[PD-05-031] PySAL and Spatial Statistics Libraries

Abstract

As spatial statistics are essential to the geographical inquiry, accessible and flexible software offering relevant functionalities is highly desired. Python Spatial Analysis Library (PySAL) represents an endeavor towards this end. It is an open-source python library and ecosystem hosting a wide array of spatial statistical and visualization methods. Since its first public release in 2010, PySAL has been applied to address various research questions, used as teaching materials for pedagogical purposes in regular classes and conference workshops serving a wide audience, and integrated into general GIS software such as ArcGIS and QGIS. This entry first gives an overview of the history and new development with PySAL. This is followed by a discussion of PySAL's new hierarchical structure, and two different modes of accessing PySAL's functionalities to perform various spatial statistical tasks, including exploratory spatial data analysis, spatial regression, and geovisualization. Next, a discussion is provided on how to find and utilize useful materials for studying and using spatial statistical functions from PySAL and how to get involved with the PySAL community as a user and prospective developer. The entry ends with a brief discussion of future development with PySAL.

Keywords: open science, open source, programming, PySAL, Python, spatial statistics

Author & citation

Kang, W. (2020). PySAL and Spatial Statistics Libraries. The Geographic Information Science & Technology Body of Knowledge (3rd Quarter 2020 Edition), John P. Wilson (ed.).
DOI: 10.22224/gistbok/2020.3.1..

Explanation

1. Definitions

Spatial statistics: spatial statistics is a branch of statistics that addresses spatial effects such as spatial dependence, spatial heterogeneity, and spatial scale, each of which could be potentially involved in spatial observations and impact statistical properties of classical methods.
PySAL: PySAL is the abbreviation for Python Spatial Analysis Library. It is an open source library and ecosystem which is written in Python and hosts a variety of classical and state-of-the-art spatial analytical functionalities.

Conda: conda is a popular package and environment management system that installs and manages packages from Anaconda Repository and Anaconda Cloud. It can effectively avoid the confusion brought about by installing multiple python interpreters (e.g. ArcGIS and QGIS have their own python interpreters) and help create a development environment/stack with specific versions of packages installed for accomplishing a specific analysis task or for testing an updated package to be released.
pip: pip is the standard python package management system. It is widely used to install packages from the Python Package Index (PyPI) which is the standard choice for python developers to distribute/release their packages.
Jupyter Notebook: Jupyter Notebook is an upgradation to IPython Notebook. It is an open-source web application supporting documents containing interactive code snippets, narratives, equations, and visualization.

2. Introduction

Spatial statistics is an important component of GIScience focusing on using statistical methods to obtain important information from geographical observations. For instance, for events with geographical locations, such as crime events, traffic accidents, and disease cases, the interest could lie in the spatial distribution of these events – whether they are geographically clustered, random, or dispersed. A series of clustering algorithms for point patterns could be used towards this end. On the other hand, we might want to know the temperature or air pollution level at all locations in a given country based on a limited number of observation stations, geostatistical methods such as kriging are usually used for such purpose. That is, to predict the values in other unobserved locations by exploiting the spatial dependence structure. Another vast set of spatial statistical methods is applied to areal data, such as US states, counties, or census tracts. Here, the interest lies in the aggregate values attached to these areas (e.g. population, per capita income) – whether large values tend to locate closer to each other, the so-called positive spatial autocorrelation. It should be noted that these application areas of these spatial statistical methods are not exclusive. Through spatial data processing, a spatial data set of a specific type could be converted into another type and thus the set of spatial statistical methods appropriate for the latter type can be applied to shed light on the observed patterns from a different point of view. For instance, point events can be aggregated to areal data by counting the occurrences within meaningful boundaries. Aside from exploring the spatial patterns of this single occurrence variable, a large amount of spatial statistical methods can be devoted to investigating the relationships between it and other variables of interest while accounting for spatial effects such as spatial dependence, and continuous or discrete spatial heterogeneity.

