ISPRS Int. J. Geo-Inf. **2015**, *4*, 815-836; doi:10.3390/ijgi4020815

OPEN ACCESS

ISPRS International Journal of
Geo-Information
ISSN 2220-9964
www.mdpi.com/journal/ijgi/

Article

Open Geospatial Analytics with PySAL

**Sergio J. Rey *, Luc Anselin †, Xun Li †, Robert Pahle †, Jason Laura †, Wenwen Li †
and Julia Koschinsky †**

GeoDa Center for Geospatial Analysis and Computation, School of Geographical Sciences and Urban Planning at Arizona State University; E-Mails: luc.anselin@asu.edu (L.A.); xun.li@asu.edu (X.L.); robert.pahle@asu.edu (R.P.); Jason.laura@asu.edu (J.L.); wenwen@asu.edu (W.L.); Julia.koschinsky@asu.edu (J.K.)

† These authors contributed equally to this work.

* Author to whom correspondence should be addressed; E-Mail: srey@asu.edu;
Tel.: +1-480-965-7533.

Academic Editors: Barend Kobben, Serena Coetzee and Wolfgang Kainz

Received: 6 February 2015 / Accepted: 20 April 2015 / Published: 13 May 2015

Abstract: This article reviews the range of delivery platforms that have been developed for the PySAL open source Python library for spatial analysis. This includes traditional desktop software (with a graphical user interface, command line or embedded in a computational notebook), open spatial analytics middleware, and web, cloud and distributed open geospatial analytics for decision support. A common thread throughout the discussion is the emphasis on openness, interoperability, and provenance management in a scientific workflow. The code base of the PySAL library provides the common computing framework underlying all delivery mechanisms.

Keywords: spatial analysis; spatial econometrics; spatial decision support systems; cyberGIS; open source software; high performance computing

1. Introduction

Recent advances in geospatial technologies have generated an avalanche of new sources and quantities of georeferenced data [1,2]. These represent both enormous potential for science and society, as well as fundamental challenges. From a societal perspective, live data streams and dashboard systems to visualize patterns of these streams are not only characterizing new technological infrastructures in smart cities, but are becoming more common in many urban areas [3]. In this context, an opportunity exists to supplement client-side closed-box desktop solutions for spatial data analysis with more modular and extendable spatial analytic software libraries. These can be flexibly integrated with live data streams, visualization libraries and decision support tools to be part of end-to-end solutions from data to analysis and visualization to generate evidence-based insights in near real-time. Further, as more open government data are released regularly in machine-readable formats there is a need for open analytics, including spatial analytics, to make sense of patterns in these increasingly larger datasets. The lack of license fees enables analysts in public, non-profit and educational institutions that are already utilizing open data to employ more powerful analytics. Finally, by making the implementation of spatial methods transparent, open spatial tools are part of the larger movement toward open science [4].

From a scientific perspective, these new big data sources represent tantalizing possibilities for studying new types of socioeconomic phenomena, and in new ways, which has stimulated the emergence of a computational social science [5–8]. A number of recent, high profile applications such as the White House's Precision Medicine Initiative [9], or Smart Cities systems [10,11] are suggestive of the potential that is latent within big data. Very often, however, these implementations are produced as proof of concept or prototypes, and some of the methodological complexities are not fully taken into account (e.g., the discussion in [12]). Similarly, there have been innumerable "mash-ups" employing big geospatial data that demonstrate the potential of web based analytical systems, as in the case of crime mapping, flu mapping and neighborhood boundaries (e.g., [13,14]; but see also [15], for a critical assessment).

While all of these applications are inspiring and innovative, they also have uncovered fundamental challenges that geospatial analysis faces before it can fully engage with the big data era. These primarily concern the ability of the software to scale beyond the case study of the prototype as well as to support interoperability with other spatial analytical services and software, and scientific replication and reproducibility. The scalability challenge arises because most of the spatial statistical and econometric software that is leveraged in these prototype systems was not originally designed for the big data era or to exploit new types of high performance computing environments [16]. Moreover, while calls for replication and reproducibility have recently moved to the forefront of the open science agenda, these concepts were not on the white boards of the designers of our previous generation spatial analytical software.

In this paper we report on our collective research efforts that have focused on addressing these challenges. We present recent examples of open spatial analytics that leverage the free and open source Python-based Spatial Analysis Library (PySAL) developed at the GeoDa Center for Geospatial Analysis and Computation. Several examples of flexible delivery formats of PySAL are discussed that include desktop programs (stand-alone and plug-in), web-based applications (web services, web-based spatial data management), and a decision support system (the Complex Systems Framework CSF). These provide specific examples of a larger flexible methodological framework to explore and explain spatial area data patterns through new techniques for space-time and spatial econometric analysis. Flexible geospatial analytics refer to: (1) a toolbox of methodological frameworks ranging from geospatial visual analytics to spatial econometric modelling; (2) a software implementation that is modular, open source, and cross-platform; and (3) the delivery of functionality through multiple user interfaces. This toolbox is delivered through traditional free standing desktop software, toolbox extensions to commercial Geographic Information Systems, and integration into web-based applications such as web services and a dashboard system for decision support.

The rest of the paper is organized as follows. In Section 2 we provide an overview of the PySAL library, discussing the motivation for and history of its development. We then present a series of desktop based geospatial analysis tools that have been built using PySAL in Section 3. These include CAST, a stand-alone tool for exploratory space-time data analysis and GeoDaSpace, which focuses on modern spatial econometrics. Also discussed are the use of PySAL with computational notebooks and shells, such as IPython [17], and the inclusion of PySAL with distributions for scientific computing in Python. In Section 4 we consider some of the challenges faced when moving PySAL from the desktop arena into the high performance computing and distributed context. We discuss work on developing architectures for tracking the provenance of spatial analytical work flows, and the design of a REST API intended to support interoperability between PySAL and other software packages. In Section 5 we then present examples of PySAL on grid, cloud and distributed decision support systems. The paper closes with a discussion of future directions for this work.

2. PySAL

2.1. Motivation for PySAL

PySAL is an open source cross-platform library of spatial analysis functions written in Python [18]. It was born from discussions between the eventual project leaders that took place in the early 2000's. At the time Python was just beginning to make inroads into scientific computing, yet geospatial analysis was largely absent from the Python scientific software stack. We saw a need to bring spatial analysis, spatial statistics and spatial econometrics into this community. Concurrently we also had existing software projects that were using Python, STARS (Rey) and PySpace (Anselin) that we felt could benefit from sharing of common algorithms and data structures so as to reduce duplication and allow for pooled resources in optimizing the new and improved code base.

From conception, PySAL has been viewed as library driven by our goals to leverage code modularization and support flexible reuse and combinations of the components in the library. As we discuss more fully in the following sections, this design has allowed PySAL to be delivered in a wide variety of platforms and for a range of different use cases. Thus we have invested significant effort in designing the library and related infrastructure to support widespread adoption for the development of high-level applications for spatial analysis. From our perspective we see PySAL serving the role as both a target of our own research on advancing geospatial analytics as well as a powerful dissemination mechanism to enable scientific discovery across the social, life and physical sciences. Our own research agendas are tightly coupled with PySAL's development. Members of the core team are very active in the development of new geospatial and space-time analytics that have been widely adopted throughout the scientific community. As we continue with this line of research, new advances are continually added to the library. We are also typical of many other social scientists in that we find PySAL very useful for our own empirical research on different types of substantive problems.

Initially released in July 2010 under the Berkeley Software Distribution License, PySAL has continually been updated on a 6-month release cycle, and at the time of writing the stable version is 1.91. Since its initial release PySAL has been downloaded over 60,000 times. That popularity has also been reflected in PySAL being included as a featured package in the leading software distributions for scientific computing in Python, Enthought and Anaconda.

2.2. PySAL Components

PySAL is designed as an integrated collection of modules with a particular focus on vector based geospatial data. The **weights** module provides for the construction, manipulation and conversion of spatial weights that are central to many types of spatial analysis. Generally speaking the spatial weights formalize the neighbor relations between pairs of spatial observations. As many different criteria can be used to define the neighbor relations, the **weights** module supports three broad classes of weights: (1) contiguity based; (2) distance based; and (3) hybrid. The classes implementing the weights are highly optimized for their memory footprint and speed of computation. The **weights** module also supports the reading and writing of some 13 different external formats for spatial weights to facilitate interoperability with other spatial analytical packages.

Global and local measures of spatial autocorrelation comprise key aspects of the exploratory spatial data analysis module: **esda**. Both global and local indicators for spatial association [19] are implemented for Moran's I, Geary's c, and the class of Getis-Ord statistics. Additionally, join count statistics for binary attributes are also provided and for all measures both analytical and permutation based approaches to inference are available. The **esda** module also contains a wide array of classification schemes for choropleth maps.

The **inequality** module implements a variety of statistics to analyze inequality in spatial distributions. Global measures of inequality such as Theil's information based statistic and the Gini index are included. Spatial versions of these inequality measures consist of the spatial decomposition of the Theil statistic [20] and the spatial Gini decomposition [21]. The latter statistics report on the degree to which overall levels inequality may mask differences in inequality between neighboring and non-neighboring pairs of observations.

Methods to form regions of spatially connected observations are provided by the **regionalization** module. These include the max-p algorithm [22], a heuristic that generates a partition of the maximum number of regions of spatially connected areas with each region specifying some threshold constraint (*i.e.*, population). Also contained in **regionalization** are methods to generate random regions for simulation purposes and methods for inference on regionalization solutions.

The **spatial dynamics** module consists of a series of methods that extend classic ESDA methods to incorporate a temporal dimension including the space-time LISA and related visualizations [23]. Similarly, research that has extended methods from the distributional dynamics literature, such as Markov chains, has resulted in new spatially explicit versions of these approaches, or so called methods of spatial distribution dynamics [24] which form the other main component of the **spatial dynamics** module.

Modern methods of spatial econometrics are provided via the **spreg** module. Included are techniques to test for and estimate spatial effects in linear regression models. Spatial dependence is handled through spatial autoregressive models while spatial heterogeneity is treated with spatial regime models. Methods of estimation include classical maximum likelihood estimation and more recently developed techniques based on the principle of generalized method of moments (GMM). [25] provides a comprehensive guide to the **spreg** module.

The most recently added core analytical module is **network**. This implements methods for the spatial analysis of phenomena that are constrained to network space [26]. **network** includes methods for global and local spatial association on networks, as well as point pattern based methods including G, F, and K functions. Utility methods for snapping point data to network edges as well as network segmentation are also included in the module.

In addition to these core modules there are **contrib** modules in PySAL where third party modules can be developed that require dependencies beyond the PySAL dependencies of NumPy and SciPy. These include support for interfacing with **shapely**, **clusterpy**, and a **viz** module that provides interfaces to visualization to frameworks such as **folium** and **d3**.

3. Open Geospatial Analytics on the Desktop

Since the majority of end users for PySAL typically work in desktop environments we have developed three main approaches to providing access to PySAL functionality on this platform. The first consist of completely new applications that have been developed from the ground up in which PySAL components play the role of computational engine. The second approach has been to integrate PySAL into an existing desktop application via the application's plugin architecture. A final approach to PySAL on the desktop has been to rely on the new paradigm of computational notebooks.

