results.nvar  = nvars (includes lam)
%        results.y     = y data vector
%        results.iter  = # of iterations taken
%        results.lmax  = 1/max eigenvalue of W (or lmax if input)
%        results.lmin  = 1/min eigenvalue of W (or lmin if input)
%  --------------------------------------------------
options = zeros(1,18); rflag = 0;
options(1,1) = 0; options(1,2) = 1.e-6; options(14) = 500;
if nargin == 3, % use default optimization options
elseif nargin == 5, rflag = 1;
elseif nargin == 6, options(1,2) = convg; % set conv option
elseif nargin == 7 % set user supplied convg and maxit options
options(1,2) = convg; options(1,14) = maxit;
else, error('Wrong # of arguments to sem');
end;
[n nvar] = size(x); results.meth = 'sem';
results.y = y; results.nobs = n; results.nvar = nvar;
if rflag == 0, opt.tol = 1e-3; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
lmin = 1/lambda(2);        lmax = 1/lambda(1);
end;
results.lmax = lmax;      results.lmin = lmin;
b0 =x\y;         % step 1) ols of x on y -> b0
eD = y - x*b0;   % step 2) e = y - x*b0
% step 3) find lambda that maximizes Lc
econverge = eD; criteria = 0.0001; converge = 1.0;
iter = 1; itermax = 100;
options = zeros(1,18); options(1,1) = 0; options(1,2) = 1.e-4;
while (converge > criteria & iter < itermax)
[lam, opt] = fmin('f_sem',lmin,lmax,options,eD,W);
tmp = speye(n) - lam*sparse(W); % step 4) find Begls
xs = tmp*x; ys = tmp*y;
begls = xs\ys;
eD = y - x*begls;               % step 5) find Eegls
converge = sum(abs(eD - econverge));
econverge = eD;                 % step 6) check convergence
iter = iter + 1;
end; % end of while loop
if (iter == itermax),
```

```
error('sem: No convergence in sem with 100 iterations');
end;
```

Finally, we construct a function **f2_sem** that evaluates the log likelihood function for the model given maximum likelihood estimates. This function is used by the **hessian** function to determine numerical measures of dispersion in the case of problems involving a large number of observations. This function is:

```
function llike = f2_sem(parm,y,x,W)
% PURPOSE: evaluates log-likelihood -- given ML parameters
%  spatial error model using sparse matrix algorithms
% ---------------------------------------------------
%  USAGE:llike = f2_sem(parm,y,X,W)
%  where: parm = vector of maximum likelihood parameters
%                parm(1:k-2,1) = b, parm(k-1,1) = rho, parm(k,1) = sige
%         y    = dependent variable vector (n x 1)
%         X    = explanatory variables matrix (n x k)
%         W    = spatial weight matrix
% ---------------------------------------------------
%  RETURNS: a  scalar equal to minus the log-likelihood
%           function value at the ML parameters
%  --------------------------------------------------
n = length(y); k = length(parm); b = parm(1:k-2,1);
lam = parm(k-1,1); sige = parm(k,1);
spparms('tight');  z = speye(n) - 0.1*sparse(W);
p = colmmd(z); z = speye(n) - lam*sparse(W);
[l,u] = lu(z(:,p));
detval = sum(log(abs(diag(u))));
eD = y - x*b; epe = eD'*z'*z*eD;
tmp2 = 1/(2*sige);
llike = (n/2)*log(pi) + (n/2)*log(sige) - detval + tmp2*epe;
```

It should be noted that an alternative approach to estimating this model would be to directly maximize the log likelihood function set forth in the function **f2_sem**. This could be accomplished using a general optimization algorithm. It might produce an improvement in speed, depending on how many likelihood function evaluations are needed when solving large problems. We provide an option for doing this in a function **semo** that relies on an optimization function **maxlik** from the *Econometrics Toolbox*. One drawback to this approach is that we cannot impose the restriction on the feasible range for λ because the **maxlik** function carries out unconstrained optimization.

3.3.2 Applied examples

We provide examples of using the functions **moran, lmerror, walds** and **lratios** that test for spatial correlation in the least-squares residuals as well as **lmsar** to test for spatial correlation in the residuals of an SAR model. These examples are based on the Anselin neighborhood crime data set. It should be noted that computation of the Moran $I-$statistic, the LM error statistic, and the Wald test require matrix multiplications involving the large spatial weight matrices C and W. This is not true of the likelihood ratio statistic implemented in the function **lratios**. This test only requires that we compare the likelihood from a least-squares model to that from a spatial error model. As we can produce SEM estimates using our sparse matrix algorithms, this test can be implemented for large models.

Example 3.7 shows a program that carries out all of the test for spatial correlation as well as estimating an SEM model.

```
% ----- Example 3.7 Testing for spatial correlation
load wmat.dat;    % standardized 1st-order contiguity matrix
load anselin.dat; % load Anselin (1988) Columbus neighborhood crime data
y = anselin(:,1); nobs = length(y);
x = [ones(nobs,1) anselin(:,2:3)];
W = wmat;
vnames = strvcat('crime','const','income','house value');
res1 = moran(y,x,W);
prt(res1);
res2 = lmerror(y,x,W);
prt(res2);
res3 = lratios(y,x,W);
prt(res3);
res4 = walds(y,x,W);
prt(res4);
res5 = lmsar(y,x,W,W);
prt(res5);
res = sem(y,x,W);% do sem estimation
prt(res,vnames); % print the output
```

Note that we have provided code in the **prt_spat** function (which is called by **prt**) to produce formatted output of the test results from our spatial correlation testing functions. From the results printed below, we see that the least-squares residuals exhibit spatial correlation. We infer this from the small marginal probabilities that indicate significance at the 99% level of confidence. With regard to the LM error test for spatial correlation in the residuals of the SAR model, we see from the marginal probability of 0.565 that we can reject spatial dependence in the residuals from this model.

```
Moran I-test for spatial correlation in residuals
Moran I                   0.23610178
Moran I-statistic         2.95890622
Marginal Probability      0.00500909
mean                     -0.03329718
standard deviation        0.09104680

LM error tests for spatial correlation in residuals
LM value                  5.74566426
Marginal Probability      0.01652940
chi(1) .01 value         17.61100000

LR tests for spatial correlation in residuals
LR value                  8.01911539
Marginal Probability      0.00462862
chi-squared(1) value      6.63500000

Wald test for spatial correlation in residuals
Wald value               14.72873758
Marginal Probability      0.00012414
```

```
chi(1) .01 value          6.63500000

LM error tests for spatial correlation in SAR model residuals
LM value                  0.33872242
Marginal Probability      0.56056759
chi(1) .01 value          6.63500000

Spatial error Model Estimates
Dependent Variable =      crime
R-squared       =    0.6515
Rbar-squared    =    0.6364
sigma^2         =   95.5675
log-likelihood  =       -166.40057
Nobs, Nvars     =     49,     3
# iterations    =     12
min and max lam =   -1.5362,   1.0000
***************************************************************
Variable         Coefficient  Asymptot t-stat    z-probability
const              59.878750       11.157027         0.000000
income             -0.940247       -2.845229         0.004438
house value        -0.302236       -3.340320         0.000837
lambda              0.562233        4.351068         0.000014
```

As an example of estimating an SEM model on a large data set, we use the Pace and Barry data set with the same model used to demonstrate the SAR estimation procedure. Here again, we can take advantage of the option to supply minimum and maximum values for the optimization search and rely on the values of -1.017 and 1 calculated in example 3.3.

```
% ----- Example 3.8 Using the sem() function with a large data set
load elect.dat;             % load data on votes in 3,107 counties
y =  log(elect(:,7)./elect(:,8)); % convert to per capita variables
x1 = log(elect(:,9)./elect(:,8)); % education
x2 = log(elect(:,10)./elect(:,8));% homeownership
x3 = log(elect(:,11)./elect(:,8));% income
n = length(y); x = [ones(n,1) x1 x2 x3];
clear x1; clear x2; clear x3;
clear elect;                % conserve on RAM memory
load ford.dat; % 1st order contiguity matrix stored in sparse matrix form
ii = ford(:,1); jj = ford(:,2); ss = ford(:,3);
n = 3107;
clear ford; % clear ford matrix to save RAM memory
W = sparse(ii,jj,ss,n,n);
clear ii; clear jj; clear ss; % conserve on RAM memory
vnames = strvcat('voters','const','educ','homeowners','income');
rmin = -1.0710; rmax = 1;
to = clock;
res = sem(y,x,W,rmin,rmax);
etime(clock,to)
prt(res,vnames);
```

We computed estimates using both the iterative procedure in the function **sem** and the optimization procedure implemented in the function **semo**. The time required for the optimization procedure was 299 seconds, which compared to 274 seconds for the iterative procedure. The optimization approach required only 7 iterations whereas the iterative procedure required 10. Needless to say, almost all of the time is spent in the log likelihood function evaluations, so we might think the iterative procedure would be slower. The fact that **maxlik** is only optimizing over two parameters in the SEM function, and five in the case of SEMO must account for the difference in time.

We present the estimates from both approaches to demonstrate that they produce estimates that are identical to 3 decimal places. Both of these functions are part of the spatial econometrics library, as it may be the case that the optimization approach would produce estimates in less time than the iterative approach in other applications. This would likely be the case if very good initial estimates were available as starting values.

```
% estimates from iterative approach using sem() function
Spatial error Model Estimates
Dependent Variable =       voters
R-squared       =    0.6606
Rbar-squared    =    0.6603
sigma^2         =    0.0133
log-likelihood  =        3202.7211
Nobs, Nvars     =   3107,     4
# iterations    =     11
min and max lam =   -1.0710,   1.0000
***************************************************************
Variable         Coefficient  Asymptot t-stat    z-probability
const              0.543129        8.769040         0.000000
educ               0.293303       12.065152         0.000000
homeowners         0.571474       36.435109         0.000000
income            -0.152842       -6.827930         0.000000
lambda             0.650523       41.011556         0.000000
% estimates from optimization approach using semo() function
Spatial error Model Estimates
Dependent Variable =       voters
R-squared       =    0.6606
Rbar-squared    =    0.6603
sigma^2         =    0.0133
log-likelihood  =        3202.7208
Nobs, Nvars     =   3107,     4
# iterations    =      5
***************************************************************
Variable         Coefficient  Asymptot t-stat    z-probability
const              0.543175        8.770178         0.000000
educ               0.293231       12.061955         0.000000
homeowners         0.571494       36.436805         0.000000
income            -0.152815       -6.826670         0.000000
lambda             0.650574       41.019490         0.000000
```

The SEM estimates indicate that after taking into account the influence of the explanatory variables, we still have spatial correlation in the residuals of the model because the parameter λ is significantly different from zero. As a confirmation of this, consider the results from an LR test implemented with the function **lratios** shown below:

```
LR tests for spatial correlation in residuals
LR value                1163.01773404
```

```
Marginal Probability      0.00000000
chi-squared(1) value      6.63500000
```

Recall that this is a test of spatial autocorrelation in the residuals from a least-squares model, and the test results provide a strong indication of spatial dependence in the least-squares residuals. Note also that this is the only test from those described in Section 3.3 that can be implemented successfully with large data sets. An alternative to these tests for large problems would be to estimate a FAR model using the residuals from least-squares.

3.4 The spatial Durbin model

The model shown in (3.24) is referred to as a spatial Durbin model by Anselin (1988) due to the analogy with a suggestion by Durbin for the case of a time series model with residual autocorrelation.

$$\begin{aligned} (I_n - \rho W)y &= (I_n - \rho W)X\beta + \varepsilon \\ y &= \rho Wy + X\beta - \rho WX\beta + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.24}$$

There is a formal equivalence of this model to a model:

$$\begin{aligned} y &= X\beta + (I_n - \rho W)^{-1}\varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.25}$$

We implement a variant of this model which we label SDM that is shown in (3.26), where y contains an nx1 vector of dependent variables and X represents the usual nxk data matrix containing explanatory variables with an associated parameter vector β_1. W is the spatial weight matrix and the parameter ρ is a coefficient on the spatial lag of the dependent variable. An additional set of explanatory variables is added to the model by constructing a spatial lag of the explanatory variables using the matrix product WX, with associated parameters β_2. This set of variables represent explanatory variables constructed as averages from neighboring observations.

$$\begin{aligned} y &= \rho Wy + X\beta_1 + WX\beta_2 + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2 I_n) \end{aligned} \tag{3.26}$$

Note that the model in (3.24) could be viewed as imposing a restriction that $\beta_2 = -\rho\beta_1$ on the model in (3.26). This model has been used in spatial econometric analysis by Pace and Barry (1998) and we incorporate a function **sdm** in our spatial econometrics library to provide maximum likelihood estimates. Noting that β_1 and β_2 can be expressed as:

$$
\begin{aligned}
\beta_1 &= (\tilde{X}'\tilde{X})^{-1}\tilde{X}'y \qquad (3.27)\\
\beta_2 &= (\tilde{X}'\tilde{X})^{-1}\tilde{X}'Wy
\end{aligned}
$$

we can write the concentrated log-likelihood function for this model as shown in (3.28) where C denotes an inessential constant.

$$
\begin{aligned}
ln(L) &= C + ln|I_n - \rho W| - (n/2)ln(e_1'e_1 - 2\rho e_2'e_1 + \rho^2 e_2'e_2) \qquad (3.28)\\
e_1 &= y - \tilde{X}\beta_1\\
e_2 &= Wy - \tilde{X}\beta_2\\
\tilde{X} &= [X \quad WX]
\end{aligned}
$$

Given a value of ρ that maximizes the concentrated likelihood function (say $\hat{\rho}$), we compute estimates for β_1 and β_2 in (3.26) using:

$$
\hat{\beta} = (\beta_1 - \hat{\rho}\beta_2) = \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} \qquad (3.29)
$$

Finally, an estimate of σ_ε^2 is calculated using:

$$
\begin{aligned}
\hat{\sigma}^2 &= (y - \hat{\rho}Wy - \tilde{X}\hat{\beta})'(y - \hat{\rho}Wy - \tilde{X}\hat{\beta})/n \qquad (3.30)\\
& \qquad (3.31)
\end{aligned}
$$

A point to note regarding use of this model is that the explanatory variables matrix $[X \quad WX]$ can suffer from severe collinearity problems in some applications. A function **bkw** from the *Econometrics Toolbox* can be used to diagnose collinear relationships in this matrix with the methods set forth in Belsley, Kuh and Welsch (1980). Chapter 3 of the manual for the *Econometrics Toolbox* discusses the use of this function for diagnosing collinearity.

3.4.1 Computational details

We use the function **f_sdm** to evaluate the concentrated log-likelihood function, which depends on the single parameter ρ from the model.

```
function llike = f_sdm(rho,y,x,W)
% PURPOSE: evaluates concentrated log-likelihood for the
%          spatial durbin model using sparse matrix algorithms
% ---------------------------------------------------
%  USAGE:llike = f_sdm(rho,y,x,W)
%  where: rho  = spatial autoregressive parameter
%         y    = dependent variable vector
%         x    = data matrix
%         W    = spatial weight matrix
% ---------------------------------------------------
%  RETURNS: a  scalar equal to minus the log-likelihood
```

```
%           function value at the parameter rho
% ---------------------------------------------------
[n k] = size(x); rho2 = rho*rho;
spparms('tight');
z = speye(n) - 0.1*sparse(W);
p = colmmd(z);
z = speye(n) - rho*sparse(W);
[l,u] = lu(z(:,p));
detval = sum(log(abs(diag(u))));
dy=W*y; xdx=[ x(:,2:k) W*x(:,2:k) ones(n,1)];
xdxtxdx=(xdx'*xdx); xdxinv=inv(xdxtxdx);
xdxy=xdx'*y; xdxdy=xdx'*dy;
bmat=xdxtxdx\[xdxy xdxdy];
bols=bmat(:,1); bolsd=bmat(:,2);
eo=y-xdx*bols; ed=dy-xdx*bolsd;
e2o=(eo'*eo); edo=(ed'*eo); e2d=(ed'*ed);
logsse=log(e2o-2*rho*edo+rho2*e2d);
llike = (n/2)*log(pi) -detval + (n/2)*logsse;
```

The function **sdm** carries out estimation using the simplex optimization algorithm to maximize the concentrated likelihood function in **f_sdm**. This function relies totally on a numerical hessian matrix approach to calculating measures of dispersion for the estimates. (This is because problems were encountered with the information matrix approach which may be due to incorrect published formulas for the expressions needed.) Code for computing measures of dispersion has been eliminated to save space.

```
function results = sdm(y,x,W,lmin,lmax,convg,maxit)
% PURPOSE: computes spatial durbin model estimates
%         (I-rho*W)y = a + X*B1 + W*X*B2 + e, using sparse algorithms
% ---------------------------------------------------
% USAGE: results = sdm(y,x,W,rmin,rmax,convg,maxit)
% where: y = dependent variable vector
%        x = independent variables matrix
%        W = contiguity matrix (standardized)
%     rmin = (optional) minimum value of rho to use in search
%     rmax = (optional) maximum value of rho to use in search
%    convg = (optional) convergence criterion (default = 1e-8)
%    maxit = (optional) maximum # of iterations (default = 500)
% ---------------------------------------------------
% RETURNS: a structure
%        results.meth  = 'sdm'
%        results.beta  = bhat [a B1 B2]' a k+(k-1) x 1 vector
%        results.rho   = rho
%        results.tstat = t-statistics (last entry is rho)
%        results.yhat  = yhat
%        results.resid = residuals
%        results.sige  = sige
%        results.rsqr  = rsquared
%        results.rbar  = rbar-squared
%        results.lik   = log likelihood
%        results.nobs  = nobs
%        results.nvar  = nvars (includes lam)
%        results.y     = y data vector
%        results.iter  = # of iterations taken
%        results.rmax  = 1/max eigenvalue of W (or rmax if input)
```

```
%        results.rmin  = 1/min eigenvalue of W (or rmin if input)
% -------------------------------------------------
% NOTES: constant term should be in the 1st column of the x-matrix
%        constant is excluded from B2 estimates
% -------------------------------------------------
options = zeros(1,18); rflag = 0;
options(1,1) = 0; options(1,2) = 1.e-6; options(14) = 500;
if nargin == 3 % use default optimization options
elseif nargin == 5, rflag = 1;
elseif nargin == 6, options(1,2) = convg;  % set convg option
elseif nargin == 7 % set user supplied convg and maxit options
options(1,2) = convg; options(1,14) = maxit;
else, error('Wrong # of arguments to sdm');
end;
[n nvar] = size(x); results.meth = 'sdm';
results.y = y; results.nobs = n; results.nvar = nvar;
if rflag == 0, opt.tol = 1e-3; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
lmin = 1/lambda(2);       lmax = 1/lambda(1);
end;
results.rmax = lmax;      results.rmin = lmin;
[rho, options] = fmin('f_sdm',lmin,lmax,options,y,x,W);
if options(10) == options(14),
fprintf(1,'sdm: convergence not obtained in %4d iterations',options(14));
else, results.iter = options(10);
end;
rho2 = rho*rho; dy=W*y; % find beta hats
xdx=[ x(:,2:nvar) W*x(:,2:nvar) ones(n,1)];
xdxtxdx=(xdx'*xdx); xdxinv=inv(xdxtxdx); xdxy=xdx'*y;
xdxdy=xdx'*dy; bmat=xdxtxdx\[xdxy xdxdy]; bols=bmat(:,1);
bolsd=bmat(:,2); beta = bols - rho*bolsd;
results.yhat = xdx*beta + rho*sparse(W)*y;
eo=y-xdx*bols; ed=dy-xdx*bolsd; e2o=(eo'*eo);
edo=(ed'*eo); e2d=(ed'*ed);
epe = (e2o-2*rho*edo+rho2*e2d); sige = epe/n;
results.rho = rho; results.resid = y - results.yhat;
results.sige = sige;
```

This function requires that the user supply a constant term in the first column of the explanatory variables matrix in the model. Because we exclude the constant from the second expression, WX in the augmented matrix $\tilde{X} = [X \quad WX]$, we need to know which column contains the constant term in the matrix X input to the function. It seems simplest to require that it be placed in the first column.

3.4.2 Applied examples

We illustrate the SDM model using Anselin's neighborhood crime data set in example 3.9.

```
% ----- Example 3.9 Using the sdm() function
load anselin.dat; % load Anselin (1988) Columbus neighborhood crime data
y = anselin(:,1); n = length(y); x = [ones(n,1) anselin(:,2:3)];
load Wmat.dat;    % load Anselin (1988) 1st order contiguity matrix
W = Wmat;
```

```
vnames = strvcat('crime','constant','income','hvalue');
% do sdm regression
res = sdm(y,x,W);
prt(res,vnames);
```

The results shown below indicate that neither of the spatially lagged explanatory variables labeled 'W-income' and 'W-hvalue' in the printed output are statistically significant at conventional levels.

```
Spatial Durbin model
Dependent Variable =        crime
R-squared       =    0.6652
Rbar-squared    =    0.6348
sigma^2         =   91.8092
log-likelihood  =       -164.41996
Nobs, Nvars     =     49,     3
# iterations    =      9
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable        Coefficient  Asymptot t-stat    z-probability
constant          42.780315         3.075397         0.002102
income            -0.913522        -2.716256         0.006602
hvalue            -0.293785        -3.305459         0.000948
W-income          -0.518148        -0.871550         0.383454
W-hvalue           0.245193         1.386415         0.165620
rho                0.426971         2.551811         0.010716
```

As a large scale problem we implement the SDM model with the Pace and Barry data set in example 3.10.

```
% ----- Example 3.10 Using the sdm() function with a large sample
load elect.dat;             % load data on votes
y =  log(elect(:,7)./elect(:,8));
x1 = log(elect(:,9)./elect(:,8));
x2 = log(elect(:,10)./elect(:,8));
x3 = log(elect(:,11)./elect(:,8));
n = length(y); x = [ones(n,1) x1 x2 x3];
clear x1; clear x2; clear x3;
clear elect;                % conserve on RAM memory
load ford.dat; % 1st order contiguity matrix stored in sparse matrix form
ii = ford(:,1); jj = ford(:,2); ss = ford(:,3);
n = 3107;
clear ford; % clear ford matrix to save RAM memory
W = sparse(ii,jj,ss,n,n);
clear ii; clear jj; clear ss; % conserve on RAM memory
vnames = strvcat('voters','const','educ','homeowners','income');
rmin = -1.0710; rmax = 1;
to = clock;
res = sdm(y,x,W,rmin,rmax);
etime(clock,to)
prt(res,vnames);
```

This estimation problem took 187 seconds and produced the following results.

```
Spatial Durbin model
Dependent Variable =       voters
R-squared       =    0.6696
Rbar-squared    =    0.6690
sigma^2         =    0.0130
log-likelihood  =        3275.3805
Nobs, Nvars     =   3107,     4
# iterations    =     10
min and max rho =   -1.0710,   1.0000
***************************************************************
Variable          Coefficient  Asymptot t-stat    z-probability
const                0.524818        10.264344         0.000000
educ                 0.154564         6.207306         0.000000
homeowners           0.575636        37.112911         0.000000
income              -0.090330        -3.962587         0.000074
W-educ               0.116203         3.920690         0.000088
W-homeowners        -0.362079       -15.338338         0.000000
W-income            -0.069325        -2.396691         0.016544
rho                  0.599802        38.409667         0.000000
```

Here we see that all of the spatially lagged explanatory variables were statistically significant. The negative sign on the coefficient for 'W-homeowners' suggest that an increase in the number of homeowners in neighboring counties leads to a decrease in voter participation. This is in contrast to the direct elasticity for homeownership on voter participation, which exhibits a positive sign. The total effect of homeownership is still positive as the sum of the two coefficients on homeownership produce a positive magnitude.

We defer discussion of comparisons between spatial autoregressive model specifications until the last section of Chapter 4, where an applied exercise deals with comparing models.

3.5 The general spatial model

A general version of the spatial model that we label SAC includes both the spatial lag term and a spatially correlated error structure as shown in (3.32).

$$\begin{aligned} y &= \rho W_1 y + X\beta + u \\ u &= \lambda W_2 u + \varepsilon \\ \varepsilon &\sim N(0, \sigma_\varepsilon^2) \end{aligned} \tag{3.32}$$

One point to note about this model is that W_1 can equal W_2, but there may be identification problems in this case. The log likelihood for this model can be maximized using our general optimization algorithm on a concentrated version of the likelihood function. The parameters β and σ^2 are concentrated out of the likelihood function, leaving the parameters ρ and λ. This eliminates the ability to use the univariate simplex optimization algorithm **fmin** that we used with the other spatial autoregressive models.

We can still produce a sparse matrix algorithm for the log likelihood function and proceed in a similar fashion to that used for the other spatial autoregressive models. One difference is that we cannot easily restrict the parameters ρ and λ to lie within the ranges defined by the maximum and minimum eigenvalues from their associated weight matrices W_1 and W_2. In the next chapter, we will show how these constraints can be easily imposed in a Bayesian setting that relies on Gibbs sampling estimation of this model.

When might one rely on this model? If there were evidence that spatial dependence existed in the error structure from a spatial autoregressive (SAR) model, the SAC would represent an appropriate approach to modeling this type of dependence in the errors. Recall, we can use the LM-test implemented in the function **lmsars** to see if spatial dependence exists in the residuals of an SAR model.

We might also rely on this model if we believe that the disturbance structure involves higher-order spatial dependence, perhaps due to second-round effects of a spatial phenomena being modeled. The function **slag** discussed in the next section can be used to generate a second order spatial weight matrix which can be added to the model.

A third example of using matrices W_1 and W_2 might be where W_1 represented a first-order contiguity matrix and W_2 was constructed as a diagonal matrix measuring the inverse distance from the central city. This type of configuration of the spatial weight matrices expresses a belief that contiguity alone does not suffice to capture the spatial effects. The distance from the central city may also represent an important factor in the phenomena we are modeling.

This raises the identification issue, should we use the distance weighting matrix in place of W_1 and the first-order contiguity matrix for W_2, or rely on the opposite configuration? Of course, comparing likelihood function values along with the statistical significance of the parameters ρ and λ from models estimated using both configurations might point to a clear answer.

The log likelihood function for this model is:

$$\begin{aligned} L &= C - (n/2)ln(\sigma^2) + ln(|A|) + ln(|B|) - (1/2\sigma^2)(e'B'Be) \\ e &= (Ay - X\beta) \\ A &= (I_n - \rho W_1) \\ B &= (I_n - \lambda W_2) \end{aligned} \tag{3.33}$$

We concentrate the function using the following expressions for β and σ^2:

$$\begin{aligned} \beta &= (X'A'AX)^{-1}(X'A'ABy) \\ e &= By - X\beta \\ \sigma^2 &= (e'e)/n \end{aligned} \tag{3.34}$$

Using the expressions in (3.34), we can evaluate the log likelihood for values of ρ and λ. The values of the other parameters β and σ^2 are calculated as a function of the maximum likelihood values of ρ, λ and the sample data in y, X.

3.5.1 Computational details

Our first task is to construct a function **f_sac** that provides a sparse matrix evaluation of the concentrated log likelihood function. This is shown below:

```
function llike = f_sac(parm,y,x,W1,W2)
% PURPOSE: evaluates log-likelihood for general spatial model
%    y = rho*W1*y + X*b + u,   u = lam*W2*u + e,
%        using sparse matrix algorithms
% ---------------------------------------------------
%  USAGE:llike = f_sac(parm,y,x,W1,W2)
%  where: parm  = (rho,lam)
%         y     = dependendent variable vector
%         x     = explanatory variables matrix
%         W1    = spatial lag weight matrix
%         W2    = spatial error weight matrix
% ---------------------------------------------------
%  RETURNS: a  scalar equal to minus the log-likelihood
%           function value at the parameters rho, lambda
% --------------------------------------------------
%  NOTE: b,sige is concentrated out of the log-likelihood
% --------------------------------------------------
%  SEE ALSO: sar, f_far, f_sar, f_sem
% ---------------------------------------------------
lam = parm(1,1); rho = parm(2,1); n = length(y);
z = speye(n) - 0.1*sparse(W1); p = colmmd(z);
z1 = speye(n) - rho*sparse(W1);
[l,u] = lu(z1(:,p)); detval1 = sum(log(abs(diag(u))));
z = speye(n) - 0.1*sparse(W2); p = colmmd(z);
z2 = speye(n) - lam*sparse(W2);
[l,u] = lu(z2(:,p)); detval2 = sum(log(abs(diag(u))));
b = (x'*z1'*z1*x)\(x'*z1'*z1*z2*y);
eD = z2*y-x*b;  epe = eD'*z1'*z1*eD; sige = epe/n; tmp2 = 1/(2*sige);
llike = (n/2)*log(pi) + (n/2)*log(sige) - detval1 - detval2 + tmp2*epe;
```

Note that we need to solve for two determinants in this problem which should increase the computational intensity and slow down the estimation procedure. Another point is that we attempt to re-use matrices like **z** and **p** to conserve on RAM memory.

The next task is to construct our MATLAB function **sac** that solves the the non-linear optimization problem using this log likelihood function. There are a number of things to note about this function. First, we provide optimization options for the user in the form of a structure variable 'info'. These options allow the user to control some aspects of the **maxlik** optimization algorithm and to print intermediate results while optimization is proceeding.

```
function results = sac(y,x,W1,W2,info)
%  PURPOSE: computes general Spatial Model
%  model: y = p*W1*y + X*b + u,  u = lam*W2*u + e
% ---------------------------------------------------
%  USAGE: results = sac(y,x,W1,W2)
%  where: y  = dependent variable vector
%         x  = independent variables matrix
%         W1 = spatial weight matrix (standardized)
%         W2 = spatial weight matrix
```

```
%      info        = a structure variable with optimization options
%      info.parm   = starting values for parameters, b, rho, lambda
%      info.convg  = (optional) convergence criterion (default = 1e-7)
%      info.maxit  = (optional) maximum # of iterations (default = 500)
%      info.method = 'bfgs', 'dfp' (default bfgs)
%      info.pflag  = flag for printing of intermediate results
% ---------------------------------------------------
%  RETURNS: a structure
%         results.meth  = 'sac'
%         results.beta  = bhat
%         results.rho   = p (rho)
%         results.lam   = L (lambda)
%         results.tstat = asymptotic t-stats (last 2 are rho,lam)
%         results.yhat  = yhat
%         results.resid = residuals
%         results.sige  = sige = e'(I-L*W)'*(I-L*W)*e/n
%         results.rsqr  = rsquared
%         results.rbar  = rbar-squared
%         results.lik   = likelihood function value
%         results.nobs  = nobs
%         results.nvar  = nvars
%         results.y     = y data vector
%         results.iter  = # of iterations taken
%  --------------------------------------------------
%  SEE ALSO: prt_spat(results), prt
% ---------------------------------------------------
% REFERENCES: Luc Anselin Spatial Econometrics (1988)
%             pages 64-65 and pages 182-183.
% ---------------------------------------------------
if nargin == 5
 if ~isstruct(info), error('sac: the options must be a structure'); end;
opti.maxit = 500; bstart = 0; opti.meth = 'bfgs'; opti.ftol = 1e-7;
opti.pflag = 0; fields = fieldnames(info); nf = length(fields);
 for i=1:nf
    if strcmp(fields{i},'parm'),       bstart = 1; parm = info.parm;
    elseif strcmp(fields{i},'convg'),  opti.ftol = info.convg;
    elseif strcmp(fields{i},'maxit'),  opti.maxit = info.maxit;
    elseif strcmp(fields{i},'method'), opti.meth = info.method;
    elseif strcmp(fields{i},'pflag'),  opti.pflag = info.pflag;
    end; end;
elseif nargin == 4 % use default options
opti.maxit = 500; bstart = 0; opti.meth = 'bfgs'; opti.ftol = 1e-7;
opti.pflag = 0;
else, error('Wrong # of arguments to sac'); end;
[n nvar] = size(x); results.meth = 'sac';
[n1 n2] = size(W1);
if n1 ~= n2, error('sac: wrong size weight matrix W1');
elseif n1 ~= n, error('sac: wrong size weight matrix W1'); end;
[n1 n2] = size(W2);
if n1 ~= n2, error('sac: wrong size weight matrix W2');
elseif n1 ~= n, error('sac: wrong size weight matrix W2'); end;
results.y = y; results.nobs = n; results.nvar = nvar;
if bstart == 0, parm = [0.5
                        0.5]; end;
[pout,f,hessn,gradn,iter,fail]=maxlik('f_sac',parm,opti,y,x,W1,W2);
if fail == 1, fprintf('sac: optimization failure \n'); end;
lam = pout(1,1); rho = pout(2,1); % fill-in results
```

```
A = speye(n) - rho*sparse(W1); B = speye(n) - lam*sparse(W2);
b0 = (x'*A'*A'*x)\(x'*A'*A*B*y); e = A*B*y - A*x*b0;
results.beta = b0; results.rho = rho; results.lam = lam;
```

A second point is that if failure occurs, we simply print a message and let the function continue to process and return a results structure consisting of failed parameter estimates. This approach allows the user to examine the failed estimates and attempt estimation based on alternative optimization options. For example, the user might elect to attempt a Davidson-Fletcher-Powell ('dfp') algorithm in place of the default Broyden-Fletcher-Goldfarb-Smith ('bfgs') routine.

With regard to optimization algorithm failures, it should be noted that the *Econometrics Toolbox* contains alternative optimization functions named **solvopt**, **dfp_min**, **frpr_min**, and **pow_min** that can be used in place of **maxlik**. Any of these functions could be substituted for **maxlik** in the function **sac**. Chapter 10 in the *Econometrics Toolbox* provides examples of using these functions as well as their documentation.

Since this is the first function where we use a structure variable as an input, we might examine how the **sac** function 'parses' this type of input variable. First note that the user can name the input structure variable anything, it is the structure fields that the function examines. The use of a structure variable to input arguments to functions is a useful MATLAB programming construct that we employ in many spatial econometrics functions. To see how this is accomplished consider the following code from **sac** that parses the input structure fields for the structure variable named 'info' in the function declaration.

```
opti.maxit = 500; bstart = 0; opti.meth = 'bfgs';
opti.ftol = 1e-7; opti.pflag = 0;
fields = fieldnames(info);
nf = length(fields);
 for i=1:nf
    if strcmp(fields{i},'parm'),       bstart = 1;
    elseif strcmp(fields{i},'convg'),  opti.ftol = info.convg;
    elseif strcmp(fields{i},'maxit'),  opti.maxit = info.maxit;
    elseif strcmp(fields{i},'method'), opti.meth = info.method;
    elseif strcmp(fields{i},'pflag'),  opti.pflag = info.pflag;
    end;
 end;
```

We rely on a MATLAB function **fieldnames** that extracts the field elements from a structure variable into a cell-array that we name 'fields'. After determining how many elements have been entered by the user with the **length** function, we use a 'for loop' to compare each field element with the valid structure fields using the string comparison function **strcmp**. Input fields specified by the user will be detected by the string comparisons and extracted to overwrite the default values set prior to parsing the input structure variable.

As we encounter functions to implement Bayesian variants of the spatial autoregressive models in Chapter 4, we will make extensive use of structure variables as inputs. This allows us to reduce the number of input arguments, making the function easier to use, and the programs easier to read.

The final task is to construct a non-concentrated version of the log likelihood function that will be used by the **hessian** function to provide numerical estimates of dispersion. A function **f2_sac** provides the non-concentrated log likelihood function.

```
function llike = f2_sac(parm,y,x,W1,W2)
% PURPOSE: evaluates log-likelihood for ML values general spatial model
%    y = rho*W1*y + X*b + u,    u = lam*W2*u + e,
%        using sparse matrix algorithms
% ---------------------------------------------------
%  USAGE:llike = f2_sac(parm,y,x,W1,W2)
%  where: parm  = (beta,rho,lam,sige) ML values
%         y     = dependendent variable vector
%         x     = explanatory variables matrix
%         W1    = spatial lag weight matrix
%         W2    = spatial error weight matrix
% ---------------------------------------------------
%  RETURNS: a  scalar equal to minus the log-likelihood
%           function value at the ML parameters
% --------------------------------------------------
%  SEE ALSO: sac, f2_far, f2_sar, f2_sem
% ---------------------------------------------------
k = length(parm); b = parm(1:k-3,1);
rho = parm(k-2,1); lam = parm(k-1,1); sige = parm(k,1);
n = length(y); z = speye(n) - 0.1*sparse(W1);
p = colmmd(z); z1 = speye(n) - rho*sparse(W1);
[l,u] = lu(z1(:,p)); detval1 = sum(log(abs(diag(u))));
z = speye(n) - 0.1*sparse(W2);
p = colmmd(z); z2 = speye(n) - lam*sparse(W2);
[l,u] = lu(z2(:,p)); detval2 = sum(log(abs(diag(u))));
eD = z2*y-x*b;  epe = eD'*z1'*z1*eD; tmp2 = 1/(2*sige);
llike = (n/2)*log(pi) + (n/2)*log(sige) - detval1 - detval2 + tmp2*epe;
```

The next section turns to illustrating the use of the estimation functions we have constructed for the general spatial autoregressive model.

3.5.2 Applied examples

Our first example illustrates the general spatial model with the Anselin Columbus neighborhood crime data set. We construct a spatial lag matrix W^2 for use in the model. As discussed in Chapter 1, higher-order spatial lags require that we eliminate redundancies that arise. Anselin and Smirnov (1994) provide details regarding the procedures as well as a comparison of alternative algorithms and their relative performance.

A function **slag** can be used to produce higher order spatial lags. The documentation for the function is:

```
PURPOSE: compute spatial lags
---------------------------------------------
USAGE: Wp = slag(W,p)
where: W = input spatial weight matrix, sparse or full
           (0,1 or standardized form)
       p = lag order (an integer)
---------------------------------------------
```

```
  RETURNS: Wp = W^p spatial lag matrix
           in standardized form if W standardized was input
           in 0,1 form in W non-standardized was input
  ---------------------------------------------
```

One point about **slag** is that it returns a standardized contiguity matrix even if a non-standardized matrix is used as an input. This seemed a useful approach to take. There is a function **normw** that standardizes spatial weight matrices so the row-sums are unity. It takes a single input argument containing the non-standardized weight matrix and returns a single argument containing the standardized matrix.

Example 3.11 uses the **sac** function to estimate three alternative models. Our example illustrates the point discussed earlier regarding model specification with respect to the use of W and W^2 by producing estimates for three models based on alternative configurations of these two spatial weight matrices.

```
% ----- Example 3.11 Using the sac function
load Wmat.dat;    % standardized 1st-order contiguity matrix
load anselin.dat; % load Anselin (1988) Columbus neighborhood crime data
y = anselin(:,1); nobs = length(y);
x = [ones(nobs,1) anselin(:,2:3)];
W = Wmat;
vnames = strvcat('crime','const','income','house value');
W2 = slag(W,2); % standardized W2 result from slag
subplot(2,1,1), spy(W);
xlabel('First-order contiguity structure');
subplot(2,1,2), spy(W2);
xlabel('Second-order contiguity structure');
pause;
res1 = sac(y,x,W2,W);% general spatial model W2,W
prt(res1,vnames);    % print the output
res2 = sac(y,x,W,W2);% general spatial model W,W2
prt(res2,vnames);    % print the output
res3 = sac(y,x,W,W); % general spatial model W,W
prt(res3,vnames);    % print the output
plt(res3);
```

The estimation results are shown below for all three versions of the model. The first two models produced estimates that suggest W^2 is not significant, as the coefficients for both ρ and λ are small and insignificant when this contiguity matrix is applied. In contrast, the first-order contiguity matrix is always associated with a significant ρ or λ coefficient in the first two models, indicating the importance of first-order effects.

The third model that uses the first-order W for both ρ and λ produced insignificant coefficients for both of these parameters.

```
General Spatial Model Estimates
Dependent Variable =      crime
R-squared      =    0.6527
Rbar-squared   =    0.6376
sigma^2        =   95.2471
log-likelihood =       -165.36509
```

```
Nobs, Nvars    =     49,     3
# iterations   =      5
***************************************************************
Variable         Coefficient      t-statistic    t-probability
const              45.421239         6.863214         0.000000
income             -1.042733        -3.226112         0.002313
house value        -0.268027        -2.935739         0.005180
rho                -0.094359        -0.392131         0.696773
lambda              0.429926         6.340856         0.000000

General Spatial Model Estimates
Dependent Variable =      crime
R-squared      =    0.6520
Rbar-squared   =    0.6369
sigma^2        =   95.4333
log-likelihood =       -166.39931
Nobs, Nvars    =     49,     3
# iterations   =      5
***************************************************************
Variable         Coefficient      t-statistic    t-probability
const              60.243770         4.965791         0.000010
income             -0.937802        -3.005658         0.004281
house value        -0.302261        -3.406156         0.001377
rho                 0.565853         4.942206         0.000011
lambda             -0.010726        -0.151686         0.880098

General Spatial Model Estimates
Dependent Variable =      crime
R-squared      =    0.6514
Rbar-squared   =    0.6362
sigma^2        =   95.6115
log-likelihood =       -165.25612
Nobs, Nvars    =     49,     3
# iterations   =      7
***************************************************************
Variable         Coefficient      t-statistic    t-probability
const              47.770500         4.338687         0.000078
income             -1.024966        -3.119166         0.003127
house value        -0.281714        -3.109463         0.003213
rho                 0.167197         0.497856         0.620957
lambda              0.368187         1.396173         0.169364
```

By way of summary, I would reject all three SAC model specifications, thinking that the SAR or SEM models (presented below) appear preferable. Note that the second SAC model specification collapses to an SAR model by virtue of the fact that the parameter λ is not significant and the parameter ρ in this model is associated with the first-order weight matrix W.

```
Spatial error Model Estimates
Dependent Variable =      crime
R-squared       =    0.6515
Rbar-squared    =    0.6364
sigma^2         =   95.5675
log-likelihood  =       -166.40057
Nobs, Nvars     =     49,     3
# iterations    =     12
```

```
min and max lam =   -1.5362,   1.0000
***************************************************************
Variable         Coefficient      t-statistic    t-probability
const              59.878750        11.157027         0.000000
income             -0.940247        -2.845229         0.006605
house value        -0.302236        -3.340320         0.001667
lambda              0.562233         4.351068         0.000075

Spatial autoregressive Model Estimates
Dependent Variable =      crime
R-squared       =    0.6518
Rbar-squared    =    0.6366
sigma^2         =   95.5033
Nobs, Nvars     =     49,     3
log-likelihood  =       -165.41269
# of iterations =     11
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable         Coefficient      t-statistic    t-probability
const              45.056482         6.186276         0.000000
income             -1.030647        -3.369256         0.001533
house value        -0.265970        -3.004718         0.004293
rho                 0.431377         3.587351         0.000806
```

An LM error test for spatial correlation in the residuals of the SAR model confirms that there is no spatial dependence in the residuals of this model. The LM error test results are shown below and they would lead us to conclude that the SAR model adequately captures spatial dependence in this data set.

```
LM error tests for spatial correlation in SAR model residuals
LM value                  0.33002340
Marginal Probability      0.56564531
chi(1) .01 value          6.63500000
```

We will take a more detailed look at comparing alternative specifications for spatial autoregressive models in Chapter 4.

An important point regarding any non-linear optimization problem such as that involved in the SAC model is that the estimates may not reflect global solutions. A few different solutions of the optimization problem based on alternative starting values is usually undertaken to confirm that the estimates do indeed represent global solutions to the problem. The function **sac** allows the user to input alternative starting values, making this relatively easy to do.

A final example uses the large Pace and Barry data set to illustrate the **sac** function in operation on large problems. Example 3.12 turns on the printing flag so we can observe intermediate results from the optimization algorithm as it proceeds.

```
% ----- Example 3.12 Using sac() on a large data set
load elect.dat;             % load data on votes in 3,107 counties
y =  log(elect(:,7)./elect(:,8)); % convert to per capita variables
x1 = log(elect(:,9)./elect(:,8)); % education
x2 = log(elect(:,10)./elect(:,8));% homeownership
x3 = log(elect(:,11)./elect(:,8));% income
```

```
n = length(y); x = [ones(n,1) x1 x2 x3];
clear x1; clear x2; clear x3;
clear elect;                % conserve on RAM memory
load ford.dat; % 1st order contiguity matrix stored in sparse matrix form
ii = ford(:,1); jj = ford(:,2); ss = ford(:,3);
n = 3107;
clear ford; % clear ford matrix to save RAM memory
W = sparse(ii,jj,ss,n,n); W2 = slag(W,2);
clear ii; clear jj; clear ss; % conserve on RAM memory
vnames = strvcat('voters','const','educ','homeowners','income');
to = clock; info.pflag = 1;
res = sac(y,x,W,W2,info);
etime(clock,to)
prt(res,vnames);
```

The results are shown below, including intermediate results that were displayed by setting 'info.plag=1'. It took 535 seconds to solve this problem involving 5 iterations by the **maxlik** function. This function tends to be faster than the alternative optimization algorithms available in the *Econometrics Toolbox*.

```
==== Iteration ==== 2
log-likelihood   bconvergence   fconvergence
     7635.7620         0.2573         0.0017
Parameter    Estimates   Gradient
Parameter 1     0.4210  -232.2699
Parameter 2     0.4504  -162.1438

==== Iteration ==== 3
log-likelihood   bconvergence   fconvergence
     7635.6934         0.0304         0.0000
Parameter    Estimates   Gradient
Parameter 1     0.4163    -2.3017
Parameter 2     0.4591    13.0082

==== Iteration ==== 4
log-likelihood   bconvergence   fconvergence
     7635.6920         0.0052         0.0000
Parameter    Estimates   Gradient
Parameter 1     0.4151    -1.8353
Parameter 2     0.4601     0.4541

==== Iteration ==== 5
log-likelihood   bconvergence   fconvergence
     7635.6920         0.0001         0.0000
Parameter    Estimates   Gradient
Parameter 1     0.4150    -0.0772
Parameter 2     0.4601    -0.0637

General Spatial Model Estimates
Dependent Variable =       voters
R-squared      =    0.6653
Rbar-squared   =    0.6650
sigma^2        =    0.0131
log-likelihood =         3303.143
Nobs, Nvars    =   3107,     4
# iterations   =      5
***************************************************************
```

Variable	Coefficient	t-statistic	t-probability
const	0.683510	13.257563	0.000000
educ	0.247956	12.440953	0.000000
homeowners	0.555176	35.372538	0.000000
income	-0.117151	-5.858600	0.000000
rho	0.415024	16.527947	0.000000
lambda	0.460054	17.827407	0.000000

From the estimation results we see evidence that a general spatial model might be appropriate for this modeling problem. The parameters ρ and λ are both statistically significant. In addition, the log-likelihood for this model is slightly higher than the SAR, SEM and SDM models (see examples 3.4, 3.8 and 3.10).

3.6 Chapter Summary

We have seen that spatial autoregressive models can be estimated using univariate and bivariate optimization algorithms to solve for estimates by maximizing the likelihood function. The sparse matrix routines in MATLAB allow us to write functions that evaluate the log likelihood for large models rapidly and with a minimum of computer RAM memory. This approach was used to construct a library of estimation functions that were used to solve a problem involving a large sample of 3,107 counties in the continental U.S., as well as a problem with 35,702 observations. These large-scale problems were solved on an inexpensive desktop computer in a relatively short amount of time.

In addition to providing functions that estimate these models, the use of a general software design allowed us to provide both printed and graphical presentation of the estimation results.

Another place where we produced functions that can be used in spatial econometric analysis was in the area of testing for spatial dependence in the residuals from least-squares and SAR models. Functions were devised to implement Moran's $i-$statistic as well as likelihood ratio and Lagrange multiplier tests for spatial autocorrelation in the residuals from least-squares and SAR models. These tests are a bit more hampered by large-scale data sets, but we will illustrate some alternative approaches in the next chapter where Bayesian methods are introduced for estimation of spatial autoregressive models.

Chapter 4

Bayesian Spatial autoregressive models

This chapter discusses spatial autoregressive models from a Bayesian perspective. It is well-known that Bayesian regression methods implemented with diffuse prior information can replicate maximum likelihood estimation results. We demonstrate this type of application, but focus on some extensions that are available with the Bayesian approach. The maximum likelihood estimation methods set forth in the previous chapter are based on the presumption that the underlying disturbance process involved in generating the model is normally distributed.

Further, many of the formal tests for spatial dependence and heterogeneity including those introduced in the previous chapter rely on characteristics of quadratic forms for normal variates to derive the asymptotic distribution of the test statistics.

There is a history of Bayesian literature that deals with heteroscedastic and leptokurtic disturbances, treating these two phenomena in a similar fashion. Lange, Little and Taylor (1989) propose a non-Bayesian regression methodology based on an independent Student-t distribution for the disturbances. They show their approach provides robust results in a wide range of applied data settings. Geweke (1993) points out that the non-Bayesian methodology of Lange, Little and Taylor (1989) is identical to a Bayesian heteroscedastic linear regression model and argues that the same robust results arise. Geweke (1993) also makes a connection between the Bayesian heteroscedastic linear regression model and a Bayesian literature that treats symmetric leptokurtic disturbance distributions through the use of scale mixtures of normal distributions.

We adopt the approach of Geweke (1993) to extend the spatial autoregressive models introduced in Chapter 3.

The extended version of the model is:

$$y = \rho W_1 y + X\beta + u \quad (4.1)$$

$$
\begin{aligned}
u &= \lambda W_2 u + \varepsilon \\
\varepsilon &\sim N(0, \sigma^2 V) \\
V &= \mathrm{diag}(v_1, v_2, \ldots, v_n)
\end{aligned}
$$

Where the change made to the basic model is in the assumption regarding the disturbances ε. We assume that they exhibit non-constant variance, taking on difference values for every observation. The magnitudes $v_i, i = 1, \ldots\ldots, n$ represent parameters to be estimated. This assumption of inherent spatial heteroscedasticity seems more appropriate than the traditional Gauss-Markov assumption that the variance of the disturbance is constant over space.

The first section of this chapter introduces a Bayesian heteroscedastic regression model and the topic of Gibbs sampling estimation without complications introduced by the spatial autoregressive model. The next section applies these ideas to the simple FAR model and implements a Gibbs sampling estimation procedure for this model. Following sections deal with the other spatial autoregressive models that we introduced in the previous chapter.

A final section provides a detailed applied exercise where we apply the models from Chapter 3 as well as the Bayesian models introduced here. Issues relating to the choice of a model specification are illustrated with this exercise based on a spatial data set for Boston area census tracts.

4.1 The Bayesian regression model

We consider the case of a heteroscedastic linear regression model with an informative prior that can be written as in (4.2).

$$
\begin{aligned}
y &= X\beta + \varepsilon \qquad (4.2) \\
\varepsilon &\sim N(0, \sigma^2 V) \\
V &= \mathrm{diag}(v_1, v_2, \ldots, v_n) \\
\beta &\sim N(c, T) \\
\sigma &\sim (1/\sigma) \\
r/v_i &\sim \mathrm{ID}\ \chi^2(r)/r \\
r &\sim \Gamma(m, k)
\end{aligned}
$$

Where y is an nx1 vector of dependent variables and X represents the nxk matrix of explanatory variables. We assume that ε is an nx1 vector of normally distributed random variates with non-constant variance. We place a normal prior on the parameters β and a diffuse prior on σ. The relative variance terms $(v_1, v_2, \ldots, v_n)$, are assumed fixed but unknown parameters that need to be estimated. The thought of estimating n parameters, $v_1, v_2, \ldots, v_n$, in addition to the k+1 parameters, β and σ using n data observations seems problematical from a degrees of freedom perspective. Bayesian methods don't encounter the same degrees of freedom constraints, because we can rely on an informative prior for the v_i parameters. This prior distribution for the v_i terms will take the form of an independent $\chi^2(r)/r$ distribution. Recall that the χ^2 distribution is a single parameter distribution, where we have represented this parameter as r. This allows us to estimate the additional n parameters v_i in the model by adding the single parameter r to our estimation procedure.

This type of prior has been used by Lindley (1971) for cell variances in an analysis of variance problem, and Geweke (1993) in modeling heteroscedasticity and outliers in the context of linear regression. The specifics regarding the prior assigned to the v_i terms can be motivated by considering that the mean of prior equals unity and the variance of the prior is $2/r$. This implies that as r becomes very large, the terms v_i will all approach unity, resulting in $V = I_n$, the traditional Gauss-Markov assumption. We will see that the role of $V \neq I_n$ is to robustify against outliers and observations containing large variances by downweighting these observations. Large r values are associated with a prior belief that outliers and non-constant variances do not exist, since this type of prior would produce $V = I_n$.

Now consider the posterior distribution from which we would derive our estimates. Following the usual Bayesian methodology, we would combine the likelihood function for our simple model with the prior distributions for β, σ and V to arrive at the posterior. For the case of an improper prior for $\beta \propto$ constant, and the parameter $\sigma \propto \sigma^{-1}$, the product of these prior densities and the likelihood function produces:

$$\begin{aligned} p(\beta, \sigma, V|y, X) &= (r/2)^{nr/2}[\Gamma(r/2)]^{-n}\sigma^{-(n+1)}\prod_{i=1}^{n} v_i^{-(r+3)/2} \\ &\cdot \quad \exp\{-\sum_{i=1}^{n}[\sigma^{-2}(y_i - x_i'\beta)^2 + r]/2v_i\} \end{aligned} \tag{4.3}$$

The posterior density in (4.3) is is not amenable to analysis, a problem that has often plagued Bayesian methods in the past. We can however derive the posterior distribution for the parameters in our model using a recent methodology known as Markov Chain Monte Carlo, sometimes referred to as Gibbs sampling.

Markov Chain Monte Carlo is based on the idea that rather than compute the posterior density of our parameters based on the expression in (4.3), we would be just as happy to have a large random sample from the posterior of our parameters, which we designate $p(\theta|D)$, using θ to denote the parameters and D the sample data. If the sample from $p(\theta|D)$ were large enough, we could approximate the form of the probability density using kernel density estimators or histograms, eliminating the need to know the precise analytical form of the density.

In addition, we could compute accurate measures of central tendency and dispersion for the density, using the mean and standard deviation of the large sample. This insight leads to the question of how to efficiently simulate a large number of random samples from $p(\theta|D)$.

Metropolis, et al. (1953) showed that one could construct a Markov chain stochastic process for $(\theta_t, t \geq 0)$ that unfolds over time such that: 1) it has the same state space (set of possible values) as θ, 2) it is easy to simulate, and 3) the equilibrium or stationary distribution which we use to draw samples is $p(\theta|D)$ after the Markov chain has been run for a long enough time. Given this result, we can construct and run a Markov chain for a very large number of iterations to produce a sample of $(\theta_t, t = 1, \ldots)$ from the posterior distribution and use simple descriptive statistics to examine any features of the posterior in which we are interested.

This approach, known as Markov Chain Monte Carlo, (MCMC) to determining posterior densities has greatly reduced the computational problems that previously plagued application of the Bayesian methodology. Gelfand and Smith (1990), as well as a host of others, have popularized this methodology by demonstrating its use in a wide variety of statistical applications where intractable posterior distributions previously hindered Bayesian analysis. A simple introduction to the method can be found in Casella and George (1990) and an expository article dealing specifically with the normal linear model is Gelfand, Hills, Racine-Poon and Smith (1990). Two recent books that deal in detail with all facets of these methods are: Gelman, Carlin, Stern and Rubin (1995) and Gilks, Richardson and Spiegelhalter (1996).

The most widely used approach to MCMC is due to Hastings (1970) which generalizes a method of Metropolis et al. (1953). A second approach (that we focus on) is known as Gibbs sampling due to Geman and Geman (1984). Hastings (1970) suggests that given an initial value θ_0 we can construct a chain by recognizing that any Markov chain that has found its way to a state θ_t can be completely characterized by the probability distribution for time $t+1$. His algorithm relies on a proposal or candidate distribution, $f(\theta|\theta_t)$ for time $t+1$, given that we have θ_t. A candidate point $\theta^\star$ is sampled from the proposal distribution and:

1. This point is accepted as $\theta_{t+1} = \theta^\star$ with probability:

$$\alpha_H(\theta_t, \theta^\star) = \min\left[1, \frac{p(\theta^\star|D)f(\theta_t|\theta^\star)}{p(\theta_t|D)f(\theta^\star|\theta_t)}\right] \tag{4.4}$$

2. otherwise, $\theta_{t+1} = \theta_t$, that is we stay with the current value of θ.

In other words, we can view the Hastings algorithm as indicating that we should toss a Bernoulli coin with probability α_H of heads and make a move to $\theta_{t+1} = \theta^\star$ if we see a heads, otherwise set $\theta_{t+1} = \theta_t$. Hastings demonstrates that this approach to sampling represents a Markov chain with the correct equilibrium distribution capable of producing samples from the posterior $p(\theta|D)$ we are interested in.

The MCMC method we will rely on is known as Gibbs sampling and dates to the work of Geman and Geman (1984) in image analysis. It is related to the EM algorithm (Dempster, Laird and Rubin, 1977) which has been used for maximum likelihood estimation in problems involving missing information. Assume a parameter vector $\theta = (\theta_1, \theta_2)$, a prior $p(\theta)$, and likelihood $l(\theta|y)$, that produces a posterior distribution $p(\theta|D) = cp(\theta)l(\theta|y)$, with c a normalizing constant. It is often the case that the posterior distribution over all parameters is difficult to work with. On the other had, if we partition our parameters into two sets θ_1, θ_2 and had initial estimates for θ_1 (treated like missing information in the EM algorithm), we could estimate θ_2 conditional on θ_1 using $p(\theta_2|D, \hat{\theta}_1)$. (Presumably, this estimate is much easier to derive. We will provide details illustrating this case in section 4.2.1.) Denote the estimate, $\hat{\theta}_2$ derived by using the posterior mean or mode of $p(\theta_2|D, \hat{\theta}_1)$, and consider than we are now able to construct a new estimate of θ_1 based on the conditional distribution $p(\theta_1|D, \hat{\theta}_2)$, which can be used to construct another value for θ_2, and so on.

In general, for the case of k parameters, the algorithm can be summarized as:

Initialize θ_0

Repeat {

Sample $\theta_1^{t+1} \sim p[\theta_1|D, (\theta_2^t, \ldots, \theta_k^t)]$
Sample $\theta_2^{t+1} \sim p[\theta_2|D, (\theta_1^{t+1}, \theta_3^t, \ldots, \theta_k^t)]$

$\vdots$

Sample $\theta_k^{t+1} \sim p[\theta_k|D, (\theta_1^{t+1}, \theta_2^{t+1}, \ldots, \theta_{k-1}^{t+1})]$
$t = t + 1$

}

Geman and Geman (1984) demonstrated that the stochastic process θ^t from this approach to sampling the complete sequence of conditional distributions represents a Markov chain with the correct equilibrium distribution. Gibbs sampling is in fact closely related to Hastings and Metropolis MCMC methods. For the case of the spatial autoregressive models, we will need to rely on Metropolis sampling within a sequence of Gibbs sampling, a procedure that is often labeled "Metropolis within Gibbs sampling".

Section 4.1.1 derives a Gibbs sampling approach to estimate the parameters of our Bayesian heteroscedastic linear model. In Section 4.2 we extend this to the case of a Bayesian FAR model which requires that we introduce "Metropolis within Gibbs sampling". Section 4.2 puts all of these ideas into practice by implementing a MATLAB function **far_g** that produces Gibbs sampling estimates for the Bayesian heteroscedastic FAR model.

4.1.1 The heteroscedastic Bayesian linear model

To produce estimates using Gibbs sampling, we need to consider the conditional distributions for each of the parameters, β, σ and V in our model. These distributions are those that would arise from assuming each of the other parameters were known. For example, the conditional distribution for β assuming that we knew σ and V would looks as follows:

$$\begin{aligned} \beta|(\sigma, V) &\sim N[H(X'V^{-1}y + \sigma^2 T^{-1}c) \ , \ \sigma^2 H]. \\ H &= (X'V^{-1}X + T^{-1})^{-1} \end{aligned} \tag{4.5}$$

Note that this is quite analogous to a generalized least-squares (GLS) version of the Theil and Goldberger (1961) estimation formulas, known as the "mixed estimator". Consider also that this would be fast and easy to compute. It is often the case that complicated joint posterior distributions for Bayesian models are associated with relatively simple conditional distributions as this example illustrates.

Next consider the conditional distribution for the parameter σ assuming that we knew the parameters β and V in the problem. This distribution would be:

$$[\sum_{i=1}^{n}(e_i^2/v_i)/\sigma^2]|(\beta, V) \sim \chi^2(n) \tag{4.6}$$

Where we let $e_i = y_i - x_i'\beta$. This result parallels the simple regression case where we know that the residuals are χ^2 distributed. A difference from the standard case is that we adjust the e_i using the relative variance terms v_i as deflators.

Finally, Geweke (1993) shows that the conditional distribution for the parameters V represent a χ^2 distribution with $r+1$ degrees of freedom as shown in (4.7)

$$[(\sigma^{-2}e_i^2 + r)/v_i]|(\beta, \sigma) \sim \chi^2(r+1) \tag{4.7}$$

Having specified the conditional distributions for all of the parameters in the model, we proceed to carry out random draws from these distributions until we collect a large sample of parameter draws. Given our earlier discussion, this sequence of draws from the series of conditional distributions for all parameters in the model represents a set of draws that converge in the limit to the true (joint) posterior distribution of the parameters. That is, despite the use of conditional distributions in our sampling scheme, a large sample of the draws can be used to produce valid posterior inferences about the mean and moments of the multivariate posterior parameter distribution for our model.

The Gibbs sampler for our heteroscedastic Bayesian regression model based on the three conditional posterior densities in (4.5) through (4.7) involves using the following steps:

1. Begin with arbitrary values for the parameters β^0, σ^0 and v_i^0 which we designate with the superscript 0.

2. Compute the mean and variance of β using (4.5) conditional on the initial values σ^0 and v_i^0.

3. Use the computed mean and variance of β to draw a multivariate normal random vector, which we label β^1.

4. Calculate expression (4.6) using β^1 determined in step 3 and use this value along with a random $\chi^2(n)$ draw to determine σ^1.

5. Using β^1 and σ^1, calculate expression (4.7) and use the value along with an $n-$vector of random $\chi^2(r+1)$ draws to determine $v_i, i = 1, \ldots, n$.

These steps constitute a single pass of the Gibbs sampler. We wish to make a large number of passes to build up a sample $(\beta^j, \sigma^j, v_i^j)$ of j values from which we can approximate the posterior distributions for our parameters.

To illustrate this approach in practice, Example 4.1 shows the MATLAB code for estimation using the Gibbs sampler set forth above. In this example, we generate a regression model data set that contains a heteroscedastic set of disturbances based on a time trend variable. Only the last 50 observations in the generated data sample contain non-constant variances. This allows us to see if the estimated v_i parameters detect this pattern of non-constant variance over the last half of the sample.

The generated data set used values of unity for the intercept term β_0 and the two slope parameters, β_1 and β_2. The prior means for the β parameters were set to zero with prior variances of 1e+12, reflecting a diffuse prior because of the very large prior uncertainty.

```
% ----- Example 4.1 Heteroscedastic Gibbs sampler
n=100; k=3; % set number of observations and variables
x = randn(n,k); b = ones(k,1);  % generate data set
tt = ones(n,1); tt(51:100,1) = [1:50]';
y = x*b + randn(n,1).*sqrt(tt); % heteroscedastic disturbances
ndraw = 1100; nomit = 100;      % set the number of draws
bsave = zeros(ndraw,k);         % allocate storage for results
ssave = zeros(ndraw,1);
vsave = zeros(ndraw,n);
c = [0.0 0.0 0.0]';             % diffuse prior b means
T = eye(k)*1e+12;               % diffuse prior b variance
Q = chol(inv(T)); q = Q*c;
b0 = x\y;                       % use ols starting values
sige = (y-x*b0)'*(y-x*b0)/(n-k);
V = ones(n,1); in = ones(n,1);  % initial value for V
rval = 4;                       % initial value for rval
qpq = Q'*Q; qpv = Q'*q;         % calculate Q'Q, Q'q only once
tic;                            % start timing
for i=1:ndraw;                  % Start the sampling
  ys = y.*sqrt(V); xs = matmul(x,sqrt(V));
  xpxi = inv(xs'*xs + sige*qpq);
  b = xpxi*(xs'*ys + sige*qpv); % update b
  b = norm_rnd(sige*xpxi) + b;  % draw MV normal mean(b), var(b)
  bsave(i,:) = b';              % save b draws
  e = ys - xs*b; ssr = e'*e;    % update sige
  chi = chis_rnd(1,n);          % do chisquared(n) draw
  sige = ssr/chi; ssave(i,1) = sige; % save sige draws
  chiv = chis_rnd(n,rval+1);    % update vi
  e = y - x*b;                  % redefine e
```

```
  vi = ((e.*e./sige) + in*rval)./chiv;
  V = in./vi; vsave(i,:) = vi'; % save the draw
end;                            % End the sampling
toc;                            % stop timing
bhat = mean(bsave(nomit+1:ndraw,:));  % calculate means and std deviations
bstd = std(bsave(nomit+1:ndraw,:));   tstat = bhat./bstd;
smean = mean(ssave(nomit+1:ndraw,1)); vmean = mean(vsave(nomit+1:ndraw,:));
tout = tdis_prb(tstat',n); % compute t-stat significance levels
% set up for printing results
in.cnames = strvcat('Coefficient','t-statistic','t-probability');
in.rnames = strvcat('Variable','variable 1','variable 2','variable 3');
in.fmt = '%16.6f'; tmp = [bhat' tstat' tout];
fprintf(1,'Gibbs estimates \n'); % print results
mprint(tmp,in);
fprintf(1,'Sigma estimate = %16.8f \n',smean);
result = theil(y,x,c,R,T);       % compare to Theil-Golberger estimates
prt(result);  plot(vmean);       % plot vi-estimates
title('mean of vi-estimates');
```

We rely on MATLAB functions **norm_rnd** and **chis_rnd** to provide the multivariate normal and chi-squared random draws. These functions are part of the *Econometrics Toolbox* and are discussed in Chapter 8 of the manual. Note also, we omit the first 100 draws at start-up to allow the Gibbs sampler to achieve a steady state before we begin sampling for the parameter distributions.

One aspect of the code that is tricky involves the way we deflate the terms e_i^2/v_i indicated in (4.6). We use 'e=ys-xs*b', where 'ys,xs' represent the vector y and matrix X in our model divided by the v_i terms. Therefore 'ssr' represents e_i^2/v_i. We then re-define e using 'y,x' when computing the v_i terms since (4.7) implies that $v_i = \iota/((e_i^2/\sigma^2) + r)$, where the e_i^2 terms are not deflated.

The results are shown below, where we find that it took only 11.9 seconds to carry out the 1100 draws. This provides a sample of 1000 draws (after discarding the initial 100 draws for start-up) on which to base our posterior inferences regarding the parameters β and σ. For comparison purposes, we produced estimates using the **theil** function from the *Econometrics Toolbox* that implements Theil and Goldberger (1961) mixed estimation. These estimates are similar, but the $t-$statistics are lower because they suffer from the heteroscedasticity. Our Gibbs sampled estimates take this into account increasing the precision of the estimates as indicated by the larger $t-$statistics.