As spatial statistics are essential to the geographical inquiry, accessible and flexible software offering relevant functionalities is highly desired. Python Spatial Analysis Library (PySAL) represents an endeavor towards this end. It is an open-source python library and ecosystem hosting a wide array of spatial statistical and visualization methods. Since its first public release in 2010, PySAL has been applied to address various research questions, used as teaching materials for pedagogical purposes in regular classes and conference workshops serving a wide audience, and integrated into general GIS software such as ArcGIS and QGIS.

This entry first gives an overview of the history and new development with PySAL. This is followed by a discussion of PySAL's new hierarchical structure, and two different modes of accessing PySAL's functionalities to perform various spatial statistical tasks, including exploratory spatial data analysis, spatial regression, and geovisualization. Next, a discussion is provided on how to find and utilize useful materials for studying and using spatial statistical functions from PySAL and how to get involved with the PySAL community as a user and prospective developer. The entry ends with a brief discussion of future development with PySAL.

3. What is PySAL?

3.1 History of PySAL

PySAL was first formally released in July 2010, representing a joint effort of its co-founders' research teams - Sergio Rey and Luc Anselin, and followed a 6-month release cycle for the first six years (Rey, 2019). It was structured as an individual python package (**pysal**) comprised of multiple modules, each of which was targeted at specific spatial analytical tasks. These include computational geometry, clustering, exploratory spatial data analysis (ESDA), spatial dynamics, and spatial econometrics. Users interested in adopting PySAL for spatial analysis could use the standard python installer **pip** to install **pysal**. This was done by executing pip install pysal from a command line interface (CLI). An alternative was to adopt the popular package and environment management system **conda** and execute conda install pysal from a CLI. The latter approach has the advantage of installing **pysal** in an environment isolated from the default python environment and not incurring issues leading to broken environments. In addition to Python 2.7, pysal started to support Python 3.4+ in version 1.11 released in Jan 2016. For a more complete history of **pysal** releases, refer to its github release page.

Over the past two years, PySAL has undergone a refactoring process and shifted from 1.X series to 2.X series (Rey et al., 2020). Its API structure has changed drastically to reflect a more explicit modular structure and to reveal functionalities which are very useful but could be buried deeply in a giant package and hidden from users if continuing with the old structure. PySAL 2.X is not just a single open source python package, but more of an ecosystem where many smaller packages tackling specific spatial analytical tasks live. In other words, the original single package PySAL was divided organically into a bunch of new python packages. For instance, the ESDA module was converted into a single python package named **esda**. Users have the flexibility of installing the metapackage **pysal** which pulls together all the smaller packages (pip install pysal>=2.0.0 or conda install --channel conda-forge pysal from the conda-forge channel). They can also install and use those smaller packages independently if the interest lies in specific spatial analytics. For instance, pip install esda (or conda install --channel conda-forge esda) from the CLI will install the package esda which hosts a suite of classical and state-of-the-art statistics for ESDA (e.g. global and local spatial autocorrelation statistics). In PySAL 2.X, the PySAL ecosystem comprises many smaller python packages either originated from PySAL 1.X or recently developed, and thus incorporating more functionalities.

The first release of PySAL 2.X was **pysal** 2.0.0 in Jan 2019 following the legacy stable release of PySAL 1.X (**pysal** 1.14.4) in Jul 2018. In addition to the drastic API changes,

PySAL 2.X also shifts away from Python 2.7 and only supports Python 3.4+ due to maintenance complications of Python 2 and 3 at the same time and the sunsetting of Python 2 (officially sunset on January 1, 2020). PySAL users are recommended to shift to 2.X series as PySAL 1.X will not be maintained or enhanced. PySAL team has prepared a document detailing how to port the code to use PySAL 2.X https://github.com/pysal/pysal/blob/master/MIGRATING.md

3.2 PySAL Usage

PySAL has been used for fulfilling three main purposes since its first public release. These include using it (1) to accomplish spatial statistical analysis tasks in an interactive computational environment (e.g. Jupyter Notebook), (2) to serve as a source of spatial statistics in GIS software with a graphical user interface (GUI), (3) to be relied upon as a package dependency for another python package.