3.1. Desktop Applications Built with PySAL

CAST, which stands for Crime Analytics in Space-Time [27] is a stand-alone desktop program to detect spatial patterns and trends in point data (such as crimes or diseases). Users can represent different attributes and spatial contexts of events in views such as maps, graphs, and calendars that can be animated over time (Figure 1). All of these views are linked to allow analysts to identify how selected subsets of the data are characterized across these dimensions. Statistical significance tests are applied to identify spatial clusters of events and temporal concentrations of crimes.

CAST emphasizes the analytics in the **spatial dynamics** and the **esda** modules in PySAL. Its interface is inspired from previous work on GeoDa and STARS [28] to implement full brushing and linking, but extends the interface functionality in important ways. CAST allows for the display of multiple shape file layers, and this can be accomplished within a single view or with each view as a separate window. Spatial, temporal and space-time queries are also supported. The program runs on the three main operating systems: Windows, MacOSX and Linux.

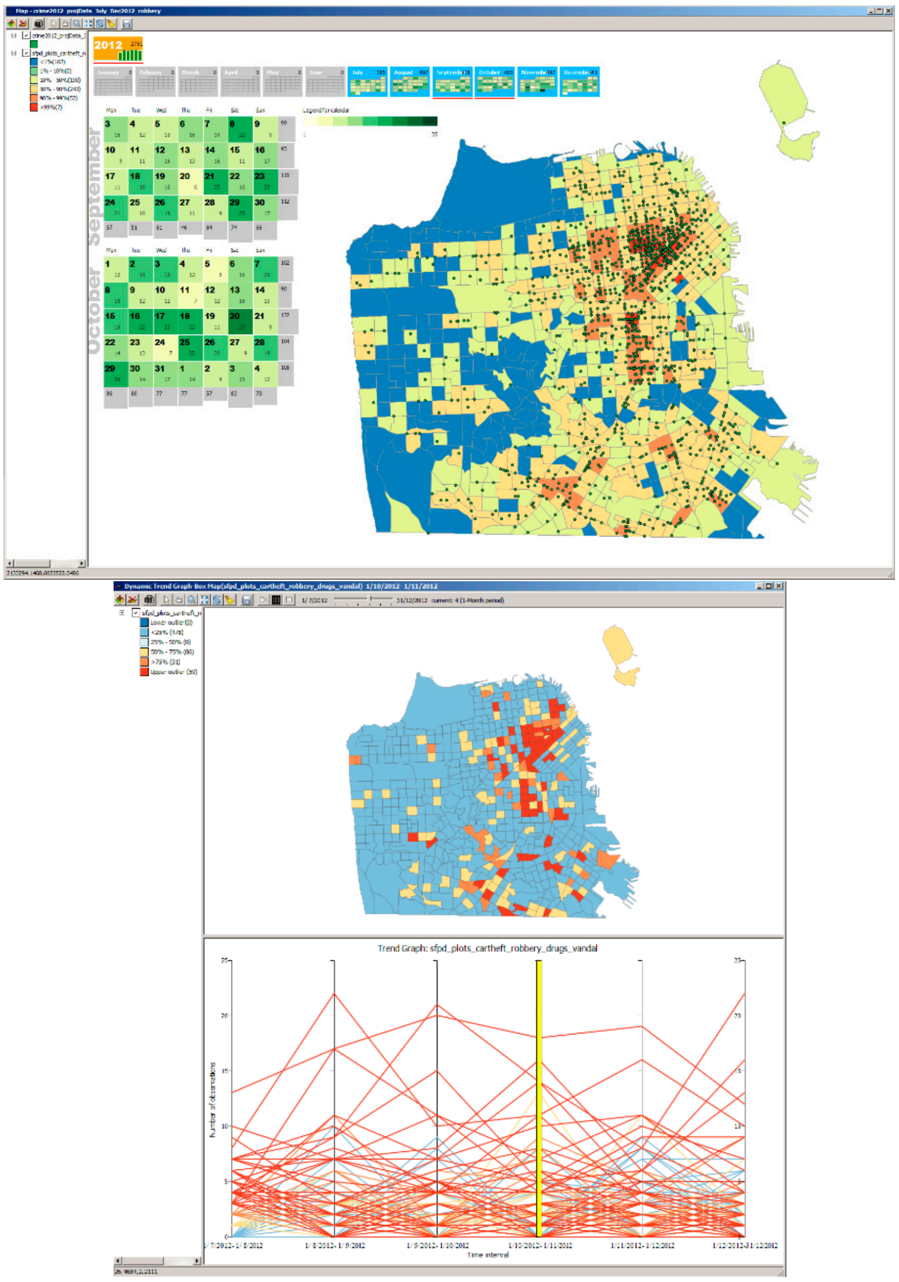
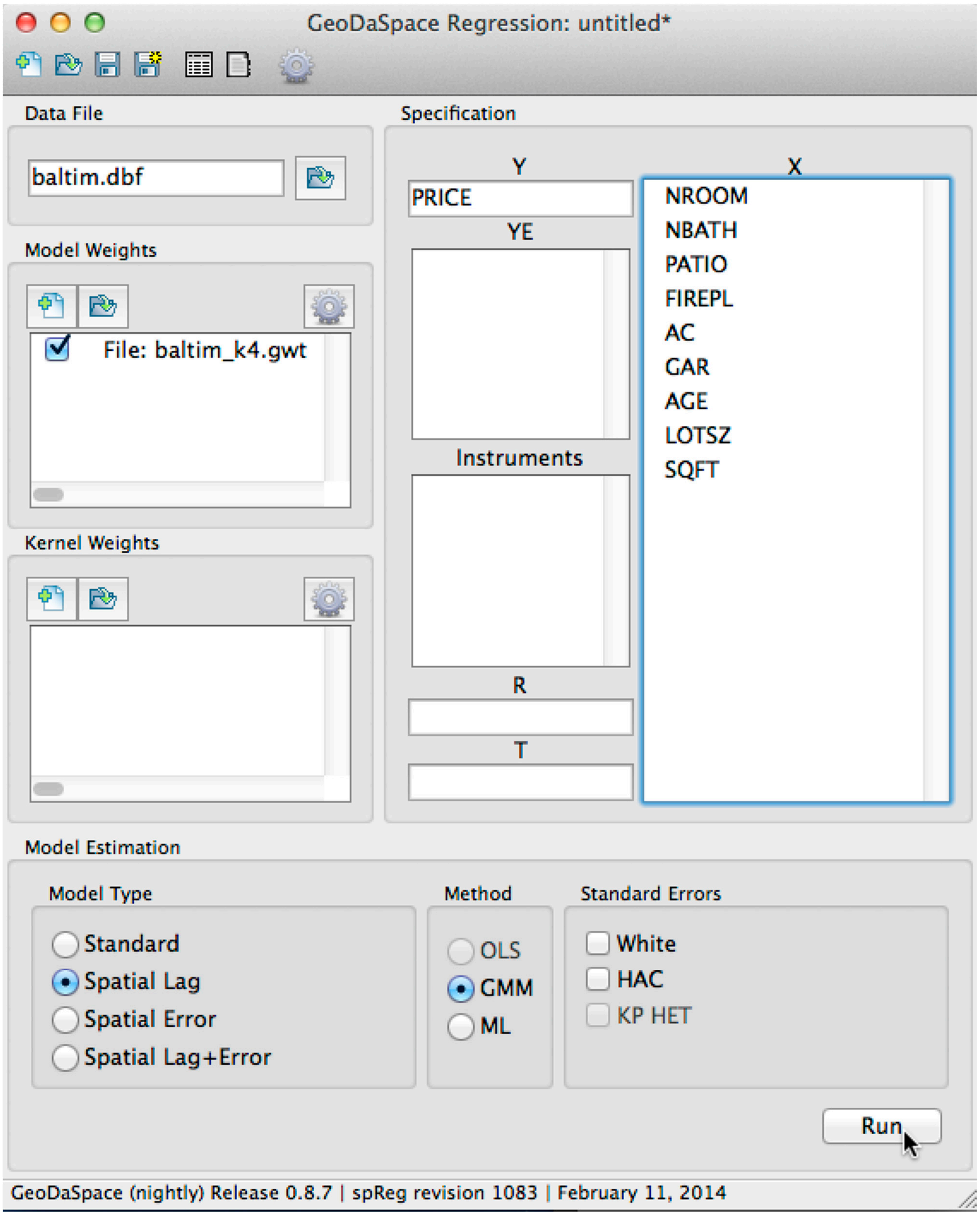


Figure 1. PySAL delivered as a stand-alone desktop application (CAST).

GeoDaSpace [25] is a second example of a desktop application that wraps a subset of PySAL functionality to provide a friendly graphical interface to advanced spatial analytics. In this case the focus is on a subset of the models and methods in the **spreg** component of the library. Figure 2 provides an example of using GeoDaSpace which illustrates a particular GUI dialogue for specifying a model together with the results of model estimation and diagnostic tests in the window to its right.

At first glance the idea of writing a completely new desktop application for geospatial analysis would appear to be a daunting task and perhaps wasteful of developer resources. However, there are two reasons this turns out not to be the case. First, a funding agency may have particular requirements for a specialized application and, indeed, this was the case for CAST. Secondly, because PySAL provided the analytical engine for the applications above, the development effort focuses only on the interface (interactive graphics for CAST, and menu driven interface for GeoDaSpace).