```
elapsed_time =  11.9025 seconds
Gibbs estimates
Variable        Coefficient      t-statistic    t-probability
variable 1         0.792589         3.042734         0.002995
variable 2         1.196825         3.802015         0.000247
variable 3         1.007050         3.507125         0.000680
Sigma estimate =       5.00810138

Theil-Goldberger Regression Estimates
R-squared      =    0.2360
Rbar-squared   =    0.2202
sigma^2        =   10.6332
Durbin-Watson  =    2.0750
Nobs, Nvars    =    100,     3
```

```
***************************************************************
Variable        Prior Mean    Std Deviation
variable 1        0.000000   1000000.000000
variable 2        0.000000   1000000.000000
variable 3        0.000000   1000000.000000

***************************************************************
      Posterior Estimates
Variable       Coefficient      t-statistic    t-probability
variable 1        0.845653         0.768487         0.444065
variable 2        1.566902         1.143459         0.255662
variable 3        0.848771         0.761171         0.448401
```

Figure 4.1 shows the mean of the 1,000 draws for the parameters v_i plotted for the 100 observation sample. Recall that the last 50 observations contained a generated time-trend pattern of non-constant variance. This pattern was detected quite accurately by the estimated v_i terms.

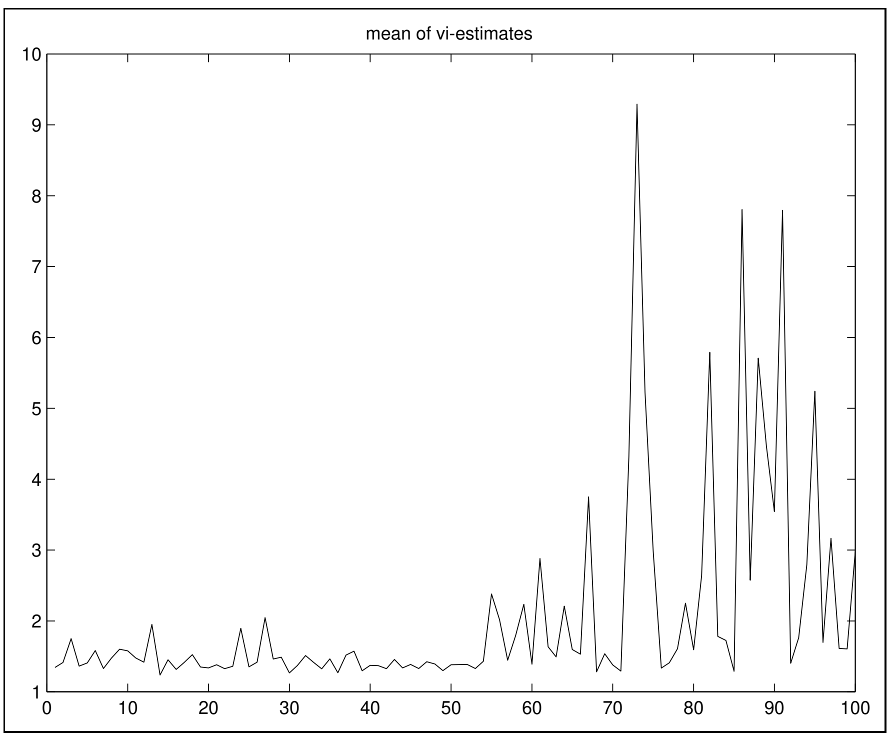

Figure 4.1: V_i estimates from the Gibbs sampler

One point that should be noted about Gibbs sampling estimation is that convergence of the sampler needs to be diagnosed by the user. The *Econometrics Toolbox* provides a set of convergence diagnostic functions that we demonstrate in Section 4.3. Fortunately, for simple regression models (and spatial autoregressive models) convergence of the sampler is usually a certainty, and convergence occurs quite rapidly. A simple approach to testing for convergence is to run the sampler once to carry out a small number of draws, say 300 to 500, and a second time to carry out a larger number of draws, say 1000 to 2000. If the means and variances for the posterior estimates are similar from both runs, convergence seems assured.

4.2 The Bayesian FAR model

In this section we turn attention to implementing a Gibbs sampling approach for the FAR model that can accommodate heteroscedastic disturbances and outliers. Note that the presence of a few spatial outliers due to enclave effects or other aberrations in the spatial sample will produce a violation of normality in small samples. The distribution of disturbances will take on a fat-tailed or leptokurtic shape. This is precisely the type of problem that the heteroscedastic modeling approach of Geweke (1993) based on Gibbs sampling estimation was designed to address.

The Bayesian extension of the FAR model takes the form:

$$\begin{aligned}
y &= \rho W y + \varepsilon \\
\varepsilon &\sim N(0, \sigma^2 V) \\
V &= \mathrm{diag}(v_1, v_2, \ldots, v_n) \\
\rho &\sim N(c, T) \\
r/v_i &\sim \mathrm{ID}\ \chi^2(r)/r \\
r &\sim \Gamma(m, k) \\
\sigma &\sim \Gamma(\nu_0, d_0)
\end{aligned} \tag{4.8}$$

Where as in Chapter 3, the spatial contiguity matrix W has been standardized to have row sums of unity and the variable vector y is expressed in deviations from the means to eliminate the constant term in the model. We allow for an informative prior on the spatial autoregressive parameter ρ, the heteroscedastic control parameter r and the disturbance variance σ. This is the most general Bayesian model, but most practitioners might rely on diffuse priors for σ and ρ.

A diffuse prior for ρ would be implemented by setting the prior mean (c) to zero and using a large prior variance for T, say 1e+12. To implement a diffuse prior for σ we would set $\nu_0 = 0, d_0 = 0$. The prior for r is based on a $\Gamma(m, k)$ distribution which has a mean equal to m/k and a variance equal to m/k^2. Recall our discussion of the role of the prior hyperparameter r in allowing the v_i estimates to deviate from their prior means of unity. Small values for r around 2 to 7 allow for non-constant variance and are associated with a prior belief that outliers or non-constant variance exist. Large values such as $r = 30$ or $r = 50$ would produce v_i estimates that are all close to unity, forcing the model to take on a homoscedastic character and produce estimates equivalent to those from the maximum likelihood FAR model discussed in Chapter 3. This would make little sense — if we wished to produce maximum likelihood estimates, it would be much quicker to use the **far** function from Chapter 3.

In example 4.1, we set $r = 4$ to allow ample opportunity for the v_i parameters to deviate from unity. Figure 4.1 shows v_i estimates close to unity for the first 50 observations despite our prior setting of $r = 4$. We will provide examples that suggests an optimal strategy for setting r is to use small values in the range from 2 to 7. If the sample data exhibits homoscedastic disturbances that are free from outliers, the v_i estimates will reflect this fact. On the other hand, if there is evidence of heterogeneity in the errors, these settings for the hyperparameter r will allow the v_i estimates to deviate substantially from unity. Estimates for the v_i parameters that deviate from unity are needed to produce an adjustment in the estimated parameters ρ and σ that take non-constant variance into account or robustify our estimates in the presence of outliers.

Econometric estimation problems amenable to Gibbs sampling methods can take one of two forms. The simplest case is where all of the conditional distributions are from well-known distributions allowing us to sample random deviates using standard computational algorithms. This was the case with our heteroscedastic Bayesian regression model in example 4.1.

A second more complicated case is where one or more of the conditional distributions can be expressed mathematically, but take an unknown form. It is still possible to implement a Gibbs sampler for these models using a host of alternative methods that are available to produce draws from distributions taking non-standard forms.

One of the more commonly used ways to deal with this situation is the Metropolis algorithm presented in Section 4.1. All spatial autoregressive models require that we sample from the conditional distribution for the parameters ρ and λ which take a non-standard form. To accomplish this, we rely on what is known as a "Metropolis within Gibbs" sampler.

To explore the nature of the non-standard conditional distribution for ρ that arises, consider the conditional distributions for the FAR model parameters where we rely on diffuse priors, $\pi(\rho)$ and $\pi(\sigma)$ for the parameters (ρ, σ) shown in (4.9). We also eliminate the individual variance terms v_i from the model to keep things simple.

$$\begin{aligned} \pi(\rho) &\propto \text{constant} \\ \pi(\sigma) &\propto (1/\sigma), \quad 0 < \sigma < +\infty \end{aligned} \tag{4.9}$$

These priors can be combined with the likelihood for this model producing a joint posterior distribution for the parameters, $p(\rho, \sigma|y)$.

$$p(\rho, \sigma|y) \propto |I_n - \rho W|\sigma^{-(n+1)}\exp\{-\frac{1}{2\sigma^2}(y - \rho W y)'(y - \rho W y)\} \tag{4.10}$$

If we treat ρ as known, the kernel for the conditional posterior (that part of the distribution that ignores inessential constants) for σ given ρ takes the form:

$$p(\sigma|\rho, y) \propto \sigma^{-(n+1)}\exp\{-\frac{1}{2\sigma^2}\varepsilon'\varepsilon\} \quad (4.11)$$

where $\varepsilon = y - \rho W y$. It is important to note that by conditioning on ρ (treating it as known) we can subsume the determinant, $|I_n - \rho W|$, as part of the constant of proportionality, leaving us with one of the standard distributional forms. From (4.11) we conclude that $\sigma^2 \sim \chi^2(n)$.

Unfortunately, the conditional distribution of ρ given σ takes the following non-standard form:

$$p(\rho|\sigma, y) \propto \sigma^{-n/2}|I_n - \rho W|\{(y - \rho W y)'(y - \rho W y)\}^{-n/2} \quad (4.12)$$

To sample from (4.12) we can rely on Metropolis sampling, within the Gibbs sampling sequence, hence it is often labeled "Metropolis within Gibbs".

Implementation of the Metropolis sampling algorithm from Section 4.1 is described here for the case of a symmetric normal candidate generating density, which we denoted $f(\theta|\theta_t)$ in (4.4). This candidate generating density should work well for the conditional distribution of ρ because, as Figure 4.2 shows, the conditional distribution of ρ is similar to a normal distribution with the same mean value. The figure also shows a $t-$distribution with 3 degrees of freedom, which would also work well in this application.

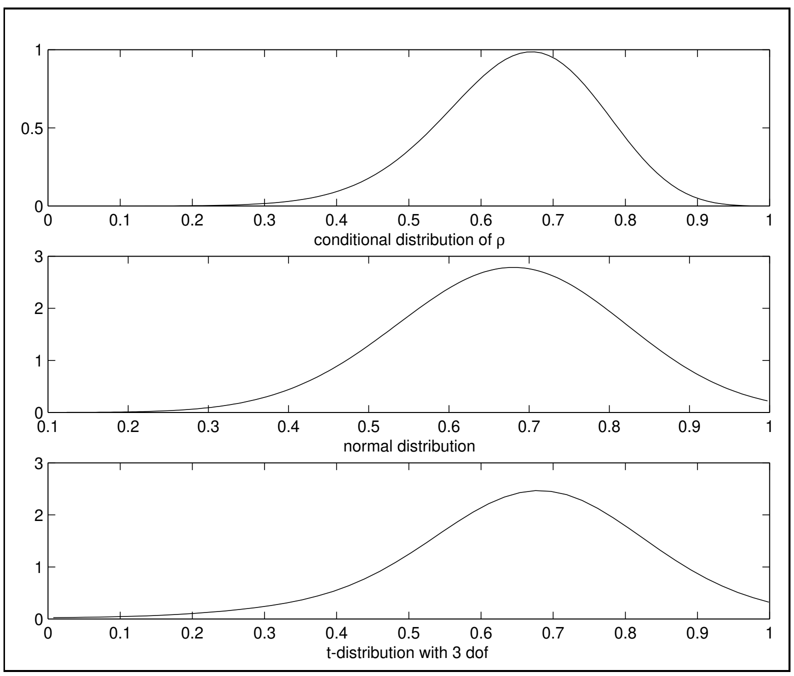

Figure 4.2: Conditional distribution of ρ

To sample from the conditional distribution $p(\rho|\sigma, y)$, let ρ_0 denote an initial value of ρ. We generate a value ρ_c from the candidate normal density using: $\rho_c = \rho_0 + cZ$, where Z is a draw from a standard normal distribution and c is a known constant. (If we wished to rely on a $t-$distribution as the candidate density, we could simply replace Z with a random draw from the $t-$distribution.)

An acceptance probability is computed using: $p = min\{1, f(\rho_c)/f(\rho_0\}$. We then draw a uniform random deviate we label U, and if $U < p$, the next draw from f is given by $\rho_1 = \rho_c$. If on the other hand, $U \geq p$, the draw is taken to be the current value, $\rho_1 = \rho_0$.

A MATLAB program to implement this approach for the case of the homoscedastic first-order spatial autoregressive (FAR) model is shown in example 4.2. We use this simple case as an introduction before turning to the slightly more complicated case of the heteroscedastic FAR model. Note also that we do not rely on the sparse matrix algorithms which we introduce after this simple example.

An implementation issue is that we need to impose the restriction:

$$1/\lambda_{min} < \rho < 1/\lambda_{max}$$

where λ_{min} and λ_{max} are the minimum and maximum eigenvalues of the standardized spatial weight matrix W. We impose this restriction using an approach that has been labeled 'rejection sampling'. Restrictions such as this, as well as non-linear restrictions, can be imposed on the parameters during Gibbs sampling by simply rejecting values that do not meet the restrictions (see Gelfand, Hills, Racine-Poon and Smith, 1990).

```
% ----- Example 4.2 Metropolis within Gibbs sampling FAR model
n=49; ndraw = 1100; nomit = 100; nadj = ndraw-nomit;
% generate data based on a given W-matrix
load wmat.dat; W = wmat; IN = eye(n); in = ones(n,1); weig = eig(W);
lmin = 1/min(weig); lmax = 1/max(weig); % bounds on rho
rho = 0.5;     % true value of rho
y = inv(IN-rho*W)*randn(n,1); ydev = y - mean(y); Wy = W*ydev;
               % set starting values
rho = -0.5;    % starting value for the sampler
sige = 100.0; % starting value for the sampler
c = 0.5;       % for the Metropolis step (adjusted during sampling)
rsave = zeros(nadj,1); % storage for results
ssave = zeros(nadj,1);  rtmp = zeros(nomit,1);
iter = 1; cnt = 0;
while (iter <= ndraw);        % start sampling;
e = ydev - rho*Wy; ssr = (e'*e); % update sige;
chi = chis_rnd(1,n); sige = (ssr/chi);
% metropolis step to get rho update
rhox = c_rho(rho,sige,ydev,W);      % c_rho evaluates conditional
rho2 = rho + c*randn(1); accept = 0;
 while accept == 0;           % rejection bounds on rho
  if ((rho2 > lmin) & (rho2 < lmax)); accept = 1;  end;
 rho2 = rho + c*randn(1); cnt = cnt+1;
 end;                         % end of rejection for rho
rhoy = c_rho(rho2,sige,ydev,W);     % c_rho evaluates conditional
ru = unif_rnd(1,0,1); ratio = rhoy/rhox; p = min(1,ratio);
```

```
if (ru < p)
 rho = rho2; rtmp(iter,1) = rho; iter = iter+1;
 end;
  if (iter >= nomit);
    if iter == nomit          % update c based on initial draws
     c = 2*std(rtmp(1:nomit,1));
    end;
   ssave(iter-nomit+1,1) = sige; rsave(iter-nomit+1,1) = rho;
  end; % end of if iter > nomit
end; % end of sampling loop
% print-out results
fprintf(1,'hit rate =  %6.4f \n',ndraw/cnt);
fprintf(1,'mean, std and t-statistic for rho %6.3f %6.3f %6.3f \n', ...
mean(rsave),std(rsave),mean(rsave)/std(rsave));
fprintf(1,'mean and std of sig %6.3f %6.3f \n',mean(ssave),std(ssave));
% maximum likelihood estimation for comparison
res = far(ydev,W);
prt(res);
```

Rejection sampling is implemented in the example with the following code fragment that examines the candidate draws in 'rho2' to see if they are in the feasible range. If 'rho2' is not in the feasible range, another candidate value 'rho2' is drawn and we increment a counter variable 'cnt' to keep track of how many candidate values are found outside the feasible range. The 'while loop' continues to draw new candidate values and examine whether they are in the feasible range until we find a candidate value within the limits. Finding this value terminates the 'while loop'. This approach ensures than any values of ρ that are ultimately accepted as draws will meet the constraints.

```
% metropolis step to get rho update
rho2 = rho + c*randn(1); accept = 0;
 while accept == 0;              % rejection bounds on rho
  if ((rho2 > lmin) & (rho2 < lmax)); accept = 1;  end;
 rho2 = rho + c*randn(1); cnt = cnt+1;
 end;                            % end of rejection for rho
```

Another point to note about the example is that we adjust the variable 'c' used as the standard deviation for the normally distributed candidate values. The adjustment relies on the initial 100 draws of 'rho' to compute a new value for 'c' based on two standard deviations of the initial draws. The following code fragment carries this out, where the initial 'rho' draws have been stored in a vector 'rtmp'.

```
    if iter == nomit          % update c based on initial draws
     c = 2*std(rtmp(1:nomit,1));
    end;
```

Consider also, that we delay collecting our sample of draws for the parameters ρ and σ until we have executed 'nomit' burn-in draws, which is 100 in this case. This allows the sampler to settle into a steady state, which might be required if poor values of ρ and σ were used to initialize the sampler. In theory, any arbitrary values can be used, but a choice of good values will speed up convergence of the sampler. A plot of the first 100 values drawn from this example is shown in Figure 4.3. We used $\rho = -0.5$ and $\sigma^2 = 100$ which were deliberately chosen as very poor starting values since the true value of $\rho = 0.5$ and $\sigma^2 = 1$. The plots of the first 100 values indicate that even if we start with very poor values, far from the true values used to generate the data, only a few iterations are required to reach a steady state. This is usually true for regression-based Gibbs samplers. Section 4.2 deals with the issue of testing for convergence of the sampler in detail.

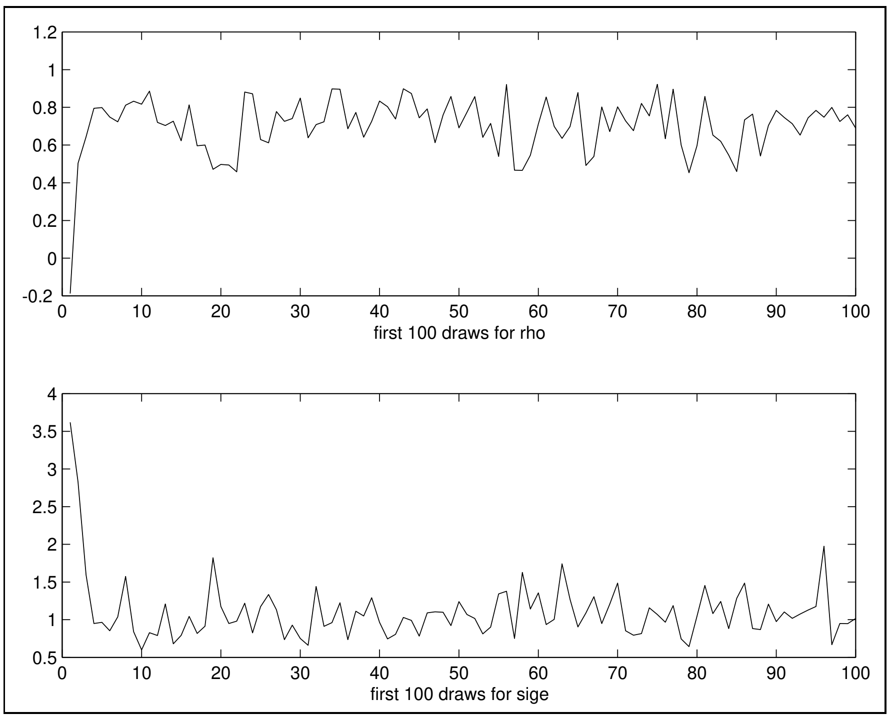

Figure 4.3: First 100 Gibbs draws for ρ and σ

The function **c_rho** evaluates the conditional distribution for ρ given σ^2 at any value of ρ. Of course, we could use our sparse matrix algorithms in this function to handle large data sample problems, a subject we will turn to later.

```
function cout = c_rho(rho,sige,y,W)
% evaluates conditional distribution of rho
% given sige for the spatial autoregressive model
n = length(y);
IN = eye(n); B = IN - rho*W;
detm = log(det(B));
epe = (n/2)*log(y'*B'*B*y);
cout = -epe -(n/2)*log(sige) + detm; cout = exp(cout);
```

We present results from executing the code shown in example 4.2, where both

Gibbs estimates based on the mean of the 1,000 draws for ρ and σ as well as the standard deviations are shown. For contrast, we present maximum likelihood estimates, which for the case of the homoscedastic Gibbs sampler implemented here with a diffuse prior on ρ and σ should produce similar estimates.

```
hit rate =  0.3325
mean, std and t-statistic for rho  0.434  0.180  2.413
mean and std of sig  1.006  0.210

First-order spatial autoregressive model Estimates
R-squared       =    0.1729
sigma^2         =    0.9408
Nobs, Nvars     =     49,     1
log-likelihood  =       -121.7366
# of iterations =     11
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
rho              0.457534         2.146273         0.036935
```

The time needed to generate 1,100 draws was around 10 seconds, which represents 100 draws per second. We will see that similar speed can be achieved even for large data samples.

From the results we see that the mean and standard deviations from the Gibbs sampler produce estimates close to the maximum likelihood estimates, and to the true values used to generate the model data.

The reader should keep in mind that we do not advocate using the Gibbs sampler in place of maximum likelihood estimation. That is, we don't really wish to implement a homoscedastic version of the FAR model using Gibbs sampling. We turn attention to the more general heteroscedastic case that allows for either diffuse or informative priors in the next section.

4.2.1 Constructing a function far_g()

We focus attention on implementation details concerned with constructing a MATLAB function **far_g** that will produce estimates for the Bayesian FAR model. This function will rely on a sparse matrix algorithm approach to handle problems involving large data samples. It will also allow for diffuse or informative priors and handle the case of heterogeneity in the disturbance variance.

The first thing we need to consider is that to produce a large number of draws, say 1,000, we would need to evaluate the conditional distribution of ρ 2,000 times. (Note that we called this function twice in example 4.2). Each evaluation would require that we compute the determinant of the matrix $(I_n - \rho W)$, which we have already seen is a non-trivial task for large data samples. To avoid this, we rely on the Pace and Barry (1997) approach discussed in the previous chapter. Recall that they suggested evaluating this determinant over a grid of values in the feasible range of ρ once at the outset. Given that we have carried out this evaluation and stored the values for the determinant and associated values of ρ, we can simply "look-up" the appropriate determinant in our function that evaluates the conditional distribution. That is, the call to the conditional distribution function will provide a value of ρ for which we need to evaluate the conditional distribution. If we already know the determinant for a grid of all feasible ρ values, we can simply look up the determinant value closest to the ρ value and use it during evaluation of the conditional distribution. This saves us the time involved in computing the determinant twice for each draw of ρ.

The code that we execute at the outset in our function **far_g** to compute determinant values over a grid of ρ values is shown below. It should not be new, as it was already used in Chapter 3 when we illustrated the Pace and Barry approach to spatial autoregressive estimation of large problems.

```
opt.tol = 1e-3; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
lmin = 1/lambda(2);        lmax = 1/lambda(1);
results.rmax = lmax; results.rmin = lmin;
% compute a detval vector based on Pace and Berry's approach
rvec = lmin-0.005:.005:lmax+0.005;
spparms('tight');
z = speye(n) - 0.1*sparse(W);
p = colmmd(z);
niter = length(rvec);
detval = zeros(niter,2);
for i=1:niter;
  rho = rvec(i); z = speye(n) - rho*sparse(W);
  [l,u] = lu(z(:,p));
  detval(i,1) = sum(log(abs(diag(u)))); detval(i,2) = rho;
end;
```

Note that we save the values of the determinant alongside the associated values of ρ in a 2-column matrix named **detval**. We will simply pass this matrix to the conditional distribution function **c_far** which is shown below:

```
function cout = c_far(rho,y,W,detval,sige,c,T)
% PURPOSE: evaluate the conditional distribution of rho given sige
%  1st order spatial autoregressive model using sparse matrix algorithms
% ---------------------------------------------------
%  USAGE:cout = c_far(rho,y,W,detval,sige,c,T)
%  where:  rho  = spatial autoregressive parameter
%          y    = dependent variable vector
%          W    = spatial weight matrix
%        detval = an (ngrid,2) matrix of values for det(I-rho*W)
%                 over a grid of rho values
%                 detval(:,1) = determinant values
%                 detval(:,2) = associated rho values
%          sige = sige value
%          c    = optional prior mean for rho
%          T    = optional prior variance for rho
% ---------------------------------------------------
%  RETURNS: a conditional used in Metropolis-Hastings sampling
%  NOTE: called only by far_g
%  --------------------------------------------------
%  SEE ALSO: far_g, c_sar, c_sac, c_sem
% ---------------------------------------------------
```

```
i1 = find(detval(:,2) <= rho + 0.005);
i2 = find(detval(:,2) <= rho - 0.005);
i1 = max(i1); i2 = max(i2);
index = round((i1+i2)/2);
detm = detval(index,1); n = length(y);
z = speye(n) - rho*sparse(W);
if nargin == 5, % diffuse prior
epe = (n/2)*log(y'*z'*z*y);
elseif nargin == 7 % informative prior
epe = (n/2)*log(y'*z'*z*y + (rho-c)^2/T);
end;
cout = -epe -(n/2)*log(sige) + detm;
```

In the function **c_far**, we find the determinant value that is closest to the ρ value for which we are evaluating the conditional distribution. This is very fast in comparison to calculating the determinant. Since we need to carry out a large number of draws, this approach works better than computing determinants for every draw. Note that in the case of maximum likelihood estimation, the opposite was true. There we only needed to 10 to 20 evaluations of the likelihood function, making the initial grid calculation approach of Pace and Barry much slower.

The other point to note about our conditional distribution function **c_far** is that we allow for an informative prior mean and variance for the spatial autoregressive parameter ρ in the model.

The function **far_g** that implements the Gibbs sampler for this model is shown below, where we rely on a structure variable 'prior' to input information regarding the Bayesian priors for ρ and the hyperparameter r regarding heteroscedasticity.

```
function results = far_g(y,W,ndraw,nomit,prior,start)
% PURPOSE: Gibbs sampling estimates of the 1st-order Spatial
%          model: y = alpha + rho*W*y + e,    e = N(0,sige*V),
%          V = diag(v1,v2,...vn), r/vi = ID chi(r)/r, r = Gamma(m,k)
%          rho = N(c,T),  sige = gamma(nu,d0)
%----------------------------------------------------------------
% USAGE: result =  far_g(y,W,ndraw,nomit,prior,start)
% where: y = nobs x 1 independent variable vector
%         W = nobs x nobs 1st-order contiguity matrix (standardized)
%       ndraw = # of draws
%       nomit = # of initial draws omitted for burn-in
%       prior = a structure variable for prior information input
%               prior.rho,  prior mean for rho,  c above, default = diffuse
%               priov.rcov, prior rho  variance, T above, default = diffuse
%               prior.nu,   informative Gamma(nu,d0) prior on sige
%               prior.d0    informative Gamma(nu,d0) prior on sige
%                           default for above: nu=0,d0=0 (diffuse prior)
%               prior.rval, r prior hyperparameter, default=4
%               prior.m,    informative Gamma(m,k) prior on r
%               prior.k,    informative Gamma(m,k) prior on r
%               prior.rmin, (optional) min value of rho to use in sampling
%               prior.rmax, (optional) max value of rho to use in sampling
%       start = (optional) (2x1) vector of rho, sige starting values
%               (defaults, rho = 0.5, sige = 1.0)
%---------------------------------------------------------------
```

```
% RETURNS: a structure:
%          results.meth   = 'far_g'
%          results.pdraw  = rho draws (ndraw-nomit x 1)
%          results.sdraw  = sige draws (ndraw-nomit x 1)
%          results.vmean  = mean of vi draws (nobs x 1)
%          results.rdraw  = r-value draws (ndraw-nomit x 1)
%          results.pmean  = rho prior mean    (if prior input)
%          results.pstd   = rho prior std dev (if prior input)
%          results.nu     = prior nu-value for sige (if prior input)
%          results.d0     = prior d0-value for sige (if prior input)
%          results.r      = value of hyperparameter r (if input)
%          results.m      = m prior parameter (if input)
%          results.k      = k prior parameter (if input)
%          results.nobs   = # of observations
%          results.ndraw  = # of draws
%          results.nomit  = # of initial draws omitted
%          results.y      = actual observations
%          results.yhat   = mean of posterior for y-predicted (nobs x 1)
%          results.time   = time taken for sampling
%          results.accept = acceptance rate
%          results.pflag  = 1 for prior, 0 for no prior
%          results.rmax  = 1/max eigenvalue of W (or rmax if input)
%          results.rmin  = 1/min eigenvalue of W (or rmin if input)
%----------------------------------------------------------------
% NOTES: use either improper prior.rval
%        or informative Gamma prior.m, prior.k, not both of them
%----------------------------------------------------------------
[n n2] = size(W);
if n ~= n2, error('far: Wrong size 1st-order contiguity matrix'); end;
nu = 0; d0 = 0; mm = 0; c = 0; T = 1000.0; rval = 4; % default values
pflag = 0; rflag = 0; rho = 0.5; sige = 1.0; % default starting values
if nargin == 6 % user supplied starting values
rho = start(1,1); sige = start(2,1);
fields = fieldnames(prior); nf = length(fields);
for i=1:nf
    if strcmp(fields{i},'nu'),       nu = prior.nu;
    elseif strcmp(fields{i},'d0'),   d0 = prior.d0;
    elseif strcmp(fields{i},'rho'),  c = prior.rho; pflag = 1;
    elseif strcmp(fields{i},'rcov'), T = prior.rcov;
    elseif strcmp(fields{i},'rval'), rval = prior.rval;
    elseif strcmp(fields{i},'m'),    mm = prior.m; kk = prior.k;
      rval = gamm_rnd(1,1,mm,kk); % initial rval
    elseif strcmp(fields{i},'rmin')
    lmin = prior.rmin; lmax = prior.rmax; rflag = 1;
    end;
end;
elseif nargin == 4 % user wants all default values
else, error('Wrong # of arguments to far_g'); end;
if (c == 0 & T == 1000.0), pflag = 0; end;
V = ones(n,1); in = ones(n,1); vi = in; ys = y.*sqrt(V); % initialize
bsave = zeros(ndraw-nomit,1);    % allocate storage for results
ssave = zeros(ndraw-nomit,1); vmean = zeros(n,1); yhat = zeros(n,1);
if mm~= 0, rsave = zeros(ndraw-nomit,1); end;
t0 = clock;
if rflag == 0, opt.tol = 1e-3; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
lmin = 1/lambda(2);        lmax = 1/lambda(1);
```

```
end;
results.rmax = lmax; results.rmin = lmin;
% compute a detval vector based on Pace and Barry's approach
rvec = lmin-0.005:.005:lmax+0.005; spparms('tight');
z = speye(n) - 0.1*sparse(W); p = colmmd(z);
niter = length(rvec); detval = zeros(niter,2);
for i=1:niter;
    rho = rvec(i); z = speye(n) - rho*sparse(W); [l,u] = lu(z(:,p));
    detval(i,1) = sum(log(abs(diag(u)))); detval(i,2) = rho;
end;
cc = 0.2; rtmp = zeros(nomit,1); iter = 1; cnt = 0; cntr = 0;
while (iter <= ndraw);        % start sampling;
% update sige;
nu1 = n + nu; e = ys - rho*sparse(W)*ys; d1 = d0 + e'*e;
chi = chis_rnd(1,nu1); t2 = chi/d1; sige = 1/t2;
% update vi
e = y - rho*sparse(W)*y; chiv = chis_rnd(n,rval+1);
vi = ((e.*e./sige) + in*rval)./chiv; V = in./vi;
ys = y.*sqrt(V);
% update rval (if needed)
if mm ~= 0, rval = gamm_rnd(1,1,mm,kk); end;
% metropolis step to get rho update
if pflag == 0;                           % case of diffuse prior
rhox = c_far(rho,ys,W,detval,sige);      % c_rho evaluates conditional function
else                                     % an informative prior
rhox = c_far(rho,ys,W,detval,sige,c,T); % c_rho evaluates conditional function
end;
rho2 = rho + cc*randn(1); accept = 0;
 while accept == 0; % rejection bounds on rho
  if ((rho2 > lmin) & (rho2 < lmax)), accept = 1;
  else, rho2 = rho + cc*randn(1); cntr = cntr+1;
  end;
 end;     % end of rejection for rho
if pflag == 0                            % case of diffuse prior
rhoy = c_far(rho2,ys,W,detval,sige);     % c_rho evaluates conditional function
else                                     % an informative prior
rhoy = c_far(rho2,ys,W,detval,sige,c,T);% c_rho evaluates conditional function
end;
ru = unif_rnd(1,0,1);
if ((rhoy - rhox) > exp(1)), p = 1;
else, ratio = exp(rhoy-rhox); p = min(1,ratio);
end;
if (ru < p), rho = rho2; rtmp(iter,1) = rho; end;
  if (iter >= nomit);
    if iter == nomit % update cc based on initial draws
        tst = 2*std(rtmp(1:nomit,1));
        if tst > 0.05,cc = 2*std(rtmp(1:nomit,1)); end;
    end;
   ssave(iter-nomit+1,1) = sige; bsave(iter-nomit+1,1) = rho;
   vmean = vmean + vi;
   yhat = yhat + randn(n,1).*sqrt(sige*vi) + rho*sparse(W)*y;
    if mm~= 0, rsave(iter-nomit+1,1) = rval; end;
  end; % end of if iter > nomit
iter = iter+1;
end; % end of sampling loop
vmean = vmean/(iter-nomit-1); yhat = yhat/(iter-nomit-1);
gtime = etime(clock,t0);
```

```
results.accept = 1 - cntr/(iter+cntr);
results.meth  = 'far_g'; results.pdraw = bsave; results.sdraw = ssave;
results.vmean = vmean; results.yhat = yhat;
if pflag == 1
results.pmean = c; results.pstd  = sqrt(diag(T));
results.nu = nu; results.d0 = d0;
end;
results.nobs  = n; results.ndraw = ndraw; results.nomit = nomit;
results.time = gtime; results.y = y; results.nvar = 1;
results.pflag = pflag;
if mm~= 0
results.rdraw = rsave; results.m = mm; results.k = kk;
else, results.r = rval; results.rdraw = 0;
end;
```

4.2.2 Using the function far_g()

As the documentation makes clear, there are a number of user options to facilitate models based on alternative prior specifications. We will discuss and illustrate the use of these alternative approaches to using the model.

First, we illustrate a diffuse prior for ρ and σ along with a prior belief in homoscedastic disturbances with example 4.3 that relies on the Gypsy moth data set introduced in Chapter 1.

```
% ----- Example 4.3 Using the far_g() function
load moth.dat;      % moth counts 1986 to 1993 for 68 Michigan counties
load first.dat;     % non-standardized 1st-order spatial weight matrix
W = normw(first);   % standardize the contiguity matrix
y91 = moth(:,6);    % 1991 moth counts for 68 counties
ydev = y91 - mean(y91); % put data in deviations from means form
% do maximum likelihood for comparison
resl = far(ydev,W);
ndraw = 1100; nomit = 100;
prior.rval = 30; % homoscedastic model,
% diffuse prior for rho is the default
resg = far_g(ydev,W,ndraw,nomit,prior);
prt(resl); prt(resg);
```

For comparison, we also produce maximum likelihood estimates which should be similar given our use of a diffuse prior for ρ and σ as well as the homoscedastic prior for the v_i terms. The results from example 4.3 are shown below where we see a warning printed by MATLAB indicating that matrix inversion in the **far** function was ill-conditioned. A check of line 101 in the function **far.m** shows that the warning arose when attempting to compute the $t-$statistic for our parameter ρ using formulas based on the theoretical information matrix. This warning should make us suspect inaccuracy in the $t-$statistic reported for the maximum likelihood estimate.

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.130879e-16.
> In MATLAB5:Toolbox:jpl:spatial:far.m at line 101

First-order spatial autoregressive model Estimates
```

```
R-squared       =    0.6881
sigma^2         = 27459712.7303
Nobs, Nvars     =     68,     1
log-likelihood  =       -1356.3526
# of iterations =     13
min and max rho =   -1.2918,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
rho              0.853698         9.941171         0.000000

Gibbs sampling First-order spatial autoregressive model
R-squared       =    0.6998
sigma^2         = 13618041.4212
r-value         =     30
Nobs, Nvars     =     68,     1
ndraws,nomit    =   1100,   100
acceptance rate =    0.5416
time in secs    =   20.9827
min and max rho =   -1.2918,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
rho              0.892970        18.122507         0.000000
```

The Bayesian and maximum likelihood estimates for ρ are similar as is the fit indicated by the $R-$squared statistic. The $t-$statistic from Gibbs sampling is nearly twice the maximum likelihood value. Estimates of σ^2 are also quite different, with the maximum likelihood estimate twice the magnitude of that from the Gibbs sampler. This would of course explain the difference in the $t-$statistics, since the $t-$statistic involves the parameter σ.

The printed output for the Bayesian FAR model reports the time taken to carry out the draws as well as the acceptance rate. We will see in the next example that setting a smaller value for the hyperparameter r speeds up the Gibbs sampler. Another way to increase the speed of the sampler is to input limits for the feasible range on ρ so these need not be computed.

What explains the large difference in the estimates for σ^2 produced by the two models? A plot of the posterior means for the parameters v_i is shown in Figure 4.4, where we see evidence of non-constant variance or outliers indicated by the large v_i estimates. These relatively large estimates arise even with a homoscedastic prior imposed by the large value of $r = 30$ used to estimate the model. This indicates that the sample data contradicts our homoscedastic prior, suggesting a more appropriate model would rely on a heteroscedastic prior based on a small value for r.

Bayesian methods are often used to solve problems of ill-conditioning in regression problems and we might take advantage of our ability to specify a prior for the parameter ρ in the Bayesian model to overcome the possible ill-conditioning in this problem. To specify a prior for the parameter ρ, we can rely on an estimate based on the sample data from the previous year, 1990. In addition, we will rely on a prior setting for the hyperparameter $r = 4$ to robustify against the heteroscedasticity.

Example 4.4 illustrates how to accomplish estimates that rely on the poste-

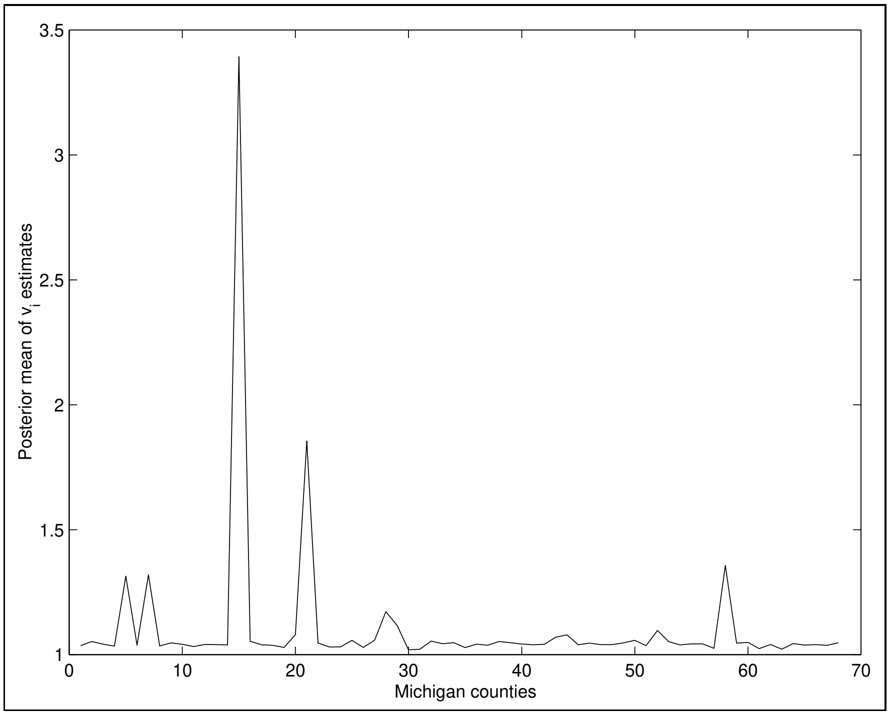

Figure 4.4: Posterior means for v_i estimates

rior mean and variance for ρ from the 1990 sample as a prior mean for the 1991 sample.

```
% ----- Example 4.4 An informative prior for far_g()
load moth.dat;     % moth counts 1986 to 1993 for 68 Michigan counties
load first.dat;    % non-standardized 1st-order spatial weight matrix
W = normw(first);  % standardize the contiguity matrix
y90 = moth(:,5);   % 1990 moth counts for 68 counties
y91 = moth(:,6);   % 1991 moth counts for 68 counties
ydev = y90 - mean(y90); % put data in deviations from means form
ndraw = 1100; nomit = 100;
prior.rval = 4;    % heteroscedastic model,
% diffuse prior for rho is the default
resg = far_g(ydev,W,ndraw,nomit,prior);
% recover the posterior mean and variance
prior.rho = mean(resg.pdraw);
prior.rcov = std(resg.pdraw)^2;
ydev = y91 - mean(y91);
resg = far_g(ydev,W,ndraw,nomit,prior);
prt(resg);
```

The results from example 4.4 are shown below. Notice that the printing function detects the presence of an informative prior and prints information regarding the prior mean and standard deviation used in the problem. The time required for this model was 15 seconds compared to nearly 21 seconds for the model based on $r = 30$. This speed difference is due to the fact that the function **chis_rnd** uses rejection sampling to produce random draws from the chi-squared distribution. The rejection rate increases for larger values of r, so this slows down the function slightly. I will argue below that an optimal strategy is to set values for r between 2 and 7 for all Bayesian FAR model estimation problems, which eliminates this as a concern.

These estimation results point to an even smaller estimate for σ^2 as they should. The maximum likelihood model estimates $\sigma^2 I_n$, whereas the Bayesian model estimates $\sigma^2 V$ and we have already seen that $V \neq I_n$ for this sample data set.

```
Gibbs sampling First-order spatial autoregressive model
R-squared       =    0.6923
sigma^2         = 10870996.0768
r-value         =      4
Nobs, Nvars     =     68,     1
ndraws,nomit    =   1100,   100
acceptance rate =    0.5740
time in secs    =   15.1158
min and max rho =   -1.2917,   1.0000
***************************************************************
Variable       Prior Mean    Std Deviation
rho              0.826548         0.077213
***************************************************************
      Posterior Estimates
Variable      Coefficient      t-statistic    t-probability
rho              0.859211        13.603231         0.000000
```

The mean of the posterior estimates for v_i returned by the model in example 4.4 with $r = 4$ are shown in Figure 4.5, where we see very large departures from unity. This explains the large difference between the maximum likelihood and Bayesian estimate for σ^2. It also suggests the Bayesian estimate is closer to truth.

One question that arises regards setting a value for the hyperparameter r in the Bayesian FAR model. It has been my experience that a good rule-of-thumb is to use values of r between 2 and 7 for all models. If the sample data do not contain outliers or non-constant variance, these values for r produce v_i estimates that are relatively constant and close to unity. If on the other hand, heteroscedasticity exists, these settings will allow for sufficient divergence of the v_i parameters from unity to accommodate the non-constant variance or robustify against the outliers. Posterior estimates for the other parameters in the spatial autoregressive models tend not to differ much over a range of values for r from 2 to 7.

Another possibility is to assign a proper prior for the parameter r. Use of a single setting for this hyperparameter in the model is improper from a strict Bayesian perspective. The function **far_g** allows the user to specify a Gamma(m, k) prior for the parameter r in the problem. Recall, the Gamma(m, k) distribution has a mean of m/k and variance equal to m/k^2.

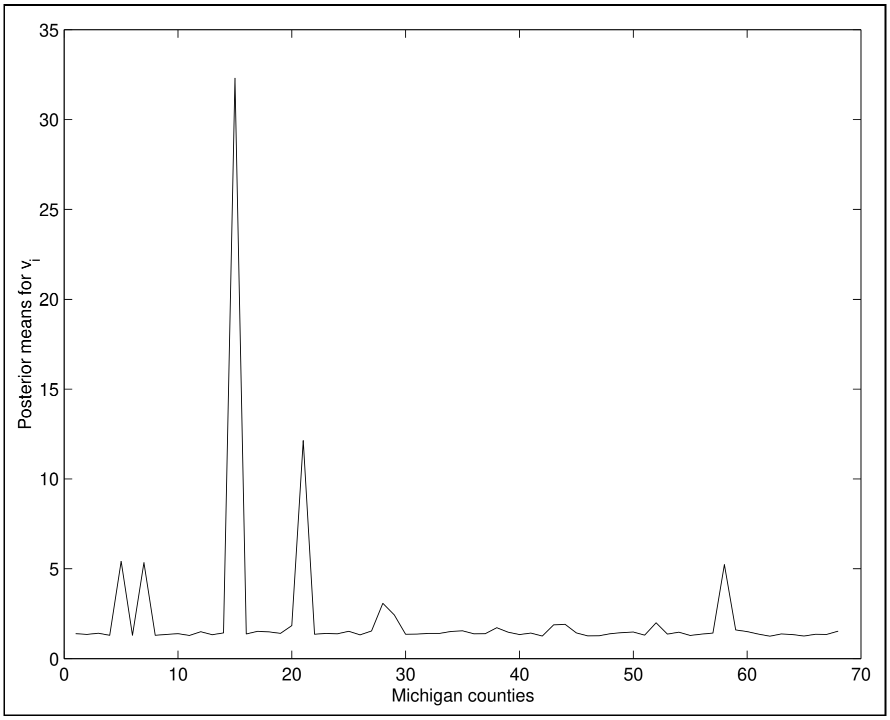

Figure 4.5: Posterior v_i estimates based on $r = 4$

A setting of $m = 8, k = 2$ indicates a prior mean of 4 and variance of 2, which is consistent with my rule-of-thumb range for r between 2 and 7.

Example 4.5 illustrates use of an informative Gamma(8,2) prior for r and eliminates the informative prior for ρ as a test of how sensitive the posterior estimate in example 4.4 is to the use of a prior for ρ and the improper prior setting of $r = 4$.

```
% ----- Example 4.5 An informative prior for r
load moth.dat;     % moth counts 1986 to 1993 for 68 Michigan counties
load first.dat;    % non-standardized 1st-order spatial weight matrix
W = normw(first);  % standardize the contiguity matrix
y91 = moth(:,6);   % 1991 moth counts for 68 counties
ydev = y91 - mean(y91); % standardize y
ndraw = 1100; nomit = 100;
prior.m = 8;       % informative prior for rval
prior.k = 2;       % with mean=4, variance = 2
% diffuse prior for rho is the default
resg = far_g(ydev,W,ndraw,nomit,prior);
prt(resg);plt(resg);
```

One point to note regarding use of an informative prior for r is that a larger number of draws may be needed to achieve convergence. This is because we are sampling on another parameter in the model. The use of an improper prior produces estimates that are conditional on the value of r that we set, whereas use of a proper prior distribution results in unconditional estimates. The estimates are unconditional because they represent an average over the values sampled for r, producing a situation where this parameter is "integrated out" of the posterior. My applied experience suggests that the estimates based on an improper setting of r between 2 and 7 versus the use of a proper Gamma(m, k) prior centered on this range seldom differ. The results from example 4.5 shown below confirm this. We increased the draws to 2100 for this model as indicated in the printed output.

```
Gibbs sampling First-order spatial autoregressive model
R-squared       =    0.6841
sigma^2         = 11228592.3742
rmean           =    4.0104
Nobs, Nvars     =     68,     1
ndraws,nomit    =   2100,   100
acceptance rate =    0.5692
time in secs    =   27.2387
min and max rho =   -1.2918,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
rho              0.840064        11.313753         0.000000
```

We see that the estimates are reasonably insensitive to the use of the informative prior for ρ based on the 1990 sample and the improper setting of $r = 4$. A function **plt** exists that presents the actual versus the mean of the posterior for the predicted values, the residuals and a plot of the posterior density for the parameter ρ. In the next section we will demonstrate the use of a function **pltdens** to produce a nonparametric density estimate of any parameter based on the draws from the Gibbs sampler. The graphical presentation of results produced by **plt** for example 4.5 are shown in Figure 4.6

An important issue in using Gibbs sampling is convergence of the sampler to the posterior distribution. We know from theory that the sampler converges in the limit as $n \to \infty$, but in any applied problem one must determine how many draws to make with the sampler. Ad-hoc convergence tests usually work well in simple regression models of the type considered here. For example, Smith and Roberts (1992) proposed a test they label the 'felt-tip pen test', that compares smoothed histograms or distributions from earlier draws in the sequence of passes through the sampler to those from later draws. If the two distributions are similar — within the tolerance of the felt-tip pen, convergence is assumed to have taken place.

The next section is devoted to an explanation and demonstration of functions that carry out convergence diagnostics for a sequence of Gibbs draws. These may be helpful in applied spatial econometric problems involving any of the Bayesian spatial autoregressive models.

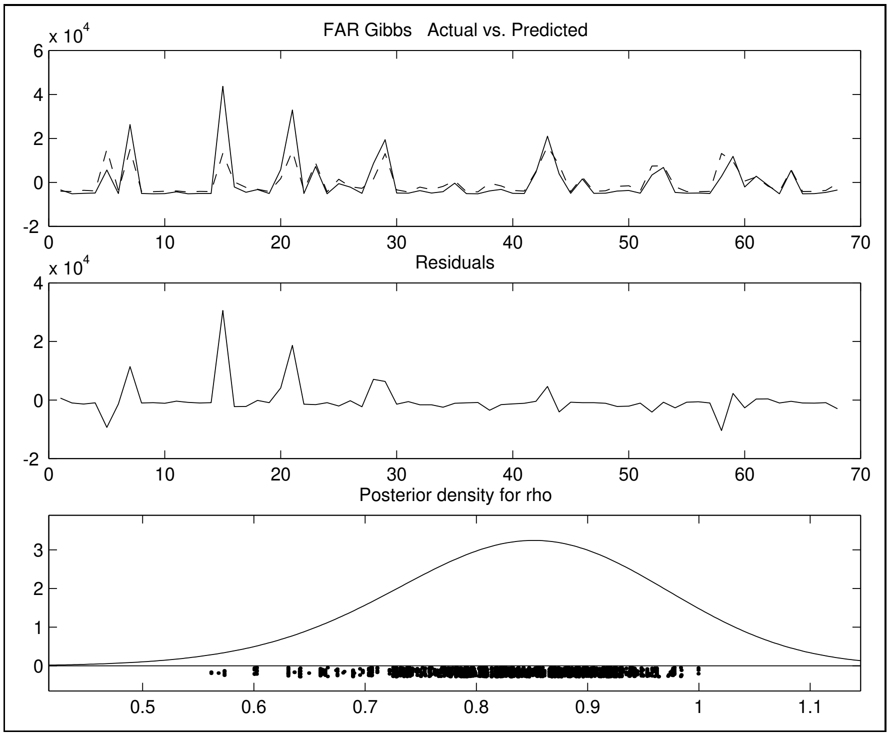

Figure 4.6: Graphical output for far_g

4.3 Monitoring convergence of the sampler

There is some evidence that linear regression models exhibit rapid convergence, and we might suppose this to be the case for spatial autoregressive models as well. It should be noted that once convergence occurs, we need to continue making passes to build up a sample from the posterior distribution which we use to make inferences about the parameters. As one might expect, convergence is a function of how complicated the set of conditional distributions are. For example, Geweke (1993) found that Gibbs sampling the tobit censured regression model produced poor results with 400 passes and much better results with 10,000 passes. We will illustrate tobit censored spatial autoregressive models in Chapter 5.

Best et al., 1995 provide a set of Splus functions that implement six different MCMC convergence diagnostics, some of which have been implemented in a MATLAB function **coda**. This function provides: autocorrelation estimates, Rafterty-Lewis (1995) MCMC diagnostics, Geweke (1992) NSE, (numerical standard errors) RNE (relative numerical efficiency) estimates, and Geweke Chi-squared test on the means from the first 20% of the sample versus the last 50%. We describe the role of each of these diagnostic measures using the

Bayesian FAR model in example 4.6.

```
% ----- Example 4.6 Using the coda() function
load Wmat.dat;    % standardized 1st-order contiguity matrix
n=49;             % from Anselin (1988) Columbus data set
W = Wmat; IN = eye(n);
rho = 0.75;       % true value of rho
y = inv(IN - rho*W)*randn(n,1); % general FAR model y
ndraw = 1100; nomit = 100;
prior.rval = 30;
% rely on a diffuse prior for rho
result = far_g(y,W,ndraw,nomit,prior);
vnames = strvcat('rho','sigma');
coda([result.pdraw result.sdraw],vnames);
```

The example program generates a sample y containing 49 observations based on a first-order contiguity matrix from Anselin's (1988) Columbus neighborhood crime data set. A Bayesian FAR model is estimated using the **far_g** function and the sample of 1000 draws for ρ and σ returned by the function are input to the function **coda** that carries out the convergence diagnostics.

The sample Gibbs draws for the parameter ρ are in the results structure variable, 'result.pdraw' which we send down to the **coda** function to produce convergence diagnostics along with the draws for the parameter σ which are in 'result.sdraw'. These two vectors of draws are combined to form a matrix "on-the-fly" using the square brackets. We also include a variable names string for labeling the output from **coda**.

The function **coda** uses a MATLAB variable 'nargout' to determine if the user has called the function with an output argument. If so, the function returns a result structure variable that can be printed later using the **prt** function. In the case where the user supplies no output argument, (as in example 4.6 above) the convergence diagnostics will be printed to the MATLAB command window.

A partial listing of the documentation for the function **coda** is shown below, where the form of the structure variable returned by **coda** when called with an output argument has been eliminated to save space.

```
PURPOSE: MCMC convergence diagnostics, modeled after Splus coda
------------------------------------------------------
USAGE:              coda(draws,vnames,info)
       or: result = coda(draws)
where: draws = a matrix of MCMC draws (ndraws x nvars)
      vnames = (optional) string vector of variable names (nvar x 1)
        info = (optional) structure setting input values
      info.q = Raftery quantile           (default = 0.025)
      info.r = Raftery level of precision (default = 0.01)
      info.s = Rafterty probability for r (default = 0.950)
      info.p1 = 1st % of sample for Geweke chi-sqr test (default = 0.2)
      info.p2 = 2nd % of sample for Geweke chi-sqr test (default = 0.5)
------------------------------------------------------
NOTES: you may supply only some of the info-structure arguments
       the remaining ones will take on default values
------------------------------------------------------
RETURNS: output to command window if nargout = 0
         autocorrelation estimates
```

```
Rafterty-Lewis MCMC diagnostics
Geweke NSE, RNE estimates
Geweke chi-sqr prob on means from info.p1 vs info.p2
a results structure if nargout = 1
```

The results from executing the example program as are follows:

```
MCMC CONVERGENCE diagnostics
Based on sample size =       1000
Autocorrelations within each parameter chain
Variable        Lag 1        Lag 5       Lag 10       Lag 50
rho             0.761        0.263        0.128       -0.011
sigma           0.214        0.001       -0.084        0.016

Raftery-Lewis Diagnostics for each parameter chain
(q=0.0250, r=0.010000, s=0.950000)
Variable       Thin       Burn   Total(N)     (Nmin)     I-stat
rho               1         25       6703        937      7.154
sigma             1         25       6703        937      7.154

Geweke Diagnostics for each parameter chain
Variable      Mean      std dev      NSE iid      RNE iid
rho       0.781235     0.097552     0.003085     1.000000
sigma     1.217190     0.341255     0.010791     1.000000

Variable    NSE 4%      RNE 4%      NSE 8%      RNE 8%      NSE 15%     RNE 15%
rho       0.007297    0.178744    0.006446    0.229030    0.005731    0.289777
sigma     0.013207    0.667631    0.011375    0.900001    0.009856    1.198759

Geweke Chi-squared test for each parameter chain
First 20% versus Last 50% of the sample
Variable             rho
NSE estimate         Mean       N.S.E.  Chi-sq Prob
i.i.d.           0.784250     0.003733     0.337867
4% taper         0.785617     0.007915     0.630768
8% taper         0.784889     0.007811     0.631370
15% taper        0.784199     0.007105     0.616461

Variable           sigma
NSE estimate         Mean       N.S.E.  Chi-sq Prob
i.i.d.           1.208643     0.012847     0.715432
4% taper         1.209183     0.015739     0.756729
8% taper         1.209553     0.015622     0.750423
15% taper        1.209950     0.014466     0.727674
```

The role of each convergence diagnostic measure is described in the following sections.

4.3.1 Autocorrelation estimates

The role of the time-series autocorrelation estimates is to provide an indication of how much independence exists in the sequence of ρ and σ parameter draws. From time-series analysis we know that if $\rho, i = 1, \ldots, n$ is a stationary correlated process, then $\bar{\rho} = (1/n)\sum_{i=1}^{n} \rho_i$ is a consistent estimate of $E(\rho)$ as $n \rightarrow \infty$, so it is permissible to simulate correlated draws from the posterior distribution to summarize the features of the posterior. This is provided that we produce a large enough sample of draws, and the amount of correlation plays a role in determining the number of draws necessary. A high degree of autocorrelation indicates that we may need to carry out more draws to achieve a sample of sufficient size to draw accurate posterior estimates.

The **coda** results indicate that our draws for the parameter ρ exhibit a large autocorrelation at lag 1, but then tails off at lags 5, 10 and 50. The autocorrelation structure for the parameter σ reflects a smaller value at lag 1 and then tails off as well. The autocorrelation structure for these two parameters indicate that we may need to produce a larger number of draws with our Gibbs sampler.

4.3.2 Raftery-Lewis diagnostics

Raftery and Lewis (1992a, 1992b, 1995) proposed a set of diagnostics that they implemented in a FORTRAN program named Gibbsit, which were converted to a MATLAB function **raftery**. This function is called by **coda**, but can also be used independently of **coda**. Given some output from a Gibbs (or MCMC) sampler, Raftery and Lewis provide an answer regarding how long to monitor the chain of draws that is based on the accuracy of the posterior summaries desired by the user. They require that the user specify three pieces of information that are set to default values by **coda**, or can be user-defined with a structure variable 'info' that is an input argument to the **coda** and **raftery** functions.

1. Which quantiles of the marginal posteriors are you interested in? Usually the answer is the 2.5% and 97.5% points, because these are the basis for a 95% interval estimate. This information is set using 'info.q', which has a default value of 0.025.

2. What is the minimum probability needed to archive the accuracy goals. A default value of 95% is used, or can be set by the user with 'info.s', which has a default value of 0.95.

3. How much accuracy is desired in the estimated quantiles? Raftery and Lewis specify this using the area to the left (in the case of info.q = 0.025) or right (in the case of info.q=0.975) of the reported quantile in the CDF. By default info.r=0.01, so that nominal reporting based on a 95% interval using the 0.025 and 0.975 quantile points should result in actual posterior values that lie between 0.94 and 0.96.

Given our draws for ρ, **raftery** dichotomizes the draws using a binary time-series that is unity if $\rho_i \leq$ 'info.q' and zero otherwise. This binary chain should be approximately Markovian so standard results for two-state Markov chains can be used to estimate how long the chain should be run to achieve the desired accuracy for the chosen quantile 'info.q'.

The function **coda** prints out three different estimates from the **raftery** function. A thinning ratio which is a function of the amount of autocorrelation in the draws, the number of draws to use for 'burn-in' before beginning to sample the draws for purposes of posterior inference, and the total number of draws needed to achieve the accuracy goals.

Some terminology will help to understand the **raftery** output. It is always a good idea to discard a number of initial draws referred to as "burn-in" draws for the sampler. Starting from arbitrary parameter values makes it unlikely that initial draws come from the stationary distribution needed to construct posterior estimates. Another practice followed by researchers involves saving only every third, fifth, tenth, etc. draw since the draws from a Markov chain are not independent. This practice is labeled "thinning" the chain. Neither thinning or burn-in are mandatory in Gibbs sampling and they tend to reduce the effective number of draws on which posterior estimates are based.

From the **coda** output, we see that the thinning estimate provided by **raftery** in the second column is 1, which seems inconsistent with the lag 1 autocorrelation in the sequence of draws. It is however consistent with the fact that the autocorrelation structure tails off rapidly for both ρ and σ. The third column reports that only 25 draws are required for burn-in, which is quite small. In the fourth column, we find the total number of draws needed to achieve the desired accuracy for each parameter. This is given as 6,703, which greatly exceeds the 1,000 draws we used. It might be advisable to run the sampler again using this larger number of draws.

On the other hand, a call to the function **raftery** with the desired accuracy ('info.r') set to 0.02, so that nominal reporting based on a 95% interval using the 0.05 and 0.95 quantile points should result in actual posterior values that lie between 0.93 and 0.97, produces the results shown below. These indicate that 1,695 draws would be adequate to produce this desired level of accuracy for the posterior.

```
% ----- Example 4.7 Using the raftery() function
q = 0.025;
r = 0.02;
s = 0.95;
res = raftery([result.pdraw result.sdraw],q,r,s);
prt(res,vnames);
Raftery-Lewis Diagnostics for each parameter chain
(q=0.0250, r=0.020000, s=0.950000)
Variable       Thin        Burn   Total(N)     (Nmin)     I-stat
rho               1          25       1695        235      7.213
sigma             1          25       1695        235      7.213
```

The Nmin reported in the fifth column represents the number of draws that would be needed if the draws represented an iid chain, which is not true in our case as indicated by the autocorrelation structure. Finally, the $i-$statistic is the ratio of the fourth to the fifth column. Raftery and Lewis indicate that values exceeding 5 for this statistic are indicative of convergence problems with the sampler and the need to carry out more draws.

4.3.3 Geweke diagnostics

The function **coda** also produces estimates of the numerical standard errors (NSE) and relative numerical efficiency (RNE) proposed by Geweke (1992). Using spectral analysis of time-series methods, we can produce an estimate of the variance of the ρ parameters we are interested in based on the sampled values using:

$$\mathrm{var}(\hat{\rho}_i) = S(0)/k \tag{4.13}$$

where $S(0)$ is the spectral density of ρ_i evaluated at $\omega = 0$. Issues arise in how one approximates $S(\omega)$, so alternative tapering of the spectral window is used. The **coda** function reports estimates of the NSE and RNE based on 4%, 8% and 15% tapering or truncation of the periodgram window. The MATLAB functions that implement these calculations are adaptions of routines provided by Geweke and Chib, that can be found on the internet at http://www.econ.umn.edu/bacc.

The first set of NSE and RNE estimates reported are based on the assumption that the draws come from an iid process. These are reported along with the means and standard deviations of the chain of draws. A second set of NSE and RNE estimates are reported based on alternative tapering of the spectral window, where the non-iid nature of the draws is taken into account by the NSE and RNE estimates. Dramatic differences between these estimates would lead one to rely on the latter set of estimates, as these differences would reflect autocorrelation in the draws.

The RNE estimates provide an indication of the number of draws that would be required to produce the same numerical accuracy if the draws represented had been made from an iid sample drawn directly from the posterior distribution. In our example, the RNE's for σ are close to unity, indicative of an iid sample. RNE estimates greater than unity, say around 3, would indicate that only 33% of the number of draws would be required to achieve the same accuracy from an iid set of draws.

These results are produced by a call to the MATLAB function **momentg**, which is called by **coda**. As with the function **raftery**, this function can be called by itself without invoking **coda**. As an example:

```
% ----- Example 4.8 Geweke's convergence diagnostics
result = far_g(y,W,ndraw,nomit,prior);
vnames = strvcat('rho','sigma');
geweke = momentg([result.pdraw result.sdraw]);
prt(geweke,vnames);
Geweke Diagnostics for each parameter chain
Variable       Mean      std dev     NSE iid      RNE iid
rho        0.781235     0.097552    0.003085     1.000000
sigma      1.217190     0.341255    0.010791     1.000000

Variable    NSE 4%       RNE 4%      NSE 8%       RNE 8%      NSE 15%      RNE 15%
rho        0.007297     0.178744    0.006446     0.229030    0.005731     0.289777
sigma      0.013207     0.667631    0.011375     0.900001    0.009856     1.198759
```

We see some evidence of the autocorrelation because the tapered estimates of the numerical standard errors increase slightly relative to those based on the iid assumption. Again, this points towards increasing the number of draws used in our Gibbs sampling estimation.

A second set of diagnostics suggested by Geweke are printed by **coda** after the NSE and RNE estimates. These diagnostics represent a test of whether the sample of draws has attained an equilibrium state based on the means of the first 20% of the sample draws versus the last 50% of the sample. If the Markov chain of draws from the Gibbs sampler has reached an equilibrium state, we would expect the means from these two splits of the sample to be roughly equal. A $Z-$test of the hypothesis of equality for these two means is carried out and the chi-squared marginal significance is reported. For our illustrative example, both parameters pass these tests. We cannot reject the hypothesis of equal means.

```
Geweke Chi-squared test for each parameter chain
First 20% versus Last 50% of the sample
Variable              rho
NSE estimate         Mean       N.S.E.  Chi-sq Prob
i.i.d.           0.784250     0.003733     0.337867
4% taper         0.785617     0.007915     0.630768
8% taper         0.784889     0.007811     0.631370
15% taper        0.784199     0.007105     0.616461

Variable            sigma
NSE estimate         Mean       N.S.E.  Chi-sq Prob
i.i.d.           1.208643     0.012847     0.715432
4% taper         1.209183     0.015739     0.756729
8% taper         1.209553     0.015622     0.750423
15% taper        1.209950     0.014466     0.727674
```

The **coda** function allows the user to specify the proportions of the sample used to carry out this test as 'info.p1' and 'info.p2' in the structure variable used to input options to **coda**. The default values based on the first 20% of the sample versus the last 50% are values used by the Splus version of CODA.

The chi-squared tests are implemented by a call inside **coda** to a MATLAB function **apm**. This function allows one to produce posterior moment estimates that represent an average over two sets of draws and can be used without invoking **coda**. This would be useful in cases where one wished to combine smaller samples from multiple MCMC or Gibbs sampling runs and examine the impact of the additional draws on NSE or test for equality of the means from the two samples. The documentation for **apm** is:

```
PURPOSE: computes Geweke's chi-squared test for
two sets of MCMC sample draws
------------------------------------------------
USAGE: result = apm(results1,results2)
where: results1 = a structure returned by gmoment
       results2 = a structure returned by gmoment
------------------------------------------------
RETURNS: a structure:
     results(i).pmean(k) = posterior mean for variable i
                           for k = nse, nse1,nse2,nse3
     results(i).nse(k)   = nse for variable i
                           for k = nse, nse1,nse2,nse3
```

```
      results(i).prob(k)  = chi-sq test prob for variable i
                            for k = nse, nse1,nse2,nse3
  -----------------------------------------------
```

As an illustration, suppose we started our **far_g** sampler at different starting values. A input variable 'start' can be used in **far_g** to set starting values for the parameters ρ and σ in the sampler as illustrated in example 4.9 below. To test convergence, we call the **apm** function with results structures returned by **momentg** based on the two sets of draws. We then use **prt** to print the means, NSE and chi-squared test results.

```
% ----- Example 4.9 Using the momentg() function
load Wmat.dat;    % standardized 1st-order contiguity matrix
n=49;             % from Anselin (1988) Columbus data set
W = Wmat; IN = eye(n);
rho = 0.75;       % true value of rho
y = inv(IN - rho*W)*randn(n,1); % general FAR model y
ndraw = 1100; nomit = 100;
prior.rval = 30;
% rely on a diffuse prior for rho
result = far_g(y,W,ndraw,nomit,prior);
ndraw1 = 1100; ndraw2 = 1100; nomit = 100;
rval = 30;                    % homoscedastic prior for r-value
start1(1,1) =  0.5; start1(2,1) = 1.0;
start2(1,1) = -0.5; start2(2,1) = 100;
result1 = far_g(y,W,ndraw1,nomit,prior,start1);
gres1 = momentg(result1.pdraw);
result2 = far_g(y,W,ndraw2,nomit,prior,start2);
gres2 = momentg(result2.pdraw);
result = apm(gres1,gres2);
prt(result)
Geweke Chi-squared test for each parameter chain
based on 2000 draws
First 50% versus Last 50% of the sample
Variable       variable 1
NSE estimate         Mean       N.S.E.  Chi-sq Prob
i.i.d.           0.788819     0.001679     0.562483
4% taper         0.788849     0.004363     0.822815
8% taper         0.788974     0.004371     0.821527
15% taper        0.788879     0.003911     0.802018
```

The results from example 4.9 indicate that the means from the two samples are equal. When executing programs that generate data as in example 4.9, a new sample of y values is generated for each program execution because we apply the random normal function **randn** without setting a seed value. If we wished to produce the same results each time, we would use the command: 'randn('seed',123456)' to fix the random number seed value.

Another useful function for examining MCMC output is the function **pltdens** from the *Econometrics Toolbox*. This function produces density plots based on a non-parametric kernel density estimator. Samples of MCMC draws can be used to produce posterior density plots with a simple call such as:

```
pltdens(result.pdraw);
```

```
% demo of pltdens options
bandwidth = 0.2; % a kernel density smoothing parameter option
positive = 1;    % a flag for densities with zero mass at negative values
kerneltype = 1;  % a Gaussian kernel type
pltdens(result.sdraw,bandwidth,positive,kerneltype);
```

4.3.4 Other tests for convergence

The conclusion from using the **coda** diagnostics point towards to increasing the number of draws used in example 4.6. It should be kept in mind that the ultimate test of how many draws are needed can be settled by comparing the inferences one would make based on a smaller set of draws to those resulting from a larger set of draws. If the inferences are identical, then the formal diagnostics may have been misleading.

Example 4.10 illustrates this approach, producing two sets of estimates, one based on 1,100 draws with the first 100 omitted and a second set of 11,000 draws with the first 1,000 omitted. We also use a value of $r = 4$ to illustrate the point made regarding using this type of value as a rule-of-thumb even in homoscedastic situations.

```
% ----- Example 4.10 Testing convergence
load Wmat.dat;    % standardized 1st-order contiguity matrix
n=49;             % from Anselin (1988) Columbus data set
W = Wmat; IN = eye(n);
randn('seed',654321);
rho = 0.5;        % true value of rho
y = inv(IN - rho*W)*randn(n,1); % general FAR model y
ndraw = 1100; nomit = 100;
prior.rval = 4;
result1 = far_g(y,W,ndraw,nomit,prior);
ndraw = 11000; nomit = 1000;
result2 = far_g(y,W,ndraw,nomit,prior);
prt(result1); prt(result2);
[h1 f1 y1] = pltdens(result1.pdraw);
[h2 f2 y2] = pltdens(result2.pdraw);
plot(y1,f1,'-k',y2,f2,'--k');
legend('1000 draws','10,000 draws');
xlabel('\rho values'); ylabel('Posterior distributions');
```

The program in example 4.10 produces kernel density estimates for the posterior distributions based on the small and large samples to graphically examine the similarity of the two posterior distributions. The printed results are shown below, where we see that the two estimates for ρ are identical to three decimal places. The estimates for σ are very close as are the $t-$statistics, $R-$squared statistics and even the acceptance rates. This provides strong evidence that only 1,000 draws are required for this problem.

```
Gibbs sampling First-order spatial autoregressive model
R-squared       =    0.1395
sigma^2         =    0.7758
r-value         =      4
Nobs, Nvars     =     49,     1
```

```
ndraws,nomit    =   1100,   100
acceptance rate =    0.8608
time in secs    =   12.0600
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
rho              0.411721         2.388924         0.020877

Gibbs sampling First-order spatial autoregressive model
R-squared       =    0.1473
sigma^2         =    0.7818
r-value         =      4
Nobs, Nvars     =     49,     1
ndraws,nomit    =  11000,  1000
acceptance rate =    0.8741
time in secs    =  114.7680
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
rho              0.411666         2.284735         0.026790
```

The graphical portrait of the two posterior distributions for ρ are also nearly identical as can be seen from Figure 4.7.

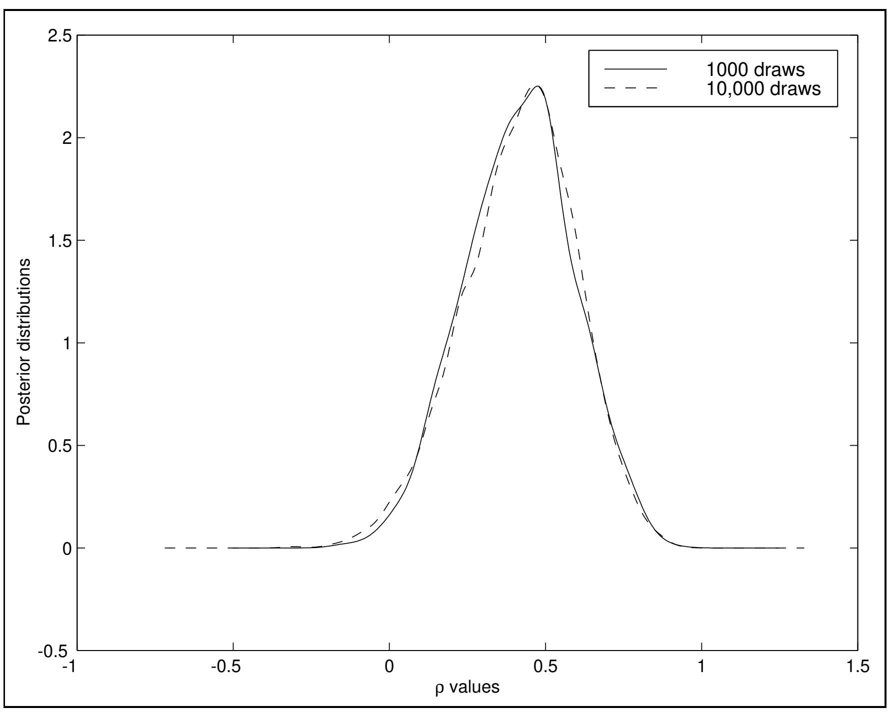

Figure 4.7: Posterior densities for ρ

Finally, note that the time required for 11,000 draws was almost exactly 10 times that required for 1,100 draws, and both times are quite reasonable.

4.4 Other Bayesian spatial autoregressive models

It should be clear that implementation of Gibbs samplers for the other spatial autoregressive models is straightforward. We need simply to determine the complete sequence of conditional distributions for the parameters in the model and code a loop to carry out the draws.

All of the spatial autoregressive models will have in common the need to produce a Metropolis within Gibbs estimate for ρ based on a conditional distribution involving the determinant $(I_n - \rho W)$. In the case of the SAC model, we need two determinants, one for $(I_n - \rho W_1)$ and another for $(I_n - \lambda W_2)$. Of course we will carry this out initially over a grid of values and store the results. These will be passed to the functions that perform the conditional distribution calculations.