3.2.1 Accomplishing Spatial Analysis Tasks

The most common mode of using PySAL is to adopt it for analyzing cleaned spatial data in an interactive computational environment. Many published research papers have adopted PySAL for such purpose. The benefit of using PySAL together with other relevant open source python packages in Jupyter Notebook is that it is an important step towards Open Science. By providing a conda environment file listing the names and versions of respective python packages, together with multiple ordered Jupyter Notebooks with narratives and code snippets for data cleaning, processing, analysis, and visualization, the complete workflow of a research paper can be easily replicated and the results can be reproduced by other interested researchers. For instance, a recent paper (Reades, De Souza, and Hubbard 2019) adopted **pysal** as part of its spatial analysis workflow and provides a public GitHub repository hosting aforementioned files and data sets. Interested readers can freely download this repository, build the conda environment the authors originally used, replicate the analysis workflow, and reproduce results of the paper. More importantly, they can easily apply the workflow to another research question or data set or extend the current workflow.

3.2.2 Desktop Applications

For users who are not familiar with python programming, desktop applications with a user-friendly GUI with PySAL functionalities integrated are a more reasonable option. PySAL developers have been working with ArcGIS and QGIS to integrate some PySAL spatial statistical functionalities in these widely used GIS software products. In this manner, the highly accessible user interface and spatial data management and manipulation capabilities provided by the GIS software are combined with the-state-of-the-art spatial statistical functionalities provided by PySAL, benefiting a wider audience. PySAL has also been used as the source of spatial analytics for standalone desktop applications. For instance, GeoDaSpace was developed for spatial econometric analysis on top of PySAL's **spreg** module (PySAL 1.X), and STARS was developed for spatiotemporal analysis on top of PySAL's **spatial_dynamics module** (PySAL 1.X) (Rey and Janikas 2006).

3.2.3 Dependency for Other Python Packages

PySAL has also been relied upon as a dependency for other python packages as a source of spatial statistical methods. For instance, the famous package for geospatial data manipulation and visualization - **geopandas**, relies on PySAL's map classifiers for obtaining

various schemes for choropleth mapping.

A newly developed package targeting at the exploration, modeling, analysis, and visualization of neighborhood dynamics – **geosnap** - utilizes PySAL's spatially explicit regionalization algorithms for geodemographic analysis, and PySAL's spatiotemporal analytical methods for exploring dynamics of neighborhoods.

4. Spatial Statistical Analysis and Visualization with PySAL 2.X

4.1 Structure of PySAL 2.X

PySAL 2.X is an ecosystem of functionalities for spatial statistics and visualization, which is in strong contrast with its initial version – a single python package **pysal**. PySAL 2.X has a three-layer hierarchical structure as shown in Table 1. The bottom layer is the individual submodules built for specific spatial analysis or visualization tasks. For instance, **mgwr** hosts a suite of methods for calibration, diagnostics, and visualization of (multiscale) geographically weighted regression models.

These submodules are classified into 4 more general categories: lib, explore, model, and viz, all of which comprise the middle layer. lib contains submodule **libpysal**, which hosts methods for constructing spatial weights matrices of various types, the formalization of spatial relationships, and the essential component of most spatial statistical analysis. Both explore and model host computationally intensive techniques for statistical modeling and inference. explore is comprised of 7 submodules for ESDA. While **esda**, **giddy**, **inequality**, and **segregation** are targeted at areal data with different spatial statistical analysis focuses, **pointpats** focuses on planar point pattern statistical analysis, and **spaghetti** hosts statistical tests on networks. **region** has its origin in **clusterpy**, a python package hosting spatially constrained clustering (also known as regionalization) algorithms, which are applied to areal data. model contains 6 submodules for spatial modeling in a regression framework. Besides **mgwr** which hosts models for investigating spatially varying relationships, **spreg** is comprised of estimators and diagnostic tests for a variety of modern spatial econometric models explicitly accounting for potential spatial dependence and/or discrete spatial heterogeneity. **spint** focuses on gravity-type models for investigating interactions between places, and **spvcm** hosts methods for estimating spatial multilevel models which allow for between-group spatial correlation. The new addition **tobler** (since Feb 2020) provides functionalities of areal interpolation. The last but not the least in the middle layer is viz, which focuses on statistical methods supporting choropleth mapping (**mapclassify** hosts a suite of map classifiers), and light-weight functions for visualizing results of spatial statistics from explore and model (**splot**).