GeoDaSpace Regression: untitled*

Data File: baltim.dbf

Model Weights: [✓] File: baltim_k4.gwt

Kernel Weights

Specification

Y: PRICE

YE

Instruments

R

T

X: NROOM, NBATH, PATIO, FIREPL, AC, GAR, AGE, LOTSZ, SQFT

Model Estimation

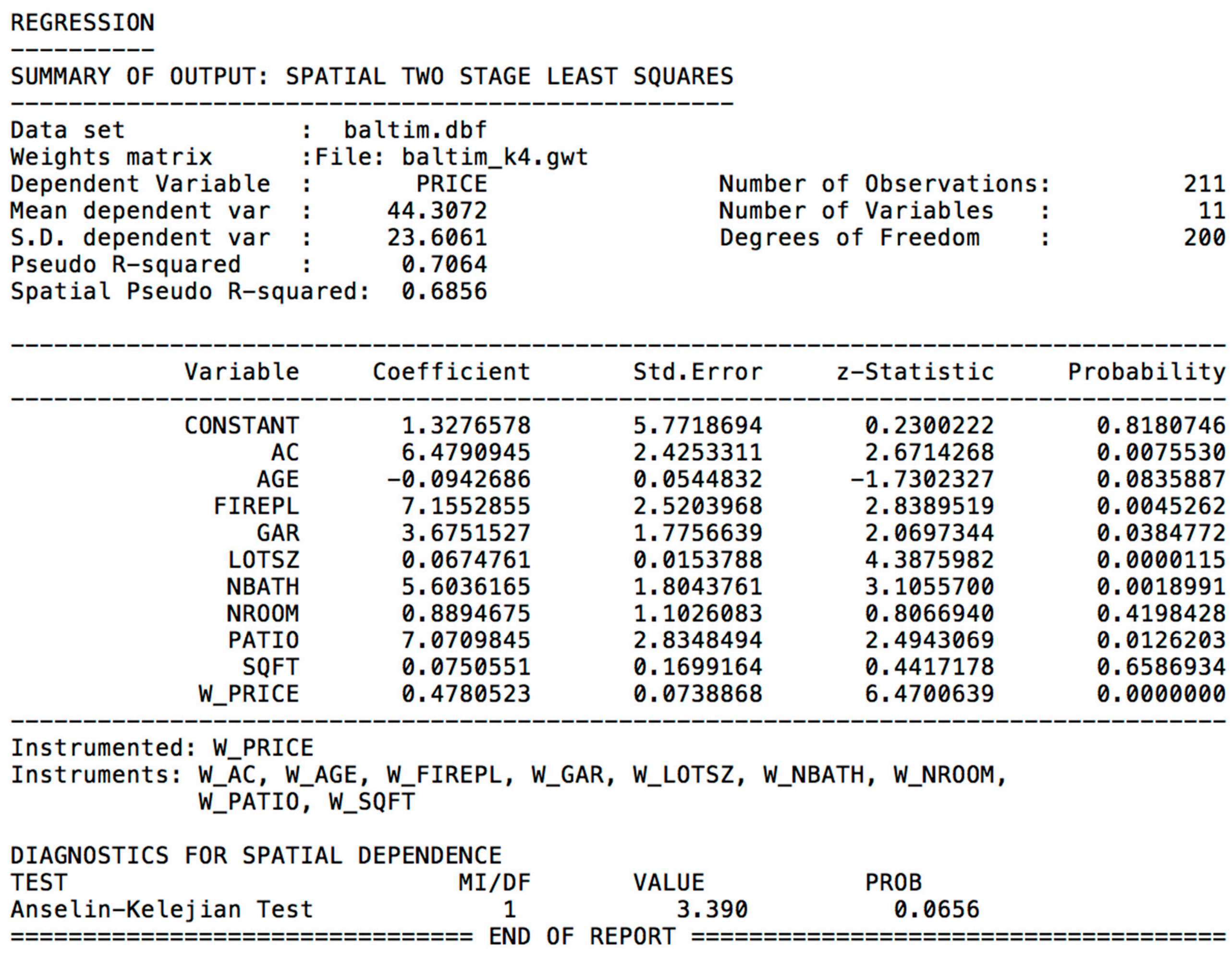
Model Type: Standard [✓ Spatial Lag] Spatial Error Spatial Lag+Error

Method: OLS [✓ GMM] ML

Standard Errors: White HAC KP HET

GeoDaSpace (nightly) Release 0.8.7 | spReg revision 1083 | February 11, 2014

Figure 2. *Cont.*



```
REGRESSION
----------
SUMMARY OF OUTPUT: SPATIAL TWO STAGE LEAST SQUARES
--------------------------------------------------
Data set            :  baltim.dbf
Weights matrix      :File: baltim_k4.gwt
Dependent Variable  :       PRICE                Number of Observations:         211
Mean dependent var  :     44.3072                Number of Variables   :          11
S.D. dependent var  :     23.6061                Degrees of Freedom    :         200
Pseudo R-squared    :      0.7064
Spatial Pseudo R-squared:  0.6856

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     z-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT       1.3276578       5.7718694       0.2300222       0.8180746
                  AC       6.4790945       2.4253311       2.6714268       0.0075530
                 AGE      -0.0942686       0.0544832      -1.7302327       0.0835887
              FIREPL       7.1552855       2.5203968       2.8389519       0.0045262
                 GAR       3.6751527       1.7756639       2.0697344       0.0384772
               LOTSZ       0.0674761       0.0153788       4.3875982       0.0000115
               NBATH       5.6036165       1.8043761       3.1055700       0.0018991
               NROOM       0.8894675       1.1026083       0.8066940       0.4198428
               PATIO       7.0709845       2.8348494       2.4943069       0.0126203
                SQFT       0.0750551       0.1699164       0.4417178       0.6586934
             W_PRICE       0.4780523       0.0738868       6.4700639       0.0000000
------------------------------------------------------------------------------------
Instrumented: W_PRICE
Instruments: W_AC, W_AGE, W_FIREPL, W_GAR, W_LOTSZ, W_NBATH, W_NROOM,
             W_PATIO, W_SQFT

DIAGNOSTICS FOR SPATIAL DEPENDENCE
TEST                           MI/DF       VALUE           PROB
Anselin-Kelejian Test            1          3.390          0.0656
================================ END OF REPORT =====================================
```

Figure 2. GeoDaSpace for spatial econometrics functionality from PySAL.

3.2. PySAL Toolkits for Desktop GIS

The second strategy for PySAL on the desktop is to leverage the Application Programming Interface (API) or plug-in architecture of an existing desktop application. One example of this is our development of a PySAL tookit for ESRI's ArcGIS program. Subsets of the functionality available in GeoDaSpace are incorporated as part of a PySAL ArcGIS Tool (Figure 3). A selection of the estimators from the **spreg** module together with methods for the construction of contiguity, distance or kernel spatial weights is exposed via the ArcGIS menu dialogue. The user is also prompted to select the dependent and independent variables to specify the model to be estimated. By integrating PySAL with ArcGIS, the spatial methods focus of the library is combined with the GIS infrastructure of ArcGIS. For instance, model results from the PySAL Toolbox, such as predicted values or residuals, can subsequently be mapped in ArcGIS.

In addition to the spatial econometrics focused toolkit for ArcGIS we have been collaborating with core members of the QGIS development team to prototype a PySAL plugin for the Processing Toolbox. Figure 4 demonstrates the application of the LISA class in the esda module of PySAL and its visualization via the QGIS interface. Other modules of PySAL are also shown in the toolbox, while plans are underway to convert the PySAL plugin to a core module within the Processing framework and release it as a component of QGIS.

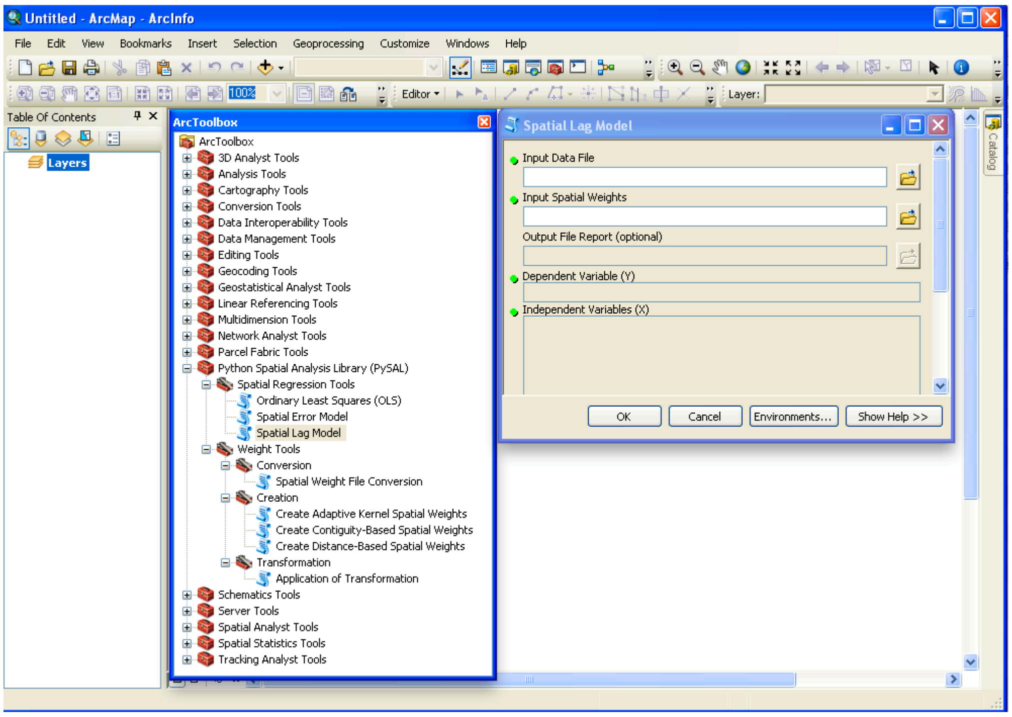


Figure 3. PySAL delivered as an ArcGIS Toolbox.