There are functions **sar_g**, **sem_g** and **sac_g** that implement Gibbs sampling estimation for the Bayesian variants of the spatial autoregressive models. The function **sar_g** (which is similar to that for the other models) is shown below:

```
function results = sar_g(y,x,W,ndraw,nomit,prior,start);
% PURPOSE: Gibbs sampling estimates of the heteroscedastic
%          spatial autoregressive model:
%          y = p*Wy + XB + e, e is N(0,sige*V)
%          V = diag(v1,v2,...vn), r/vi = ID chi(r)/r, r = Gamma(m,k)
%          B = N(c,T),  sige = gamma(nu,d0), p = diffuse prior
%---------------------------------------------------
% USAGE: results = sar_g(y,x,W,ndraw,nomit,prior,start)
% where: y = dependent variable vector (nobs x 1)
%        x = independent variables matrix (nobs x nvar)
%        W = 1st order contiguity matrix (standardized, row-sums = 1)
%            (either full or sparse)
%    prior = a structure for:  B = N(c,T),  sige = gamma(nu,d0)
%            prior.beta, prior means for beta,   c above (default 0)
%            priov.bcov, prior beta covariance , T above (default 1e+12)
%            prior.rval, r prior hyperparameter, default=4
%            prior.m,    informative Gamma(m,k) prior on r
%            prior.k,    (default: not used)
%            prior.nu,   a prior parameter for sige
%            prior.d0,   (default: diffuse prior for sige)
%            prior.rmin = (optional) min rho used in sampling
%            prior.rmax = (optional) max rho used in sampling
%    ndraw = # of draws
%    nomit = # of initial draws omitted for burn-in
%    start = (optional) structure containing starting values:
%            defaults: beta=ones(k,1),sige=1,rho=0.5, V=ones(n,1)
%            start.b   = beta starting values (nvar x 1)
%            start.p   = rho starting value   (scalar)
%            start.sig = sige starting value  (scalar)
%            start.V   = V starting values (n x 1)
```

```
%---------------------------------------------------
% NOTE:  if the model contains a constant term,
%        1st column of x-matrix must contain the constant
%---------------------------------------------------
% RETURNS:  a structure:
%          results.meth  = 'sar_g'
%          results.bdraw = bhat draws (ndraw-nomit x nvar)
%          results.pdraw = rho draws  (ndraw-nomit x 1)
%          results.sdraw = sige draws (ndraw-nomit x 1)
%          results.vmean = mean of vi draws (nobs x 1)
%          results.yhat  = mean of posterior y-predicted (nobs x 1)
%          results.rdraw = r draws (ndraw-nomit x 1) (if m,k input)
%          results.bmean = b prior means, prior.beta from input
%          results.bstd  = b prior std deviations sqrt(diag(prior.bcov))
%          results.r     = value of hyperparameter r (if input)
%          results.nobs  = # of observations
%          results.nvar  = # of variables in x-matrix
%          results.ndraw = # of draws
%          results.nomit = # of initial draws omitted
%          results.y     = actual observations (nobs x 1)
%          results.yhat  = predicted values
%          results.nu    = nu prior parameter
%          results.d0    = d0 prior parameter
%          results.time  = time taken for sampling
%          results.accept= acceptance rate
%          results.rmax  = 1/max eigenvalue of W (or rmax if input)
%          results.rmin  = 1/min eigenvalue of W (or rmin if input)
% ----------------------------------------------------
% set default values
mm = 0; rval = 4; nu = 0; d0 = 0; c = zeros(k,1); T = eye(k)*1e+12;
b0 = ones(k,1); sige = 1; V = ones(n,1);  p0 = 0.5; rflag = 0;
[n junk] = size(y); results.y = y; [n1 k] = size(x); results.x = x;
if nargin == 7
 b0 = start.b; sige = start.sig; V = start.V; p0 = start.p;
 fields = fieldnames(prior); nf = length(fields);
 for i=1:nf
    if strcmp(fields{i},'rval'),     rval = prior.rval;
    elseif strcmp(fields{i},'m'),    mm = prior.m; kk = prior.k;
        rval = gamm_rnd(1,1,mm,kk);  % initial value for rval
    elseif strcmp(fields{i},'beta'), c = prior.beta;
    elseif strcmp(fields{i},'bcov'), T = prior.bcov;
    elseif strcmp(fields{i},'nu'),   nu = prior.nu;
    elseif strcmp(fields{i},'d0'),   d0 = prior.d0;
    elseif strcmp(fields{i},'rmin')
    lmin = prior.rmin; lmax = prior.rmax; rflag = 1;
    end;
end;
elseif nargin == 5 % default prior and starting values
else, error('Wrong # of arguments to sar_g');
end;
Q = inv(chol(T)); QpQ = Q'*Q; Qpc = Q'*c;
cc=0.2;  [n k] = size(x); rho = p0;
t0 = clock;
if rflag == 0, opt.tol = 1e-4; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
lmin = 1/lambda(2);        lmax = 1/lambda(1);
end;
```

```
results.rmax = lmax; results.rmin = lmin;
% compute a detval vector based on Pace and Berry's approach
rvec = lmin-0.005:.005:lmax+0.005; spparms('tight');
z = speye(n) - 0.1*sparse(W); p = colmmd(z); niter = length(rvec);
detval = zeros(niter,2);
for i=1:niter;
    rho = rvec(i); z = speye(n) - rho*sparse(W); [l,u] = lu(z(:,p));
    detval(i,1) = sum(log(abs(diag(u)))); detval(i,2) = rho;
end;
          bsave = zeros(ndraw-nomit,k); in = ones(n,1);
          if mm~= 0, rsave = zeros(ndraw-nomit,1); end;
          ssave = zeros(ndraw-nomit,1); psave = zeros(ndraw-nomit,1);
          vmean = zeros(n,1); yhat = zeros(n,1); rtmp = zeros(nomit,1);
cntr = 0; iter = 1; rho = p0;
% scale y,x for the conditional distribution function evaluation
yd = y - mean(y);
if (ones(n,1) == x(:,1))
xd = x(:,2:k) - matmul(mean(x(:,2:k)),ones(n,k-1)); cflag = 1;
else, xd = x - matmul(mean(x),ones(n,k)); cflag = 0;
end;
          while (iter <= ndraw); % start sampling;
          B = speye(n) - rho*sparse(W); ys = y.*sqrt(V); xs = matmul(x,sqrt(V));
          ysd = yd.*sqrt(V); xsd = matmul(xd,sqrt(V));
          % update beta
          xpxi = inv(xs'*xs + sige*QpQ); xpy = (xs'*B*ys + sige*Qpc);
          b0 = xpxi*xpy; bhat = norm_rnd(sige*xpxi) + b0;
          % update sige
          nu1 = n + nu; e = B*ys - xs*bhat; d1 = d0 + e'*e;
          chi = chis_rnd(1,nu1); t2 = chi/d1; sige = 1/t2;
         % update vi
         e = B*y - x*bhat; chiv = chis_rnd(n,rval+1);
         vi = ((e.*e./sige) + in*rval)./chiv; V = in./vi;
         % update rval (if needed)
         if mm ~= 0, rval = gamm_rnd(1,1,mm,kk); end;
         % metropolis step to get rho update
          bt = bhat(cflag+1:length(bhat),1);
          rhox = c_sar(rho,ysd,xsd,bt,sige,W,detval);
          accept = 0; rho2 = rho + cc*randn(1,1);
          while accept == 0
           if ((rho2 > lmin) & (rho2 < lmax)), accept = 1;
           else, rho2 = rho + cc*randn(1,1); cntr = cntr+1;
           end;
          end;
          rhoy = c_sar(rho2,ysd,xsd,bt,sige,W,detval);
          ru = unif_rnd(1,0,1);
          if ((rhoy - rhox) > exp(1)), p = 1;
          else,  ratio = exp(rhoy-rhox);  p = min(1,ratio);
          end;
              if (ru < p), rho = rho2; end;
    if iter > nomit % if we are past burn-in, save the draws
    bsave(iter-nomit,1:k) = bhat'; ssave(iter-nomit,1) = sige;
    psave(iter-nomit,1) = rho; vmean = vmean + vi;
    yhat = yhat + rho*sparse(W)*y + x*bhat;
    if mm~= 0, rsave(iter-nomit+1,1) = rval; end;
    end;
    rtmp(iter,1) = rho;
    if iter == nomit % update cc based on initial draws
```

```
        cc = 2*std(rtmp(1:nomit,1));
    end;
iter = iter + 1;
end; % end of sampling loop
vmean = vmean/(iter-nomit); yhat = yhat/(iter-nomit);
gtime = etime(clock,t0);
% find acceptance rate
results.accept = 1 - cntr/(iter+cntr); results.meth  = 'sar_g';
results.bdraw = bsave; results.pdraw = psave;
results.sdraw = ssave; results.vmean = vmean;
results.yhat  = yhat;  results.bmean = c;
results.bstd  = sqrt(diag(T)); results.nobs  = n;
results.nvar  = k; results.ndraw = ndraw;
results.nomit = nomit; results.time = gtime;
results.nu = nu; results.d0 = d0;
if mm~= 0, results.rdraw = rsave; results.m = mm; results.k = kk;
else, results.r = rval;results.rdraw = 0;
end;
```

Some implementation details regarding the code used in **sar_g** need to be discussed. The most important detail is that Metroplis sampling relies heavily on good "coverage" by the proposal density, and my experience has shown that scaling the data used to evaluate the conditional distribution for ρ (and λ) is required to achieve this coverage.

The scaling is accomplished by transforming y and X to deviation from the means form which is carried out by the following code fragment:

```
% scale y,x for the conditional distribution function evaluation
yd = y - mean(y);
if (ones(n,1) == x(:,1))
xd = x(:,2:k) - matmul(mean(x(:,2:k)),ones(n,k-1)); cflag = 1;
else, xd = x - matmul(mean(x),ones(n,k)); cflag = 0;
end;
```

These variables are then transformed by V^{-1} to produce variables 'ysd' and 'xsd' that are passed to the conditional distribution function **c_sar**. This also requires that we exclude the constant term, which is obliterated by the transformation to deviation from the means form. This exclusion is accomplished by the statement:

```
 bt = bhat(cflag+1:length(bhat),1);
```

This is inserted prior to calling **c_sar**. Note that we pass the variable 'bt' that excludes the intercept term to **c_sar**, not the vector 'bhat' that includes the intercept. This scaling accounts for the requirement that users with models including a constant term must place the vector of ones for the constant term in the first column of the matrix X, a point made in the function documentation NOTES section.

If you need convincing that scaling is essential, modify the function to use the unscaled variables 'ys' and 'xs' in the call to the conditional distribution function and examine the sequence of Gibbs draws for the parameter ρ. (If you make this modification, you also need to call **c_sar** using the 'bhat' variable in place of 'bt'.)

An important point when programming Gibbs sampling functions is that diffuse priors for all parameters in the model should result in estimates that replicate maximum likelihood results. This is a convenient way to check your Gibbs sampling functions. You will find that using the scaled variables 'ysd' and 'xsd' produces estimates that replicate maximum likelihood results, whereas the unscaled variables produce estimates that tend to center ρ on zero.

Another implementation detail is that we allow for an informative prior on the parameters β in the model, but none on ρ. This was partially a design consideration since most users will likely rely on a diffuse prior for the spatial autoregressive parameters ρ and λ in these models. It was also necessitated by the fact that a normal prior seems inappropriate for these models given the restriction on the feasible range for the spatial autoregressive parameters. LeSage (1997) suggests an alternative to using a normal prior distribution for ρ, but I believe it lacks intuitive appeal.

As the other Bayesian spatial autoregressive functions are quite similar, we leave it to the reader to examine the code for these functions. We turn attention to illustrating the use of these functions with some applied data sets.

4.4.1 Applied examples

First we present example 4.11 that generates SEM models for a set of λ parameters ranging from 0.1 to 0.9, based on the spatial weight matrix from the Columbus neighborhood crime data set. Both maximum likelihood and Gibbs estimates are produced by the program and a table is printed that compares the estimation results. The hyperparameter r was set to 30 in this example, which should produce estimates similar to the maximum likelihood results.

```
% ----- Example 4.11 Using the sem_g function
load wmat.dat;    % standardized 1st-order contiguity matrix
load anselin.dat; % load Anselin (1988) Columbus neighborhood crime data
y = anselin(:,1);  n = length(y);
x = [ones(n,1) anselin(:,2:3)];
W = wmat; IN = eye(n);
vnames = strvcat('crime','const','income','house value');
rvec = 0.1:.1:.9; b = ones(3,1);
nr = length(rvec); results = zeros(nr,6);
ndraw = 1100; nomit = 100; prior.rval = 30; bsave = zeros(nr,6);
for i=1:nr, rho = rvec(i);
u = (inv(IN-rho*W))*randn(n,1);
y =  x*b  + u;
% do maximum likelihood for comparison
resml = sem(y,x,W); prt(resml);
results(i,1) = resml.lam;
results(i,2) = resml.tstat(4,1);
bsave(i,1:3) = resml.beta';
% call Gibbs sampling function
result = sem_g(y,x,W,ndraw,nomit,prior); prt(result);
results(i,3) = mean(result.pdraw);
results(i,4) = results(i,3)/std(result.pdraw);
```

```
results(i,5) = result.time;
results(i,6) = result.accept;
bsave(i,4:6) = mean(result.bdraw);
end;
in.rnames = strvcat('True lam','0.1','0.2','0.3', ...
                    '0.4','0.5','0.6','0.7','0.8','0.9');
in.cnames = strvcat('ML lam','lam t','Gibbs lam','lam t', ...
                    'time','accept');
mprint(results,in);
in2.cnames = strvcat('b1 ML','b2 ML','b3 ML',...
                     'b1 Gibbs','b2 Gibbs','b3 Gibbs');
mprint(bsave,in2);
```

From the results presented in Table 4.1, we see that 1100 draws took around 22 seconds and the acceptance rate was between 44 and 49 percent. Note that the acceptance rate falls to around 44 percent for the λ value of 0.9, which is close to the upper limit of unity. This creates a slight increase in rejections of candidate values for λ that lie outside the feasible range for this parameter.

The Gibbs estimates are very similar to the maximum likelihood results, as they should be. In addition to presenting estimates for λ, we provide estimates for the parameters β in the problem and find that Gibbs and maximum likelihood estimates for these parameters are also very similar.

Table 4.1: SEM model comparative estimates

True λ	ML λ	$t-$statistic	Gibbs λ	$t-$statistic	time in seconds	acceptance rate
0.1	-0.1858	-0.8888	-0.1231	-0.5077	22.5557	0.4953
0.2	0.1928	1.0508	0.1898	0.9126	21.8660	0.4964
0.3	0.3538	2.1617	0.3383	2.0643	22.0213	0.4955
0.4	-0.0269	-0.1334	0.0020	0.0104	21.9332	0.4959
0.5	0.5436	4.0958	0.5060	3.7862	29.3263	0.4808
0.6	0.2343	1.3098	0.2335	1.1521	27.6904	0.4802
0.7	0.5987	4.9053	0.5888	4.8577	22.1155	0.4900
0.8	0.7815	9.7816	0.7621	9.2089	22.1736	0.4701
0.9	0.9064	21.2946	0.8764	16.3488	22.3031	0.4434
True λ	β_1 ML	β_2 ML	β_3 ML	β_1 Gibbs	β_2 Gibbs	β_3 Gibbs
0.1	0.7492	1.0102	0.9991	0.7691	1.0070	0.9998
0.2	1.3577	0.9529	1.0091	1.3410	0.9541	1.0091
0.3	1.6000	0.9909	0.9949	1.5478	0.9926	0.9953
0.4	1.4806	0.9565	1.0084	1.5306	0.9554	1.0078
0.5	0.7884	0.9863	0.9990	0.8304	0.9822	0.9993
0.6	0.8175	0.9959	1.0123	0.8767	0.9962	1.0115
0.7	1.6407	0.9889	0.9922	1.7074	0.9903	0.9914
0.8	1.7638	0.9761	1.0198	1.7549	0.9790	1.0198
0.9	0.5776	1.0480	0.9817	0.5472	1.0422	0.9824

As another example, we use the large data set from Pace and Barry containing 3,107 observations to estimate an SAR model with the function **sar_g**. We sort the data from west to east using longitude to facilitate our examination of the v_i estimates.

```
% ----- Example 4.12 Using sar_g with a large data set
load elect.dat;                    % load data on votes in 3,107 counties
lat = elect(:,5); lon = elect(:,6);
[lons li] = sort(lon);             % sort from west to east
lats = lat(li,1);                  % apply the sort index to others
elects = elect(li,:);              % apply the sort index to others
y =  elects(:,7)./elects(:,8);     % convert to per capita variables
x1 = log(elects(:,9)./elects(:,8)); % education
x2 = log(elects(:,10)./elects(:,8));% homeownership
x3 = log(elects(:,11)./elects(:,8));% income
n = length(y); x = [ones(n,1) x1 x2 x3];
clear x1; clear x2; clear x3;
clear elect;               % conserve on RAM memory
[junk W junk] = xy2cont(lons,lats);
vnames = strvcat('voters','const','educ','homeowners','income');
rmin = 0; rmax = 1;
ndraw = 2100; nomit=1100;
resml = sar(y,x,W,rmin,rmax);
prt(resml,vnames);
prior.rmin = rmin; prior.rmax = rmax; prior.rval = 4;
res = sar_g(y,x,W,ndraw,nomit,prior);
prt(res,vnames);
```

We might expect to see differences in this example between the maximum likelihood and Gibbs estimates depending on the amount of heteroscedasticity in the Pace and Barry data set. Both sets of estimates are reported below. As a test for convergence of the Gibbs sampler we produced two sets of estimates, one based on 1100 draws with the first 100 omitted and a second based on 2100 draws with the first 1100 omitted. As the results indicate, both sets of estimates were nearly identical.

```
Spatial autoregressive Model Estimates
Dependent Variable =       voters
R-squared       =    0.6546
Rbar-squared    =    0.6543
sigma^2         =    0.0040
Nobs, Nvars     =   3107,     4
log-likelihood  =        5107.4582
# of iterations =     10
min and max rho =    0.0000,   1.0000
***************************************************************
Variable        Coefficient      t-statistic    t-probability
const              0.686229        23.685701         0.000000
educ               0.129302        15.126176         0.000000
homeowners         0.212400        27.112444         0.000000
income            -0.074652        -8.334795         0.000000
rho                0.623363        42.450682         0.000000
% results based on 1100 draws with 100 omitted
Gibbs sampling spatial autoregressive model
Dependent Variable =       voters
R-squared       =    0.5621
sigma^2         =    0.0175
r-value         =      4
```

```
Nobs, Nvars     =   3107,     4
ndraws,nomit    =   1100,   100
acceptance rate =    0.9991
time in secs    =  843.1012
min and max rho =    0.0000,   1.0000
***************************************************************
      Posterior Estimates
Variable        Coefficient      t-statistic    t-probability
const              0.893489        15.435189         0.000000
educ               0.175233        10.146969         0.000000
homeowners         0.240468        11.324459         0.000000
income            -0.110930        -5.498762         0.000000
rho                0.559845        27.827889         0.000000
% results based on 2100 draws with 1100 omitted
Gibbs sampling spatial autoregressive model
Dependent Variable =       voters
R-squared       =    0.5622
sigma^2         =    0.0174
r-value         =      4
Nobs, Nvars     =   3107,     4
ndraws,nomit    =   2100,  1100
acceptance rate =    0.9915
time in secs    = 1402.3937
min and max rho =    0.0000,   1.0000
***************************************************************
      Posterior Estimates
Variable        Coefficient      t-statistic    t-probability
const              0.898655        15.021938         0.000000
educ               0.176437        10.009073         0.000000
homeowners         0.241422        11.122395         0.000000
income            -0.112253        -5.333194         0.000000
rho                0.558832        24.127430         0.000000
```

The Gibbs sampler required 843 seconds to generate 1100 draws and 1402 seconds for 2100 draws, which compares to around 150 seconds for maximum likelihood estimates. An important point regarding the timing comparison is that the Bayesian heteroscedastic model estimates more parameters than the maximum likelihood model, because it includes the variance scaling terms v_i. If one wished to estimate a heteroscedastic version of the maximum likelihood model, it would be possible to specify a relationship for the non-constant variance and produce maximum likelihood estimates using methods described in Anselin (1988). This would require additional specification and estimation work that would make the Bayesian approach look more competitive in terms of time required to produce estimates. Further, the Bayesian model does not require a restrictive specification for the heteroscedastic disturbance term.

We see a difference between the maximum likelihood and the heteroscedastic Bayesian estimates based on Gibbs sampling. This difference is explained by the presence of five v_i estimates greater than three in magnitude, indicating the presence of a handful of outliers. A plot of the mean from the v_i draws is shown in Figure 4.8.

It is interesting that only 5 outlying observations produced a fairly dramatic difference between the maximum likelihood and Bayesian parameter estimates. As we would expect, the robust Bayesian estimates lead to a lower $R-$squared

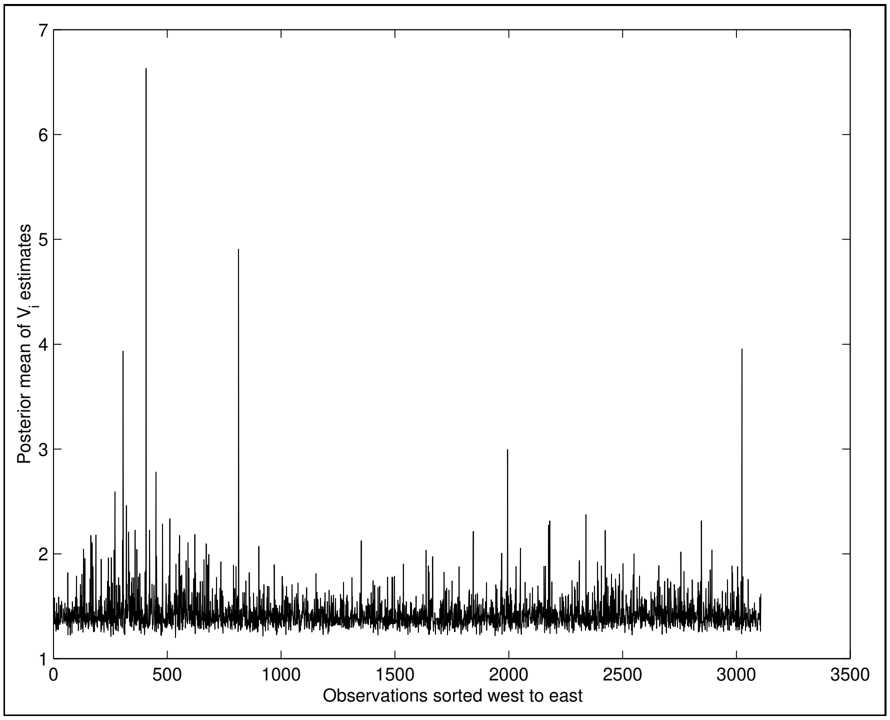

Figure 4.8: V_i estimates for Pace and Barry dataset

statistic, since robust estimates avoid any attempt to fit outlying sample data observations.

Given the evidence of outlying sample data points, it would be difficult to accept the maximum likelihood estimates that rely on normality. The robust Bayesian estimates should provide a better basis for statistical inference. For inferences regarding the spatial autoregressive parameter ρ in the SAR model, only 2 of the 1000 draws from the sample of 2100 with the first 1100 omitted were greater than or equal to the maximum likelihood estimate of 0.623. This can literally be interpreted as the posterior probability that $\rho \geq 0.623$, suggesting a 99.80 percent probability that $\rho < 0.623$, the maximum likelihood estimate.

4.5 An applied exercise

We carry out an applied exercise using the Pace and Barry data set for the 1980 presidential election. A series of alternative spatial autoregressive models are estimated using both maximum likelihood and Bayesian methods and we take up the issue of an appropriate specification for the model relationship.

Example 4.13 estimates SAR, SEM, and SAC models with four variants of the SAC models based on a first and second-order spatial weight matrix used in all four possible alternative configurations. One point to note is that these models are not comparable to the one from example 4.12 because the variable y was transformed to logs in example 4.13. This was done to create some variety, not for substantive reasons. In fact, the log transformation seems more appropriate given that the explanatory variables in the matrix X are in logged form.

Another reason for differences in results is that we rely on a first-order contiguity matrix here and we generate the matrix using the function **xy2cont** in example 4.12 which might produce slightly different results.

```
% ----- Example 4.13 Model specification
load elect.dat;             % load data on votes in 3,107 counties
y =  log(elect(:,7)./elect(:,8)); % convert to per capita variables
x1 = log(elect(:,9)./elect(:,8)); % education
x2 = log(elect(:,10)./elect(:,8));% homeownership
x3 = log(elect(:,11)./elect(:,8));% income
n = length(y); x = [ones(n,1) x1 x2 x3];
clear x1; clear x2; clear x3;
clear elect;                % conserve on RAM memory
load ford.dat; % 1st order contiguity matrix stored in sparse matrix form
ii = ford(:,1); jj = ford(:,2); ss = ford(:,3);
n = 3107;
clear ford; % clear ford matrix to save RAM memory
W1 = sparse(ii,jj,ss,n,n);
W2 = slag(W1,2);
clear ii; clear jj; clear ss; % conserve on RAM memory
vnames = strvcat('voters','const','educ','homeowners','income');
result1 = sar(y,x,W1); % do sar model based on W matrix
rmin1 = result1.rmin; rmax1 = result1.rmax;
result2 = sar(y,x,W2); % do sar model based on W2 matrix
rmin2 = result2.rmin; rmax2 = result2.rmax;
result3 = sem(y,x,W1,rmin1,rmax1); % do sem model based on W1 matrix
result4 = sem(y,x,W2,rmin2,rmax2); % do sem model based on W2 matrix
result5 = sac(y,x,W1,W2);          % do sac model based on W1,W2 matrix
result6 = sac(y,x,W2,W1);          % do sac model based on W2,W1 matrix
result7 = sac(y,x,W1,W1);          % do sac model based on W1,W1 matrix
result8 = sac(y,x,W2,W2);          % do sac model based on W2,W2 matrix
% do Bayesian models
ndraw = 1100; nomit = 100;
prior1.rmin = rmin1; prior1.rmax = rmax1; prior1.rval = 4;
bresult1 = sar_g(y,x,W1,ndraw,nomit,prior1);
prior2.rmin = rmin2; prior2.rmax = rmax2; prior2.rval = 4;
bresult2 = sar_g(y,x,W2,ndraw,nomit,prior2);
bresult3 = sem_g(y,x,W1,ndraw,nomit,prior1);
bresult4 = sem_g(y,x,W2,ndraw,nomit,prior2);
prior3.rmin = rmin1; prior3.rmax = rmax1;
prior3.lmin = rmin2; prior3.lmax = rmax2; prior3.rval = 4;
bresult5 = sac_g(y,x,W1,W2,ndraw,nomit,prior3);
prior4.rmin = rmin2; prior4.rmax = rmax2;
prior4.lmin = rmin1; prior4.lmax = rmax1; prior4.rval = 4;
bresult6 = sac_g(y,x,W2,W1,ndraw,nomit,prior4);
prior5.rmin = rmin1; prior5.rmax = rmax1;
prior5.lmin = rmin1; prior5.lmax = rmax1; prior5.rval = 4;
bresult7 = sac_g(y,x,W1,W1,ndraw,nomit,prior5);
```

```
prior6.rmin = rmin2; prior6.rmax = rmax2;
prior6.lmin = rmin2; prior6.lmax = rmax2; prior6.rval = 4;
bresult8 = sac_g(y,x,W2,W2,ndraw,nomit,prior6);
% print all results
prt(result1,vnames); prt(bresult1,vnames);
prt(result2,vnames); prt(bresult2,vnames);
prt(result3,vnames); prt(bresult3,vnames);
prt(result4,vnames); prt(bresult4,vnames);
prt(result5,vnames); prt(bresult5,vnames);
prt(result6,vnames); prt(bresult6,vnames);
prt(result7,vnames); prt(bresult7,vnames);
prt(result8,vnames); prt(bresult8,vnames);
```

A question we can pose is which model specification represents the most appropriate model? The alternative models can be viewed as nesting each other, making this question easier to deal with. For example, the SAC model nests both the SAR and SEM models. An insignificant λ coefficient estimate in the SAC model would point to an SAR model as more appropriate, whereas an insignificant ρ estimate suggests the SEM model would be more appropriate.

Another criterion we might use is rejection of negative spatial autocorrelation estimates for either ρ or λ, as these suggest that neighboring counties exhibit more dissimilar relationships than distant counties, a result counter to intuition.

A final specification issue is the choice between heteroscedastic and homoscedastic models. As we already saw in example 4.12, there is some evidence of a handful of outlying sample data observations, making us lean towards the Bayesian heteroscedastic models. It may be the case that the model specification we select as most appropriate does not exhibit different parameter estimates for the homoscedastic versus heteroscedastic models, making this an irrelevant issue.

We present a table of results for each spatial autoregressive model with a discussion of these results. The spatial autoregressive models based on W1 and W2 weight matrices are presented in Table 4.2.

Table 4.2: SAR model comparisons

Variable	sar($t-$stat)	sar_g($t-$stat)	sar($t-$stat)	sar_g($t-$stat)
	W1 matrix	W1 matrix	W2 matrix	W2 matrix
const	0.6490 (15.36)	0.8039 (12.66)	0.6240 (14.66)	0.7825 (11.75)
educ	0.2540 (16.11)	0.3425 (15.67)	0.2108 (12.88)	0.3048 (12.43)
homeown	0.4761 (32.15)	0.4471 (15.54)	0.4859 (32.75)	0.4487 (15.89)
income	-0.1173 (-7.03)	-0.1955 (-7.94)	-0.0916 (-5.41)	-0.1793 (-6.75)
rho	0.5288 (36.20)	0.5078 (30.12)	0.6190 (36.95)	0.5835 (27.54)
R^2	0.6356	0.5986	0.6321	0.5877
ln likel	3159.4467		3186.9297	

By virtue of the log transformation, we can interpret the coefficient estimates as elasticities. What we see is that the Bayesian estimates indicate a larger positive elasticity with respect to education and a larger negative elasticity for income than the maximum likelihood estimates. Given the large $t-$statistics, these differences are significant using conventional tests. This pattern will be repeated in the other model specifications that are presented below. The elasticity with respect to homeownership is fairly similar, and this pattern will be repeated in the other specifications as well.

Again, the large difference in these elasticities is somewhat surprising given that only a handful of outliers exist in the sample of 3,107 observations. Another pattern that arises here and in the SAC model (presented below) is that R^2 values for the Bayesian models are a bit lower than the maximum likelihood, suggesting some robustification is taking place.

With regard to the use of W1 or W2 as a weight matrix, the model based on W2 exhibits a slightly higher log likelihood function value, but a lower R^2. There is little to recommend one of these specifications over the other.

We turn attention to the spatial error model estimates presented in Table 4.3. Here again we see larger Bayesian estimates for the education and income elasticities for both the case of W1 and W2 spatial weight matrices. Again, the likelihood function is slightly higher for the model based on the second order weight matrix W2, and the R^2 is slightly higher for the model based on W1. These differences are small however, and it is difficult to choose between these alternative weight matrix specifications.

Table 4.3: SEM model comparisons

Variable	sem($t-$stat)	sem_g($t-$stat)	sem($t-$stat)	sem_g($t-$stat)
	W1 matrix	W1 matrix	W2 matrix	W2 matrix
const	0.5431 (8.76)	0.7007 (11.15)	0.3922 (6.34)	0.5447 (8.97)
educ	0.2933 (12.06)	0.3391 (13.69)	0.2300 (9.61)	0.2782 (11.93)
homeown	0.5714 (36.43)	0.5279 (29.39)	0.5686 (36.54)	0.5325 (28.28)
income	-0.1528 (-6.82)	-0.2324 (-10.22)	-0.1083 (-4.93)	-0.1830 (-8.51)
lambda	0.6505 (41.01)	0.7213 (49.84)	0.7714 (46.46)	0.8275 (60.84)
R^2	0.6606	0.6663	0.6539	0.6555
ln likel	3202.7211		3233.4214	

What we find in the results for the SAC model presented in Table 4.4 is that both W1 and W2 are significant. This may explain the results for the SAR and SEM models, where support for the significance of both W1 and W2 is provided by the sample data.

In these models we see a Bayesian elasticity estimate for education that is nearly double the maximum likelihood estimates. The negative income elasticity estimate from Bayesian estimation is more than double the maximum likelihood estimate. As in the SAR and SEM models, the elasticity for homeownership is very similar from the two estimation methods. There is also evidence of robustification in that the R^2 for the Bayesian estimates is lower than that from maximum likelihood.

The model based on W1,W2 exhibits a slightly higher likelihood function value than the model based on W2,W1. In addition, this model exhibits a higher likelihood function value than the SAR and SEM models examined above.

Table 4.4: SAC model comparisons

Variable	sac($t-$stat) W1,W2 matrix	sac_g($t-$stat) W1,W2 matrix	sac($t-$stat) W2,W1 matrix	sac_g($t-$stat) W2,W1 matrix
const	0.6835 (13.25)	0.9460 (13.21)	0.5857 (10.24)	0.8675 (13.24)
educ	0.2479 (12.44)	0.4166 (15.17)	0.2450 (11.48)	0.4166 (17.05)
homeown	0.5551 (35.37)	0.5155 (19.93)	0.5567 (35.77)	0.5381 (23.37)
income	-0.1171 (-5.85)	-0.2601 (-9.24)	-0.1128 (-5.50)	-0.2541 (-10.19)
rho	0.4150 (16.52)	0.4700 (15.96)	0.5907 (18.94)	0.6529 (23.60)
lambda	0.4600 (17.82)	0.3861 (11.61)	0.3108 (11.76)	0.2666 (10.13)
R^2	0.6656	0.6114	0.6619	0.6251
ln likel	3303.143		3290.8999	

The SAC model would appear to be the best specification. We also examine two more SAC models. The first is based on W1 for both the spatial autoregressive lag and the spatially correlated disturbances, and the second model relies on W2 for both spatial correlation terms. These results presented in Table 4.5 rule out either of these models based on our criterion that rejects models with negative spatial autoregressive coefficient estimates.

Table 4.5: Alternative SAC model comparisons

Variable	sac($t-$stat) W1,W1 matrix	sac_g($t-$stat) W1,W1 matrix	sac($t-$stat) W2,W2 matrix	sac_g($t-$stat) W2,W2 matrix
const	-0.0339 (-0.48)	0.6599 (8.82)	-0.1588 (-2.12)	0.6522 (10.04)
educ	0.1707 (7.16)	0.3321 (11.47)	0.1708 (7.61)	0.3451 (13.82)
homeown	0.5015 (33.31)	0.4063 (16.65)	0.5180 (34.76)	0.4343 (20.73)
income	-0.0917 (-4.33)	-0.2331 (-8.46)	-0.0816 (-3.94)	-0.2186 (-9.15)
rho	0.8681 (75.01)	0.8701 (40.71)	0.9340 (100.12)	0.9412 (98.89)
lambda	-0.4767 (-13.80)	-0.4665 (-7.53)	-0.6324 (-13.74)	-0.6014 (-13.84)
R^2	0.7251	0.6602	0.6968	0.5885
ln likel	3250.7121		3298.5278	

By way of conclusion, the evidence points to an SAC model based on W1 or W2 for the spatially lagged dependent variable and W1 or W2 for the spatial error term. Interestingly, we need not worry about which specification we use as the elasticity estimates for the variables in both specifications of the model are almost identical. There is however a large difference in the elasticity estimates produced by the Bayesian and maximum likelihood estimation methods.

The Bayesian elasticity estimates for education and income are roughly double the maximum likelihood estimates. The estimates for the variance scaling parameters indicate only a handful of outliers for this large data sample, but these outliers appear to make a difference in the inferences one would draw from robust versus non-robust estimates.

4.6 Chapter Summary

We have seen that the spatial autoregressive models can be extended to allow for Bayesian prior information as well as non-constant variances over space. These models require a Gibbs sampling estimation approach, making them more time intensive than maximum likelihood estimation methods. Nonetheless, the time required is quite reasonable, even for large sample problems. As in the case of maximum likelihood estimation, we rely on MATLAB sparse matrix algorithms for our Gibbs sampler. Estimation of these models takes around five to six times as long as maximum likelihood methods, but the estimates produced are robust with respect to non-constant variance of the disturbances over space and aberrant or outlying sample observations. Both of these situations are often encountered in applied work with spatial data sets. A fair comparison of the time required to produce maximum likelihood versus Bayesian estimates would require that we extend the maximum likelihood model to include a heteroscedastic disturbance relationship. This would involve specification and estimation not considered in the comparison times noted above.

There are several advantages to the Bayesian spatial autoregressive models. One advantage is that they provide a simple way to check the assumption of homogeneity in the disturbances made by maximum likelihood methods. Implemented with a diffuse prior for the parameters ρ, λ and β, but a prior belief in heteroscedasticity of the disturbances, the Bayesian spatial autoregressive models will produce estimates very similar to maximum likelihood in the absence of non-constant variance or aberrant observations. In the case of non-constant variance or outliers, the Bayesian estimates will diverge from maximum likelihood, indicating a violation of homoscedasticity. Because maximum likelihood estimates require normality in the disturbances, this check is quite useful.

A second advantage of the Bayesian models is that unlike the methods used to accommodate non-constant variance in maximum likelihood methods, (see Anselin, 1988), the practitioner need not specify a model for the changing variance over space. The Bayesian approach set forth here can automatically detect both non-constant variance as well as the presence of aberrant observations or outliers in the spatial sample.

A third advantage of the Bayesian models is the ability to introduce prior information regarding the parameters in the form of subjective non-sample information. This can be helpful in overcoming collinearity and other weak data problems that degrade the precision of the estimates. Practitioners may encounter cases where Gibbs sampling estimation can produce estimates when ill-conditioning or identification problems prevent maximization of the likelihood function with optimization methods. There are other situations where maximum likelihood computation of the information matrix or numerical hessian matrix used to obtain estimates of dispersion for the parameters may be plagued by ill-conditioning preventing calculation of estimates for dispersion.

A final advantage lies in the variance estimates for every point in space produced by the Bayesian model. This allows one to identify and examine observations that may represent outliers or neighborhood enclave effects. It can also be used to suggest changes in the model specification when systematic non-constant variance over space arises. For example, one might map the estimated v_i to study systematic drift in the disturbances of the model over space. If we observe higher disturbance variances in eastern regions of our spatial sample and lower disturbance variances in western regions, it might indicate an important explanatory variable is missing from the model.

Chapter 5

Limited dependent variable models

This chapter extends spatial autoregressive models to the case of limited dependent variables where we devise logit and probit variants of the spatial autoregressive models. Spatial autoregressive tobit models are also developed to deal with cases where sample observations are censored or truncated.

Logit and probit models arise when the dependent variable y in our spatial autoregressive model takes values 0, 1, 2, ..., representing counts of some event or a coding system for qualitative outcomes. For example, $y = 0$ might represent a coding scheme indicating a lack of highways in our sample of geographical regions, and $y = 1$ denotes the presence of a highway. Another example is where the values taken by y represent counts. We might have $y = 0, 1, 2, \ldots$, denoting the number of foreign direct investment projects in a given county where our sample of observations represent counties for a particular state.

Tobit models reflect situations where the dependent variable y contains observations that are unobserved because of censoring. We present an example where all house values greater than \$50,000 were assigned a constant magnitude of \$50,000, reflecting a censoring or truncation process that took place in reporting house values.

Despite the fact that spatial samples often involve limited dependent variables (see Bolduc, 1989, 1995, and Dubin, 1995), McMillen (1992) is the only literature regarding estimation of spatial autoregressive models for this type of sample data. He proposed an approach based on the EM algorithm for estimating heteroscedastic spatial autoregressive probit models. McMillen (1995) investigates the impact of heteroscedasticity that is often present in spatial models on probit estimation for ordinary regression models. This line of work was motivated by a feeling that large sample models could not be handled by the EM method in McMillen (1992). McMillen and McDonald (1998) propose a non-parametric locally linear probit method for GWR models that we will discuss in Chapter 6. This represents an ordinary regression model applied to a spatially weighted subset of the sample observations, not the case of spatial autoregressive estimation. I am unaware of any literature for the case of spatial autoregressive tobit models where the dependent variables are censored.

In this chapter we show how Bayesian methods can be used to produce estimates for spatial autoregressive logit, probit and tobit models by simply adding one additional conditional distribution to the Gibbs sampling methods we developed in Chapter 4. This provides a unified treatment of spatial autoregressive logit, probit and tobit models that is easy to implement.

5.1 Introduction

Spatial autoregressive modeling of limited dependent variable data samples would be interpreted in the framework of a probability model that attempts to describe the $Prob(\text{event } i \text{ occurs}) = F(W_1y, W_2u, X\text{: parameters})$. If the outcomes represent two possibilities, $y = 0, 1$, the model is said to be binary, whereas models with more than two outcomes are referred to as multinomial or polychotomous. We restrict attention to binary models.

Traditional spatial autoregressive models could be used to carry out a spatial autoregression using the binary response variable $y = 0, 1$, but two problems arise. To illustrate the nature of these problems, consider the SAR model. The first problem that arises is that the errors are by construction heteroscedastic. Because the actual y values take on either 0 or 1 values, the errors for the SAR model would take values of $\rho Wy + X\beta = -\rho Wy - X\beta$ when $y = 0$ and $\iota - \rho Wy - X\beta$, when $y = 1$. Note also that these heteroscedastic errors are a function of the parameter vector β and the scalar parameter ρ.

The second problem with using spatial autoregressive models in this setting is that the predicted values can take on values outside the (0,1) interval, which is problematic given a probability model framework. In this setting we would like to see:

$$lim_{\rho Wy+X\beta \rightarrow +\infty} Prob(y = 1) = 1 \tag{5.1}$$

$$lim_{\rho Wy+X\beta \rightarrow -\infty} Prob(y = 1) = 0 \tag{5.2}$$

Two distributions that have been traditionally used to produce an outcome that ensures predicted values between zero and one are the logisitic and normal distributions. For the case of a simple regression model, these result in the logit model shown in (5.3) and probit model shown in (5.4), where Φ denotes the cumulative normal probability function.

$$Prob(y = 1) = e^{X\beta}/(1 + e^{X\beta}) \tag{5.3}$$

$$Prob(y = 1) = \Phi(X\beta) \tag{5.4}$$

The logistic distribution is similar to the normal except in the tails where it is fatter, resembling a Student $t-$distribution. Green (1997) and others indicate that the logistic distribution resembles a $t-$distribution with seven degrees of freedom.

McMillen (1992) proposed methods for estimating SAR and SEM probit models containing spatial heteroscedasticity that rely on the EM algorithm. Bayesian methods for logit/probit and tobit variants of spatial autoregressive models that exhibit heteroscedasticity are developed in this chapter. The approach taken draws on work by Chib (1992) and Albert and Chib (1993) as well as the Bayesian estimation of spatial autoregressive models set forth in Chapter 4.

Accounting for heteroscedasticity in logit/probit and tobit models is important because estimates based on the assumption of homoscedasticity in the presence of heteroscedastic disturbances are inconsistent (see Green, 1997). The proposed Bayesian estimation methodology overcomes several drawbacks associated with McMillen's (1992) EM approach to estimating these models in the presence of heteroscedastic disturbances.

EM estimation methods rely on an iterative sequencing between the E-step that involves estimation and the M-step that solves a conditional maximization problem. The maximization problem is conditional on parameters determined in the E-step, making it easier to solve than the entire problem involving all of the parameters in the problem.

The Bayesian approach proposed here extends work of Chib (1992) for the tobit model and Albert and Chib (1993) for the probit model to the case of spatial autoregressive models. The basic idea exhibits a similarity to the EM algorithm proposed by McMillen (1992), where the censored or latent unobserved observations on the dependent variable y in the model are replaced by estimated values. Given estimates of the missing y values, the EM algorithm proceeds to estimate the other parameters in the model using methods applied to non-binary data samples. In other words, conditional on the estimated y-values, the estimation problem is reduced to a traditional estimation problem which can be solved using maximum likelihood methods.

There are some drawbacks to McMillen's EM estimator that we will overcome using the Bayesian approach set forth in this chapter. One drawback to McMillen's EM estimator is that the information matrix approach to determining measures of precision for the parameter estimates cannot be used. The likelihood function for the heteroscedastic probit model contains a number of integrals equal to the number of observations, so evaluating the likelihood function for these models is impossible. McMillen (1992) overcomes this problem using a non-linear weighted least-squares interpretation of the probit estimator conditional on the spatial lag parameters ρ in the SAR model and λ in the SEM model. This rules out estimates of dispersion for these important parameters. The use of a covariance matrix conditional on the spatial lag parameters produces biased, but consistent confidence intervals that may be too small.

Another problem with McMillen's approach is the need to specify a functional form for the non-constant variance over space. That is, one must specify a model for the noise vector ε such that, $[\text{var}(\varepsilon_i)]^{1/2} = g(Z_i)\gamma$, where g is a continuous, twice differentiable function and Z_i is a vector of explanatory variables for var(ε_i). This approach was illustrated by McMillen (1992) for a simple 2-variable model where both of the variables in squared form were used to form the Z_i vector, i.e., $g_i = exp(\gamma_1 X_{1i}^2 + \gamma_2 X_{2i}^2)$. In larger models a practitioner would need to devote considerable effort to testing the functional form and variables involved in the model for var(ε_i). Assuming success in finding a few candidate specifications, there is still the problem of inferences that may vary across alternative specifications for the non-constant variance.

We rely on Gibbs sampling to estimate the spatial logit/probit and tobit models. During sampling, we introduce a conditional distribution for the censored or latent observations conditional on all other parameters in the model. This distribution is used to produce a random draw for each censored value of y_i in the case of tobit, and for all y_i in the probit model. The conditional distribution for the latent variables takes the form of a normal distribution centered on the predicted value truncated at the right (or left) at the censoring point in the case of tobit, and truncated by 0 from the left and right in the case of probit, for $y_i = 1$ and $y_i = 0$ respectively.

An important difference between the EM approach and the Gibbs sampling approach set forth here is the proof outlined by Gelfand and Smith (1990) that Gibbs sampling from the sequence of complete conditional distributions for all parameters in the model produces a set of draws that converge in the limit to the true (joint) posterior distribution of the parameters. Because of this, we overcome the bias inherent in the EM algorithm's use of conditional distributions. Valid measures of dispersion for all parameters in the model can be constructed from the large sample of parameter draws produced by the Gibbs sampler.

The Gibbs sampling approach to estimating heteroscedastic spatial autoregressive models presented in Chapter 4 can be adapted to produce probit and tobit estimates by adding a single conditional distribution for the censored or latent observations. Intuitively, once we have a sample for the unobserved latent dependent variables, the problem reduces to the Bayesian heteroscedastic spatial autoregressive models presented in Chapter 4. The conditional distributions presented in Chapter 4 remain valid for all other parameters in the model.

Another important advantage of the method proposed here is that heteroscedasticity and outliers can be easily accommodated with the use of the methods outlined in Chapter 4. For the case of the probit model, an interesting interpretation can be given to the family of $t-$distributions that arise from the$\chi^2(r)$ prior introduced in Chapter 4. Recall that models involving binary data can rely on any continuous probability function as the probability rule linking fitted probabilities with the binary observations. Probit models arise from a normal probability rule and logit models from a logistic probability rule. When one introduces the latent variables z_i in the probit model to reflect unobserved values based on the binary dependent variables y_i, we have an underlying conditional spatial autoregression involving z and the model variables X, W_1, W_2, where X represents the explanatory variables and W_1, W_2 denotes the spatial weight matrices. The heteroscedastic spatial autoregressive model introduced in Chapter 4 can be viewed in the case of binary dependent variables as a probability rule based on a family of $t-$distributions that represent a mixture of the underlying normal distribution used in the probit regression. (It is well known that the normal distribution can be modeled as a mixture of $t-$distributions, see Albert and Chib, 1993).

The most popular choice of probability rule to relate fitted probabilities with binary data is the logit function corresponding to a logistic distribution for the cumulative density function. Albert and Chib (1993) show that the quantiles of the logistic distribution correspond to a t-distribution around 7 or 8 degrees of freedom. We also know that the normal probability density is similar to a $t-$distribution when the degrees of freedom are large. This allows us to view both the probit and logit models as special cases of the family of models introduced here using a chi-squared prior based on our hyperparameter r specifying alternative degrees of freedom to model spatial heterogeneity and outliers.

By using alternative values for the prior hyperparameter that we labeled r in Chapter 4, one can test the sensitivity of the fitted probabilities to alternative distributional choices for the model. For example, if we rely on a value of r near 7 or 8, the estimates resulting from the Bayesian version of the heteroscedastic probit model correspond to those one would achieve using a logit model. On the other hand, using a large degrees of freedom parameter, say $r = 30$ would lead to estimates that produce fitted probabilities based on the probit model choice of a normal probability rule. The implication of this is that the heteroscedastic spatial probit model we introduce represents a more general model than either probit or logit. The generality derives from the family of $t-$distributions associated with alternative values of the hyperparameter r in the model.

A final advantage of the method described here is that estimates of the non-constant variance for each point in space are provided, and the practitioner need not specify a functional form for the non-constant variance. Spatial outliers or aberrant observations as well as patterns of spatial heterogeneity will be identified in the normal course of estimation. This represents an improvement over the approach described by McMillen (1992), where a separate model for the non-constant variance needs to be specified.

5.2 The Gibbs sampler

For spatial autoregressive models with uncensored y observations where the error process is homoscedastic and outliers are absent, the computational intensity of the Gibbs sampler is a decided disadvantage over maximum likelihood methods. As demonstrated in Chapter 4, it takes three to five times as long to compute these estimates. For the case of probit and tobit models, the Gibbs sampler is very competitive to the EM algorithm presented in McMillen (1992), because numerous probit or tobit maximum likelihood problems need to be solved to implement the EM method.

Before turning attention to the heteroscedastic version of the Bayesian spatial autoregressive logit/probit and tobit models, consider the case of a homoscedastic spatial autoregressive tobit model where the observed y variable is censored. One can view this model in terms of a latent but unobservable variable z such that values of $z_i \leq$ limit, produce an observed variable $y_i =$ limit. It can also be the case that censoring occurs for values that exceed a limit value, when $z_i \geq$ limit produce an observed variable $y_i =$ limit.

Similarly, spatial autoregressive probit models can be associated with a latent variable $z_i < 0$ that produce an observed variable $y_i = 0$ and $z_i \geq 0$ resulting in $y_i = 1$. In both the tobit and probit models the posterior distribution of z conditional on all other parameters takes the form of truncated normal distribution (see Chib, 1992, Albert and Chib, 1993).

For the spatial tobit model, the conditional distribution of z_i given all other parameters is a truncated normal distribution constructed by truncating a $N[\tilde{y}_i, \sigma_{ti}^2]$ distribution from the right (or left) at the limit value. We denote the predicted value for z_i by $\tilde{y}_i$ which represents:

1. the ith row of $\tilde{y} = B^{-1}X\beta$ for the SAR model
2. the ith row of $\tilde{y} = X\beta$ for the SEM model
3. the ith row of $\tilde{y} = A^{-1}X\beta + A^{-1}B^{-1}u$ for the SAC model
4. $A = (I_n - \rho W_1)$, $B = (I_n - \lambda W_2)$

The variance of the prediction is $\sigma_{ti}^2 = \sigma_\varepsilon^2 \sum_j \omega_{ij}^2$, where ω_{ij} denotes the ijth element of $(I_n - \rho W)^{-1}\varepsilon$ for the SAR, SEM and SAC models.

The pdf of the latent variables z_i is then:

$$f(z_i|\rho, \beta, \sigma) = \begin{cases} [1 - \Phi(\tilde{y}_i/\sigma_{ti})]^{-1}\exp[-(z_i - \tilde{y}_i)^2/2\sigma_{it}], & \text{if } z_i \leq k \\ y_i & \text{if } z_i > k \end{cases} \tag{5.5}$$

Where we assume left truncation without loss of generality and k denotes the truncation limit.

Similarly, for the case of probit, the conditional distribution of z_i given all other parameters is:

$$f(z_i|\rho, \beta, \sigma) \sim \begin{cases} N(\tilde{y}_i, \sigma_{pi}^2), & \text{truncated at the left by 0 if } y_i = 1 \\ N(\tilde{y}_i, \sigma_{pi}^2), & \text{truncated at the right by 0 if } y_i = 0 \end{cases} \tag{5.6}$$

Where $\sigma_{pi}^2 = \sum_j \omega_{ij}^2$. In the case of maximum likelihood probit the model is unable to identify both β and σ_ε^2, and there is a need to scale the problem so σ_ε^2 equals unity. For our Gibbs sampler, we can produce an estimate of σ_ε^2 using the draws for the latent variable z_i. The predicted value $\tilde{y}_i$ takes the same form for the SAR, SEM and SAC models as described above for the case of tobit. We rely on the latent draws for z_i to calculate the predicted values.

The tobit expression (5.5) indicates that we rely on the actual observed y values for non-censored observations and use the sampled latent variables for the unobserved values of y. For the case of the probit model, we replace values of $y_i = 1$ with the sampled normals truncated at the left by 0 and values of $y_i = 0$ with sampled normals truncated at the right by 0.

Given these sampled continuous variables from the conditional distribution of z_i, we can implement the remaining steps of the Gibbs sampler described in Chapter 4 to determine draws from the conditional distributions for ρ, β and σ using the sampled z_i values in place of the censored or binary variables y_i.

5.3 Heteroscedastic models

The models described in this section can be expressed in the form shown in (5.5), where we relax the usual assumption of homogeneity for the disturbances used in SAR, SEM and SAC modeling. Given the discussion in Section 5.2, we can initially assume the existence of non-censored observations on the dependent variable y because these can be replaced with sampled values z as motivated in the previous section.

$$\begin{aligned} y &= \rho W_1 y + X\beta + u \qquad (5.7)\\ u &= \lambda W_2 u + \varepsilon \\ \varepsilon &\sim \mathrm{N}(0, \sigma^2 V), \qquad V = \mathrm{diag}(v_1, v_2, \ldots, v_n) \end{aligned}$$

Where $v_i, i = 1, \ldots, n$ represent a set of relative variance parameters to be estimated. We restrict the spatial lag parameters to the interval $1/\mu_{min} < \rho, \lambda < 1/\mu_{max}$.

It should be clear that we need only add one additional step to the Gibbs sampler developed in Chapter 4 for Bayesian heteroscedastic spatial autoregressive models. The additional step will provide truncated draws for the censored or limited dependent variables.

The above reasoning suggest the following Gibbs sampler. Begin with arbitrary values for the parameters $\sigma^0, \beta^0, \rho^0$ and v_i^0, which we designate with the superscript 0.

1. Calculate $p(\sigma|\rho^0, \beta^0, v_i^0)$, which we use along with a random $\chi^2(n)$ draw to determine σ^1.

2. Calculate $p(\beta|\rho^0, \sigma^1 v_i^0)$ using σ^1 from the previous step. Given the means and variance-covariance structure for β, we carry out a multivariate random draw based on this mean and variance to determine β^1.

3. Calculate $p(v_i|\rho^0, \sigma^1 \beta^1)$, which is based on an $n-$vector of random $\chi^2(r+1)$ draws to determine $v_i^1, i = 1, \ldots, n$.

4. Use metropolis within Gibbs sampling to determine ρ^1 as explained in Chapter 4, using the the values σ^1, β^1 and $v_i^1, i = 1, \ldots, n$ determined in the previous steps.

5. Sample the censored y_i observations from a truncated normal centered on the predictive mean and variance determined using $\rho^1, \sigma^1, \beta^1, v_i^1$ (as described in Section 5.2) for the probit and tobit models.

In the above development of the Gibbs sampler we assumed the hyperparameter r that determines the extent to which the disturbances take on a leptokurtic character was known. It is unlikely in practice that investigators would have knowledge regarding this parameter, so an issue that confronts us when attempting to implement the heteroscedastic model is setting the hyperparameter r. As already discussed in Chapter 4, using a small value near $r = 4$, produces estimates that will be robust in the presence of non-constant variance and outliers. For the probit model, use of small values of r between 2 and 7 will produce estimates close to those based on a logit probability rule. To examine the sensitivity of inferences to use of a logit versus probit probability rule, one can produce estimates based on a larger value of $r = 30$ for comparison.

Regarding the use of logit versus probit probability rules, Green (1997) states that the issue of which distributional form should be used on applied econometric problems is unresolved. He further indicates that inferences from either logit or probit models are often the same.

5.4 Implementing probit models

We have functions **sarp_g**, **semp_g**, **sacp_g** that carry out Gibbs sampling estimation of the heteroscedastic probit spatial autoregressive models. The function **sarp_g** which we use to discuss implementation of the probit models is shown below.

```
PURPOSE: Gibbs sampling spatial autoregressive Probit model
         y = p*Wy + Xb + e, e is N(0,sige*V)
         y is a 0,1 vector
         V = diag(v1,v2,...vn), r/vi = ID chi(r)/r, r = Gamma(m,k)
         B = N(c,T),  sige = gamma(nu,d0), p = diffuse prior
---------------------------------------------------
USAGE: results = sarp_g(y,x,W,ndraw,nomit,prior,start)
where: y = dependent variable vector (nobs x 1)
       x = independent variables matrix (nobs x nvar)
       W = 1st order contiguity matrix (standardized, row-sums = 1)
   ndraw = # of draws
   nomit = # of initial draws omitted for burn-in
   prior = a structure for:  B = N(c,T),  sige = gamma(nu,d0)
           prior.beta, prior means for beta,   c above (default 0)
           prior.bcov, prior beta covariance , T above (default 1e+12)
           prior.rval, r prior hyperparameter, default=4
           prior.m,    informative Gamma(m,k) prior on r
           prior.k,    (default: not used)
           prior.nu,   a prior parameter for sige
           prior.d0,   (default: diffuse prior for sige)
           prior.rmin = (optional) min rho used in sampling
           prior.rmax = (optional) max rho used in sampling
   start = (optional) structure containing starting values:
           defaults: beta=1,sige=1,rho=0.5, V= ones(n,1)
```

```
            start.b   = beta starting values (nvar x 1)
            start.p   = rho starting value   (scalar)
            start.sig = sige starting value  (scalar)
            start.V   = V starting values (n x 1)
---------------------------------------------------
 NOTE:  if the model contains a constant term,
        1st column of x-matrix must contain constant term
---------------------------------------------------
 RETURNS:  a structure:
          results.meth  = 'sarp_g'
          results.bdraw = bhat draws (ndraw-nomit x nvar)
          results.sdraw = sige draws (ndraw-nomit x 1)
          results.vmean = mean of vi draws (1 x nobs)
          results.ymean = mean of y draws (1 x nobs)
          results.rdraw = r draws (ndraw-nomit x 1) (if m,k input)
          results.pdraw = p draws    (ndraw-nomit x 1)
          results.pmean = b prior means, prior.beta from input
          results.pstd  = b prior std deviations sqrt(diag(T))
          results.r     = value of hyperparameter r (if input)
          results.r2mf  = McFadden R-squared
          results.rsqr  = Estrella R-squared
          results.nobs  = # of observations
          results.nvar  = # of variables in x-matrix
          results.zip   = # of zero y-values
          results.ndraw = # of draws
          results.nomit = # of initial draws omitted
          results.y     = actual observations (nobs x 1)
          results.yhat  = predicted values
          results.nu    = nu prior parameter
          results.d0    = d0 prior parameter
          results.time  = time taken for sampling
          results.accept= acceptance rate
          results.rmax  = 1/max eigenvalue of W (or rmax if input)
          results.rmin  = 1/min eigenvalue of W (or rmin if input)
--------------------------------------------------------------------
[n junk] = size(y); yin = y; results.y = y; [n1 k] = size(x);
[n3 n4] = size(W); rflag = 0;
if nargin == 7
    b0 = start.b; sige = start.sig; V = start.V; p0 = start.p;
elseif nargin == 6   % we supply starting values
b0 = ones(k,1); sige = 1; V = ones(n,1);  p0 = 0.5;
else, error('Wrong # of arguments to sarp_g');
end;
fields = fieldnames(prior); nf = length(fields);
mm = 0; rval = 4; nu = 0; d0 = 0; c = zeros(k,1); T = eye(k)*1e+12;
for i=1:nf
    if strcmp(fields{i},'rval'), rval = prior.rval;
    elseif strcmp(fields{i},'m'), mm = prior.m; kk = prior.k;
        rval = gamm_rnd(1,1,mm,kk);    % initial value for rval
    elseif strcmp(fields{i},'beta'), c = prior.beta;
    elseif strcmp(fields{i},'bcov'), T = prior.bcov;
    elseif strcmp(fields{i},'nu'),   nu = prior.nu;
    elseif strcmp(fields{i},'d0'),   d0 = prior.d0;
    elseif strcmp(fields{i},'rmin'), lmin = prior.rmin;
    lmax = prior.rmax; rflag = 1;
    end;
end;
```

```
Q = inv(chol(T)); QpQ = Q'*Q; Qpc = Q'*c;
cc=0.2; t0 = clock;
if rflag == 0, opt.tol = 1e-4; opt.disp = 0;
lambda = eigs(sparse(W),speye(n),2,'BE',opt);
lmin = 1/lambda(2);        lmax = 1/lambda(1);
end; results.rmax = lmax; results.rmin = lmin;
% compute a detval vector based on Pace and Barry's approach
rvec = lmin-0.005:.005:lmax+0.005; spparms('tight');
z = speye(n) - 0.1*sparse(W); p = colmmd(z);
niter = length(rvec); detval = zeros(niter,2);
for i=1:niter;
  rho = rvec(i); z = speye(n) - rho*sparse(W); [l,u] = lu(z(:,p));
  detval(i,1) = sum(log(abs(diag(u)))); detval(i,2) = rho;
end;
rho = p0; IN = eye(n); in = ones(n,1);
index = zeros(n,1); % find an index of censored values
for i=1:n;
if y(i,1) == 0, index(i,1) = 1.0; end;
end;
nzip = sum(index); B = speye(n) - rho*sparse(W);
bsave = zeros(ndraw-nomit ,k); % storage for samples values
if mm~= 0, rsave = zeros(ndraw-nomit ,1); end;
vmean = zeros(n,1); ymean = zeros(n,1);
yhat = zeros(n,1); ssave = zeros(ndraw-nomit ,1);
psave = zeros(ndraw-nomit,1); rtmp = zeros(nomit,1);
cntr = 0; iter = 1;
% scale x for the conditional distribution function evaluation
if (ones(n,1) == x(:,1))
 xd = x(:,2:k) - matmul(mean(x(:,2:k)),ones(n,k-1)); cflag = 1;
else, xd = x - matmul(mean(x),ones(n,k)); cflag = 0;
end;
while (iter <= ndraw); % start sampling;
         % update beta
         ys = y.*sqrt(V);  xs = matmul(x,sqrt(V));
         yd = y - mean(y); ysd = yd.*sqrt(V);
         xsd = matmul(xd,sqrt(V));
         xpxi = inv(xs'*xs + sige*QpQ);
         xpy = (xs'*B*ys + sige*Qpc);
         bhat = xpxi*xpy;
         bhat = norm_rnd(sige*xpxi) + bhat;
         % update sige
         nu1 = n + nu;  e = B*ys - xs*bhat;
         d1 = d0 + e'*e; chi = chis_rnd(1,nu1);
         t2 = chi/d1;    sige = 1/t2;
         % update vi
         e = B*y - x*bhat; chiv = chis_rnd(n,rval+1);
         vi = ((e.*e./sige) + in*rval)./chiv; V = in./vi;
         % update rval
         if mm ~= 0, rval = gamm_rnd(1,1,mm,kk); end;
         % metropolis step to get rho update
          bt = bhat(cflag+1:length(bhat),1);
          rhox = c_sar(rho,ysd,xsd,bt,sige,W,detval);
          accept = 0; rho2 = rho + cc*randn(1,1);
          while accept == 0
           if ((rho2 > lmin) & (rho2 < lmax)), accept = 1;
           else, rho2 = rho + cc*randn(1,1);
           cntr = cntr+1; % counts acceptance rate
```

```
           end;
          end;
          rhoy = c_sar(rho2,ysd,xsd,bt,sige,W,detval);
          ru = unif_rnd(1,0,1);
          if ((rhoy - rhox) > exp(1)), p = 1;
          else, ratio = exp(rhoy-rhox); p = min(1,ratio);
          end;
              if (ru < p), rho = rho2; end;
          rtmp(iter,1) = rho;
          % update 0,1 y-values
          B = speye(n) - rho*sparse(W);
          yh = x*bhat + rho*sparse(W)*y; BIB = inv(B'*B);
          for i=1:n;
            aa = yh(i,1); sigi = BIB(i,i);
            if yin(i,1) == 0;
             y(i,1) = aa + sqrt(sigi)*nmrt_rnd(-aa/sqrt(sigi));
            elseif yin(i,1) == 1
             y(i,1) = aa + sqrt(sigi)*nmlt_rnd(-aa/sqrt(sigi));
            end;
          end;
    if iter > nomit % if we are past burn-in, save the draws
    bsave(iter-nomit,1:k) = bhat'; ssave(iter-nomit,1) = sige;
    psave(iter-nomit,1) = rho;     vmean = vmean + vi;
    ymean = ymean + y;
    yhat = yhat + norm_cdf(x*bhat + rho*sparse(W)*y);
    if mm~= 0, rsave(iter-nomit,1) = rval; end;
    end;
    if iter == nomit, cc = 2*std(rtmp(1:nomit,1)); end;
    iter = iter + 1;
end; % end of sampling loop
gtime = etime(clock,t0);    vmean = vmean/(ndraw-nomit);
ymean = ymean/(ndraw-nomit); yhat = yhat/(ndraw-nomit);
bmean = mean(bsave);        rmean = mean(psave);
bout = [bmean'
        rmean];
tmp = find(yin ==1); P = length(tmp); cnt0 = n-P; cnt1 = P;
P = P/n; % proportion of 1's
like0 = n*(P*log(P) + (1-P)*log(1-P));    % restricted likelihood
Wy = sparse(W)*yin;
like1 = pr_like(bout,yin,[x Wy]);         % unrestricted Likelihood
r2mf = 1-(abs(like1)/abs(like0));         % McFadden pseudo-R2
term0 = (2/n)*like0; term1 = 1/(abs(like1)/abs(like0))^term0;
rsqr = 1-term1;                           % Estrella R2
results.accept = 1 - cntr/(iter+cntr); results.r2mf = r2mf;
results.rsqr = rsqr;   results.meth  = 'sarp_g';
results.bdraw = bsave; results.pdraw = psave;
results.sdraw = ssave; results.vmean = vmean;
results.yhat  = yhat;  results.ymean = ymean;
results.pmean = c;     results.pstd  = sqrt(diag(T));
results.nobs  = n;     results.nvar  = k;
results.ndraw = ndraw; results.nomit = nomit;
results.time = gtime;  results.nu = nu;
results.d0 = d0;
if mm~= 0, results.rdraw = rsave; results.m = mm; results.k = kk;
else, results.r = rval; results.rdraw = 0;
end;
results.zip = nzip;
```

This function is similar to the **sar_g** function from Chapter 4, with the added code fragment shown below to carry out draws for the latent variable z_i. These draws are made using a right-truncated normal distribution function **nmrt_rnd** and left-truncated normal distribution function **nmlt_rnd** from the *Econometrics Toolbox*.

```
% update 0,1 y-values
B = speye(n) - rho*sparse(W);
yh = x*bhat + rho*sparse(W)*y; BIB = inv(B'*B);
for i=1:n;
 aa = yh(i,1); sigi = BIB(i,i);
  if yin(i,1) == 0;
   y(i,1) = aa + sqrt(sigi)*nmrt_rnd(-aa/sqrt(sigi));
  elseif yin(i,1) == 1
   y(i,1) = aa + sqrt(sigi)*nmlt_rnd(-aa/sqrt(sigi));
  end;
end;
```

Another difference relates to the measures of fit calculated. These are $R-$squared measures proposed by McFadden (1984) and a more recent proposal by Estrella (1998) for use with limited dependent variable models. Computing these measures of fit requires that we evaluate the likelihood function, so we use the means of the sampled parameter draws.

We produce a predicted value that will lie between 0 and 1 using the normal probability density function, **norm_pdf** from the *Econometrics Toolbox*. We also return the mean of the draws for the latent variable z_i as these might be useful for inference, especially in tobit models where interest often centers on the truncated or censored sample values. The code for this is shown in the lines below:

```
ymean = ymean + y;
yhat = yhat + norm_cdf(x*bhat + rho*sparse(W)*y);
```

5.5 Comparing EM and Bayesian probit models

Following an example provided by McMillen (1992) for his EM algorithm approach to estimating SAR and SEM probit models, we employ the data set from Anselin (1988) on crime in Columbus, Ohio. McMillen censored the dependent variable on crime such that $y_i = 1$ for values of crime greater than 40 and $y_i = 0$ for values of crime less than or equal to 40. The explanatory variables in the model are neighborhood housing values and neighborhood income. Example 5.1 carried out the same data transformation and estimates SAR logit and probit models.

Logit model estimates are generated by setting $r = 7$ and probit estimates are those based on $r = 40$. For comparison we also present results from **sar_g** that ignore the limited dependent variable nature of the y variable and maximum likelihood estimates produced by the **sar** function.

```
% ----- Example 5.1 Gibbs sampling probit models
```

```
load anselin.data;
y = anselin(:,1); [n junk] = size(y);
x = [ones(n,1) anselin(:,2:3)];
vnames = strvcat('crime','constant','income','hvalue');
load Wmat.data; W = Wmat;
yc = zeros(n,1);
% now convert the data to 0,1 values
for i=1:n
  if y(i,1) > 40.0, yc(i,1) = 1; end;
end;
ndraw = 1100; nomit = 100;
prior.rval = 7; % logit estimates
prt(sar(yc,x,W),vnames);
result = sar_g(yc,x,W,ndraw,nomit,prior);
prt(result,vnames);
plt(result,vnames);
pause;
result2 = sarp_g(yc,x,W,ndraw,nomit,prior);
prt(result2,vnames);
plt(result2,vnames);
prior.rval = 40; % probit estimates
result3 = sarp_g(yc,x,W,ndraw,nomit,prior);
prt(result3,vnames);
plt(result3,vnames);
```

The printed results are shown below and the graphical results provided by the **plt** function are shown in Figure 5.1 for the case of the logit model.

```
Spatial autoregressive Model Estimates
Dependent Variable =         crime
R-squared       =    0.5216
Rbar-squared    =    0.5008
sigma^2         =    0.1136
Nobs, Nvars     =     49,     3
log-likelihood  =       -1.1890467
# of iterations =     12
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
constant         0.679810         2.930130         0.005259
income          -0.019912        -1.779421         0.081778
hvalue          -0.005525        -1.804313         0.077732
rho              0.539201         2.193862         0.033336

Gibbs sampling spatial autoregressive model
Dependent Variable =         crime
R-squared       =    0.5066
sigma^2         =    0.1279
r-value         =      7
Nobs, Nvars     =     49,     3
ndraws,nomit    =   1100,   100
acceptance rate =    0.9718
time in secs    =   25.5119
min and max rho =   -1.5362,   1.0000
***************************************************************
      Posterior Estimates
Variable      Coefficient      t-statistic    t-probability
```

```
constant         0.815839         3.854791         0.000358
income          -0.026239        -2.011421         0.050161
hvalue          -0.005526        -1.586039         0.119582
rho              0.480589         3.495093         0.001061

Gibbs sampling spatial autoregressive Probit model
Dependent Variable =         crime
McFadden R^2    =    0.4048
Estrella R^2    =    0.4999
sigma^2         =    1.3547
r-value         =      7
Nobs, Nvars     =     49,     3
# 0, 1 y-values =     30,    19
ndraws,nomit    =   1100,   100
acceptance rate =    0.9982
time in secs    =   78.4667
min and max rho =   -1.5362,   1.0000
***************************************************************
      Posterior Estimates
Variable      Coefficient      t-statistic    t-probability
constant         3.350147         2.899936         0.005705
income          -0.177615        -2.062167         0.044866
hvalue          -0.035168        -1.515838         0.136403
rho              0.268768         1.545313         0.129123

Gibbs sampling spatial autoregressive Probit model
Dependent Variable =         crime
McFadden R^2    =    0.4034
Estrella R^2    =    0.4983
sigma^2         =    1.3484
r-value         =     40
Nobs, Nvars     =     49,     3
# 0, 1 y-values =     30,    19
ndraws,nomit    =   1100,   100
acceptance rate =    0.9822
time in secs    =   79.0866
min and max rho =   -1.5362,   1.0000
***************************************************************
      Posterior Estimates
Variable      Coefficient      t-statistic    t-probability
constant         3.689735         3.104312         0.003259
income          -0.196563        -2.362067         0.022461
hvalue          -0.037356        -1.658856         0.103949
rho              0.352575         2.298016         0.026158
```

There is a remarkable difference between the maximum likelihood estimates that ignore the limited dependent nature of the variable y and the logit/probit estimates, which we would expect. The logit and probit estimates for the β coefficients are relatively similar, but the estimate for ρ from these two models differ. The logit model produces an estimate of ρ that is not significantly different from zero whereas the probit model estimate is significant.

