GeoDa Center for Geospatial Analysis and Computation
Arizona State University

Working Paper 2008-12

Dynamic Manipulation of Spatial Weights Using Web Services

Sergio J. Rey, Luc Anselin, and Myunghwa Hwang

Dynamic Manipulation of Spatial Weights Using Web Services*

Sergio J. Rey
GeoDa Center
School of Geographical Sciences
Arizona State University

Luc Anselin
GeoDa Center
School of Geographical Sciences
Arizona State University

Myunghwa Hwang
GeoDa Center
School of Geographical Sciences
Arizona State University

Abstract

Spatial analytical tools are mostly provided in a desktop environment, which tends to restrict user access to the tools. This project intends to exploit up-to-date web technologies to extend user accessibility to spatial analytic tools. The first step is to develop web services for widely used spatial analysis such as spatial weights manipulation and provide easy-to-use web-based user interface to the services. Users can create, transform, and convert spatial weights for their data sets on web browsers without installing any specialized software.

1. Introduction

Spatial lag operators, often referred to as weight matrices, appear across many different areas of spatial analysis and take on numerous forms. In general terms, a spatial weight $w_{i,j}$ represents a spatial relation between two locations i and j. That spatial relation is often defined in terms of contiguity or distance, although these do not exhaust the possibilities. The collection of all such weights for a system of n locations forms the matrix W. Often these weights are required in the initial phase of any type of spatial analysis. For example, in spatial statistics the study of spatial autocorrelation is concerned with various approaches towards estimating the structure and strength of the covariance between a random variable measured at pairs of locations i and j. Given a cross-sectional data set with n observations, there are $(n^2 - n)/2$ pairs of observations and associated covariances which creates a degrees of freedom problem. A spatial weights matrix is used to reduce the dimensionality of the parameter space. In practice, the information embedded in the weights matrix is not stored as such, but efficient sparse formats such as linked lists or dictionaries are used instead [6].

In the regionalization literature adjacency matrices are key data structures in many algorithms for spatially constrained clustering [9]. Similarly, shortest path finding algorithms make heavy demands on adjacency matrices [1]. Node-node and node-arc adjacency matrices are fundamental to representation of spatial objects and their connectivity in modern geographic information systems [24]. More broadly, adjacency matrices are found throughout the fields of computer graphics and visualization, pattern recognition, and computational geometry [21].

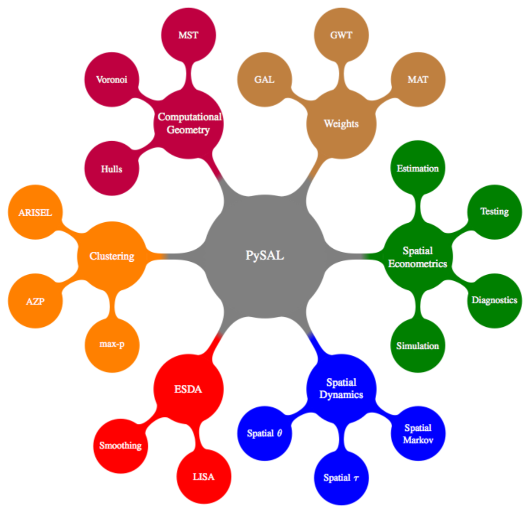
The prominent role of spatial weights across the spectrum of spatial analytical methods creates a situation where many specialized software efforts will tend to result in considerable duplication in dealing with their construction, manipulation, and transformation. At the same time, the growing adoption of the methods of spatial statistics by substantive researchers is constrained by the lack of facilities to deal with spatial weights in most commonly used statistical packages. This provides a major motivation for the development and dissemination of spatial weights functionality in the form of a readily accessible software library.

In this paper we describe a web services approach to address these issues. We present an approach that builds upon earlier efforts in the delivery of spatial analytical capability through a web interface [4] and extend it by providing a web interface to an underlying analytical library PySAL [19]. This represents an open source library for spatial analysis written in the object-oriented language Python. PySAL

*This research was supported in part by a grant from the National Cancer Institute (1R01CA126858) for the projects entitled "Geospatial factors and impacts: measurement and use." The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Cancer Institute or the National Institutes of Health.

is designed to avoid duplication in effort in the development of common core spatial analytical methods, and it covers a wide set of areas of spatial analysis as summarized in Figure 1. PySAL is also designed to contribute to the rapidly evolving area of scientific computing with Python [12], in which spatial analysis is still largely absent.

