The Review of Regional Studies, Vol. 37, No. 1, 2007, pp. 5 – 27 *ISSN 1553-0892*

The Review of Regional Studies
Oklahoma State University

PySAL: A Python Library of Spatial Analytical Methods

Sergio J. Rey

Regional Analysis Laboratory, Department of Geography, San Diego State University, e-mail: serge@rohan.sdsu.edu

Luc Anselin

Spatial Analysis Laboratory and National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign, e-mail: anselin@uiuc.edu

Abstract

PySAL is an open source library for spatial analysis written in the object-oriented language Python. It is built upon shared functionality in two exploratory spatial data analysis packages—GeoDA and STARS—and is intended to leverage the shared development of these components. This paper presents an overview of the motivation behind and the design of PySAL, as well as suggestions for how the library can be used with other software projects. Empirical illustrations of several key components in a variety of spatial analytical problems are given, and plans for future development of PySAL are discussed.

Keywords: ESDA; Spatial econometrics; Python; GeoDA; STARS

JEL classification: C21; C88; R15

Rey's research was supported in part by National Science Foundation Grants BCS-0602581 and BCS-0433132. Anselin's research was supported in part by National Science Foundation Grants BCS-9978058 and BCS-043390 and by Grant RO1-CA-95949-01 from the National Cancer Institute. Myunghwa Hwang and Yongwook Kim assisted in the development of the PySAL code.

1. INTRODUCTION

This paper describes PySAL, an open source library for spatial analysis written in the object-oriented language Python. PySAL grew out of the software development activities that were part of the Center for Spatially Integrated Social Sciences Tools Project (Goodchild et al. 2000). This National Science Foundation infrastructure project had as its goals to:

- *facilitate dissemination* of spatial analysis software to social sciences;
- *develop a library* of spatial data analysis modules;
- *develop prototypes* implementing state of the art methods; and
- initiate and nurture a community of *open source* developers.

PySAL is a collaborative effort between Luc Anselin's research group at UIUC and Sergio Rey's research group at SDSU to develop a cross-platform library of spatial analysis functions written in Python. This combines the development activities of GeoDA/PySpace (Anselin, Syabri, and Kho 2006) and STARS—Space Time Analysis of Regional Systems (Rey and Janikas 2006). Both will continue to exist and exploit a common library of functions.

One particular subcomponent of PySAL is referred to as PySpace, an open source software development effort focused on the implementation of spatial statistical methods in general and spatial regression analysis in particular using Python and Numerical Python. Current activities deal with a set of classes and methods to carry out diagnostics for spatial correlation in linear regression models and to estimate spatial lag and spatial error specifications.

The goal of PySAL is to leverage existing software tools development underlying GeoDA/PySpace and STARS to yield a core library and application programming interface (API) that will serve three needs. First, to avoid duplication of effort in the development of core spatial data analysis functions, the teams are collaborating on key modules that can be shared across the different projects. As a result of this reorganization, the two projects will be able to focus on increased specialization and modularization of related functionality. For example, PySpace development can focus on advanced spatial econometric methods while STARS development can continue implementing new space-time methods, yet both will draw on jointly developed spatial weights classes. This avoids the need for separate but largely parallel efforts and also increases standardization of core classes and methods.[1] By pooling developer time on the shared weights classes, we have freed up resources that are being used for advances along specialized interests of the two projects.

The third need that PySAL seeks to address is a current void in the Python community where advanced spatial analytic modules are largely absent. While much work is

[1] We provide illustrations of this in Section 3.

being done on cartographic and GIS libraries in Python (Coles, Wagner, and Koormann 2004; Butler and Gillies 2005; Gillies and Lautaportti 2006), functionality dealing with state of the art spatial statistical and spatial econometric analysis is largely absent. Filling this void is important given the rapidly growing scientific community that has adopted Python as the language of choice.[2]

The existing Python-related cartographic and GIS efforts are part of a much larger movement in Open Source Geographic Information Systems. A recent inventory of open source packages that are designed to deal with spatial data identified over 237 such efforts (Lewis 2007). However, a close examination of the objectives of the projects listed reveals that the vast majority focus on spatial data manipulation and presentation. There is still a dearth of functionality that implements spatial statistical, econometric, and modeling techniques. This lack of software tools for geospatial analysis in the open source GIS movement mimics the early days of commercial GIS development. This then prompted many scholars to identify the lack of software support as an impediment for the dissemination of spatial analysis methods in empirical research (e.g., Haining 1989) and led to considerable efforts to remedy the situation (for a review, see Fischer and Getis 1997; Anselin 2005). The advantage of the current open source GIS efforts is that the very open source nature of the different projects facilitates their extension and integration with other software tools. Specifically, this provides opportunities to develop geospatial analysis tools that can be readily integrated with a wide range of mapping and other GIS functionality.

PySAL is intended to fill a particular niche in the growing field of spatial data analysis software.[3] There are currently two broad classes of implementation of spatial analysis packages. The first is those that are self-contained and implement a subset of analytical methods in user-friendly graphical interfaces. Chief among these are GeoDa, GeoVista Studio (Takatsuka and Gahegan 2002), CommonGIS (Andrienko and Andrienko 2005; Andrienko, Andrienko, and Voss 2003), among others. At the other extreme are efforts at implementing spatial analysis methods in packages for particular programming and data analysis environments. Prominent examples here include the R-Geo project (Bivand and Gebhardt 2000) and the econometrics toolbox for MATLAB (LeSage 1999). PySAL is envisaged as supporting both types of efforts, since the Python environment lends itself to command line execution through its interpreter as well as the bundling of code in user-friendly executables with a graphical user interface.

In the remainder of the paper we first briefly outline the overall design and main components of the library. We next provide several illustrations of how the modules in the library can be combined and delivered in a number of ways to address various spatial analytical questions, including computational geometry, the study of spatial dynamics,

[2] For example, see Langtangen (2006). Also, an overview of scientific computing projects using Python is given in wiki.python.org/moin/NumericAndScientific.

[3] For a recent overview of the field of spatial analysis software for the social sciences see Rey and Anselin (2006).

smoothing of rates, regionalization, spatial econometrics, and spatial analytical Web services. We close with some concluding comments.

2. DESIGN AND COMPONENTS

PySAL is not intended to reinvent a complete Geographic Information System. Rather, it is designed as a library that would enable sophisticated spatial analysis through various delivery formats. This ranges from simple command line interactive scripts, to self-contained packages with a graphical user interface and add-on modules to commercial off-the-shelf programs (e.g., to augment the spatial statistical toolbox of the ArcGIS software). The functionality of the library is geared to facilitate spatial statistical exploration and spatial econometric modeling and to avoid duplication of basic GIS functionality. The modular structure of the Python language effectively allows us to build upon other efforts in geovisualization and spatial data manipulation of the open source GIS movement.