A final point with respect to interpreting the estimates is that the marginal impacts of the variables on the fitted probabilities is usually the inference aim of these models. The estimates would need to be converted according to the probability rule implied by the alternative underlying $t-$distributions associated

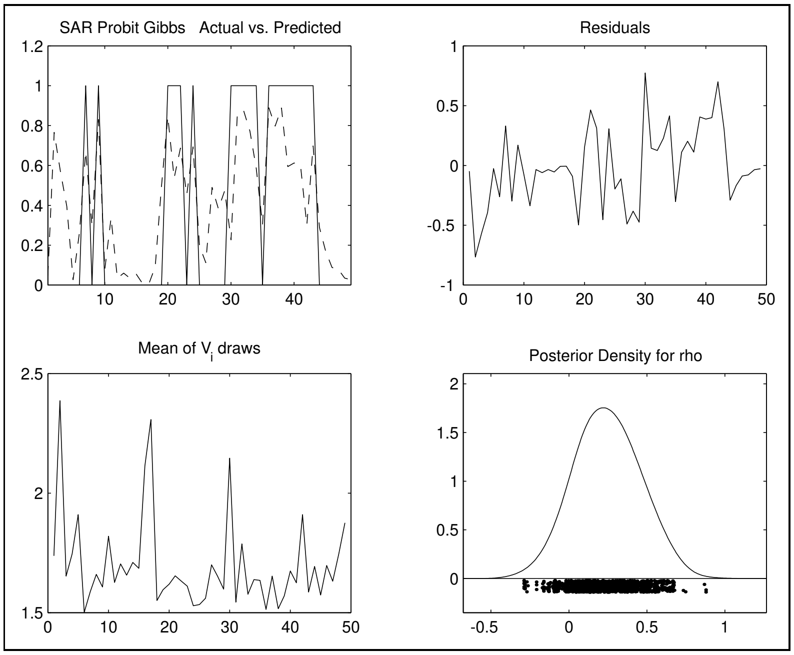

Figure 5.1: Results of plt() function for SAR logit

with the r value employed. These marginal impacts would be very similar (as in the case of non-spatial maximum likelihood logit versus probit marginal impacts) despite the apparent differences in the coefficients. For the purpose of computing marginal impacts, one needs to evaluate the posterior density of p_k, which we denote $\hat{\pi}(p_k)$ for k ranging over all sample observations. (It is conventional to assess marginal impacts across all observations in the model and then average these.) For the heteroscedastic model p_k will take the form: $\Phi(\tilde{y}_k) = \Phi(\rho W_k v_k^{(-1/2)} y + v_k^{(-1/2)} x_k' \beta)$. To find the posterior density estimate of $\hat{\pi}(p_k)$ we would employ the draws for ρ^i, v_k^i, β^i in normal densities, denoted $\mathrm{N}(\mu, \sigma)$ as indicated in (5.8)

$$
\begin{aligned}
\hat{\pi}(p_k) &= (1/m) \sum_{i=1}^{m} \mathrm{N}[\tilde{y}_k^i, (1/v_k^i) x_k' (X' V_i^{-1} X)^{-1} x_k] / \mathrm{N}[0,1] \qquad (5.8) \\
\tilde{y}_k^i &= \rho^i W_k (1/\sqrt{v_k^i}) y + (1/\sqrt{v_k^i}) x_k' \beta^i \\
V_i^{-1} &= \mathrm{diag}(1/v_j^i), j = 1, \ldots, n
\end{aligned}
$$

Table 5.1 shows a comparison of McMillen's EM algorithm estimates and those from Gibbs sampling. The Gibbs estimates are based on 2,500 draws with the first 100 discarded for startup or burn-in. Gibbs SAR and SEM estimates are reported for both $r = 4$ which should be close to a logit model and $r = 40$ which would represent the probit case. Results for two values of r were reported because the inferences are different from these two models.

Table 5.1: EM versus Gibbs estimates

	EM SAR	Gibbs SAR $r = 4$	Gibbs SAR $r = 40$	EM SEM	Gibbs SEM $r = 4$	Gibbs $r = 40$
CONSTANT	2.587	3.237	3.548	2.227	4.842	3.293
$t-$value	2.912	2.708	3.294	3.115	2.195	2.672
INCOME	-0.128	-0.176	-0.181	-0.123	-0.284	-0.186
$t-$value	-2.137	-2.124	-2.598	-2.422	-1.898	-2.075
HOUSING	-0.029	-0.032	-0.038	-0.025	-0.046	-0.035
$t-$value	-1.617	-1.264	-1.777	-1.586	-1.560	-1.677
ρ	0.429	0.245	0.359	0.279	0.105	0.185
$t-$value	—	1.319	2.128	—	0.373	0.621

The greatest difference between the EM and Gibbs estimates resides in the estimates for ρ in the SAR model and λ in the SEM model.

One reason why the EM and Gibbs estimates may be different is that McMillen's approach requires a model for the non-constant variance which was specified as: $v_i = 0.0007\text{INCOME}^2 + 0.0004\text{HOUSING}^2$. This is quite different from the approach taken in the Bayesian model that relies on v_i estimates from Gibbs sampling.

5.6 Implementing tobit models

We have functions **sart_g**, **semt_g**, **sact_g** for heteroscedastic tobit spatial autoregressive models. Documentation for the function **sart_g** which we use to discuss implementation of the tobit models is shown below.

Tobit models can involve either left or right truncation. That is, censored observations can be y values that fall below a limiting value or censoring can take place above a limit value. The functions allow the user to specify the type of censoring and a limit value. This defaults to the typical case of left censoring at zero.

```
PURPOSE: Gibbs sampling spatial autoregressive Tobit model
         y = p*Wy + Xb + e, e is N(0,sige*V)
         y is a censored vector (assumed to be at zero)
         V = diag(v1,v2,...vn), r/vi = ID chi(r)/r, r = Gamma(m,k)
         B = N(c,T),  sige = gamma(nu,d0), p = diffuse prior
---------------------------------------------------
USAGE: results = sart_g(y,x,W,prior,ndraw,nomit,start)
where: y = dependent variable vector (nobs x 1)
       x = independent variables matrix (nobs x nvar)
       W = 1st order contiguity matrix (standardized, row-sums = 1)
   ndraw = # of draws
   nomit = # of initial draws omitted for burn-in
```

```
    prior = a structure for:  B = N(c,T),  sige = gamma(nu,d0)
            prior.beta, prior means for beta,   c above (default 0)
            prior.bcov, prior beta covariance , T above (default 1e+12)
            prior.rval, r prior hyperparameter, default=4
            prior.m,    informative Gamma(m,k) prior on r
            prior.k,    (default: not used)
            prior.nu,    a prior parameter for sige
            prior.d0,    (default: diffuse prior for sige)
            prior.trunc = 'left' or 'right' censoring (default = left)
            prior.limit = value for censoring (default = 0)
    start = (optional) structure containing starting values:
            defaults: beta=1,sige=1,rho=0.5, V= ones(n,1)
            start.b   = beta starting values (nvar x 1)
            start.p   = rho starting value   (scalar)
            start.sig = sige starting value  (scalar)
            start.V   = V starting values (n x 1)
---------------------------------------------------
 NOTE:  1st column of x-matrix must contain iota vector (constant term)
---------------------------------------------------
 RETURNS:  a structure:
          results.meth  = 'sart_g'
          results.bdraw = bhat draws (ndraw-nomit x nvar)
          results.sdraw = sige draws (ndraw-nomit x 1)
          results.vmean = mean of vi draws (1 x nobs)
          results.rdraw = sige draws (ndraw-nomit x 1)
          results.pdraw = p draws    (ndraw-nomit x 1)
          results.ymean = mean of y draws (1 x nobs)
          results.pmean = b prior means, prior.beta from input
          results.pstd  = b prior std deviations sqrt(diag(T))
          results.r     = value of hyperparameter r (if input)
          results.nobs  = # of observations
          results.nvar  = # of variables in x-matrix
          results.nobsc = # of censored y-values
          results.ndraw = # of draws
          results.nomit = # of initial draws omitted
          results.y     = actual observations (nobs x 1)
          results.yhat  = predicted values
          results.nu    = nu prior parameter
          results.d0    = d0 prior parameter
          results.time  = time taken for sampling
          results.accept= acceptance rate
          results.rmax = 1/max eigenvalue of W (or rmax if input)
          results.rmin = 1/min eigenvalue of W (or rmin if input)
---------------------------------------------------
```

The major difference between the tobit and probit models resides in sampling for the latent variable z_i, with the code for this shown below:

```
% update censored y-values
yh = y - e;
for i=1:n;
 if tflag == 0 % left censoring
  if (yin(i,1) <= vflag)
       aa = yh(i,1);
       y(i,1) = aa + sqrt(sige)*nmrt_rnd(-aa/sqrt(sige));
  end;
 else % right censoring
```

```
      if (yin(i,1) >= vflag)
           aa = yh(i,1);
           y(i,1) = aa + sqrt(sige)*nmlt_rnd(-aa/sqrt(sige));
      end;
     end;
    end;
```

For the case of left censoring we draw from a right-truncated normal distribution using the **nmrt_rnd** function and for right censoring we rely on **nmlt_rnd** to produce a draw from a left-truncated normal. The variable 'vflag' is nothing more than the user-supplied truncation limit value or the default value of zero. Note that we only sample censored observations and rely on the actual y values for the case of uncensored observations.

A key to understanding how the Gibbs sampler works on these problems is the generation of predicted values for the censored observations. These values may also be useful for purposes of inference regarding the censored observations. The tobit spatial autoregressive functions return a structure variable field 'results.ymean' that represents the mean of the sampled values for the censored observations as well as the actual values for uncensored observations.

To illustrate, example 5.2 demonstrates using data generated for an SAR model based on the contiguity matrix from Anselin's neighborhood crime data. We censor y values less than zero. Because we know the value of the censored observations in this example, we can see how accurate the mean of the draws for these observations is. The mean of the posterior distribution for the censored observations would be represented by the mean of these draws. One can also use the standard deviation of these draws for purposes of inference.

```
% ----- Example 5.2 SAR Tobit Model
load anselin.dat;
xt = [anselin(:,2:3)]; n = length(xt);
% center and scale the data so our y-values
% are evenly distributed around zero, the censoring point
xs = studentize(xt);
x = [ones(n,1) xs]; [n k] = size(x);
load wmat.dat; W = wmat;
sige = 5.0; evec = randn(n,1)*sqrt(sige);
rho = 0.65; beta = ones(k,1);
B = eye(n) - rho*W; BI = inv(B);
y = BI*x*beta + BI*evec; % generate SAR model
yc = y;
% now censor negative values
for i=1:n
 if y(i,1) <= 0, yc(i,1) = 0; end;
end;
Vnames = strvcat('crime','constant','income','hvalue');
ndraw = 1100; nomit = 100;
prior.rval = 4;
prior.limit = 0; prior.trunc = 'left';
result = sart_g(yc,x,W,ndraw,nomit,prior);
prt(result,Vnames);
tt=1:n;
plot(tt,y,'-k',tt,result.ymean,'--k');
%title('actual vs mean of draws for latent variable');
```

```
xlabel('Observations');
ylabel('actual vs posterior mean of latent draws');
```

Figure 5.2 shows a plot of the mean of the simulated values for the censored observations against the actual y variables. Ordinarily, we wouldn't know the values of the censored y values, but in this case the generated data set provides this information.

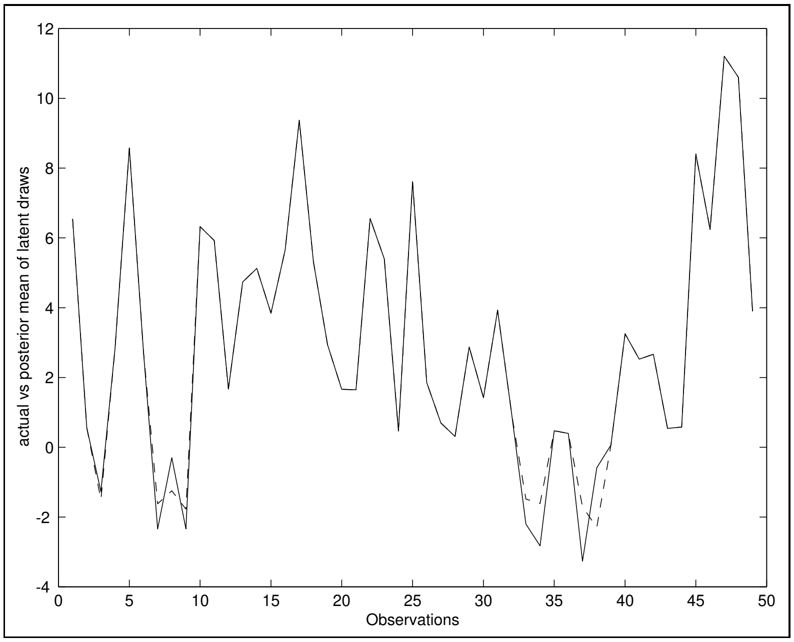

Figure 5.2: Actual vs. simulated censored y-values

The printed results for an SAR tobit model based on the censored data as well as a set of SAR maximum likelihood estimates based on the actual data are presented below.

```
Spatial autoregressive Model Estimates
Dependent Variable =         crime
R-squared       =    0.6018
Rbar-squared    =    0.5845
sigma^2         =    4.8963
Nobs, Nvars     =     49,     3
log-likelihood  =       -93.625028
# of iterations =     11
min and max rho =   -1.5362,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
constant         1.279824         1.996443         0.051824
income           0.599044         1.496695         0.141303
hvalue           1.146929         3.066180         0.003624
```

```
rho              0.564742        2.988512        0.004487

Gibbs sampling spatial autoregressive Tobit model
Dependent Variable =         crime
R-squared          =    0.5546
Imputed R-squared  =    0.5860
sigma^2            =    4.8531
r-value            =      4
Nobs, Nvars        =     49,     3
# censored values  =      8
ndraws,nomit       =   1100,   100
acceptance rate    =    0.9683
time in secs       =   21.9479
min and max rho    =   -1.5362,   1.0000
***************************************************************
      Posterior Estimates
Variable      Coefficient      t-statistic    t-probability
constant         1.491787         4.158658         0.000138
income           0.707966         1.669128         0.101885
hvalue           1.246787         2.858284         0.006379
rho              0.481235         3.282253         0.001971
```

We see that these two sets of estimates would lead to very similar inferences, so the tobit model is correctly imputing the censored observations.

5.7 An applied example

In a well-known paper, Harrison and Rubinfeld (1978) used a housing data set for the Boston SMSA with 506 observations (one observation per census tract) containing 14 variables. We will use this spatial data set to illustrate specification and testing for spatial autoregressive models. Our dependent variable will be median housing prices for each of the 506 census tracts. The explanatory variables in the data set are shown in Table 5.2.

Table 5.2: Variables in the Boston data set

CRIM	per capita crime rate by town
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	proportion of non-retail business acres per town
CHAS	Charles River dummy (= 1 if tract bounds river; 0 otherwise)
NOX	nitric oxides concentration (parts per 10 million)
RM	average number of rooms per dwelling
AGE	proportion of owner-occupied units built prior to 1940
DIS	weighted distances to five Boston employment centres
RAD	index of accessibility to radial highways
TAX	full-value property-tax rate per \$10,000
PTRATIO	pupil-teacher ratio by town
B	$1000(Bk-0.63)^2$ where Bk is the proportion of blacks by town
LSTAT	percent lower status of the population
MEDV	Median value of owner-occupied homes in \$1000's

Belsley, Kuh, and Welsch (1980) used the data to examine the effects of robust estimation and published the observations in an appendix on pages 244-261. It should be noted that their published data was transformed in various ways, so the data in their appendix does not match our data which is in the raw untransformed format. Pace (1993), Gilley and Pace (1996), and Pace and Gilley (1997) have used this data set with spatial econometric models and longitude-latitude coordinates for the census tracts have been added to the dataset. Pace and Gilley (1997) point out that this data set included 16 censored observations for housing values greater than \$50,000. For these observations, values were set to \$50,000. This enables us to compare tobit spatial autoregressive estimates to maximum likelihood and Bayesian heteroscedastic estimates that ignore the sample truncation.

Our regression model will simply relate the median house values to all of the explanatory variables, simplifying our specification task. We will focus on alternative spatial autoregressive model specifications.

The first task involves scaling and standardization of the data set. Belsley, Kuh and Welsch (1980) used this data set to illustrate numerically ill-conditioned data that contained outliers and influential observations. Poor scaling will adversely impact our numerical hessian approach to determining the variance-covariance structure for maximum likelihood spatial autoregressive parameter estimates. Intuitively, the **hessian** function attempts to compute a numerical derivative by perturbing each parameter in turn and examining the impact on the likelihood function. If the parameters vary widely in magnitude because the data is poorly scaled, this task will be more difficult and we may calculate negative variances.

Example 5.3 demonstrates the nature of these scaling problems, carrying out a least-squares regression. We see that the coefficient estimates vary widely in magnitude, so we scale the variables in the model using a function **studentize** from the *Econometric Toolbox* that subtracts the means and divides by the standard deviations. Another least-squares regression is then carried out to illustrate the impact of scaling on the model coefficients.

```
% ----- Example 5.3 Least-squares on the Boston dataset
load boston.raw; % Harrison-Rubinfeld data
[n k] = size(boston);y = boston(:,k);  % median house values
x = [ones(n,1) boston(:,1:k-1)];       % other variables
vnames = strvcat('hprice','constant','crime','zoning','industry', ...
         'charlesr','noxsq','rooms2','houseage','distance', ...
         'access','taxrate','pupil/teacher','blackpop','lowclass');
res = ols(y,x); prt(res,vnames);
ys = studentize(y); xs = studentize(x(:,2:k));
res2 = ols(ys,xs);
vnames2 = strvcat('hprice','crime','zoning','industry','charlesr', ...
         'noxsq','rooms2','houseage','distance','access','taxrate', ...
         'pupil/teacher','blackpop','lowclass');
prt(res2,vnames2);
% sort actual and predicted by housing values from low to high
yhat = res2.yhat; [ysort yi] = sort(ys); yhats = yhat(yi,1);
tt=1:n; % plot actual vs. predicted
plot(tt,ysort,'ok',tt,yhats,'+k');
```

```
ylabel('housing values');
xlabel('census tract observations');
```

The results indicate that the coefficient estimates based on the unscaled data vary widely in magnitude from 0.000692 to 36.459, whereas the scaled variables produce coefficients ranging from 0.0021 to -0.407.

```
Ordinary Least-squares Estimates (non-scaled variables)
Dependent Variable =    hprice
R-squared      =    0.7406
Rbar-squared   =    0.7338
sigma^2        =   22.5179
Durbin-Watson  =    1.2354
Nobs, Nvars    =    506,    14
***************************************************************
Variable          Coefficient      t-statistic    t-probability
constant            36.459488         7.144074         0.000000
crime               -0.108011        -3.286517         0.001087
zoning               0.046420         3.381576         0.000778
industry             0.020559         0.334310         0.738288
charlesr             2.686734         3.118381         0.001925
noxsq              -17.766611        -4.651257         0.000004
rooms2               3.809865         9.116140         0.000000
houseage             0.000692         0.052402         0.958229
distance            -1.475567        -7.398004         0.000000
access               0.306049         4.612900         0.000005
taxrate             -0.012335        -3.280009         0.001112
pupil/teacher       -0.952747        -7.282511         0.000000
blackpop             0.009312         3.466793         0.000573
lowclass            -0.524758       -10.347146         0.000000

Ordinary Least-squares Estimates (scaled variables)
Dependent Variable =    hprice
R-squared      =    0.7406
Rbar-squared   =    0.7343
sigma^2        =    0.2657
Durbin-Watson  =    1.2354
Nobs, Nvars    =    506,    13
***************************************************************
Variable          Coefficient      t-statistic    t-probability
crime               -0.101017        -3.289855         0.001074
zoning               0.117715         3.385011         0.000769
industry             0.015335         0.334650         0.738032
charlesr             0.074199         3.121548         0.001905
noxsq               -0.223848        -4.655982         0.000004
rooms2               0.291056         9.125400         0.000000
houseage             0.002119         0.052456         0.958187
distance            -0.337836        -7.405518         0.000000
access               0.289749         4.617585         0.000005
taxrate             -0.226032        -3.283341         0.001099
pupil/teacher       -0.224271        -7.289908         0.000000
blackpop             0.092432         3.470314         0.000565
lowclass            -0.407447       -10.357656         0.000000
```

The program in example 5.3 also produces a plot of the actual versus predicted values from the model sorted by housing values from low to high. From this plot (shown in Figure 5.3), we see large predicted errors for the highest housing values. This suggests a log transformation on the dependent variable y in the model would be appropriate. The figure also illustrates that housing values above \$50,000 have been censored to a value of \$50,000.

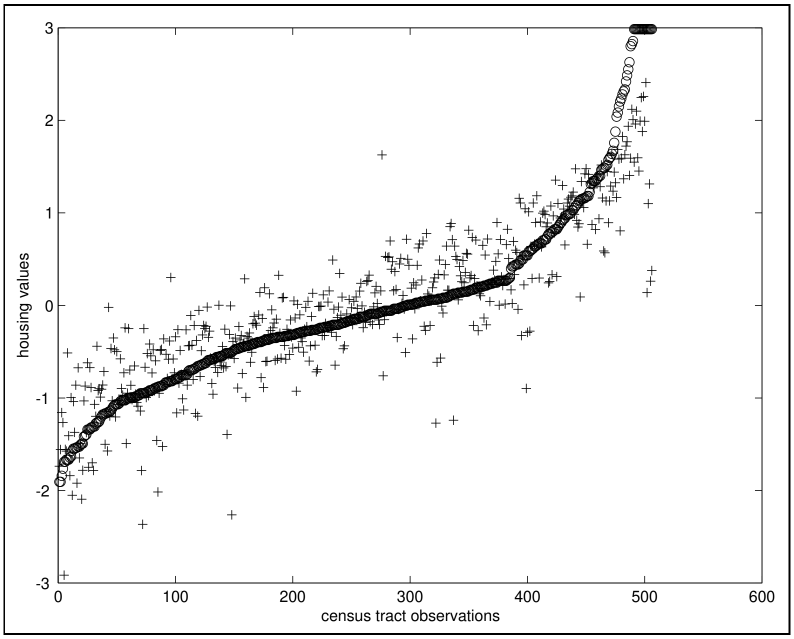

Figure 5.3: Actual vs. Predicted housing values

We adopt a model based on the scaled data and a log transformation for the dependent variable and carry out least-squares estimation again in example 5.4. As a test for spatial autocorrelation in the least-squares residuals, we employ a first-order spatial autoregressive model on the residuals. We also carry out a Moran's I test for spatial autocorrelation, which may or may not work depending on how much RAM memory you have in your computer. Note that we can directly use the **prt** function without first returning the results from **moran** to a results structure. (This is true of all functions in the spatial econometrics toolbox.) We rely on our function **xy2cont** to generate a spatial contiguity matrix needed by **far** and **moran** to test for spatial autocorrelation.

```
% ----- Example 5.4 Testing for spatial correlation
load boston.raw; % Harrison-Rubinfeld data
load latitude.data; load longitude.data;
[W1 W W3] = xy2cont(latitude,longitude); % create W-matrix
[n k] = size(boston);y = boston(:,k);      % median house values
x = boston(:,1:k-1);                       % other variables
vnames = strvcat('hprice','crime','zoning','industry','charlesr', ...
         'noxsq','rooms2','houseage','distance','access','taxrate', ...
         'pupil/teacher','blackpop','lowclass');
```

```
ys = studentize(log(y)); xs = studentize(x);
res = ols(ys,xs); prt(res,vnames);
resid = res.resid; % recover residuals
rmin = 0; rmax = 1;
res2 = far(resid,W,rmin,rmax); prt(res2);
prt(moran(ys,xs,W));
```

The results shown below indicate strong evidence of spatial autocorrelation in the residuals from the least-squares model. Our FAR model produced a spatial correlation coefficient estimate of 0.647 with a large $t-$statistic and this indication of spatial autocorrelation in the residuals is confirmed by the Moran test results.

```
Ordinary Least-squares Estimates
Dependent Variable =    hprice
R-squared      =    0.7896
Rbar-squared   =    0.7845
sigma^2        =    0.2155
Durbin-Watson  =    1.0926
Nobs, Nvars    =    506,    13
***************************************************************
Variable          Coefficient      t-statistic    t-probability
crime               -0.216146        -7.816230         0.000000
zoning               0.066897         2.136015         0.033170
industry             0.041401         1.003186         0.316263
charlesr             0.062690         2.928450         0.003564
noxsq               -0.220667        -5.096402         0.000000
rooms2               0.156134         5.435516         0.000000
houseage             0.014503         0.398725         0.690269
distance            -0.252873        -6.154893         0.000000
access               0.303919         5.377976         0.000000
taxrate             -0.258015        -4.161602         0.000037
pupil/teacher       -0.202702        -7.316010         0.000000
blackpop             0.092369         3.850718         0.000133
lowclass            -0.507256       -14.318127         0.000000

(Test for spatial autocorrelation using FAR model)
First-order spatial autoregressive model Estimates
R-squared       =    0.3085
sigma^2         =    0.1452
Nobs, Nvars     =    506,     1
log-likelihood  =       -911.39591
# of iterations =      9
min and max rho =    0.0000,   1.0000
***************************************************************
Variable      Coefficient      t-statistic    t-probability
rho              0.647624        14.377912         0.000000

Moran I-test for spatial correlation in residuals
Moran I                     0.35833634
Moran I-statistic          14.70009315
Marginal Probability        0.00000000
mean                       -0.01311412
standard deviation          0.02526858
```

Example 5.5 estimates a series of alternative spatial autoregressive models using maximum likelihood in an attempt to find the most appropriate model.

```
% ----- Example 5.5 Spatial autoregressive model estimation
load boston.raw; % Harrison-Rubinfeld data
load latitude.data; load longitude.data;
[W1 W W3] = xy2cont(latitude,longitude); % create W-matrix
W2 = slag(W,2);                          % 2nd order W-matrix
[n k] = size(boston);y = boston(:,k);    % median house values
x = boston(:,1:k-1);;                    % other variables
vnames = strvcat('hprice','constant','crime','zoning','industry','charlesr', ...
         'noxsq','rooms2','houseage','distance','access','taxrate', ...
         'pupil/teacher','blackpop','lowclass');
ys = studentize(log(y)); xs = [ones(n,1) studentize(x)];
rmin = 0; rmax = 1;
tic; res1 = sar(ys,xs,W,rmin,rmax); prt(res1,vnames); toc;
tic; res2 = sem(ys,xs,W,rmin,rmax); prt(res2,vnames); toc;
tic; res3 = sac(ys,xs,W,W2);        prt(res3,vnames); toc;
```

The results from example 5.5 are presented in Table 5.3. We see that all three models produced estimates indicating significant spatial autocorrelation. For example, the SAR model produced a coefficient estimate for ρ equal to 0.4508 with a large $t-$statistic, and the SEM model produced an estimate for λ of 0.7576 that was also significant. The SAC model produced estimates of ρ and λ that were both significant at the 99% level.

Which model is best? The log-likelihood function values are much higher for the SEM and SAC models, so this would be evidence against the SAR model. A further test of the SAR model would be to use the function **lmsar** that tests for spatial autocorrelation in the residuals of the SAR model. If we find evidence of residual spatial autocorrelation, it suggests that the SAC model might be most appropriate. Note that the SAC model exhibits the best log-likelihood function value.

The results from the **lmsar** test shown below indicate the presence of spatial autocorrelation in the residuals of the SAR model, suggesting that the SEM or SAC models would be more appropriate.

```
LM error tests for spatial correlation in SAR model residuals
LM value                  60.37309581
Marginal Probability       0.00000000
chi(1) .01 value           6.63500000
```

It seems difficult to choose between the SEM and SAC models as their likelihood function values are very similar, but the lack of statistical significance of the coefficient λ in the SAC model might make us lean toward the SEM model.

Regarding inferences, an interesting point is that the Charles River location dummy variable was statistically significant in the least-squares version of the model, but not in any of the three spatial autoregressive models. Intuitively, taking explicit account of the spatial nature of the data eliminates the need for this locational dummy variable. Other differences in the inferences that would be made from least-squares versus the SEM and SAC models center on the magnitudes of 'pupil/teacher' ratio and 'lower class' population variables. The least-squares estimates for these two variables are roughly twice the magnitude of those from the SAC model. Since the other two spatial autoregressive models

Table 5.3: SAR,SEM,SAC model comparisons

Variable	sar	$t-$stat	sem	$t-$stat	sac	$t-$statistic
constant	-0.0019	-0.11	-0.0079	-0.12	-0.0073	-0.13
crime	-0.1653	-6.88	-0.1867	-8.44	-0.1895	-8.47
zoning	0.0806	3.00	0.0563	1.81	0.0613	1.97
industry	0.0443	1.25	-0.0002	-0.00	0.0041	0.08
charlesr	0.0171	0.91	-0.0145	-0.67	-0.0118	-0.55
noxsq	-0.1296	-3.43	-0.2202	-3.68	-0.2133	-3.65
rooms2	0.1608	6.54	0.1985	8.32	0.2001	8.30
houseage	0.0185	0.59	-0.0649	-1.74	-0.0567	-1.50
distance	-0.2152	-6.10	-0.2245	-3.42	-0.2549	-3.96
access	0.2722	5.62	0.3521	5.44	0.3538	5.56
taxrate	-0.2212	-4.16	-0.2575	-4.52	-0.2582	-4.53
pup/teach	-0.1023	-4.08	-0.1223	-3.83	-0.1213	-3.85
blackpop	0.0775	3.77	0.1290	4.80	0.1260	4.70
lowclass	-0.3376	-10.14	-0.3803	-10.62	-0.3805	-10.72
rho	0.4509	12.34			0.7123	13.61
lambda			0.7576	19.13	0.1269	1.49
R^2	0.8421		0.8708		0.8688	
ln likel	-85.0928		-58.5977		-57.6506	

produce similar estimates for these two variables, we would infer that the least-squares estimates for these coefficients are exhibiting upward bias.

To a lesser extent, we would draw a different inference regarding the magnitude of impact for the 'rooms2' variable from the two spatial autoregressive models that we think most appropriate (SEM and SAC) and least-squares. The SEM and SAC models produce estimates around 0.2 compared to a value of 0.156 from least-squares. Interestingly, the SAR model estimate for this variable is 0.160, close to that from least-squares.

We used this data set to explore the accuracy of the numerical hessian versus information matrix approach to determining the variance-covariance matrix for the estimates. Since this is a reasonably large dataset with a large number of explanatory variables, it should provide a good indication of how well the numerical hessian approach does. The $t-$statistics from the SAR, SEM and SAC models estimated with both the information matrix approach and the numerical hessian are presented in Table 5.4.

The numerical approach appears to work very well, producing identical inferences for all coefficients except λ in the SAC model. The SAC model produced the greatest divergence between the $t-$statistics and unfortunately large discrepancies exists for both spatial parameters ρ and λ in this model. With the exception of the coefficient for λ in the SAC model, we would draw the same inferences regarding the significance of these parameters from both approaches to computing measures of dispersion. Pace and Barry (1998) express the belief that the information matrix approach (whether computed numerically or analytically) may not work well for spatial autoregressive models because of the

Table 5.4: Information matrix vs. numerical hessian measures of dispersion

time	10.6	51.9	77.4	67.8	207.7	398.1
Variable	SAR	SAR	SEM	SEM	SAC	SAC
	(info)	(hess)	(info)	(hess)	(info)	(hess)
const	-0.111	-0.111	-0.120	-0.120	-0.132	-0.401
crime	-6.888	-6.763	-8.440	-8.475	-8.476	-8.467
zoning	3.009	3.007	1.819	1.819	1.971	2.440
indus	1.255	1.255	-0.004	-0.004	0.087	0.124
chas	0.918	0.922	-0.679	-0.691	-0.555	-0.686
noxsq	-3.433	-3.397	-3.683	-3.687	-3.653	-5.687
rooms2	6.547	6.521	8.324	8.344	8.308	8.814
age	0.595	0.593	-1.741	-1.753	-1.505	-1.909
dis	-6.103	-6.037	-3.420	-3.433	-3.968	-7.452
access	5.625	5.577	5.446	5.476	5.569	7.669
taxrate	-4.166	-4.147	-4.526	-4.526	-4.530	-5.233
ptratio	-4.088	-4.025	-3.838	-3.885	-3.856	-5.116
b	3.772	3.744	4.803	4.808	4.708	6.364
lstat	-10.149	-9.900	-10.626	-10.954	-10.729	-12.489
rho	12.349	10.353	19.135	20.892	13.616	38.140
lam					2.904	1.491

asymmetry of the profile likelihood that arises from the restricted range of the parameter ρ. Additionally, they argue that the necessary "smoothness" needed to compute well-behaved second derivatives may not always occur as Ripley (1988) documents for a particular case. Given this, we should perhaps qualify our evaluation of success regarding the variance-covariance calculations. On the other hand, for this particular data set and model, the variance-covariance structure for the spatial autoregressive models is remarkably similar to that from the least-squares model as indicated by the similar magnitudes for the $t-$statistics. Further, for the single case of the Charles River dummy variable where the least-squares and spatial $t-$statistics differed, we have a plausible explanation for this difference.

Another point to keep in mind is that normality and constant variance of the disturbances may not be a good assumption. The presence of outliers and non-constant variance might impact on the use of both the information matrix and numerical hessian approaches to calculating measures of dispersion.

The times (in seconds) required by both information matrix and numerical hessian approaches to estimating the variance-covariance structure of the parameters are reported in Table 5.4. We see that the information matrix approach was quite a bit faster for the SAR and SAC models and slightly slower for the SEM model. One point to consider regarding execution times is that the spatial contiguity weight matrix is not exceptionally sparse for this problem. Of the (506x506)=256,036 elements, there are 3,006 non-zero entries which is 1.17 percent of the elements. This would bias upward the times reported for the numerical hessian approach since the sparse algorithms can't work to their fullest.

Example 5.6 turns attention to the Bayesian heteroscedastic models for this data set. Spatial autoregressive models are estimated that ignore the sample truncation along with a series of tobit spatial autoregressive models for comparison.

Example 5.6 sorts by the dependent variable, median housing values and also sorts the explanatory variables using the index vector 'yind' returned by the **sort** function applied to the y values. We also need to sort the latitude and longitude vectors. After carrying out Gibbs sampling estimation for the SAR, SEM and SAC models, we add to the prior structure variable a field for right-truncation and we supply the limit value which is the log of 50,000 standardized. Since the y vector is sorted, this transformed limit value must equal the last 16 observations, so we use the last to define the limit value.

```
% ----- Example 5.6 Right-censored Tobit for the Boston data
load boston.raw; % Harrison-Rubinfeld data
load latittude.data; load longitude.data;
[n k] = size(boston);y = boston(:,k);     % median house values
% sort by median house values
[ys yind] = sort(y); xs = boston(yind,1:k-1);
lats = latittude(yind,1); lons = longitude(yind,1);
[W1 W W3] = xy2cont(lats,lons); % create W-matrix
W2 = slag(W,2);
clear W1 W3;
vnames = strvcat('hprice','constant','crime','zoning','industry', ...
'charlesr','noxsq','rooms2','houseage','distance','access','taxrate', ...
'pupil/teacher','blackpop','lowclass');
y = studentize(log(ys)); x = [ones(n,1) studentize(xs)];
% define censoring limit
limit = y(506,1); % median values >=50,000 are censored to 50
ndraw = 1100; nomit = 100; prior.rval = 4;
% do maximum likelihood
res1 = sar(y,x,W);    prt(res1,vnames);
rmin = res1.rmin; rmax = res1.rmax;
res2 = sem(y,x,W);    prt(res2,vnames);
lmin = res2.lmin;  lmax = res2.lmax;
res3 = sac(y,x,W,W2); prt(res3,vnames);
res4 = sac(y,x,W2,W); prt(res4,vnames);
% ignore censoring
prior.rmin = rmin; prior.rmax = rmax; prior.rval = 4;
res1g = sar_g(y,x,W,ndraw,nomit,prior);     prt(res1g,vnames);
prior.lmin = rmin; prior.lmax = rmax;
res2g = sem_g(y,x,W,ndraw,nomit,prior);     prt(res2g,vnames);
prior.rmin = rmin; prior.rmax = rmax;
prior.lmin = lmin; prior.lmax = lmax;
res3g = sac_g(y,x,W,W2,ndraw,nomit,prior);  prt(res3g,vnames);
% use Tobit for censoring
prior.trunc = 'right'; prior.limit = limit;
res1t = sart_g(y,x,W,ndraw,nomit,prior);    prt(res1t,vnames);
prior.lmin = rmin; prior.lmax = rmax;
res2t = semt_g(y,x,W,ndraw,nomit,prior);    prt(res2t,vnames);
prior.rmin = rmin; prior.rmax = rmax;
prior.lmin = lmin; prior.lmax = lmax;
res3t = sact_g(y,x,W,W2,ndraw,nomit,prior); prt(res3t,vnames);
```

Intuitively, we might not expect a large difference in the parameter estimates for this case where only 16 of the 506 sample observations are censored. The results are presented in Table 5.5 for the SAR and tobit version of the SAR model along with a discussion of these results.

Table 5.5: SAR and SAR tobit model comparisons

Variable	sar	$t-$statistic	sar tobit	$t-$statistic
constant	-0.0346	-2.28	-0.0358	-2.27
crime	-0.1552	-5.31	-0.1250	-4.43
zoning	0.0442	1.99	0.0539	2.13
industry	0.0456	1.65	0.0290	0.99
charlesr	0.0250	1.59	-0.0038	-0.21
noxsq	-0.0807	-2.41	-0.0471	-1.46
rooms2	0.3111	10.34	0.2475	7.98
houseage	-0.0493	-1.69	-0.0481	-1.60
distance	-0.1640	-5.34	-0.1577	-4.98
access	0.1633	3.61	0.1655	3.48
taxrate	-0.2059	-4.41	-0.2074	-4.21
pupil/teacher	-0.1220	-6.13	-0.0754	-3.60
blackpop	0.1190	5.49	0.0926	4.21
lowclass	-0.3044	-7.64	-0.2323	-6.35
rho	0.2466	8.34	0.4243	15.81
R^2	0.8098		0.8116	

Contrary to our expectation that these two sets of estimates would produce identical inferences, some interesting and perhaps substantive differences arise. First, we see a difference in the autoregressive parameter ρ which is nearly twice as large for the tobit model.

In comparing the estimates that ignore sample censoring to the tobit estimates, we find evidence that the 'noxsq' air pollution variable is insignificant in the tobit model. Both the maximum likelihood and Bayesian SAR estimates indicate this variable is significant, whereas the tobit version of the SAR model points to insignificance. Figure 5.4 shows a plot of the posterior mean of the V_i estimates from both the Bayesian SAR model and the Bayesian tobit model. An interesting difference arises in the V_i estimates for the censored observations at the end of the sample. The tobit model does not produce large values whereas the non-tobit model detects these observations as outliers and produces large V_i estimates for these observations. This might explain the difference in the estimates since the non-tobit model is downweighting the censored observations whereas the tobit model is imputing values for these observations using the draws for the latent variable z_i in this model.

With the exception of the 'noxsq' variable, none of the other variables in the SAR and tobit SAR models would lead to different inferences using the 0.05 level of significance. This is as we would expect, given only 16 censored observations in a relatively large sample of 506 observations. Nonetheless, the dramatic difference in the estimate for ρ between the tobit and non-tobit versions of the

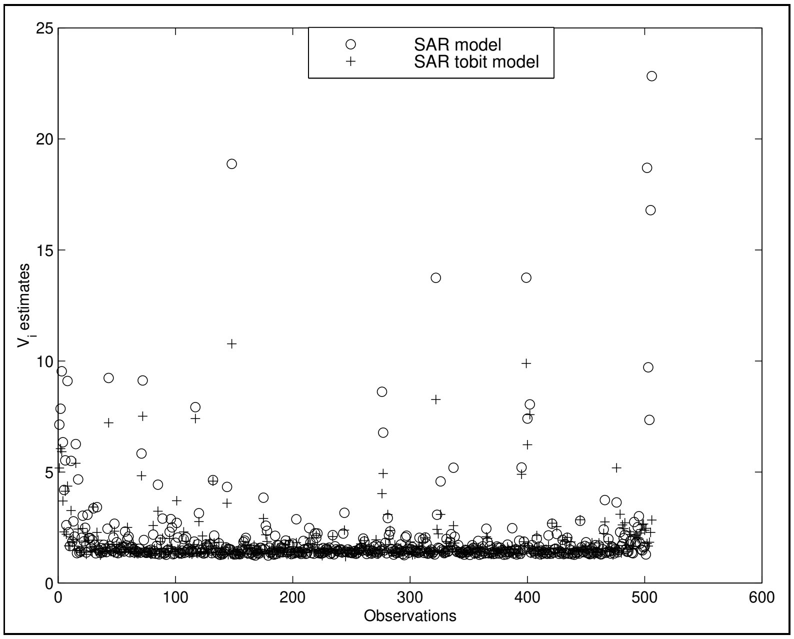

Figure 5.4: V_i estimates for the Boston data set

model suggest that one shouldn't ignore sample censoring.

The estimates from the Bayesian spatial error model and the tobit version of this model are presented in Table 5.6.

As in the case of the SAR model, we see a large difference in the magnitude of the spatial autoregressive parameter λ in these two versions of the model. There is also a difference in the inferences one would draw regarding the 'zoning' variable, which is significant in the non-tobit model and insignificant in the tobit model. The 'noxsq' variable is not significant at the 0.01 level in either the tobit or non-tobit versions of the model.

Finally, we present estimates from the SAC model and the SAC tobit version of the model in Table 5.7. Here we see a difference in the inference one would draw regarding the significance of the spatial coefficient λ which is insignificant in the non-tobit model and significant in the tobit version of the model.

We see similar evidence regarding the weak significance of the 'noxsq' variable in both of these models since this variable is not significant at the 0.01 level.

Summarizing, we found important differences in the inferences one would draw from the Bayesian heteroscedastic and maximum likelihood estimates. The maximum likelihood estimates for all spatial autoregressive model specifications indicated that air pollution measured by 'noxsq' was negative and significant at the 0.01 level. None of the Bayesian heteroscedastic models produced estimates for this variable that were significant at the 0.01 level. This might represent a

Table 5.6: SEM and SEM tobit model comparisons

Variable	sem	$t-$statistic	sem tobit	$t-$statistic
constant	-0.0626	-0.89	-0.0320	-1.01
crime	-0.1627	-5.89	-0.1567	-5.58
zoning	0.0480	2.04	0.0410	1.32
industry	-0.0067	-0.18	0.0102	0.25
charlesr	-0.0175	-0.99	0.0060	0.24
noxsq	-0.1249	-2.45	-0.1142	-2.32
rooms2	0.3850	17.46	0.3049	7.71
houseage	-0.1537	-5.38	-0.0985	-2.72
distance	-0.1493	-2.59	-0.1807	-4.02
access	0.2550	4.10	0.2165	3.61
taxrate	-0.2293	-4.82	-0.2290	-4.17
pupil/teacher	-0.0717	-3.16	-0.1153	-4.41
blackpop	0.1445	6.93	0.1375	5.63
lowclass	-0.1872	-5.14	-0.3062	-6.69
lambda	0.8195	24.75	0.6736	12.62
R^2	0.8565		0.8301	

Table 5.7: SAC and SAC tobit model comparisons

Variable	sac	$t-$statistic	sac tobit	$t-$statistic
constant	-0.0395	-1.14	-0.0495	-1.52
crime	-0.1553	-5.70	-0.1458	-5.29
zoning	0.0434	1.83	0.0426	1.77
industry	0.0166	0.46	0.0079	0.21
charlesr	0.0078	0.41	-0.0051	-0.28
noxsq	-0.1070	-2.23	-0.1002	-2.34
rooms2	0.3677	13.57	0.3291	12.05
houseage	-0.1192	-3.73	-0.1110	-3.60
distance	-0.1762	-4.05	-0.1696	-3.98
access	0.1980	3.13	0.1957	3.48
taxrate	-0.2237	-4.11	-0.2324	-4.37
pupil/teacher	-0.1017	-4.39	-0.1000	-4.24
blackpop	0.1406	6.33	0.1335	6.30
lowclass	-0.2616	-6.08	-0.2681	-7.00
rho	0.7420	14.52	0.7031	13.48
lambda	0.1113	1.62	0.1479	2.52
R^2	0.8544		0.8363	

difference in inferences that would have important policy ramifications.

In addition to the differences between maximum likelihood and Bayesian heteroscedastic estimates, there are also differences between the Bayesian SAR, SEM and SAC models and the tobit variants of these models. All three models exhibits differences regarding the spatial autoregressive parameters ρ and λ. A plot of the v_i parameters estimated by these models showed that the non-tobit versions of the models treated the censored observations as outliers and downweighted them. In addition to the differences in the spatial autoregressive parameters from these models there were also other parameters where we would draw a different inference from the two versions of the models.

5.8 Chapter Summary

A Gibbs sampling approach to estimating heteroscedastic spatial autoregressive probit and tobit models was presented. With the exception of McMillen (1992) who set forth an EM algorithm approach to estimating spatial autoregressive models in the presence of heteroscedastic disturbances, no other methods exist for producing estimates under these conditions. It was argued that the Bayesian approach set forth here has several advantages over the EM algorithm approach suggested by McMillen (1992). First, the method produces posterior distributions for all parameters in the model whereas McMillen's approach does not provide estimates of precision for the spatial parameters ρ and λ. The posteriors allow for inferences regarding the mean and dispersion of all parameters, including the important spatial parameters.

A second advantage is that the Gibbs sampled measures of dispersion based on the posterior distributions are valid whereas the EM algorithm produces consistent estimates of dispersion that are likely to overstate parameter precision.

Perhaps the greatest advantage of the Bayesian approach introduced here is that no model for the non-constant variance need be specified by the investigator. The Gibbs sampling approach produces estimates of the non-constant variance for every observation in space. These estimates can be used to draw inferences regarding the presence of spatial outliers or general patterns of non-constant variance over space.

Another point is that the EM methods introduced in McMillen do not apply to tobit models where the likelihood function takes a more complicated form than the probit model. The Gibbs sampling approach introduced here applies to the tobit model as well as probit and is equally easy to implement. In addition, the Gibbs sampling approach to estimating the heteroscedastic spatial probit model subsumes a logit version of the model as a special case.

Finally, because the approach introduced here is quite similar to the Gibbs sampling approach for spatial autoregressive models presented in Chapter 4, it provides a unified methodology for estimating spatial autoregressive models that involve continuous or dichotomous dependent variables.

An applied example demonstrated that ignoring sample truncation or censoring may lead to important differences in the inferences one would make. These differences arose in a sample of 506 observations with only 16 censored observations.

Chapter 6

Locally linear spatial models

This chapter discusses estimation methods that attempt to accommodate spatial heterogeneity by allowing the parameters of the model to vary with the spatial location of the sample data. The first section deals with spatial and distance expansion models introduced by Casetti (1972,1992). A more recent variant labeled a DARP model in Casetti (1982) and Casetti and Can (1998) is the subject of Section 6.2.

Non-parametric locally linear regression models (sometimes labeled geographically weighted regression) represent another way to deal with spatial heterogeneity. These models are covered in Section 6.3. Section 6.4 provides applied illustrations of the methods with the Pace and Barry data set and the Boston data set from Harrison and Rubinfeld (1978). These models are extended to the case of limited dependent variables in Section 6.5. Chapter 7 sets forth a Bayesian approach to locally linear spatial regression models.

6.1 Spatial expansion

The first model of this type was introduced by Casetti (1972) and labeled a spatial expansion model. The model is shown in (6.1), where y denotes an nx1 dependent variable vector associated with spatial observations and X is an nxnk matrix consisting of terms x_i representing kx1 explanatory variable vectors, as shown in (6.2). The locational information is recorded in the matrix Z which has elements $Z_{xi}, Z_{yi}, i = 1, \ldots, n$, that represent latitude and longitude coordinates of each observation as shown in (6.2).

The model posits that the parameters vary as a function of the latitude and longitude coordinates. The only parameters that need be estimated are the parameters in β_0 that we denote, β_x, β_y. These represent a set of $2k$ parameters. Recall our discussion about spatial heterogeneity and the need to utilize a parsimonious specification for variation over space. This represents one approach to this type of specification.

We note that the parameter vector β in (6.1) represents an nkx1 matrix in this model that contains parameter estimates for all k explanatory variables at every observation. The parameter vector β_0 contains the $2k$ parameters to be estimated.

$$\begin{aligned} y &= X\beta + \varepsilon \\ \beta &= ZJ\beta_0 \end{aligned} \tag{6.1}$$

Where:

$$\begin{aligned} y &= \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \; X = \begin{pmatrix} x_1' & 0 & \dots & 0 \\ 0 & x_2' & & \\ \vdots & & \ddots & \\ 0 & & & x_n' \end{pmatrix} \; \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} \; \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \\ Z &= \begin{pmatrix} Z_{x1} \otimes I_k & Z_{y1} \otimes I_k & 0 & \dots \\ 0 & \ddots & \ddots & \\ \vdots & & Z_{xn} \otimes I_k & Z_{yn} \otimes I_k \end{pmatrix} J = \begin{pmatrix} I_k & 0 \\ 0 & I_k \\ \vdots & \\ 0 & I_k \end{pmatrix} \\ \beta_0 &= \begin{pmatrix} \beta_x \\ \beta_y \end{pmatrix} \end{aligned} \tag{6.2}$$

This model can be estimated using least-squares to produce estimates of the $2k$ parameters β_x, β_y. Given these estimates, the remaining estimates for individual points in space can be derived using the second equation in (6.1). This process is referred to as the "expansion process". To see this, substitute the second equation in (6.1) into the first, producing:

$$y = XZJ\beta_0 + \varepsilon \tag{6.3}$$

Here it is clear that X, Z and J represent available information or data observations and only β_0 represent parameters in the model that need be estimated.

The model would capture spatial heterogeneity by allowing variation in the underlying relationship such that clusters of nearby or neighboring observations measured by latitude-longitude coordinates take on similar parameter values. As the location varies, the regression relationship changes to accommodate a linear fit through clusters of observations in close proximity to one another.

Another way to implement this model is to rely on a vector of distances rather than the latitude-longitude coordinates. This implementation defines the distance from a central observation as:

$$d_i = \sqrt{(Z_{xi} - Z_{xc})^2 + (Z_{yi} - Z_{yc})^2} \tag{6.4}$$

Where Z_{xc}, Z_{yc} denote the latitude-longitude coordinates of the centrally located observation and Z_{xi}, Z_{yi} denote the latitude-longitude coordinates for observation i in the data sample.

This approach allows one to ascribe different weights to observations based on their distance from the central place origin. The formulation discussed above would result in a distance vector that increased with distance from the central observation. This would be suitable if one were modeling a phenomena reflecting a "hollowing out" of the central city or a decay of influence with distance from the central point.

The distance expansion model can be written as:

$$\begin{aligned} y &= X\beta + \varepsilon \\ \beta &= DJ\beta_0 \end{aligned} \tag{6.5}$$

Where $D = \text{diag}(d_1, d_2, \ldots, d_n)$, represents the distance of each observation from the central place and β_0 represents a kx1 vector of parameters for the central place. The matrix J in (6.5) is an nxk matrix, $J = (I_k, I_k, \ldots, I_k)'$.

6.1.1 Implementing spatial expansion

Estimating this model is relatively straightforward as we can rely on least-squares. One issue is that there are a number of alternative expansion specifications. For example, one approach would be to construct a model that includes the base k explanatory variables in the matrix X, estimated with fixed parameters, plus an additional $2k$ expansion variables based on the latitude-longitude expansion. Another approach would be to include the base k variables in the matrix X and only $2(k-1)$ variables in expansion form by excluding the constant term from the expansion process. Yet another approach would be to rely on a simple expansion of all variables as was illustrated in (6.1). The second approach was taken in implementing the MATLAB function **casetti** that carries out spatial expansion estimation.

This choice was made because it seems unwise to include the constant term in the expansion as one can overfit the sample data when the intercept is allowed to vary over space. A motivation for not relying on a simple expansion of all variables is that we would like our model to partition the influence of explanatory variables into fixed plus spatial effects. A simple expansion assigns all influence to spatial effects and also falls prey to the overfitting problem by allowing the intercept term to vary.

The expansion implemented by our function **casetti** can be written as:

$$y = \alpha + X\beta + XZ_x\beta_x + XZ_y\beta_y + \varepsilon \tag{6.6}$$

The function allows the user to specify an option for distance expansion based on a particular point in the spatial data sample or the latitude-longitude expansion. In the case of distance expansion, all k explanatory variables including the constant term are used as non-expansion variables estimated with fixed parameters and $k-1$ variables excluding the constant are included as distance-expanded variables. This version of the model can be written as:

$$y = \alpha + X\beta + XD\beta_0 + \varepsilon \tag{6.7}$$

For the case of distance expansion, a distance vector is calculated as: $d_i = \sqrt{(Z_{xi} - Z_{xc})^2 + (Z_{yi} - Z_{yc})^2}$, where Z_{xc}, Z_{yc} denote the latitude-longitude coordinates of the centrally located observation and Z_{xi}, Z_{yi} denote the coordinates for observation i in the data sample. The distance of the central point is zero of course.

An optional input is provided to carry out isotropic normalization of the x-y coordinates which essentially puts the coordinates in deviations from the means form then standardizes by using the square root of the sum of the variances in the x-y directions. That is:

$$\begin{aligned} x^\star &= (x - \bar{x})/\sqrt{(\sigma_x^2 + \sigma_y^2)} \\ y^\star &= (y - \bar{y})/\sqrt{(\sigma_x^2 + \sigma_y^2)} \end{aligned} \tag{6.8}$$

This normalization is carried out by a function **normxy** in the spatial econometrics library. This normalization should make the center points xc, yc close to zero and produces a situation where the coefficients for the "base model" represent a central observation. The distance-expanded estimates provide information about variation in the model parameters with reference to the central point.

The function **casetti** is presented below.

```
function results=casetti(y,x,xc,yc,option)
% PURPOSE: computes Casetti's spatial expansion regression
%         y = a + X*Bo + X*Zx*Bx + X*Zy*By + e
%     or: y = a + X*Bo + X*D*Bd + e
%---------------------------------------------------
% USAGE: results = casetti(y,x,xc,yc,option)
% where:       y = dependent variable vector
%              x = independent variables matrix
%             xc = latittude (or longitude) coordinate
%             yc = longitude (or latittude) coordinate
%        option  = a structure variable containing options
%        option.exp  = 0 for x-y expansion (default)
%                    = 1 for distance from ctr expansion
%        option.ctr  = central point observation # for distance expansion
%        option.norm = 1 for isotropic x-y normalization (default=0)
%---------------------------------------------------
% RETURNS:
%        results.meth   = 'casetti'
%        results.b0     = bhat (underlying b0x, b0y)
%        results.t0     = t-stats (associated with b0x, b0y)
%        results.beta   = spatially expanded estimates (nobs x nvar)
%        results.yhat   = yhat
%        results.resid  = residuals
%        results.sige   = e'*e/(n-k)
%        results.rsqr   = rsquared
```

```
%        results.rbar   = rbar-squared
%        results.nobs   = nobs
%        results.nvar   = # of variables in x
%        results.y      = y data vector
%        results.xc     = xc
%        results.yc     = yc
%        results.ctr    = ctr (if input)
%        results.dist   = distance vector (if ctr used)
%        results.exp    = exp input option
%        results.norm   = norm input option
% -------------------------------------------------
% NOTES: assumes x(:,1) contains a constant term
% -------------------------------------------------
nflag = 0; exp = 0; ctr = 0; nflag = 0;
if nargin == 5 % user options
 if ~isstruct(option)
    error('casetti: must supply the option argument as a structure variable');
 else, fields = fieldnames(option); nf = length(fields);
  for i=1:nf
    if strcmp(fields{i},'exp'),     exp = option.exp;
    elseif strcmp(fields{i},'ctr'), ctr = option.ctr;
    elseif strcmp(fields{i},'norm') nflag = option.norm;
    end;
  end; % end of for i
 end; % end of if else
elseif nargin == 4  % x-y expansion
exp = 0; option.exp = 0;
else, error('Wrong # of arguments to casetti');
end;
[nobs nvar] = size(x);
if x(:,1) ~= ones(nobs,1)
 error('casetti: first column in x-matrix must be a constant vector');
end;
if nflag == 1, [xc yc] = normxy(xc,yc); end;
results.meth = 'casetti'; results.y = y;
results.nobs = nobs;      results.nvar = nvar;
results.xc = xc;          results.yc = yc;
results.option = option;  results.exp = exp;
results.norm = nflag;
switch exp
case {0} % x-y expansion
xt = x(:,2:nvar); xx = matmul(xt,xc); xy = matmul(xt,yc);
xmat = [x xx xy]; b0 = xmat\y;
xpxi = inv(xmat'*xmat); results.b0 = b0;
beta = zeros(nobs,2*(nvar-1)); yhat = zeros(nobs,1);
xx = matmul(ones(nobs,nvar-1),xc); xy = matmul(ones(nobs,nvar-1),yc);
xxxy = [x xx xy]; tvar = length(b0);
    yhat(:,1) = xmat(:,1:nvar)*b0(1:nvar,1);
    for j=nvar+1:tvar
    beta(:,j-nvar) = xxxy(:,j)*b0(j,1);
    yhat(:,1) = yhat(:,1)  + xmat(:,j)*b0(j,1);
    end;
results.beta = beta;  results.yhat = yhat;
e = results.y - results.yhat; results.resid = e;
sigu = e'*e; results.sige = sigu/(nobs-tvar);
tmp2 = results.sige*(diag(xpxi));
results.t0 = results.b0./(sqrt(tmp2));
```

```
ym = y - mean(y); rsqr1 = sigu; rsqr2 = ym'*ym;
results.rsqr = 1.0 - rsqr1/rsqr2; % r-squared
rsqr1 = rsqr1/(nobs-nvar-2*(nvar-1));
rsqr2 = rsqr2/(nobs-1.0);
results.rbar = 1 - (rsqr1/rsqr2); % rbar-squared
case{1} % distance from the center expansion
xi = xc(ctr); yi = yc(ctr);
d = sqrt((xc-xi).*(xc-xi) + (yc-yi).*(yc-yi));
dvec = d; % distance weighting function
results.dist= dvec;
% transform x-variables using distance vector
xt = x(:,2:nvar); xx = matmul(xt,dvec);
xmat = [x xx];    b0 = xmat\y;
xpxi = inv(xmat'*xmat); results.b0 = b0;
beta = zeros(nobs,(nvar-1)); yhat = zeros(nobs,1);
xx = matmul(ones(nobs,nvar-1),dvec); xxxy = [x xx]; tvar = length(b0);
    yhat(:,1) = xmat(:,1:nvar)*b0(1:nvar,1);
    for j=nvar+1:tvar
    beta(:,j-nvar) = xxxy(:,j)*b0(j,1);
    yhat(:,1) = yhat(:,1)  + xmat(:,j)*b0(j,1);
    end;
results.beta = beta; results.yhat = yhat;
results.b0 = b0; results.nvar = nvar; results.ctr = ctr;
results.resid = y - results.yhat;
sigu = results.resid'*results.resid;
results.sige = sigu/(nobs-tvar);
xpxi = inv(xmat'*xmat);
    tmp = results.sige*(diag(xpxi));
    results.t0 = results.b0./(sqrt(tmp));
ym = y - mean(y); rsqr1 = sigu; rsqr2 = ym'*ym;
results.rsqr = 1.0 - rsqr1/rsqr2; % r-squared
rsqr1 = rsqr1/(nobs-2*nvar);
rsqr2 = rsqr2/(nobs-1.0);
results.rbar = 1 - (rsqr1/rsqr2); % rbar-squared
otherwise
error('casetti: check option input argument');
end;
```

Note that given the exclusion of the constant term from the spatial expansion formulation, we need to impose that the user place the constant term vector in the first column of the explanatory variables matrix X input to the function. We perform a check of this using the following MATLAB code:

```
[nobs nvar] = size(x);
if x(:,1) ~= ones(nobs,1)
 error('casetti: first column in x-matrix must be a constant vector');
end;
```

Another point to note about the code is use of the **matmul** command from the *Econometrics Toolbox*. This command allows non-conformable matrix multiplication where the two matrices involved are conformable in at least one dimension. This is a useful programming device in situations where we wish to multiply a vector times each column of a matrix. There are also corresponding functions **matadd, matdiv, matsub** for addition, division and subtraction that can be used in these situations.

A code fragment below from **casetti** shows how we implement the distance expansion estimation. The case of latitude-longitude expansion is similar. After computing the distance, we use the function **matmul** to carry out a non-conformable matrix multiplication of the nx$k-1$ matrix 'xt' (that excludes the constant term) times the nx1 vector of distance coordinates. (Note that MATLAB doesn't allow non-conformable matrix multiplications.) The function **matmul** will multiply the distance vector times every column of the 'xt' matrix which is what we wish to accomplish.

```
xi = xc(ctr); yi = yc(ctr);
d = sqrt((xc-xi).*(xc-xi) + (yc-yi).*(yc-yi));
dvec = d; % distance weighting function
results.dist= dvec;
% transform x-variables using distance vector
xt = x(:,2:nvar); xx = matmul(xt,dvec);
xmat = [x xx];    b0 = xmat\y;
xpxi = inv(xmat'*xmat); results.b0 = b0;
beta = zeros(nobs,(nvar-1)); yhat = zeros(nobs,1);
xx = matmul(ones(nobs,nvar-1),dvec); xxxy = [x xx]; tvar = length(b0);
    yhat(:,1) = xmat(:,1:nvar)*b0(1:nvar,1);
    for j=nvar+1:tvar
    beta(:,j-nvar) = xxxy(:,j)*b0(j,1);
    yhat(:,1) = yhat(:,1)  + xmat(:,j)*b0(j,1);
    end;
```

After transforming the data matrix X using the distance vector, we place the two matrices together in a MATLAB variable 'xmat' and carry out our regression to calculate the 'base' coefficients β_0. We rely on the MATLAB 'slash' operator to solve the problem $\hat{\beta}_0 = (X'X)^{-1}X'y$. These base model estimates are then subjected to the distance expansion process to arrive at estimates for each point in space. While carrying out the expansion process, we also compute the predicted values and store them in a MATLAB variable 'yhat'. Note that the prediction involves using both the constant or base coefficient estimates associated with the k elements in the X matrix, as well as the $k-1$ expanded estimates based on the variables excluding the constant term in the last $k-1$ columns of the matrix 'xmat'.

Given the estimates and predicted values, we are in a position to fill-in the remaining fields of the results structure that will be returned by **casetti**.

Of course, we have an associated function to print the results structure and another to provide graphical presentation of the estimation results. Printing these estimation results is a bit challenging because of the large number of parameter estimates that we produce using this method. Graphical presentation may provide a clearer picture of the variation in coefficients over space. A call to **plt** using the 'results' structure variable will produce plots of the coefficients in both the x- and y-directions. For the case of the latitude-longitude expansion, we sort the x- and y-directions from left to right. This provides a visual picture of how the coefficients vary over space. If the x-coordinates are largest for the east and smallest for the west, the plot will show coefficient variation from west to east as in map space. Similarly, if the y-coordinates are smallest for the south and largest in the north, the plot will present coefficient variation from south to north. (Note that if you enter latitude-longitude coordinates, the x-direction plots will be from east to west, but the y-direction plots will be south to north.)

For the case of distance expansion estimates, the plots present coefficients sorted by distance from the central point given in the input structure field 'option.ctr'. The central observation (smallest distance) will be on the left of the graph and the largest distance on the right.

Another point to note regarding graphical presentation of the estimates relates to the fact that we present the coefficients in terms of the individual variables total impact on the dependent variable y. It was felt that users would usually be concerned with the total impact of a particular variable on the dependent variable as well as the decomposition of impacts into spatial and non-spatial effects. The printed output provides the coefficient estimates for the base model as well as the expansion coefficients that can be used to analyze the marginal effects from the spatial and non-spatial decomposition. To provide another view of the impact of the explanatory variables on the dependent variable, the graphical presentation plots the coefficient estimates in a form representing their total impact on the dependent variable. That is we graph:

$$\begin{aligned} \gamma_{xi} &= \beta_i + Z_x \beta_{xi} \\ \gamma_{yi} &= \beta_i + Z_y \beta_{yi} \\ \gamma_{di} &= \beta_i + D\beta_{0i} \end{aligned} \tag{6.9}$$

Where γ_x, γ_y are plotted for the x-y expansion and γ_d is graphed for the distance expansion. This should provide a feel for the total impact of variable i on the dependent variable since it takes into account the non-spatial impact attributed to β_i, as well as the spatially varying impacts in the x-y direction or with respect to distance.

6.1.2 Applied examples

Example 6.1 provides an illustration of using the function **casetti** based on the Columbus neighborhood crime data set from Anselin (1988). Both types of expansion models are estimated by changing the structure variable 'option' field '.exp'. For the case of distance expansion, we rely on a central observation number 32 which lies near the center of the spatial sample of neighborhoods. One point to note is that the x-coordinate in Anselin's data set represents the south-north direction and the y-coordinate reflects the west-east direction.

```
% ----- Example 6.1 Using the casetti() function
% load Anselin (1988) Columbus neighborhood crime data
load anselin.dat;
y = anselin(:,1); n = length(y); x = [ones(n,1) anselin(:,2:3)];
% Anselin (1988) x-y coordinates
xc0 = anselin(:,4); yc0 = anselin(:,5);
vnames = strvcat('crime','const','income','hse value');
% do Casetti regression using x-y expansion (default)
```

```
res1 = casetti(y,x,xc,yc);
prt(res1,vnames); % print the output
plt(res1,vnames); % graph the output
pause;
% do Casetti regression using distance expansion
option.exp = 1; option.ctr = 32;
res2 = casetti(y,x,xc,yc,option);
prt(res2,vnames); % print the output
plt(res2,vnames); % graph the output
```

The default option is to implement an x-y expansion, which produces the result structure variable 'res1'. The next case relies on the structure variable 'option' to select distance expansion. The printed output is shown below where the expansion estimates are shown for only the first five observations to save space. Both the base estimates as well as the expansion estimates are presented in the printed output. If you are working with a large model containing numerous observations, you can rely on the printing option that places the output in a file. Recall from Chapter 2, we need simply open an output file and input the 'file-id' as an option to the **prt** function.

Another point to note regarding the printed output is that in the case of a large number of explanatory variables, the printed estimates will 'wrap'. A set of estimates that take up 80 columns will be printed for all observations, and remaining estimates will be printed below for all observations. This 'wrapping' will continue until all of the parameter estimates are printed.

```
Casetti X-Y Spatial Expansion Estimates
Dependent Variable =        crime
R-squared      =    0.6330
Rbar-squared   =    0.5806
sige           =  117.4233
Nobs, Nvars    =     49,     3
***************************************************************
Base x-y estimates
Variable         Coefficient      t-statistic    t-probability
const              69.496160        15.105146         0.000000
income             -4.085918        -1.951941         0.057048
hse value           0.403956         0.517966         0.606965
x-income           -0.046062        -1.349658         0.183731
x-hse value         0.026732         2.027587         0.048419
y-income            0.121440         2.213107         0.031891
y-hse value        -0.048606        -2.341896         0.023571
***************************************************************
Expansion estimates
 Obs#    x-income x-hse value    y-income y-hse value
    1     -1.6407      0.9522      5.1466     -2.0599
    2     -1.6813      0.9757      4.9208     -1.9695
    3     -1.6909      0.9813      4.7009     -1.8815
    4     -1.5366      0.8918      4.6645     -1.8670
    5     -1.7872      1.0372      5.3519     -2.1421

Casetti Distance Spatial Expansion Estimates
Dependent Variable =        crime
R-squared      =    0.6568
Rbar-squared   =    0.6169
```

```
sige         =  107.2568
Nobs, Nvars  =     49,     3
central obs  =     32
***************************************************************
Base centroid estimates
Variable         Coefficient      t-statistic    t-probability
const              57.818344        11.071091         0.000000
income              0.192534         0.217136         0.829063
hse value          -0.318927        -1.162013         0.251224
d-income           -1.124634        -1.647297         0.106312
d-hse value         0.073983         0.310034         0.757935
***************************************************************
Expansion estimates
 Obs#      income   hse value
    1     -1.4229      0.0936
    2     -1.1369      0.0748
    3     -0.8925      0.0587
    4     -1.1372      0.0748
    5     -1.5614      0.1027
```

We turn attention to interpreting the output from this example. For this purpose we compare the 'base' spatial expansion estimates to those from least-squares which are presented below. The addition of the four x-y expansion variables increased the fit of the model slightly as indicated by the higher adjusted R^2 statistic. We see that the intercept estimate is relatively unaffected by the inclusion of expansion variables, but the coefficients on income and house value take on very different magnitudes. The significance of the income variable falls as indicated by the lower $t-$statistic, and the house value variable becomes insignificant.

Three of the four x-y expansion variables are significant at the 0.05 level, providing evidence that the influence of these variables on neighborhood crime varies over space. Keep in mind that depending on the amount of inherent variation in the x-y coordinates, we may introduce a substantial amount of collinearity into the model when we add expansion variables. These are likely highly correlated with the base variables in the model, and this may account for the lack of significance of the house value variable in the 'base model'. A strict interpretation of these 'base' estimates would be that income expansion in the x-direction (south-north) is not significant, whereas it is in the y-direction (west-east).

```
Ordinary Least-squares Estimates
Dependent Variable =      crime
R-squared      =    0.5521
Rbar-squared   =    0.5327
sigma^2        =  130.8386
Durbin-Watson  =    1.1934
Nobs, Nvars    =     49,     3
***************************************************************
Variable         Coefficient      t-statistic    t-probability
const              68.609759        14.484270         0.000000
income             -1.596072        -4.776038         0.000019
house value        -0.274079        -2.655006         0.010858
```

In order to interpret the x-y expansion estimates, we need to keep in mind that the x-direction reflects the south-north direction with larger values of xc indicating northward movement. Similarly, larger values for yc reflect west-east movement. Using these interpretations, the base model estimates indicate that income exerts an insignificant negative influence on crime as we move from south to north. Considering the y-direction representing west-east movement, we find that income exerts a positive influence as we move in the easterly direction. One problem with interpreting the expansion estimates is that the base model coefficient is -4.085, indicating that income exerts a negative influence on crime. It is difficult to assess the total impact on neighborhood crime from both the base model coefficient representing non-spatial impact plus the small 0.121 value for the expanded coefficient reflecting the impact of spatial variation.

If we simply plotted the expansion coefficients, they would suggest that income in the y-direction has a positive influence on crime, a counterintuitive result. This is shown in the plot of the expanded coefficients sorted by the south-north and east-west directions in Figure 6.1. We are viewing a positive coefficient on the y-income variable in the graph.

Figure 6.2 shows a graph of the total impact of income on the dependent variable crime that takes into account both the base model non-spatial impact plus the spatial impact indicated by the expansion coefficient. Here we see that the total impact of income on crime is negative, except for neighborhoods in the extreme east at the right of the graph. The coefficient graphs produced using the **plt** function on the results structure from **casetti** are identical to those shown in Figure 6.2. You can of course recover the spatial expansion estimates from the results structure returned by **casetti**, sort the estimates in the x-y directions and produce your own plots of just the expansion estimates if this is of interest. As an example, the following code would produce this type of graph, where we are assuming the existence of a structure 'result' returned by **casetti**.

```
[xcs xci] = sort(result.xc);
[ycs yci] = sort(result.yc);
beta = result.beta;
[nobs nvar] = size(beta);
nvar = nvar/2;
betax =  beta(xci,1:nvar); % sort estimates
betay =  beta(yci,nvar+1:2*nvar);
tt=1:nobs;
for j=1:nvar
plot(tt,betax(:,j)); pause;
end;
for j=1:nvar
plot(tt,betay(:,j)); pause;
end;
```

The distance expansion method produced a slightly better fit to the data as indicated by the adjusted R^2 statistic. This is true despite the fact that only the constant term is statistically significant at conventional levels, with the distance-expanded income variable significant at the 90% level in the base model.