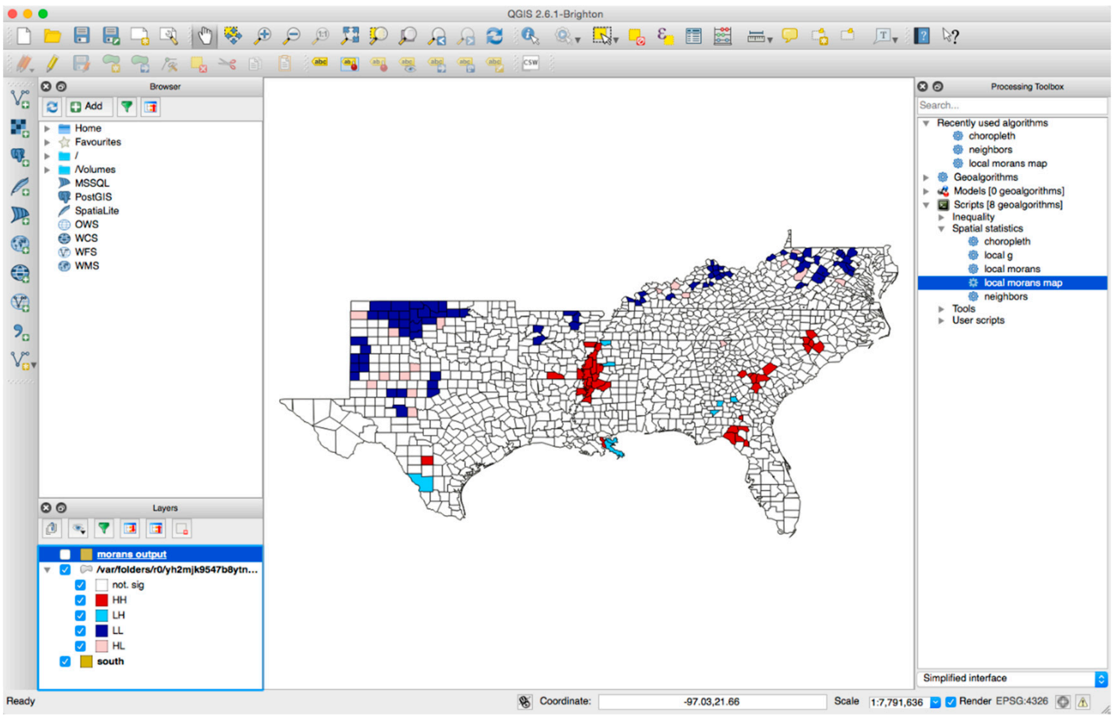


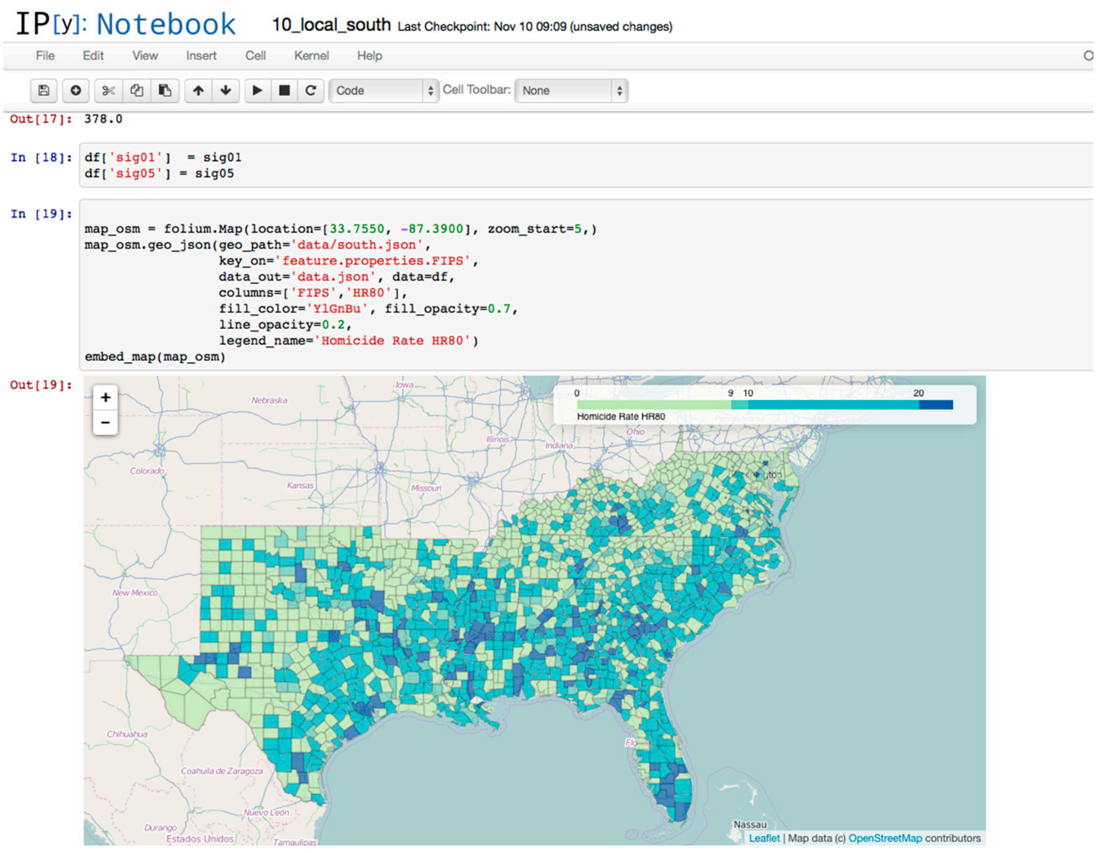
Figure 4. PySAL in the Processing Toolbox of QGIS.

This second strategy of using the plug-in architecture of a desktop GIS application has a number of advantages. First, geoprocessing, map projections, and support for a wide array of spatial data types can be provided by the GIS application, while the particular PySAL functionality is integrated as a plug-in. This largely removes a major development cost for the PySAL team, which, previously, had to develop similar functionality to bootstrap our analytical work. Now, freed from having to spend developer time on these concerns, our team can instead focus on new geospatial analytics in the library.

3.3. Interactive Computing on the Desktop

The final approach to PySAL on the desktop is inspired by the use case of a single researcher working on data analysis from an exploratory perspective. In this mode the sophisticated researcher would like to have access to all the functionality in the library and clearly a GUI-based interface that exposes only a subset of PySAL would constrain this functionality. PySAL can be employed in several ways for this type of analysis. First, since it is a Python module, importing the PySAL library into a running Python interpreter at the command line offers a very flexible user interface for interactive computation. Secondly, PySAL could also be imported into scripts for use in batch processing and simulation work. Often these first two modes are used together in carrying out exploratory work, as the researcher will start at the interactive prompt to explore certain methods and try different sequences of operations. Once satisfied with the workflow, the sequence of commands can be combined in a script for later use or for replication purposes.

A third approach to using PySAL for exploratory scientific computing on the desktop is the IPython notebook [17]. The notebook is a browser based interactive computing environment where the researcher can combine exploratory scripting with textual comments in LaTeX or Markdown format together with embedded scientific visualizations and other rich media. This offers a revolutionary framework for carrying out scientific computing on the desktop. Figure 5 illustrates the use of PySAL and folium within a notebook session. The notebook consists of cells that can group together code blocks, as in cells “In [29]:” and “In [30]:” in the figure, as well as the output or visualization resulting from the execution of a block, as in the map in “Out [30]:”.



```
Out[17]: 378.0

In [18]: df['sig01']  = sig01
         df['sig05'] = sig05

In [19]:
         map_osm = folium.Map(location=[33.7550, -87.3900], zoom_start=5,)
         map_osm.geo_json(geo_path='data/south.json',
                          key_on='feature.properties.FIPS',
                          data_out='data.json', data=df,
                          columns=['FIPS','HR80'],
                          fill_color='YlGnBu', fill_opacity=0.7,
                          line_opacity=0.2,
                          legend_name='Homicide Rate HR80')
         embed_map(map_osm)

Out[19]:
```

Figure 5. PySAL in the IPython notebook.

4. Open Spatial Analytics Middleware

Desktop applications are typically used by a single researcher on a single machine. Moving into a distributed and high performance computing context requires architectures that can support interoperability between components and services, as well as exploit the new capabilities of the available hardware. This is not a simple matter of installing the desktop version of the PySAL library on new hardware, but rather a number of fundamental challenges arise that must be addressed if the potential of high performance computing (HPC) for spatial analysis are to be realized. Key among these are the issues of provenance and meta-data for spatial analytical workflows, which become particularly critical when considering distributed computing and interoperability between different services [16]. While there has been much work on provenance and meta-data for spatial data and geoprocessing, similar frameworks for chaining together spatial analytical methods are in their infancy. Without these frameworks, the replication and reproducibility of geospatial analysis on HPC and distributed platforms will remain out of reach.

A second set of challenges pertains to the demands of interoperability. To ensure that different spatial analytical packages and services can be chained together for problem solving it is essential that each module in the chain offers a discoverable interface so that its functionality can be understood and exploited. Automation of this discovery process remains a holy grail of distributed services and computing.

We are addressing the challenges of provenance and analytical discoverability through middleware that enables us to move PySAL into HPC and distributed domains. We discuss these efforts below.

4.1. Provenance and Meta-Data for Scientific Workflow Support

The PySAL provenance architecture [31] is designed based on the following principles: (1) it should be light-weighted; (2) it should support easy integration with spatial analytical and visualization modules within and outside of the realm of PySAL; (3) it should facilitate automated invocation of PySAL functions and chaining of spatial analytical workflows; (4) it should provide a machine-understandable and human-readable data structure; and (5) it should enable the replication of analytical results.

A major obstacle that keeps existing provenance schemes, such as the XML-based (Extensible Markup Language) W3C PROV from receiving widespread adoption in operational software systems is the complex encoding of provenance information. This complexity creates an additional computational burden. Consequently, we have opted to base the construction of the PySAL provenance module on JSON (JavaScript Object Notation) instead of XML. Figure 6a demonstrates an example of the provenance structure for spatial weights, a fundamental operation in spatial statistics necessary to quantify the spatial dependence among observations. In general, besides the input file format and operation, additional parameters, such as whether the values in a spatial weights matrix need to be standardized or not are also encoded, by the key "transform". Adopting JSON as the encoding scheme for provenance makes the structure easy to read, by both machine and researchers. In addition, the integration with Python is facilitated since the JSON file follows the same key-value pair definition as Python's build-in dictionary data structure. Therefore, a seamless integration of PySAL and the provenance module can be realized. Also, because the JSON structure removes redundant open and close tags required in XML, as well as other scheme related information, human readability is greatly improved.

(a) Provenance for spatial weights generation

```
{
  "input1": {
    "data1":{
      "type": "shp",
      "uri": "http://toae.org/pub/columbus.shp"}
  },
  "weight_type": "rook",
  "transform": "O"
}
```

(b) Provenance structure for generating local Moran's I

```
{
  "input1": {
    "weights":{
      "type": "prov",
      "uri": "http://toae.org/pub/wrook1.wmd"},
    "attributes":{
      "type": "text",
      "type": "CRIME",
      "uri": "http://toae.org/pub/columbus.dbf"},
  },
  "analysis_type": "Moran_local",
  "transform": "O"
  "output": {
      "p_norm": "float"
```

Figure 6. Provenance meta-data for spatial analytical workflows in JavaScript Object Notation (JSON) format.

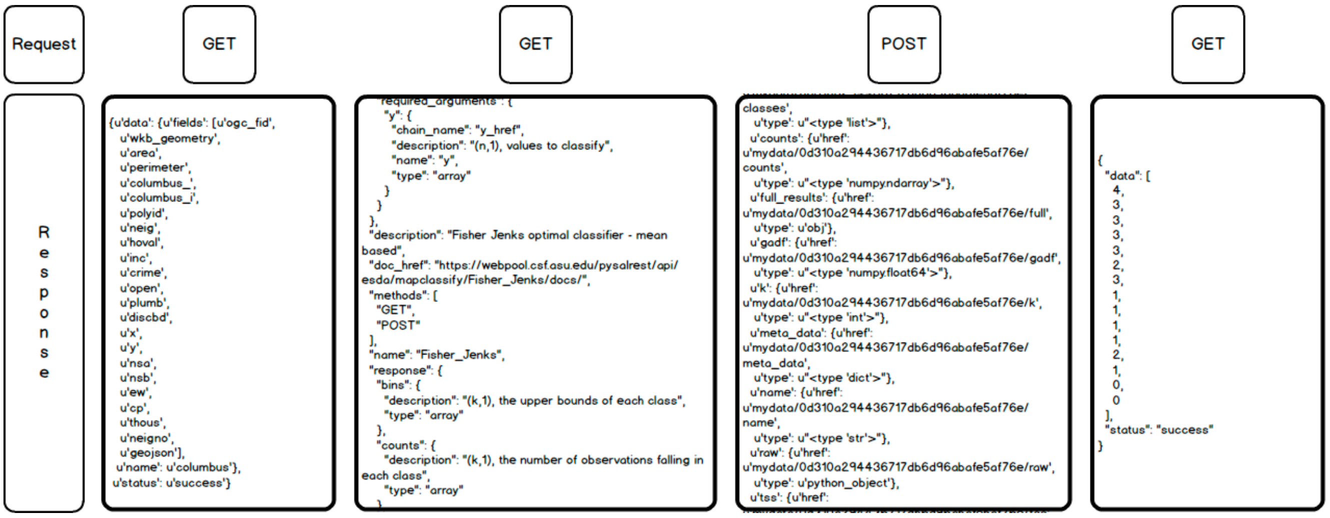
The provenance framework is designed to be extensible and enables automatic data reproduction and validation as part of an analytical workflow. Figure 6b demonstrates the contents of a provenance file to invoke the Local Moran's I functionality from the PySAL spatial autocorrelation module. Rather than defining the actual input data as such, a recursive approach is taken by providing a link to another provenance file that details the creation of the spatial weights (as in Figure 6a). This embedded design thus provides the ability for recursive provenance traversing and validation of intermediate data.

Through this provenance module and the associated provenance engine, all atomic steps in a spatial analysis workflow can be automatically recorded in both a machine- and human-readable format and thus facilitate the cross-validation of scientific findings. It can also be delivered through a web service interface, such as OGC WPS, or REST API to further enhance the interoperability of spatial analytical modules across different software packages.