Table 1. Hierarchical Structure of PySAL 2.X

Top	Middle	Submodule / Package		
		Name	Description	Documentation Website

PySAL	lib	**libpysal**	spatial weights, computational geometry, and example data sets	https://pysal.org/libpysal/
	explore	**esda**	Exploratory spatial data analysis	https://pysal.org/esda/
		giddy	Geospatial distribution dynamics analysis	http://pysal.org/giddy/
		inequality	(Spatial) inequality analysis	https://inequality.readthedocs.io/
		segregation	(Spatial) segregation measures and inference	https://segregation.readthedocs.io/
		pointpats	Point pattern analysis	https://pointpats.readthedocs.io/
		spaghetti	Network-based spatial analysis	https://pysal.org/spaghetti/
		region	Spatially explicit regionalization algorithms	https://github.com/pysal/region
	model	**mgwr**	Multiscale geographically weighted regression	https://mgwr.readthedocs.io/
		spglm	Generalized linear modeling with sparse matrix	https://spglm.readthedocs.io/
		spint	Spatial interaction models	https://spint.readthedocs.io/
		spreg	Spatial regression models	https://spreg.readthedocs.io/
		spvcm	Spatial multilevel modeling	https://github.com/pysal/spvcm
		tobler	Aerial interpolation	https://pysal.org/tobler/
	viz	**mapclassify**	Classifiers for choropleth mapping	https://pysal.org/mapclassify/
		splot	Visualization for spatial statistical anslysis	https://splot.readthedocs.io/
Metapackage - **pysal**				https://pysal.org/pysal/

Figure 1 displays the dependency tree of the PySAL ecosystem where the PySAL metapackage and subpackages are highlighted in red. The metapackage **pysal** has all the subpackages as dependencies while several subpackages could be dependencies for other subpackages. For instance, **esda**, **giddy**, **segregation**, **pointpats**, **mgwr**, **libpysal**, **mgwr**, **spreg**, **spglm**, **spint**, **spvcm**, **tobler**, and **splot** rely on **libpysal** to construct spatial weight matrix or access example datasets.