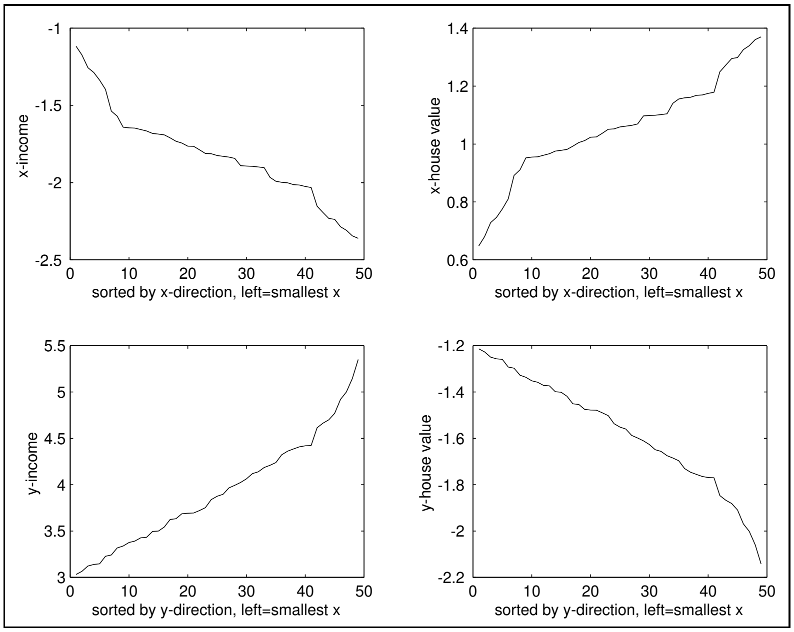

Figure 6.1: Spatial x-y expansion estimates

These estimates take on a value of zero for the central observation since the distance is zero at that point. This makes them somewhat easier to interpret than the estimates for the case of x-y coordinates. Given that the expansion coefficients take on values of zero at the central point, estimates at this point reflect the non-spatial base coefficients. As we move outward from the center, the expansion estimates take over and adjust the constant coefficients to account for variation over space. The printed distance expansion estimates reported for the base model reflect values near the distance-weighted average of all points in space. Given this, we would interpret the coefficients for the model at the central point to be (62.349, -0.855, -0.138) for the intercept, income and house value variables respectively. In Figure 6.3 we see the total impact of income and house values on neighborhood crime as we move away from the central point. Both income and house values have a negative effect on neighborhood crime as we move away from the central city. Note that this is quite different from the pattern shown for the x-y expansion. Anselin (1988) in analyzing the x-y model shows that heteroscedastic disturbances produce problems that plague inferences for the model. Adjusting for the heteroscedastic disturbances dramatically alters the inferences. We turn attention to this issue when we discuss the DARP version of this model in Section 6.2.

The plots of the coefficient estimates provide an important source of information about the nature of coefficient variation over space, but you should keep in mind that they do not indicate levels of significance, simply point estimates.

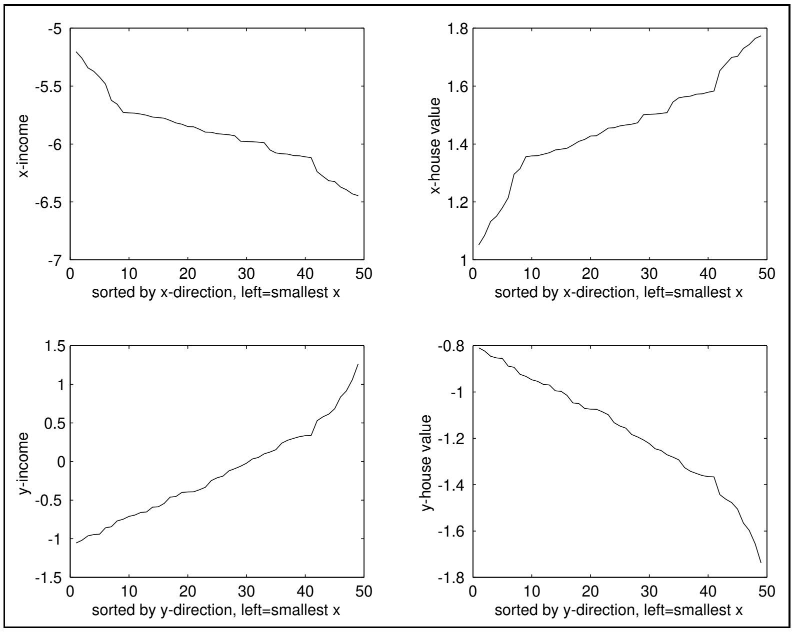

Figure 6.2: Spatial x-y total impact estimates

Our **plt** wrapper function works to call the appropriate function **plt_cas** that provides individual graphs of each coefficient in the model as well as a two-part graph showing actual versus predicted and residuals. Figure 6.4 shows the actual versus predicted and residuals from the distance expansion model. This plot is produced by the **plt** function when given a results structure from the **casetti** function.

6.2 DARP models

A problem with the spatial expansion model is that heteroscedasticity is inherent in the way the model is constructed. To see this, consider the slightly altered version of the distance expansion model shown in (6.10), where we have added a stochastic term u to reflect some error in the expansion relationship.

$$
\begin{aligned}
y &= X\beta + e \\
\beta &= DJ\beta_0 + u
\end{aligned}
\tag{6.10}
$$

Now consider substituting the second equation from (6.10) into the first, producing:

$$
y = XDJ\beta_0 + Xu + e \tag{6.11}
$$

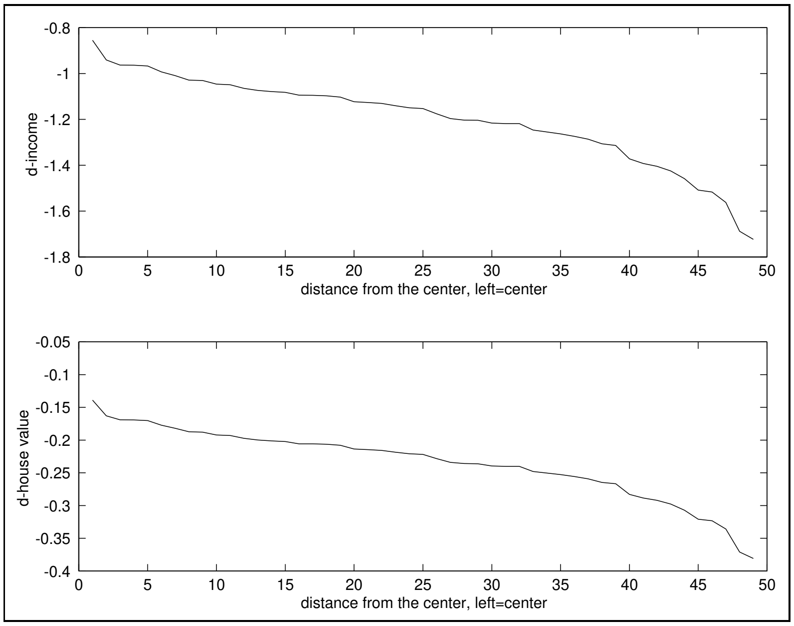

Figure 6.3: Distance expansion estimates

It should be clear that the new composite disturbance term $Xu + e$ will reflect heteroscedasticity unless the expansion relationship is exact and $u = 0$.

Casetti (1982) and Casetti and Can (1998) propose a model they label DARP, an acronym for Drift Analysis of Regression Parameters, that aims at solving this problem. This model case be viewed as an extended expansion model taking the form:

$$\begin{aligned} y &= X\beta + e \\ \beta &= f(Z, \rho) + u \end{aligned} \tag{6.12}$$

Where $f(Z, \rho)$ represents the expansion relationship based on a function f, variables Z and parameters ρ. Estimation of this model attempts to take into account that the expanded model will have a heteroscedastic error as shown in (6.11).

To keep our discussion concrete, we will rely on $f(Z, \rho) = DJ\beta_0$, the distance expansion relationship in discussing this model. To take account of the spatial heteroscedasticity, an explicit model of the composite disturbance term: $\varepsilon = Xu + e$ is incorporated during estimation. This disturbance is assumed to have a variance structure that can be represented in alternative ways shown below. We will rely on these alternative scalar and matrix representations in the mathematical development and explanation of the model.

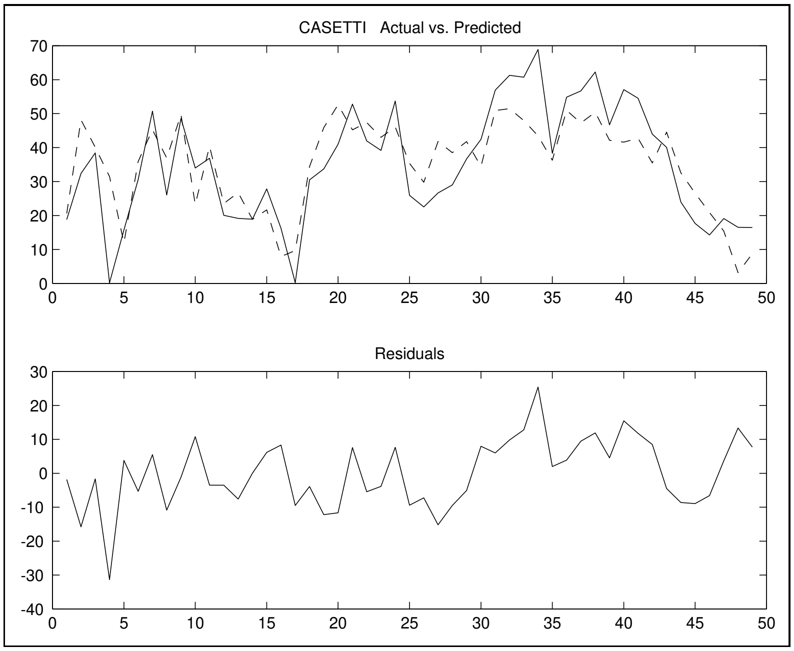

Figure 6.4: Actual versus Predicted and residuals

$$
\begin{aligned}
E(\varepsilon\varepsilon') &= \Phi = \sigma^2 \Psi \\
\Psi &= \exp(\mathrm{diag}(\gamma d_1, \gamma d_2, \ldots, \gamma d_n)) \\
\Phi &= \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2) \\
\sigma_i^2 &= \exp(\gamma_0 + \gamma_i d_i)
\end{aligned} \tag{6.13}
$$

Where d_i denotes the squared distance between the ith observation and the central point, and $\sigma^2, \gamma_0, \gamma_1$ are parameters to be estimated. Of course, a more general statement of the model would be that $\sigma_i^2 = g(h_i, \gamma)$, indicating that any functional form g involving some known variable h_i and associated parameters γ could be employed to specify the non-constant variance over space.

Note that the alternative specifications in (6.13) imply σ_i^2 has two components, a constant scalar component $\exp(\gamma_0)$ and $\exp(\gamma_i d_i)$ reflecting the non-constant component modeled as a function of distance from the central point.

An intuitive motivation for this type of variance structure is based on considering the nature of the composite disturbance: $Xu + e$. The constant scalar σ^2 reflects the constant component e, while the role of the scalar parameter γ_1 associated with distance measures the average impact of Xu, the non-constant variance component. Somewhat loosely, consider a linear regression involving the residuals on a constant term plus a vector of distances from the central place. The constant term estimate would reflect $\sigma^2 = \exp(\gamma_0)$, while the distance coefficient is intended to capture the influence of the non-constant Xu component: $\Psi = \exp(\gamma_1 d)$.

If $\gamma_1 = 0$, we have that $\Psi = \sigma^2 I_n$, a constant scalar value across all observations in space. This would be indicative of a situation where u, the error made in the expansion specification is small. This homoscedastic case indicates that a simple deterministic spatial expansion specification for spatial coefficient variation is performing well.

On the other hand, if $\gamma_1 > 0$, we find that moving away from the central point produces a positive increase in the variance. This is interpreted as evidence that 'parameter drift' is present in the relationship being modeled. The motivation is that increasing variance would be indicative of larger errors (u) in the expansion relationship as we move away from the central point.

Note that one can assume a value for γ_1 rather than estimate this parameter. If you impose a positive value, you are assuming a DARP model that will generate locally linear estimates since movement in the parameters will increase with movement away from the central point. This is because allowing increasing variance in the stochastic component of the expansion relation brings about more rapid change or adjustment in the parameters. Another way to view this is that the change in parameters need not adhere as strictly to the deterministic expansion specification. We will argue in Chapter 7 that a Bayesian approach to this type of specification is more intuitively appealing.

On the other hand, negative values for γ_1 suggest that the errors made by the deterministic expansion specification are smaller as we move away from the central point. This indicates that the expansion relation works well for points farther from the center, but not as well for the central area observations. Casetti and Can (1998) interpret this as suggesting "differential performance" of the base model with movement over space, and they label this phenomena as 'performance drift'. The intuition here is most likely based on the fact that the expansion relationship is of a locally linear nature. Given this, better performance with distance is indicative of a need to change the deterministic expansion relationship to improve performance. Again, I will argue that a Bayesian model represents a more intuitively appealing way to deal with these issues in the next chapter.

Estimation of the parameters of the model require either feasible generalized least squares (FGLS) or maximum likelihood (ML) methods. Feasible generalized least squares obtains a consistent estimate of the unknown parameter γ_1 and then proceeds to estimate the remaining parameters in the model conditional on this estimate.

As an example, consider using least-squares to estimate the expansion model and associated residuals, $\hat{e} = y - XDJ\hat{\beta}_0$. We could then carry out a regression of the form:

$$\log(\hat{e}^2) = \gamma_0 + \gamma_1 d + \nu \tag{6.14}$$

Casetti and Can (1998) argue that the estimate $\hat{\gamma}_1$ from this procedure would be consistent. Given the estimate $\hat{\gamma}_1$ and our knowledge of the distances in the vector d, we can construct an estimate of Ψ, which we designate $\hat{\Psi}$. We can then proceed with generalized least-squares to produce:

$$\begin{aligned}
\hat{\beta}_{FGLS} &= (X'\hat{\Psi}^1 X)^{-1} X'\hat{\Psi}^{-1} y \\
\hat{\sigma}^2_{FGLS} &= (y - X\hat{\beta}_{FGLS})\hat{\Psi}^{-1}(y - X\hat{\beta}_{FGLS})/(n-k)
\end{aligned}$$

Of course, the usual GLS variance-covariance matrix for the estimates applies:

$$var - cov(\hat{\beta}_{FGLS}) = \hat{\sigma}^2 (X'\hat{\Psi}^{-1}X)^{-1} \tag{6.15}$$

Casetti and Can (1998) also suggest using a statistic: $\hat{\gamma}_1^2/4.9348\sum d_i$ that is chi-squared distributed with one degree of freedom to test the null hypothesis that $\gamma_1 = 0$.

Maximum likelihood estimation involves using optimization routines to solve for a minimum of the negative of the log-likelihood function. We have already seen how to solve optimization problems in the context of spatial autoregressive models in Chapter 3. We will take the same approach for this model. The log-likelihood is:

$$L(\beta, \gamma_1 | y, X, d) = C - (1/2)ln|\sigma^2\Psi| - (1/2)(y - X\beta)'\Psi^{-1}(y - X\beta) \tag{6.16}$$

As in Chapter 3 we can construct a MATLAB function to evaluate the negative of this log-likelihood and rely on our function **maxlik**. The asymptotic variance-covariance matrix for the estimates β is equal to that for the FGLS estimates shown in (6.15). The asymptotic variance-covariance matrix for the parameters (σ^2, γ_1) is given by:

$$var - cov(\sigma^2, \gamma_1) = 2(D'D)^{-1} D = (\iota, d) \tag{6.17}$$

In the case of maximum likelihood estimates, a Wald statistic based on $\hat{\gamma}_1^2/2\sum d_i$ which is chi-squared distributed with one degree of freedom can be used to test the null hypothesis that $\gamma_1 = 0$. Note that the maximum likelihood estimate of γ_1 is more efficient than the FGLS estimate. This can be seen by comparing the ML estimate's asymptotic variance of $2\sum d_i$, to that for the FGLS which equals $4.9348\sum d_i$. Bear in mind, tests regarding the parameter γ_1 are quite often the focus of this methodology as it provides exploratory evidence regarding 'performance drift' versus 'parameter drift', so increased precision regarding this parameter may be important.

We have two functions to evaluate the negative of the log likelihood function for the DARP model, one for the case of an x-y expansion and another for distance expansion. The function used for distance expansion is shown below:

```
function llike = darp_lik2(parm,y,x,d)
% PURPOSE: evaluate the log-likelihood for the DARP model
```

```
%         based on distance expansion model
% ---------------------------------------------------
%  USAGE:llike = darp_lik2(parm,y,x,d)
%  where: parm  = a parameter vector containing:
%         parm(1,1)     = sige
%         parm(2,1)     = gamma
%         parm(3:k+2,1) = beta
%         y    = dependent variable vector
%         x    = explanatory variables matrix
%         d    = distance from central place
% ---------------------------------------------------
%  RETURNS: a  scalar equal to minus the log-likelihood
%           function value given the parameters
%  --------------------------------------------------
%  SEE ALSO: darp
% ---------------------------------------------------
[n k] = size(x); evar = length(parm);
sige = parm(1,1); if sige < 0.0001; sige = 0.0001; end;
gamma = parm(2,1); beta = parm(3:evar,1);
phi = exp(sige + gamma*d); phii = ones(n,1)./phi;
detphi = 0.5*log(prod(phi)); ys = sqrt(phii).*y;
% form expansion x-matrix
xt = x(:,2:k); xx = matmul(xt,d); xmat = [x xx];
xs = matmul(xmat,sqrt(phii));
e = (ys-xs*beta); epe = (e'*e);
llike =   0.5*epe + detphi;
```

The documentation for the function **darp** along with the portion of the function that implements DARP for distance expansion is shown below. The x-y expansion is similar.

```
function results=darp(y,x,xc,yc,option)
% PURPOSE: computes Casetti's DARP model
%---------------------------------------------------
% USAGE: results = darp(y,x,xc,yc,option)
% where:       y = dependent variable vector
%              x = independent variables matrix
%             xc = lattitude (or longitude) coordinate
%             yc = longitude (or lattitude) coordinate
%       option  = a structure variable containing options
%       option.exp  = 0 for x-y expansion (default)
%                   = 1 for distance from ctr expansion
%       option.ctr  = central point observation # for distance expansion
%       option.iter = # of iterations for maximum likelihood routine
%       option.norm = 1 for isotropic x-y normalization (default=0)
%---------------------------------------------------
% RETURNS:
%       results.meth   = 'darp'
%       results.b0     = bhat (underlying b0x, b0y)
%       results.t0     = t-stats (associated with b0x, b0y)
%       results.beta   = spatially expanded estimates (nobs x nvar)
%       results.yhat   = yhat
%       results.resid  = residuals
%       results.sige   = e'*e/(n-k)
%       results.rsqr   = rsquared
%       results.rbar   = rbar-squared
%       results.nobs   = nobs
```

```
%        results.nvar   = # of variables in x
%        results.y      = y data vector
%        results.xc     = xc
%        results.yc     = yc
%        results.ctr    = ctr (if input)
%        results.dist   = distance vector
%        results.exp    = exp input option
%        results.norm   = norm input option
%        results.iter   = # of maximum likelihood iterations
% ---------------------------------------------------
% NOTES: assumes x(:,1) contains a constant term
% ---------------------------------------------------
if nargin == 5 % user options
 if ~isstruct(option)
    error('darp: must supply the option argument as a structure variable');
 else, fields = fieldnames(option); nf = length(fields);
 expand = 0; ctr = 0; iter = 0; nflag = 0;
  for i=1:nf
    if strcmp(fields{i},'exp'),      expand = option.exp;
    elseif strcmp(fields{i},'ctr'),  ctr = option.ctr;
    elseif strcmp(fields{i},'iter'), iter = option.iter;
    elseif strcmp(fields{i},'norm'), nflag = option.norm;
    end;
  end; % end of for i
 end; % end of if else
elseif nargin == 4, expand = 0; option.exp = 0; iter = 0; nflag = 0;
else, error('Wrong # of arguments to darp');
end;
[nobs nvar] = size(x);
if x(:,1) ~= ones(nobs,1)
 error('darp: first column in x-matrix must be a constant vector');
end;
if nflag == 1, [xc yc] = normxy(xc,yc); end;
results.meth = 'darp'; results.y = y;   results.nobs = nobs;
results.nvar = nvar;   results.xc = xc; results.yc = yc;
results.option = option;results.exp = expand; results.norm = nflag;
switch expand
case{1} % distance from the center expansion
xi = xc(ctr); yi = yc(ctr);
% calculate distance weighting function
d = (xc-xi).*(xc-xi) + (yc-yi).*(yc-yi); dvec = d;
results.dist= dvec;
% transform x-variables using distance vector
xt = x(:,2:nvar); xx = matmul(xt,dvec);
xmat = [x xx]; b0 = xmat\y;  % get base model estimates
% find FGLS expansion estimates and  residuals
beta = zeros(nobs,(nvar-1)); yhat = zeros(nobs,1);
xx = matmul(ones(nobs,nvar-1),dvec); xxxy = [x xx];
tvar = length(b0);
    yhat(:,1) = xmat(:,1:nvar)*b0(1:nvar,1);
    for j=nvar+1:tvar
    beta(:,j-nvar) = xxxy(:,j)*b0(j,1);
    yhat(:,1) = yhat(:,1)  + xmat(:,j)*b0(j,1);
    end;
% use residuals to provide an initial set of FGLS estimates
e = y - yhat; e2 = log(e.*e);
% regress residuals on distance vector
```

```
n = length(e); res = ols(e2,[ones(n,1) dvec]);
% pull out sige, gamma1
sige = res.beta(1,1); gamma1 = res.beta(2,1);
% do FGLS
xt = x(:,2:nvar); xx = matmul(xt,dvec);
xmat = [x xx]; phi = exp(sige + gamma1*dvec); phii = ones(n,1)./phi;
ys = sqrt(phii).*y; xs = matmul(xmat,sqrt(phii));
b0 = xs\ys; % FGLS estimates
% do maximum likelihood estimation
if iter == 0, info.maxit = 500;
else, info.maxit = iter;
end;
parm = zeros(tvar+2,1); parm(1,1) = sige;
parm(2,1) = gamma1;     parm(3:tvar+2,1) = b0;
[b,llf,hessn,grad,iter,fail] = maxlik('darp_lik2',parm,info,y,x,dvec);
if fail == 1
    % warn user and rely on FGLS estimates
fprintf(1,'darp: convergence failure --- returning FGLS estimates\n');
results.b0 = b0; results.gamma(1) = gamma1;
results.iter = iter;  results.lik = 0; fflag = 1;
elseif iter == info.maxit
fprintf(1,'darp: no convergence in %d iterations\n',iter);
fprintf(1,'increase # of iterations -- FGLS estimates are being returned \n')
results.b0 = b0; results.gamma(1) = gamma1;
results.iter = iter;  results.lik = 0; fflag = 1;
elseif fail == 0, results.b0 = b(3:tvar+2,1); sige = b(1,1);
results.gamma(1) = b(2,1); results.iter = iter;  results.lik = llf;
gamma1 = b(2,1); fflag = 0;
end;
% find FGLS or ML expansion estimates and  residuals
beta = zeros(nobs,(nvar-1)); yhat = zeros(nobs,1);
xx = matmul(ones(nobs,nvar-1),dvec); xxxy = [x xx];
tvar = length(b0);
    yhat(:,1) = xmat(:,1:nvar)*b0(1:nvar,1);
    for j=nvar+1:tvar
    beta(:,j-nvar) = xxxy(:,j)*b0(j,1);
    yhat(:,1) = yhat(:,1)  + xmat(:,j)*b0(j,1);
    end;
results.beta = beta; results.yhat = yhat;
results.nvar = nvar; results.ctr = ctr;
results.resid = y - results.yhat;
sigu = results.resid'*results.resid;
results.sige = sigu/(n-tvar);
phi = exp(sige + gamma1*dvec); phii = ones(n,1)./phi;
xs = matmul(xmat,sqrt(phii));  xpxi = inv(xs'*xs);
tmp = sige*(diag(xpxi)); results.t0 = results.b0./(sqrt(tmp));
% compute chi-squared(1) statistic for gamma1
if fflag == 0, results.chi(1) = (gamma1*gamma1)/2*sum(dvec);
results.cprob(1) = 1-chis_prb(results.chi(1),1);
else, results.chi(1) = (gamma1*gamma1)/4.9348*sum(dvec);
results.cprob(1) = 1-chis_prb(results.chi(1),1);
end;
ym = y - mean(y); rsqr1 = sigu; rsqr2 = ym'*ym;
results.rsqr = 1.0 - rsqr1/rsqr2; % r-squared
rsqr1 = rsqr1/(nobs-2*nvar); rsqr2 = rsqr2/(nobs-1.0);
results.rbar = 1 - (rsqr1/rsqr2); % rbar-squared
otherwise
```

```
error('darp: check option input argument');
end;
```

Because we are relying on maximum likelihood estimation which may not converge, we provide FGLS estimates as output in the event of failure. A message is printed to the MATLAB command window indicating that this has occurred. We also rely on the FGLS estimates to provide starting values for the **maxlik** routine, which should speed up the optimization process.

The function also computes and returns probabilities for the chi-squared distributed statistics that are used to draw inferences regarding the parameters γ_1 in the case of distance expansion and both γ_1, γ_2 for x-y expansion.

The DARP model can be invoked with either the x-y or distance expansion as in the case of the spatial expansion model. Specifically, for x-y expansion the variance specification is based on:

$$\log(\hat{e}^2) = \gamma_0 + \gamma_1 xc + \gamma_2 yc + \nu \tag{6.18}$$

This generalizes the distance expansion approach presented in the text. Of course, we have an accompanying **prt** and **plt** function to provide printed and graphical presentation of the estimation results.

Example 6.2 shows how to use the function **darp** for both x-y and distance expansion using the Columbus neighborhood crime data set.

```
% ----- Example 6.2 Using the darp() function
% load Anselin (1988) Columbus neighborhood crime data
load anselin.dat; y = anselin(:,1); n = length(y);
x = [ones(n,1) anselin(:,2:3)];
xc = anselin(:,4); yc = anselin(:,5); % Anselin  x-y coordinates
vnames = strvcat('crime','const','income','hse value');
% do Casetti darp using x-y expansion
res1 = darp(y,x,xc,yc);
prt(res1,vnames); % print the output
plt(res1,vnames); % plot the output
pause;
% do Casetti darp using distance expansion from observation #32
option.exp = 1; option.ctr = 32;
res2 = darp(y,x,xc,yc,option);
prt(res2,vnames); % print the output
plt(res2,vnames); % plot the output
```

The printed results are shown below, where we report not only the estimates for β_0 and the first five expansion estimates, but estimates for the parameters γ as well. A chi-squared statistic to test the null hypothesis that $\gamma_1 = 0$ is provided as well as a marginal probability level. For the case of the x-y expansion, we see that γ_1 parameter is negative and significant by virtue of the large chi-squared statistic and associated marginal probability level of 0.0121. The inference we would draw is that performance drift occurs the south-north direction. For the γ_2 parameter, we find a positive value that is not significantly different from zero because of the marginal probability level of 0.8974. This indicates that the simple deterministic expansion relationship is working well in the west-east direction. Note that these results conform to those found with the spatial expansion model, where we indicated that parameter variation in the west-east direction was significant, but not for the south-north.

```
DARP X-Y Spatial Expansion Estimates
Dependent Variable =        crime
R-squared        =    0.6180
Rbar-squared     =    0.5634
sige             =  122.2255
gam1,chi(1),prob =   -0.0807,   6.2924,   0.0121
gam2,chi(1),prob =    0.0046,   0.0166,   0.8974
# of iterations  =     16
log-likelihood   =       -181.3901
Nobs, Nvars      =     49,     3
***************************************************************
Base x-y estimates
Variable         Coefficient      t-statistic    t-probability
const              66.783527         6.024676         0.000000
income             -2.639184        -0.399136         0.691640
hse value           0.249214         0.095822         0.924078
x-income           -0.048337        -0.537889         0.593247
x-hse value         0.021506         0.640820         0.524819
y-income            0.084877         0.564810         0.574947
y-hse value        -0.037460        -0.619817         0.538436
***************************************************************
Expansion estimates
 Obs#    x-income x-hse value    y-income y-hse value
    1     -1.3454      0.6595      4.0747     -1.6828
    2     -1.3786      0.6758      3.8959     -1.6089
    3     -1.3865      0.6796      3.7218     -1.5370
    4     -1.2600      0.6176      3.6930     -1.5251
    5     -1.4655      0.7183      4.2372     -1.7499

DARP Distance Expansion Estimates
Dependent Variable =        crime
R-squared         =    0.6142
Rbar-squared      =    0.5693
sige              =  117.8286
gamma,chi(1),prob =   -0.0101,   0.1603,   0.6889
# of iterations   =     10
log-likelihood    =       -137.10461
Nobs, Nvars       =     49,     3
central obs       =     32
***************************************************************
Base centroid estimates
Variable         Coefficient      t-statistic    t-probability
const              58.914770         5.499457         0.000002
income             -0.712425        -0.717969         0.476407
hse value          -0.303043        -0.922558         0.361051
d-income           -0.005250        -0.874035         0.386639
d-hse value         0.000899         0.432865         0.667135
***************************************************************
Expansion estimates
 Obs#      income   hse value
    1     -0.6848      0.0980
    2     -0.4372      0.0626
    3     -0.2694      0.0386
    4     -0.4374      0.0626
    5     -0.8247      0.1180
```

For the case of the distance expansion we find a single γ parameter that is negative but insignificant. This would be interpreted to mean that the deterministic expansion relationship is not suffering from performance drift over space.

A comparison of the base model estimates from **darp** versus those from **casetti** show relatively similar coefficient estimates as we would expect. In the case of the x-y expansion all of the signs are the same for spatial expansion and darp models. The distance expansion does exhibit a sign change for the coefficient on income, which goes from positive in the expansion model to negative in the darp model. Correcting for the heteroscedastic character of the estimation problem produces dramatic changes in the statistical significance found for the base model estimates. They all become insignificant, a finding consistent with results reported by Anselin (1988) based on a jacknife approach to correcting for heteroscedasticity in this model. (Anselin finds that the income coefficient is still marginally significant after the correction.)

One approach to using this model is to expand around every point in space and examine the parameters γ for evidence that indicates where the model is suffering from performance or parameter drift. Example 4.3 shows how this might be accomplished using a 'for loop' over all observations. For this purpose, we wish to recover only the estimated values for the parameter γ along with the marginal probability levels.

```
% ----- Example 6.3 Using darp() over space
% load Anselin (1988) Columbus neighborhood crime data
load anselin.dat; y = anselin(:,1); n = length(y);
x = [ones(n,1) anselin(:,2:3)];
xc = anselin(:,4); yc = anselin(:,5); % Anselin  x-y coordinates
vnames = strvcat('crime','const','income','hse value');
% do Casetti darp using distance expansion from all
% observations in the data sample
option.exp = 1;
output = zeros(n,2);
tic;
for i=1:n % loop over all observations
 option.ctr = i;
res = darp(y,x,xc,yc,option);
output(i,1) = res.gamma(1);
output(i,2) = res.cprob(1);
end;
toc;
in.cnames = strvcat('gamma estimate','marginal probability');
in.rflag = 1;
mprint(output,in)
```

We use the MATLAB 'tic' and 'toc' commands to time the operation of producing maximum likelihood estimates across the entire sample. The results are shown in Table 6.1. It took 70 seconds to solve the maximum likelihood estimation problem 49 times, calculate expansion estimates and produce all of the ancillary statistics.

From the estimated values of γ and the associated marginal probabilities, we infer that the model suffers from performance drift over the initial 9 obser-

Table 6.1: DARP model results for all observations

Obs#	$\hat{\gamma}$	probability	Obs#	$\hat{\gamma}$	probability
1	-0.2198	0.0714	26	0.1142	0.4264
2	-0.2718	0.0494	27	0.1150	0.4618
3	-0.3449	0.0255	28	0.0925	0.5584
4	-0.4091	0.0033	29	0.1070	0.5329
5	-0.2223	0.0532	30	-0.2765	0.1349
6	-0.3040	0.0266	31	0.0453	0.8168
7	-0.4154	0.0126	32	-0.6580	0.0012
8	-0.2071	0.1477	33	-0.3293	0.0987
9	-0.5773	0.0030	34	-0.5949	0.0024
10	0.1788	0.1843	35	-0.8133	0.0000
11	0.1896	0.1526	36	-0.5931	0.0023
12	0.1765	0.1621	37	-0.4853	0.0066
13	0.1544	0.1999	38	-0.4523	0.0121
14	0.1334	0.2214	39	-0.5355	0.0016
15	0.1147	0.2708	40	-0.6050	0.0005
16	0.1429	0.2615	41	-0.6804	0.0001
17	0.1924	0.2023	42	-0.7257	0.0001
18	-0.1720	0.3112	43	-0.7701	0.0000
19	0.1589	0.3825	44	-0.5150	0.0001
20	-0.3471	0.0810	45	-0.3997	0.0005
21	0.2020	0.2546	46	-0.3923	0.0003
22	0.1862	0.2809	47	-0.3214	0.0004
23	0.1645	0.3334	48	-0.3586	0.0001
24	0.0904	0.6219	49	-0.4668	0.0000
25	0.1341	0.4026			

vations and observations 32 to 49. We draw this inference from the negative γ estimates that are statistically significant for these observations. (Note that observation #8 is only marginally significant.) Over the middle range of the sample, from observations 10 to 31 we find that the deterministic distance expansion relationship works well. This inference arises from the fact that none of the estimated γ parameters are significantly different from zero.

In the next chapter we will provide evidence that observations 2, 4 and 34 represent potential outliers whose influence extends over neighboring observations in the range of observations 1 to 9 and 31 to 45. The ability of outliers to impact an entire sub-sequence of the estimates when using locally linear regression methods is taken up in detail. These related findings suggest that the DARP model is performing well in detecting these aberrant observations.

6.3 Non-parametric locally linear models

These models represent an attempt to draw on the flexibility and tractability of non-parametric estimators. It is generally believed that these methods hold a computational advantage over spatial autoregressive models because large matrix manipulations or inversion of large sparse matrices are not required to produce estimates. We present evidence contrary to this belief with applied examples based on the large Pace and Barry (1997) data set.

Locally linear regression methods introduced in McMillen (1996,1997) and labeled geographically weighted regressions (GWR) in Brunsdon, Fotheringham and Charlton (1996) (BFC hereafter) are discussed in this section. The main contribution of the GWR methodology is the use of distance-weighted sub-samples of the data to produce locally linear regression estimates for every point in space. Each set of parameter estimates is based on a distance-weighted sub-sample of "neighboring observations", which has a great deal of intuitive appeal in spatial econometrics. While this approach has a definite appeal, it also presents some problems and a Bayesian approach to these models that corrects these problems is set forth in the next chapter.

The distance-based weights used by BFC for data at observation i take the form of a vector W_i determined using a vector of distances, d_i between observation i and all other observations in the sample. This distance vector along with a distance decay parameter are used to construct a weighting function that places relatively more weight on neighboring sample observations from the spatial data sample.

A host of alternative approaches have been suggested for constructing the weight function. One approach suggested by BFC is:

$$W_i = \sqrt{\exp(-d_i/\theta)} \tag{6.19}$$

The parameter θ is a decay parameter that BFC label "bandwidth". Changing the bandwidth results in a different exponential decay profile, which in turn produces estimates that vary more or less rapidly over space.

Another weighting scheme is the tri-cube function proposed by McMillen (1998):

$$W_i = (1 - (d_i/q_i)^3)^3 \quad \mathrm{I}(d_i < q_i) \tag{6.20}$$

Where q_i represents the distance of the qth nearest neighbor to observation i and I() is an indicator function that equals one when the condition is true and zero otherwise. Still another approach is to rely on a Gaussian function ϕ:

$$W_i = \phi(d_i/\sigma\theta) \tag{6.21}$$

Where ϕ denotes the standard normal density and σ represents the standard deviation of the distance vector d_i.

The notation used here may be confusing since we usually rely on subscripted variables to denote scalar elements of a vector. Our notation uses the subscripted variable d_i to represent a vector of distances between observation i and all other sample data observations.

BFC use a single value of θ, the bandwidth parameter for all observations. This value is determined using a cross-validation procedure often used in locally linear regression methods. Cross-validation relies on a score function taking the form shown in (6.22) to determine a value for θ.

$$\sum_{i=1}^{n}[y_i - \hat{y}_{\neq i}(\theta)]^2 \tag{6.22}$$

Where $\hat{y}_{\neq i}(\theta)$ denotes the fitted value of y_i with the observations for point i omitted from the calibration process. A value of θ that minimizes this score function is used as the distance-weighting bandwidth to produce GWR estimates. Note that for the case of the tri-cube weighting function, we would compute a value for q denoting the number of nearest neighbors beyond which we impose zero weights. The score function would be evaluated using alternative values of q to find a value that minimizes the function.

The non-parametric GWR model relies on a sequence of locally linear regressions to produce estimates for every point in space by using a sub-sample of data information from nearby observations. Let y denote an nx1 vector of dependent variable observations collected at n points in space, X an nxk matrix of explanatory variables, and ε an nx1 vector of normally distributed, constant variance disturbances. Letting W_i represent an nxn diagonal matrix containing distance-based weights for observation i that reflect the distance between observation i and all other observations, we can write the GWR model as:

$$W_i^{1/2}y = W_i^{1/2}X\beta_i + W_i^{1/2}\varepsilon_i \tag{6.23}$$

The subscript i on β_i indicates that this kx1 parameter vector is associated with observation i. The GWR model produces n such vectors of parameter estimates, one for each observation. These estimates are calculated using:

$$\hat{\beta}_i = (X'W_iX)^{-1}(X'W_iy) \tag{6.24}$$

Keep in mind the confusing notation, $W_i^{1/2}y$ denotes an n-vector of distance-weighted observations used to produce estimates for observation i. Note also, that $W_i^{1/2}X$ represents a distance-weighted data matrix, not a single observation and $W_i^{1/2}\varepsilon_i$ represents an n-vector.

Note that these GWR estimates for β_i are conditional on the bandwidth parameter θ or q the number of neighbors we select in the case of the tri-cube weighting function. That is, changing θ (or q), will produce a different set of GWR estimates.

6.3.1 Implementing GWR

The first step in computing GWR estimates is to find an optimal bandwidth θ or q-value that minimizes the score function. We construct two MATLAB functions, one to compute the scores associated with different bandwidths and another to determine scores associated with different $q-$values, where q represents the number of nearest neighbors in the tri-cube weighting method. These univariate functions can be minimized using the MATLAB simplex algorithm **fmin** to find an optimal bandwidth or q value.

The function **scoref** is used for exponential and Gaussian weights to determine a value for θ, and the function **scoreq** is used to determine q, the number of nearest neighbors in the tri-cube weighting function.

```
function score = scoref(bdwt,y,x,east,north,flag)
% PURPOSE: evaluates cross-validation score to determine optimal
%          bandwidth for gwr with gauusian or exponential weighting
% ------------------------------------------------------
% USAGE: score = scoref(y,x,east,north,bdwt,flag);
% where: y = dependent variable
%        x = matrix of explanatory variables
%     east = longitude (x-direction) coordinates
%    north = lattitude (y-direction) coordinates
%     bdwt = a bandwidth to use in computing the score
%     flag = 0 for Gaussian weights
%          = 1 for BFG exponential
% ------------------------------------------------------
% RETURNS: score = a cross-validation criterion
% ------------------------------------------------------
[n k] = size(x); res = zeros(n,1); wt = zeros(n,1); d = zeros(n,1);
for iter = 1:n;
   dx = east - east(iter,1);   dy = north - north(iter,1);
   d = (dx.*dx + dy.*dy);  sd = std(sqrt(d));
   if flag == 0,     % Gausian weights
     wt = sqrt(stdn_pdf(sqrt(d)/(sd*bdwt)));
   elseif flag == 1, % exponential weights
     wt = sqrt(exp(-d/bdwt));
   end;
   wt(iter,1) = 0.0;
% computational trick to speed things up
nzip = find(wt >= 0.01); % pull out non-zero obs
ys = y(nzip,1).*sqrt(wt(nzip,1));
xs = matmul(x(nzip,:),sqrt(wt(nzip,1)));
bi = xs\ys;
yhat = x(iter,:)*bi;     % compute predicted values
res(iter,1) = y(iter,1) - yhat; % compute residuals
end; % end of for iter loop
tmp = res'*res; score = sqrt(tmp/n);
```

We rely on some computational tricks in **scoref** to improve the speed and reduce memory requirements. The distance computations are "vectorized' so we generate an entire vector of weights d rather than use a 'for loop' over the elements of the vector. This dramatically increases the speed because MATLAB is relatively slow in executing 'for loops'. A second trick is to calculate the regression based only on observations associated with non-negligible weight elements. In a large data set it may be the case that the weight assigned to most observations is near zero and these observations can be excluded from the regression. The code to accomplish this trick is:

```
nzip = find(wt >= 0.01); % pull out non-zero obs
ys = y(nzip,1).*sqrt(wt(nzip,1));
xs = matmul(x(nzip,:),sqrt(wt(nzip,1)));
```

Non-zero weights are somewhat arbitrary in the case of Gaussian and exponential weighting methods. Consider that the Gaussian function assigns positive weight to everything between $-\infty$ and $+\infty$, but we only consider weights taking on values greater than 0.01 as non-zero. Since the maximum weight will be unity in the case of exponential weighting because exp(0) equals unity, this cut-off value for non-negligible weights seems appropriate.

The function **scoreq** is a bit more complicated because we need to carry out a grid search over integer values of q, the number of nearest neighbors to use.

```
function q = scoreq(qmin,qmax,y,x,east,north)
% PURPOSE: evaluates cross-validation score to determine optimal
%          q for gwr based on tricube weighting
% ------------------------------------------------------
% USAGE: score = scoreq(qmin,qmax,y,x,east,north);
% where: qmin = minimum # nearest neighbors to use in CV search
%        qmax = maximum # nearest neighbors to use in CV search
%        y    = dependent variable
%        x = matrix of explanatory variables
%     east = longitude (x-direction) coordinates
%    north = lattitude (y-direction) coordinates
% ------------------------------------------------------
% RETURNS: q = # of nearest neighbors that minimum the score
%              function
% ------------------------------------------------------
[n k] = size(x); res = zeros(n,1); d = zeros(n,1);
qgrid = qmin:qmax; nq = length(qgrid); wt = zeros(n,nq);
for iter = 1:n;
dx = east - east(iter,1); dy = north - north(iter,1);
d = (dx.*dx + dy.*dy);
    % sort distance to find q nearest neighbors
    ds = sort(d); dmax = ds(qmin:qmax,1);
    for j=1:nq;
      nzip = find(d <= dmax(j,1));
      wt(nzip,j) = (1-(d(nzip,1)/dmax(j,1)).^3).^3;
      wt(iter,j) = 0.0;
    end; % end of j loop
for j=1:nq;
% computational trick to speed things up
% use wt to pull out non-zero observations
nzip = find(wt(:,j) > 0);
ys = y(nzip,1).*sqrt(wt(nzip,j));
xs = matmul(x(nzip,:),sqrt(wt(nzip,j)));
bi = xs\ys;
yhat = x(iter,:)*bi; % compute predicted values
res(iter,j) = y(iter,1) - yhat; % compute residuals
end; % end of for j loop over q-values
end; % end of for iter loop
tmp = res.*res; score = sum(tmp);
[smin sind] = min(score); q = qgrid(sind);
```

In the case of the tri-cube weighting function, zero weights are assigned to neighbors at a greater distance than the qth neighbor. This means that zero weights are well-defined for this case and we modify the code for finding non-zero weights and associated y, X observations as follows:

```
nzip = find(wt(:,j) > 0);
ys = y(nzip,1).*sqrt(wt(nzip,j));
xs = matmul(x(nzip,:),sqrt(wt(nzip,j)));
```

The function sorts distances and finds a vector of maximum distances associated with the grid of q values being searched. A weight matrix is then generated for the entire grid of q values and regressions are carried out for every value of q in the grid. After collecting a matrix of scores, the minimum is found and the associated q value that produced the minimum score is returned by the function.

Both **scoref** and **scoreq** implement the weighted least-squares regressions using y and X pre-multiplied by $W_i^{1/2}$, which conserves on memory compared to the mathematical notion that is often used:

$$\beta_i = (X'W_iX)^{-1}(X'W_iy) \tag{6.25}$$

Computing expression (6.25) requires a diagonal matrix W_i that uses a great deal more RAM memory.

Another aspect of **scoref** and **scoreq** is that they exclude the ith observation in the sample when producing an estimate β_i for observation i by setting the weight vector element 'wt' to zero. This is required for cross-validation computation of the score function as indicated in the discussion surrounding (6.22).

Given the optimal bandwidth or q value indicating the optimal number of nearest neighbors, estimation of the GWR parameters β and associated statistics can proceed via generalized least-squares. The function **gwr** that produces these estimates is shown below.

```
function result = gwr(y,x,east,north,info);
% PURPOSE: compute geographically weighted regression
%----------------------------------------------------
% USAGE: results = gwr(y,x,east,north,info)
% where:   y = dependent variable vector
%          x = explanatory variable matrix
%       east = x-coordinates in space
%      north = y-coordinates in space
%       info = a structure variable with fields:
%       info.bwidth = scalar bandwidth to use or zero
%                     for cross-validation estimation (default)
%       info.dtype  = 'gaussian'    for Gaussian weighting (default)
%                   = 'exponential' for exponential weighting
%                   = 'tricube'     for tri-cube weighting
%       info.q      = q-nearest neighbors to use for tri-cube weights
%                     (default: CV estimated)
%       info.qmin   = minimum # of neighbors to use in CV search
%       info.qmax   = maximum # of neighbors to use in CV search
%                     defaults: qmin = nvar+2, qmax = 5*nvar
% ---------------------------------------------------
%  NOTE: res = gwr(y,x,east,north) does CV estimation of bandwidth
% ---------------------------------------------------
% RETURNS: a results structure
%        results.meth   = 'gwr'
%        results.beta   = bhat matrix    (nobs x nvar)
```

```
%        results.tstat  = t-stats matrix (nobs x nvar)
%        results.yhat   = yhat
%        results.resid  = residuals
%        results.sige   = e'e/(n-dof) (nobs x 1)
%        results.nobs   = nobs
%        results.nvar   = nvars
%        results.bwidth = bandwidth for gaussian or exponential weights
%        results.q      = q nearest neighbors for tri-cube weights
%        results.dtype  = input string for Gaussian, exponential weights
%        results.iter   = # of simplex iterations for cv
%        results.north  = north (y-coordinates)
%        results.east   = east  (x-coordinates)
%        results.y      = y data vector
%---------------------------------------------------
if nargin == 5 % user options
 if ~isstruct(info)
    error('gwr: must supply the option argument as a structure variable');
 else, fields = fieldnames(info); nf = length(fields);
 [n k] = size(x); bwidth = 0; dtype = 0; q = 0; qmin = k+2; qmax = 5*k;
  for i=1:nf
    if strcmp(fields{i},'bwidth'),    bwidth = info.bwidth;
    elseif strcmp(fields{i},'dtype'), dstring = info.dtype;
       if strcmp(dstring,'gaussian'),       dtype = 0;
       elseif strcmp(dstring,'exponential'), dtype = 1;
       elseif strcmp(dstring,'tricube'),     dtype = 2;
       end;
    elseif strcmp(fields{i},'q'),     q = info.q;
    elseif strcmp(fields{i},'qmax'),  qmax = info.qmax;
    elseif strcmp(fields{i},'qmin'),  qmin = info.qmin;
    end;
  end; % end of for i
 end; % end of if else
elseif nargin == 4, bwidth = 0; dtype = 0; dstring = 'gaussian';
else, error('Wrong # of arguments to gwr');
end;
result.north = north; result.east = east;
switch dtype
case{0,1}      % set bandwidth
if bwidth == 0 % cross-validation determination
options = zeros(1,18); options(1,1) = 0; options(1,2) = 1.e-3; options(14) = 500;
if dtype == 0     % Gaussian weights
[bdwt options] = fmin('scoref',0.1,10,options,y,x,east,north,dtype);
elseif dtype == 1 % exponential weights
[bdwt options] = fmin('scoref',0.25,20,options,y,x,east,north,dtype);
end;
 if options(10) == options(14),
 fprintf(1,'gwr: cv convergence not obtained in %4d iterations',options(14));
 else, result.iter = options(10);
 end;
else, bdwt = bwidth*bwidth; % user supplied bandwidth
end;
case{2} % set q-nearest neigbhor
if q == 0 % cross-validation determination
q = scoreq(qmin,qmax,y,x,east,north);
else,     % rely on user-supplied q-value
end;
otherwise
```

```
end;
% do GWR using bdwt as bandwidth
[n k] = size(x);    bsave = zeros(n,k); ssave = zeros(n,k);
sigv  = zeros(n,1); yhat  = zeros(n,1); resid = zeros(n,1);
wt = zeros(n,1);    d = zeros(n,1);
for iter=1:n;
    dx = east - east(iter,1);
    dy = north - north(iter,1);
    d = (dx.*dx + dy.*dy);
    sd = std(sqrt(d));
    % sort distance to find q nearest neighbors
    ds = sort(d);
    if dtype == 2, dmax = ds(q,1); end;
       if dtype == 0,     % Gausian weights
        wt = stdn_pdf(sqrt(d)/(sd*bdwt));
       elseif dtype == 1, % exponential weights
        wt = exp(-d/bdwt);
       elseif dtype == 2, % tricube weights
wt = zeros(n,1);
nzip = find(d <= dmax);
        wt(nzip,1) = (1-(d(nzip,1)/dmax).^3).^3;
       end; % end of if,else
% computational trick to speed things up
% use non-zero wt to pull out y,x observations
nzip = find(wt >= 0.01); ys = y(nzip,1).*sqrt(wt(nzip,1));
xs = matmul(x(nzip,:),sqrt(wt(nzip,1)));
xpxi = inv(xs'*xs); b = xpxi*xs'*ys;
yhatv = xs*b; % compute predicted values
yhat(iter,1) = x(iter,:)*b;
resid(iter,1) = y(iter,1) - yhat(iter,1);
e = ys - yhatv; % compute residuals
nadj = length(nzip); sige = (e'*e)/nadj;
sdb = sqrt(sige*diag(xpxi)); % compute t-statistics
% store coefficient estimates and std errors in matrices
% one set of beta,std for each observation
bsave(iter,:) = b'; ssave(iter,:) = sdb';  sigv(iter,1) = sige;
end;
% fill-in results structure
result.meth = 'gwr'; result.nobs = nobs; result.nvar = nvar;
if (dtype == 0 | dtype == 1), result.bwidth = sqrt(bdwt);
else, result.q = q;
end;
result.beta = bsave; result.tstat = bsave./ssave;
result.sige = sigv;  result.dtype = dstring;
result.y = y;        result.yhat = yhat;
% compute residuals and conventional r-squared
result.resid = resid;
sigu = result.resid'*result.resid;
ym = y - mean(y); rsqr1 = sigu; rsqr2 = ym'*ym;
result.rsqr = 1.0 - rsqr1/rsqr2; % r-squared
rsqr1 = rsqr1/(nobs-nvar); rsqr2 = rsqr2/(nobs-1.0);
result.rbar = 1 - (rsqr1/rsqr2); % rbar-squared
```

Some important points about the function that might affect users are that the default range of values searched for an optimal q value is $k+2$ to $5k$, where k is the number of variables in the X matrix. We need to use at least $k+1$ parameters to carry out a regression, and using $k+1$ produced some problems with inversion of $X'X$, so $k+2$ was chosen. This seems to work in practice without matrix inversion problems. An input option allows users to specify 'info.qmin' and 'info.qmax' to define the range used in the search. A grid search is done which is relatively inefficient, so selection of a wide range of q values for the search will slow down the estimation process. If the optimal value of q reported in the printed output is at the upper limit of $5k$, you should use the input options 'info.qmin' and 'info.qmax' and produce another set of estimates.

There are also ranges set for the simplex optimization function **fmin** when searching for an optimal bandwidth parameter in the case of Gaussian or exponential weighting methods. The range used for Gaussian weighting is 0.1 to 10 and for the exponential weighting 0.25 to 20. No input options are provided for these ranges which appear to work well in applied situations. Nonetheless, knowledge of these limits is helpful when examining the optimal bandwidth printed in the output. If the optimal bandwidth falls near the upper limits, it may suggest that a constant parameter regression model is more appropriate than a locally linear model.

This function also relies on the computational tricks for conserving memory and speed, using observations associated with non-negligible weights. Another point to note regarding the function is that an estimate of σ^2 for each observation is produced based on the number of non-zero observations contained in the variable 'nadj' as a divisor for the sum of squared residuals. This divisor is different from BFC where n minus the trace of the matrix $X'W_iX$ is used, so their estimates may differ from those produced by the **gwr** function.

6.3.2 Applied examples

Example 6.4 demonstrates using the function **gwr** on Anselin's Columbus neighborhood crime data set with all three weighting methods. The estimates are plotted by the program with the graph shown in Figure 6.5.

```
% ----- Example 6.4 Using the gwr() function
% load the Anselin data set
load anselin.dat; y = anselin(:,1); nobs = length(y);
x = [ones(nobs,1) anselin(:,2) anselin(:,3)];
[n k] = size(x); north = anselin(:,4); east = anselin(:,5);
vnames = strvcat('crime','constant','income','hvalue');
info.dtype = 'gaussian';    % gaussian distance weighting
tic; result1 = gwr(y,x,east,north,info); toc;
info.dtype = 'exponential'; % exponential distance weighting
tic; result2 = gwr(y,x,east,north,info); toc;
info.dtype = 'tricube';     % tri-cube distance weighting
tic; result3 = gwr(y,x,east,north,info); toc;
% plot results for comparison
tt=1:nobs;
subplot(3,1,1),
plot(tt,result1.beta(:,1),tt,result2.beta(:,1),'--',tt,result3.beta(:,1),'-.');
ylabel('Constant term');
subplot(3,1,2),
plot(tt,result1.beta(:,2),tt,result2.beta(:,2),'--',tt,result3.beta(:,2),'-.');
ylabel('Household income');
```

```
subplot(3,1,3),
plot(tt,result1.beta(:,3),tt,result2.beta(:,3),'--',tt,result3.beta(:,3),'-.');
ylabel('House value');
```

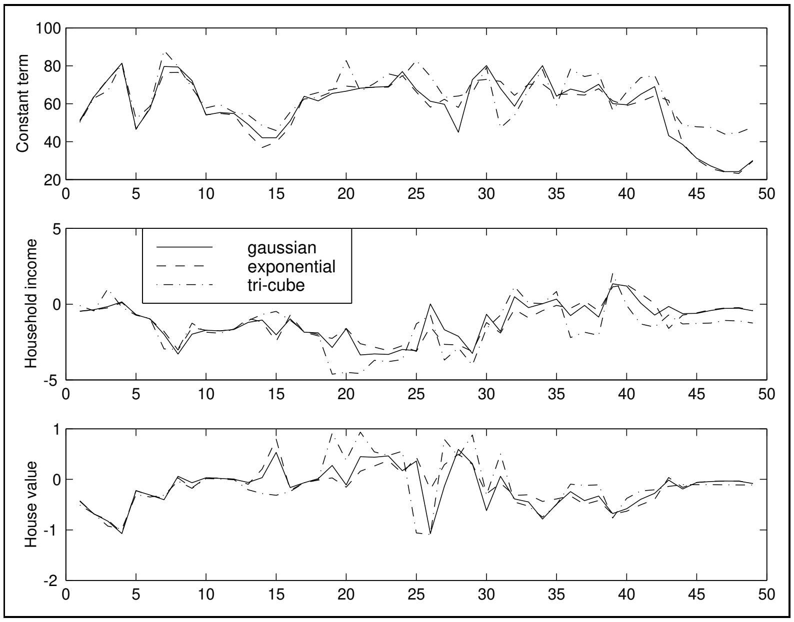

Figure 6.5: GWR estimates

From the graph we see that alternative weighting methods have some impact on the resulting estimates. Printed results from these models can be voluminous, so we present estimates for only a handful of observations below.

It might be of interest that our work in programming for speed appears to have paid off as the execution times required for the three models in example 6.4 were 4.16, 2.48 and 1.56 seconds respectively. The nature of our trick that operates only on observations with non-negligible weights means that even in very large samples the actual work involved in computing the estimates may involve only small matrix inversions. Execution times will of course slow down as the number of observations increase because more of these small matrix inversions will be required.

```
Geometrically weighted regression estimates
Dependent Variable =         crime
R-squared      =    0.9417
Rbar-squared   =    0.9392
Bandwidth      =    0.6519
# iterations   =     15
Decay type     =     gaussian
Nobs, Nvars    =     49,     3
***************************************
```

```
Obs =    1, x-coordinate= 42.3800, y-coordinate= 35.6200, sige=  1.4026
Variable      Coefficient      t-statistic    t-probability
constant        51.208258        14.379087         0.000000
hvalue          -0.434424        -5.767336         0.000001
income          -0.460959        -2.618684         0.011713

Obs =    2, x-coordinate= 40.5200, y-coordinate= 36.5000, sige=  3.2344
Variable      Coefficient      t-statistic    t-probability
constant        63.569815        14.426028         0.000000
hvalue          -0.683569        -6.747911         0.000000
income          -0.370017        -1.436461         0.157226

Obs =    3, x-coordinate= 38.7100, y-coordinate= 36.7100, sige=  4.6286
Variable      Coefficient      t-statistic    t-probability
constant        72.673700        12.846078         0.000000
hvalue          -0.826778        -7.338805         0.000000
income          -0.161602        -0.369296         0.713498
```

6.4 Applied exercises

We produce GWR estimates for the Pace and Barry (1997) data set as an applied illustration. Example 6.5 estimates the same model that we used in Chapters 4 and 5 for the spatial autoregressive methods. One difference is that we use latitude and longitude coordinates (in the 5th and 6th columns of the data file) needed for the GWR method.

The example program sorts by longitude from low to high so we can produce a graph of the estimates over space moving from the western to eastern U.S. Of course, sorting the data has no impact on the estimates, because regression estimates are unchanged by a re-ordering the observations.

```
% ----- Example 6.5 GWR on the Pace and Barry data set
load elect.dat;            % load data on votes in 3,107 counties
north = elect(:,5); east = elect(:,6);
% sort the data from west to east
[easts ind] = sort(east); norths = north(ind,1);
elects = elect(ind,:); nobs = 3107;
clear elect; % save RAM memory
y =  log(elects(:,7)./elects(:,8)); % convert to per capita variables
x1 = log(elects(:,9)./elects(:,8)); % education
x2 = log(elects(:,10)./elects(:,8));% homeownership
x3 = log(elects(:,11)./elects(:,8));% income
n = length(y); x = [ones(n,1) x1 x2 x3]; clear elects x1 x2 x3;
vnames = strvcat('votes','constant','education','homeowners','income');
info.dtype = 'gaussian';
info.bwidth = 0.3528;
tic; result = gwr(y,x,easts,norths,info); toc;
% plot estimates from west to east
subplot(2,2,1),
plot(easts/1000000,result.beta(:,1),'.k');
ylabel('Constant term');
xlabel('longitude   west-east');
subplot(2,2,2),
plot(easts/1000000,result.beta(:,2),'.k');
ylabel('education');
```

```
xlabel('longitude   west-east');
subplot(2,2,3),
plot(easts/1000000,result.beta(:,3),'.k');
ylabel('homeowners');
xlabel('longitude   west-east');
subplot(2,2,4),
plot(easts/1000000,result.beta(:,4),'.k');
ylabel('income');
xlabel('longitude   west-east');
```

Despite our efforts to code the **gwr** function for speed, it took 47 minutes to produce estimates based on a Gaussian weighting scheme. The number of iterations required to find the optimal bandwidth using the simplex search was 15, so we averaged over 3 minutes for one pass through the observations to produce a set of 3,107 estimates. Using the optimal bandwidth of 0.3511 as an input to the **gwr** function eliminates the need for the simplex search, producing estimates in 239 seconds.

One way to speed up the estimation procedure would be to change the tolerance used by the simplex optimization function **fmin**. This is set to 1e-3, in the function **gwr**. My experience indicates that the estimates are not very sensitive to changes in the third decimal place of the bandwidth parameter, making this a feasible option. Another way to speed up estimation without sacrificing accuracy is to place a narrower range on the simplex search. In large problems, a sub-sample might be used to estimate a bandwidth parameter that would allow an intelligent range to be set. If you encounter a bandwidth estimate near the boundary of your range, you should then adjust the range and produce another set of estimates.

Somewhat worse timing results were found for the case of the tri-cube weighting where it took 69 minutes to find an optimal $q = 19$ nearest neighbors. Again, given a value of q, the function can produce estimates in around 200 seconds. We could have reduced the estimation time by supplying inputs for the grid search range over alternative q values. The optimal q was found to be 19, which was near the upper limit of $5k = 20$ in this problem. Another strategy would be to attempt a more intelligent approach than the grid search over alternative values of q. We will provide some additional timing results for the Boston data set containing 506 observations shortly.

A graph of the four parameters as they vary over space from west to east is shown in Figure 6.6. This set of estimates was produced using the Gaussian weighting method and the estimated bandwidth of 0.3511 reported in the printed output. By virtue of the log transformation for all variables in the model, these parameters can be interpreted as elasticities of voter turn-out with respect to the explanatory variables. It is interesting that all the elasticities span zero, having a positive impact on voter turnout in some regions of the county and negative impacts in others.

To test the sensitivity of the estimates to small changes in the bandwidth parameter, a second set of estimates was generated using a bandwidth of 0.37, which is fairly near the estimated optimal value of 0.3511. These estimates are shown in Figure 6.7, where we see only slight differences in the estimates based

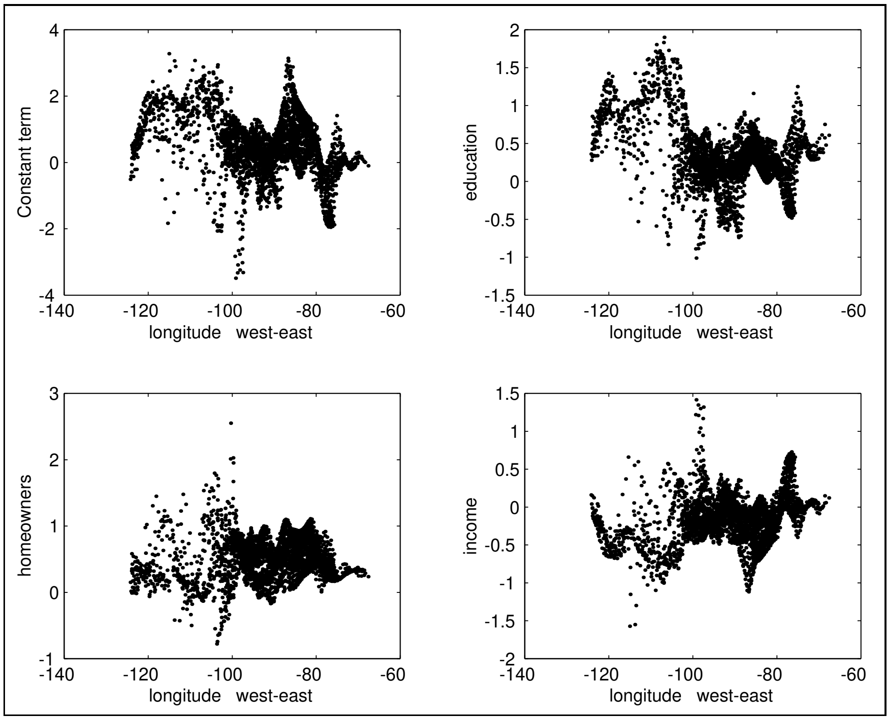

Figure 6.6: GWR estimates based on bandwidth=0.3511

on a slightly larger bandwidth value of 0.37 versus those generated with 0.3511, indicating that the estimates are not extremely sensitive to small changes in this parameter.

The estimates for this large data set that contains outliers (as we saw in Chapter 4) are also sensitive to the weighting method used. Figure 6.8 shows a set of estimates produced using tri-cube weighting that indicated an optimal q nearest neighbors of 19. These estimates are much more sensitive to the outliers, producing estimates with more variation as indicated by the vertical scale in Figure 6.8. There are a handful of outlying estimates with the most dramatic case being a single estimate above 6 for the homeownership variable as well as some large negative estimates near -2 and -3 for the education variable. Gaussian weighting relies on more of the sample observations because the normal probability density assigns positive weight for all values on the real line. In contrast, the tri-cube weighting function sets weights of zero to observations beyond the distance of the q nearest neighbors (19 in this case).

The exponential weighting method produced matrix inversion problems precluding estimates for this method. This can occur when the simplex search attempts to explore bandwidth values that reduce the number of observations with non-negligible weight to less than the number of variables in the model.

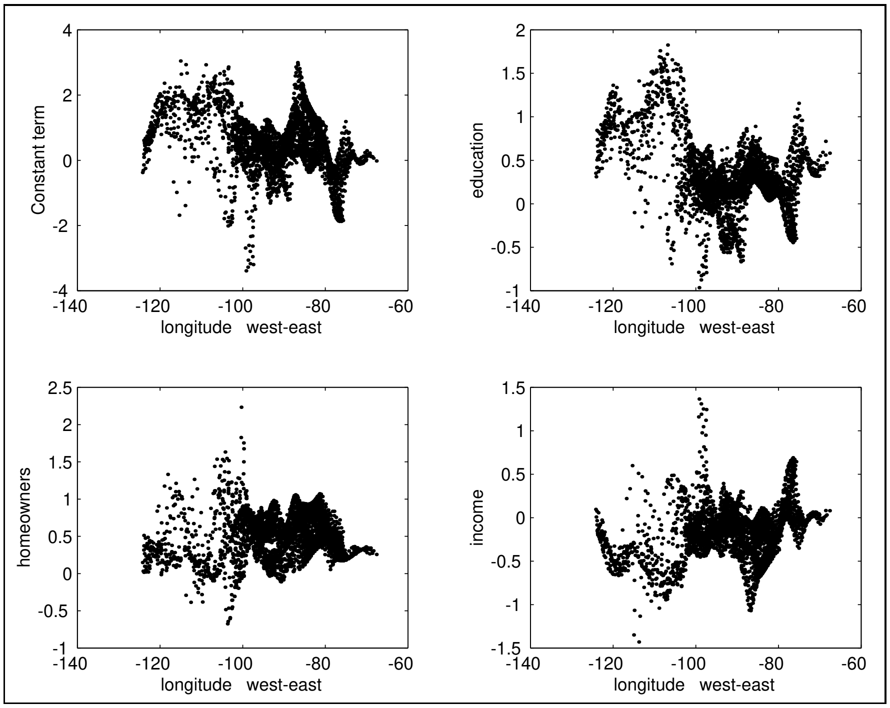

Figure 6.7: GWR estimates based on bandwidth=0.37

It is also possible that a particular set of weights might generate an extremely ill-conditioned matrix X. The weights can interact with the levels of the variables in unpredictable ways. A solution to this type of problem would be to use a ridge regression method in place of least-squares. Ridge regression based on estimation of the Hoerl-Kennard optimal ridge parameter is available in the *Econometrics Toolbox*. This function takes around 1.8 times as long as least-squares estimation so this would impose a penalty. Note also that bias would be introduced in the estimates, and in the case of GWR estimates for every observation it would be difficult to assess the amount of bias introduced.

As another example of using GWR estimation, we apply the method to our Boston data set. A first issue we need to deal with is the fact that many of the explanatory variables in this data set represent dummy variables. For example, there is a dummy variable for location near the Charles River. When carrying out local linear regressions, these variables will create problems as they will take on constant values for subsamples of the observations. In the case of the Charles River dummy variable, subsets of observations away from the river will all take on values of zero, producing an invertibility problem.

In addition to the dummy variables in the data set, other variables such as the property tax rates are uniform over municipal areas like the city of Boston.

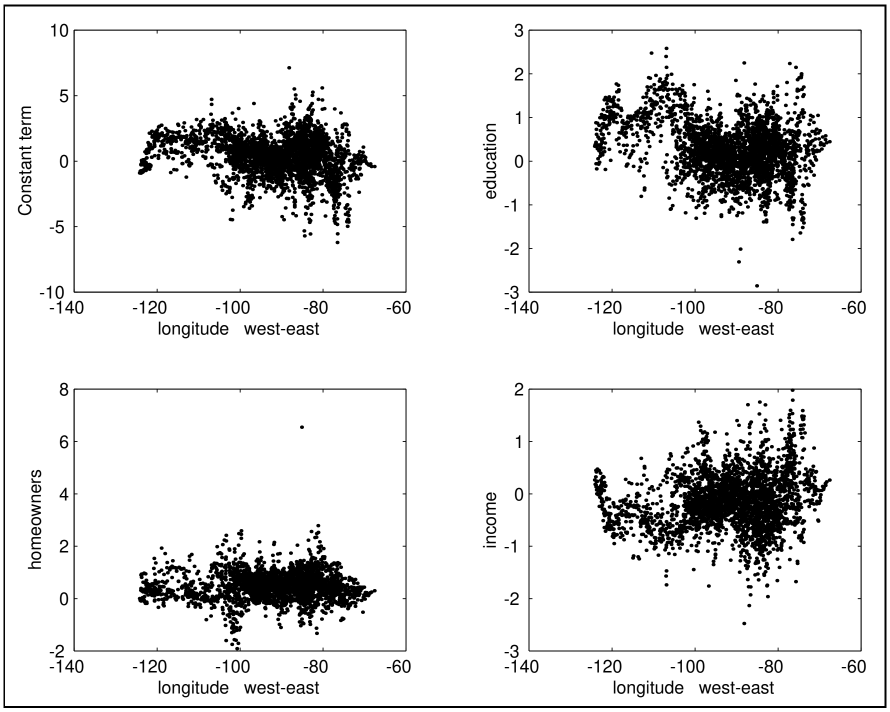

Figure 6.8: GWR estimates based on tri-cube weighting

Here too, when we attempt to produce estimates for a subset of observations in the city of Boston the tax rate variable will take on constant values creating a perfect linear combination with the intercept term in the model.

For large data sets, examining these issues could be quite a task. Further complicating the task is the fact that one does not know a priori what the sample size will be as it is a function of the estimated bandwidth parameter in the model.