We designed the modules in PySAL to be agnostic of the delivery mechanism, so that they can be flexibly integrated with alternative GUIs (e.g., Tkinter or wxPython), combined as external libraries with other software (e.g., ArcGIS), or mixed and matched with existing modules developed by others. The set of components in PySAL is designed to cover all steps of a spatial data analysis process, starting with reading various data formats and carrying out basic computational geometry, and moving on to a collection of specialized methods useful in spatial exploratory analysis and modeling. Intentionally, a key feature of PySAL is that it is self-contained and does not have any tight dependencies on external libraries beyond those available within Python. At the same time, because it is a library, components of PySAL can be combined with functionality from a different GIS or analytical package to carry out specialized analyses. Moreover, PySAL gains the high degree of portability across different platforms and operating systems inherent in the Python language.

A graphical overview of the key components of the current incarnation of PySAL is presented in Figure 1. It is organized into six main categories of functionality dealing with basic data operations such as the construction and manipulation of spatial weights and essential computational geometry functions, data exploration such as clustering methods and exploratory spatial data analysis, and spatial modeling such as spatial dynamics and spatial econometrics. Table 1 provides a complementary classification of the functionality included in PySAL. Here, a distinction is made between data analytic functions, intended to ease the reading, manipulation, and writing of common spatial data formats, and ESDA and modeling functions.

The weights module includes functionality to construct spatial weights from a range of input formats (including the standard ESRI shape files) and store the information efficiently in an internal data structure. This can then be exported to different file formats such as the GAL and GWT formats used by GeoDa and R and the MAT format used by the Matlab spatial econometric libraries. The computational geometry module supports

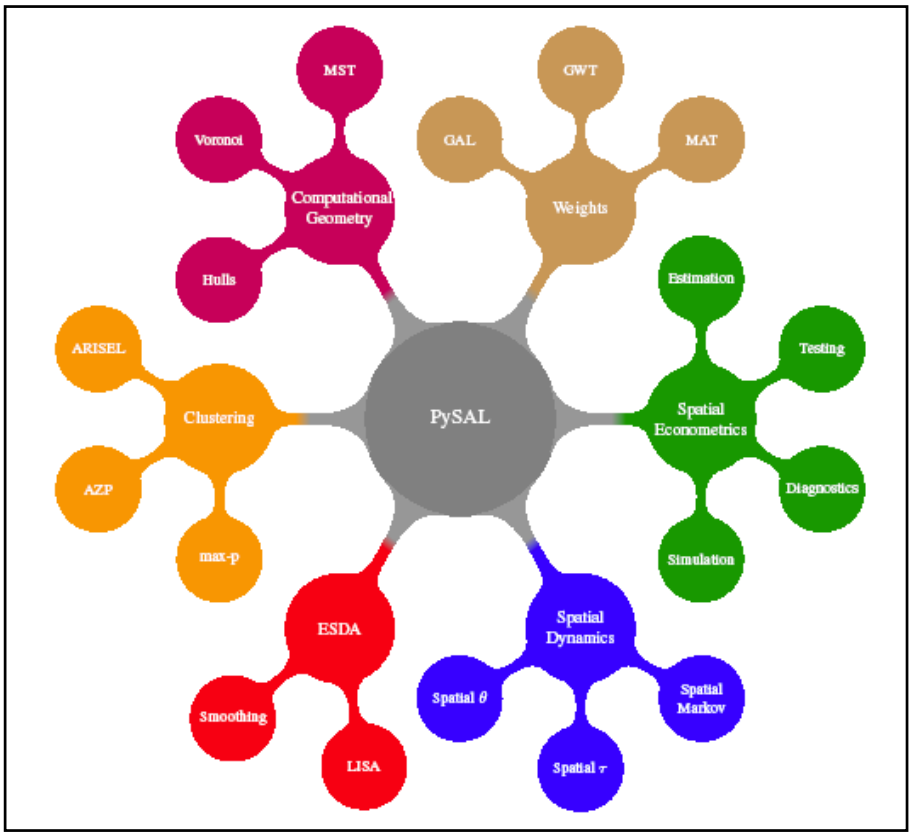

FIGURE 1. PySAL Components

various other modules in providing basic manipulations of spatial data such as the construction of Voronoi diagrams (Thiessen polygons), convex hulls, and minimum spanning trees. These underlie the derivation of network-based spatial weights as well as various computations in the clustering module.

Data exploration is supported by the clustering and ESDA modules. The clustering module implements a range of regionalization methods that can be used to simplify the data and provide alternatives to rate smoothing operations (in the ESDA module). They also form the basis for the construction of alternative spatial weights structures. The ESDA module contains different methods to implement the smoothing of rates as well as standard LISA functionality such as the Moran scatter plot, local Moran, and Gi statistics.

Spatial modeling is implemented in the spatial dynamics and spatial econometrics modules. The former contains a number of tools to track the change over time of spatial structure, developed with an eye towards applications in studies of regional economic convergence. These include spatial Markov analysis as well as spatial θ and spatial τ

TABLE 1

PySAL Functionality By Component

Component	Capabilities
	Data Analytic Functions
File Input-Output	Read and write common spatial data formats
Map Calculations	Map algebra
Computational Geometry	Geometric summaries of spatial patterns
Spatial Weights	Efficient construction/manipulation of spatial weights matrices
Rate Smoothing	Spatial and nonspatial smoothing of rate data
	ESDA and Modeling Functions
Spatial Autocorrelation	Local and global spatial autocorrelation
Space-Time Correlation	Spatial and temporal correlation measures
Markov and Mobility	Spatial Markov and distributional dynamics
Regionalization	Spatially constrained clustering
Spatial Regression	Classic spatial econometric methods
Spatial Panel Regression	Spatial methods for panel data

measures of convergence. The spatial econometrics module contains a collection of diagnostics for spatial effects, specification tests, and estimation methods as well as simulation tools to embed various forms of spatial dependence in artificial data sets. Detailed illustrations of selected functionality are provided in the next section.

3. EMPIRICAL ILLUSTRATIONS

We present a selection of applications of modules within PySAL and illustrate how they can be exposed through various delivery mechanisms, including alternative GUIs. The examples are intended to be suggestive, not exhaustive, and highlight how particular core modules, jointly developed in PySAL, have been integrated into the two ongoing projects, GeoDA/PySpace and STARS.

3.1 Computational Geometry and Spatial Weights

Figure 2 contains the nearest neighbor graph for a point distribution. Here we have implemented efficient nearest neighbor algorithms for general *k*-nearest neighbor determination in large point sets. Combining these methods together with classes in the spatial weights module, we can generate alternative spatial weights matrices based on nearest neighbor relations for both point data sets as well as areal/polygon data sets where representative points are used in developing the topological relationships.