pysal==2.2.0
- libpysal 4.2.2 [required: >=4.2.2]
  - beautifulsoup4 4.8.2 [required: any]
  - jinja2 2.11.1 [required: any]
  - numpy 1.18.1 [required: >=1.3]
  - pandas 1.0.1 [required: any]
  - python 3.7.6 [required: >=3.6]
  - requests 2.23.0 [required: any]
  - scipy 1.4.1 [required: >=0.11]
- esda 2.2.1 [required: >=2.2.1]
  - libpysal 4.2.2 [required: any]
  - numpy 1.18.1 [required: any]
  - python 3.7.6 [required: >3.4]
  - scikit-learn 0.22.1 [required: any]
  - scipy 1.4.1 [required: >=0.11]
- giddy 2.3.0 [required: >=2.3.0]
  - libpysal 4.2.2 [required: >=4.0.0]
  - esda 2.2.1 [required: >=2.0.0]
  - mapclassify 2.2.0 [required: >=2.0.0]
  - python 3.7.6 [required: >=3]
  - quantecon 0.4.5 [required: any]
  - scipy 1.4.1 [required: >=0.11]
- inequality 1.0.0 [required: >=1.0.0]
  - numpy 1.18.1 [required: >=1.3]
  - python 3.7.6 [required: >=3.5]
  - scipy 1.4.1 [required: >=0.11]
- segregation 1.2.0 [required: >=1.2.0]
  - geopandas 0.7.0 [required: any]
  - libpysal 4.2.2 [required: any]
  - matplotlib-base 3.1.3 [required: any]
  - numpy 1.18.1 [required: any]
  - pandas 1.0.1 [required: any]
  - python 3.7.6 [required: >=3]
  - scikit-learn 0.22.1 [required: any]
  - scipy 1.4.1 [required: any]
  - seaborn 0.10.0 [required: any]
  - tqdm 4.43.0 [required: any]
  - zstd 1.4.4 [required: any]
- pointpats 2.1.0 [required: >=2.1.0]
  - libpysal 4.2.2 [required: >=4.0.0]
  - matplotlib-base 3.1.3 [required: any]
  - numpy 1.18.1 [required: >=1.3]
  - pandas 1.0.1 [required: any]
  - python 3.7.6 [required: >=3.4]
  - scipy 1.4.1 [required: >=0.11]
- spaghetti 1.4.2.post1 [required: >=1.4.1]
  - esda 2.2.1 [required: any]
  - numpy 1.18.1 [required: >=1.3]
  - pandas 1.0.1 [required: any]
  - python 3.7.6 [required: >=3]
  - rtree 0.9.4 [required: any]
  - scipy 1.4.1 [required: >=0.11]
- mgwr 2.1.1 [required: >=2.1.1]
  - libpysal 4.2.2 [required: >=4.0.0]
  - numpy 1.18.1 [required: >=1.3]
  - python 3.7.6 [required: >=3.5]
  - scipy 1.4.1 [required: >=0.11]
  - spglm 1.0.7 [required: >=1.0.6]
  - spreg 1.0.4 [required: any]
- spglm 1.0.7 [required: >=1.0.7]
  - libpysal 4.2.2 [required: any]
  - numpy 1.18.1 [required: any]
  - python 3.7.6 [required: >=3.5]
  - scipy 1.4.1 [required: any]
  - spreg 1.0.4 [required: any]
- spint 1.0.6 [required: >=1.0.6]
  - libpysal 4.2.2 [required: >=4.0.0]
  - numpy 1.18.1 [required: >=1.3]
  - python 3.7.6 [required: >=3]
  - scipy 1.4.1 [required: >=0.11]
  - spglm 1.0.7 [required: >=1.0.7]
  - spreg 1.0.4 [required: >=1.0.4]
- spreg 1.0.4 [required: >=1.0.4]
  - libpysal 4.2.2 [required: any]
  - numpy 1.18.1 [required: any]
  - python 3.7.6 [required: >=3]
  - scipy 1.4.1 [required: any]
- spvcm 0.3.0 [required: >=0.3.0]
  - libpysal 4.2.2 [required: any]
  - numpy 1.18.1 [required: any]
  - pandas 1.0.1 [required: any]
  - python 3.7.6 [required: >=3.5]
  - scipy 1.4.1 [required: any]
  - seaborn 0.10.0 [required: any]
  - spreg 1.0.4 [required: any]
- tobler 0.2.0 [required: >=0.2.0]
  - geopandas 0.7.0 [required: any]
  - libpysal 4.2.2 [required: >=4.2.0]
  - numpy 1.18.1 [required: >=1.16]
  - pandas 1.0.1 [required: any]
  - python 3.7.6 [required: >=3.5]
  - python-dateutil 2.8.0 [required: <=2.8.0]
  - quilt3 3.1.10 [required: any]
  - rasterio 1.1.2 [required: any]
  - rasterstats 0.14.0 [required: any]
  - scipy 1.4.1 [required: any]
  - statsmodels 0.11.1 [required: any]
  - tqdm 4.43.0 [required: any]
- mapclassify 2.2.0 [required: >=2.2.0]
  - deprecated 1.2.7 [required: any]
  - numpy 1.18.1 [required: >=1.3]
  - pandas 1.0.1 [required: any]
  - python 3.7.6 [required: >=3.5]
  - scikit-learn 0.22.1 [required: any]
  - scipy 1.4.1 [required: >=0.11]
- splot 1.1.2 [required: >=1.1.2]
  - esda 2.2.1 [required: any]
  - geopandas 0.7.0 [required: >=0.4.0]
  - giddy 2.3.0 [required: any]
  - ipywidgets 7.5.1 [required: any]
  - libpysal 4.2.2 [required: any]
  - mapclassify 2.2.0 [required: any]
  - matplotlib-base 3.1.3 [required: any]
  - numpy 1.18.1 [required: any]
  - python 3.7.6 [required: >=3.5]
  - seaborn 0.10.0 [required: any]
  - spreg 1.0.4 [required: any]
- python 3.7.6 [required: >=3.7]
- python-dateutil 2.8.0 [required: <=2.8.0]
- urllib3 1.24.3 [required: <1.25]