4.2. PySAL REST API for Discoverability and Interoperability

PySAL-REST is a server side API generated by introspecting the PySAL code base, wrapping the PySAL library, and exposing spatial analytical functionality via a RESTful interface. REpresentational State Transfer [32] is a set of implementation constraints designed to support interoperability between distributed software components. These requirements include: (1) a clear separation of concerns as embodied by a layered client-server architecture; (2) stateless communication where each message must contain all information required to perform some task, e.g., initiate a spatial processing task; and finally (3) standardization of communication protocols, essential to allow the independent modification of the client or server. PySAL-REST is designed to comply with these requirements and to provide a backend spatial analysis platform accessible via standard HTTP verbs (GET, POST, PUT, DELETE) in a highly distributed environment.

PySAL-REST utilizes JSON as the standard communication protocol. JSON is a standard, plain text format to support communication between computing components [33] that finds wide usage as an efficient communication format across the web [30,34]. Figure 7, below, illustrates a sequence of analytical steps required to: (1) GET request information about an available dataset; (2) GET request for the arguments required to run a given analytical method; (3) POST request to perform an atomic analytical task; and (4) GET request to access a component of the analytical result.



```
{u'data': {u'fields': [u'ogc_fid',
   u'wkb_geometry',
   u'area',
   u'perimeter',
   u'columbus_',
   u'columbus_i',
   u'polyid',
   u'neig',
   u'hoval',
   u'inc',
   u'crime',
   u'open',
   u'plumb',
   u'discbd',
   u'x',
   u'y',
   u'nsa',
   u'nsb',
   u'ew',
   u'cp',
   u'thous',
   u'neigno',
   u'geojson'],
  u'name': u'columbus'},
 u'status': u'success'}
```

```
    "required_arguments": {
     "y": {
       "chain_name": "y_href",
       "description": "(n,1), values to classify",
       "name": "y",
       "type": "array"
     }
    }
  },
  "description": "Fisher Jenks optimal classifier - mean
based",
  "doc_href": "https://webpool.csf.asu.edu/pysalrest/api/
esda/mapclassify/Fisher_Jenks/docs/",
  "methods": [
    "GET",
    "POST"
  ],
  "name": "Fisher_Jenks",
  "response": {
    "bins": {
      "description": "(k,1), the upper bounds of each class",
      "type": "array"
    },
    "counts": {
      "description": "(k,1), the number of observations falling in
each class",
      "type": "array"
    }
```

```
classes',
   u'type': u"<type 'list'>"},
  u'counts': {u'href':
u'mydata/0d310a294436717db6d96abafe5af76e/
counts',
   u'type': u"<type 'numpy.ndarray'>"},
  u'full_results': {u'href':
u'mydata/0d310a294436717db6d96abafe5af76e/full',
   u'type': u'obj'},
  u'gadf': {u'href':
u'mydata/0d310a294436717db6d96abafe5af76e/gadf',
   u'type': u"<type 'numpy.float64'>"},
  u'k': {u'href':
u'mydata/0d310a294436717db6d96abafe5af76e/k',
   u'type': u"<type 'int'>"},
  u'meta_data': {u'href':
u'mydata/0d310a294436717db6d96abafe5af76e/
meta_data',
   u'type': u"<type 'dict'>"},
  u'name': {u'href':
u'mydata/0d310a294436717db6d96abafe5af76e/
name',
   u'type': u"<type 'str'>"},
  u'raw': {u'href':
u'mydata/0d310a294436717db6d96abafe5af76e/raw',
   u'type': u'python_object'},
  u'tss': {u'href':
```

```
{
  "data": [
    4,
    3,
    3,
    3,
    3,
    2,
    3,
    1,
    1,
    1,
    1,
    2,
    1,
    0,
    0
  ],
  "status": "success"
}
```

Figure 7. PySAL REST-API.

Within the context of PySAL-REST, automated code introspection and reflection provide key benefits over the development of independent interface layers. First, as new functionality is realized within the PySAL library, it is immediately available via PySAL-REST without additional development efforts. Second, meta-data describing each atomic spatial analysis operation is extracted directly from the PySAL source code, providing a self-documenting API that updates information essential to analysis chaining without intervention. Finally, code reflection allows analytical results to be bootstrapped with Analytical Metadata, providing essential provenance information. When combined, these three items offer a backend solution ideally suited to provide interoperable, self-describing and documenting methods to support Web-based applications and services.

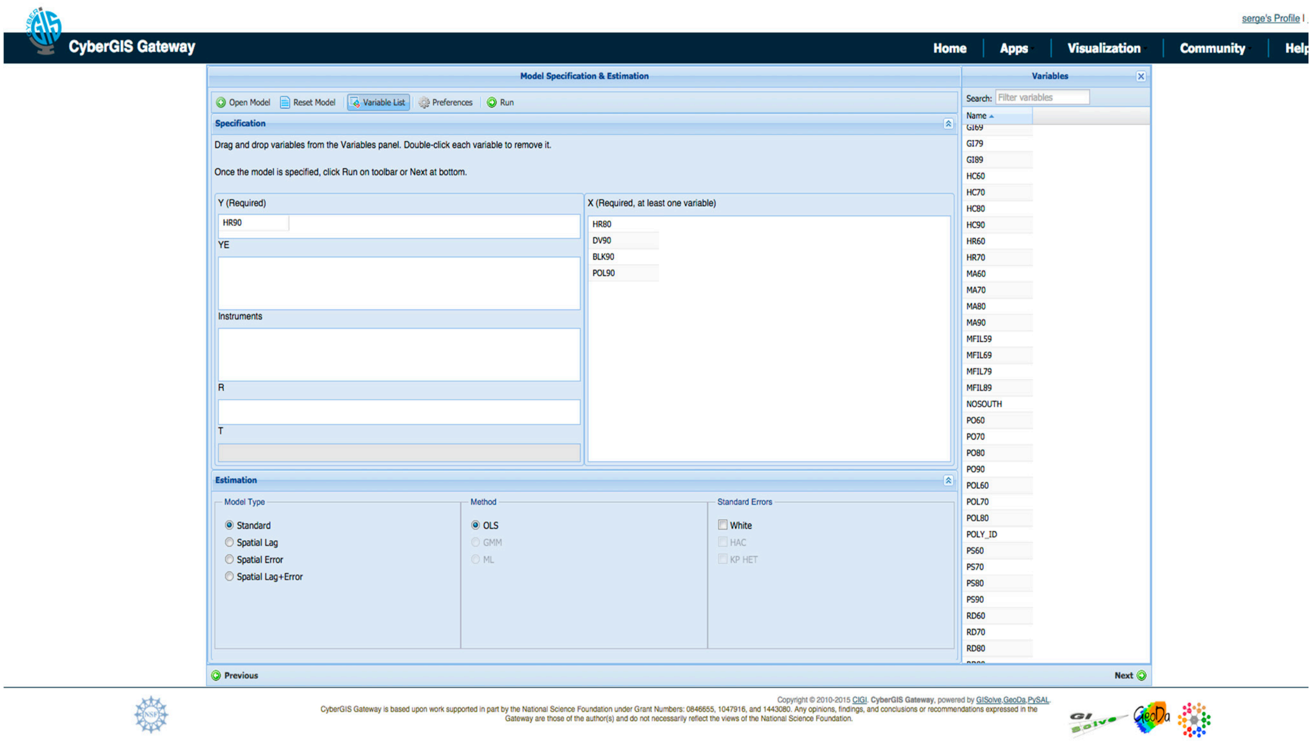
5. Web, Cloud and Distributed Open Geospatial Analytics

The middleware discussed above supports the movement of PySAL into Web, Cloud, Grid and distributed compute environments. Here we outline a selection of these implementations that cover a range of use cases for HPC spatial analysis.

5.1. Scientific Gateways

The CyberGIS Gateway [35] is an online CyberGIS software environment that supports cyberinfrastructure-enabled geospatial analysis and problem solving. The Gateway implements a number of high-performance and collaborative geospatial packages and applications on the NSF XSEEDE and Open Science Grid infrastructures, providing access to powerful computer resources via browser based user interfaces. Figure 8 illustrates the CG-PySAL component of the Gateway which exposes components of the **spreg** module of PySAL in support of spatial econometric analysis.

In addition to supporting the CyberGIS Gateway, a CyberGIS Toolkit brings together a suite of loosely coupled open-source geospatial software modules with an emphasis on scalability. It provides a platform for researching the scalability of spatial analysis algorithms and their coupling with other packages to support flexible workflows. Figure 9 reports the integration of one research effort focusing on the parallelization of one component of the PySAL library, namely scalable implementations of optimal choropleth map classifications for massive data sets [29,36].



Specification

Drag and drop variables from the Variables panel. Double-click each variable to remove it.

Once the model is specified, click Run on toolbar or Next at bottom.

Y (Required): HR90

YE

Instruments

R

T

X (Required, at least one variable): HR80, DV90, BLK90, POL90

Estimation

Model Type: [✓ Standard] Spatial Lag, Spatial Error, Spatial Lag+Error

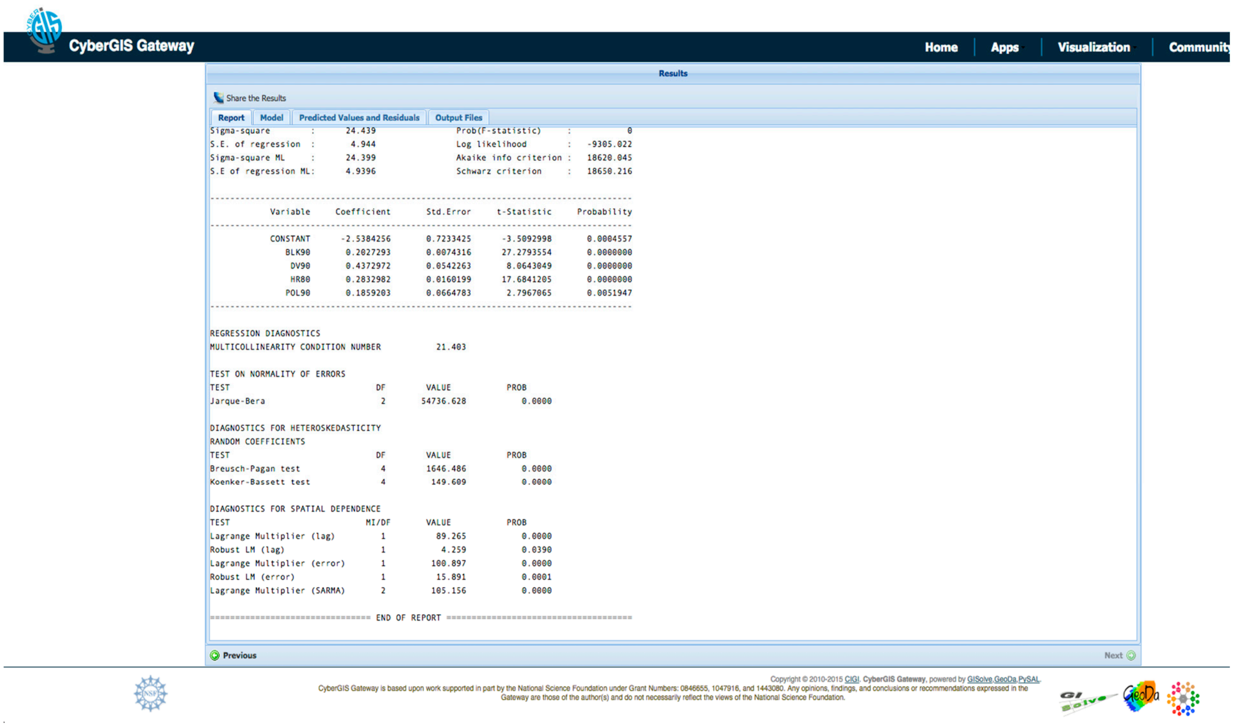
Method: [✓ OLS] GMM, ML

Standard Errors: White, HAC, KP HET

Variables: GI69, GI79, GI89, HC60, HC70, HC80, HC90, HR60, HR70, MA60, MA70, MA80, MA90, MFIL59, MFIL69, MFIL79, MFIL89, NOSOUTH, PO60, PO70, PO80, PO90, POL60, POL70, POL80, POLY_ID, PS60, PS70, PS80, PS90, RD60, RD70, RD80