Example 6.6 produces estimates for a truncated version of the Harrison-Rubinfeld Boston data set where we eliminated problematical variables that would produce perfect linear combinations over locally linear subsets of the observations. This left us with five explanatory variables plus a constant term. (The program does not contain all of the plot statements to save space.)

```
% ----- Example 6.6 GWR estimates for the Boston data set
load boston.trunc; % Harrison-Rubinfeld data
load latitude.data; load longitude.data;
[n k] = size(boston); y = boston(:,k);  % median house values
x = boston(:,1:k-1);                    % other variables
vnames = strvcat('hprice','crime','rooms2','houseage', ...
         'distance','blackpop','lowclass');
ys = studentize(y); n = length(y); xs = [ones(n,1) studentize(x)];
clear x; clear y; clear boston;
```

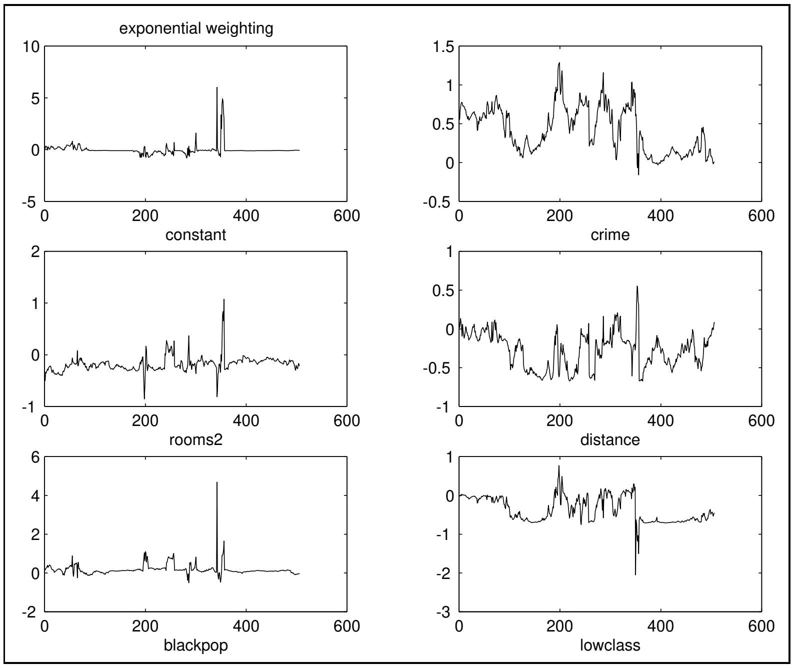

Figure 6.9: Boston GWR estimates - exponential weighting

```
info.dtype = 'exponential';
tic; res1 = gwr(ys,xs,latitude,longitude,info); toc;
info.dtype = 'gaussian';
tic; res2 = gwr(ys,xs,latitude,longitude,info); toc;
info.dtype = 'tricube';
tic; res3 = gwr(ys,xs,latitude,longitude,info); toc;
% plot estimates for comparison
tt=1:n;
subplot(3,2,1), plot(tt,res1.beta(:,1),'-k');
title('exponential weighting'); xlabel('constant');
subplot(3,2,2), plot(tt,res1.beta(:,2),'-k'); xlabel('crime');
subplot(3,2,3), plot(tt,res1.beta(:,3),'-k'); xlabel('rooms2');
subplot(3,2,4), plot(tt,res1.beta(:,4),'-k'); xlabel('distance');
subplot(3,2,5), plot(tt,res1.beta(:,5),'-k'); xlabel('blackpop');
subplot(3,2,6), plot(tt,res1.beta(:,6),'-k'); xlabel('lowclass');
```

This program took 306 seconds to produce estimates based on exponential weights, 320 seconds for the gaussian weights and 294 seconds for tri-cube weights.

Recall that this data set contains outliers as well as 16 sample censored observations. The impact of the outliers on the locally linear estimates is clearly visible in Figures 6.9 to 6.11 where the estimates for each of the three weighting methods are shown.

These estimates are extremely sensitive to the weighting method used with

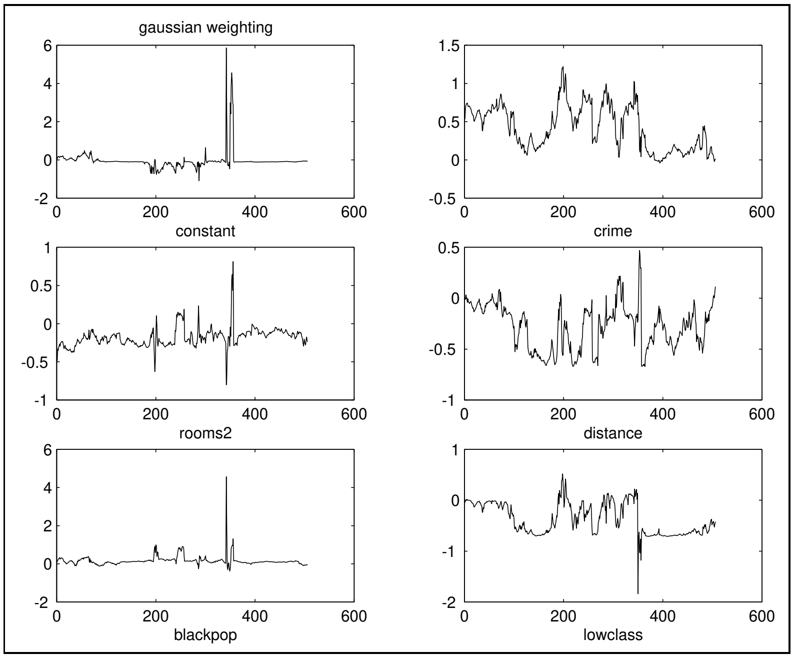

Figure 6.10: Boston GWR estimates - Gaussian weighting

the tri-cube estimates exhibiting the most variability. Intuitively, we would expect this result since this weighting method relies on smaller sub-samples than the exponential or Gaussian weights. Observations 357 through 488 represent census tracts in the city of Boston. The dramatic changes in the estimates in the observation range for Boston may indicate that the city represents a different regime from the other census tracts.

The GWR estimates are sensitive to the sample used as well. To demonstrate this, we produce another set of estimates based on the city of Boston observations 357 to 488. These estimates for the 132 observation city sample are graphed along with those based on the full sample in Figure 6.12 for the Gaussian weighting method and Figure 6.13 for tri-cube weighting. Estimates based on exponential weighting are not presented since they were relatively similar to the results from Gaussian weights.

The times required to produce estimates for the 132 observation sample were: 33 seconds for the exponential weighting, 28 seconds for Gaussian weighting, and 20 seconds for tri-cube weighting. Because the number of explanatory variables in the model based on the full 506 observations and the city sub-sample of 132 observations are the same, we can draw an inference on how the **gwr** function scales up. The time required to produce estimates for the 506 observation sample was 9.3 times as long in the case of the exponential weighting method, 11.4 times as long for the gaussian weighting, and 14.7 times as long for the

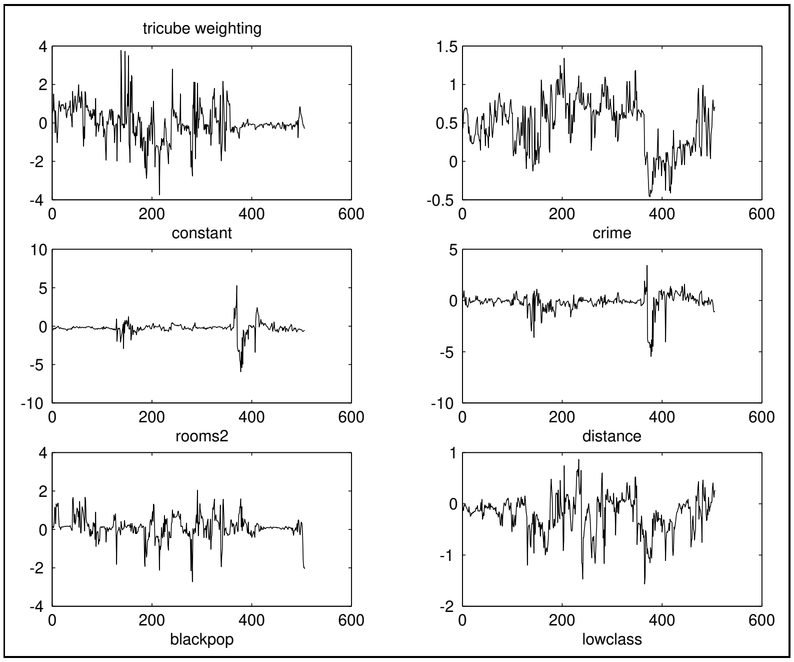

Figure 6.11: Boston GWR estimates - tri-cube weighting

tri-cube weighting. This averages a 12-fold increase in time required when the observations increased by 3.82 times. This implies that the execution time required is almost increasing quadratically with the number of observations. (A quadratic increase in time would lead to 14.67 times as long.)

As a test of this, consider that the 3107 observations used in example 6.5 represent a six-fold increase in observations over 506, so the time required should increase by a quadratic factor of 36 implying 190 minutes required to produce Gaussian weighted estimates in example 6.5. The actual time required was 123 minutes. There may be a timing advantage for the tri-cube weighting method in larger problems as this method took only 69 minutes, whereas the conjectured 36-fold increase would predict an execution time of 176 minutes.

The parameter comparisons indicate that estimates produced using the city sub-sample differed dramatically from those based on the full sample for the Gaussian and exponential weighting methods. They are most similar for the tri-cube weights, but still quite different. One disturbing aspect of the comparisons shown in the figures is that the majority of the six estimates exhibited ranges where the two sets of estimates were of opposite signs.

A partial explanation for these differences would be that the city sub-sample produces "edge effects" because census tracts on the edge of the city of Boston are not able to draw on sample information from neighboring observations. However, if the city of Boston represents another regime, eliminating neighboring

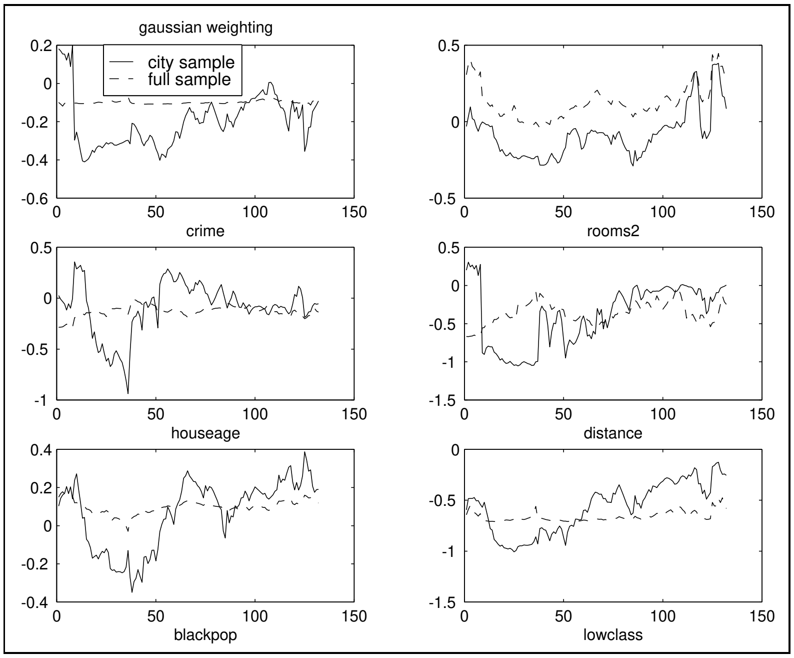

Figure 6.12: Boston city GWR estimates - Gaussian weighting

census tracts that are dissimilar might mitigate some of the influence of aberrant observations arising from these neighbors.

These two applied examples point to some problems that plague the use of locally linear regression methods in spatial analysis. One problem with the non-parametric approach is that valid inferences cannot be drawn for the GWR regression parameters. To see this, consider that the locally linear estimates use the same sample data observations (with different weights) to produce a sequence of estimates for all points in space. Given a lack of independence between estimates for each location, conventional measures of dispersion for the estimates will likely be incorrect. These (invalid) conventional measures are what we report in the results structure from **gwr**, as this is the approach taken by Brunsdon, Fotheringham and Charlton (1996).

Another problem illustrated in the applied examples is that non-constant variance over space, aberrant observations due to spatial enclave effects, or shifts in regime can exert undue influence on the locally linear estimates. Consider that all nearby observations in a sub-sequence of the series of locally linear estimates may be "contaminated" by an outlier at a single point in space. In Chapter 7 we introduce a Bayesian approach that solves this problem by robustifying against aberrant observations. Aberrant observations are automatically detected and downweighted to lessen their influence on the estimates. The non-parametric implementation of the GWR model assumes no outliers.

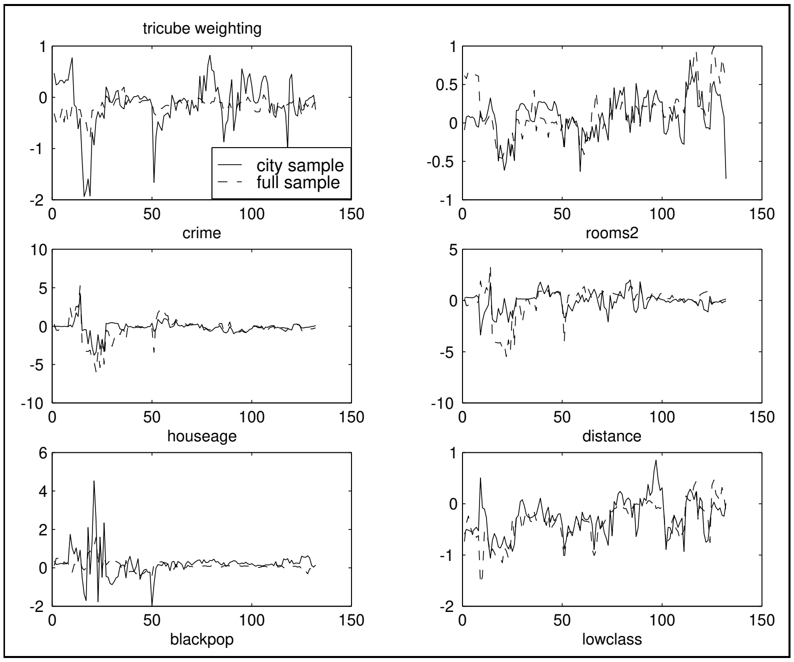

Figure 6.13: Boston city GWR estimates - tri-cube weighting

A third problem is that the non-parametric estimates may suffer from "weak data" problems because they are based on a distance weighted sub-sample of observations. The effective number of observations used to produce estimates for some points in space may be very small. We saw an example of this problem in attempting to implement the GWR with exponential weights for the Pace and Barry data set. Matrix inversion problems arose that precluded obtaining estimates. This problem can be solved with a Bayesian approach that incorporate subjective prior information during estimation. Use of subjective prior information is a Bayesian approach for overcoming "weak data" problems and we will illustrate this in the next chapter.

6.5 Limited dependent variable GWR models

McMillen and McDonald (1998) propose the idea of using locally linear probit models. They point out that one need only apply standard logit or probit methods to the distance weighted sub-samples of the data in place of least-squares. They motivate this by considering the log-likelihoods from locally linear least-squares and probit shown in (6.26) and (6.27).

$$L = \sum_{j=1}^{n} w_{ij}[\log((y_j - X_j\beta_i)/\sigma_i) - \log\sigma_i] \tag{6.26}$$

$$L = \sum_{j=1}^{n} w_{ij}[I_j\log\Phi(X_j\beta_i) + (1 - I_j)\log\Phi(-X_j\beta_i)] \tag{6.27}$$

Where I_j is the dependent variable taking on values of 0 or 1 and Φ denotes the standard normal cumulative density function.

The standard GWR estimates β_i, σ_i can be derived by maximizing the log-likelihood in (6.26) for each observation. This suggests that any maximum likelihood method could be given a locally linear interpretation by adapting it to a distance discounted likelihood like those shown in (6.26) and (6.27). In practice, this means that we need simply alter the **scoref** and **gwr** functions already written to compute logit or probit estimates in place of the least-squares estimates.

There are already functions **logit** and **probit** to implement these estimation methods in the *Econometrics Toolbox*, so altering the **scoref** and **gwr** functions to call these routines in place of the least-squares estimates is relatively simple.

Functions **gwr_logit** and **gwr_probit** were implemented by taking this approach. Some caveats arise in extending the GWR model to the case of logit and probit. One is that we cannot operate on sub-samples using our computational trick that ignores observations with negligible weight, because we could produce a sub-sample with values of all zeros or ones. We need to employ the entire sample which makes this method more computationally intense. These two functions rely on a Gaussian weighting function, since this is the only weighting method that ensures we assign weights to the full sample. Another issue is that solving for logit and probit estimates is a computationally intense operation involving non-linear optimization, so we should not expect great performance from these functions in large samples.

The documentation for **gwr_logit** is shown below, and **gwr_probit** is nearly identical. The only optional input is a bandwidth parameter, since the weighting method is fixed as Gaussian.

```
function result = gwr_logit(y,x,east,north,info);
% PURPOSE: compute geographically weighted logit regression
%----------------------------------------------------
% USAGE: results = gwr_logit(y,x,east,north,info)
% where:   y = dependent variable vector with 0,1 values
%          x = explanatory variable matrix
%       east = x-coordinates in space
%      north = y-coordinates in space
%       info = a structure variable with fields:
%       info.bwidth = scalar bandwidth to use
%                     (default = cross-validation estimate)
% ---------------------------------------------------
%  NOTES: res = gwr_logit(y,x,east,north) does CV estimation of bandwidth
%         Uses Gaussian weighting, and scoref_log for CV
% ---------------------------------------------------
```

```
% RETURNS: a results structure
%        results.meth  = 'gwr_logit'  for logit model
%        results.beta  = bhat matrix    (nobs x nvar)
%        results.tstat = t-stats matrix (nobs x nvar)
%        results.yhat  = yhat
%        results.resid = residuals
%        results.sige  = e'e/(n-dof) (nobs x 1)
%        results.rsqr  = Estrella R^2
%        results.lik   = -Log likelihood
%        results.nobs  = nobs
%        results.nvar  = nvars
%        results.bwidth= bandwidth if gaussian or exponential
%        results.north = north (y-coordinates)
%        results.east  = east  (x-coordinates)
%        results.y     = y data vector
%        results.one   = # of 1's
%        results.nzip  = # of 0's
%---------------------------------------------------
```

These functions return some information specific to logit and probit regression that is presented in the printed output. For example an R^2 measured proposed by Estrella (1998) is provided as well as a count of the number of 0 and 1 values in the dependent variable vector.

The actual code to produce the GWR logit estimates is shown below. We simply call the **logit** function in a loop over the observations using a distance-weighted data matrix 'xs'. Logit estimates are transferred from the results structure returned by the **logit** function to a matrix that will contain estimates for all observations at the end of the loop.

```
wt = zeros(n,1);
for iter=1:n;
    dx = east - east(iter,1);
    dy = north - north(iter,1);
    d = (dx.*dx + dy.*dy);
    % Gausian weights
sd = std(sqrt(d));
    wt = stdn_pdf(sqrt(d)/(sd*bdwt));
wt = sqrt(wt);
xs = matmul(wt,x);
res = logit(y,xs);
bsave(iter,:) = res.beta';
yhat(iter,1) = 1/(1+exp(-x(iter,:)*res.beta));
resid(iter,1) = y(iter,1) - yhat(iter,1);
sigv(iter,1) = res.sige;
tstat(iter,:) = res.tstat';
like(iter,1) = res.lik;
end;
```

We also need to create a score function that can be used to produce a cross-validation bandwidth estimate based on logit (or probit) regressions. Consider that we are calling this score function named **score_log** with the simplex optimization function **fmin**. This means that we are solving an optimization problem where each evaluation of the score function to be optimized requires the solution of n optimization problems to produce logit estimates for each observation. It is somewhat amazing that it works!

```
function score = scoref_log(bdwt,y,x,east,north)
% PURPOSE: evaluates cross-validation score to determine optimal
%          bandwidth for gwr_logit with gauusian weighting
% ------------------------------------------------------
% USAGE: score = scoref_log(bdwt,y,x,east,north);
% where:
%     bdwt = a bandwidth to use in computing the score
%        y = dependent variable
%        x = matrix of explanatory variables
%     east = longitude (x-direction) coordinates
%    north = lattitude (y-direction) coordinates
% ------------------------------------------------------
% RETURNS: score = a cross-validation criterion
% ------------------------------------------------------
[n k] = size(x); res = zeros(n,1); wt = zeros(n,1);
for iter = 1:n;
   dx = east - east(iter,1);
   dy = north - north(iter,1);
   d = (dx.*dx + dy.*dy);
   sd = std(sqrt(d));
   wt = stdn_pdf(sqrt(d)/(sd*bdwt));    % Gausian weights
   wt(iter,1) = 0.0;
wt = sqrt(wt);      xs = matmul(x,wt);
tmp = logit(y,xs);  bi = tmp.beta;
% compute predicted values
yhat = 1/(1+exp(-x(iter,:)*bi));
% compute residuals
res(iter,1) = y(iter,1) - yhat;
end; % end of for iter loop
tmp = res'*res;
score = sqrt(tmp/n);
```

Example 6.7 illustrates using these functions to produce logit and probit GWR estimates based on a binary transformed version of the Columbus crime data set. We graph the logit and probit estimates for comparison, but keep in mind logit estimates are generally around 1.7 times the magnitude of probit estimates (see Green, 1997).

```
% ----- Example 6.8 GWR logit and probit estimates
% load the Anselin data set
load anselin.dat; y = anselin(:,1); nobs = length(y);
x = [ones(nobs,1) anselin(:,2:3)]; [nobs nvar] = size(x);
north = anselin(:,4); east = anselin(:,5);
% convert data to 0,1 values
yc = ones(nobs,1);
for i=1:nobs
 if y(i,1) < 20.0
 yc(i,1) = 0.0;
 end;
end;
vnames = strvcat('crime','constant','income','hvalue');
tic; result1 = gwr_logit(yc,x,east,north); toc;
tic; result2 = gwr_probit(yc,x,east,north); toc;
```

```
prt(result1,vnames);
% plot results for comparison (see also plt)
tt=1:nobs;
subplot(2,2,1),
plot(tt,result1.beta(:,1),tt,result2.beta(:,1),'--');
xlabel('Constant term');
subplot(2,2,2),
plot(tt,result1.beta(:,2),tt,result2.beta(:,2),'--');
xlabel('Household income');
subplot(2,2,3),
plot(tt,result1.beta(:,3),tt,result2.beta(:,3),'--');
xlabel('House value'); legend('logit','probit');
subplot(2,2,4);
plot(tt,yc,'ok',tt,result1.yhat,'+k',tt,result2.yhat,'*k');
legend('actual','logit','probit');
xlabel('actual vs. predicted');
```

The time required to produce the logit estimates was 62 seconds and 169 seconds for the probit estimates. This is truly amazing given the computational intensity of finding the cross-validation bandwidth estimates. Figure 6.14 shows the graph produced by the program in example 6.7, where we see that both sets of estimates are quite similar, when we take into account the scale factor difference of 1.7 between these two types of estimates.

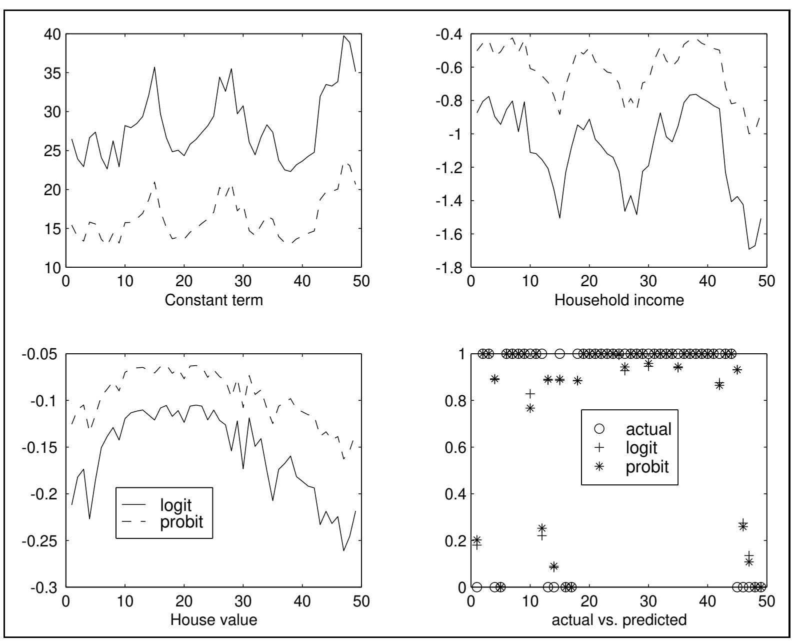

Figure 6.14: GWR logit and probit estimates for the Columbus data

6.6 Chapter Summary

We have seen that locally linear regression models can be estimated using distance weighted sub-samples of observations to produce different estimates for every point in space. This approach can deal with spatial heterogeneity and provide some feel for parameter variation in the relationships over space.

Some problems arise in using spatial expansion models because they tend to produce heteroscedastic disturbances by construction. This problem is overcome to some extent with the DARP model approach.

The non-parametric locally linear regression models produce problems with respect to inferences about the parameters as they vary over space. Because the distance weighted sub-samples used to construct the estimates are not independent, traditional methods for constructing measures of dispersion used in regression modeling cannot be used. It would be possible to produce bootstrap estimates of dispersion for the purposes of inference, but this would add to the computational intensity of the method.

Another problem is that the estimates and associated inferences for these models are conditional on the distance decay parameter used and the weighting method. In addition, we found that changing the sample can produce dramatic changes in the parameters for these models, including changes in the sign of the parameters.

The computational intensity of these methods may be greater than generally recognized. Intuitively, since the method requires only least-squares estimation, it would seem to have computational advantages over spatial autoregressive models where one must deal with eigenvalue and determinant calculations for large sparse weight matrices. We saw however that for the same 3,107 observation problem estimated using spatial autoregressive models in 100 to 300 seconds, it took 50 to 70 minutes to produce GWR estimates. The computational intensity arises from the need to solve for a cross-validation estimate of the important distance decay parameter in these models (or to find a cross-validation estimate of the optimal number of nearest neighbors in the case of tri-cube weighting). Given a decay parameter or the number of nearest neighbors, estimates can be produced in around 300 seconds, not unlike the case of spatial autoregressive models. Unfortunately, it is unlikely that one would have knowledge of these parameters.

Chapter 7

Bayesian Locally linear spatial models

A Bayesian treatment of the spatial expansion model and locally linear geographically weighted regressions is set forth in this chapter. We saw in Chapter 6 that Casetti's spatial expansion model suffers from heteroscedasticity by construction. A Bayesian variant of this model can rely on the Gibbs sampling methods we introduced for the Bayesian heteroscedastic linear model in Section 4.4.1 to remedy this situation. The Bayesian version of the spatial expansion model is the subject of Section 7.1.

While the use of locally linear regression represents a true contribution to spatial econometrics, we saw in Chapter 6 that problems arise when outliers exist in the sample data. We demonstrate that a Bayesian treatment can resolve these problems and has a great many advantages over the GWR method discussed in Chapter 6.

One problem with the non-parametric approach was that valid inferences cannot be drawn for the GWR regression parameters because the locally linear estimates re-use sample data observations (with different weights) to produce a sequence of estimates for all points in space. Given a lack of independence between estimates for each location, conventional regression measures of dispersion for the estimates will likely be incorrect. A Bayesian implementation using the Gibbs sampler to estimate the parameters will produce valid measures of dispersion for the estimates. Recall, the Gibbs sampler does not require independent sample observations to produce valid estimates of dispersion. A lack of independence only means we may need to carry out more draws during sampling.

Another problem is that non-constant variance over space, aberrant observations due to spatial enclave effects, or shifts in regime can exert undue influence on locally linear estimates. All nearby observations in a sub-sequence of the series of locally linear estimates may be "contaminated" by an outlier at a single point in space. This is a serious issue because the goal of GWR estimation is detection of changes in the relationship over space. Using GWR estimates, we are likely to infer changes in the relationship when in fact the changes are due to the presence of outliers.

The Bayesian approach solves this problem by producing estimates that are robust against aberrant observations taking a similar approach to that used in Chapter 4 for Bayesian spatial autoregressive models. Aberrant observations are automatically detected and downweighted to mitigate their influence on the estimates. This ensures that inferences of spatial changes in the relationship represent true changes rather than aberrant observations.

Section 7.2 illustrates a Bayesian variant of the GWR model (that we label BGWRV) which accomodates outliers and non-constant variance using the same methods as in Section 7.1.

A third problem is that non-parametric estimates based on a distance weighted sub-sample of observations may suffer from "weak data" problems. The effective number of observations used to produce estimates for some points in space may be very small. As we saw in Chapter 6, this can result in situations where the matrix $X'X$ is non-invertible. This problem can be solved by incorporating subjective prior information during estimation. Use of subjective prior information is a common Bayesian approach to overcoming "weak data" problems. Section 7.3 extends the heteroscedastic Bayesian GWR model from Section 7.2 to include prior information based on an explicit specification of the relationship used to smooth the parameters over space. This allows us to subsume the non-parametric GWR method as part of a much broader class of spatial econometric models. For example, this Bayesian variant of the GWR (which we label BGWR) can be implemented with parameter smoothing relationships that result in: 1) a locally linear variant of the spatial expansion methods introduced by Casetti (1972,1992), 2) a parameter smoothing relation appropriate for monocentric city models where parameters vary systematically with distance from the center of the city, 3) a parameter smoothing scheme based on contiguity relationships, and 4) a parameter smoothing scheme based on GWR-type distance decay.

Finally, section **??** provides an applied exercise that compares the GWR and BGWR methods using a data set on employment, payroll and establishments from Cuyahoga county. We demonstrate that outliers in this data set adversely affect the GWR estimates, but not the BGWR results.

7.1 Bayesian spatial expansion

As illustrated in Chapter 6, the spatial expansion model contains a heteroscedastic disturbance structure by construction. This was motivated by working with a distance expansion model shown in (7.1), where we add a stochastic term u to reflect some error in the expansion relationship.

$$y = X\beta + e$$

$$\beta = DJ\beta_0 + u \tag{7.1}$$

Substituting the second equation from (7.1) into the first, produces:

$$y = XDJ\beta_0 + Xu + e \tag{7.2}$$

It should be clear that the new composite disturbance term $Xu + e$ will reflect heteroscedasticity unless the expansion relationship is exact and $u = 0$.

We can accomodate this situation using an extended model that does not assume $e \sim N(0, \sigma^2 I_n)$, relying instead on the heteroscedastic disturbance assumption from Chapter 4, $e \sim N(0, \sigma^2 V)$, where $V = \mathrm{diag}(v_1, v_2, \ldots, v_n)$. This has the virtue that the heteroscedastic form is not modeled directly, but rather estimated. Formally, our Bayesian spatial expansion model can be stated as:

$$\begin{aligned}
y &= X\beta + \varepsilon & (7.3)\\
\beta &= DJ\beta_0 + u & (7.4)\\
\beta &= ZJ\beta_0 + u & (7.5)\\
u &\sim N(0, \sigma^2) &\\
\varepsilon &\sim N(0, \sigma^2 V) &\\
V &= \mathrm{diag}(v_1, v_2, \ldots, v_n) &\\
r/v_i &\sim \mathrm{ID}\ \chi^2(r)/r &\\
r &\sim \Gamma(m, k) &\\
\sigma &\sim \Gamma(\nu_0, d_0) &
\end{aligned}$$

Where we use expression (7.4) for the distance expansion model and (7.5) for x-y expansion.

A point to note is that we don't model the disturbance in (7.3) which has a variance-covariance structure:

$$E(Xu + \varepsilon)(Xu + \varepsilon)' = \sigma^2 (I_n + Xuu'X') \tag{7.6}$$

We model $\sigma^2(I_n + Xuu'X')$ using the flexible form, $\sigma^2 V$ involving the variance scalar parameters v_i. A virtue of proceeding in this way is that implementation of the Bayesian model with a large hyperparameter r leads to $V = I_n$ and estimates from ordinary spatial expansion. Another advantage is that the estimates v_i provide a diagnostic measure of where the spatial expansion restriction is inconsistent with the sample data. Given that a heteroscedastic prior is implemented using a value for the hyperparameter $r = 4$, large and systematic deviations from unity in the v_i estimates would be indicative of a problem with the spatial expansion relationship. In cases where the spatial expansion relation is consistent with the sample data, we would expect to see v_i estimates near unity.

A disadvantage of this approach is that we ignore the implied theoretical structure of the disturbances that includes the matrix X. This type of model could also be implemented in our Bayesian setting, as illustrated by Cooper (1972).

Because the spatial expansion model relies on least-squares estimates, our Bayesian variant of this model is simple to implement and executes quickly. The Gibbs sampler for this model relies on the conditional distributions for β_0, σ and V shown below.

$$
\begin{aligned}
p(\beta_0|\sigma, V) &\sim N[\hat{\beta}_0, \sigma^2(\tilde{X}'\tilde{X})^{-1}] & (7.7)\\
\hat{\beta}_0 &= (\tilde{X}'\tilde{X})^{-1}\tilde{X}'\tilde{y} \\
\tilde{X} &= V^{1/2}X, \quad \tilde{y} = V^{1/2}y \\
& & (7.8)
\end{aligned}
$$

$$p(\sigma|\beta_0, V) = [\sum_{i=1}^{n}(e_i^2/v_i)/\sigma^2]|(\beta_0, V) \sim \chi^2(n) \qquad (7.9)$$

$$p(V|\beta_0, \sigma) = [(\sigma^{-2}e_i^2 + r)/v_i]|(\beta_0, \sigma) \sim \chi^2(r+1) \qquad (7.10)$$

After cycling through the conditional distributions to obtain an updated draw for β_0, σ and V, we update the spatial expansion estimates using expression (7.11) for x-y expansion or (7.12) for distance expansion.

$$
\begin{aligned}
\beta &= ZJ\beta_0 & (7.11)\\
\beta &= DJ\beta_0 & (7.12)
\end{aligned}
$$

The expansion estimates for β are calculated using each draw of the base estimates β_0 with the means serving as posterior estimates. Valid measures of dispersion for both β_0 and β can be computed using draws from the sampler.

7.1.1 Implementing Bayesian spatial expansion

We devised a function **bcasetti** shown below to implement this variant of the spatial expansion model. Only the code that implements Gibbs sampling for the case of a distance expansion model is shown below, as the case of x-y expansion is quite similar.

```
function results=bcasetti(y,x,xc,yc,ndraw,nomit,option)
% PURPOSE: computes a heteroscedastic Bayesian variant of
%          Casetti's spatial expansion regression
%          y = X*g + e, e = N(0,sige*V)
%          g = Zx*b0x + Zy*b0y, for x-y expansion
%          g = D b0,            for distance expansion
%          V = diag(v1,v2,...vn), r/vi = ID chi(r)/r,
%---------------------------------------------------
% USAGE: results = bcasetti(y,x,xc,yc,ndraw,nomit,option)
% where:       y = dependent variable vector
%              x = independent variables matrix
```

```
%            xc = latittude (or longitude) coordinate
%            yc = longitude (or latittude) coordinate
%         ndraw = # of draws
%         nomit = # of draws to omit for burn-in
%       option  = a structure variable containing options
%       option.exp  = 0 for x-y expansion (default)
%                   = 1 for distance from ctr expansion
%       option.ctr  = central point observation # for distance expansion
%       option.norm = 1 for isotropic x-y normalization (default=0)
%       option.rval = rval for prior (r above, default = 4)
%---------------------------------------------------
% RETURNS:
%       results.meth   = 'bcasetti'
%       results.b0draw = (underlying b0x, b0y draws) ndraw-nomit x nvar
%       results.beta   = mean of spatially expanded estimates (nobs x nvar)
%       results.vmean  = mean of vi draws (nobs x 1)
%       results.yhat   = mean of posterior predicted values
%       results.resid  = mean of residuals based on predicted values
%       results.sdraw  = sige draws (ndraw-nomit x 1)
%       results.rsqr   = rsquared
%       results.rbar   = rbar-squared
%       results.rval   = rval (from input)
%       results.nobs   = nobs
%       results.nvar   = # of variables in x
%       results.y      = y data vector
%       results.xc     = xc
%       results.yc     = yc
%       results.ctr    = ctr (if input)
%       results.dist   = distance vector (if ctr used)
%       results.exp    = exp input option
%       results.norm   = norm input option
%       results.time   = time taken for sampling
% --------------------------------------------------
% NOTE: assumes x(:,1) contains a constant term
% --------------------------------------------------
nflag = 0;
if nargin == 7, fields = fieldnames(option); nf = length(fields);
 exp = 0; ctr = 0; nflag = 0; rval = 4;
  for i=1:nf
    if strcmp(fields{i},'exp'),      exp = option.exp;
    elseif strcmp(fields{i},'ctr'),  ctr = option.ctr;
    elseif strcmp(fields{i},'norm'), nflag = option.norm;
    elseif strcmp(fields{i},'rval'), rval = option.rval;
    end;
  end; % end of for i
elseif nargin == 6, exp = 0; rval = 4; option.exp = 0;
else, error('Wrong # of arguments to bcasetti');
end;
[nobs nvar] = size(x); if nflag == 1, [xc yc] = normxy(xc,yc); end;
results.meth = 'bcasetti'; results.y = y; results.nobs = nobs;
results.nvar = nvar; results.xc = xc; results.yc = yc; results.rval = rval;
results.option = option; results.exp = exp; results.norm = nflag;
switch exp
case{1} % distance  expansion
xi = xc(ctr); yi = yc(ctr);
d = sqrt((xc-xi).*(xc-xi) + (yc-yi).*(yc-yi));
dvec = d; results.dist= dvec;
```

```
% transform x-variables using distance vector
xt = x(:,2:nvar); xx = matmul(xt,dvec); xmat = [x xx];
[junk tvar] = size(xmat); V = ones(nobs,1); in = ones(nobs,1);
sige = 1; yhmean = zeros(nobs,1); vmean = zeros(nobs,1);
bhmean = zeros(nobs,nvar-1); t0 = clock;
for iter=1:ndraw; % begin sampling
xmatv = matmul(xmat,sqrt(V)); yv = y.*sqrt(V);
% update b0
xpxi = inv(xmatv'*xmatv); xpy = xmatv'*yv;
b0 = xpxi*xpy; a = chol(xpxi);
b0 = sqrt(sige)*a'*randn(length(b0),1) + b0;
% update sige
e = yv - xmatv*b0; chi = chis_rnd(1,nobs);
sige = (e'*e)/chi;
% update vi
e = y - xmat*b0; chiv = chis_rnd(nobs,rval+1);
vi = ((e.*e./sige) + rval)./chiv; V = in./vi;
beta = zeros(nobs,(nvar-1)); yhat = zeros(nobs,1);
xx = matmul(ones(nobs,nvar-1),dvec);
xxxy = [x xx]; tvar = length(b0);
    yhat(:,1) = xmat(:,1:nvar)*b0(1:nvar,1);
    for j=nvar+1:tvar
    beta(:,j-nvar) = xxxy(:,j)*b0(j,1);
    yhat(:,1) = yhat(:,1)  + xmat(:,j)*b0(j,1);
    end;
if iter > nomit, % save draws
    vmean = vmean + vi;         bsave(iter-nomit,:) = b0';
    ssave(iter-nomit,1) = sige; yhmean = yhmean + yhat;
    bhmean = bhmean + beta;
end;
end; gtime = etime(t0,clock); % end of sampling loop
vmean = vmean/(ndraw-nomit); yhat = yhmean/(ndraw-nomit);
beta = bhmean/(ndraw-nomit); results.vmean = vmean
results.beta = beta;    results.yhat = yhat;
results.b0draw = bsave; results.nvar = nvar;
results.ctr = ctr;      results.sdraw = ssave;
results.yhat = yhat;    results.rval = rval;
results.time = gtime;
otherwise, error('casetti: check option input argument');
end;
```

This function is relatively similar to the **casetti** function, with two differences. One is the addition of the vector V to produce robust estimates. The second is that a multivariate normal draw is carried out to obtain β_0. This draw of the base estimates is used to compute expansion estimates for every draw. Residuals deflated by V based on these expansion estimates are used to update the estimate of σ and undeflated residuals are used to update V. To conserve on storage, we save only the mean of the draws for the expansion estimates, as inferences usually center on the base estimates.

7.1.2 Applied examples

Example 7.1 illustrates using the function with Anselin's Columbus neighborhood crime data set where estimates from the homoscedastic **casetti** function are plotted against those from **bcasetti**. We demonstrate that using a large value of 40 for the hyperparameter r produces estimates similar to ordinary spatial expansion, and compare these estimates to the heteroscedastic case based on a hyperparameter value of $r = 4$.

```
% ----- Example 7.1 Using the bcasetti() function
load anselin.dat; % load Anselin (1988) Columbus neighborhood crime data
y = anselin(:,1); n = length(y);
x = [ones(n,1) anselin(:,2:3)];
% Anselin (1988) x-y coordinates
xc = anselin(:,4); yc = anselin(:,5);
vnames = strvcat('crime','const','income','hse value');
% do Casetti regression using x-y expansion (the default)
res1 = casetti(y,x,xc,yc);
ndraw = 1250; nomit = 250;
option.rval = 40; % homoscedastic estimates
res2 = bcasetti(y,x,xc,yc,ndraw,nomit,option);
option.rval = 4;  % heteroscedastic estimates
res3 = bcasetti(y,x,xc,yc,ndraw,nomit,option);
tt=1:n;
subplot(2,2,1),,
plot(tt,res1.beta(:,1),tt,res2.beta(:,1),'ok',tt,res3.beta(:,1),'--k');
xlabel('x-income');
subplot(2,2,2),,
plot(tt,res1.beta(:,2),tt,res2.beta(:,2),'ok',tt,res3.beta(:,2),'--k');
xlabel('y-income'); legend('casetti','Bayes r=40','Bayes r=4');
subplot(2,2,3),
plot(tt,res1.beta(:,3),tt,res2.beta(:,3),'ok',tt,res3.beta(:,3),'--k');
xlabel('x-house');
subplot(2,2,4),
plot(tt,res1.beta(:,4),tt,res2.beta(:,4),'ok',tt,res3.beta(:,4),'--k');
xlabel('y-house');
```

The graphical comparison of the estimates is shown in Figure 7.1, where we see that a homoscedastic prior based on $r = 40$, reproduced the estimates from **casetti**. In contrast, we see a large difference in the estimates based on the heteroscedastic prior with $r = 4$.

An insight as to why these large differences arise is provided by the v_i estimates returned by the **bcasetti** function. These are shown in Figure 7.2, where we see evidence of outliers at observations 2, 4 and 34.

The impact of these outliers is twofold. First, the robust base estimates will be different, since they take the outliers into account. This is consistent with our knowledge of the impact outliers have on least-squares estimates. Any differences in the base estimates will be propagated when we implement the expansion relationship using different base estimates. This can produce situations where we will see very different expansion estimates. This is the case here, as can be seen in Table 7.1 where the base estimates along with the first six expansion estimates are shown for the ordinary spatial expansion model and the Bayesian model estimated with a heteroscedastic prior based on $r = 4$.

Another difference between the two sets of estimates is that the homoscedastic model overstates the precision of the estimates producing higher $t-$statistics for the base x- and y-estimates. Anselin (1988) comes to a similar conclusion

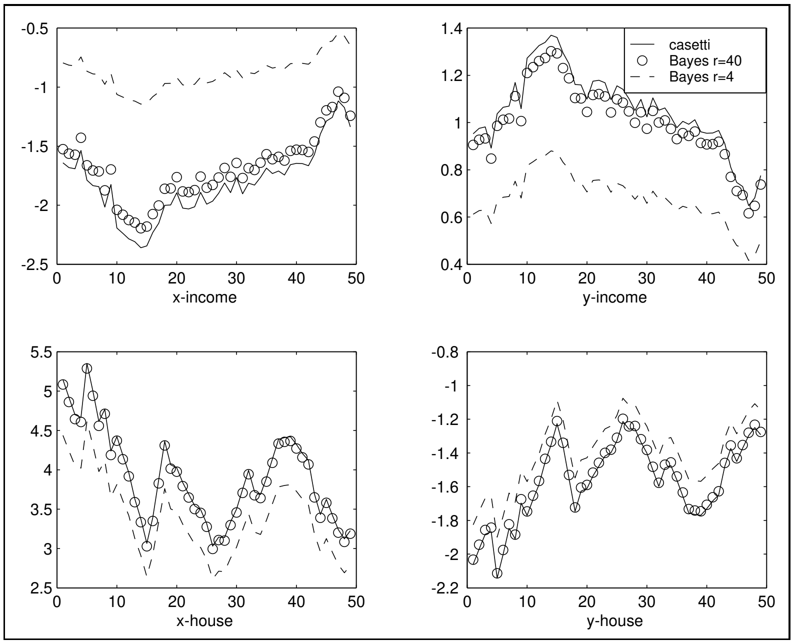

Figure 7.1: Spatial expansion versus robust estimates

using maximum likelihood methods to correct for heteroscedasticity — the significance of the base estimates decreases when corrected for heteroscedasticity inherent in the spatial expansion estimates.

The printed output also showed a slightly lower $R^2 = 0.6179$ for the Bayesian model versus $R^2 = 0.6330$ for the Casetti model, consistent with robust estimates. Finally, the time required by the **bcasetti** to produce 1,250 draws was only 19 seconds which compares favorably to 1 second to produce estimates using the **casetti** function.

As another illustration, we implement these two models on the Boston data set involving 506 observations. This was done in example 7.2, where we implement a distance expansion model based on census tract 411 at the center of the central business district.

```
% ----- Example 7.2 Boston data spatial expansion
load boston.raw; % Harrison-Rubinfeld data
load latittude.data; load longitude.data;
[n k] = size(boston);y = boston(:,k);  % median house values
x = boston(:,1:k-1);       % other variables
vnames = strvcat('hprice','const','crime','zoning','industry', ...
         'charlesr','noxsq','rooms2','houseage','distance','access', ...
         'taxrate','pupil/teacher','blackpop','lowclass');
ys = studentize(y); xs = [ones(n,1) studentize(x)];
clear x; clear y; clear boston;
info.exp = 1;   % distance expansion
```

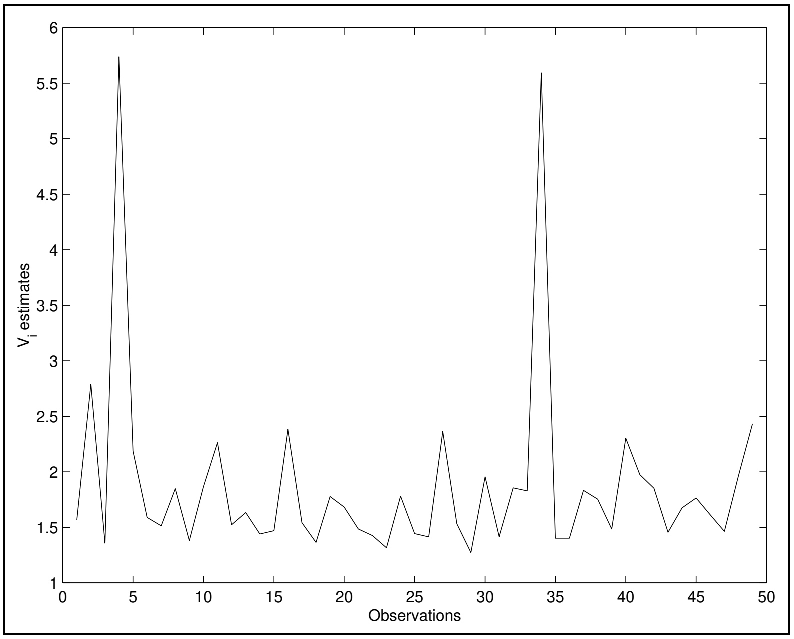

Figure 7.2: Mean of the v_i draws for $r = 4$

```
info.ctr = 411; % Boston CBD
res1 = casetti(ys,xs,latittude,longitude,info);
info.rval = 4; % heteroscedastic prior
ndraw = 1250; nomit=250;
res2 = bcasetti(ys,xs,latittude,longitude,ndraw,nomit,info);
tt=1:n;
subplot(2,2,1), plot(tt,res1.beta(:,1),tt,res2.beta(:,1),'--k');
xlabel('x-crime'); legend('casetti','Bayesian');
subplot(2,2,2), plot(tt,res1.beta(:,2),tt,res2.beta(:,2),'--k');
xlabel('y-crime');
subplot(2,2,3), plot(tt,res1.beta(:,10),tt,res2.beta(:,10),'--k');
xlabel('x-noxsq');
subplot(2,2,4), plot(tt,res1.beta(:,11),tt,res2.beta(:,11),'--k');
xlabel('y-noxsq');
pause;
plot(tt,res2.vmean);
ylabel('mean of v_i draws');
```

Recall, this data set exhibits a large number of outliers as well as sample truncation for 16 observations. We would expect to see dramatic differences between robust Bayesian estimates and those from ordinary spatial expansion.

An experiment using x-y expansion in place of distance expansion took 230 seconds to produce 1,250 draws, which scales up nicely from the 49 observation neighborhood crime data set, which took 20.55 seconds. This data sample contains 10 times the observations and took slightly more than 10 times as long to

Table 7.1: Bayesian and ordinary spatial expansion estimates

Base estimates				
Variable	Casetti	$t-$stat	Bayesian	$t-$stat
const	69.4961	15.1051	70.7090	15.3599
income	-4.0859	-1.9519	-4.9189	-2.0902
hse value	0.4039	0.5179	0.7339	0.7972
x-income	-0.0460	-1.3496	-0.0249	-0.5763
x-hse value	0.0267	2.0275	0.0181	1.0151
y-income	0.1214	2.2131	0.1108	1.6134
y-hse value	-0.0486	-2.3418	-0.0455	-1.6419
R^2	0.6330		0.6179	
Casetti expansion estimates				
Obs#	x-income	x-hse value	y-income	y-hse value
1	-1.6407	0.9522	5.1466	-2.0599
2	-1.6813	0.9757	4.9208	-1.9695
3	-1.6909	0.9813	4.7009	-1.8815
4	-1.5366	0.8918	4.6645	-1.8670
5	-1.7872	1.0372	5.3519	-2.1421
6	-1.8342	1.0645	5.0009	-2.0016
Bayesian expansion estimates				
Obs#	x-income	x-hse value	y-income	y-hse value
1	-1.0208	0.6992	4.7341	-1.9231
2	-1.0460	0.7164	4.5264	-1.8387
3	-1.0521	0.7206	4.3242	-1.7566
4	-0.9561	0.6548	4.2907	-1.7430
5	-1.1120	0.7616	4.9229	-1.9998
6	-1.1412	0.7816	4.6001	-1.8687

produce the same 1,250 draws for the case of x-y expansion. For the distance expansion in example 7.2, it took 134 seconds to generate 1,250 draws.

Four of the 13 expansion coefficients are shown in Figure 7.3. These are fairly representative of the differences in other coefficients that appeared in the comparison. Two patterns emerge, one where the robust nature of the Bayesian estimates is clear. This is illustrated by the expansion estimates for 'crime', 'zoning' and the 'Charles river' variables. Note that in the case of the Charles river dummy variable the estimates are similar, but we still see evidence that the Bayesian estimates are more robust.

The other pattern that emerged in comparing estimates was a sign reversal between the Bayesian and non-Bayesian estimates. This is illustrated in the plot for the 'industry' variable in Figure 7.3 as well as the 'rooms2' and 'distance' variables in Figure 7.4.

Figure 7.4 provides a plot of the estimates for the 'noxsq' and 'houseage' variables, where we see almost identical estimates from ordinary and Bayesian expansion. Consider that these two variables are most likely to be consistent with a distance from the central city expansion. Our Bayesian model in the case of these two variables produced estimates similar to ordinary expansion because the heteroscedasticity is not present when the expansion relationship provides a good fit to the sample data.

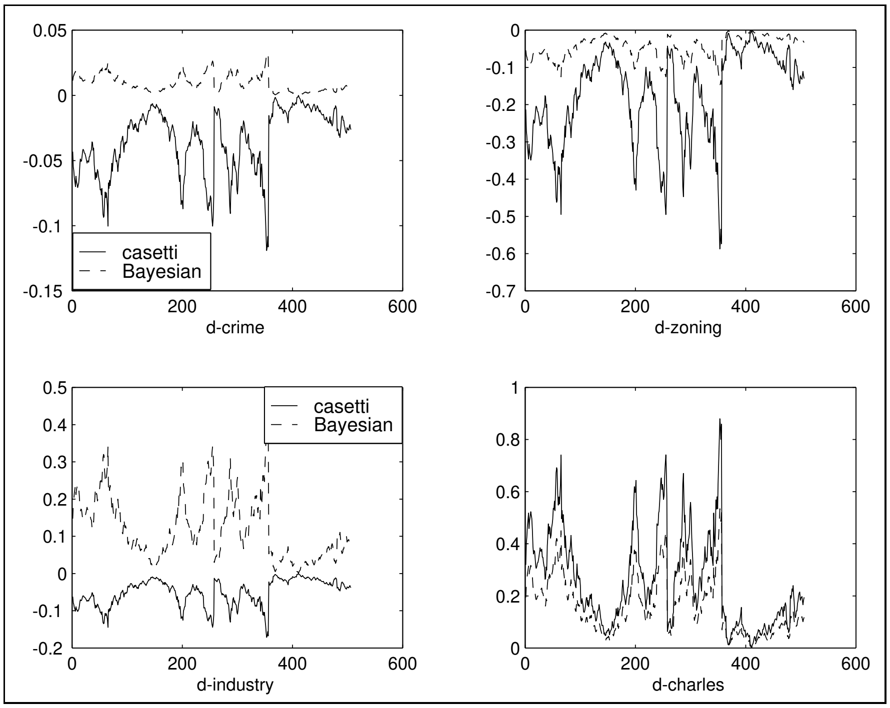

Figure 7.3: Expansion vs. Bayesian expansion for Boston

The introduction of the variance scaling parameters v_i are the source of the observed differences in estimates from **casetti** and **bcasetti**. As noted earlier, these terms affect both the base estimates β_0 as well as the expansion process. Figure 7.5 shows the mean of the v_i estimates produced by the Gibbs sampler, where we see ample evidence of non-constant variance. Observations 357 to 488 represent census tracts in the city of Boston, which appear to exhibit a larger variance that census tracts not in the city. The same is true of other census tracts from observations 170 to 270.

Table 7.2 shows the base estimates along with $t-$statistics from the two models. An asterisk '*' symbol was appended to variable names for cases where the two sets of estimates were significantly different at the 0.90 level based on a $t-$test. These results indicate that 8 of 27 estimates were significantly different.

A difference also exists between the estimate for σ^2 from spatial expansion which was 0.2014 and the Bayesian model where this parameter was 0.0824. We would expect the robust Bayesian estimate to be smaller than that from the ordinary spatial expansion model. This partially accounts for upward bias in

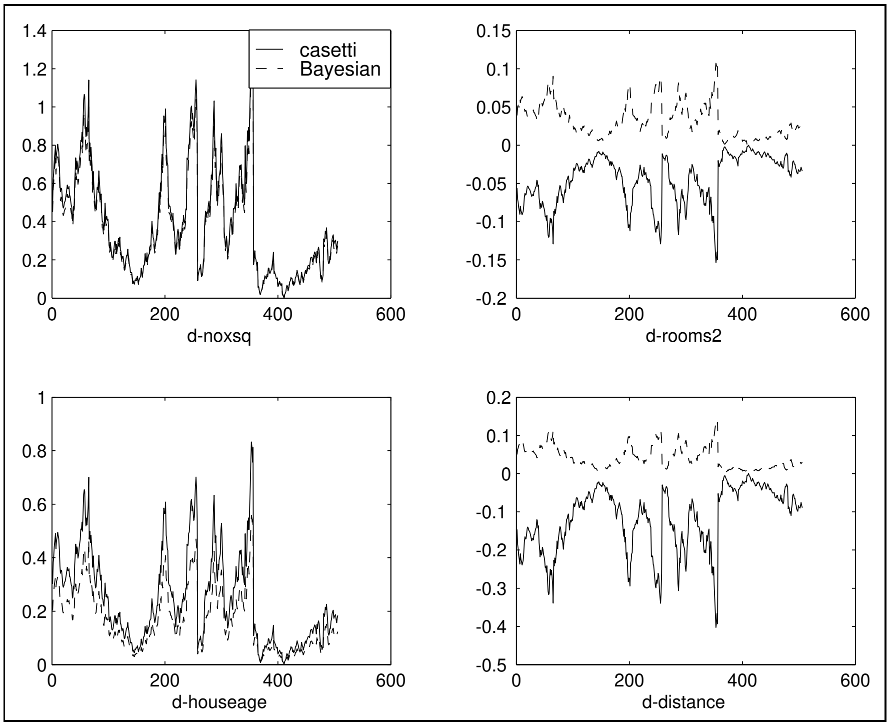

Figure 7.4: Expansion vs. Bayesian expansion for Boston (continued)

the $t-$statistics that are based on $\sigma^2(X'X)^{-1}$.

7.2 Producing robust GWR estimates

GWR estimates can be produced using the Bayesian heteroscedastic linear model that we applied to spatial expansion in the previous section. This model, which we label BGWRV is shown in (7.13).

$$\begin{aligned} W_i^{1/2}y &= W_i^{1/2}X\beta_i + \varepsilon_i \\ \varepsilon_i &= N(0, \sigma_i^2 V_i), \quad V_i = \mathrm{diag}(v_{1i} \ldots, v_{ni}) \end{aligned} \tag{7.13}$$

Estimates are based on the Gibbs sampler, where we simply call on a function to carry out draws for each distance weighted sub-sample of observations. These robust estimates replace the least-squares estimates from the GWR. This is similar to the method we used to implement the logit and probit variants of the GWR model in Section 6.5.

The role of the v_i terms in the BGWRV model is to downweight aberrant observations by adjusting the distance-based weight matrix W_i. This will be

Table 7.2: Casetti versus Bayesian expansion estimates

Variable	Casetti	$t-$stat	Bayesian	$t-$stat
constant	-0.2074	-3.34	-0.1101	-2.27
crime*	-0.0649	-1.18	-0.0643	-1.45
zoning*	0.1211	1.18	0.0596	0.74
industry	0.1524	1.74	0.0254	0.37
charlesr	0.0998	2.30	-0.0790	-1.98
noxsq	-0.4777	-6.72	-0.2705	-4.78
rooms2*	0.0659	1.67	0.0853	1.69
houseage	-0.1326	-1.52	-0.2235	-3.06
distance	-0.8867	-8.17	-0.5032	-5.68
access	0.3961	4.08	0.2083	2.85
taxrate*	-0.1381	-1.22	-0.1941	-2.47
pupil/teacher	-0.4195	-7.69	-0.3232	-7.48
blackpop*	0.1115	2.59	0.0661	1.76
lowclass	-0.5804	-11.46	-0.3847	-6.73
d-crime	-0.3170	-1.63	-0.1963	-1.27
d-zoning*	-0.0301	-0.62	0.0079	0.21
d-industry	-0.1487	-1.71	-0.0376	-0.55
d-charlesr	-0.0434	-1.02	0.1031	2.77
d-noxsq	0.2228	2.63	0.1353	2.09
d-rooms2*	0.3432	9.57	0.3152	8.11
d-houseage	-0.0388	-0.70	0.0271	0.59
d-distance	0.2108	4.04	0.1412	3.42
d-access	-0.1019	-1.11	0.0348	0.50
d-taxrate	-0.0842	-1.09	0.0031	0.05
d-pupil/teacher*	0.1630	4.57	0.1230	4.74
d-blackpop	0.0038	0.04	0.0846	1.22
d-lowclass	0.3421	6.49	0.2208	4.65
R^2	0.8090		0.7625	

made clear when we examine the conditional distributions for the parameters β_i, σ_i and V_i in this model.

The conditional posterior distribution of β_i given σ_i, V_i is a multivariate normal shown in (7.14).

$$
\begin{aligned}
p(\beta_i | \ldots) &\propto N(\hat{\beta}_i, \sigma^2 R) \\
\hat{\beta}_i &= R(\tilde{X}_i' V_i^{-1} \tilde{y}_i) \\
R &= (\tilde{X}_i' V_i^{-1} \tilde{X}_i)^{-1} \\
\tilde{X}_i &= V_i^{1/2} W_i X
\end{aligned}
\tag{7.14}
$$

To see the role of the variance scalars in the diagonal matrix V_i, we apply the definition of $\tilde{X}_i$ and expand R from (7.14) to produce:

$$
R = (X' W_i^{1/2} V_i^{-1} W_i^{1/2} X)^{-1} \tag{7.15}
$$

The distance-based weights in the diagonal matrix W_i will be divided by the variance scalars in the diagonal matrix V_i. Large values in V_i associated with outliers will produce smaller distance weights for these observations, effectively downweighting their influence in determining the estimate β_i. We will elaborate further on this point when we examine the conditional posterior distribution for the V_i terms below.

The conditional distribution for σ_i assuming β_i and V_i are known takes the form of a $\chi^2(n^\star)$ distribution, where $n^\star$ denotes observations associated with non-zero weights in W_i.

$$\begin{aligned} P\{[\sum_{j=1}^{n^\star}(u_j^2/v_{ij})/\sigma_i^2] \mid \beta_i, V_i\} &\sim \chi^2(n^\star) \\ u_j &= W_i^{1/2}y_j - W_i^{1/2}X_j\beta_i \end{aligned} \tag{7.16}$$

This result follows that for the homoscedastic GWR model, with two differences. First, the errors u_j are adjusted for non-constant variance using the known v_{ij}. Second, the degrees of freedom in the posterior conditional distribution is $n^\star$ since some of the weights in W_i may be zero, effectively eliminating these observations.

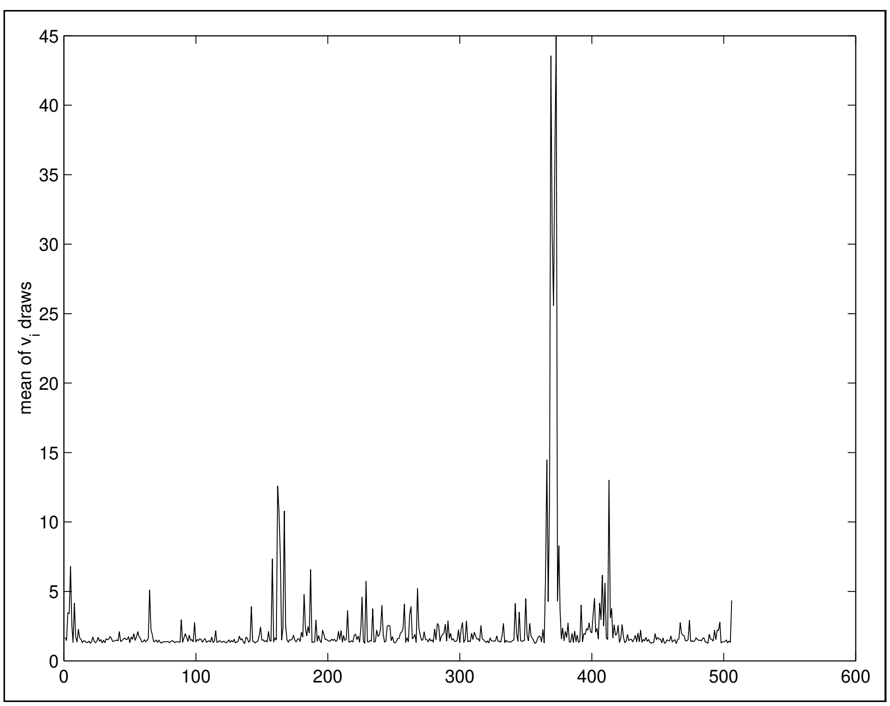

Figure 7.5: v_i estimates for Boston

Finally, the conditional posterior distribution for V_i is shown in (7.17), where u_j is defined as in (7.16).

$$P\{[(u_j^2/\sigma_i^2) + r]/v_{ij} \mid \beta_i, \sigma_i\} \sim \chi^2(r+1) \tag{7.17}$$

To see the role of the parameter v_{ij}, consider two cases. First, suppose (u_j^2/σ_i^2) is small (say zero), because the GWR distance-based weights work well to relate y and X for observation j. In this case, observation j is not an outlier. Assume that we use a small value of the hyperparameter r, say $r = 5$, which means our prior belief is that heterogeneity exits. The conditional posterior will have a mean and mode of:

$$\begin{aligned} \text{mean}(v_{ij}) &= (\sigma_i^{-2}u_j^2 + r)/(r+1) = r/(r+1) = (5/6) \\ \text{mode}(v_{ij}) &= (\sigma_i^{-2}u_j^2 + r)/(r-1) = r/(r-1) = (5/4) \end{aligned} \tag{7.18}$$

Where the results in (7.18) follow from the fact that the mean of the prior distribution for V_{ij} is $r/(r-2)$ and the mode of the prior equals $r/(r+2)$.

In the case shown in (7.18), the impact of $v_{ij} \approx 1$ in the model is neglible, and the typical distance-based weighting scheme would dominate. For the case of exponential weights, a weight, $w_{ij} = \exp(-d_i/\theta)/v_{ij}$ would be accorded to observation j. Note that a prior belief in homogeneity that assigns a large value of $r = 20$, would produce a similar weighting outcome. The conditional posterior mean of $r/(r+1) = 20/21$, is approximately unity, as is the mode of $(r+1)/r = 20/19$.

Second, consider the case where (u_j^2/σ_i^2) is large (say 20), because the GWR distance-based weights do not work well to relate y and X for observation j. Here, we have the case of an outlier for observation j. Using the same small value of the hyperparameter $r = 5$, the conditional posterior will have a mean and mode of:

$$\begin{aligned} \text{mean}(v_{ij}) &= (20 + r)/(r+1) = (25/6) \\ \text{mode}(v_{ij}) &= (20 + r)/(r-1) = (25/4) \end{aligned} \tag{7.19}$$

For this aberrant observation case, the role of $v_{ij} \approx 5$ will be to downweight the distance associated with this observation. The distance-based weight, $w_{ij} = \exp(-d_i)/\theta v_{ij}$ would be deflated by a factor of approximately 5 for this abberant observation. It is important to note that, a prior belief of homogeneity (expressed by a large value of $r = 20$) in this case would produce a conditional posterior mean of $(20 + r)/(r + 1) = (40/21)$. Downweighting of the distance-based weights would be only by a factor of 2, rather than 5 found for the smaller value of r.

It should be clear that as r becomes very large, say 50 or 100, the posterior mean and mode will be close to unity irrespective of the fit measured by u_j^2/σ_i^2 This replicates the distance-based weighting scheme used in the non-Bayesian GWR model.

A graphical illustration of how this works in practice can be seen in Figure 7.6. The figure depicts the adjusted distance-based weights, $W_i V_i^{-1}$ alongside the GWR weights W_i for observations 31 to 36 in the Anselin (1988) Columbus neighborhood crime data set. We saw in section 7.1 that the v_i estimate for observation #34 produced by the Bayesian spatial expansion model was large, indicating an outlier (see Figure 7.2).

Beginning with observation #31, the aberrant observation #34 is downweighted when estimates are produced for observations 31 to 36 (excluding observation #34 itself). A symbol 'o' has been placed on the BGWR weight in the figure to help distinguish observation 34. This downweighting of the distance-based weight for observation #34 occurs during estimation of β_i for observations 31 to 36, all of which are near #34 in terms of the GWR distance measure. It will be seen that this alternative weighting produces a divergence in the BGWR and GWR estimates for observations neighboring on #34.

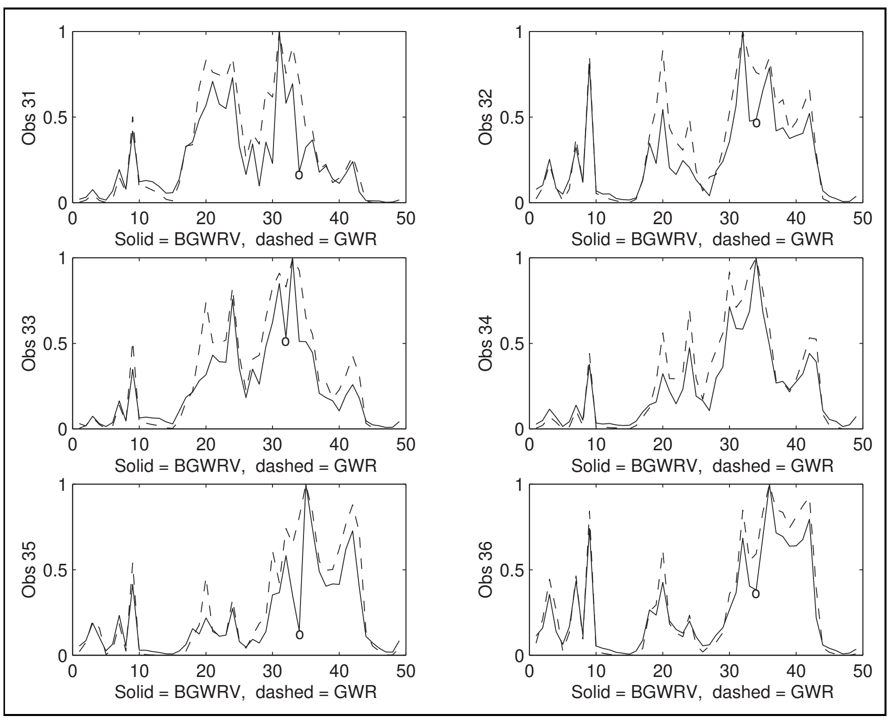

Figure 7.6: Distance-based weights adjusted by V_i

7.2.1 Gibbs sampling BGWRV estimates

A function **bgwrv** was designed to implement the Gibbs sampling estimation of the BGWRV model. The documentation is shown below along with the code that implements the Gibbs sampler.

```
% PURPOSE: compute Bayesian robust geographically weighted regression
%          Wi*y = Wi*X*bi + ei, ei is N(0,sige*Vi)
%          Vi = diag(v1i,v2i,...vni),
%          r/vi = ID chi(r)/r, r = Gamma(m,k)
%---------------------------------------------------
% USAGE: results = bgwrv(y,x,east,north,ndraw,nomit,info)
% where:   y = dependent variable vector
%          x = explanatory variable matrix
%       east = x-coordinates in space
%      north = y-coordinates in space
%      ndraw = # of draws
%      nomit = # of initial draws omitted for burn-in
%       info = a structure variable with fields:
%       info.rval, r prior hyperparameter, default=4
%       info.bwidth = scalar bandwidth to use or zero
%                     for cross-validation estimation (default)
%       info.dtype  = 'gaussian'    for Gaussian weighting (default)
%                   = 'exponential' for exponential weighting
%                   = 'tricube'     for tri-cube weighting
%       info.q      = q-nearest neighbors to use for tri-cube weights
%                     (default: CV estimated)
%       info.qmin   = minimum # of neighbors to use in CV search
%       info.qmax   = maximum # of neighbors to use in CV search
%                     defaults: qmin = nvar+2, qmax = 4*nvar
% ---------------------------------------------------
%  NOTE: res = bgwrv(y,x,east,north) does CV estimation of bandwidth
% ---------------------------------------------------
% RETURNS: a results structure
%        results.meth  = 'bgwrv'
%        results.bdraw = beta draws (ndraw-nomit x nobs x nvar)
%        results.smean = mean of sige draws (nobs x 1)
%        results.vmean = mean of vi draws (nobs x 1)
%        results.nobs  = nobs
%        results.nvar  = nvars
%        results.ndraw = ndraw
%        results.nomit = nomit
%        results.rval  = rval (from input)
%        results.bwidth= bandwidth if gaussian or exponential
%        results.q     = q nearest neighbors if tri-cube
%        results.dtype = input string for Gaussian, exponential weights
%        results.iter  = # of simplex iterations for cv
%        results.ycoord = north (y-coordinates)
%        results.xcoord = east  (x-coordinates)
%        results.y     = y data vector
%        results.x     = x data matrix
%---------------------------------------------------
dmat = zeros(nobs,nobs); % generate big distance matrix
    for j=1:nobs;
        easti = east(j,1); northi = north(j,1);
dx = east - easti; dy = north - northi;
d = dx.*dx + dy.*dy;
        dmat(:,j) = d;
    end;
wt = zeros(nobs,nobs);   % generate distance decay matrix
if dtype == 1,           % exponential weights
        wt = exp(-dmat/bdwt);
```

```
elseif dtype == 0,       % gaussian weights
        sd = std(sqrt(dmat)); tmp = matdiv(sqrt(dmat),sd*bdwt);
        wt = stdn_pdf(tmp);
elseif dtype == 2
% case of tricube weights handled a bit differently
   % sort distance to find q nearest neighbors
ds = sort(dmat); dmax = ds(q+1,:);
        for j=1:nobs;
    nzip = find(dmat(:,j) <= dmax(1,j));
        wt(nzip,j) = (1-(dmat(nzip,j)/dmax(1,j)).^3).^3;
        end; % end of j-loop
end;  % end of if-else
wt = sqrt(wt);
% storage for estimates
bsave = zeros(ndraw-nomit,nobs,nvar); smean = zeros(nobs,1);
vmean = ones(nobs,nobs);  prior.rval = rval;
t0 = clock;
for i=1:nobs
 nzip = find(wt(:,i) > 0.01);
 ys = y(nzip,1).*wt(nzip,i); xs = matmul(x(nzip,:),wt(nzip,i));
 res = gwr_g(ys,xs,ndraw,nomit,prior);
 bsave(:,i,:) = res.bdraw;
 vmean(i,nzip) = vmean(i,nzip) + res.vmean';
 smean(i,1) = mean(res.sdraw);
end; % end loop over nobs
gtime = etime(clock,t0);
vout = mean(vmean);
```

We rely on a bandwidth or distance decay parameter determined by optimizing the cross-validation score as in the case of the GWR model, so this code is not shown. This may not be a good strategy since the existence of outliers will impact the bandwidth estimate for the parameter θ. An area for future research would be devising a way to use Metroplis sampling to estimate a bandwidth parameter.