The spatial weights module also supports additional graph-based definitions of weights using point data. These include Gabriel, sphere of influence, and relative neighbor criteria. For polygon-based shape files, the module also contains efficient classes for derivation of queen- and rook-based contiguity matrices on the fly. These classes free the user from the tedious and error-prone task of constructing weight

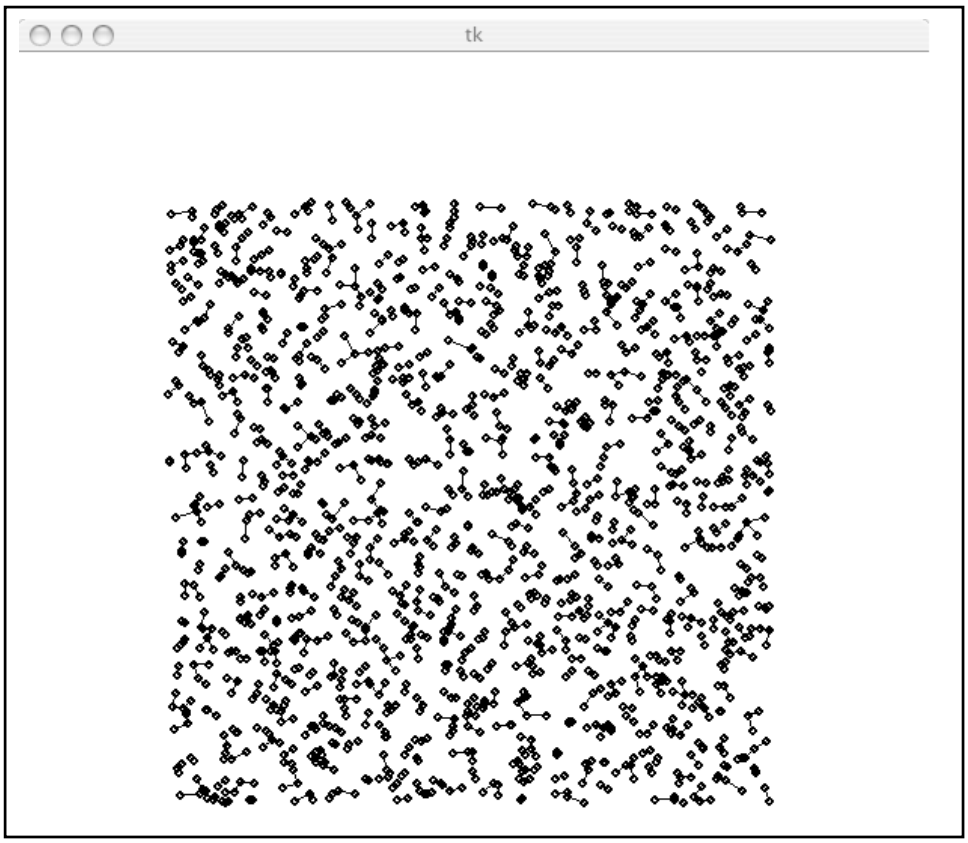

FIGURE 2. Nearest Neighbor Graphs

matrices by hand. For all of these spatial weights, the associated classes implement manipulation and summarization methods that are commonly needed in spatial analysis, including measures of sparseness, connectivity, and various eigenvalue-based metrics, among many others. The weights module also supports the reading and writing of common spatial weights matrices formats including GAL, GWT, and full matrices.

3.2 Spatial Dynamics

With the increasing availability of spatial longitudinal data sets, there is a growing demand for exploratory methods that integrate both the spatial and temporal dimensions of the data. The spatial dynamics component of PySAL implements a number of new exploratory space-time data analysis measures.

These new measures approach the issue of space-time analysis in two different ways. The first introduces a spatial dimension into what are classic measures of mobility or dynamics. For example, in the study of regional income distributions popular approaches

to measure economic mobility include rank concordance statistics, rank correlation statistics, and Markov models. All of these generate indicators that summarize the amount of movement within the variate distribution over time. However, like many classic statistics they are silent about the role of geography in the dynamics. In PySAL, the spatial dynamic module implements spatialized versions of these three mobility indicators, including a spatial-τ statistic, spatial-Θ (Rey 2004), and spatial-Markov model (Rey 2001). Each of these methods speaks to the role of spatial clustering and context in the evolution of the distribution of interest. That is, they investigate the extent to which the dynamics of the process are spatially dependent.

The second approach to spatial dynamics in PySAL starts with exploratory spatial data analysis methods and extends these measures to integrate the time dimension. One example of this is the *spatial time path*, two examples of which are shown in Figure 3. The time path can be viewed as a dynamic extension of a LISA statistic (Anselin, 1995) in that the *Y*-axis of the graph corresponds to the value of the spatial lag of the variable while the *X*-axis is the original value for a particular spatial unit. In contrast to a Moran scatter plot (upper-right panel of Figure 3), which displays the (y_i, Wy_i) values for all locations at one point in time, the time path focuses on a single location *i* but displays the ($y_{i,t}$, $Wy_{i,t}$) over all time periods.

These measures look at spatial dynamics from a slightly different perspective from the first in that they focus on the spatial dimension and explore its evolution over time. They can be used for comparative analyses such as in Figure 3 where the paths for per-capita incomes for California (bottom left) and Florida (bottom right) are contrasted. The spatial dynamics for Florida are more erratic than is the case for California. At the same time, a casual glance suggests the relationships are similar in that there is positive correlation between each state's income and that of its regional neighbors over time. However, by exploiting the interactive capabilities of the software, temporal animation reveals that the directionality of the dynamics is different in the two cases with Florida and its neighbors moving upward towards the center of the distribution, while California and its neighbors are moving downwards towards the mean.

In addition to the time paths, the spatial dynamics module includes a number of other new measures that are extensions of ESDA methods to incorporate time. These include a bi-variate LISA, which allows for consideration of space-time lags between two different variables as well as space-time principal components, which is a multivariate extension of the bi-variate LISA.

As with most of the modules in PySAL, the spatial dynamics classes can be combined with other modules to accomplish a complex analytical task. An example of this is seen in Figure 4 where a new type of spatial weights matrix is obtained through a consideration of the time series covariance of per capita incomes for each pair of states

FIGURE 3. Spatial Time Paths

over a 72-year period. The join structure for the original simple contiguity matrix is presented as a simple network, yet each join is now colored to signify whether that pair of states displays strong (blue) or weak (red) temporal co-movement. A hybrid contiguity matrix could be defined by only using the strong links. Also included on the figure is the spider graph for Colorado. These blue links show states with which Colorado has its strongest temporal correlation. This suggests a second type of hybrid contiguity matrix based on the intersection of the simple contiguity and the spider contiguity joins.

3.3 Smoothing of Rates

An important aspect of exploratory spatial analysis of rates or proportions is to correct for the inherent variance instability of the rates. Ignoring this aspect may lead to spurious indications of outliers and clusters due to higher variance when the population at risk is small. Several techniques for smoothing rates have been incorporated into PySAL modules. They consist of a porting of the rate smoothing functionality in GeoDa (implemented in C++) to Python (for a more extensive discussion, see also Anselin, Kim, and Syabri 2004; Anselin, Syabri, and Kho 2006).

Functionality of the rate smoothing modules can be classified into three major categories: data input, rate computation, and smoothing. The first includes the capacity to read in data on counts of events (e.g., number of diseased persons) and population at risk from various file formats, including SEER, either as aggregates or by age group. Rate computation takes the data and computes rates for individual spatial units (e.g., counties) as well as for aggregates (e.g., all the counties in a state) and implements both direct and indirect age standardization. Rate smoothing implements a number of common methods, including Empirical Bayes and spatial rate smoothing. The latter is an interesting instance where the modular nature of PySAL is exploited since it requires functionality from the spatial weights module to implement the spatial averaging of rates.