Figure 1. PySAL



We have designed PySAL in a modular fashion with access to the analytical functionality provided via a range of interfaces. Examples include graphical user interfaces built with toolkits such as wxPython [17] in the spatial econometrics toolkit PySpace [5] and TkInter [10] in the exploratory space-time data analysis package STARS [20]. At the same time, users interested in using PySAL through the command line can call PySAL as a standard python module or, using other shells such as IPython [16] and even from the R [11] interpretor via RPy [14].

Here, we focus on the design and implementation of a web services interface to PySAL with particular emphasis on the spatial weights core of the library. In the remainder of the paper, we first describe the functionality offered by these services. Next we present the architecture of the web service system from both a component and an implementation perspective. We then describe some example illustrations of the system. The paper closes with an outline of ongoing and future work.

2. Functionality

The functionality pertaining to the creation and manipulation of spatial weights is organized into three core categories:

- weights creation
- weights transformation
- weights conversion

Each of these is conceptualized as a web service and together they are organized into a work flow which provides the needed functionality. Each of these categories is now briefly described in turn.

2.1 Weights creation

Weight creation consists of all the operations necessary to build a spatial weights object from the geographic information contained within a boundary file of polygons or a collection of point coordinates. This includes reading this geographic information, as well as processing it to derive the topology of the data objects and converting it into an efficient data structure.

Specifically, given a file source for the geographic information, the spatial weight creation service includes the following operations:

1. read the geographic information (from a URL or from a user's desktop client),
2. identify neighbor relations for each observation according to user-specified neighbor criteria,
3. generate an adjacency matrix,
4. apply a weighting method defined by the user to the adjacency matrix and generate a spatial weights,
5. write the spatial weights to an output file

In its current implementation, the service supports neighborhood definitions based on contiguity, distance, and nearest neighbors. These criteria are used to dynamically derive the neighbor relations from standard commercial format files such as ESRI's shapefiles.

The original implementation of PySAL relied on the shapelib library [23] for the reading of shapefiles. However, in the current version we have replaced shapelib with a pure Python implementation of classes to read shapefiles. This facilitated the developing of a binning algorithm to to derive the topology and construct either rook (shared edge) or queen (shared vertices) contiguity matrices on the fly as part of the file reading stage. The algorithm also flags island polygons (i.e., those without any contiguity to other polygons) and offers several options for modifying the spatial weights matrix for these observations, following the suggestions in [8].

PySAL also supports the construction of spatial weights from shapefiles that contain point data. This can be done using graph based topological criteria such as Gabriel, sphere of influence and relative neighbor criteria. Similarly, a collection of $k-$nearest neighbor algorithms are available for point data. In cases where representative points (centroids) are used to develop the topological relationships between polygons, these same graph and distance based methods are also applicable. These classes free the researcher from the often tedious and error prone tasks involved with constructing spatial weights by hand.

In addition to reading of point and polygon shapefiles to dynamically derive the spatial arrangement of the data, the spatial weights creation service can also directly read common spatial weights formats, including GAL, GWT, and full matrices.

2.2 Weights transformation

Higher up in the processing stack, the transformation service takes the weights from the creation service stage and provides a series of transformations as well as descriptive statistics of the characteristics of the weights. These include measures of sparseness, distribution of contiguity cardinalities, and various eigenvalue-based metrics of the weight matrix structure.

The transformation functionality itself includes the construction of higher orders of contiguity, row standardization, conversion of general to binary weights, powering of weights, inverse of weights, computation of general boundary share based weights, algebraic operations on weights and set based operations (e.g., union, intersection).

2.3 Weights conversion

With the proliferation of GIS and spatial analysis software packages, a large variety of formats for spatial weights have appeared. This creates the need to provide functionality to convert between these different formats and allow weights constructed in one package to be applied in analyses carried out in other packages.

Arguable the most population format for spatial weights are the GAL format for binary contiguity and the GWT format for general weights. These formats were initially proposed in the SpaceStat software [2, 3] and implemented in the widely distibuted GeoDa package [7]. Subsequently, they were adopted by the open source spdep R package as well as the commercial ClusterSeer software.

Other formats include text file formats, such as DAT (MatLab Econometrics Library), TXT (WinBugs) and XML, as well as binary formats, including the original SpaceStat matrix format, SWM (ArcGIS 9.3), DBF (ArcGIS 9.3), MAT (MatLab), WK1 (MatLab Lotus format), and WK (MLwiN).

The weights conversion component of our web services includes functionality to move the information on the spatial adjacency and the value of the weights from these input formats into the internal data structures used by PySAL and to output the weights values into a limited number of commonly used output formats.