The sampling work is done by a function **gwr_g**, which implements a sampler for the heteroscedastic linear regression model that we introduced in example 4.1 of Chapter 4.

The code relies on the computational trick introduced in Chapter 6 where we employ only observations with non-negligible weights to scale down the size of the matrices used during sampling. This works very well for the case of tri-cube and exponential weighting to reduce the size of the problem we need to sample. Consider that even in a sample of thousands of observations, the sampling problem would involve a reasonably small number of observations, probably less than 30 for the case of tri-cube weights.

A three-dimensional MATLAB matrix structure is used to return the draws for the parameters that form an n by k matrix for each draw. We will show how to compute posterior means using this three-dimensional matrix in example 7.3.

In Chapter 6 we saw that for models involving a large number of observations such as the Pace and Barry data set, it took around 200 seconds for the **gwr** function to cycle through the observations with a given bandwidth and compute GWR estimates. (Recall that it took an hour to find the optimal cross-validation bandwidth for that model.) The prospects for Gibbs sampling estimation may seem bleak, but a few experiments indicate otherwise.

Experiments indicated that only 550 observations are needed to achieve convergence with the first 50 omitted for burn-in. The time needed to produce 550 draws is a function of the number of observations containing non-negligible weights and the variables in the model. Figure 7.7 shows a plot of the time needed to produce 550 draws against the number of effective observations from the Boston data set that contains 506 observations and six variables. An exponential weighting method was used so the number of non-negligible observations varies over the sample. The times ranged from 3.2 seconds for a 40 observation sample to 18.4 seconds for a sample of 389 observations. The graph makes it quite clear that execution times are a linear function of the number of observations. Another experiment with the 49 observations and three variables in the Columbus neighborhood crime data set produced with exponential weights resulted in samples ranging from 6 observations that took 1.5 seconds for 550 draws up to a sample with 39 observations that required 3.2 seconds.

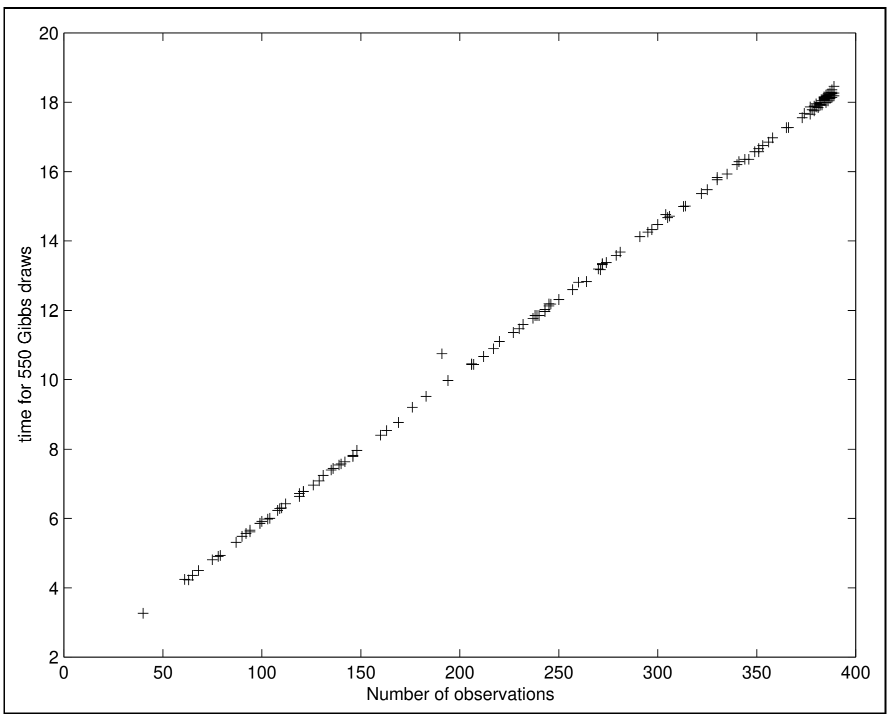

Figure 7.7: Observations versus time for 550 Gibbs draws

One way to increase the speed of the sampling process is to rely on the tricube weighting method. For the 506 observation Boston data set, the optimal number of nearest neighbors was 24, so we need only work with a sample size of 24 when carrying out the Gibbs draws. With this size sample, it took 2.5 seconds to generate 550 draws. A desirable aspect of the tri-cube in this setting is that the sample size is fixed by the optimal number of nearest neighbors throughout the Gibbs sampling process. Of course with 3,000 observations this is still 7,500 seconds or a 125 minutes. Another caveat is that tri-cube weighting tends to reduce the sample size which makes it difficult to identify outliers. Some experience with the Boston sample indicates that either this weighting method adjusts very rapidly or the sample size is too small, because all v_i estimates are near unity. Further, we know that v_i estimates near unity are unreasonable for the Boston data set. Given v_i estimates equal to unity, the BGWRV estimates collapse to GWR estimates.

The worst weighting method in terms of execution time is the Gaussian, where more observations receive non-negligible weight, so the time required for 550 draws per observation can be prohibitive in large samples.

7.2.2 Applied examples

We apply the function **bgwrv** to the Anselin (1988) neighborhood crime data set in example 7.3, where estimates from a GWR model are compared to those from two BGWRV models, one with a homoscedastic prior based on $r = 40$ and another based on the heteroscedastic prior with $r = 4$.

We would expect the estimates from GWR and BGWRV to be almost identical for the homoscedastic prior based on $r = 40$, whereas outliers or non-constant variance might lead to differences for the heteroscedastic prior based on $r = 4$.

```
% ----- Example 7.3 Using the bgwrv() function
load anselin.dat; % load Anselin (1988) Columbus neighborhood crime data
y = anselin(:,1); n = length(y);
x = [ones(n,1) anselin(:,2:3)];
% Anselin (1988) x-y coordinates
xc = anselin(:,4); yc = anselin(:,5);
vnames = strvcat('crime','const','income','hse value');
% do GWR regression using exponential weights
info.dtype = 'exponential';
res1 = gwr(y,x,xc,yc,info);
ndraw = 1250; nomit = 250;
option.dtype = 'exponential';
option.bwidth = res1.bwidth;
option.rval = 40; % homoscedastic estimates
res2 = bgwrv(y,x,xc,yc,ndraw,nomit,option);
option.rval = 4;  % heteroscedastic estimates
res3 = bgwrv(y,x,xc,yc,ndraw,nomit,option);
% compare gwr estimates with posterior means
b1 = res2.bdraw(:,:,1); c1 = res3.bdraw(:,:,1);
b2 = res2.bdraw(:,:,2); c2 = res3.bdraw(:,:,2);
b3 = res2.bdraw(:,:,3); c3 = res3.bdraw(:,:,3);
b1mean = squeeze(mean(b1)); c1mean = squeeze(mean(c1));
b2mean = squeeze(mean(b2)); c2mean = squeeze(mean(c2));
b3mean = squeeze(mean(b3)); c3mean = squeeze(mean(c3));
tt=1:n;
subplot(2,2,1),
plot(tt,res1.beta(:,1),tt,b1mean,'ok',tt,c1mean,'--k');
```

```
xlabel('constant');  title('exponential weights');
subplot(2,2,2),,
plot(tt,res1.beta(:,2),tt,b2mean,'ok',tt,c2mean,'--k');
xlabel('income'); legend('GWR','Bayes r=40','Bayes r=4');
subplot(2,2,3),
plot(tt,res1.beta(:,3),tt,b3mean,'ok',tt,c3mean,'--k');
xlabel('house');
subplot(2,2,4),
plot(tt,res3.vmean);
ylabel('mean of the v_i draws');
xlabel(`observations');
```

In example 7.3, we compute the means of the draws for the parameter estimates returned by the **bgwrv** function for comparison with the GWR estimates. These are returned in a structure field that represents a three-dimensional matrix, which requires some special handling. Notice how each vector of n parameter estimates is extracted from the 3-dimensional results structure field '.bdraw'. The first dimension contains draws, the second observations and the third parameters. We reference all elements in the first two dimensions and a single element in the third dimension, which yields all draws for all observations of a single parameter.

If we wished to obtain a matrix containing all draws for observation 1, for all parameters we would use the code shown below to reference all elements in the first and third dimensions, but only the initial element in the second dimension, which contains observations.

```
bmat = res2.bdraw(:,1,:);
```

Similarly, we could obtain the first draw in the results structure for all n observations of the first parameter by extracting elements from the first element of the first and third dimensions as shown below.

```
bmat = res2.bdraw(1,:,1);
```

In the example we use the MATLAB **squeeze** function to eliminate a 'singleton' dimension in the variables 'b1,b2,b3' and 'c1,c2,c3' that will remain after executing the MATLAB **mean** function. The results from applying the **mean** function will be a 3-dimensional matrix, but one of the dimensions will be collapsed to a single element by the averaging operation.

Figure 7.8 shows the comparison of GWR and BGWRV estimates from the heteroscedastic version of the model. First, we note that since the homoscedastic prior value of $r = 40$ produced estimates nearly identical to the GWR model, we eliminated these from the graph. For the heteroscedastic prior based on $r = 4$, we see definite evidence of a departure between the BWR and BGWRV estimates. The v_i estimates are also presented in the graph and they point to observations 2,4, 20 and 34 as outliers.

In the case of all three parameters we see a pattern where the GWR and BGWRV estimates follow very different trajectories around observation 34. Recall this was attributed to the aberrant observation 34, which happens to be a central city observation. The large v_i estimate for this observation clearly points to it is as an outlier.

An examination of crime incidents, household income and house values for observation 34 shows that this neighborhood has the highest crime rate, an income of 13,906 dollars near the median of 13,380 and a house value of 22,500 dollars below the median of 33,500. Because the dependent variable for this observation represents the highest value in the sample, while the independent variables do not take on extreme values, this observation is a good candidate as an aberrant observation.

Another place where the GWR and BGWRV income coefficient estimates differ substantially is around observations 2 and 4, where we see small spikes in the v_i estimates. Neighborhood 4 has the lowest crime rate in the sample, the fifth highest house value but the sixth lowest household income. Again, we have an extreme value for the dependent variable associated with contrary values for the independent variables.

In general, a dramatically different trajectory for the the two sets of estimates begins around observation #20 and continues to observation #44. The v_i estimates shown in the fourth panel of Figure 7.8 provide an explanation for

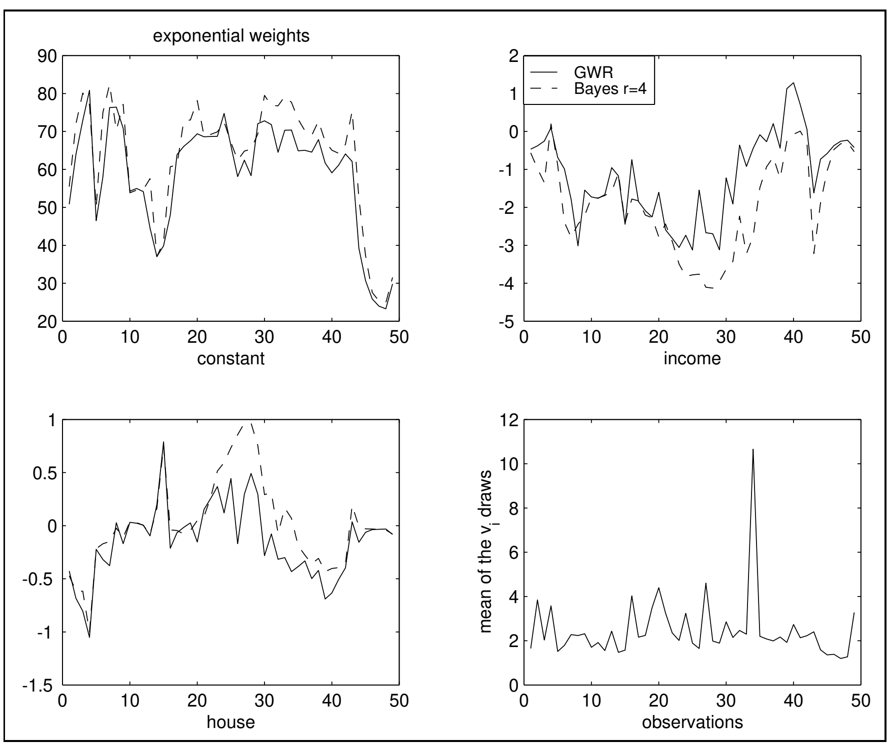

Figure 7.8: GWR versus BGWRV estimates for Columbus data set

these differences.

An interesting question is — are these differences significant in a statistical sense? We can answer this question using the 1,000 draws produced by the program in example 7.3 to compute a two standard deviation band around the BGWRV estimates. If the GWR estimates fall within this confidence interval, we would conclude the estimates are not significantly different. Figure 7.9 shows the GWR estimates and the confidence bands for the BGWRV estimates. The actual BGWRV estimates were omitted from the graph for clarity. We see that the GWR estimates are near or outside of the two standard deviation confidence interval for observations 20 through 44 where the two sets of estimates take different trajectories. This implies we would draw different inferences from the GWR and BGWRV estimates for these observations which represent a large part of the sample.

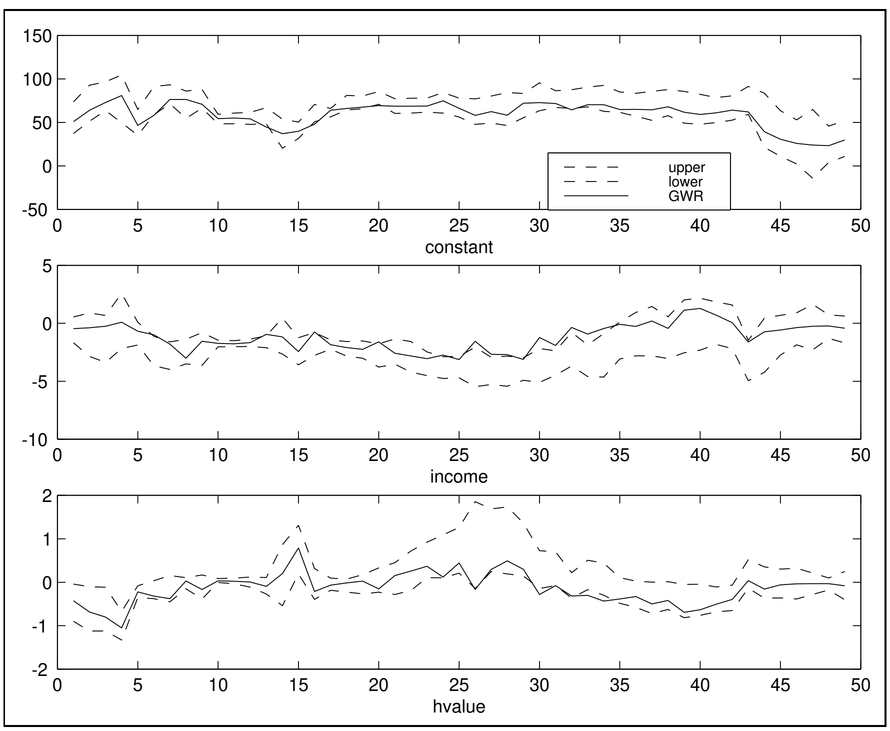

Figure 7.9: GWR versus BGWRV confidence intervals

As a test for convergence, we produced another set of estimates based on only 550 draws with the first 50 omitted for burn-in. It seemed possible that the BGWRV model might converge quickly since it relies on the cross-validation estimate for the bandwidth. Since Gibbs sampling these models is computationally intensive, rapid convergence would be a fortunate circumstance. A graph of the two sets of estimates based on 500 draws and 1000 draws is shown in

Figure 7.10. We see evidence near the beginning and end of the sample that convergence was not obtained with the smaller sample of 550 draws.

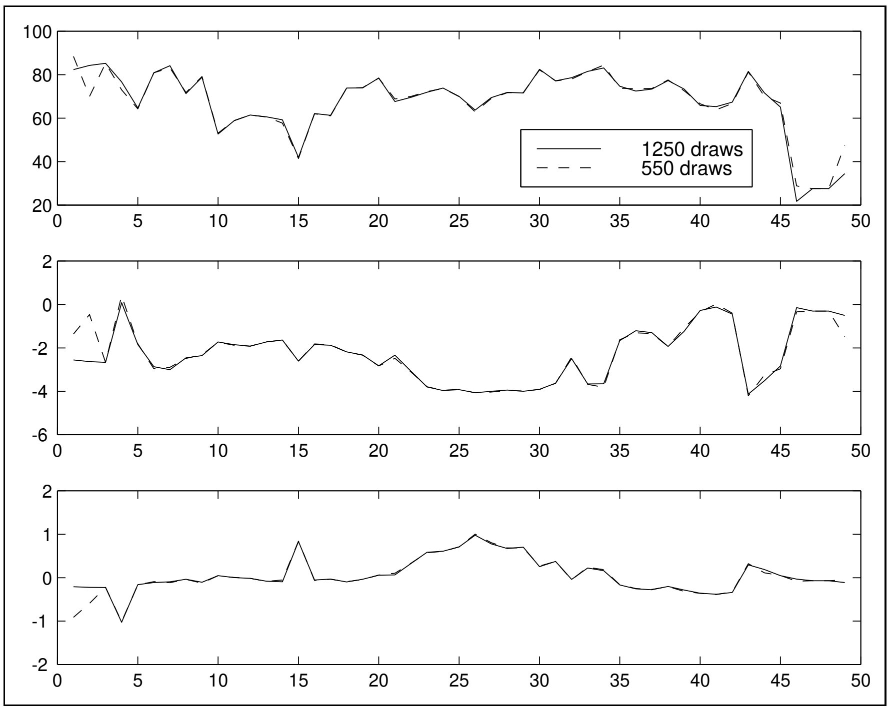

Figure 7.10: GWR versus BGWRV estimates

A generated data set was used to demonstrate the problems created by outliers for the GWR model, and to show that Bayesian robust estimates do not suffer from these problems. The program in example 7.4 generates a variable y using coefficients that vary over a regular grid according to the quadrant in which the observation falls. Coefficients of 1 and -1 were used for two explanatory variables.

After producing GWR estimates based on this data set, we introduce a single outlier at observation #60 created by multiplying the explanatory variables by 10. Another set of GWR estimates along with BGWRV model estimates are produced using this outlier contaminated data set. If the BGWRV model is producing robust estimates, we would expect to see estimates that are similar to those from the GWR model based on the data set with no outliers.

```
% ----- Example 7.4 GWR and BGWRV comparison with generated data
randn('seed',10); n = 101; k = 2; evec = randn(n,1);
xc = -50:50; xc = xc'; yc = -50:50; yc = yc';
x = randn(n,2); y = zeros(n,1);
for i=1:n
 if (xc(i,1) < 0 & yc(i,1) < 0)
```

```
  y(i,1) = x(i,1) + x(i,2) + evec(i,1)*0.2;
 elseif (xc(i,1) < 0 & yc(i,1) >= 0)
  y(i,1) = x(i,1) -x(i,2) + evec(i,1)*0.2;
 elseif (xc(i,1) > 0 & yc(i,1) < 0)
  y(i,1) = -x(i,1) + x(i,2) + evec(i,1)*0.2;
 elseif (xc(i,1) >= 0 * yc(i,1) >= 0)
  y(i,1) = -x(i,1) - x(i,2) + evec(i,1)*0.2;
 end;
end;
info.dtype = 'exponential';
res1 = gwr(y,x,xc,yc); % GWR estimates for clean data
x(60,:) = 10*x(60,:);  % introduce an outlier
res2 = gwr(y,x,xc,yc); % GWR estimates for dirty data
ndraw = 1100; nomit = 100;
info.rval = 4; info.bdwidth = res2.bwidth;
res3 = bgwrv(y,x,xc,yc,ndraw,nomit,info); % BGWRV estimates
tmp = mean(res3.bdraw); bout = squeeze(tmp);
[n k] = size(bout);      beta1 = zeros(n,k);
for i=1:k;  beta1(:,i) = bout(:,i); end;
tt=1:n;
subplot(2,1,1), plot(tt,res1.beta(:,1),'ok',tt,res2.beta(:,1),'-k', ...
tt,beta1(:,1),'--k');
xlabel('coefficient 1');
subplot(2,1,2), plot(tt,res1.beta(:,2),'ok',tt,res2.beta(:,2),'-k', ...
tt,beta1(:,2),'--k');
xlabel('coefficient 2');
legend('GWR no outlier','GWR outlier','BGWRV outlier');
pause;
subplot(2,1,1),
plot(tt,res3.vmean,'-k');
ylabel('vi estimates');
xlabel('observations');
subplot(2,1,2),
plot(tt,res1.sige,'ok',tt,res2.sige,'-k',tt,res3.smean,'--k');
ylabel('sige estimates');
xlabel('observations');
legend('GWR no outlier','GWR outlier','BGWRV outlier');
% compare t-statistics
tmp = std(res3.bdraw); tout = squeeze(tmp);
[n k] = size(tout);      tbeta1 = zeros(n,k);
for i=1:k;  tbeta1(:,i) = beta1(:,i)./tout(:,i); end;
subplot(2,1,1), plot(tt,res1.tstat(:,1),'ok',tt,res2.tstat(:,1),'-k', ...
tt,tbeta1(:,1),'--k');
xlabel('t-statistic coefficient 1');
subplot(2,1,2), plot(tt,res1.tstat(:,2),'ok',tt,res2.tstat(:,2),'-k', ...
tt,tbeta1(:,2),'--k');
xlabel('t-statistic coefficient 2');
legend('GWR no outlier','GWR outlier','BGWRV outlier');
pause;
```

Another issue we would like to explore with this experiment is the quality of the inferences from the GWR and BGWRV models when outliers exist. A graph of the estimates for σ^2 from the three models is produced for comparison. If the BGWRV model is producing robust σ^2 estimates, these parameter values should be similar to those from the GWR based on the non-contaminated data set. A similar comparison was made of the $t-$statistics from both GWR models

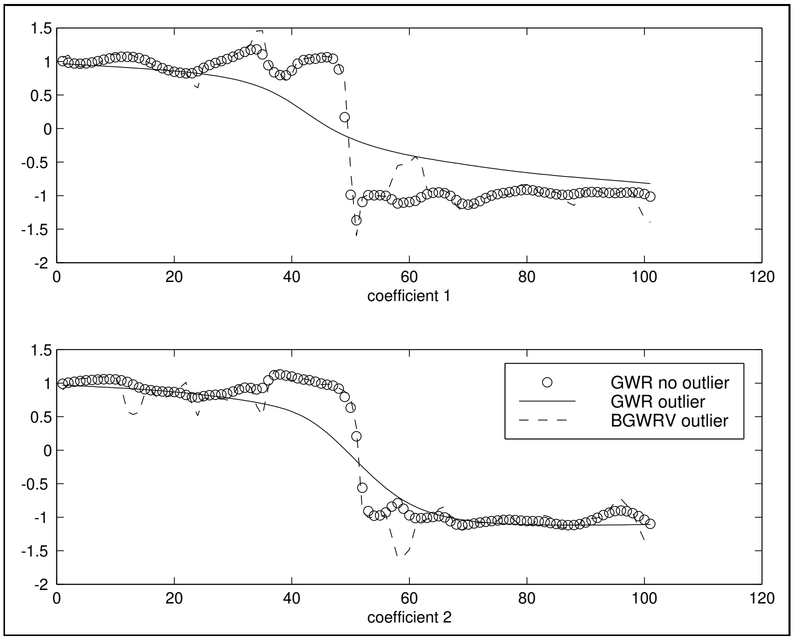

Figure 7.11: β_i estimates for GWR and BGWRV with an outlier

and the BGWRV model. Of course, BGWRV $t-$statistics were constructed from the posterior means and standard deviations of the MCMC draws.

Figure 7.11 shows the three sets of β_i estimates produced by the three models. The impact of the single outlier at observation #60 on the GWR estimates is quite evident. By comparison, the BGWRV model produced estimates almost identical to those from the GWR model based on the data set without the outlier. Note that the goal of GWR estimation would be to detect changes in the parameters of the relationship as we move over space. In this example, the impact of a single outlier virtually destroys our ability to detect the transition in the parameters from 1 to -1 at observation #50. In the absence of the outlier, the GWR model does a good job of detecting the parameter transition point, and the BGWRV model does an equally good job in the presence of the outlier.

In addition to differences in the parameters β_i, we are also be interested in estimates of σ_i and the $t-$statistics used to draw inferences about β_i. The estimates for σ_i^2 from all the three models are shown in Figure 7.12 along with the v_i estimates from the BGWRV model. We see that the single outlier contaminates the variance estimate for the noise in the model dramatically. As in the case of the estimates for β_i, the BGWRV model reproduces the GWR estimates based on the clean data set. Note that the v_i estimates clearly point to observation #60 as an outlier, confirming the diagnostic value of these parameter estimates.

Finally, a comparison of the $t-$statistics from the three models is shown in

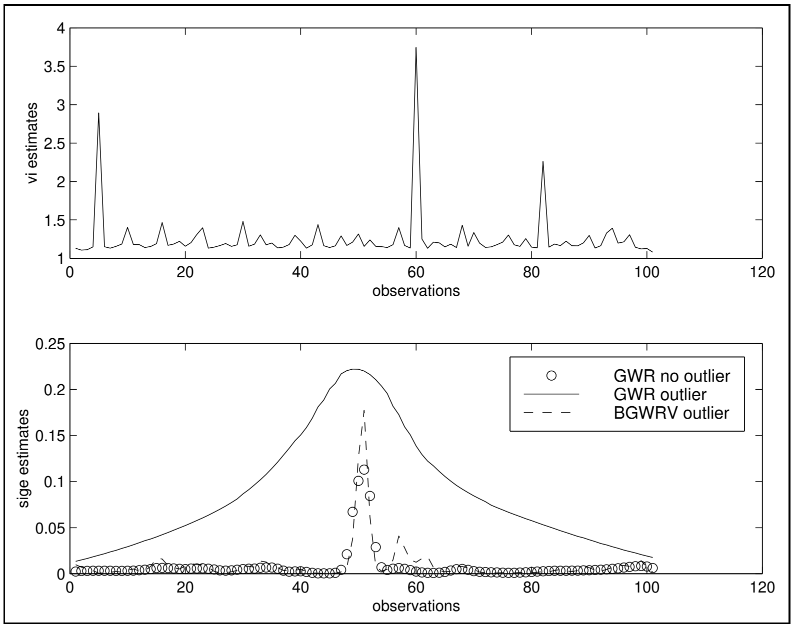

Figure 7.12: σ_i and v_i estimates for GWR and BGWRV with an outlier

Figure 7.13. These results indicate that not only are the β_i estimates adversely impacted by the single outlier, but the $t-$statistics which we would use to draw inferences about the β_i are seriously impacted as well. For coefficients near the transition point at observation #50, the GWR $t-$statistics from the contaminated data set indicate a slow transition from positive to negative values resulting in very imprecise estimates during the transition of the coefficients from 1 to -1. In contrast, the GWR estimates based on the data set with no outlier and the BGWRV estimates both move rapidly from positive negative, exhibiting a great deal more precision during the transition period.

If the goal of locally linear estimation is to make inferences regarding spatial variation in the relationship, contamination from outliers will lead to an erroneous conclusion that the relationship is changing. In fact the relationship may be stable but subject to the influence of a single outlying observation. In contrast, the BGWRV estimates indicate changes in the parameters of the relationship as we move over space that abstract from aberrant observations. From the standpoint of inference, we can be relatively certain that changing BGWRV estimates truly reflect a change in the underlying relationship as we move through space. In contrast, the GWR estimates are more difficult to interpret, since changes in the estimates may reflect spatial changes in the relationship, or the presence of an aberrant observation. Keep in mind that the amount of robustness achieved with the BGWRV method can be controlled using the hyperparameter r

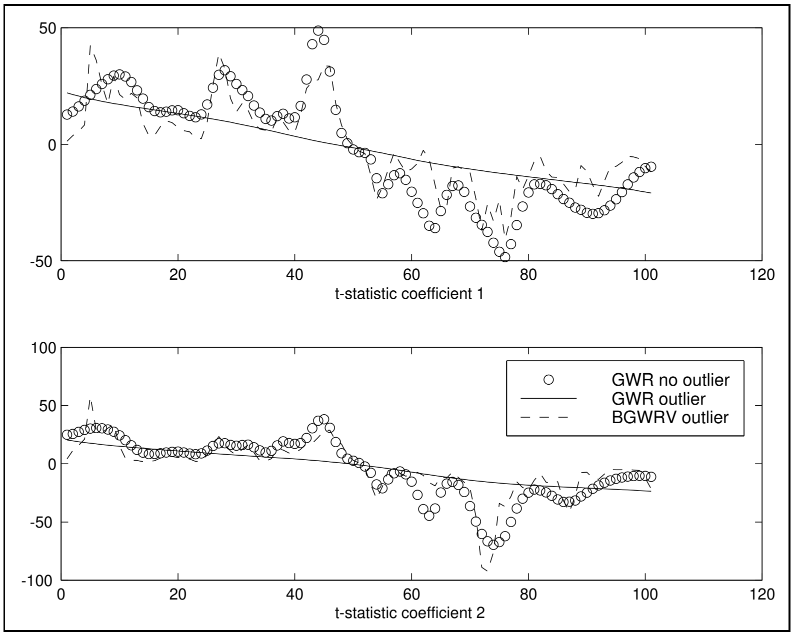

Figure 7.13: $t-$statistics for the GWR and BGWRV with an outlier

One drawback to the BGWRV method is that it may produce estimates that are too robust and exhibit very little variation over space. The only control one has over this is use of alternative hyperparameter settings. The rule-of-thumb value of $r = 4$ does not have the universal applicability it did in the case of Bayesian spatial autoregressive models. This may be due to the use of a bandwidth determined by GWR cross-validation methods. In the face of outliers, the bandwidth estimate produced by cross-validation may not be very good. In Section 7.3 we extend the BGWRV model to allow more control over these aspects of the model.

7.2.3 A Bayesian probit GWR model

As in the case of the GWR model, we can implement robust Bayesian logit/probit models. A function **probit_g** in the *Econometrics Toolbox* implements a heteroscedastic version of the logit/probit model for standard regression models. Replacing the function **gwr_g** that carries out Gibbs sampling estimation of the heteroscedastic linear regression model results in a logit/probit variant of the BGWRV model.

Because this development parallels the development in Chapter 6, we leave it to the reader to examine the function **bgwrv_p** that implements this method. One point is that the Bayesian heteroscedastic probit model subsumes the logit model as a special case where the prior hyperparameter r is around 7. Larger values of $r = 30$ produce estimates similar to the probit model, and other values of r generate a family of models that can be interpreted as alternative link functions based on $t-$distributions with varying degrees of freedom. For this reason, there is no corresponding logit function. Given the computational intensity of the GWR logit and probit models, Gibbs sampling is quite competitive in terms of computational time needed.

Another point is that using the Gibbs sampling function **probit_g** during cross-validation would require excessive computation, so ordinary logit or probit regressions are used. Logit regression is used for hyperparameter values of r less than 20 and probit for values greater than 20.

7.3 Extending the BGWR model

We can extend the BGWRV model from the previous section to include prior information regarding the parameter smoothing relationship over space. Bayesian methods are ideally suited to this task because they allow us to introduce a stochastic parameter smoothing relationship as prior information in the model. If we have knowledge that the problem we are modeling represents the case of a "concentric city", or a problem where contiguity should be dominant, this approach can be used to augment the sample data information.

The approach which we label BGWR is best described using matrix expresssions shown in (7.20) and (7.21). First, note that (7.20) is the same as the non-parametric GWR relationship, but the addition of (7.21) provides an explicit statement of the parameter smoothing that takes place across space. The parameter smoothing relation shown in (7.21) involves a locally linear combination of neighboring areas, where neighbors are defined in terms of the GWR distance weighting function that decays over space. We will refer this this as a distance smoothing relationship to distinguish it from other smoothing relations we introduce later.

$$W_i y = W_i X \beta_i + W_i \varepsilon_i \tag{7.20}$$

$$\beta_i = \begin{pmatrix} w_{i1} \otimes I_k & \ldots & w_{in} \otimes I_k \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} + u_i \tag{7.21}$$

The terms w_{ij} represent a distance-based weight that is normalized so the row-vector $(w_{i1}, \ldots, w_{in})$ sums to unity, and we set $w_{ii} = 0$. Specifically, $w_{ij} = \exp(-d_{ij}/\theta)/\sum_{j=1}^{n} \exp(-d_{ij}/\theta)$, for the case of exponential weighting. Analogous definitions can be derived for Gaussian and tri-cube weighting methods. By normalizing so the weights sum to unity, the prior indicates that we believe the parameters for location i should reflect a linear combination of parameters from nearby observations. We omit observation i from the weights by setting it to zero, so it is not part of the linear combination.

A point to note is that the parameter smoothing relationship in (7.21) is stochastic by virtue of the term u_i that represents prior uncertainty about the smoothing relationship.

To complete our model specification, we add distributions for the terms ε_i and u_i:

$$\begin{aligned}
\varepsilon_i &\sim N[0, \sigma^2 V_i], \qquad V_i = \mathrm{diag}(v_1, v_2, \ldots, v_n) &(7.22)\\
u_i &\sim N[0, \sigma^2 \delta^2 (X' W_i X)^{-1})] &(7.23)
\end{aligned}$$

The $V_i = \mathrm{diag}(v_1, v_2, \ldots, v_n)$, represent our variance scaling parameters introduced in Section 7.2. The distribution for the stochastic parameter u_i in the parameter smoothing relationship is normally distributed with mean zero and a variance based on Zellner's (1971) $g-$prior. This prior variance is proportional to the GWR parameter variance-covariance matrix, $\sigma^2(X'W_iX)^{-1})$ with δ^2 acting as the scale factor. The use of this prior specification allows individual parameters β_i to vary by different amounts depending on their magnitude.

The parameter δ^2 acts as a scale factor to impose tight or loose adherence to the parameter smoothing specification. Consider a case where δ is very small, then the smoothing restriction would force β_i to look like a distance-weighted linear combination of other β_i from neighboring observations. On the other hand, as $\delta \to \infty$ (and $V_i = I_n$) we produce the non-parametric GWR estimates. To see this, we rewrite the BGWR model in a compact matrix form shown in (7.24).

$$\begin{aligned}
\tilde{y}_i &= \tilde{X}_i \beta_i + \varepsilon_i &(7.24)\\
\beta_i &= J_i \gamma + u_i &\\
& &(7.25)
\end{aligned}$$

Where the definitions of the matrix expressions are:

$$\begin{aligned}
\tilde{y}_i &= W_i^{1/2} y\\
\tilde{X}_i &= W_i^{1/2} X\\
J_i &= \begin{pmatrix} w_{i1} \otimes I_k & \ldots & w_{in} \otimes I_k \end{pmatrix}\\
\gamma &= \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix}
\end{aligned}$$

As indicated earlier, the notation is somewhat confusing in that $\tilde{y}_i$ denotes an $n-$vector, not a scalar magnitude. Similarly, ε_i is an $n-$vector and $\tilde{X}_i$ is an n by k matrix. Note that (7.24) can be written in the form of a Theil-Goldberger (1961) estimation problem as shown in (7.26).

$$\begin{pmatrix} \tilde{y}_i \\ J_i\gamma \end{pmatrix} = \begin{pmatrix} \tilde{X}_i \\ -I_k \end{pmatrix} \beta_i + \begin{pmatrix} \varepsilon_i \\ u_i \end{pmatrix} \tag{7.26}$$

Assuming $V_i = I_n$, the estimates β_i take the form:

$$\begin{aligned} \hat{\beta}_i &= R(\tilde{X}_i'\tilde{y}_i + \tilde{X}_i'\tilde{X}_i J_i\gamma/\delta^2) \\ R &= (\tilde{X}_i'\tilde{X}_i + \tilde{X}_i'\tilde{X}_i/\delta^2)^{-1} \end{aligned}$$

As δ approaches ∞, the terms associated with the Theil-Goldberger "stochastic restriction", $\tilde{X}_i'\tilde{X}_i J_i\gamma/\delta^2$ and $\tilde{X}_i'\tilde{X}_i/\delta^2$ become zero, and we have the GWR estimates shown below.

$$\hat{\beta}_i = (\tilde{X}_i'\tilde{X}_i)^{-1}(\tilde{X}_i'\tilde{y}_i) \tag{7.27}$$

In practice, an alternative to specifying the parameter δ using subjective prior information is to use a diffuse prior for δ and estimate this parameter. Unless the parameter smoothing relationship is fairly consistent with the sample data, the estimated value may be large, leading to BGWR estimates that are similar to those from the BGWRV model that doesn't include a parameter smoothing relationship. There are cases where we know the data is weak or seems counter to other knowledge we have about the relationship being modeled. In these cases, we might choose to rely on an informative prior for δ that imposes the parameter smoothing relationship fairly tightly in the model.

Details concerning estimation of the parameters in the BGWR model are taken up in the next section. Before turning to these issues, we consider some alternative spatial parameter smoothing relationships that might be used in place of (7.21) in the BGWR model.

One alternative smoothing specification would be a relationship that we label "monocentric city smoothing", set forth in (7.28). This relation assumes data observations are ordered by distance from the center of the spatial sample.

$$\begin{aligned} \beta_i &= \beta_{i-1} + u_i \\ u_i &\sim N[0, \sigma^2\delta^2(X'W_i^2X)^{-1}] \end{aligned} \tag{7.28}$$

Given that the observations are ordered by distance from the center, the smoothing relation indicates that β_i should be similar to the coefficient β_{i-1} from a neighboring concentric ring. We rely on the same GWR distance weighted sub-samples of the data created by the transformation: $W_i^{1/2}y, W_i^{1/2}X$. This means that the estimates still have a "locally linear" interpretation as in the GWR, but depending on how tightly we impose the prior smoothing relationship, estimates should be similar to those from neighboring concentric rings. The same distibutional assumption is used for the term u_i in this parameter smoothing relationship, which allows us to estimate the parameters making only minor changes to the approach used for the BGWR model in (7.20) and (7.21).

Another alternative is a "spatial expansion smoothing" based on the ideas introduced in Chapter 6. This is shown in (7.29), where Z_{xi}, Z_{yi} denote lattitude-longitude coordinates associated with observation i.

$$\begin{aligned} \beta_i &= \begin{pmatrix} Z_{xi} \otimes I_k & Z_{yi} \otimes I_k \end{pmatrix} \begin{pmatrix} \beta_x \\ \beta_y \end{pmatrix} + u_i \\ u_i &\sim N[0, \sigma^2\delta^2(X'W_i^2X)^{-1})] \end{aligned} \tag{7.29}$$

This parameter smoothing relation relates the parameters at a particular latitude-longitude coordinate to a locally linear combination of parameters from nearby latitude-longitude coordinates. As in the case of the monocentric city specification, we retain the same assumptions regarding the stochastic term u_i.

Finally, we could adopt a "contiguity smoothing" relationship based on a first-order spatial contiguity matrix as shown in (7.30). The terms c_{ij} represent the ith row of a row-standardized first-order contiguity matrix. This creates a parameter smoothing relationship that averages over the parameters from observations that neighbor observation i.

$$\begin{aligned} \beta_i &= \begin{pmatrix} c_{i1} \otimes I_k & \ldots & c_{in} \otimes I_k \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} + u_i \\ u_i &\sim N[0, \sigma^2\delta^2(X'W_i^2X)^{-1})] \end{aligned} \tag{7.30}$$

The contiguity smoothing relationship may be fairly similar to the distance smoothing relationship in (7.21), depending on the type of weighting method used and the particular data sample.

These alternative approaches to specifying a geographically weighted regression model suggest that researchers need to think about which type of spatial parameter smoothing relationship is most appropriate for their application. Additionally, where the nature of the problem does not clearly favor one approach over another, tests of alternative models based on different smoothing relations might be carried out. Posterior odds ratios can be constructed to shed light on which smoothing relationship is most consistent with the sample data.

7.3.1 Estimation of the BGWR model

We rely on Gibbs sampling to produce estimates for the BGWR model. To implement the Gibbs sampler we need to derive the conditional posterior distributions for each group of parameters, β_i, σ, δ, and V_i in the model. Let $P(\beta_i|\sigma, \delta, V_i, \gamma)$ denote the conditional density of β_i, where γ represents the values of other β_j for observations $j \neq i$. Using similar notation for the the other conditional densities, the Gibbs sampling process can be viewed as follows:

1. start with arbitrary values for the parameters $\beta_i^0, \sigma^0, \delta^0, V_i^0, \gamma^0$

2. for each observation $i = 1, \ldots, n$,
   (a) sample a value, β_i^1 from $P(\beta_i|\sigma^0, \delta^0, V_i^0, \gamma^0)$
   (b) sample a value, σ^1 from $P(\sigma|\delta^0, V_i^1, \gamma^1)$
   (c) sample a value, V_i^1 from $P(V_i|\beta_i^1, \sigma^0, \delta^0, \gamma^0)$
3. use the sampled values $\beta_i^1, i = 1, \ldots, n$ from each of the n draws above to update γ^0 to γ^1.
4. sample a value, δ^1 from $P(\delta|\sigma^1, V_i^1, \gamma^1)$
5. go to step 1 using $\beta_i^1, \sigma^1, \delta^1, V_i^1, \gamma^1$ in place of the arbitrary starting values.

The sequence of draws outlined above represents a single pass through the sampler, and we make a large number of passes to collect a large sample of parameter values from which we construct our posterior distributions.

Note that this sampler is quite computationally intensive as it requires a loop over all observations for each draw. This is in contrast to the case of the BGWRV model from Section 7.2 where we sampled all draws for each observation, requiring a single pass through the sample. The difference arises from the need to update the parameters in γ from other observations that are used in the distance and contiguity smoothing relationships.

For the case of the concentric city prior we could rely on the GWR estimate for the first observation and then proceed to carry out draws for the remaining observations as we did for the BGWRV model. The draw for observation 2 would rely on the posterior mean computed from the draws for observation 1 to define the parameter smoothing prior. Assuming the observations are ordered by distance from a central observation, this would achieve our goal of stochastically restricting observations from nearby concentric rings to be similar. Observation 2 would be similar to 1, 3 would be similar to 2, and so on.

Another way to implement these models would be to use the GWR estimates as elements in γ. This would allow us to proceed by making multiple draws for each observation, requiring only one pass over the observations as in the case of the BGWRV model. A difference would be that the parameter smoothing relationship doesn't evolve, but is restricted to the estimated GWR values.

We rely on the compact statement of the BGWR model in (7.24) to facilitate presentation of the conditional distributions that we rely on during sampling.

The conditional posterior distribution of β_i given σ, δ, γ and V_i is a multivariate normal shown in (7.31).

$$p(\beta_i | \ldots) \propto N(\hat{\beta}_i, \sigma^2 R) \tag{7.31}$$

Where:

$$\begin{aligned} \hat{\beta}_i &= R(\tilde{X}_i' V_i^{-1} \tilde{y}_i + \tilde{X}_i' \tilde{X}_i J_i \gamma / \delta^2) \\ R &= (\tilde{X}_i' V_i^{-1} \tilde{X}_i + \tilde{X}_i' \tilde{X}_i / \delta^2)^{-1} \end{aligned} \tag{7.32}$$

This result follows from the assumed variance-covariance structures for ε_i, u_i and the Theil-Goldberger (1961) representation shown in (7.26).

The conditional posterior distribution for σ is a $\chi^2(m)$ distribution shown in (7.33), where m denotes the number of observations with non-zero weights.

$$\begin{aligned} p(\sigma|\ldots) &\propto \sigma^{-(m+1)}\exp\{-\frac{1}{2\sigma^2}(\varepsilon_i' V_i^{-1}\varepsilon_i)\} \\ \varepsilon_i &= \tilde{y}_i - \tilde{X}_i\beta_i \end{aligned} \tag{7.33}$$

The conditional posterior distribution for V_i is shown in (7.34), which indicates that we draw an m-vector based on a $\chi^2(r+1)$ distribution. As in the case of the BGWRV model from Section 7.2, the individual elements of the matrix V_i act on the spatial weighting scheme because the estimates involve terms like: $\tilde{X}_i' V_i^{-1}\tilde{X}_i = X'W_i^{1/2}V_i^{-1}W_i^{1/2}X$. For the exponential weighting method, the terms $W_i = \exp(-d_i/\theta)$ will be adjusted by the V_i estimates, which are large for aberrant observations or outliers. In the event of an outlier, observation i will receive less weight when the spatial distance-based weight is divided by a large V_i value as shown below.

$$p\{[(e_i^2/\sigma^2) + r]/V_i \mid \ldots\} \propto \chi^2(r+1) \tag{7.34}$$

Finally, the conditional distribution for δ is a $\chi^2(nk)$ distribution based on (7.35).

$$p(\delta|\ldots) \propto \delta^{-nk}\exp\{-\sum_{i=1}^{n}(\beta_i - J_i\gamma)'(\tilde{X}_i'\tilde{X}_i)^{-1}(\beta_i - J_i\gamma)/2\sigma^2\delta^2\} \tag{7.35}$$

Now consider the modifications needed to the conditional distributions to implement the alternative spatial smoothing relationships set forth in Section 7.3.1. Since we used the same assumptions for the disturbance terms ε_i and u_i, we need only alter the conditional distributions for β_i and δ.

First, consider the case of the monocentric city smoothing relationship. The conditional distribution for β_i is multivariate normal with mean $\hat{\beta}_i$ and variance-covariance $\sigma^2 R$ as shown in (7.36).

$$\begin{aligned} \hat{\beta}_i &= R(\tilde{X}_i' V_i^{-1}\tilde{y}_i + \tilde{X}_i'\tilde{X}_i\beta_{i-1}/\delta^2) \\ R &= (\tilde{X}_i' V_i^{-1}\tilde{X}_i + \tilde{X}_i'\tilde{X}_i/\delta^2)^{-1} \end{aligned} \tag{7.36}$$

The conditional distribution for δ is a $\chi^2(nk)$ based on the expression in (7.37).

$$p(\delta|\ldots) \propto \delta^{-nk}\exp\{-\sum_{i=1}^{n}(\beta_i - \beta_{i-1})'(\tilde{X}_i'\tilde{X}_i)^{-1}(\beta_i - \beta_{i-1})/\sigma^2\delta^2\} \tag{7.37}$$

For the spatial expansion and contiguity smoothing relationships, we have the same conditional expressions for β_i and δ as in the basic model from (7.20) and (7.21). We need only modify the definition of J, to match the different smoothing relations. One additional change is needed for the case of the spatial expansion smoothing relationship, where a conditional distribution for the parameters β_x, β_y must be added to the model. This distribution is a multivariate normal with mean $\hat{\beta} = (\hat{\beta}_x \hat{\beta}_y)'$ and variance-covariance matrix $\sigma^2(J_i'\tilde{X}_i'Q^{-1}\tilde{X}_iJ_i)^{-1}$ as defined in (7.38).

$$\begin{aligned} \hat{\beta} &= (J_i'\tilde{X}_i'Q^{-1}\tilde{X}_iJ_i)^{-1}(J_i'\tilde{X}_i'Q^{-1}\tilde{y}_i) \\ Q &= (V_i + \tilde{X}_i(\tilde{X}_i'\tilde{X}_i)^{-1}\tilde{X}_i'/\delta^2) \end{aligned} \tag{7.38}$$

7.3.2 Informative priors

Implementing the extended BGWR model with diffuse priors on δ may lead to large values that essentially eliminate the parameter smoothing relationship from the model. This would produce estimates similar to those from the BGWRV model introduced in Section 7.2. In cases where the sample data is weak or objective prior information suggests spatial parameter smoothing should follow a particular specification, we can use an informative prior for the parameter δ. A Gamma(a, b) prior distribution with a mean of a/b and variance of a/b^2 seems appropriate. Given this prior, we can eliminate the conditional density for δ and replace it with a random draw from the Gamma(a, b) distribution. We will have more to say about choice of the hyperparameter δ when we illustrate use of the BGWR model.

A similar approach can be taken for the hyperparameter r. Using a Gamma prior distribution with $a = 8, b = 2$ that indicates small values of r around 4, should provide a fair amount of protection against spatial heterogeneity. In the absence of heterogeneity, the resulting V_i estimates will be near unity so the BGWR distance weights will be similar to those from GWR, despite the use of a small value for r.

Additionally, a $\chi^2(c, d)$ natural conjugate prior for the parameter σ can be used in place of the diffuse prior set forth here. This would affect the conditional distribution used during Gibbs sampling in only a minor way.

Some other alternatives offer additional flexibility when implementing the BGWR model. For example, one can restrict specific parameters to exhibit no variation over the spatial sample observations. This might be useful if we wish to restrict the constant term over space. An alternative would be a situation where only the constant term is allowed to vary over space. These alternatives can be implemented by adjusting the prior variances in the parameter smoothing relationship:

$$var - cov(\beta_i) = \sigma^2\delta^2(\tilde{X}_i'\tilde{X}_i)^{-1} \tag{7.39}$$

For example, assuming the constant term is in the first column of the matrix

$\tilde{X}_i$, setting the first row and column elements of $(\tilde{X}_i'\tilde{X}_i)^{-1}$ to zero would restrict the intercept term to remain constant over all observations.

7.3.3 Implementation details

We devised a function **bgwr** to carry out Gibbs sampling estimation of the extended Bayesian GWR model. The documentation for the function is shown below, where a great many input options are available. The function **bgwr** has associated **prt** and **plt** methods to produced printed and graphical presentation of the results.

Options are input to the function using a structure variable named 'prior' to indicate alternative parameter smoothing relationships, a choice of weighting method and settings for the Bayesian prior. Note that only three of the four parameter smoothing relationships discussed in Section 7.3 are implemented. The parameter smoothing relationship based on spatial expansion is not implemented. Another point to note is that you can implement a contiguity smoothing relationship by either specifying a spatial weight matrix or relying on the function **xy2cont** to calculate this matrix using the x-y coordinates.

```
 PURPOSE: compute Bayesian geographically weighted regression
          model: y = Xb(i) + e,      e = N(0,sige*V),
                 b(i) = f[b(j)] + u, u = delta*sige*inv(x'x)
          V = diag(v1,v2,...vn), r/vi = ID chi(r)/r, r = Gamma(m,k)
          delta = gamma(s,t),
          f[b(j)] = b(i-1) for concentric city prior
          f[b(j)] = W(i) b for contiguity prior
          f[b(j)] = [exp(-d/b)/sum(exp(-d/b)] b for distance prior
---------------------------------------------------
 USAGE: results = bgwr(y,x,xcoord,ycoord,ndraw,nomit,prior)
 where: y = dependent variable vector
        x = explanatory variable matrix
        xcoord = x-coordinates in space
        ycoord = y-coordinates in space
        prior = a structure variable with fields:
        prior.rval,   improper r value, default=4
        prior.m,      informative Gamma(m,k) prior on r
        prior.k,      informative Gamma(m,k) prior on r
        prior.dval,   improper delta value (default=diffuse)
prior.dscale, scalar for delta with diffuse prior (default = 1);
        prior.s,      informative Gamma(s,t) prior on delta
        prior.t,      informative Gamma(s,t) prior on delta
        prior.ptype, 'concentric' for concentric city smoothing
                     'distance'   for distance based smoothing (default)
                     'contiguity' for contiguity smoothing
                     'casetti'    for casetti smoothing (not implemented)
        prior.ctr, observation # of central point (for concentric prior)
        prior.W,   (optional) prior weight matrix (for contiguity prior)
        prior.bwidth = scalar bandwidth to use or zero
                       for cross-validation estimation (default)
        prior.dtype  = 'gaussian'    for Gaussian weighting (default)
                     = 'exponential' for exponential weighting
                     = 'tricube'     for tri-cube weighting
        prior.q      = q-nearest neighbors to use for tri-cube weights
```

```
                    (default: CV estimated)
        prior.qmin   = minimum # of neighbors to use in CV search
        prior.qmax   = maximum # of neighbors to use in CV search
                       defaults: qmin = nvar+2, qmax = 5*nvar
        ndraw = # of draws
        nomit = # of initial draws omitted for burn-in
 ---------------------------------------------------
 RETURNS: a results structure
        results.meth   = 'bgwr'
        results.bdraw  = beta draws (ndraw-nomit x nobs x nvar) (a 3-d matrix)
        results.smean  = mean of sige draws (nobs x 1)
        results.vmean  = mean of vi draws (nobs x 1)
results.lpost  = mean of log posterior (nobs x 1)
        results.rdraw  = r-value draws (ndraw-nomit x 1)
        results.ddraw  = delta draws (if diffuse prior used)
        results.r      = value of hyperparameter r (if input)
        results.d      = value of hyperparameter delta (if input)
        results.m      = m prior parameter (if input)
        results.k      = k prior parameter (if input)
        results.s      = s prior parameter (if input)
        results.t      = t prior parameter (if input)
        results.nobs   = nobs
        results.nvar   = nvars
        results.ptype  = input string for parameter smoothing relation
        results.bwidth = bandwidth if gaussian or exponential
        results.q      = q nearest neighbors if tri-cube
        results.dtype  = input string for Gaussian,exponential,tricube weights
        results.iter   = # of simplex iterations for cv
        results.y      = y data vector
        results.x      = x data matrix
        results.xcoord = x-coordinates
        results.ycoord = y-coordinates
        results.ctr    = central point observation # (if concentric prior)
        results.dist   = distance vector (if ptype = 0)
        results.time   = time taken for sampling
---------------------------------------------------
 NOTES: use either improper prior.rval
        or informative Gamma prior.m, prior.k, not both of them
---------------------------------------------------
```

The user also has control over options for assigning a prior to the hyperparameter r to produce robust estimates. Either an improper prior value can be set using our rule-of-thumb value of $r = 4$, or a proper prior based on a Gamma(m,k) distribution. Here, one would try to rely on a prior centered in the range of 4 to 10, because larger values produce estimates that are not robust to heteroscedasticity or outliers. As an example, $m = 8, k = 2$ would implement a prior with the mean of $r = 4$ and the variance of $r = 2$, since the mean of the Gamma distribution is m/k, and the variance is (m/k^2).

The hyperparameter δ can be handled in four ways: 1) we can simply assign an improper prior value using say, 'prior.dval=20' as an input option, 2) we can input nothing about this parameter, producing a default implementation based on a diffuse prior where δ will be estimated, 3) we can assign a Gamma(s,t) prior using the approach set forth above for the hyperparameter r, and 4) we can use a diffuse prior with a scalar on δ input using the 'prior.dscale' option.

Implementation with a large value for the hyperparameter r and diffuse prior for δ will reproduce the non-parametric GWR estimates, if the estimated value for δ is large. This indicates that the prior restriction is inconsistent with the sample data. One would typically use the **bgwr** function in situations where you wish to impose a parameter smoothing relationship on the model, so an informative prior would be in order. The prior setting for δ in this case controls how tightly the parameter smoothing relationship is imposed. Setting δ may be problematical because the scale is unknown and depends on the inherent variability in the GWR estimates. Consider that $\delta = 1$ will assign a prior variance for the parameters in the smoothing relationship based on the variance-covariance matrix of the GWR estimates. This may represent a tight or loose imposition of the parameter smoothing relationship, depending on the amount of variability in the GWR estimates. If the estimates vary widely over space, this choice of δ may not produce estimates that conform very tightly to the parameter smoothing relationship. In general we can say that smaller values of δ reflect a tighter imposition of the spatial parameter smoothing relationship and larger values reflect a looser imposition, but this is unhelpful in particular modeling situations.

A practical approach to setting values for δ would be to generate estimates based on a diffuse prior for δ and examine the posterior mean for this parameter. Setting values of δ smaller than the posterior mean based on a diffuse implementation should produce a prior that imposes the parameter smoothing relationship more tightly on the model estimates. One might use magnitudes for δ based scaling down the diffuse δ estimate by 0.5, 0.25 and 0.1 to examine the impact of the parameter smoothing relationship on the BGWR estimates. To make this approach easy to implement with a single Gibbs sampling run, we provide an input option 'prior.dscale' that allows a scalar to be applied to the mean of the first 'nomit' Gibbs draws for δ. Using this option suggests one might set the number of omitted draws to a larger value, so a more accurate estimate of δ is used to produce the scaled value of δ. Values for 'dscale' less than unity will impose the parameter smoothing relationship more tightly on the model, and values greater than unity impose the relationship in a looser fashion. One might use magnitudes for 'dscale' based scaling down the diffuse δ estimate by 0.5, 0.25 and 0.1 to examine the impact of tighter parameter smoothing relationship on the BGWR estimates.

Using values greater than unity will tend to eliminate the parameter smoothing relationship from the model, and in conjunction with a small value for r result in estimates similar to those from the **bgwrv** function in Section 7.2. If you merely want robust GWR estimates you should use the **bgwrv** function, as it executes more rapidly than **bgwr**. Using a large value for the hyperparameter r and scaling up the δ value will result in GWR estimates, obtained in a very computationally intense fashion.

In some cases where the prior smoothing relationship is extremely consistent with the data, a diffuse prior for δ will result in estimates different from **bgwrv**. In these rare cases, a relatively smaller value will be estimated for the hyperparameter δ, imposing the parameter smoothing relationship in a relatively tight fashion.

Given the flexibility to implement a host of alternative prior settings, the question arises — which settings are most consistent with the sample data? The **bgwr** function returns a vector 'results.lpost', containing the mean of the log posterior for the model based on Gibbs sampled estimates of this quantity. This can be used to compute the posterior probabilities associated with models based on different prior settings, as we will demonstrate in the next section.

7.3.4 Applied Examples

The program in example 3.5 shows how to use the **bgwr** function to produce estimates for the Anselin neigbhorhood crime data set. Estimates based on all three parameter smoothing relationships are produced with $r = 4$, indicating a prior belief in heteroscedasticity or outliers. An informative improper prior for the parameter δ based on $\delta = 1$ is used to illustrate the impact of the parameter smoothing relationships on the GWR estimates. An implementation based on the scaling option for δ set to 0.5 produced estimates of δ equal to 12 for the concentric city prior, 8.4 for the distance prior and 5.2 for the contiguity prior. This suggests our prior setting of $\delta = 1$ represents a very tight imposition of the parameter smoothing relationships. We do this to clearly illustrate the impact of the parameter smoothing relationship in the model.

We specify 550 draws with the first 50 to be discarded for "burn-in". GWR estimates are graphed for comparison with the three sets of Bayesian estimates. Note that we save computing time using the bandwidth parameter calcuated by the GWR model as an input to the **bgwr** function, with the option 'prior.bwidth' that specifies this value. The program took around 275 seconds to generate 550 draws, or around 2 draws per second for this relatively small data set. Use of the tri-cube weighting method produced 550 draws in around 225 seconds. Note that the speed advantage of the tri-cube method is not as great in this small sample as it would be for larger samples. This is because the number of nearest neighbors determined by cross-validation was 11, and exponential weighting produced samples as small as 6 observations, and a maximum of 39. By comparison, the 506 observation Boston data set indicated 24 nearest neighbors versus a minium sample of 40 observations for exponential weighting and a maximum of 389. There is a trade-off between computational gains from reducing the effective sample size of our problem and the ability to detect outliers. This was already noted in connection with the BGWRV model, but in this model the use of parameter smoothing relationships that incorporate information from all observations in the sample may change the nature of this trade-off.

```
% ----- Example 7.5 Using the bgwr() function
% load the Anselin data set
load anselin.dat; y = anselin(:,1); nobs = length(y);
x = [ones(nobs,1) anselin(:,2:3)]; [junk nvar] = size(x);
east = anselin(:,4); north = anselin(:,5);
vnames = strvcat('crime','constant','income','hvalue');
```

```
load wmat.dat; W = wmat; % first-order contiguity matrix
ndraw = 550; nomit = 50;
info.dtype = 'exponential';
res1 = gwr(y,x,east,north,info);
prior.ctr = 21;
% these apply to all of the runs below
prior.dtype = info.dtype;
prior.delta = 1;
prior.rval = 4;
prior.bwidth = res1.bwidth;
prior.ptype = 'concentric';
bres1 = bgwr(y,x,east,north,ndraw,nomit,prior);
% NOTE: we rely on a diffuse prior for delta, the default
prior.ptype = 'distance';
bres2 = bgwr(y,x,east,north,ndraw,nomit,prior);
prior.ptype = 'contiguity';
prior.W = W;
bres3 = bgwr(y,x,east,north,ndraw,nomit,prior);
% compute posterior probabilities for 3 models
psum = bres1.lpost+bres2.lpost+bres3.lpost;
prob1 = bres1.lpost./psum;
prob2 = bres2.lpost./psum;
prob3 = bres3.lpost./psum;
tt=1:nobs; % graph posterior probabilities
plot(tt,prob1,'-k',tt,prob2,'--k',tt,prob3,'-.k');
%title('log posterior for 3 models');
legend('concentric','distance','contiguity');
xlabel('observations');
ylabel('probabilities');
pause;
% compute posterior means for the parametes
tmp = mean(bres1.bdraw); bout = squeeze(tmp);
[n k] = size(bout);      beta1 = zeros(n,k);
for i=1:k;  beta1(:,i) = bout(:,i); end;
tmp = mean(bres2.bdraw); bout = squeeze(tmp);
[n k] = size(bout);      beta2 = zeros(n,k);
for i=1:k;  beta2(:,i) = bout(:,i); end;
tmp = mean(bres3.bdraw); bout = squeeze(tmp);
[n k] = size(bout);      beta3 = zeros(n,k);
for i=1:k;  beta3(:,i) = bout(:,i); end;
% plot estimates for comparison
subplot(3,1,1), plot(tt,res1.beta(:,1),'-k',tt,beta1(:,1),'--k', ...
tt,beta2(:,1),'-.k',tt,beta3(:,1),'ok');
title('exponential weighting');  xlabel('constant');
subplot(3,1,2), plot(tt,res1.beta(:,2),'-k',tt,beta1(:,2),'--k', ...
tt,beta2(:,2),'-.k',tt,beta3(:,2),'ok');  xlabel('income');
subplot(3,1,3), plot(tt,res1.beta(:,3),'-',tt,beta1(:,3),'--k', ...
tt,beta2(:,3),'-.k',tt,beta3(:,3),'ok');  xlabel('house value');
legend('gwr','concentric','distance','contiguity');
pause;
subplot(1,1,1),
plot(tt,bres1.vmean,'-k',tt,bres2.vmean,'--k',tt,bres3.vmean,'-.k');
ylabel('vi estimates');
xlabel('observations');
legend('concentric,','distance','contiguity');
pause;
subplot(1,1,1),
```

```
plot(tt,res1.sige,'ok',tt,bres1.smean,tt,bres2.smean,'--k',tt,bres3.smean,'-.k');
ylabel('sige estimates');
xlabel('observations');
legend('gwr','concentric','distance','contiguity');
```

The posterior probabilities for the three models can be calculated as the log posterior for every observation divided by the sum of the log posterior for all three models at each observation. Expression (7.40) shows the log posterior for a single observation of our BGWR model. Posterior probabilities based on these quantities provide an indication of which parameter smoothing relationship fits the sample data best as we range over observations.

$$\mathrm{Log}P_i = \sum_{j=1}^{n} W_{ij}\{\mathrm{log}\phi([y_j - X_i\beta_i]/\sigma_i v_{ij}) - \mathrm{log}\sigma_i v_{ij}\} \tag{7.40}$$

A Bayesian solution to the model specification is to average over alternative model specifications using these posterior probabilities as weights (see Leamer, 1984). Model estimates for this Bayesian average model would be created by averaging the Gibbs draws for all parameters, β_i, σ_i, v_i using the posterior probabilities as weights. Estimates of dispersion for the β_i parameters would be based on the standard deviation for the posterior probability weighted Gibbs draws for these parameters.

The posterior probabilities are shown in Figure 7.14, where we see evidence that all three models find some support in the sample data over various ranges of observations. We might expect the concentric city prior to exhibits relatively higher posterior probability for observations near outliers, because it relies on a less restrictive parameter smoothing specification. This allows the estimates to adjust more rapidly in the presence of outliers. This prior relies on the single previous observation for smoothing whereas the distance and contiguity priors tie the estimates for every observation to those from all other observations.

Keep in mind that the posterior probabilities and the log posterior reflect a measure of fit to the sample data, as is clear from (7.40). In applications where robust estimates are desired, it is not clear that choice of models should be made using these probabilities. Robust estimates require a trade-off between fit and insensitivity to aberrant observations.

Parameter estimates for the three Bayesian models and the GWR are shown in Figure 7.15. The impact of the very tight imposition for the parameter smoothing relationships is evident as these estimates are much smoother than the GWR estimates. All three sets of Bayesian estimates depart from the GWR estimates for observations near the outliers. The reasons for this were provided in our discussion of the role of the v_i parameters in the BGWRV model. In addition to the robustness produced by the v_i parameters, these models also draw on information provided by estimates from other observations to create even more smoothing of the estimates. As we would expect, estimates associated with the distance and contiguity prior are smoother than those based on the concentric city prior which relies on only a single other parameter for smoothing.

The distance and contiguity priors also produce similar estimates as we might expect.

If we were carrying out a formal analysis, we would wish to explore the sensitivity of these estimates to our improper prior based on $\delta = 1$. This might be done by comparing estimates based on a diffuse prior and the scaling for $\delta =$ 0.5 which produced estimates that were more similar across the three Bayesian models. They were still different from the GWR estimates near the outlier observations. As we loosen the prior on parameter smoothing the estimates will of course become more similar, but scaling δ by 0.5 represents a fairly tight prior. This suggests that all three parameter smoothing relationships applied as informative priors would lead to similar inferences. Note also, these inferences would be different from those produced by the GWR model.

Another comparison that might be explored is the sensitivity of the estimates to the distance weighting method used. An advantage of the BGWR model over the GWR is that introduction of the parameter smoothing relationship should reduce sensitivity of the estimates to changes in weighting methods. Intuitively, since we rely on other parameter estimates as well as the sample data, changes in the weighting method should not have as great an impact on the estimates. This allows us to rely on the tri-cube weighting method which is faster than

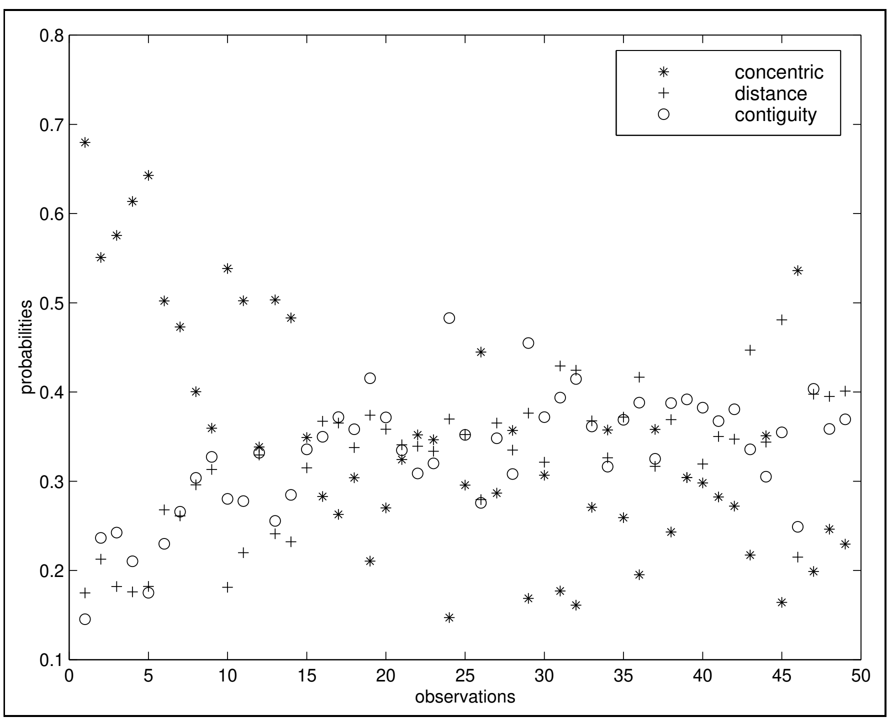

Figure 7.14: Posterior probabilities for $\delta = 1$, three models

Gaussian or exponential weighting since it works with smaller matrices.

The goal of exploring alternative estimates based on different weighting methods and parameter smoothing priors applied with varying tightness is to map out a range of settings for which the estimates are insensitive to these specification settings. Ideally, we would like to conclude that inferences from our analysis are robust with respect to a particular parameter smoothing specification or perhaps all three parameter smoothing relationships, as well as the weighting method used. A less than ideal outcome would be the case where we need to qualify our inferences by restricting them to a range prior settings. In the case of the ideal outcome, we would solve the problems that confronted us in Chapter 6, where we found that GWR estimates varied a great deal as we changed the weighting method or data sample. We will pursue a more systematic comparison of this type in the next section, where we apply the BGWR model to the a sample of data from Cuyahoga county, Ohio.

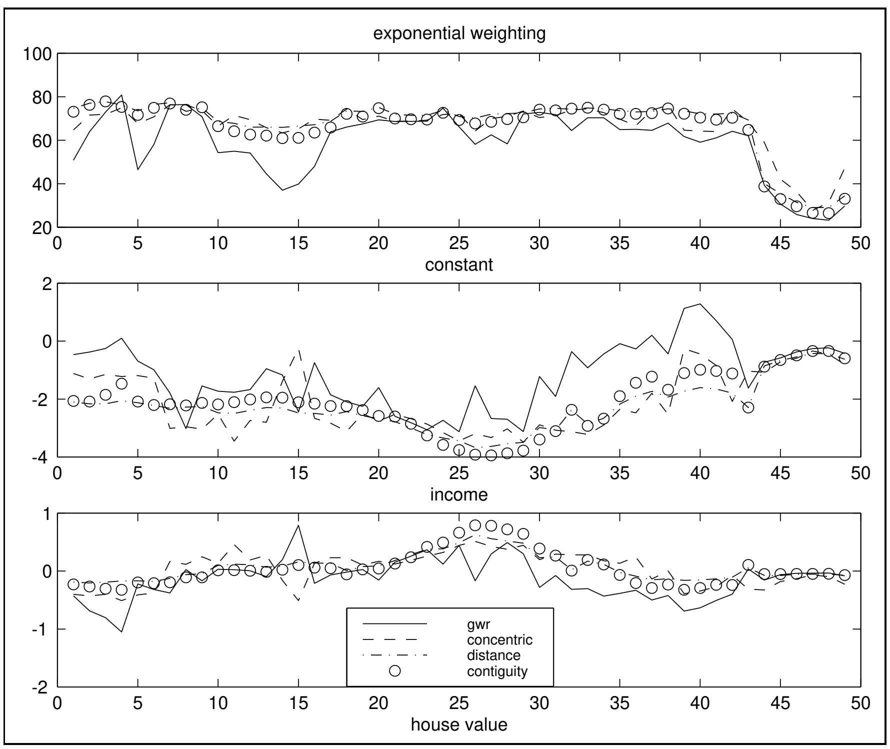

Figure 7.15: GWR and β_i estimates for the Bayesian models

Figure 7.16 shows the v_i estimates from the three Bayesian models as well as the σ_i estimates. An interesting point is that use of the parameter smoothing relationships represent an alternative approach to overcoming the impact of outliers on the estimates. Because we stochastically restrict the estimates, the impact of outliers will be diminished, so we might expect to see smaller v_i estimates. On the other hand, in cases where we tightly impose a parameter smoothing prior that is inconsistent with the sample data, we might generate new outliers. This would occur if the parameter restriction inhibits the model from producing a locally linear fit that adjusts to rapid changes in the sample data observations. For this data sample we see smaller v_i magnitudes than in the BGWRV model, where no parameter smoothing relationship was involved.