Figure 5 illustrates an application of spatial rate smoothing to age-standardized prostate cancer rates in counties covered by the Appalachian Cancer Network. This application utilizes the core rate manipulation and smoothing functionality of the library coupled to a graphical front end implemented in wxPython. This is an example of delivery of the functionality where the user is completely shielded from the Python programming environment, even though it is readily accessible if desired.

The wxPython graphical user interface is cross-platform and provides a local look and feel on each platform. It consists of a Python wrapper around the well known C++ wxWidgets library. In Figure 5, the particular look and feel is that of the Mac OS X operating system. Using simple menus, the user can select the data, spatial weights (for spatial rate smoothing), and smoothing technique and the result is presented on a map, as shown in the figure. Functionality such as this can also be readily delivered in compiled form, in which case the user no longer would have access to the original source code.

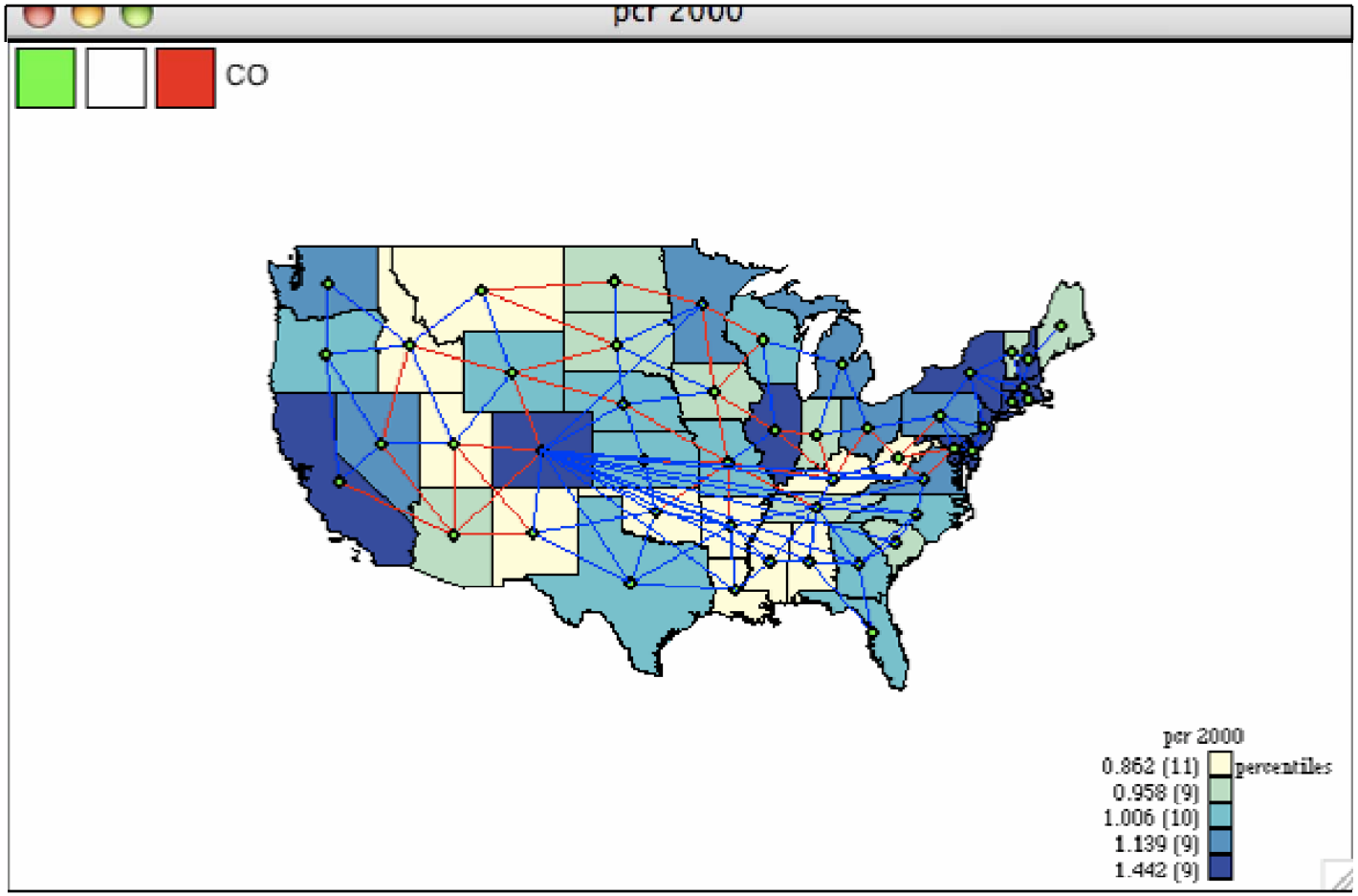

FIGURE 4. Spider and Temporal Contiguity Graphs

The same smoothing modules can also be used in conjunction with a different graphical user interface. For example, rate smoothing is included in STARS, which uses the Tkinter Python GUI. In addition, using the command line in with the Python interpreter, specific smoothing functions can be used individually in an interactive computing environment.

3.4 Regionalization

The regionalization and clustering module of PySAL implements a number of new and existing methods that can be used to define groupings of fundamental units according to a variety of constraints. These methods include contiguity constrained clustering, Automatic Zoning Procedure (AZP), and the max-p region algorithm (Duque, Anselin, and Rey 2007). Figure 6 demonstrates the application of the AZP method to U.S. income dynamics.