Copyright © 2010-2015 CIGI. CyberGIS Gateway, powered by GISolve, GeoDa, PySAL.
CyberGIS Gateway is based upon work supported in part by the National Science Foundation under Grant Numbers: 0846655, 1047916, and 1443080. Any opinions, findings, and conclusions or recommendations expressed in the Gateway are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Figure 8. *Cont.*



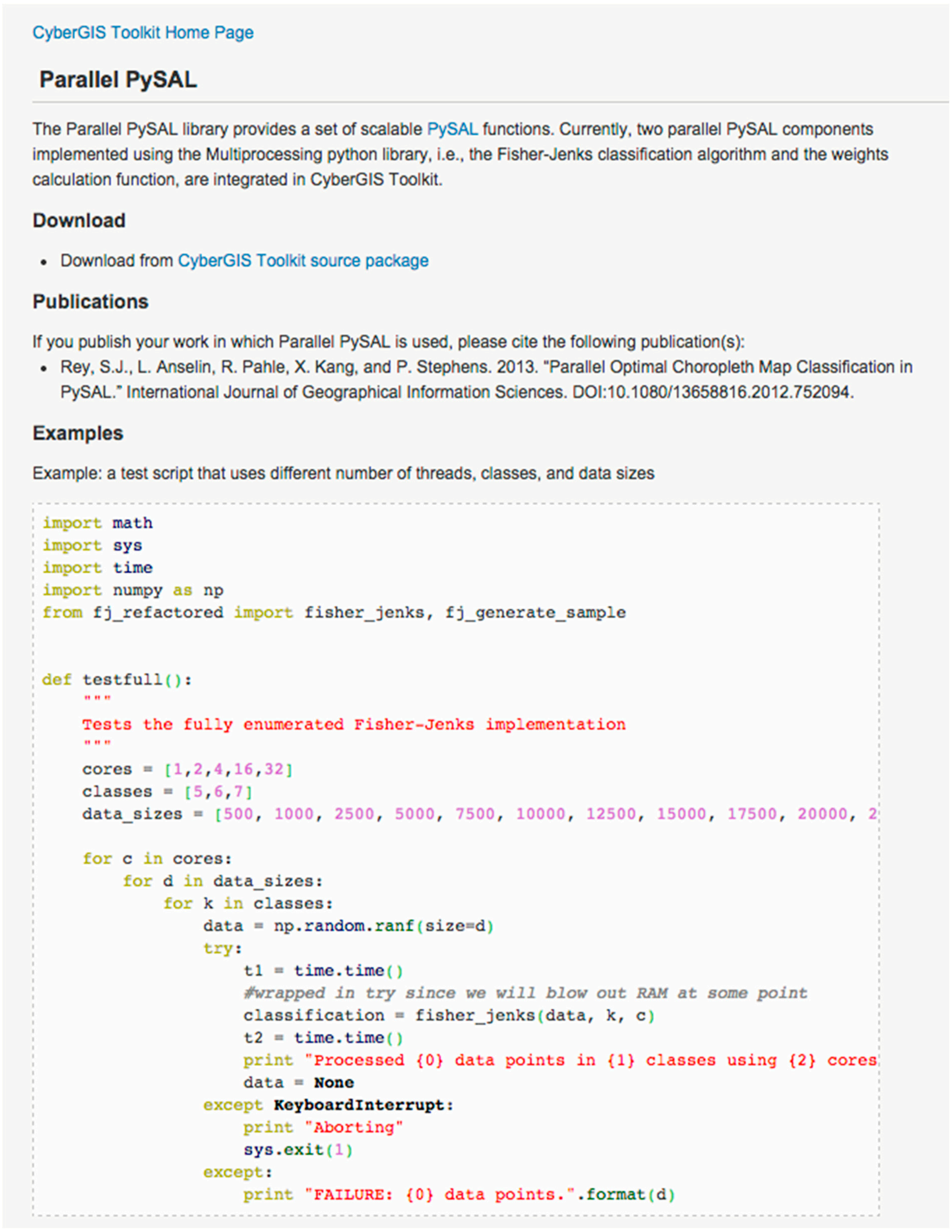
```
Sigma-square        :      24.439           Prob(F-statistic)     :          0
S.E. of regression  :       4.944           Log likelihood        :   -9305.022
Sigma-square ML     :      24.399           Akaike info criterion :   18620.045
S.E of regression ML:      4.9396           Schwarz criterion     :   18650.216

------------------------------------------------------------------------------------
           Variable     Coefficient       Std.Error     t-Statistic     Probability
------------------------------------------------------------------------------------
           CONSTANT      -2.5384256       0.7233425      -3.5092998       0.0004557
              BLK90       0.2027293       0.0074316      27.2793554       0.0000000
               DV90       0.4372972       0.0542263       8.0643049       0.0000000
               HR80       0.2832982       0.0160199      17.6841205       0.0000000
              POL90       0.1859203       0.0664783       2.7967065       0.0051947
------------------------------------------------------------------------------------

REGRESSION DIAGNOSTICS
MULTICOLLINEARITY CONDITION NUMBER         21.403

TEST ON NORMALITY OF ERRORS
TEST                            DF        VALUE           PROB
Jarque-Bera                      2     54736.628          0.0000

DIAGNOSTICS FOR HETEROSKEDASTICITY
RANDOM COEFFICIENTS
TEST                            DF        VALUE           PROB
Breusch-Pagan test               4      1646.486          0.0000
Koenker-Bassett test             4       149.609          0.0000

DIAGNOSTICS FOR SPATIAL DEPENDENCE
TEST                          MI/DF       VALUE           PROB
Lagrange Multiplier (lag)        1        89.265          0.0000
Robust LM (lag)                  1         4.259          0.0390
Lagrange Multiplier (error)      1       100.897          0.0000
Robust LM (error)                1        15.891          0.0001
Lagrange Multiplier (SARMA)      2       105.156          0.0000

================================ END OF REPORT =====================================
```

Copyright © 2010-2015 CIGI. CyberGIS Gateway, powered by GISolve, GeoDa, PySAL.
CyberGIS Gateway is based upon work supported in part by the National Science Foundation under Grant Numbers: 0846655, 1047916, and 1443080. Any opinions, findings, and conclusions or recommendations expressed in the Gateway are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Figure 8. PySAL in the CyberGIS Gateway.

5.2. PySAL-Cloud

The functionality of PySAL can also be wrapped in a cloud-computing platform to provide cloud-based spatial analysis services. These services allow users to apply spatial analysis on data residing on their desktop or stored anywhere in the cloud using the powerful computing capabilities provided by the cloud. Since all the computational work is running on the cloud, users can access these cloud-based services worldwide from any device. Figure 10 illustrates PySAL-cloud, a prototype of a cloud-based spatial analysis service. It allows users to access their data locally, in a private cloud (e.g., via Dropbox or CartoDB; see Figure 10a), or to retrieve data from a public cloud (e.g., automatically crawling data from Google Maps or searching data from Socrata; see Figure 10b). The PySAL-Cloud service also engages the latest HTML5 techniques to visualize maps, tables and plots in a configurable dashboard style and implements brushing and linking capability (see Figure 10c). Through a graphical user interface, the functionality from PySAL can be invoked to explore spatial data and run spatial regressions (see Figure 10d). For third-part developers, all of PySAL's functionality can be wrapped as cloud APIs, which can be used to integrate with other cloud-based services to compose various applications. The PySAL cloud service is implemented on a high performance computing infrastructure. This allows for the optimization of the code for handling large spatial data by utilizing specialized parallel and distributed computing capability that takes advantage of the HPC infrastructure. This customization and optimization go beyond what can be implemented in the standard stable and cross platform release of the library. For example, in [37] an experimental approach is outlined that uses Hadoop to create spatial weights from very large spatial datasets.



CyberGIS Toolkit Home Page

Parallel PySAL

The Parallel PySAL library provides a set of scalable PySAL functions. Currently, two parallel PySAL components implemented using the Multiprocessing python library, i.e., the Fisher-Jenks classification algorithm and the weights calculation function, are integrated in CyberGIS Toolkit.

Download

- Download from CyberGIS Toolkit source package

Publications

If you publish your work in which Parallel PySAL is used, please cite the following publication(s):

- Rey, S.J., L. Anselin, R. Pahle, X. Kang, and P. Stephens. 2013. "Parallel Optimal Choropleth Map Classification in PySAL." International Journal of Geographical Information Sciences. DOI:10.1080/13658816.2012.752094.

Examples

Example: a test script that uses different number of threads, classes, and data sizes

```
import math
import sys
import time
import numpy as np
from fj_refactored import fisher_jenks, fj_generate_sample


def testfull():
    """
    Tests the fully enumerated Fisher-Jenks implementation
    """
    cores = [1,2,4,16,32]
    classes = [5,6,7]
    data_sizes = [500, 1000, 2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000, 2

    for c in cores:
        for d in data_sizes:
            for k in classes:
                data = np.random.ranf(size=d)
                try:
                    t1 = time.time()
                    #wrapped in try since we will blow out RAM at some point
                    classification = fisher_jenks(data, k, c)
                    t2 = time.time()
                    print "Processed {0} data points in {1} classes using {2} cores
                    data = None
                except KeyboardInterrupt:
                    print "Aborting"
                    sys.exit(1)
                except:
                    print "FAILURE: {0} data points.".format(d)
```

Figure 9. Parallel PySAL in the CyberGIS Toolkit.