3. Design

3.1 Architecture

Web services can be broadly classified into two distinct groups, based on the protocols used to exchange and access information. These categories are referred to as REST (Representational State Transfer) and SOAP (Simple Object Access Protocol) based services. Most web services developed according to the standards established by the Open Geospatial Consortium (OGC) are weakly based on a REST compliant architecture. However, they also support SOAP binding for message exchange [13, 15]. Consequenlty, and with an eye towards future expansion, we have selected a SOAP-based architecture for our spatial weights web services. This facilitates the composition of several distributed services to accomplish new tasks, which meets the objectives of modularity underlying the PySAL library.

Figure 2 depicts a three-tiered architecture of the planned web services. This consists of a server, where the actual web services are hosted, middleware to handle messages and manage user accounts and data, and a client interface. The PySAL library contains the core computing functionality to create, manipulate, transform and convert weights and is contained within the server component. This functionality is provided in the form of web services. Each web service is tied to an XML-based document that describes its interface (the so-called Web Service Description Document or WSDL), and communicates with clients or other web services through SOAP-based messages. WSDL documents include information about the data types required for input and output. By interpreting the WSDL documents, clients can create service request messages that can be understood by the system.

The middleware application mediates between the front-end client interface and the back end service server. This consists of receiving a simplified message encoded in Javascript Object Notation (JSON) from the client, and creating and sending a request SOAP message to the service server. Conversely it also receives a response SOAP message from the server and creates and sends a simplified JSON message to the client.

In addition, in our architecture, the middleware also provides user authentication and data management. The latter functionality is included to circumvent inefficiencies in the transfer of data (such as geographic boundary files and weights files) using SOAP-based XML messages. The middleware manages the data transfer by allowing users to up-

Figure 2. Architecture

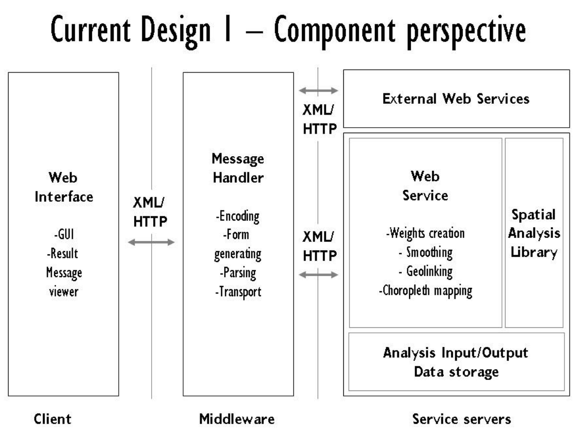
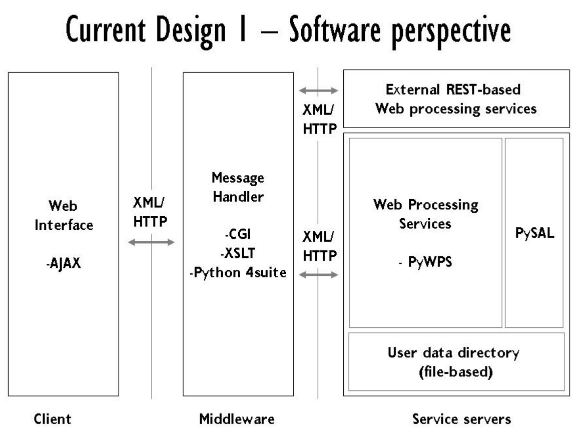


Figure 3.



load information from their desktop into a temporary data store and returns results to the client. The middleware uses JSON messages between itself and the client in order to reduce additional inefficiency that may be caused by double messaging.

This architecture reflects our view that there are generally two types of end users who may interact with this system. In a strict sense, the web services can be seen as an external modular library that can be incorporated into any existing application through the API (Application Programming Interface). In this way, other programs, such as web-aware GIS software, can access the specialized spatial analytical functions included in PySAL through SOAP based XML messaging. However, this limits use to a relative small audience of application developers.

In order to widen the dissemination of advanced spatial analytical techniques, we see our analytical web services as an alternative to a desktop application, by providing a means to access this functionality through a browser, via a desktop-like GUI. In contrast to our earlier implementation through Java-applets that operated on the client desktop [4], we now use a division of labor between the client and the server, with efficient communication between the two.