Figure 1. Dependency Tree of PySAL 2.X. The dependence tree was built using the python package conda-tree (https://github.com/rvalieris/conda-tree). Adjustments are made to make the PySAL structure clearer. Source: author.

4.2 Two modes of accessing PySAL 2.X

There are two ways to access the functionalities in PySAL 2.X - either to install pysal as a metapackage or to install individual submodules, each of which has been packaged and released as a python package.

4.2.1 PySAL metapackage

The one-stop-shop solution for accessing all the available functionalities in the PySAL 2.X ecosystem is to directly install **pysal** with **pip** (pip install pysal) or **conda**. For the latter, the PySAL team recommends installing **pysal** and any of the subpackages from the conda-forge channel (conda install --channel condaforge pysal) as PySAL developers help maintain the packages in this channel and ensure they are up-todate. Comparing the installation options with **pip** and **conda**, the latter is recommended to avoid potentially broken environments.

Due to its explicit hierarchical structure, users can easily access any desired function. For instance, after properly installing **pysal** and importing it in the following manner from pysal.explore import esda; from esda import Moran gives users the functionality for the estimation and inference of Moran's I, the most widely used statistic for global spatial autocorrelation. For a complete list of the application programming interfaces (APIs), refer to pysal's documentation website. It should be noted that the metapackage **pysal** follows a 6-month release cycle. Its first release was version 2.0.0 in Jan 2019 supporting Python 3.5 and 3.6, the second was version 2.1.0 in July 2019 supporting Python 3.6 and 3.7, and the third, also the most recent release, was version 2.2.0 in Feb 2020 supporting Python 3.6 and 3.7. This implies that while installing pysal enables users to access all functionalities, they might not have access to the most updated ones.

4.2.2 PySAL submodules

Each of PySAL 2.X's 16 submodules has been packaged and released as an independent package; note that the number of submodules is expected to increase as more spatial statistical functionalities are added to PySAL 2.X. Currently, all of them support Python 3.6 and 3.7. These packages can be installed with **pip** or **conda** in the same fashion as **pysal**. One important difference is that they are not constrained by the 6-month release cycle and can be released at any time. Therefore, directly installing these smaller packages will potentially give users newer functionalities. For instance, on Dec 20, 2019 version 2.3.0 was released for the subpackage **giddy** which hosts functionalities of spatial analysis of longitudinal data. This release features an extension of (spatial) Markov Chains to deal with non-ergodic cases. Users would have access to this new feature if they install the newest version of **giddy** after Dec 20, 2019. However, for users choosing to install the metapackage **pysa**l, they would have to wait till Feb 2020 when a newer version of **pysal** was released to access this new feature.

5. Guidance for PySAL Users

There are many existing materials for learning PySAL, and several ways for prospective developers wishing to contribute to PySAL.