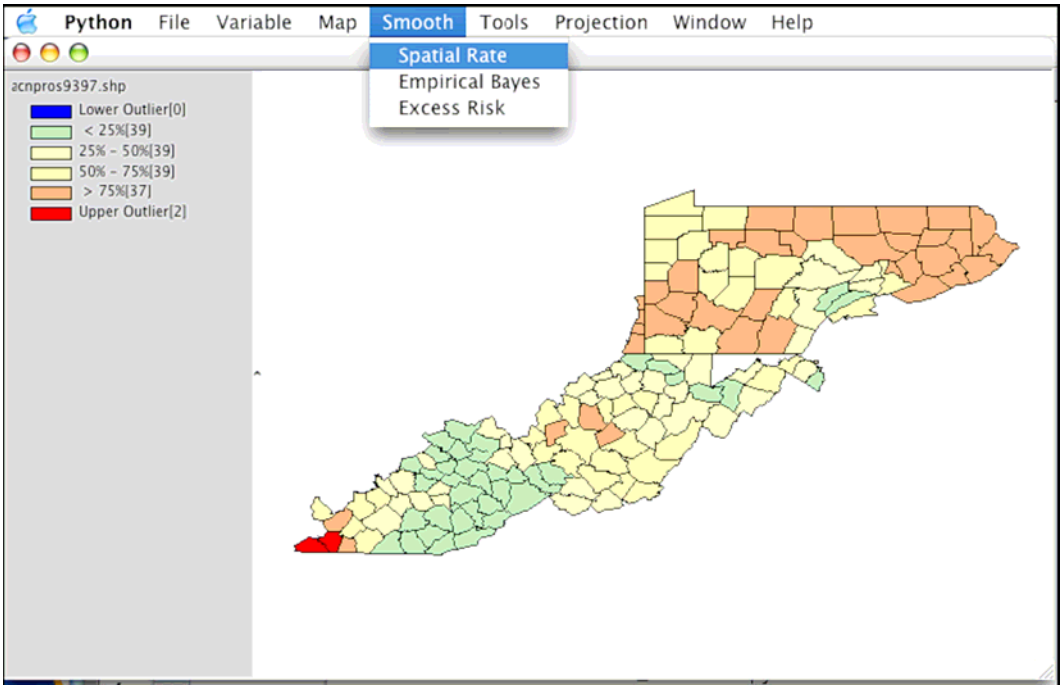

FIGURE 5. Spatial Smoothing of ACN County Prostate Rates

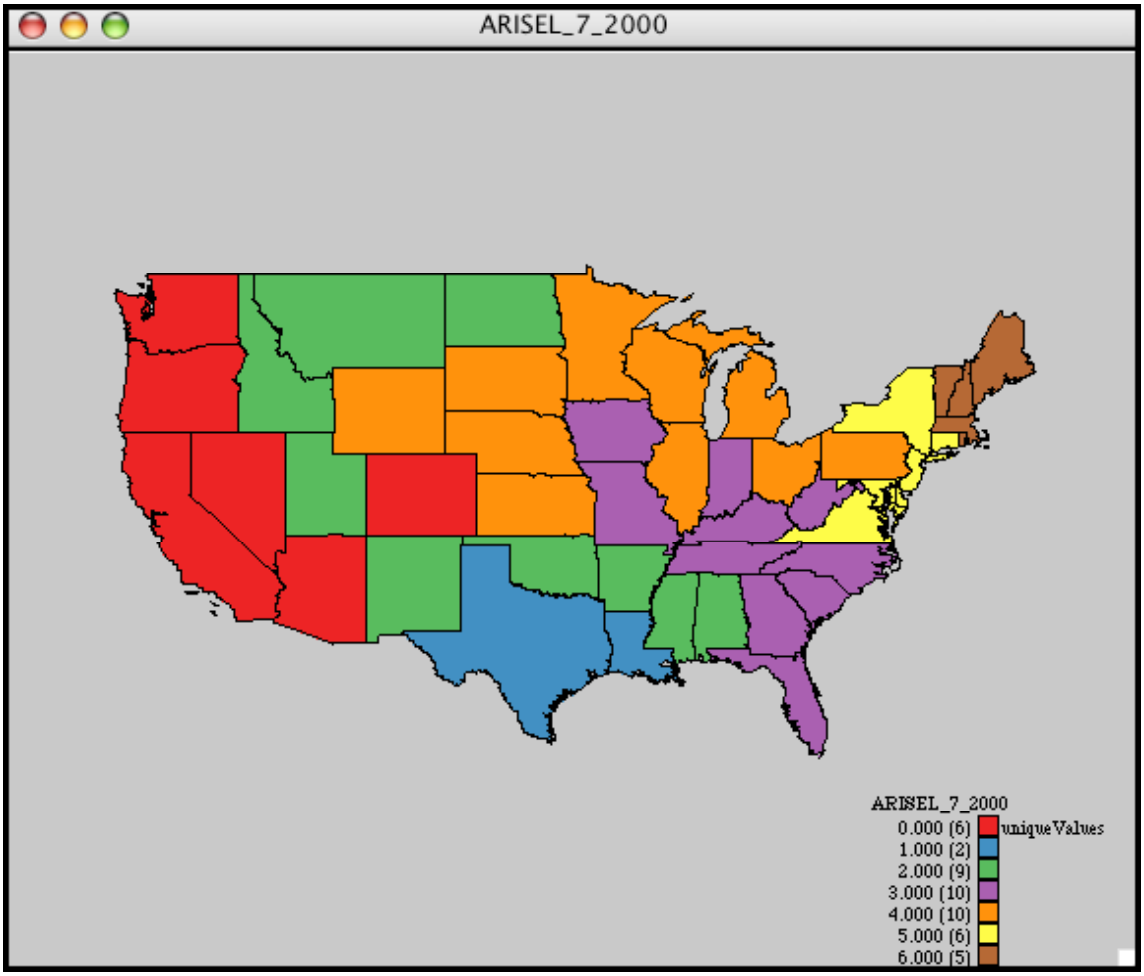

FIGURE 6. Regionalization of State Incomes using AZP

The regionalization module can also be used together with other modules in PySAL to develop new approaches to spatial analytical problems. One example is the integration of the spatially constrained clustering algorithms together with the spatial smoothing module to develop new approaches towards spatial rate estimation (Rey et al. 2007). This work explored alternative ways in which the variance instability problem (see Section 3) could be addressed by defining the neighborhood smoothing regions using the constrained clustering algorithms.

3.5 Spatial Econometrics

The spatial econometric modules in PySAL are primarily intended to provide support for two types of activities: (1) to allow rapid prototyping of newly suggested techniques, and (2) to put together customized combinations of tests and estimation methods. The development efforts are focused on general method of moments estimators, semi-parametric approaches, spatial panel data models, and specifications with discrete dependent variables. In this sense, these modules complement the spatial econometric functionality of GeoDa, which is aimed at providing a user-friendly environment for more established spatial econometric techniques such as Maximum Likelihood estimation.

For example, PySAL implements code to estimate regression models containing a spatially lagged dependent variable (a spatial lag model) by means of the spatial two-stage least squares method (Anselin 1988; Kelejian and Prucha 1998). In addition to the traditional estimates of standard errors and a heteroskedastic robust form (White 1980; Anselin 1988), this also implements the recently suggested heteroskedastic and spatial autocorrelation robust form, or HAC estimator (Kelejian and Prucha 2007). The latter takes a non-parametric approach to allow for remaining spatial error autocorrelation of unspecified form using a kernel estimation method.

The PySAL code for the HAC estimator was recently applied in Anselin and Lozano-Gracia (2007) to estimate a spatial hedonic model with over 100,000 observations, using a spatial lag model that included other endogenous variables as well. In addition to allowing for remaining spatial error autocorrelation in a spatial lag model, the spatial two-stage least squares approach in PySAL is also not constrained to intrinsically symmetric spatial weights, as is the case for the ML estimators in GeoDa.

Figures 7 through 9 illustrate an application of the spatial econometric module to a replication of the analysis of U.S. county homicides in Baller et al. (2001). The implementation uses the command line only, taking the model specification information from a separate module that contains all the information on the data set, variables, and spatial weights. For example, in Figure 7 the contents of such a model are shown, including a dictionary for the model variables and for the data (respectively, spec and data) as well as two lists of dictionaries with spatial weights needed for the spatial lag (mweights) and for the kernel estimation (kweights). Each of these dictionaries contains several attributes of the data and weights needed by the modules that implement data input and spatial weights construction. The module can be edited by means of a text editor and *imported* into the current session to be used by the spatial regression module. In the current example, an asymmetric spatial weights matrix for five nearest neighbors is used to construct the spatial lag.