The web client contains a graphical user interface that dynamically generates input forms for each service operation based on WSDL definitions. Also, upon receiving a response message from the middleware application, the client creates a window to visualize the results message. By using a dynamic form generator we avoid repetitive client-side modifications that are required whenever the web service interfaces change. We also customize each result viewer according to types of outputs generated by each analytical operation.

The architecture outlined in Figure 2 is enabled by the specific software programs illustrated in Figure 3. As mentioned, the computational core is provided by the modular PySAL library, which is contained in the server. The functions in these modules are wrapped into a web service application by using the Python soaplib library (http://trac.optio.webfactional.com/). The main role of this library is to serialize and de-serialize Python objects into SOAP-based XML messages. In addition, it automatically generates WSDL documents. In the middle-tier, we develop several Python CGI programs to handle message manipulation, user account and data management. The Message Handler uses other helper Python libraries, such as simplejson (http://code.google.com/p/simplejson/) for parsing and encoding JSON messages and 4suite (http://4suite.org/index.xhtml) for reformatting XML messages by using Extensible Stylesheet Language Transformations (XSLT).

In addition, to implement the web-based client program, we exploit the Ext JS Javascript library (http://extjs.com/) to a develop desktop-like user interface. We also utilize Asynchronous Javascript and XML (AJAX) to exchange messages between server/middleware and client program.

Apart from the browser client interface, the analytical services can also be accessed from within other programs. For example, the services can be incorporated into a Java application by means of the JAX-WS library or into a Python application by means of the soaplib library. This approach supports a variety of end users with different levels of technical ability but similar analytical requirements.

3.2 Interface

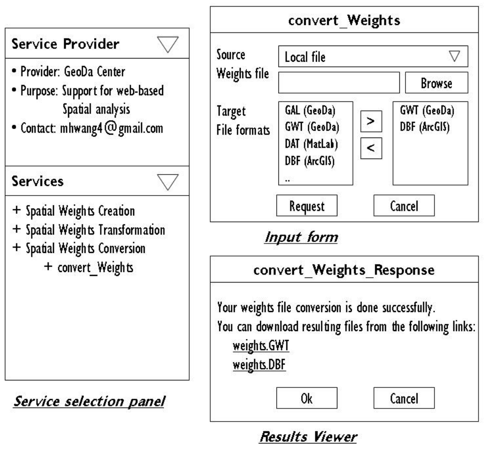
In the current implementation, the web client application is designed as a Javascript component that can be plugged into a standard web page. When first invoked, this application generates a *service selection panel*. As shown in the left panel of Figure 4, this provides information on the types of services available. Specific functionality can be reviewed in a tree layout. Each service node contains a sub-list of supported operations.

When a user selects a node in the service selection panel, the dynamic form generator in the application creates an *input form* for the selected operation, as illustrated in the top right panel of Figure 4. Each such input form includes a Request button. Upon pressing this button, a JSON message is created that takes the user selections and passes this to the message handler in the middleware application. After the operation is carried out, a *results viewer* window is created that allows the user to access the results, as shown in the lower right panel of Figure 4.

In most practical situations, users will have the input file with geographic information stored on their own desktop. Functionality is included in the middleware (as a cgi application) to upload these files to the temporary data store, where authentication ensures that each user has a data space that only they can access. The message handler then creates a SOAP message with information on the location of the user files and sends it to the end-point URL of the service. Upon receiving a response message from the service, the message handler converts it into JSON message and delivers it back to the web client application which, in turn, generates a results viewer.

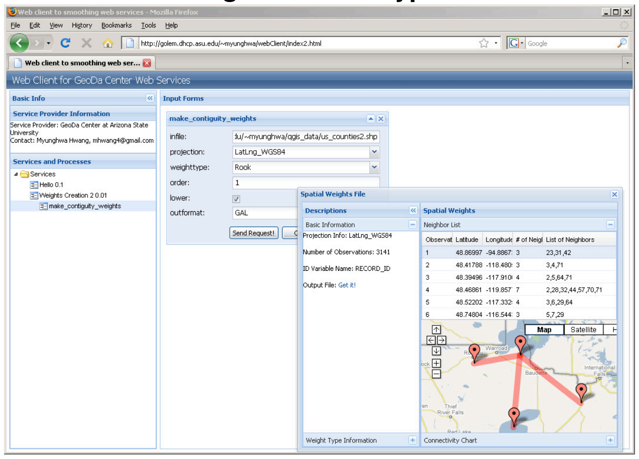
An illustration of the user interface for the web client is illustrated in Figure 5. This shows a more graphically pleasing design for the service selection panel, input form and results viewer. The output highlights the flexibility of the system and its ability to interface with other web services, showing panels listing summary descriptive characteristics of the created weights, with the linkage structure illustrated on a Google Map background. Alternatively, interaction with the web services can be implemented in a rudimentary fashion (using Web Processing Services), providing output in the form of lists, as illustrated in Figure 6.