5.1 For General PySAL Users

The foremost material for users interested in using PySAL for research or development is the official website of PySAL (https://pysal.org/). It contains information ranging from the installation guide, PySAL developers, PySAL structure, news, community guidelines, and PySAL Jupyter Book. Jupyter Book is an online book that can be conveniently built from Jupyter Notebooks and markdown files. PySAL Jupyter Book is an important project facing interested users to give them an overview of a vast amount of spatial analytical functionalities PySAL provides.

Aside from the organization website, the metapackage as well as all the16 subpackages host up-to-date and consistent documentation websites; links to all documentation websites can be found at https://pysal.org/documentation. Every PySAL package also hosts a series of example Jupyter Notebooks demonstrating the usage of contained functions with embedded, executable, and interactive python code snippets, as well as useful and properly rendered Markdown narratives and math formulas. These Notebooks can be accessed from the GitHub repo of the respective package and are used as the source materials to build PySAL Jupyter Book.

Another important resource is the materials from PySAL workshops at international conferences. Users have free access to these materials (mostly Jupyter Notebooks) and can replicate spatial analysis with the Notebooks. Last but certainly not the least is the published books and papers about PySAL, such as “spreg: Modern Spatial Econometrics in Practice” (Anselin and Rey 2014), Geographic Data Science with PySAL and the PyData Stack, and a recent paper “The PySAL ecosystem: philosophy and implementation” (Rey et al., 2021) prepared by the current PySAL team.

5.2 Contributing to PySAL

Prospective PySAL developers have many channels to communicate with current PySAL developers. They can join the PySAL Gitter room to post instant messages or submit an issue on the GitHub Issues page for the metapackage pysal or individual subpackages. Students can also participate in Google Summer of Code (GSOC) to work on a proposed PySAL project with summer pay. PySAL has been supporting GSOC projects since 2016 - five projects have greatly enhanced PySAL while three 2020 GSOC projects dedicated to enhancing functionalities in **libpysal**, **esda**, and **spreg** are ongoing. For a list of potential 2020 PySAL GSOC projects, see this GitHub wiki page. Another way to get face-to-face interactions with current PySAL developers is to participate in PySAL sprints at major conferences such as AAG and SciPy. At the sprints, developers usually work together on specific PySAL related issues (e.g. fixing bugs, implementing new spatial statistics, refactoring existing functions, enhancing documentation).

6. Future Development

The new structure of PySAL enables more rapid and focused development and easier integration of other prospective spatial analysis packages. New package **access** (https://access.readthedocs.io/) hosting measures of spatial accessibility to services is expected to join PySAL soon. Another package for spatial optimization **spopt** (https://github.com/pysal/spopt) is under active development and will also be integrated into the PySAL ecosystem once it is ready. More functionalities such as Bayesian and panel spatial econometrics and discrete choice models for (multiscale) geographically weight regression models are expected to be implemented and integrated into PySAL.

References

Anselin, L. and Rey, S. J. (2014) Modern spatial econometrics in practice: A guide to GeoDa, GeoDaSpace and PySAL. GeoDa Press LLC.

Reades, J., De Souza, J., and Hubbard, P. (2019). Understanding Urban Gentrification through Machine Learning. Urban Studies. 56(5).

Rey, S. J. (2019). PySAL: the first 10 years. Spatial Economic Analysis 14(3): 273-282.

Rey, S. J. and Janikas, M. V. (2006). STARS: Space-Time Analysis of Regional Systems. Geographical Analysis. 38(1): 67-86.

Rey, S. J., Anselin, L., Amaral, P., Arribas-Bel, D., Cortes, R., Gaboardi, J., Kang, W., Knaap, E., Li, Z., Lumnitz, S., Oshan, T., Shao, H., and Wolf, L. (2021). The PySAL ecosystem: philosophy and implementation. Geographical Analysis 54(3): 467-487.