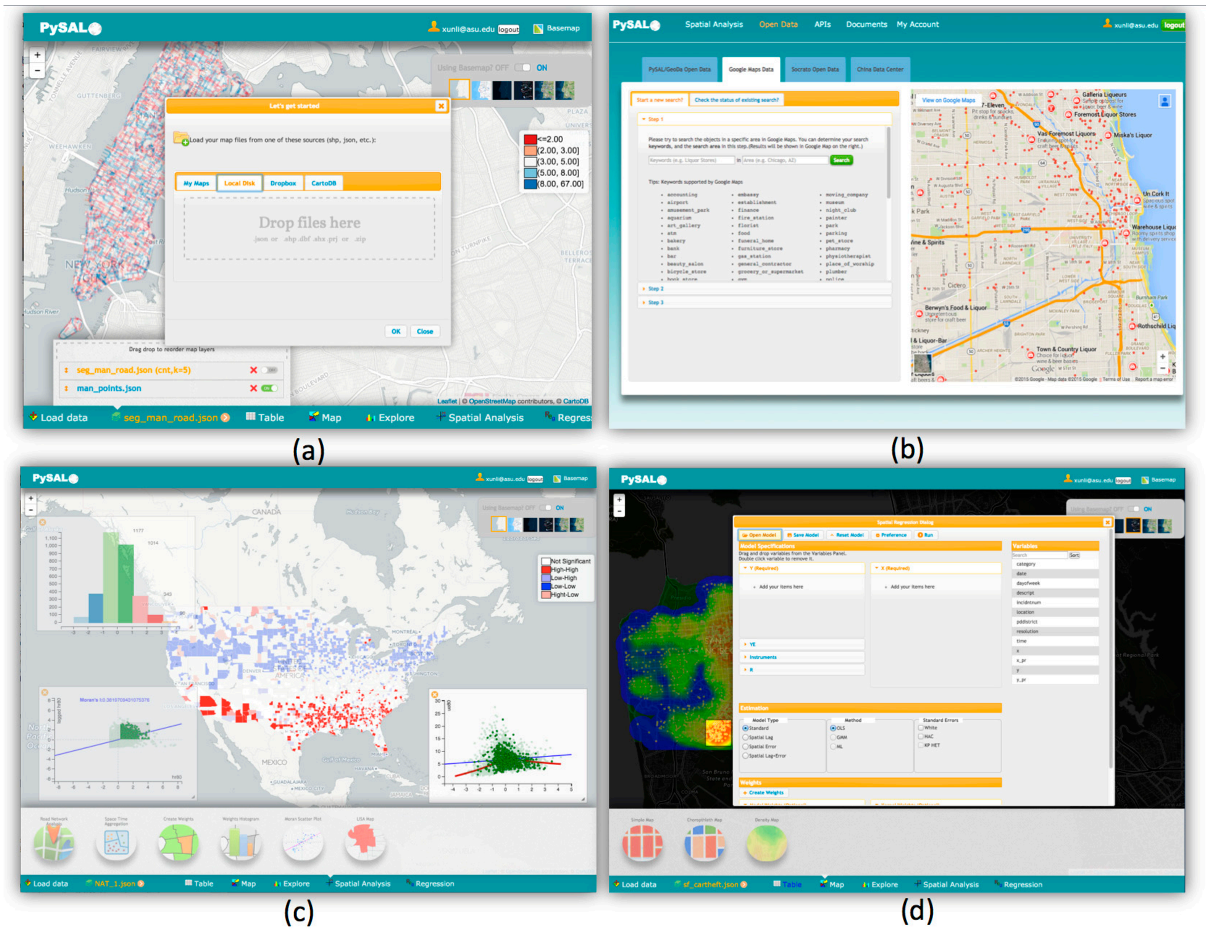


Figure 10. PySAL-Cloud: A prototype of a cloud-based spatial analysis platform.

5.3. Complex Systems Framework and Decision Support

The *Complex Systems Framework* (*CSF*) is an open-source example of a flexible decision-support system that integrates a range of computational modules such as PySAL within a common cyber-framework specifically designed for use in a decision-making context [38]. It provides a high-level environment that can link together different sophisticated computational models so that the output of one sub- model or process can provide input to another one. *CSF* applications run on a compute cluster consisting of 1200 processing cores and a combined 2400 GB of RAM to be able to generate results in decision time (Figure 11).

Figure 12 illustrates an example to detect patterns of crimes at different levels of spatial aggregation and for different time periods. The results are displayed on multiple dashboards that can be viewed on screens of different sizes, ranging from decision environments like Decision Theaters, to large-screen TVs in data fusion centers, desktop monitors, and mobile devices such as phones and touchpads. Since the displays are browser-based, they are platform independent (e.g., they run on iPad, Android, Windows, Mac or Linux). The data input for the analysis can be static (previously collected data) or near-real time, in which case the displays are automatically updated as soon as the database updates. The system is designed to be flexible and customizable: For instance, different content can be displayed on screens accessible to different groups of analysts and practitioners.

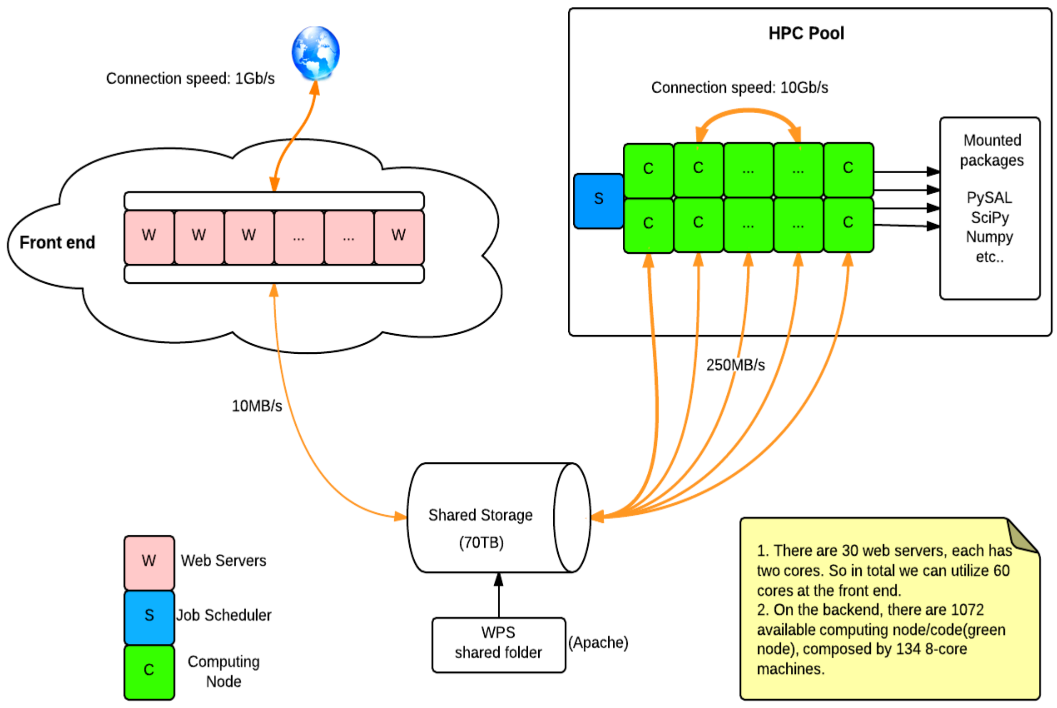


Figure 11. Complex systems framework components.

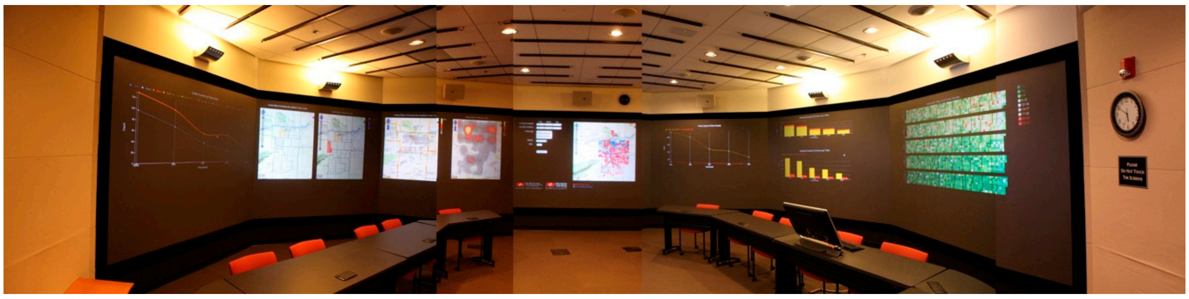


Figure 12. PySAL delivered in the Complex Systems Framework web-based decision support system.

While the gateway, toolkit, cloud and CSF represent specific operational examples of moving PySAL into the HPC and distributed context, we have discovered many new challenges and opportunities in these efforts. Moving forward, we envision a broader integration framework informed by these efforts that is portrayed in Figure 13. This platform follows a loose-coupling and modular design principle to facilitate easy extension in its architecture and functions, as well as to enhance reusability of the existing software components. From bottom to up, this framework is composed of five layers: the computing layer, software layer, web application layer, decision support layer and application layer. As our goal is to offer a powerful online spatial analysis and decision-making tool, the computing resources, or the hardware that hosts the software components becomes an important consideration in the software design. Scalability is achieved by running the software module on a HPC facility, such as ASU Advanced Computing Center (A2C2)'s super computers, GeoDa's high performance cluster, or other existing HPC resources. A scheduler and load balancer is deployed on the distributed computing nodes to redirect a user's computing request to a less-occupied node to realize a high throughout and a responsive online analysis system.

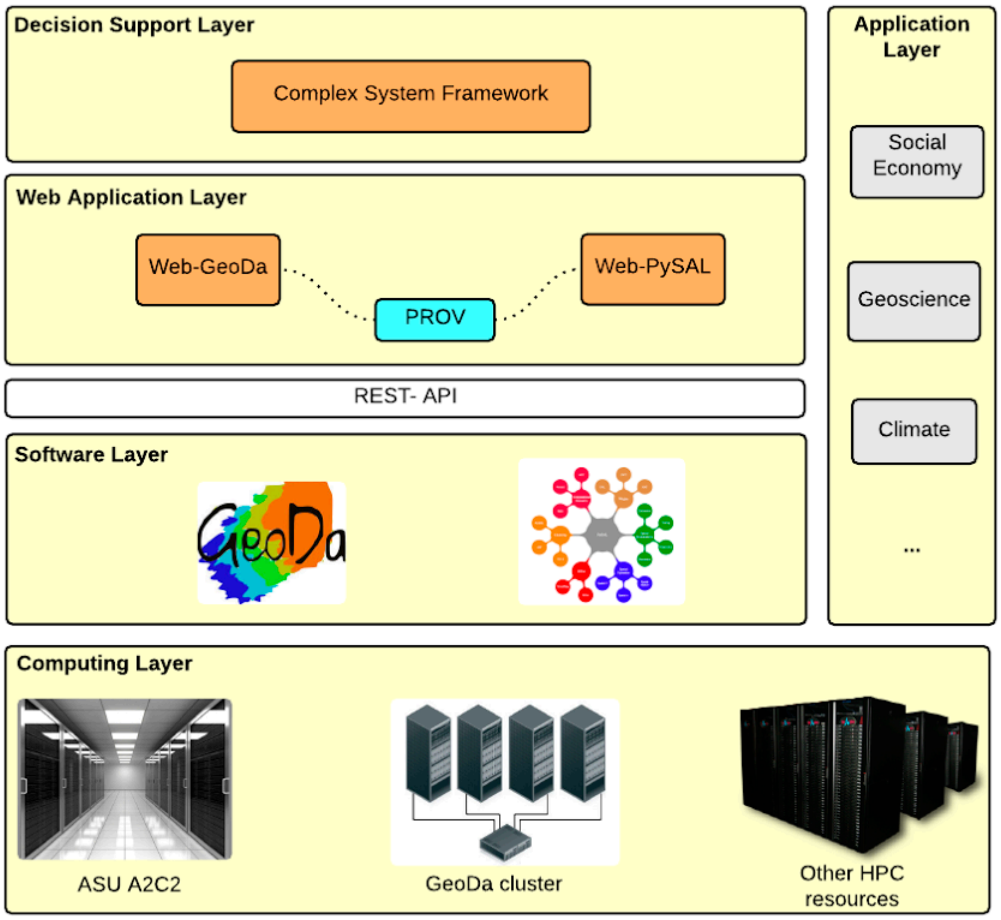


Figure 13. Open geospatial analytics integration framework

The actual spatial analysis is conducted by the open source GeoDa [39] and the PySAL library. To openly share these routines in a distributed environment, a set of RESTful APIs was developed to wrap atomic functions in both packages. These APIs help to hide the complexity of software installation and hardware configuration from tool consumers. On top of the software layer, there lies the web application layer, which provides the web-version of GeoDa and PySAL for online analysis. These two modules directly interact with the REST-API to access remotely the spatial analysis functions. They also provide online interfaces to allow an end user to perform the analysis easily as long as an Internet connection is available to him/her. To ensure the replicability of the analysis result, a provenance model is developed to allow the on-the-fly generation of the metadata that tracks the footprint of (intermediate and final) data products in a spatial analytical workflow. Meanwhile, a provenance engine is integrated into PySAL-Web to allow the interpretation of provenance metadata to reproduce and validate the results automatically. Going one layer up, we further deployed the complex system framework to enable the combination of analysis and workflows from GeoDa-Web and PySAL-Web to foster effective decision-making. This decision making tool can be used to support a variety of social-economic and geoscience applications.