Figure 4.



Your weights file conversion is done successfully.
You can download resulting files from the following links:

Figure 5. Prototype



http://golem.dhcp.asu.edu/~myunghwa/webClient/index2.html

Web Client for GeoDa Center Web Services

Service Provider Information
Service Provider: GeoDa Center at Arizona State University
Contact: Myunghwa Hwang, mhwang4@gmail.com

make_contiguity_weights

infile:	du/~myunghwa/qgis_data/us_counties2.shp
projection:	LatLng_WGS84
weighttype:	Rook
order:	1
lower:	✓
outformat:	GAL

Projection Info: LatLng_WGS84
Number of Observations: 3141
ID Variable Name: RECORD_ID
Output File: Get it!

Observat	Latitude	Longitude	# of Neigl	List of Neighbors
1	48.86997	-94.8867:	3	23,31,42
2	48.41788	-118.480:	3	3,4,71
3	48.39496	-117.910	4	2,5,64,71
4	48.46861	-119.857	7	2,28,32,44,57,70,71
5	48.52202	-117.332:	4	3,6,29,64
6	48.74804	-116.544:	3	5,7,29

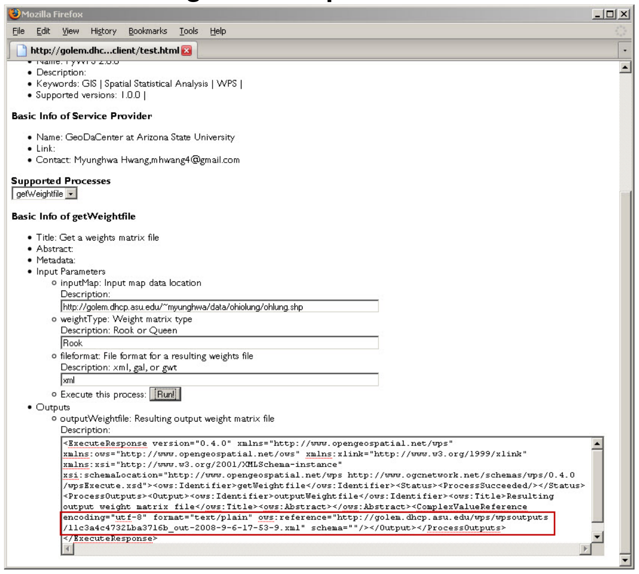
4 Future Research

The web services spatial weights system described here is an initial step in providing a network distributed interface to the PySAL library. Designing it around the weights core of PySAL reflects the central place of spatial weights in much of spatial analysis. In subsequent research we intend to continue to refine the web-based interface to the weights core and eventually begin to extend the same architecture to provide access to other components of the PySAL library from Figure 1.

While we feel the computational components of PySAL lend themselves very nicely to being integrated in the web services system, we see a major challenge in extending the system to support the kind of interactive and dynamic graphics available in desktop top spatial analysis packages such as GeoDa [7] and STARS [20]. Parallel to the work reported here is ongoing research on efficient web-based geovisualization methods [22] that we plan to explore in addressing these challenges.

Finally, while much of the work described here focuses on the anlytical dimensions of the interface, we are keenly aware of the role that open source can play in facilitating learning and advances in scientific communities. Spatial analysis as a web service poses particular challenges in this regard as it is the underlying functionality of PySAL that is made available to the end user and not the source code and implementation [18]. We are currently exploring approaches towards providing source code and documentation browsing, along side the analytical functionality, as components of the next generation of web services.

Figure 6. Output Window


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://golem.dhc...client/test.html

- Description:
- Keywords: GIS | Spatial Statistical Analysis | WPS |
- Supported versions: 1.0.0 |

Basic Info of Service Provider

- Name: GeoDaCenter at Arizona State University
- Link:
- Contact: Myunghwa Hwang,mhwang4@gmail.com