The v_{ij} estimates are averaged over all Gibbs draws for each observation, and then averaged again to produce an n by 1 vector that can be used as a diagnostic measure to detect aberrant observations. By virtue of the averaging, we interpret large v_i values as reflecting observations that consistently produced large residuals during estimation of each β_i parameter. From the figure, we see evidence that observations #2, #4 and #34 represent outliers as in the case of the BGWRV model.

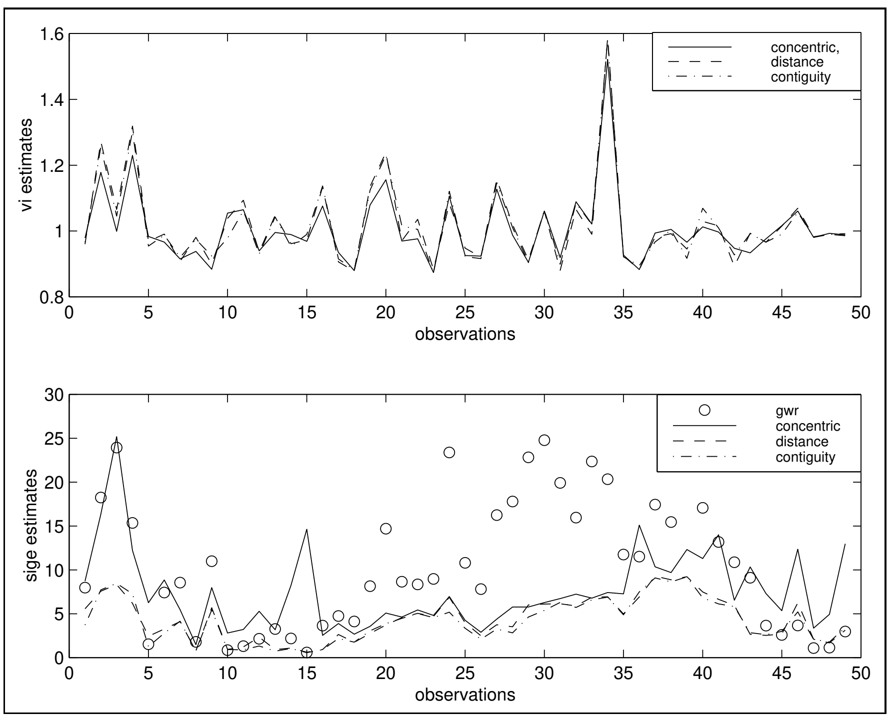

Figure 7.16: v_i estimates for the three models

Ultimately, the role of the parameters V_i in the model and the prior assigned to these parameters reflects our prior knowledge that distance alone may not be reliable as the basis for spatial relationships between variables. If distance-based weights are used in the presence of aberrant observations, inferences will be contaminated for whole neighborhoods and regions in our analysis. Incorporating this prior knowledge turns out to be relatively simple in the Bayesian framework, and it appears to effectively robustify estimates against the presence of spatial outliers.

The Bayesian estimates of σ_i are uniformly smaller than those from the GWR model, which we would expect in the presence of the additional v_i parameters. For strict comparability one would want to calculate disturbance variance estimates based on $\sigma_i^2 v_i$, since these are the comparable magnitudes

7.4 An applied exercise

To illustrate alternative parameter smoothing relationships we use a data set consisting of employment, payroll earnings and the number of establishments in all fifty zip (postal) codes from Cuyahoga county in Ohio during the first quarter of 1989. The data set was created by aggregating establishment level data used by the State of Ohio for unemployment insurance purposes. It represents employment for workers covered by the state unemployment insurance program. The regression model used was:

$$\ln(E_i/F_i) = \beta_{0i} + \beta_{1i}\ln(P_i/E_i) + \beta_{2i}\ln(F_i) + \varepsilon_i \tag{7.41}$$

Where E_i is employment in zip code i, P_i represents payroll earnings and F_i denotes the number of establishments. The relationship indicates that employment per firm is a function of earnings per worker and the number of firms in the zip code area. For presentation purposes we sorted the sample of 50 observations by the dependent variable from low to high, so observation #1 represents the zip code district with the smallest level of employment per firm.

Three alternative parameter smoothing relationships were used, the monocentric city prior centered on the central business district where employment was highest, the distance decay prior and the contiguity prior. We would expect the monocentric city prior to work well in this application. An initial set of estimates based on a diffuse prior for δ are discussed below and would typically be generated to calibrate the tightness of alternative settings for the prior on the parameter smoothing relations.

A Gaussian distance weighting method was used, but estimates based on the exponential weighting method were quite similar. All three BGWR models were based on a hyperparameter $r = 4$ reflecting a heteroscedastic prior.

A graph of the three sets of estimates is shown in Figure 7.11, where it should be kept in mind that the observations are sorted by employment per firm from low to high. This helps when interpreting variation in the estimates over the observations.

The first thing to note is the relatively unstable GWR estimates for the constant term and earnings per worker when compared to the BGWR estimates. Evidence of parameter smoothing is clearly present. Bayesian methods attempt to introduce a small amount of bias in an effort to produce a substantial increase in precision. This seems a reasonable trade-off if it allows clearer inferences. The diffuse prior for the smoothing relationships produced estimates for δ^2 equal to

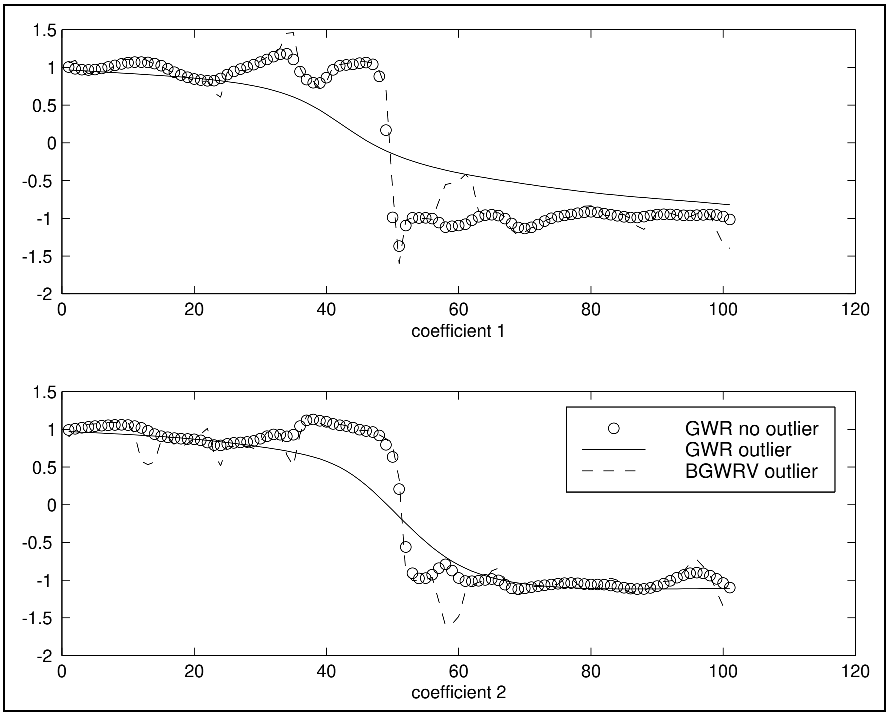

Figure 7.17: Ohio GWR versus BGWR estimates

138 for the monocentric city prior, 142 and 113 for the distance and contiguity priors. These large values indicate that the sample data are inconsistent with these parameter smoothing relationships, so their use would likely introduce some bias in the estimates. From the plot of the coefficients it is clear that no systematic bias is introduced, rather we see evidence of smoothing that impacts only volatile GWR estimates that take rapid jumps from one observation to the next.

Note that the GWR and BGWR estimates for the coefficients on the number of firms are remarkably similar. There are two factors at work to create a divergence between the GWR and BGWR estimates. One is the introduction of v_i parameters to capture non-constant variance over space and the other is the parameter smoothing relationship. The GWR coefficient on the firm variable is apparently insensitive to any non-constant variance in this data set. In addition, the BGWR estimates are not affected by the parameter smoothing relationships we introduced. An explanation for this is that a least-squares estimate for this coefficient produced a $t-$statistic of 1.5, significant at only the 15% level. Since our parameter smoothing prior relies on the variance-covariance matrix from least-squares (adjusted by the distance weights), it is likely that the parameter smoothing relationships are imposed very loosely for this coefficient. Of course, this will result in estimates equivalent to the GWR estimates.

A final point is that all three parameter smoothing relations produced relatively similar estimates. The monocentric city prior was most divergent with the distance and contiguity priors very similar. We would expect this since the latter priors rely on the entire sample of estimates whereas the contiguity prior relies only on the estimate from a neighboring observation.

The times required for 550 draws with these models were: 320 seconds for the monocentric city prior, 324 seconds for the distance-based prior, and 331 seconds for the contiguity prior.

Turning attention to the question of which parameter smoothing relation is most consistent with the sample data, a graph of the posterior probabilities for each of the three models is shown in the top panel of Figure 7.12. It seems quite clear that the monocentric smoothing relation is most consistent with the data as it receives slightly higher posterior probability values for all observations. There is however no dominating evidence in favor of a single model, since the other two models receive substantial posterior probability weight over all observations, summing to over 60 percent.

For purposes of inference, a single set of parameters can be generated using these posterior probabilities to weight the three sets of parameters. This represents a Bayesian solution to the model specification issue (see Leamer, 1984). In this application, the parameters averaged using the posterior probabilities would look very similar to those in Figure 7.11, since the weights are roughly equal and the coefficients are very similar.

Figure ?? also shows a graph of the estimated v_i parameters from all three versions of the BGWR model. These are nearly identical and point to observations at the beginning and end of the sample as regions of non-constant variance as well as observations around 17, 20, 35, 38 and 44 as perhaps outliers. Because the observations are sorted from small to large, the large v_i estimates at the beginning and end of the sample indicate our model is not working well for these extremes in firm size. It is interesting to note that outlying GWR estimates by comparison with the smoothed BGWR estimates correlate highly with observations where the v_i estimates are large.

A final question is — how sensitive are these inferences regarding the three models to the diffuse prior used for the parameter δ? If we really want to test alternative smoothing priors in an attempt to find a single best model, the appropriate approach is to impose the priors in a relatively tight fashion. The posterior probabilities will tend to concentrate on the model that is most consistent with the data in the face of a very strict implementation of the smoothing relationship. To illustrate this, we constructed another set of estimates and posterior probabilities based on scaling δ to 0.1 times the estimate of δ from the diffuse prior. This should reflect a fairly tight imposition of the prior for the parameter smoothing relationships.

The posterior probabilities from these three models are shown in Figure 7.13 and the estimates are shown in Figure ??. We see that both the estimates and posterior probabilities are relatively unchanged from the diffuse prior implementation. This suggests that even with this tighter imposition of the prior,

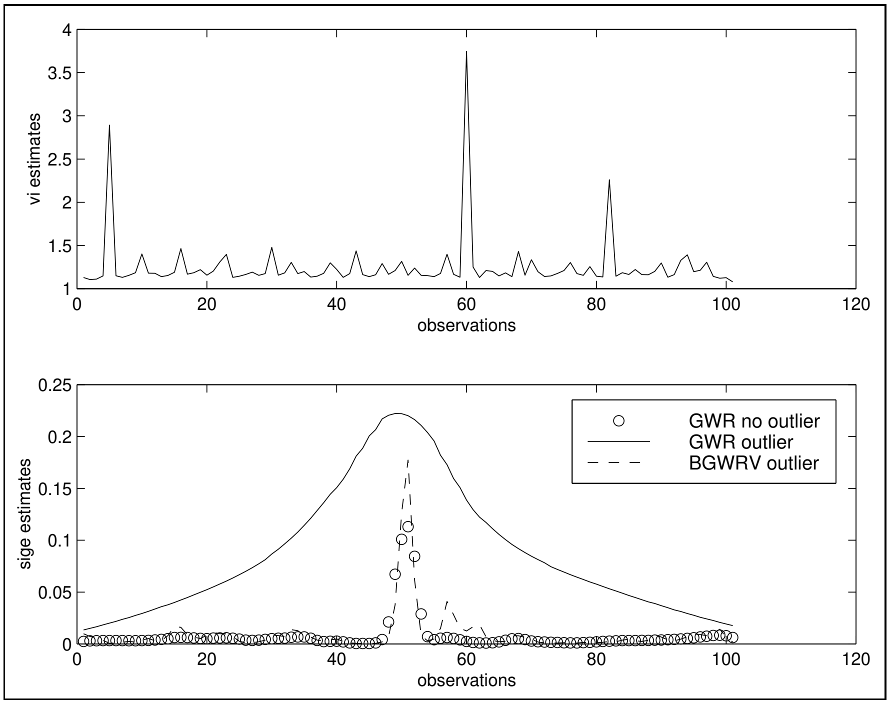

Figure 7.18: Posterior probabilities and v_i estimates

all three parameter smoothing relationships are relatively compatible with the sample data. No smoothing relationship obtains a distinctive advantage over the others.

We need to keep the trade-off between bias and efficiency in mind when implementing tight versions of the parameter smoothing relationships. For this application, the fact that both diffuse and tight implementation of the parameter smoothing relationships produced similar estimates indicates our inferences would be robust with respect to relatively large changes in the smoothing priors. The BGWR models based on these smoothing relationships also shown the potential value of imposing subjective prior information and using robust estimation methods in achieving better inferences that abstract from the influence of aberrant observations.

7.5 Chapter Summary

This chapter introduced Bayesian approaches to implementing Casetti's spatial expansion model and the locally linear regression models based on distance-weighted sub-samples of the observations. These Bayesian variants of the models introduced in Chapter 6 have some advantages.

Figure 7.19: Posterior probabilities for a tight prior

For the case of spatial expansion where heteroscedastic disturbances are present by construction, we showed how the Bayesian heteroscedastic linear model introduced in Section 4.1.1 could be used to estimate this model.

The Bayesian estimates provide a direct inference regarding the appropriateness of the spatial expansion restrictions at each point in space as does the DARP method. An advantage over the DARP model is that no restrictive specification for the non-constant variance is needed. Recall that the DARP method relies on a variance specification based on either distance from a central observation or the latitude-longitude coordinates of each observation in the sample. This may or may not represent an appropriate specification for the non-constant variance. It is used simply because few other alternatives are available in a maximum likelihood setting. As we have seen in numerous examples, the presence of outliers creates variances that take on a spike character for the aberrant observations. This would represent a situation where the smooth variance

Figure 7.20: Estimates based on a tight imposition of the prior

specification implied by the DARP model is inappropriate.

Since DARP estimation requires solving a non-linear optimization problem for the estimates at each point in the sample data, the Gibbs sampling approach used to produce Bayesian estimates is relatively competitive in terms of computational speed.

The non-parametric locally linear models known as geographically weighted regressions introduced in Chapter 6 also exhibited problems. One problem is that inferences about the parameters as they vary over space cannot be drawn using traditional methods from least-squares. This is because re-use of the sample observations creates a lack of independence between the estimates.

In contrast, Bayesian estimates based on Gibbs sampling do not require sample independence, and by virtue of the thereom from Gelfand and Smith (1990), they provide valid inferences. Another problem that arose with the GWR estimates was a sensitivity to aberrant observations which tend to contaminate whole subsequences of the estimates covering entire regions in space. The BGWRV model provides robust estimates in these cases by automatically detecting and downweighting these observations. This mitigates the influence of outliers on the estimates, so inferences regarding changes in the model relationship over space can be clearly attributed to spatial variation. A further advantage of the BGWRV approach is that a diagnostic plot can be used to identity observations associated with regions of non-constant variance or spatial outliers. An applied illustration based on a generated data set confirmed the advantages of the BGWRV method.

A final BGWR model was introduced that subsumes the spatial expansion, DARP and GWR models as special cases. In addition, the BGWR model provides an ability to incorporate an explicit relationship to describe parameter variation over space. Informative prior parameter smoothing relationships that relied on: distance decay relationships, contiguity relationships, monocentric distance from a central point, or the lattitude-longitude locations proposed by Casetti (1972) were set forth. An applied exercise demonstrated the value of the smoothing relations introduced in the BGWR method using a sample of 50 employment observations for Cuyahoga county in Ohio.

Chapter 8

Future directions

This chapter discusses some places where Bayesian methods can make future contributions to spatial econometric modeling. The discussion is divided into sections that discuss alternative aspects of model specification that confront practitioners. The issue of variable selection has filled the econometrics literature for years. It is often the case that a large number of possible explanatory variables exist, but researchers seek to avoid overfitting by including too many of these variables. Bayesian methods suggested by Leamer (1978,1983) provide a solution to this problem that relies on combining models using posterior probabilities associated with each model. Unfortunately, diffuse priors cannot be used and the influence of the prior distribution has a fairly important influence on the resulting posterior probabilities used to produce the average model.

Section ?? takes up the issue of what explanatory variables should be used in spatial autoregressive and locally linear non-parametric models. A Bayesian literature labeled "Bayesian model averaging" has developed recently (Raftery, Madigan and Hoeting, 1998 and Fernandez, Ley and Steel, 1998) that has much to offer those attempting to construct spatial econometric models. Two innovations in this literature are: 1) a set of priors that exert relatively weak influence on the posterior probabilities and, 2) a Markov chain sampling approach to exploring the very large space of models based on all possible combinations of the candidate explanatory variables.

Specification of the appropriate spatial structure in spatial autoregressive models represents another place where these methods could be applied. These issues are discussed in Section ??.

Perhaps the most important contribution that Bayesian methods could make is in the area of specifying the spatial weight matrix used for spatial autoregressive models. Traditional approaches have simply assumed the spatial weight matrix is known, making all model estimates and inferences conditional on the weight matrix used to construct the model. There are many applications where a weight matrix based on geographic contiguity seems inappropriate because the administrative units available don't match the scale of the spatial process being modeled. Estimating the spatial weight matrix as part of the model would represent a major step forward in spatial autoregressive modeling. Section ?? discusses possibilities for doing this that draw on the literature from Bayesian model averaging introduced in Section ??.

A final section discusses Bayesian approaches that would allow us to incorporate the bandwidth parameter as part of the estimation process in GWR models. This is in contrast to the approach taken currently, where this parameter is determined in a separate optimization process that takes place prior to estimation of the GWR parameters.

All of these ideas for future contributions rely on some basic Bayesian ideas. One is the Bayesian view that estimates should be unconditional, integrating over all parameters in the model to produce posterior estimates that incorporate uncertainty about all aspects of the problem being modeled. These include uncertainty introduced due to model specification of the explanatory variables, model specification choices regarding the type of spatial structure used, the weight matrix used, the bandwidth employed in GWR models and so on.

A second idea is that put forth by Leamer (1978,1983). If $M = \{M_1, \ldots, M_K\}$ denotes the set of all possible models being considered and if Δ denotes the quantity of interest, perhaps a future observation or the utility derived from a particular course of action, then the posterior distribution of Δ given the data D is:

$$\mathrm{pr}(\Delta|D) = \sum_{k=1}^{K} \mathrm{pr}(\Delta|M_k, D)\mathrm{pr}(M_k|D) \tag{8.1}$$

This is an average of the posterior distributions under each model weighted by the corresponding posterior model probabilities. It has been labeled Bayesian Model Averaging or (BMA) by (Raftery, Madigan and Hoeting, 1998 and Fernandez, Ley and Steel, 1998). In (??), the posterior probability of model M_k is given by:

$$\mathrm{pr}(M_k|D) = \mathrm{pr}(D|M_k)\mathrm{pr}(M_k) / \sum_{l=1}^{K} \mathrm{pr}(D|M_l)\mathrm{pr}(M_l) \tag{8.2}$$

Where $\mathrm{pr}(D|M_k)$ represents the marginal likelihood of model M_k which depends on the vector of parameters for model k, the prior density for these parameters and the model likelihood. All of the probabilities are conditional on the set M of models being considered and the size of this set can be quite large in applied settings. Until recently this has limited the applied usefulness of Leamer's suggestion.

For example, consider a regression model where 15 candidate explanatory variables exist. This leads to a set of $2^{15} = 32,768$ different regression models, if we consider all possible combinations of explanatory variables.

Amazingly enough, Raftery, Madigan and Hoeting (1997) devise a way to implement this approach using a Metropolis sampling scheme to systematically move through model space, producing an MCMC sample from the space of all possible models.

A key insight they exploit is that the Bayes factor or posterior odds ratios from two alternative model specifications can be used to construct a Markov chain that traverses model neighborhoods. They argue that this systematic movement of the Markov Chain via a Hastings step generates a stochastic process that moves through model space. Under unrestrictive regularity conditions, the average of the parameters from the sampled models converges to the model uncertainty solution proposed by Leamer (1978, 1983).

Given that one moves through model space systematically sampling from the space of all possible models, we do not need to actually visit and evaluate every one of the marginal likelihoods for thousands of models. We simply run the Gibbs sampler for a long enough time to gather a representative sample from the large model space on which we construct posterior probabilities and an average model.

8.1 Specification of explanatory variables

The current literature on BMA has developed the theory and implementation algorithms for the case of linear regression models. These could of course be immediately applied to the case of GWR models where linear regression methods are used to estimate a sequence of locally linear regressions based on distance weights.

For each regression, we simply implement the methods from Raftery, Madigan and Hoeting (1997) or Fernandez, Ley and Steel (1998). The latter authors indicate that a model involving 47 observations and 15 possible explanatory variables requires a mere 80 seconds on a 120MHz 604 PowerPC Macintosh personal computer to produce 100,000 draws after a burn-in of 25,000 draws. A total of 3378 distinct models were visited and the best 10% of these models accounted for 73.5% of the posterior model probability. Computational gains are achieved by avoiding calculation of the marginal likelihood except in cases where a new model is encountered during sampling.

Extapolating this to the case of GWR, we would need to spend 80 seconds sampling each observation. My experience with these methods indicates that a mere 5,000 to 10,000 draws would suffice in place of the 125,000 draws used in Fernandez, Ley and Steel (1998), and they confirm this. Convergence appears rapid with the number of new models visited tailing off rapidly after the initial 5,000 draws. This reduces the time required to around 8 seconds per observation, making this approach feasible for GWR models. Note also that in many cases less than 15 possible explanatory variables would be involved, decreasing the size of the model space and increasing the speed.

To elaborate on this method, consider the Markov chain $M_t, t = 1, 2, \ldots$ with state space M and equilibrium distribution $\mathrm{pr}(M_i|D)$. If we simulate this Markov chain for $t = 1, 2, \ldots, N$, then under unrestrictive regularity conditions, for any function $g(M_i)$ defined on M, the average:

$$\hat{G} = (1/N)\sum_{t=1}^{N} g(M(t)) \tag{8.3}$$

converges to $E(g(M))$ as N approaches ∞. To compute the solution in (??) using this Markov chain, we define $g(M)$ in (??) as $\mathrm{pr}(\Delta|M, D)$.

The Markov chain is constructed using a neighborhood which consists of the model itself, M_k and the set of models with either one variable more or one variable fewer than the model M_k which we designate as M_j. The Metropolis step then involves considering the current model state M_k versus a draw for the neighboring model M_j. The new model M_j is accepted with probability:

$$\min\left(1, \frac{\mathrm{pr}(M_j|D)}{\mathrm{pr}(M_k|D)}\right) \tag{8.4}$$

For the case where we do not accept the new model M_j, we stay with the current model state M_k.

As a concrete illustration of these ideas, example 8.1 implements BMA for a linear regression model based on the methods of Raftery, Madigan and Hoeting (1997). A function **bma_g** from the *Econometrics Toolbox* is used to produce these BMA estimates. The true model used to generate the dependent variable y contains only two explanatory variables, X_1, X_2, whereas four variables are entered as inputs in the matrix 'x' supplied as an argument to the function **bma_g**. This example contains only $4^2 = 16$ distinct models.

```
% ----- Example 8.1 Bayesian model averaging
nobs = 100;   % generate a model based on x1,x2
vin = [1 2];  % variables in the model
vout = [3 4]; % variables not in the model
nv1 = length(vin); nv2 = length(vout);
nvar = nv1+nv2; % total # of variables
x = randn(nobs,nvar);
xtrue = x(:,vin); % true model based on variables in the model
[junk nvart] = size(xtrue);
btrue = ones(nv1,1);
y = 10*ones(nobs,1) + xtrue*btrue + 0.5*randn(nobs,1);
ndraw = 5000;
result = bma_g(ndraw,y,x);
prt(result);
```

The results are printed below showing the posterior probabilities greater than 1% for each of the distinct models found during sampling. Visits to each model are also recorded, indicating that 2,262 visits were made to the three models that exhibited posterior probabilities greater than 1%. The remainder of the models visited during the 5,000 draws had posterior probabilities less than 1%. A total of 12 of the 16 distinct models were visited during the 5,000 draws. (This information is not printed, but recorded in the results structure returned by **bma_g**.)

We see that the procedure correctly identifies the true model that included variables 1 and 2, assigning this model a posterior probability of 91.047%. The estimates and $t-$statistics printed represent an average constructed using all posterior probabilities as weights, including those not reported because they were less than 1%. We see that the averaged model estimates are very close to the values used to generate the dependent variable y.

```
Bayesian Model Averaging Estimates
R-squared      =    0.9092
sigma^2        =    0.2171
Nobs, Nvars    =    100,     4
ndraws         =   5000
nu,lam,phi     =  4.000, 0.250, 3.000
# of models    =     12
time(seconds)  =   25.6046
*************************************************************
Model averaging information
Model       v1     v2     v3     v4      Prob   Visit
model 1      1      1      1      0     3.085    1054
model 2      1      1      0      1     5.664    1099
model 3      1      1      0      0    91.047     109

*************************************************************
      Posterior Estimates
Variable      Coefficient        t-statistic    t-probability
const           10.069695         214.933067         0.000000
v1               0.989554          23.938665         0.000000
v2               0.952322          21.523651         0.000000
v3              -0.000240          -0.004958         0.996054
v4              -0.002806          -0.065627         0.947806
```

The time required to implement this method based on 5,000 draws was 25.6 seconds, which is much slower than the approach implemented by Fernandez, Ley and Steel (1998).

In addition to application of this method to the GWR models, it should also be useful in determining the appropriate set of explanatory variables in spatial autoregressive models. All of the information needed to compute the marginal likelihood for these models with alternative sets of explanatory variables is available. The most computationally intensive part of evaluating the marginal likelihood is computing the likelihood function. As we saw in Chapter 3, we can rapidly evaluate the likelihood function even for large data samples using the sparse matrix algorithms available in MATLAB.

A point to note concerning implementation of BMA is that improper priors cannot be used. Choosing a "sensible" prior for the parameters in the model is an issue that remains to be resolved. The expression in (??) depends on both the prior odds for model k versus j and the Bayes factors for model k versus j. It is well known that Bayes factors are sensitive to the choice of prior distribution for the parameters we assign to each model. Fernandez, Ley and Steel (1998) provide a good solution for the case of linear regression models, but an extension to the case of spatial autoregressive models needs to be developed. This should not be difficult as a fairly flat but informative prior for the spatial parameters ρ and λ would work.

8.2 Specification of spatial structure

Here, the specification issue relates to which of the alternative models from the taxonomy of spatial autoregressive models is most appropriate. The issue of specifying a prior over these models is more difficult because the models might differ in both the explanatory variables used in the matrix X as well as the spatial structure which may include both ρ and λ in some models and only one of these parameters in another model.

Reliance on the Markov chain that visits neighbors, where neighboring models are defined by models including one more or one fewer explanatory variable is no longer possible. We need to include visits to SAR, SEM and SAC models, as well as models based on different sets of explanatory variables.

A theme in the BMA literature is the desire to specify informative priors (needed for the method) that are not too informative. This allows the sample data evidence to play a dominant role in determining the posterior probabilities for the models.

To see the difficulties encountered in the case of spatial autoregressive models, consider a case where one assigned truly informative subjective prior probabilities to each of the three models, SAR, SEM and SAC. (These must be assigned without resort to the sample data.) Given these subjective prior probabilities, a set of Bayes factors could be computed for the three models, but the model averaging solution would take place over models of different dimension. The SAR and SEM models include only ρ or λ, whereas the SAC model includes both of these parameters. It is unclear whether averaging over models of different dimensionality would produce desirable results.

Note however that this is precisely what is done for the case of regression models, where models with differing sets of explanatory variables are averaged to produce results. This appeared to work well in example 8.1. However, the addition or exclusion of a linear explanatory variable in regression models is quite different from inclusion or exclusion of the non-linear terms that differentiate SAR, SEM and SAC models.

8.3 Specification of the spatial weight matrix

Spatial autoregressive model estimates are conditional on the choice of weight matrix used. Traditional practice is to assume a known contiguity matrix and proceed with estimation.

In the context of MCMC sampling for a given model, say SAR, it should be theoretically possible to add a conditional distribution for elements of the weight matrix, treating these as additional parameters to be estimated. This would also require that we consider all other parameters in the model conditional on elements of the weight matrix.

First, there is the daunting task of constructing conditional distributions for the weight matrix W as a function of the other parameters in the model. In addition, we need conditional distributions for the parameters ρ, λ, σ and β with respect to the individual elements of the weight matrix W.

To clarify this idea, recall that we derived the conditional distribution for σ by subsuming the known weight matrix as part of the constant of proportionality, leading to a simple chi-squared conditional distribution for this parameter. In the case where elements of the matrix W are considered parameters of the model, this would no longer be valid. We would arrive at a conditional distribution for σ that depends on all n^2 elements of the weight matrix W. A similar situation would arise for the parameters β, ρ and λ. These conditional distributions would also depend on all elements of the spatial weight matrix.

Despite the apparent complexity, there are approaches that can be taken. First, we can avoid the difficulty associated with considering all n^2 elements of the spatial weight matrix by relying on a more parsimonious representation. Consider that given two n by 1 vectors containing latitude-longitude coordinates, we can construct a contiguity matrix using the delaunay algorithm implemented in the function **xy2cont**.

Suppose we assign a normal prior distribution centered on the latitude-longitude coordinates with prior variances based on the sample variances of these two vectors. This would allow us to generate a distribution of spatial contiguity matrices. These would be constructed by adding a normally distributed perturbation to the latitude or longitude vector with standard deviations for the perturbation based on the empirically determined variance. The distribution of weight matrices would be centered on the known contiguity matrix, but allow for alternative spatial weights that are normally distributed around the given weight matrix W.

A finite representative grid of weight matrices $W_i^x, i = 1, \ldots, K$ and $W_j^y, j = 1, \ldots, K$ can then be constructed from the normal distribution of weight matrices, where W_i^x denotes one of the K matrices constructed using the distribution that arises from a normal perturbation applied to the latitude coordinate vector. Similarly, W_j^y reflects a weight matrix from the distribution based on perturbation of the longitude coordinate.

At this point, a conventional Bayesian solution would involve computing posterior probability weights for the set of all 2^{2K} possible combinations of these weight matrices. Estimates, including an estimate of the spatial weight matrix would be based on the average over all models each involving a different weight matrix. The weights used in constructing the average are the posterior probabilities associated with each model. The problem with this approach is that even a moderately sized grid from the normal distribution of matrices based on say, $K = 10$, would require that we evaluate the marginal likelihood for $2^{20} = 1,048,576$ models!

The BMA approach might provide a solution to this problem. This would involve setting up a Markov chain to visit neighboring models, where we define a neighboring model as follows. Given that we are currently in a model state defined by W_k^x, W_k^y, draw a new model based on either W_j^x or W_j^y, where $j = k+1$ or $j = k-1$. We choose whether to construct the new model based on W^x or W^y using a coin flip. If we encounter a situation where W_j^x equals K, we rely on $j = K-1$ and similarly for the case of W_K^y. A similar procedure is used in cases where $j = K = 1$, we rely on $j = K + 1$. This approach ensures that in the limit we would visit all possible combinations of the 2^{2K} models based on the constructed grid of weight matrices.

The Metropolis step then involves considering the current model state M_k based on weight matrix k versus a draw for the neighboring model M_j that is based on weight matrix W_j^x or W_j^y. The new model M_j is accepted with probability:

$$\min\left(1, \frac{\mathrm{pr}(M_j|D}{\mathrm{pr}(M_k|D)}\right) \tag{8.5}$$

For the case where we do not accept the new model M_j, we stay with the current model state M_k. The intuition behind the Metropolis step is that the sampler moves us towards places in model space with progressively higher posterior probability. Steps backward to lower probability parts of model space are taken once in a while to ensure proper movement over all regions of model space.

Given proper but relatively uninformative priors for the parameters in the spatial autoregressive model, it should be straightforward to compute the model probabilities $\mathrm{pr}(M_k|D)$. As noted above, devising appropriate priors is not a trivial task.

On drawback to this approach is that the set of models considered is based on a distribution of weight matrices centered on the known spatial weight matrix, so we are limiting the range of uncertainty regarding the spatial weight matrix that we incorporate in our Bayesian solution. The use of a grid to select only $2K$ sets of weight matrices limits the range of alternative weights even more. Nonetheless, this would provide a first step in the direction of estimating a spatial weight matrix, which has the advantage that the estimates from the model incorporate some of the uncertainty associated with fixing this matrix at the outset of our analysis.

8.4 Estimating bandwidth in GWR models

GWR estimates as well as BGWR model estimates are conditional on the bandwidth, or q the number of nearest neighbors used in the case of tri-cube weighting. A BMA solution to this would incorporate the bandwidth parameter as part of the estimation problem, simultaneously determining the bandwidth along with the other parameters in the model.

As in the cases discussed in Sections ?? to ??, the first task would be to find an appropriate informative prior for this parameter that is fairly vague, so the sample data plays an important role in determining the results.

Given that this problem is solved, the case of tri-cube weighting seems most amenable to the BMA approach. We can define a neighboring model as one based on one more or one fewer neighbors. We saw in Chapter 7 that the posterior probability for a given model could be easily calculated during Gibbs sampling. The same would be true of the marginal likelihood and Bayes factors given an appropriate prior over the number of nearest neighbors. It would be relatively easy to consider models based all values of q in the limited range from $k+2$ to $5k$ that we used when implementing the GWR model for tri-cube weighting. A posterior probability weighted average over these models could then be constructed in a relatively straightforward fashion.

The case of exponential and Gaussian weighting that depend on the bandwidth parameter are more difficult, but limits were placed on the range over which we conducted the simplex search for optimal values of this parameter. This implies that an informative but relatively flat prior could be devised. Again, given this prior, the marginal likelihood and Bayes factors could in theory be derived, allowing a BMA solution.

8.5 Chapter Summary

The advent of Metropolis MCMC sampling has greatly altered the nature and scope of Bayesian models that can be computationally implemented. This text demonstrated cases where Bayesian methods previously hindered by the need to rely on high-dimensional integration can be quickly solved on desktop computers. Application of this approach to spatial econometric models and methods is in its infancy, but holds great promise for the future.

Problems of model specification that have long plagued practitioners might be amenable to Bayesian model averaging solutions that have been implemented for the standard linear regression model. These solutions would have the virtue of incorporating model uncertainty regarding: specification of the variables to include in the model, the spatial weight matrix used, and the spatial structure of the model. Further, the solution would resolve all of these issues in a single coherent framework.

Some of the approaches already developed for linear regression models are directly applicable to GWR models, while others await some additional work on specifying appropriate priors and other implementation details that are at present unresolved.

References

Albert, James H. and Siddhartha Chib. 1993. "Bayesian Analysis of Binary and Polychotomous Response Data", *Journal of the American Statistical Association*, Vol. 88, pp. 669-679.

Amemiya, T. 1985. *Advanced Econometrics*, (Cambridge, MA: Harvard University Press).

Anselin, L. 1980. *Estimation Methods for Spatial Autoregressive Structures,* (Ithaca, New York: Regional Science Dissertation and Monograph Series #8).

Anselin, L. 1988. *Spatial Econometrics: Methods and Models*, (Dordrecht: Kluwer Academic Publishers).

Anselin, L. and R.J.G. Florax. 1994. "Small Sample Properties of Tests for Spatial Dependence in Regression Models: Some Further Results", Research paper 9414, *Regional Research Institute*, West Virginia University, Morgantown, West Virginia.

Anselin, L. and D.A. Griffith. 1988. "Do spatial effects really matter in regression analysis? *Papers of the Regional Science Association*, Vol. 65, pp. 11-34.

Anselin, L. and S. Rey. 1991. "Properties of tests for spatial dependence in linear regression models", *Geographical Analysis*, Vol. 23, pages 112-31.

Anselin, L. and A. Smirnov. 1994. "Efficient Algorithms for Constructing Proper Higher Order spatial lag operators", *West Virginia University Research Paper 9432.*

Becker, R.A., J.M. Chambers, and A.R. Wilks. 1988. *The new S Language: a programming environment for data analysis and graphics.* (Pacific Grove, CA: Wadsworth and Brooks/Cole Advanced Books and Software).

Belsley, David A. Edwin Kuh, and Roy E. Welsch. 1980. *Regression Diagnostics* (New York: John Wiley & Sons Inc.).

Best, N.G., M.K. Cowles, and S.K. Vines. 1995. *CODA: Manual version 0.30.* Biostatistics Unit, Cambridge U.K. http://www.mrc-bsu.cam.ac.uk

Blommestein, Hans J. 1985. "Elimination of Circular Routes in Spatial Dynamic Regression Equations," *Regional Science and Urban Economics*, Vol 15, pp. 121-130.

Bolduc, D., M.G. Dagenais and M.J. Gaudry. 1989. "Spatially autocorrelated errors in origin-destination models: A new specification applied to urban travel demand in Winnipeg", *Transportation Research B*, Vol. 23, pp. 361-372.

Bolduc, D., R. Laferriere and Gino Santarossa. 1995. "Spatial autoregressive error components in travel flow models: An application to aggregate model choice", in *New directions in spatial econometrics*, L. Anselin and R.J.G.M Florax (eds.) (Berlin: Springer-Verlag).

Brundson, C. A. S. Fotheringham, and M. Charlton. 1996. "Geographically weighted regression: a method for exploring spatial nonstationarity," *Geographical Analysis*, Vol. 28, pp. 281-298.

Casella, G. and E.I. George. 1992. "Explaining the Gibbs Sampler", *American Statistician*, Vol. 46, pp. 167-174.

Casetti, Emilio. 1972. "Generating Models by the Expansion Method: Applications to Geographic Research", *Geographical Analysis*, Vol. 4, pp. 81-91.

Casetti, Emilio. 1982. "Drift Analysis of Regression Parameters: An Application to the Investigation of Fertility Development Relations", *Modeling and Simulation 13, Part 3:*, pp. 961-66.

Casetti, E. 1992. "Bayesian Regression and the Expansion Method", *Geographical Analysis*, Vol. 24, pp. 58-74.

Casetti, E. and A. Can. 1998. "The Econometric estimation and testing of DARP models." Paper presented at the RSAI meetings, Sante Fe, New Mexico.

Chib, Siddhartha. 1992. "Bayes Inference in the Tobit Censored Regression Model", *Journal of Econometrics*, Vol. 51, pp. 79-99.

Chow, G. 1983. *Econometrics*, (New York: McGraw-Hill).

Cliff, A. and J. Ord. 1972. "Testing for spatial autocorrelation among regression residuals", *Geographical Analysis*, Vol. 4, pp. 267-84.

Cliff, A. and J. Ord. 1973. *Spatial Autocorrelation*, (London: Pion).

Cliff, A. and J. Ord. 1981. *Spatial Processes, Models and Applications*, (London: Pion).

Cooper, J. Phillip. 1973. "Time Varying Regression Coefficients: A Mixed Estimation Approach and Operational Limitations of the General Markov Structure," *Annals of Economic and Social Measurement*, Vol. 2, no. 4, pp. 525-530.

Cressie, Noel. 1991. *Statistics for Spatial Data*, (New York: Wiley).

Dempster, A.P. N.M. Laird and D.B. Rubin. 1977. "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, Series B, Vol. 39, pp. 1-38.

Dhrymes, P. 1981. *Distributed Lags: Problems of Estimation and Formulation*, (Amsterdam: North-Holland).

Dubin, Robin. 1995. "Estimating logit models with spatial dependence", in *New directions in spatial econometrics*, L. Anselin and R.J.G.M Florax (eds.) (Berlin: Springer-Verlag).

Estrella, Artuto. 1998. "A new measure of fit for equations with dichotomous dependent variable", *Journal of Business & Economic Statistics*, Vol. 16, no. 2, pp. 198-205.

Fomby, T., R. Hill, and S. Johnson. 1984. *Advanced Econometric Methods*, (New York: Springer).

Fernandez, Carmen, Eduardo Ley and Mark F.J. Steel. 1998. "Benchmark priors for Bayesian model averaging", Working paper #98-06, Fedea, Madrid, Spain.

Gelfand, Alan E., and A.F.M Smith. 1990. "Sampling-Based Approaches to Calculating Marginal Densities", *Journal of the American Statistical Association*, Vol. 85, pp. 398-409.

Gelfand, Alan E., Susan E. Hills, Amy Racine-Poon and Adrian F.M. Smith. 1990. "Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling", *Journal of the American Statistical Association*, Vol. 85, pp. 972-985.

Gelman, Andrew, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 1995. *Bayesian Data Analysis*, (London: Chapman & Hall).

Geman, S., and D. Geman. 1984. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 6, pp. 721-741.

George, Alan and Joseph Liu. 1981. *Computer Solution of Large Sparse Positive Definite Systems*, (Englewood Cliffs: Prentice-Hall).

Geweke, John. 1991. "Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments", *Federal Reserve Bank of Minneapolis, Research Department* Staff Report 148.

Geweke, John. 1993. "Bayesian Treatment of the Independent Student t Linear Model", *Journal of Applied Econometrics*, Vol. 8, pp. 19-40.

Gilks, W.R., S. Richardson and D.J. Spiegelhalter. 1996. *Markov Chain Monte Carlo in Practice*, (London: Chapman & Hall).

Gilley, O.W., and R. Kelley Pace. 1996. "On the Harrison and Rubinfeld Data," *Journal of Environmental Economics and Management*, Vol. 31 pp. 403-405.

Green, W. H. 1997. *Econometric Analysis, third edition*, (Upper Saddle River, N.J: Prentice Hall).

Hanselmann, D. and B. Littlefield. 1997. *The Student Edition of MATLAB, Version 5 User's Guide*. (New Jersey: Prentice Hall).

Harrison, D. and D.L. Rubinfeld, D.L. 1978. 'Hedonic prices and the demand for clean air', *Journal of Environmental Economics & Management*, Vol.5, pp. 81-102.

Hastings, W. K. 1970. "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, Vol. 57, pp. 97-109.

Intrilligator, M. 1978. *Econometric Models, Techniques, and Applications*, (Englewood Cliffs: Prentice-Hall).

Kelejian, H. and W. Oates. 1989. *Introduction to Econometrics: Principles and Applications*, (New York: Harper and Row).

Kelejian, H. H. and D. P. Robinson. 1995. "Spatial Correlation: A suggested alternative to the autoregressive model", in *New Directions in Spatial Econometrics*, L. Anselin and R.J.G.M. Florax (eds.). (Berlin: Springer).

Kmenta, J. 1971. *Elements of Econometrics*, (New York: Macmillan).

Lange, K.L., R.J.A. Little, and J.M.G. Taylor. 1989. "Robust Statistical Modeling Using the t Distribution," *Journal of the American Statistical Association*, Vol. 84, pp. 881-896.

Leamer, Edward E. 1978. *Specification Searches*, (New York: Wiley).

Leamer, Edward E. 1983. "Model Choice and Specification Analysis", in *Handbook of Econometrics, Volume 1*, Zvi Griliches and Michael D. Intriligator, eds. (North-Holland: Amsterdam).

LeSage, J.P. 1997. "Bayesian estimation of spatial autoregressive models", *International Regional Science Review*, Vol. 20, pp. 113-129.

Lindley, David V. 1971. "The estimation of many parameters," in *Foundations of Statistical Science*, V.P. Godambe and D.A. Sprout (eds.) (Toronto: Holt, Rinehart, and Winston).

Maddala, G.S. 1977. *Econometrics*, (New York: McGraw-Hill).

McMillen, Daniel P. 1992. "Probit with spatial autocorrelation", *Journal of Regional Science*, Vol. 32, pp. 335-348.

McMillen, Daniel P. 1996. "One Hundred Fifty Years of Land Values in Chicago: A Nonparametric Approach," *Journal of Urban Economics*, Vol. 40, pp. 100-124.

McMillen, Daniel P. and John F. McDonald. 1997. "A Nonparametric Analysis of Employment Density in a Polycentric City," *Journal of Regional Science*, Vol. 37, pp. 591-612.

McFadden, Daniel. 1984. "Econometric Analysis of Qualitative Response Models", in Zvi Griliches and Michael D. Intriligator, eds. *Handbook of Econometrics, Volume 2*, (North-Holland: Amsterdam).

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller. 1953. "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, Vol. 21, pp. 1087-1092.

Pace, R. Kelley. 1993. "Nonparametric Methods with Applications to Hedonic Models," *Journal of Real Estate Finance and Economics* Vol. 7, pp. 185-204.

Pace, R. Kelley and Ronald Barry. 1997. "Quick Computation of Spatial Autoregressive Estimators", forthcoming in *Geographical Analysis*.

Pace, R. Kelley, and R. Barry. 1998. "Simulating mixed regressive spatially autoregressive estimators," *Computational Statistics*, Vol. 13 pp. 397-418.

Pace, R. Kelley, and O.W. Gilley. 1997. "Using the Spatial Configuration of the Data to Improve Estimation," *Journal of the Real Estate Finance and Economics* Vol. 14 pp. 333-340.

Pindyck, R. and D. Rubinfeld. 1981. *Econometric Models and Economic Forecasts*, (New York: McGraw-Hill).

Raftery, Adrian E., David Madigan and Jennifer A. Hoeting. 1997. "Bayesian model averaging for linear regression models,", *Journal of the American Statistical Association*, Vol. 92, pp. 179-191.

Schmidt, P. 1976. *Econometrics*, (New York: Marcel Dekker).

Smith, A.F.M and G.O. Roberts. 1992. "Bayesian Statistics without Tears: A Sampling-Resampling Perspective", *The American Statistician*, Vol. 46, pp. 84-88.

Theil, Henri and Arthur S. Goldberger. 1961. "On Pure and Mixed Statistical Estimation in Economics," *International Economic Review*, Vol. 2, pp. 65-78.

Vinod, H. and A. Ullah. 1981. *Recent Advances in Regression Methods*, (New York: Marcel Dekker).

Zellner, Arnold. (1971) *An Introduction to Bayesian Inference in Econometrics*. (New York: John Wiley & Sons.)

Econometrics Toolbox functions

The *Econometrics Toolbox* is organized in a set of directories, each containing a different library of functions. When your Internet browser unpacks the compressed file containing the *Econometrics Toolbox* the files will be placed in the appropriate directories.

To install the toolbox:

1. create a single subdirectory in the MATLAB toolbox directory:

   ```
   C:\matlab\toolbox\econ
   ```

   Where we have used the name **econ** for the directory.

2. Copy the system of directories to this subdirectory.

3. Use the graphical path tool in MATLAB to add these directories to your path. (On a unix or linux system, you may need to edit your environment variables that set the MATLAB path.) the graphical path tool in MATLAB to add these directories to your path. (On a unix or linux system, you may need to edit your environment variables that set the MATLAB path.)

A listing of the contents file from each subdirectory is presented on the following pages.

A library of spatial econometrics functions are in the subdirectory **spatial**.

```
------- spatial econometrics functions -----------

bcasetti   - Bayesian spatial expansion model
bgwr       - Bayesian geographically weighted regression
bgwrv      - robust geographically weighted regression
casetti    - Casetti's spatial expansion model
darp       - Casetti's darp model
far        - 1st order spatial AR model    - y = pWy + e
far_g      - Gibbs sampling Bayesian far model
gwr        - geographically weighted regression
gwr_logit  - logit version of GWR model
gwr_probit - probit version of GWR model
lmerror    - LM error statistic for regression model
lmsar      - LM error statistic for sar model
lratios    - Likelihood ratio statistic for regression models
moran      - Moran's I-statistic
normw      - normalizes a spatial contiguity matrix
normxy     - isotropic normalization of x-y coordinates
sac        - spatial model  - y = p*W1*y + X*b + u, u = c*W2*u + e
sac_g      - Gibbs sampling Bayesian sac model
sacp_g     - Gibbs sampling Bayesian sac probit model
sact_g     - Gibbs sampling Bayesian sac tobit model
sar        - spatial autoregressive model  - y = p*W*y + X*b + e
sar_g      - Gibbs sampling Bayesian sar model
sarp_g     - Gibbs sampling Bayesian sar probit model
sart_g     - Gibbs sampling Bayesian sar tobit model
sdm        - spatial Durbin model  y = a + X*b1 + W*X*b2 + e
sdm_g      - Gibbs sampling Bayesian sdm model
sdmp_g     - Gibbs sampling Bayesian sdm probit model
sdmt_g     - Gibbs sampling Bayesian sdm tobit model
sem        - spatial error model  - y = X*b +u, u=c*W + e
sem_g      - Gibbs sampling Bayesian spatial error model
semo       - spatial error model (optimization solution)
semp_g     - Gibbs sampling Bayesian spatial error probit model
semt_g     - Gibbs sampling Bayesian spatial error tobit model
slag       - creates spatial lags
walds      - Wald test for regression models
xy2cont    - constructs a contiguity matrix from x-y coordinates

------- demonstration programs  -----------

bcasetti_d - Bayesian spatial expansion demo
bgwr_d     - demo of Bayesian GWR
bgwr_d2    - BGWR demo with Harrison-Rubinfeld Boston data
bgwrv_d    - BGWRV demo
casetti_d  - Casetti model  demo
darp_d     - Casetti darp demo
darp_d2    - darp for all data observations
far_d      - demonstrates far using a small data set
far_d2     - demonstrates far using a large data set
far_gd     - far Gibbs sampling with small data set
far_gd2    - far Gibbs sampling with large data set
gwr_d      - geographically weighted regression demo
gwr_d2     - GWR demo with Harrison-Rubinfeld Boston data
gwr_logitd - GWR logit demo
gwr_probitd- GWR probit demo
```

```
lmerror_d  - lmerror demonstration
lmsar_d    - lmsar demonstration
lratios_d  - likelihood ratio demonstration
moran_d    - moran demonstration
sac_d      - sac model demo
sac_d2     - sac model demonstration large data set
sac_gd     - sac Gibbs sampling demo
sac_gd2    - sac Gibbs demo with large data set
sacp_gd    - sac Gibbs probit demo
sact_gd    - sac Gibbs tobit demo
sact_gd2   - sac tobit right-censoring demo
sar_d      - sar model demonstration
sar_d2     - sar model demonstration large data set
sar_gd     - sar Gibbs sampling demo
sar_gd2    - sar Gibbs demo with large data set
sarp_gd    - sar probit Gibbs sampling demo
sart_gd    - sar tobit model Gibbs sampling demo
sart_gd2   - sar tobit right-censoring demo
sdm_d      - sdm model demonstration
sdm_d2     - sdm model demonstration large data set
sdm_gd     - sdm Gibbs sampling demo
sdm_gd2    - sdm Gibbs demo with large data set
sdmp_g     - sdm Gibbs probit demo
sdmt_g     - sdm Gibbs tobit demo
sem_d      - sem model demonstration
sem_d2     - sem model demonstration large data set
sem_gd     - sem Gibbs sampling demo
sem_gd2    - sem Gibbs demo with large data set
semo_d     - semo function demonstration
semo_d2    - semo demo with large data set
semp_gd    - sem Gibbs probit demo
semt_gd    - sem Gibbs tobit demo
semt_gd2   - sem tobit right-censoring demo
slag_d     - demo of slag function
walds_d    - Wald test demonstration
xy2cont_d  - xy2cont demo

------- support functions  -----------

anselin.dat- Anselin (1988) Columbus crime data
boston.dat - Harrison-Rubinfeld Boston data set
c_far      - used by far_g
c_sac      - used by sac_g
c_sar      - used by sar_g
c_sdm      - used by sdm_g
c_sem      - used by sem_g
darp_lik1  - used by darp
darp_lik2  - used by darp
elect.dat  - Pace and Barry 3,107 obs data set
f2_far     - far model likelihood
f2_sac     - sac model likelihood
f2_sar     - sar model likelihood
f2_sdm     - sdm model likelihood
f2_sem     - sem model likelihood
f3_sem     - semo model likelihood
f_far      - far model likelihood (concentrated)
f_sac      - sac model likelihood (concentrated)
```

```
f_sar      - sar model likelihood (concentrated)
f_sdm      - sdm model likelihood (concentrated)
f_sem      - sem model likelihood (concentrated)
ford.dat   - Pace and Barry 1st order contiguity matrix
gwr_g      - used by BGWRV
latit.dat  - latittude for HR data
longi.dat  - longitude for HR data
plt_spat   - plots results from spatial models
prt_gwr    - prints gwr_reg results structure
prt_spat   - prints results from spatial models
scoref     - used by gwr
scoref_log - used by gwr_logit
scoref_prob- used by gwr_probit
scoreq     - used by gwr
wmat.dat   - Anselin (1988) 1st order contiguity matrix
```

The *regression function library* is in a subdirectory **regress**.

```
regression function library

------- regression program functions -----------

ar_g        - Gibbs sampling Bayesian autoregressive model
bma_g       - Gibbs sampling Bayesian model averaging
boxcox      - Box-Cox regression with 1 parameter
boxcox2     - Box-Cox regression with 2 parameters
hmarkov_em  - Hamilton's Markov switching regression
hwhite      - Halbert White's heteroscedastic consistent estimates
lad         - least-absolute deviations regression
lm_test     - LM-test for two regression models
logit       - logit regression
mlogit      - multinomial logit regression
nwest       - Newey-West hetero/serial consistent estimates
ols         - ordinary least-squares
ols_g       - Gibbs sampling Bayesian linear model
olsar1      - Maximum Likelihood for AR(1) errors ols model
olsc        - Cochrane-Orcutt AR(1) errors ols model
olst        - regression with t-distributed errors
probit      - probit regression
probit_g    - Gibbs sampling Bayesian probit model
ridge       - ridge regression
rtrace      - ridge estimates vs parameters (plot)
robust      - iteratively reweighted least-squares
sur         - seemingly unrelated regressions
switch_em   - switching regime regression using EM-algorithm
theil       - Theil-Goldberger mixed estimation
thsls       - three-stage least-squares
tobit       - tobit regression
tobit_g     - Gibbs sampling Bayesian tobit model
tsls        - two-stage least-squares
waldf       - Wald F-test

-------- demonstration programs -----------------

ar_gd       - demonstration of Gibbs sampling ar_g
bma_gd      - demonstrates Bayesian model averaging
box_cox_d   - demonstrates Box-Cox 1-parameter model
boxcox2_d   - demonstrates Box-Cox 2-parmaeter model
```

```
demo_all    - demos most regression functions
hmarkov_emd - demos Hamilton's Markov switching regression
hwhite_d    - H. White's hetero consistent estimates demo
lad_d       - demos lad regression
lm_test_d   - demos lm_test
logit_d     - demonstrates logit regression
mlogit_d    - demonstrates multinomial logit
nwest_d     - demonstrates Newey-West estimates
ols_d       - demonstrates ols regression
ols_d2      - Monte Carlo demo using ols regression
ols_gd      - demo of Gibbs sampling ols_g
olsar1_d    - Max Like AR(1) errors model demo
olsc_d      - Cochrane-Orcutt demo
olst_d      - olst demo
probit_d    - probit regression demo
probit_gd   - demo of Gibbs sampling Bayesian probit model
ridge_d     - ridge regression demo
robust_d    - demonstrates robust regression
sur_d       - demonstrates sur using Grunfeld's data
switch_emd  - demonstrates switching regression
theil_d     - demonstrates theil-goldberger estimation
thsls_d     - three-stage least-squares demo
tobit_d     - tobit regression demo
tobit_gd    - demo of Gibbs sampling Bayesian tobit model
tsls_d      - two-stage least-squares demo
waldf_d     - demo of using wald F-test function

-------- Support functions ------------------------

ar1_like    - used by olsar1   (likelihood)
bmapost     - used by bma_g
box_lik     - used by box_cox  (likelihood)
box_lik2    - used by box_cox2 (likelihood)
boxc_trans  - used by box_cox, box_cox2
chis_prb    - computes chi-squared probabilities
dmult       - used by mlogit
fdis_prb    - computes F-statistic probabilities
find_new    - used by bma_g
grun.dat    - Grunfeld's data used by sur_d
grun.doc    - documents Grunfeld's data set
lo_like     - used by logit    (likelihood)
maxlik      - used by tobit
mcov        - used by hwhite
mderivs     - used by mlogit
mlogit_lik  - used by mlogit
nmlt_rnd    - used by probit_g
nmrt_rnd    - used by probit_g, tobit_g
norm_cdf    - used by probit, pr_like
norm_pdf    - used by prt_reg, probit
olse        - ols returning only residuals (used by sur)
plt         - plots  everything
plt_eqs     - plots equation systems
plt_reg     - plots regressions
pr_like     - used by probit   (likelihood)
prt         - prints everything
prt_eqs     - prints equation systems
prt_gibbs   - prints Gibbs sampling models
```

```
prt_reg     - prints regressions
prt_swm     - prints switching regression results
sample      - used by bma_g
stdn_cdf    - used by norm_cdf
stdn_pdf    - used by norm_pdf
stepsize    - used by logit,probit to determine stepsize
tdis_prb    - computes t-statistic probabilities
to_like     - used by tobit    (likelihood)
```

The utility functions are in a subdirectory **util**.

```
utility function library

 -------- utility functions ----------------------------

accumulate  - accumulates column elements of a matrix
cal         - associates obs # with time-series calendar
ccorr1      - correlation scaling to normal column length
ccorr2      - correlation scaling to unit column length
fturns      - finds turning-points in a time-series
growthr     - converts time-series matrix to growth rates
ical        - associates time-series dates with obs #
indicator   - converts a matrix to indicator variables
invccorr    - inverse for ccorr1, ccorr2
lag         - generates a lagged variable vector or matrix
levels      - generates factor levels variable
lprint      - prints a matrix in LaTeX table-formatted form
matdiv      - divide matrices that aren't totally conformable
mlag        - generates a var-type matrix of lags
mode        - calculates the mode of a distribution
mprint      - prints a matrix
mth2qtr     - converts monthly to quarterly data
nclag       - generates a matrix of non-contiguous lags
plt         - wrapper function, plots all result structures
prt         - wrapper function, prints all result strucutres
sacf        - sample autocorrelation function estimates
sdiff       - seasonal differencing
sdummy      - generates seasonal dummy variables
shist       - plots spline smoothed histogram
spacf       - sample partial autocorrelation estimates
tally       - computes frequencies of distinct levels
tdiff       - time-series differencing
tsdate      - time-series dates function
tsprint     - print time-series matrix
unsort      - unsorts a sorted vector or matrix
vec         - turns a matrix into a stacked vector

-------- demonstration programs -----------------------

cal_d.m     - demonstrates cal function
fturns_d    - demonstrates fturns and plt
ical_d.m    - demonstrates ical function
lprint_d.m  - demonstrates lprint function
mprint_d.m  - demonstrates mprint function
sacf_d      - demonstrates sacf
spacf_d     - demonstrates spacf
tsdate_d.m  - demonstrates tsdate function
tsprint_d.m - demonstrates tsprint function
```

```
util_d.m    - demonstrated some of the utility functions

-------- functions to mimic Gauss functions -------------

cols        - returns the # of columns in a matrix or vector
cumprodc    - returns cumulative product of each column of a matrix
cumsumc     - returns cumulative sum of each column of a matrix
delif       - select matrix values for which a condition is false
indexcat    - extract indices equal to a scalar or an interval
invpd       - makes a matrix positive-definite, then inverts
matadd      - adds non-conforming matrices, row or col compatible.
matdiv      - divides non-conforming matrices, row or col compatible.
matmul      - multiplies non-conforming matrices, row or col compatible.
matsub      - divides non-conforming matrices, row or col compatible.
prodc       - returns product of each column of a matrix
rows        - returns the # of rows in a matrix or vector
selif       - select matrix values for which a condition is true
seqa        - a sequence of numbers with a beginning and increment
stdc        - std deviations of columns  returned as a column vector
sumc        - returns sum of each column
trimc       - trims columns of a matrix (or vector) like Gauss
trimr       - trims rows of a matrix (or vector) like Gauss
```

A set of graphing functions are in a subdirectory **graphs**.

```
graphing function library

 -------- graphing programs ---------------------------

pairs       - scatter plot (uses histo)
pltdens     - density plots
tsplot      - time-series graphs

-------- demonstration programs -----------------------

pairs_d     - demonstrates pairwise scatter
pltdens_d   - demonstrates pltdens
tsplot_d    - demonstrates tsplot

------- support functions -----------------------------

histo       - used by pairs
plt_turns   - plots turning points from fturns function
```

A library of routines in the subdirectory **diagn** contain the regression diagnostics functions.

```
regression diagnostics library

-------- diagnostic programs ---------------

bkw              - BKW collinearity diagnostics
bpagan           - Breusch-Pagan heteroscedasticity test
cusums           - Brown,Durbin,Evans cusum squares test
dfbeta           - BKW influential observation diagnostics
diagnose         - compute diagnostic statistics
```

```
rdiag           - graphical residuals diagnostics
recresid        - compute recursive residuals
studentize      - standarization transformation

------- demonstration programs -------------

bkw_d           - demonstrates bkw
bpagan_d        - demonstrates bpagan
cusums_d        - demonstrates cusums
dfbeta_d        - demonstrates dfbeta, plt_dfb, plt_dff
diagnose_d      - demonstrates diagnose
rdiag_d         - demonstrates rdiag
recresid_d      - demonstrates recresid

------- support functions ------------------

ols.m           - least-squares regression
plt             - plots everything
plt_cus         - plots cusums test results
plt_dfb         - plots dfbetas
plt_dff         - plots dffits
```

The vector autoregressive library is in a subdirectory **var_bvar**.

```
 vector autoregressive function library

------- VAR/BVAR program functions -----------

 becm_g     - Gibbs sampling BECM estimates
 becmf      - Bayesian ECM model forecasts
 becmf_g    - Gibbs sampling BECM forecasts
 bvar       - BVAR model
 bvar_g     - Gibbs sampling BVAR estimates
 bvarf      - BVAR model forecasts
 bvarf_g    - Gibbs sampling BVAR forecasts
 ecm        - ECM (error correction) model estimates
 ecmf       - ECM model forecasts
 lrratio    - likelihood ratio lag length tests
 pftest     - prints Granger F-tests
 pgranger   - prints Granger causality probabilities
 recm       - ecm version of rvar
 recm_g     - Gibbs sampling random-walk averaging estimates
 recmf      - random-walk averaging ECM forecasts
 recmf_g    - Gibbs sampling random-walk averaging forecasts
 rvar       - Bayesian random-walk averaging prior model
 rvar_g     - Gibbs sampling RVAR estimates
 rvarf      - Bayesian RVAR model forecasts
 rvarf_g    - Gibbs sampling RVAR forecasts
 var        - VAR model
 varf       - VAR model forecasts

------ demonstration programs  --------------

 becm_d     - BECM model demonstration
 becm_gd    - Gibbs sampling BECM estimates demo
 becmf_d    - becmf demonstration
 becmf_gd   - Gibbs sampling BECM forecast demo
 bvar_d     - BVAR model demonstration
```

```
 bvar_gd    - Gibbs sampling BVAR demonstration
 bvarf_d    - bvarf demonstration
 bvarf_gd   - Gibbs sampling BVAR forecasts demo
 ecm_d      - ECM model demonstration
 ecmf_d     - ecmf demonstration
 lrratio_d  - demonstrates lrratio
 pftest_d   - demo of pftest function
 recm_d     - RECM model demonstration
 recm_gd    - Gibbs sampling RECM model demo
 recmf_d    - recmf demonstration
 recmf_gd   - Gibbs sampling RECM forecast demo
 rvar_d     - RVAR model demonstration
 rvar_g     - Gibbs sampling rvar model demo
 rvarf_d    - rvarf demonstration
 rvarf_gd   - Gibbs sampling rvar forecast demo
 var_d      - VAR model demonstration
 varf_d     - varf demonstration

------- support functions  -----------------

 johansen   - used by ecm,ecmf,becm,becmf,recm,recmf
 lag        - does ordinary lags
 mlag       - does var-type lags
 nclag      - does contiguous lags (used by rvar,rvarf,recm,recmf)
 ols        - used for VAR estimation
 prt        - prints results from all functions
 prt_coint  - used by prt_var for ecm,becm,recm
 prt_var    - prints results of all var/bvar models
 prt_varg   - prints results of all Gibbs var/bvar models
 rvarb      - used for RVARF forecasts
 scstd      - does univariate AR for BVAR
 theil_g    - used for Gibbs sampling estimates and forecasts
 theilbf    - used for BVAR forecasts
 theilbv    - used for BVAR estimation
 trimr      - used by VARF,BVARF, johansen
 vare       - used by lrratio
```

The co-integration library functions are in a subdirectory **coint**.

```
 co-integration library

 ------ co-integration testing routines --------

 adf        - carries out Augmented Dickey-Fuller unit root tests
 cadf       - carries out ADF tests for co-integration
 johansen   - carries out Johansen's co-integration tests

 ------ demonstration programs -----------------

 adf_d      - demonstrates adf
 cadf_d     - demonstrates cadf
 johansen_d - demonstrates johansen

 ------ support functions ----------------------

 c_sja      - returns critical values for SJ maximal eigenvalue test
 c_sjt      - returns critical values for SJ trace test
 cols       - (like Gauss cols)
```

```
detrend    - used by johansen to detrend data series
prt_coint  - prints results from adf,cadf,johansen
ptrend     - used by adf to create time polynomials
rows       - (like Gauss rows)
rztcrit    - returns critical values for cadf test
tdiff      - time-series differences
trimr      - (like Gauss trimr)
ztcrit     - returns critical values for adf test
```

The Gibbs convergence diagnostic functions are in a subdirectory **gibbs**.

```
Gibbs sampling convergence diagnostics functions

--------- convergence testing functions ---------

apm       - Geweke's chi-squared test
coda      - convergence diagnostics
momentg   - Geweke's NSE, RNE
raftery   - Raftery and Lewis program Gibbsit for convergence

--------- demonstration programs ----------------

apm_d     - demonstrates apm
coda_d    - demonstrates coda
momentg_d - demonstrates momentg
raftery_d - demonstrates raftery

--------- support functions ---------------------

prt_coda  - prints coda, raftery, momentg, apm output  (use prt)
empquant  - These were converted from:
indtest   - Rafferty and Lewis FORTRAN program.
mcest     - These function names follow the FORTRAN subroutines
mctest    -
ppnd      -
thin      -
```

Distribution functions are in the subdirectory **distrib**.

```
Distribution functions library

------- pdf, cdf, inverse functions -----------

beta_cdf  - beta(a,b) cdf
beta_inv  - beta inverse (quantile)
beta_pdf  - beta(a,b) pdf
bino_cdf  - binomial(n,p) cdf
bino_inv  - binomial inverse (quantile)
bino_pdf  - binomial pdf
chis_cdf  - chisquared(a,b) cdf
chis_inv  - chi-inverse (quantile)
chis_pdf  - chisquared(a,b) pdf
chis_prb  - probability for chi-squared statistics
fdis_cdf  - F(a,b) cdf
fdis_inv  - F inverse (quantile)
fdis_pdf  - F(a,b) pdf
fdis_prb  - probabililty for F-statistics
```

```
gamm_cdf  - gamma(a,b) cdf
gamm_inv  - gamma inverse (quantile)
gamm_pdf  - gamma(a,b) pdf
hypg_cdf  - hypergeometric cdf
hypg_inv  - hypergeometric inverse
hypg_pdf  - hypergeometric pdf
logn_cdf  - lognormal(m,v) cdf
logn_inv  - lognormal inverse (quantile)
logn_pdf  - lognormal(m,v) pdf
logt_cdf  - logistic cdf
logt_inv  - logistic inverse (quantile)
logt_pdf  - logistic pdf
norm_cdf  - normal(mean,var) cdf
norm_inv  - normal inverse (quantile)
norm_pdf  - normal(mean,var) pdf
pois_cdf  - poisson cdf
pois_inv  - poisson inverse
pois_pdf  - poisson pdf
stdn_cdf  - std normal cdf
stdn_inv  - std normal inverse
stdn_pdf  - std normal pdf
tdis_cdf  - student t-distribution cdf
tdis_inv  - student t inverse (quantile)
tdis_pdf  - student t-distribution pdf
tdis_prb  - probabililty for t-statistics

------- random samples -----------------------

beta_rnd  - random beta(a,b) draws
bino_rnd  - random binomial draws
chis_rnd  - random chi-squared(n) draws
fdis_rnd  - random F(a,b) draws
gamm_rnd  - random gamma(a,b) draws
hypg_rnd  - random hypergeometric draws
logn_rnd  - random log-normal draws
logt_rnd  - random logistic draws
nmlt_rnd  - left-truncated normal draw
nmrt_rnd  - right-truncated normal draw
norm_crnd - contaminated normal random draws
norm_rnd  - multivariate normal draws
pois_rnd  - poisson random draws
tdis_rnd  - random student t-distribution draws
unif_rnd  - random uniform draws (lr,rt) interval
wish_rnd  - random Wishart draws

-------- demonstration and test programs ------

beta_d    - demo of beta distribution functions
bino_d    - demo of binomial distribution functions
chis_d    - demo of chi-squared distribution functions
fdis_d    - demo of F-distribution functions
gamm_d    - demo of gamma distribution functions
hypg_d    - demo of hypergeometric distribution functions
logn_d    - demo of lognormal distribution functions
logt_d    - demo of logistic distribution functions
pois_d    - demo of poisson distribution functions
stdn_d    - demo of std normal distribution functions
```

```
 tdis_d    - demo of student-t distribution functions
 trunc_d   - demo of truncated normal distribution function
 unif_d    - demo of uniform random distribution function

-------- support functions ---------------------

 betacfj   - used by fdis_prb
 betai     - used by fdis_prb
 bincoef   - binomial coefficients
 com_size  - test and converts to common size
 gammalnj  - used by fdis_prb
 is_scalar - test for scalar argument
```

Optimization functions are in the subdirectory **optimize**.

```
Optimization functions library

 --------------- optimization functions -----------------

 dfp_min    - Davidson-Fletcher-Powell
 frpr_min   - Fletcher-Reeves-Polak-Ribiere
 maxlik     - general all-purpose optimization routine
 pow_min    - Powell conjugate gradient
 optsolv    - yet another general purpose optimization routine

 --------------- demonstration programs -----------------

 optim1_d     - dfp, frpr, pow, maxlik demo
 optim2_d     - optsolv demo
 optim3_d     - fmins demo

 --------------- support functions ----------------------

 apprgrdn      - computes gradient for optsolv
 box_like1     - used by optim3_d
 gradt         - computes gradient
 hessian       - evaluates hessian
 linmin        - line minimization routine (used by dfp, frpr, pow)
 stepsize      - stepsize determination
 tol_like1     - used by optim1_d, optim2_d
 updateh       - updates hessian
```