The central element in the spatial econometric functionality is the spmodel class, similar in concept to the object-oriented design of model classes in the R language. Figure 8 illustrates the construction of an object model of the spmodel class in the spreg

module. Some of the arguments that are passed to the constructor include a data object (spreg.db), a model specification object (spreg.spec) as well as weights objects and some model options, e.g., the specification of a lag spatial model, using gmm as the estimation method and hac as the option for the variance-covariance estimator. Once the model object is created, its attributes can be accessed using the familiar dot notation. For example, in Figure 8, the name of the input data set, number of observations, number of variables, the model specification, and the spatial weights are illustrated. Note how the spatial weights are themselves instances of the weights class constructed in the spatial weights modules.

```
# spec: model specification: y dep var, X exogenous, yend endogenous
#       H instruments
spec = {}
spec['y'] = 'HR90'
spec['X'] = ['RD90','PS90','MA90','DV90','UE90','SOUTH']

# data: data source
data = {}
data['fname']='natn.csv'
data['idvar']='FIPSNO'
data['dType']='listvars'
data['formatheader']= 0
data['numonly']=0

# mweights: spatial weights for use in model lag 0 error 1
# if different
mweights = []
mw = {}
mw['wtfile']='natk5.gwt'
mw['wtType']='binary'
mw['headline']=0
mw['sep']=','
mw['rowstand']=1
mw['power']=1
mw['dmax']=0
mweights.append(mw)

# kweights: kernel weights
# if none specified, no kernel
kweights = []
kw = {}
kw['wtfile']='natk20.gwt'
kw['wtType']='epanech'
kw['headline']=0
kw['sep']=','
kw['rowstand']=0
kw['power']=1
kw['dmax']=0
kweights.append(kw)
```

FIGURE 7. Spatial Regression Model Specification

```
>>> model = spreg.spmodel(spreg.db,spreg.spec,mweights=spreg.mw1,
... kweights=spreg.kw1,space='lag',method='gmm',option='hac')
>>> model.fname
'natn.csv'
>>> model.nobs
3085
>>> model.k
8
>>> model.spec
{'y': 'HR90', 'X': ['RD90', 'PS90', 'MA90', 'DV90', 'UE90', 'SOUTH']}
>>> model.mw1
<weights.spweight instance at 0x1641670>
>>> model.kw1
<weights.spweight instance at 0x14ce8f0>
```

FIGURE 8. Spatial Regression Model Object Attributes

The estimation results are obtained by invoking one of the methods in the spmodel class. In Figure 9 this is illustrated for the twosls method. It is invoked on the command line by means of the dot notation, applied to the model instance of the spmodel class. This yields the output of the estimates, standard errors and measures of fit, in the familiar GeoDa format. Three tables are listed, for the traditional standard errors, the heteroskedastic robust form and the HAC. The latter is implemented using an Epanechnikov kernel function with an adaptive bandwidth for the 20 nearest neighbors. The standard errors increase slightly relative to the classic estimate.

In the example, one diagnostic is included by default (it can also be invoked separately as a method of the spmodel class), the Anselin and Kelejian (1997) generalized Moran's I test for residuals in a spatial lag model. As shown in Figure 9, the null hypothesis is strongly rejected, providing a solid motivation for the use of the HAC standard errors.

3.6 Spatial Analytical Web Services

The core libraries are designed in such a way as to enable a variety of front ends through which users can interface with the functionality in PySAL. In previous examples, we have illustrated the use of two different GUIs and the shell/command line. A third form of user interface is the Web browser, where the PySAL functionality is delivered in the form of a spatial analytical Web service.

A straightforward way to accomplish this is to include components of the library as common gateway interface (cgi) scripts on a Web server. The user interacts with this through a Web page, which sends a form to the server that includes all the parameters needed to carry out the analysis. The results are then delivered as a new Web page. To the user, the experience is similar to an interactive GUI on the desktop.

```
>>> model.twosls()
Data:  natn.csv  N:  3085  df:  3077
Dependent Variable:  HR90
Instruments:  W_RD90 W_PS90 W_MA90 W_DV90 W_UE90 W_SOUTH
Spatial Weights:  natk5.gwt  Type:  binary
Kernel Weights:  natk20.gwt  Type:  epanech
R2 (var): 0.44097474     R2 (corr): 0.44015616
2SLS Results
CONSTANT          5.27970466   1.05421367   5.00819218 5.8040651e-07
RD90              3.70854698   0.14648314  25.31722759 1.2849701e-128
PS90              1.37504128   0.10015256  13.72946666  1.11636e-41
MA90             -0.08427221   0.02755448  -3.05838455 0.0022444905
DV90              0.54414517   0.05500481   9.89268364 9.7467151e-23
UE90             -0.28049426   0.04118603  -6.81042228 1.1655819e-11
SOUTH             1.31132254   0.28790335   4.55473172 5.4490157e-06
W_HR90            0.18870532   0.03971433   4.75156802 2.1109551e-06
Data:  natn.csv  N:  3085  df:  3077
Dependent Variable:  HR90
Instruments:  W_RD90 W_PS90 W_MA90 W_DV90 W_UE90 W_SOUTH
Spatial Weights:  natk5.gwt  Type:  binary
Kernel Weights:  natk20.gwt  Type:  epanech
R2 (var): 0.44097474     R2 (corr): 0.44015616
2SLS Results, White Variance
CONSTANT          5.27970466   1.04716434   5.04190649 4.8759527e-07
RD90              3.70854698   0.22598487  16.41059827 4.3949434e-58
PS90              1.37504128   0.16795804   8.18681414 3.8851091e-16
MA90             -0.08427221   0.02794873  -3.01524329 0.0025887092
DV90              0.54414517   0.08031388   6.77523167 1.4823565e-11
UE90             -0.28049426   0.05110650  -5.48842574 4.384525e-08
SOUTH             1.31132254   0.29345646   4.46854213 8.1594953e-06
W_HR90            0.18870532   0.04286644   4.40216873 1.1082167e-05
Data:  natn.csv  N:  3085  df:  3077
Dependent Variable:  HR90
Instruments:  W_RD90 W_PS90 W_MA90 W_DV90 W_UE90 W_SOUTH
Spatial Weights:  natk5.gwt  Type:  binary
Kernel Weights:  natk20.gwt  Type:  epanech
R2 (var): 0.44097474     R2 (corr): 0.44015616
2SLS Results, HAC Variance with kernel epanech
CONSTANT          5.27970466   1.08618007   4.86080053 1.2276991e-06
RD90              3.70854698   0.24481531  15.14834572 4.7483522e-50
PS90              1.37504128   0.17801139   7.72445672 1.5089075e-14
MA90             -0.08427221   0.02796515  -3.01347276  0.002603817
DV90              0.54414517   0.08076990   6.73697932 1.9225394e-11
UE90             -0.28049426   0.05241424  -5.35148941 9.363273e-08
SOUTH             1.31132254   0.31097070   4.21686840 2.5485898e-05
W_HR90            0.18870532   0.04587635   4.11334626 4.0018136e-05
Anselin-Kelejian Test for Residual Spatial Autocorrelation
Moran's I: -0.0552 LM: 11.02 p: 0.000903468
>>>
```

FIGURE 9. Spatial Two Stage Least Squares with HAC Error Variance

As an illustration, Figure 10 shows the results from the regionalization and clustering component of PySAL applied to define regional industrial clusters in the state of California (Rey et al. 2005). In this work, network and graph theoretical constructs were used in conjunction with spatially constrained clustering to identify groups of functionally interdependent industries within a regional economy. A Web-based front end allows for the exploration of different dimensions of a cluster. For example, clicking on one of the nodes (industry) in the cluster graph (left panel) generates a map of the spatial distribution of that industry within the state (center panel) as well as a view of its supply chain in San Diego County (right panel). Other Web-based views (not shown here) allow for the exploration of the location of individual firms as well as pattern-based text searches of firm profiles and capabilities.