Supported Processes
getWeightfile

Basic Info of getWeightfile

- Title: Get a weights matrix file
- Abstract:
- Metadata:
- Input Parameters
  - inputMap: Input map data location
    Description:
    http://golem.dhcp.asu.edu/~myunghwa/data/ohiolung/ohlung.shp
  - weightType: Weight matrix type
    Description: Rook or Queen
    Rook
  - fileformat: File format for a resulting weights file
    Description: xml, gal, or gwt
    xml
  - Execute this process: Run!
- Outputs
  - outputWeightfile: Resulting output weight matrix file
    Description:

```
<ExecuteResponse version="0.4.0" xmlns="http://www.opengeospatial.net/wps"
xmlns:ows="http://www.opengeospatial.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps http://www.ogcnetwork.net/schemas/wps/0.4.0
/wpsExecute.xsd"><ows:Identifier>getWeightfile</ows:Identifier><Status><ProcessSucceeded/></Status>
<ProcessOutputs><Output><ows:Identifier>outputWeightfile</ows:Identifier><ows:Title>Resulting
output weight matrix file</ows:Title><ows:Abstract></ows:Abstract><ComplexValueReference
encoding="utf-8" format="text/plain" ows:reference="http://golem.dhcp.asu.edu/wps/wpsoutputs
/11c3a4c4732Lba3716b_out-2008-9-6-17-53-9.xml" schema=""/></Output></ProcessOutputs>
</ExecuteResponse>
```

References

[1] R. Ahuja, K. Mehlhorn, J. Orlin, and R. Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM (JACM)*, 37(2):213–223, 1990.

[2] L. Anselin. *SpaceStat, a Software Program for Analysis of Spatial Data*. National Center for Geographic Information and Analysis (NCGIA), University of California, Santa Barbara, CA, 1992.

[3] L. Anselin. Computing environments for spatial data analysis. *Journal of Geographical Systems*, 2(3):201–220, 2000.

[4] L. Anselin, Y.-W. Kim, and I. Syabri. Web-based analytical tools for the exploration of spatial data. *Journal of Geographical Systems*, 6:197–218, 2004.

[5] L. Anselin and J. Le Gallo. Panel data spatial econometrics with PySpace, 2004. Department of Agricultural and Consumer Economics, University of Illinois, Urbana-Champaign, IL.

[6] L. Anselin and O. Smirnov. Efficient algorithms for constructing proper higher order spatial lag operators. *Journal of Regional Science*, 36:67–89, 1996.

[7] L. Anselin, I. Syabri, and Y. Kho. GeoDa: An introduction to spatial data analysis. *Geographical Analysis*, 38:5–22, 2006.

[8] R. Bivand and B. Portnov. Exploring spatial data analysis techniques using R: The case of observations with no neighbors. In L. Anselin, R. J. G. M. Florax, and S. J. Rey, editors, *Advances in Spatial Econometrics: Methodology, Tools and Applications*, pages 121–142. Springer, 2004.

[9] J. Duque, R. Ramos, and J. Surinach. Supervised regionalization methods: A survey. *International Regional Science Review*, 30(3):195, 2007.

[10] J. Grayson. *Python and Tkinter Programming*. Manning Greenwich, CT, 2000.

[11] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

[12] H. Langtangen. *Python Scripting for Computational Science*. Springer, 2006.

[13] R. Lucchi, M. Millot, and C. Elfers. Resource oriented architecture and REST: Assessment of impact and advantages on INSPIRE, 2008. EUR 23397 EN.

[14] W. Moreira and G. Warnes. Rpy (r from python), 2006.

[15] Open Geographic Consortium. *Specification best practices*, 2006.

[16] F. Perez and B. Granger. IPython: A system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29, 2007.

[17] N. Rappin and R. Dunn. *wxPython in action*. Manning, Greenwich, 2006.

[18] S. J. Rey. Show me the code: open source and spatial analysis, 2008. Working paper, GeoDa Center for Geospatial Analysis and Computation. Arizona State University.

[19] S. J. Rey and L. Anselin. PySAL: A Python library for spatial analytical methods. *The Review of Regional Studies*, 37:5–27, 2007.

[20] S. J. Rey and M. V. Janikas. STARS: Space-time analysis of regional systems. *Geographical Analysis*, 38:67–86, 2006.

[21] H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, San Francisco, 2006.

[22] C. Schmidt and B. Dev. A scalable tile map service for distributing dynamic choropleth maps, 2008. Working paper, GeoDa Center for Geospatial Analysis and Computation. Arizona State University.

[23] F. Warmerdam. Shapefile c library v1. 2. *Web site: http://shapelib. maptools. org/(last visit February 2006)*, 2006.

[24] M. Worboys and M. Duckham. *GIS, A Computing Perspective, Second Edition*. CRC Press, Boca Raton, 2004.