6. Conclusion

The flexible delivery of open geospatial visual analytics in different desktop and web-based applications leverages the initial investment in developing the PySAL library for multiple purposes and audiences. Especially the web-based applications contribute to a technological infrastructure that promises to improve the continuous monitoring of desired outcomes through the near real-time analysis of different data sources (including open, administrative and crowd sourced data). As such these applications help disseminate the methods and tools developed within the university for use by analysts addressing policy-relevant challenges in other domains.

Building on our experiences in developing geospatial visual analytics we see several directions for future research. In order to scale existing spatial analytical methods to take advantage of HPC platforms it is necessary that current implementations be refactored to the particular characteristics of computer hardware at hand. Given the heterogeneity of such hardware, a one-size-fits-all solution is not possible. We are exploring highly optimized implementations on our research clusters and the provision of access to those methods via web based interfaces.

A second, and related, challenge is to avoid the trap of one-off implementations that are essentially proof-of-concepts. There is an inherent tension between research code that is used to develop a novel solution to a computational problem, and the general lack of production quality code that is intended for use by the wider research community. This hampers the advancement of science, as the prototypes are often not easily extended nor able to support interoperability with other tools. Through our work on PySAL we have demonstrated that, by building on a core library, it is possible to not only explore new approaches and prototypes, but also then integrate the advances and lessons learned from the prototype back into the library. This iterative process requires that we consider mechanisms to increase the modularity and interoperability of PySAL. Moving forward, we see the focus on provenance frameworks, meta-data and replication support as becoming increasingly important to open geospatial analytics.

Acknowledgments

Funding from the National Science Foundation (NSF SI2-SSI: CyberGIS Software Integration for Sustained Geospatial Innovation; NSF SES-1421935: New Approaches for Spatial Distribution Dynamics), the National Institutes of Health (2-R01CA126858: Geospatial Factors and Impacts II), the National Institute of Justice, Office of Justice Programs, U.S. Department of Justice (2009-SQ-B9-K101) and the Environmental Systems Research Institute (ESRI) is gratefully acknowledged.

Author Contributions

L.A. directed spreg module development and designed the architecture of the geovisual analytics framework; J.K. wrote an extended abstract that the present paper is based on; J.L. designed the PySAL REST API; W.L. developed the provenance model for scientific workflow; X.L. developed the PySAL based desktop application CAST, the ArcGIS Toolbox of PySAL, and the prototype of PySAL-Cloud; R.P. designed the architecture of the Complex Systems Framework; S.R. wrote the paper and directed the PySAL project.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Miller, H.J. The data avalanche is here. Shouldn't we be digging? *J. Reg. Sci.* **2010**, *50*, 181–201.
2. Arribas-Bel, D. Accidential, open and everywhere: Emerging data sources for the understanding of cities. *Appl. Geogr.* **2014**, *49*, 45–53.
3. Batty, M.; Axhausen, K.W.; Giannotti, F.; Pozdnoukhov, A.; Bazzani, A.; Wachowicz, M.; Ouzounis, G.; Portugali, Y. Smart cities of the future. *Eur. Phys. J.-Spec. Top.* **2012**, *214*, 481, doi:10.1140/epjst/e2012-01703-3.
4. Rey, S.J. Open regional science. *Ann. Reg. Sci.* **2014**, *52*, 825–837.
5. Lazer, D.; Pentland, A.; Adamic, L.; Aral, S.; Barabasi, A.-L.; Brewer, D.; Christakis, N.; Contractor, N.; Fowler, J.; Gutmann, M.; *et al.* Computational social science. *Science* **2009**, *323*, 721–723.
6. King, G. Ensuring the data-rich future of the social sciences. *Science* **2011**, *331*, 719–721.
7. Edelman, B. Using Internet data for economic research. *J. Econ. Perspect.* **2012**, *26*, 189–206.
8. Golder, S.A.; Macy, M.W. Digital footprints: Opportunities and challenges for online social research. *Annu. Rev. Sociol.* **2014**, *40*, 129–152.
9. The White House. *Fact Sheet: President Obama's Precision Medicine Initiative*; Office of the Press Secretary: Washington, DC, USA, 2015.
10. Batty, M. *The New Science of Cities*; MIT Press: Cambridge, UK, 2013.
11. Townsend, A. *Smart Cities: Big Data, Civic Hackers and the Quest for a New Utopia*; W.W. Norton & Co.: New York, NY, USA, 2014.
12. Goodchild, M.F. The quality of big (geo) data. *Dialogues Hum. Geogr.* **2013**, *3*, 280–284.
13. Crenshaw, J.; Schwartz, R.; Hong, J.; Sadeh, N. The livehoods project: Utilizing social media to understand the dynamics of a city. In Proceedings of the 6th International AAAI Conference on Weblogs and Social Media (ICWSM), Trinity College, Dublin, Ireland, 4–8 June 2012.
14. Kumar, S.; Morstatter, F.; Liu, H. *Twitter Data Analytics*; Springer-Verlag: Berlin, Germany, 2013.
15. Ruths, D.; Pfeffer, J. Social medial for large studies of behavior. *Science* **2014**, *346*, 1063–1064.
16. Anselin, L.; Rey, S.J. Spatial econometrics in an age of CyberGIScience. *Int. J. Geogr. Inf. Sci.* **2012**, *26*, 2211–2226.
17. Pérez, F.; Granger, B. IPython: A system for interactive scientific computing. *Comput. Sci. Eng.* **2007**, *9*, 21–29.
18. Rey, S.J.; Anselin, L. PySAL: A Python library of spatial analytical methods. In *Handbook of Applied Spatial Analysis*; Fisher, M.M., Getis, A., Eds.; Springer: Berlin, Germany, 2010; pp. 175–193.
19. Anselin, L. Local indicators of spatial association—LISA. *Geogr. Anal.* **1995**, *27*, 93–115.
20. Rey, S.J. Spatial analysis of regional income inequality. In *Spatially Integrated Social Science: Examples in Best Practice*; Goodchild, M.F., Janelle, D., Eds.; Oxford University Press: Oxford, UK, 2004; pp. 280–299.
21. Rey, S.J.; Smith, R.J. A spatial decomposition of the Gini coefficient. *Lett. Spat. Resour. Sci.* **2013**, *6*, 55–70.
22. Duque, J.; Anselin, L.; Rey, S. The max-p regions problem. *J. Reg. Sci.* **2012**, *53*, 397–419.

23. Rey, S.J.; Murray, A.T.; Anselin, L. Visualizing regional income distribution dynamics. *Lett. Spat. Resour. Sci.* **2011**, *4*, 81–90.
24. Rey, S.J. Spatial dynamics and space-time data analysis. In *Handbook of Regional Science*; Fischer, M.M., Nijkamp, P., Eds.; Springer: Berlin, 2014; pp. 1365–1383.
25. Anselin, L.; Rey, S.J. *Modern Spatial Econometrics in Practice: A Guide to GeoDa, GeoDaSpace and PySAL*; GeoDa Press: Chicago, IL, USA, 2014.
26. Okabe, A.; Sugihara, K. *Spatial Analysis along Networks: Statistical and Computational Methods*; Wiley: New York, NY, USA, 2012.
27. Rey, S.J.; Anselin, L.; Li, X.; Koschinsky, J. *Guide to Using the Crime Analytics for Space-Time (CAST) Desktop Software Program*; Technical Report; GeoDa Center for Geospatial Analysis and Computation, Arizona State University: Tempe, AZ, USA, 2013.
28. Rey, S.J.; Janikas, M.V. STARS: Space-time analysis of regional systems. *Geogr. Anal.* **2006**, *38*, 67–86.
29. Laura, J.; Rey, S.J. Improved parallel optimal choropleth map classification. In *Modern Accelerator Technologies for Geographic Information Science*; Shi, X., Kindratenko, X., Yang, C., Eds.; Springer: Berlin, Germany, 2013; pp. 197–212.
30. Lawrence, R. The space efficiency of XML. *Inf. Softw. Tech.* **2004**, *46*, 753–759.
31. Anselin, L.; Rey, S.J.; Li, W. Metadata and provenance for spatial analysis: The case of spatial weights. *Int. J. Geogr. Inf. Sci.* **2014**, *28*, 2261–2280.
32. Fielding, R.T. Architectural styles and the design of network-based software architectures. Ph.D. Thesis, University of California, Irvine, CA, USA, 2000.
33. ECAM. *The JSON Data Interchange Format*; ECAM-404 2013; ECAM International: Geneva, Switzerland, 2014.
34. Severance, C. Discovering JavaScript object notation. *Computer* **2012**, *45*, 6–8.
35. Wang, S.; Anselin, L.; Bhaduri, B.; Crosby, C.; Goodchild, M.; Liu, Y.; Nyerges, T. CyberGIS software: A synthetic review and integration roadmap. *Int. J. Geogr. Inf. Sci.* **2013**, *27*, 2122–2145.
36. Rey, S.J.; Anselin, L.; Pahle, R.; Kang, X.; Stephens, P. Parallel optimal choropleth map classification in PySAL. *Int. J. Geogr. Inf. Sci.* **2013**, *27*, 1023–1039.
37. Li, X.; Li, W.; Anselin, L.; Rey, S.J.; Koschinsky, J. A map-reduce algorithm to create contiguity weights for spatial analysis of big data. In Proceedings of the 3rd ACM SIGSpatial International Workshop on Analytics for Big Geospatial Data, 4 November 2014, Dallas, TX, USA; pp. 50–53.
38. Pahle, R. Complex Systems Framework—Integrating Analytics, Visualization and Collaboration. GeoDa Center for Geospatial Analysis and Computation Working Paper. Unpublished work, 2014.
39. Anselin, L.; Syabri, I.; Kho, Y. GeoDa: An introduction to spatial data analysis. *Geogr. Anal.* **2006**, *38*, 5–22.