A more elaborate form of a Web interface can be developed by exploiting the HTTP and SOAP (simple object access protocol) Web service functionality built into the Python language and extension modules. Figure 11 illustrates the architecture of a prototype spatial analytical Web service to construct spatial weights from ESRI shape files, using standards supported by the Open GIS Consortium (OGC). This combines three components, that each can operate on a different physical server, allowing for a distributed system.

The front end is the Web interface (shown in Figure 12) through which the user interacts with the system by means of a set of Python cgi scripts that manage information flows between the front end and the two other components, the Data Server and the Analysis Server. Through the interface, a Web feature service (Data Server, using the Mapserver cgi) is queried for a list of available data sources, which then become available in a drop down list on the Web interface, transparent to the user. This could easily be generalized to query a collection of Web feature services for available data sets. Alternatively, users can specify the URL for the data source explicitly, which can be anywhere on the Internet, including other compliant Web feature services. In addition, the type of weights matrix (rook or queen) can be selected.

The information on the data source and weights type is then passed to the Analysis Server, using the SOAP protocol. This back-end operation consists of a set of Python scripts to handle the interaction between the different services and to interface with the PySAL library for the actual computation of the weights. The data are extracted from the data server, the weights are computed and stored on the analytical server, and the URL of this location is passed back to the user interface. The weights information can also be transferred in other ways, using a standard XML format, as illustrated in Figure 13.

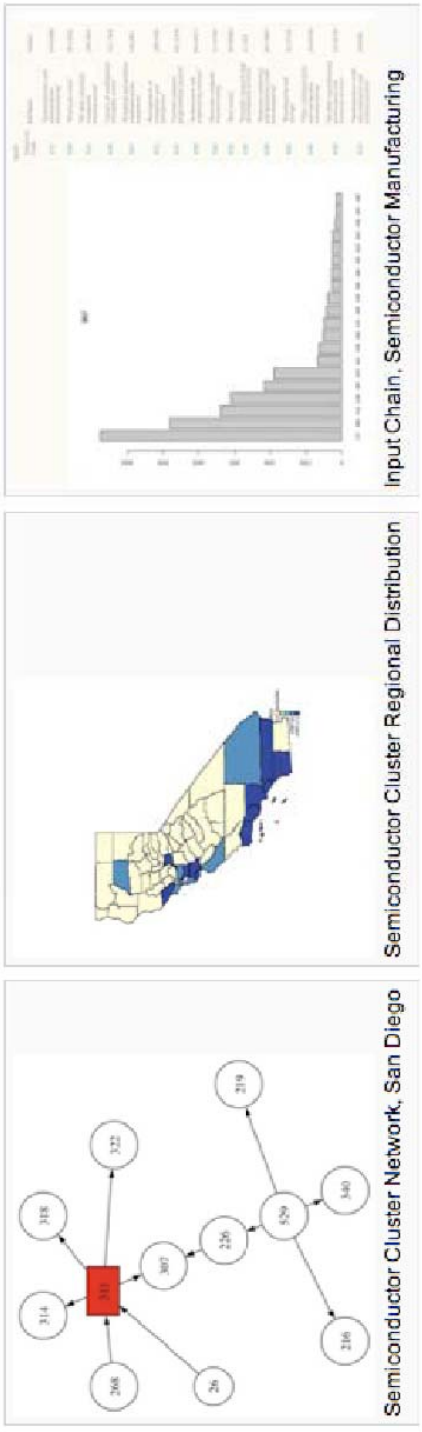

FIGURE 10. Regional Industrial Clustering

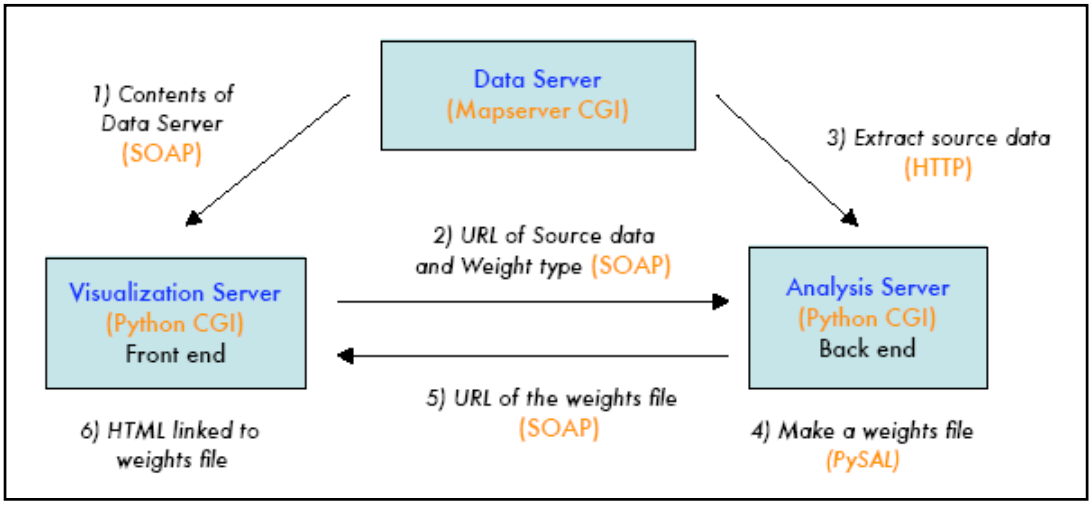

FIGURE 11. Architecture of Spatial Weights Web Service

FIGURE 12. Weights Web Service User Interface

```
- <ExecuteResponse version="0.4.0" xsi:schemaLocation="http://www.opengeospatial.net/wps
  http://www.bnhelp.cz/schema/wps/0.4.0/wpsExecute.xsd">
    <ows:Identifier> makeWeight </ows:Identifier>
  - <Status>
      <ProcessSucceeded/>
    </Status>
  - <ProcessOutputs>
    - <Output>
        <ows:Identifier> outputWeight </ows:Identifier>
        <ows:Title> Resulting weight file </ows:Title>
        <!--Element Abstract not set-->
      - <ComplexValue format="text/xml">
        - <Value>
          - <SAL:weightfile inputfile="inputfile" numRec="88" type="GAL" wtype="Rook Contiguity">
            - <SAL:record id="1" numNeighbors="4">
                <SAL:neighbors> 2,6,7,11 </SAL:neighbors>
              </SAL:record>
            - <SAL:record id="2" numNeighbors="3">
                <SAL:neighbors> 1,4,11 </SAL:neighbors>
              </SAL:record>
            - <SAL:record id="3" numNeighbors="5">
                <SAL:neighbors> 5,10,15,87,88 </SAL:neighbors>
              </SAL:record>
            - <SAL:record id="4" numNeighbors="3">
                <SAL:neighbors> 2,11,13 </SAL:neighbors>
```

FIGURE 13. Weights in XML Format

4. CONCLUSION

The main efforts thus far have been on the development of the core analytical functionality and coupling these modules with the graphical toolkits used in the two source projects: Tkinter for STARS and wxPython for OpenGeoDa/PySpace. Future work will explore use of PySAL with alternative front-ends including jython (Pedroni and Rappin 2002), RPy (Moriera and Warnes 2004), and ArcGIS. Additionally, we are investigating alternative shell/command line environments beyond the basic Python interpretor, such as iPython (Pérez 2006). At the same time we will regularly be integrating new developments in spatial analysis into the computational classes within PySAL.

Our plans are to continue refining the core components of the library and the associated application programming interface (API). We are also evaluating alternative licensing schemes with an eye towards leveraging the strengths of the open source and spatial analysis communities. We envisage a formal release of PySAL in the near future.

REFERENCES

Andrienko, G. and N. Andrienko, 2005. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer: New York.

Andrienko, G., N. Andrienko, and H. Voss, 2003. "GIS for Everyone: The CommonGIS Project and Beyond," in M. Peterson (ed.), *Maps and the Internet*. Elsevier Science: Amsterdam.

Anselin, L., 1988. *Spatial Econometrics: Methods and Models*. Kluwer Academic Publishers: Dordrecht, The Netherlands.

_____, 1995. "Local Indicators of Spatial Association (LISA)," *Geographical Analysis* 27, 93–116.

_____, 2005. "Spatial Statistical Modeling in a GIS Environment," in D.J. Maguire, M. Batty, and M.F. Goodchild (eds.), *GIS, Spatial Analysis and Modeling*. ESRI Press: Redlands, CA.

Anselin, L. and H.H. Kelejian, 1997. "Testing for Spatial Error Autocorrelation in the Presence of Endogenous Regressors," *International Regional Science Review* 20, 153–182.

Anselin, L., Y.W. Kim, and I. Syabri, 2004. "Web-Based Analytical Tools for the Exploration of Spatial Data," *Journal of Geographical Systems* 6, 197–218.

Anselin, L. and N. Lozano-Gracia, 2007. "Errors in Variables and Spatial Effects in Hedonic House Price Models of Ambient Air Quality," *Empirical Economics*, forthcoming.

Anselin, L., I. Syabri, and Y. Kho, 2006. "GeoDa: An Introduction to Spatial Data Analysis," *Geographical Analysis* 48, 5–22.

Baller, R., L. Anselin, S. Messner, G. Deane, and D. Hawkins, 2001. "Structural Covariates of U.S. County Homicide Rates: Incorporating Spatial Effects," *Criminology* 39(3), 561–590.

Bivand, R.S. and A. Gebhardt, 2000. "Implementing Functions for Spatial Statistical Analysis Using the R Language," *Journal of Geographical Systems* 2, 307–317.

Butler, H. and S. Gillies, 2005. "Open Source Python GIS Hacks," in *Open Source Geospatial '05*, Minneapolis.

Coles, J., J.O. Wagner, and F. Koormann, 2004. User's Manual for Thuban 1.0. Technical report, Intevation GmbH.

Duque, J.C., L. Anselin, and S.J. Rey, 2007. "The Max-P Region Problem," Regional Analysis Laboratory Working Paper 20070301.

Fischer, M.M. and A. Getis, 1997. *Recent Developments in Spatial Analysis*. Springer-Verlag: Berlin.

Gillies, S. and K. Lautaportti, 2006. Python Cartographic Library (PCL). trac.gispython.org/projects/PCL/wiki.

Goodchild, M.F., L. Anselin, R.P. Appelbaum, and B.H. Harthorn, 2000. Toward Spatially Integrated Social Science," *International Regional Science Review* 23, 139–159.

Haining, R., 1989. "Geography and Spatial Statistics: Current Positions, Future Developments," in B. Macmillan (ed.), *Remodelling Geography*. Basil Blackwell: Oxford.

Kelejian, H H. and I. Prucha, 1998. "A Generalized Spatial Two Stage Least Squares Procedures for Estimating a Spatial Autoregressive Model with Autoregressive Disturbances," *Journal of Real Estate Finance and Economics* 17, 99–121.

_____, 2007. "HAC Estimation in a Spatial Framework," *Journal of Econometrics*, forthcoming.

Langtangen, H.P., 2006. *Python Scripting for Computational Science*. Springer-Verlag: Berlin.

LeSage, J.P., 1999. Spatial Econometrics Using MATLAB, available at spatial-econometrics.com

Lewis, B., 2007. Open Source GIS. Technical Report, OpenSourceGIS.org.

Moriera, W. and G. Warnes, 2004. "'rpy,' A Robust Python Interface to the R Programming Language." ryp.sf.net.

Pedroni, S. and N. Rappin, 2002. *Jython Eessentials*. O'Reilly.

Pérez, F., 2006. *IPython: An Enhanced Interactive Python*. Department of Applied Mathematics, University of Colorado: Boulder.

Rey, S.J., 2001. "Spatial Empirics for Economic Growth and Convergence," *Geographical Analysis* 33, 195–214.

_____, 2004. "Spatial Dependence in the Evolution of Regional Income Distributions," in A. Getis, J. Múr, and H. Zoeller (eds.), *Spatial Econometrics and Spatial Statistics*. Palgrave: New York.

Rey, S.J. and L. Anselin, 2006. "Recent Advances in Software for Spatial Analysis in the Social Sciences," *Geographical Analysis* 38, 1–4.

Rey, S., L. Anselin, J. Duque, and X. Li, 2007. "Max-P Region Based Estimation of Disease Rates," in *Regional Science Association International, Toronto.* Regional Analysis Laboratory Working Paper 20070420.

Rey, S., J. Duque, O. Smirnov, Y. Kim, and P. Stephens, 2005. "Identifying Value-Added Industry Clusters in San Diego County," Technical Report, Regional Analysis Laboratory Technical Report: REGAL 20050616.

Rey, S.J. and M.V. Janikas, 2006. "STARS: Space-Time Analysis of Regional Systems," *Geographical Analysis* 38, 67–86.

Takatsuka, M. and M. Gahegan, 2002. "GeoVista Studio: A Codeless Visual Programming Environment for Geoscientific Data Analysis and Visualization," *Journal of Computers & Geosciences* 28, 1131–1144.

White, H., 1980. "A Heteroskedastic-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity," *Econometrica* 48, 817–838.